HUMAN DECISION-MAKING IN COMPUTER SECURITY INCIDENT RESPONSE

JONATHAN MICHAEL SPRING

PhD
Computer Science
Faculty of Engineering
University College London (UCL)

2019

Supervisors:
Prof. David Pym
Dr. Phyllis Illari
Dr. Emiliano De Cristofaro

Jonathan Michael Spring:
Human decision-making in computer security incident response
Creative Commons license BY-NC 4.0, 2019
version 1.1

DECLARATION

I, Jonathan Michael Spring, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

London, UK, 2019

Jonathan Michael Spring

DEDICATION

This work is dedicated to my parents, for their continued love and support.

ABSTRACT

BACKGROUND: Cybersecurity has risen to international importance. Almost every organization will fall victim to a successful cyberattack. Yet, guidance for computer security incident response analysts is inadequate.

RESEARCH QUESTIONS: What heuristics should an incident analyst use to construct general knowledge and analyse attacks? Can we construct formal tools to enable automated decision support for the analyst with such heuristics and knowledge?

METHOD: We take an interdisciplinary approach. To answer the first question, we use the research tradition of philosophy of science, specifically the study of mechanisms. To answer the question on formal tools, we use the research tradition of program verification and logic, specifically Separation Logic.

RESULTS: We identify several heuristics from biological sciences that cybersecurity researchers have re-invented to varying degrees. We consolidate the new mechanisms literature to yield heuristics related to the fact that knowledge is of clusters of multi-field mechanism schema on four dimensions. General knowledge structures such as the intrusion kill chain provide context and provide hypotheses for filling in details. The philosophical analysis answers this research question, and also provides constraints on building the logic. Finally, we succeed in defining an incident analysis logic resembling Separation Logic and translating the kill chain into it as a proof of concept.

CONCLUSION: These results benefits incident analysis, enabling it to expand from a tradecraft or art to also integrate science. Future research might realize our logic into automated decision-support. Additionally, we have opened the field of cybersecuity to collaboration with philosophers of science and logicians.

IMPACT STATEMENT

The potential impact of this thesis is both within and outside the academy.

Improvements to the practice of incident response will take some significant effort to transition, not least because the attention of such personnel is a scarce resource. The learning curve on scientific knowledge generation heuristics is steep. Nonetheless, I will provide two specific improvements that can impact incident response and Computer Network Defense (CND):

- robust knowledge creation and structured-observation design heuristics;
- a logical language to communicate reasoning steps without revealing sensitive data.

The thesis should enable further work within the academy. This thesis will, in some ways, chart out new fields of philosophy of cybersecurity in general and of computer security incident response in particular. Furthermore, this thesis will sensitize the fields of logic and philosophy of science to cybersecurity. The three research or development areas that appear most fruitful going forwards are: automated decision support for incident response; meta-logical analysis of the properties of the logic defined; and extension of the account of explanation and mechanism discovery both into cybersecurity and back into the philosophical literature on scientific explanation generally.

The third area for impact will be in how we train new incident analysts. There are at least three areas in which this impact might be realized. First, standards bodies could work to fill the gap Chapter 2 will identify in existing standards. Second, both Chapter 3 and Chapter 4 will identify likely challenges; future work on pedagogy could focus on how to best enable analysts to overcome them. Chapter 4, Chapter 5, and Chapter 7 will offer qualitative progress on some norms and expectations for analytic steps, but integrating these ideas into training and measuring results is future work.

In all three of these areas – practice, academics, and pedagogy – in order to bring about impact the work will need to be disseminated, integrated into education, and potentially engage public policy makers or business leaders about integrating these changes into the way computer-security incidents are handled.

CONTENTS

1 INTRODUCTION AND MOTIVATION 23
1.1 Summary of the argument 23
1.2 Impact 24
1.3 Important definitions 25
1.4 Publications and submissions adapted for thesis 26
1.5 Publications not adapted in thesis 27
1.6 Disclaimer 29

I BACKGROUND

2 LITERATURE REVIEW – INCIDENT RESPONSE 33
2.1 Introduction 33
2.2 Scope 35
2.2.1 Scope—Topic 36
2.2.2 Scope—Publication Venues 38
2.3 Related literature reviews 43
2.4 Methods 47
2.4.1 Search strategy 47
2.4.2 Appraisal strategy 48
2.4.3 Synthesis methods 49
2.4.4 Limitations 52
2.5 Results 54
2.5.1 IETF 55
2.5.2 ISO 56
2.5.3 FIRST 56
2.5.4 Intelligence community 59
2.5.5 Referenced Documents 60
2.6 Discussion 71
2.6.1 Comments and clarifications on Table 2.12 76
2.6.2 Note on case studies 83
2.7 Gaps and Work Plan 84
2.7.1 Research question 84
2.8 Related work 85
2.8.1 Interdisciplinary overlaps 86
2.8.2 Parts of information and computer science 89
2.9 Conclusion 92
3 LITERATURE REVIEW – SCIENCE AND SECURITY 95
3.1 Purported impediments to a science of security 96
3.2 Philosophy of science – historical background 97
3.3 Existing statements of science and security 100
3.3.1 Prepositions: ‘of’ or ‘for’ security 111
3.4 Practicing Science of Security 112
3.4.1 Scientific methods 114
3.4.2 Evaluating Results 120
3.4.3 The nature of scientific inquiry 126
3.4.4 Scientific Language(s) 130

3.4.5 Engineering or Science? 133
3.5 A Science of Security very much exists 135
3.6 Research plan 137

II GENERALISING, APPLYING, AND FORMALISING KNOWLEDGE

4 GENERAL KNOWLEDGE OF MECHANISMS 141
4.1 Introduction 142
4.2 Generality in philosophy of science 145
4.2.1 Turning away from laws 145
4.2.2 Generality and mechanisms 147
4.3 Building mechanistic knowledge in cybersecurity 152
4.3.1 The three challenges for cybersecurity 153
4.3.2 Three examples of cybersecurity mechanisms 158
4.4 Building general knowledge in cybersecurity 168
4.4.1 Constraining: On improving coordination in cybersecurity 173
4.5 Conclusion 176
5 THE INTRUSION KILL CHAIN AS A CASE STUDY 181
5.1 Introduction 181
5.2 Reasoning with the kill chain as a mechanism 184
5.2.1 Higher-level mechanisms 190
5.2.2 On lower-level mechanisms 192
5.3 Examples of Incident Analysis 194
5.3.1 Example 1: The Cuckoo's Egg 195
5.3.2 Example 2: Airport Security 200
5.4 Conclusions 204
6 SEPARATION LOGIC AS A CASE STUDY 207
6.1 Introduction to Separation Logic 207
6.2 Solving a Hard Problem 212
6.3 Why Separation Logic Works 217
6.3.1 The separating conjunction 219
6.3.2 Hoare triples 220
6.3.3 Frame Rule 221
6.3.4 Automatable abduction 222
6.4 The Semantics of Separation Logic 223
6.4.1 Bunched Logic 224
6.4.2 The Semantics of Bunched Logic 225
6.4.3 The Resource Semantics of Separation Logic 229
6.5 Deployable Proof Theory for Separation Logic 233
6.5.1 Separation Logic and the Frame Rule 234
6.5.2 Deployability via Contextual Refinement 235
6.5.3 Bi-abduction 236
6.6 Conclusion 243

III A LOGIC OF REASONING IN INCIDENT ANALYSIS

7 LOGIC DEFINITION 249
7.1 Philosophical and historical analyses 249

7.1.1 Summary of requirements 252
7.2 Logic design choices 253
7.3 Logic Definitions 255
7.3.1 Expressions 256
7.3.2 Basics and syntax 257
7.3.3 Model 259
7.3.4 Semantics 262
7.3.5 Abduction 264
7.3.6 On the metatheory of the security incident analysis logic 265
7.4 A worked example 266
7.4.1 A logic of the kill chain 270
7.4.2 Using more granular knowledge 273
7.4.3 Composition of attacks into a campaign 275
7.5 Benefits of this logic 279
7.6 Conclusions 280

IV CONSOLIDATION
8 SUMMARY 283
9 CONCLUSIONS 285

V BACKMATTER

BIBLIOGRAPHY 293

LIST OF FIGURES

Figure 3.1 The mathematical modeling cycle 120
Figure 4.1 GameOver Zeus botnet mechanism 159
Figure 4.2 Simple kill-chain diagram 163
Figure 5.1 Improved kill chain mechanism 185
Figure 5.2 Mechanistic translation of common language for computer security incidents 190
Figure 5.3 Mechanistic translation of a drive-by download 193
Figure 5.4 Model of accounting systems generated by Stoll during incident analysis 197
Figure 5.5 Representation of game-theoretic model as a mechanism 201
Figure 6.1 Anatomy of a pointer 214
Figure 6.2 Example pointer error – memory leaks 215
Figure 6.3 Pointers forming a linked list 230
Figure 6.4 Satisfaction relation for BI pointer logic 231

LIST OF TABLES

Table 2.1 Results of ACM CSUR search for extant literature reviews 45
Table 2.2 All documents found to be relevant through the search methodology 54
Table 2.3 IETF database search results 55
Table 2.4 ISO database search results 57
Table 2.5 Summary of results found through FIRST 58
Table 2.6 intelligence community search results 60
Table 2.7 Data formats cited by IETF standards 62
Table 2.8 ISO database search results 65
Table 2.9 Publications referenced by NIST SP800-61 67
Table 2.10 Documents referenced by CERT/CC sources 69
Table 2.11 Resources referenced by intelligence community documents 70
Table 2.12 Results: document categorization 75
Table 3.1 Common complains against science of security 97
Table 3.2 Summary of responses to the common complaints against a science of security 113

ACRONYMS

ACM	Association for Computing Machinery
ACoD	Art into Science: A Conference for Defense
AES	Advanced Encryption Standard
AirCERT	Automated Incident Reporting
APWG	Anti-Phishing Working Group
ARMOR	Assistant for Randomized Monitoring Over Routes
ARPA	Advanced Research Projects Agency, from 1972–1993 and since 1996 called DARPA
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge (by MITRE)
BCP	Best Current Practice, a series of documents published by IETF
BGP	Border Gateway Protocol
BI	logic of bunched implications
BIS	Department for Business, Innovation, and Skills (United Kingdom)
BLP	Bell-Lapadula, a model of access control
CAE	Center of Academic Excellence
CAIDA	Center for Applied Internet Data Analysis, based at University of California San Diego
CAPEC	Common Attack Pattern Enumeration and Classification (by MITRE)
CCIPS	Computer Crime and Intellectual Property Section of the US Department of Justice (DoJ)
CCE	Common Configuration Enumeration (by NIST)
CCSS	Common Configuration Scoring System (by NIST)
CEE	Common Event Expression (by MITRE)
CERT/CC	CERT® Coordination Center operated by Carnegie Mellon University
CIA	Central Intelligence Agency (US)

CIS	Center for Internet Security
CND	Computer Network Defense
CNO	Computer Network Operations
CPE	Common Platform Enumeration (by NIST)
CSIR	Computer Security Incident Response
CSIRT	Computer Security Incident Response Team
CTL	Concurrent Time Logic
CVE	Common Vulnerabilities and Exposures (by MITRE)
CVRF	Common Vulnerability Reporting Framework
CVSS	Common Vulnerability Scoring System, maintained by FIRST
CWE	Common Weakness Enumeration (by MITRE)
CWSS	Common Weakness Scoring System, maintained by MITRE
CybOX	Cyber Observable Expression, maintained by MITRE
DARPA	Defense Advanced Research Projects Agency
DHS	US Department of Homeland Security
DNS	Domain Name System
DoD	US Department of Defense
DoJ	US Department of Justice
ENISA	EU Agency for Network and Information Security
EPSRC	Engineering and Physical Sciences Research Council (United Kingdom)
EU	European Union
FAA	Federal Aviation Administration (US)
FBI	US Federal Bureau of Investigation
FDA	US Food and Drug Administration
FIRST	Forum of Incident Response and Security Teams
FISMA	Federal Information Security Management Act (US)
FS-ISAC	Financial Services Information Sharing and Analysis Center (ISAC)

GCHQ	Government Communications Headquarters (United Kingdom)
GFIRST	Government FIRST
HotSoS	Symposium on the Science of Security
HTTP	Hypertext Transfer Protocol, a standard by W3C
HTCIA	High Technology Crime Investigation Association
IC	intelligence community
ICT	information and communications technology
IEEE	Institute of Electrical and Electronic Engineers
IEP	Information Exchange Policy
IETF	Internet Engineering Task Force
IDS	intrusion detection system
IODEF	Incident Object Description Exchange Format
IODEF+	Incident Object Description Exchange Format Extensions (RFC 5901)
IDMEF	Intrusion Detection Message Exchange Format (RFC 4765)
ISAC	Information Sharing and Analysis Center
ISC	Internet Storm Centerpart of the privately-run SANS Institute
ISO	International Organization for Standardization
ISP	Internet Service Provider
ITU	International Telecommunications Union, an agency of the UN
LAX	Los Angeles International Airport
LBNL	Lawrence Berkeley National Laboratory
MAEC	Malware Attribute Enumeration and Characterization (by MITRE)
MITRE	the Mitre Corporation
MMDEF	Malware Metadata Exchange Format
NATO	North Atlantic Treaty Organization
NCA	National Crime Agency (UK)

NCCIC	US National Cybersecurity and Communications Integration Center
NDA	non-disclosure agreement
NIDPS	Network Intrusion Detection and Prevention System
NIST	National Institute of Standards and Technology, part of the US Department of Commerce
NSA	National Security Agency (US)
NSF	National Science Foundation (US)
OCIL	Open Checklist Interactive Language (by NIST)
OVAL	Open Vulnerability and Assessment Language (by MITRE)
OWASP	Open Web Application Security Project
OWL	Ontology Web Language
pDNS	passive Domain Name System (DNS) traffic analysis
RAM	Random Access Memory
RCT	Randomized Controlled Trial
REN-ISAC	Research and Education Networking Information Sharing and Analysis Center (ISAC)
RID	Real-time Inter-network Defense
RISCS	Research Institute in Science of Cyber Security (United Kingdom)
SANS Institute	Sysadmin, Audit, Network, and Security Institute
SCAP	Security Content Automation Protocol (by NIST)
SiLK	System for Internet-level Knowledge, an open-source analysis tool set published by CERT® Coordination Center (CERT/CC)
SoK	Systematization of Knowledge paper in IEEE Oakland conference
STIX	Structured Threat Information Expression (by MITRE)
TAXII	Trusted Automated eXchange of Indicator Information (by MITRE)
TCP/IP	Transmission Control Protocol / Internet Protocol
TLA	Temporal Logic of Actions
TLP	Traffic Light Protocol

TSA	Transport Security Administration (US)
TTPs	Tools, tactics, and procedures
UN	United Nations
UML	Unified Modeling Language, see Larman (2004)
US	United States of America
US-CERT	US Computer Emergency Readiness Team, a branch of NCCIC within DHS
URL	Uniform Resource Locator
VERIS	Vocabulary for Event Recording and Incident Sharing
W3C	World Wide Web Consortium
XCCDF	Extensible Configuration Checklist Description Format (by NIST)
XML	Extensible Markup Language, a standard by W3C

1

INTRODUCTION AND MOTIVATION

This thesis is about how we can be better at cybersecurity.[1] The thesis approaches improving cybersecurity by enriching the cognitive tools we have to reason about problems in the domain. Specifically, via the central problem of incident analysis, a human-centric task within the computer security incident management process. The ultimate tangible output of the thesis is a logic within which incident analysis can be expressed and refined. The qualitative reasoning tools developed along the way that guide the development of the logic are also valuable and more quickly applicable.

1.1 SUMMARY OF THE ARGUMENT

Based on the literature review in Chapter 2, the research question I will address in this thesis will be: How to satisfactorily make clear and explicit the reasoning process used by individual Computer Security Incident Response (CSIR) analysts while they pursue technical details?

This breaks down into three component questions:

- What heuristics should an incident analyst use to construct general knowledge and analyse attacks?
- Is it possible to construct formal tools that enable automated decision support for the analyst with such heuristics and knowledge?
- How can an incident analyst apply general knowledge to improve analysis of specific incidents?

1 Or, if you prefer a different term, information security, or computer security, or IT security, or ICT security.

My strategy to answer these questions is to work as much by analogy to existing fields of study as possible, but where necessary build accounts of general knowledge and formal tools where needed. First, I will argue that CSIR functions like a scientific endeavour. Therefore, in many more cases than is currently appreciated, the heurstics of the sciences, and particularly life sciences, can meet gaps in CSIR analysis.

Building on this claim that CSIR is like a science, I will claim that CSIR analysts (should) build general knowledge as other sciences (should). One good model is knowledge of mechanisms. To demonstrate how such models apply to CSIR, I will provide several examples of mechanisms in CSIR at various levels. An important recurring example across chapters will be the intrusion kill chain model of cyberattacks. Since cyberattacks are a central phenomenon within CSIR, discovering (a model of) an attack, formalizing a model of an attack, and communicating general knowledge about the structure of attacks are key CSIR analysis functions. The recurring example of the kill chain helps ensure my (mental and formal) tools support such analysis.

Finally, I claim that CSIR analysts should reason (about mechanisms) explicitly. In order to locate an adequate formal representation of reasoning in CSIR, I will perform a case study on Separation Logic, a logic for program verification in which both reasoning and abduction are explicit and scalable. One important result of the case study is that logics should be built to match analytic needs in order to be useful. To this end, since there is no extant formal tool for reasoning in CSIR, it will be necessary to construct one. I will take specific care to include temporal and spatial operators to permit modes of reasoning that the philosophical analysis of CSIR identifies as valuable.

1.2 IMPACT

Although any security practitioner or researcher is capable of learning to look at their problems in the light of scientific model discovery,

there is no turn-key or push-button solution to change one's thinking habits. One benefit of the abstraction level this thesis takes is the wide applicability of the results. The trade-off is that each practitioner or researcher who wishes to do so cannot do so casually. Through future work, the logic defined in Chapter 7 may eventually amplify efforts via some automation.

The impact of moving towards a more scientific incident analysis is an analysis process better grounded in evidence and more likely to produce reliable insights from collected evidence. The impact of better incident analysis is likely better incident response, in turn likely leading to better remediation of incidents and better lessons learned. Better lessons learned will eventually propagate to better preparation for, and prevention of, security incidents. The impact of more scientific incident analysis also should make scientific research in security better connected to security practice, thereby also improving the relevance of research practice.

1.3 IMPORTANT DEFINITIONS

Although I will focus on a specific aspect of cybersecurity, namely Computer Security Incident Response (CSIR), I feel it is important to explicitly define cybersecurity. I had preferred the better-defined 'information security,' but for this thesis I will generally discuss cybersecurity as the practice to which CSIR contributes. Shirey (2007) defines information security as "measures that implement and assure security services in information systems, including in computer systems and in communication systems" where 'information systems' mean "an organized assembly of computing and communication resources and procedures – i.e., equipment and services, together with their supporting infrastructure, facilities, and personnel – that create, collect, record, process, store, transport, retrieve, display, disseminate, control, or dispose of information to accomplish a specified set of functions." Thus

defined, information security is quite expansive. Even so, I will define cybersecurity as strictly a superset of information security. The additional item I take to be included in cybersecurity is the security of the shared social space created by the various human agents who manipulate information enabled by these information systems ('The origins of cyberspace'). This definition is consistent with, for example, the US military's definition of cyberspace as a domain of warfare; however, in the military usage the purpose of the shared social space is to conduct warfare. In my usage, *cybersecurity* is: measures to implement and assure security services for the full range of human social uses of shared (cyber)space, in addition to security services for information systems.

The remainder of this chapter describes my publications both before and during my time at UCL. Several chapters of the thesis have already been published, and the remaining material chapters are under submission. The various publications that are not incorporated into the thesis form a body of background experience that has informed my problem specification and approach, but are not directly in support of a more scientific approach to incident analysis.

1.4 PUBLICATIONS AND SUBMISSIONS ADAPTED FOR THESIS

The chapters in the thesis are largely derived from material that has been either published or submitted for publication. Specifically:

- Chapter 2 is an expanded version of a paper submitted to ACM Computing Surveys in April 2018.[2]
- Chapter 3 is derived from a paper published at NSPW 2017.[3]
- Chapter 4 is derived from a paper in Philsophy & Technology.[4]

2 Jonathan M Spring and Phyllis Illari (Apr. 2018b). 'Review of Human Decision-making during Incident Analysis'. In: *arXiv preprint* 1903.10080.

3 Jonathan M Spring et al. (2nd Oct. 2017). 'Practicing a Science of Security: A philosophy of science perspective'. In: *New Security Paradigms Workshop*. Santa Cruz, CA, USA.

4 Jonathan M Spring and Phyllis Illari (2018a). 'Building General Knowledge of Mechanisms in Information Security'. In: *Philosophy & Technology*. DOI: 10.1007/s13347-018-0329-z.

- Chapter 5 is a reworking of a paper published in the Journal of Cybersecurity.[5]
- Chapter 6 is derived from a paper in Philosophy & Technology.[6]
- Chapter 7 is derived from a paper published and presented at the Conference on Decision and Game Theory for Security, 2018.[7]
- The concluding Chapters 8 and 9 are new for this thesis.

1.5 PUBLICATIONS NOT ADAPTED IN THESIS

I have published work both during my time at UCL and prior to joining UCL that are relevant to this thesis but from which no text is adapted. There are five additional publications from my time at UCL. Three publications not represented in the thesis are applications of the philosophical work presented in the thesis to usable security research methods,[8] usable security studies,[9] and network security measurement.[10] The other two publications are further philosophical developments, on refining the philsophical tools for use in security[11] and malicious software classification.[12]

Publications prior to my time at UCL focus mostly on my experience of incident analysis and threat intelligence. In both of these areas, the publications alternate between technical contributions and synthesizing general knowledge from past work. The work that unexpectedly began

5 Jonathan M Spring and Eric Hatleback (Jan. 2017). 'Thinking about intrusion kill chains as mechanisms'. In: *Journal of Cybersecurity* 3.3, pp. 185–197.

6 David Pym et al. (2018). 'Why separation logic works'. In: *Philosophy & Technology*. DOI: 10.1007/s13347-018-0312-8.

7 Jonathan M Spring and David Pym (31st Oct. 2018). 'Towards Scientific Incident Response'. In: *GameSec*. LNCS 11199. Seattle, WA: Springer.

8 Kat Krol et al. (26th May 2016). 'Towards robust experimental design for user studies in security and privacy'. In: *Learning from Authoritative Security Experiment Results (LASER)*. IEEE. San Jose, CA, pp. 21–31.

9 Albese Demjaha et al. (18th Feb. 2018). 'Metaphors considered harmful? An exploratory study of the effectiveness of functional metaphors for end-to-end encryption'. In: *Workshop on Usable Security (USEC)*. San Diego, CA: ISOC.

10 Leigh B Metcalf et al. (18th Oct. 2017). 'Open-source measurement of fast-flux networks while considering domain-name parking'. In: *Learning from Authoritative Security Experiment Results (LASER)*. USENIX. Arlington, VA, USA, pp. 13–24.

11 Eric Hatleback and Jonathan M Spring (2018). 'A Refinement to the General Mechanistic Account'. In: *European Journal for Philosophy of Science*.

12 Giuseppe Primiero et al. (2018). 'On Malfunction, Mechanisms, and Malware Classification'. In: *Philosophy & Technology* Online First.

this philosophical thread was a deceptively simple question: what blacklist should I use? This question led to a complicated study to construct a baseline from which blacklists could be compared.[13] These surprising results were eventually corroborated by further work[14] as well as by other studies within the security community (Kührer et al., 2014; Trost, 2014; Verizon, 2015). My passive DNS analysis expertise led to collaborations at major conferences, on typosquatting[15] at USENIX and on the internet of things, back when it was just "customer premise equipment", at Black Hat.[16]

Three publications cover analysis methods, and another is tightly allied with analysis and tools. The first discussed indicator expansion,[17] a method for noting how network touchpoints are related by different data sources. A technical report on CND strategy discussed when indicators should be used for defence – thus revealing to adversaries defender knowledge – and when such equities should be withheld.[18] Another conference paper provided an analysis of the advancement of adversary capability, with the goal of supporting long-term defence capability planning that can account for future, not just current, adversary capability.[19] With several of the System for Internet-level Knowledge (SiLK) engineers, I captured the analysis and design choices that went into the design of the tools and why they are optimized for certain kinds of analytic tasks.[20]

13 Leigh B Metcalf and Jonathan M Spring (Sept. 2013). *Everything you wanted to know about blacklists but were afraid to ask*. Tech. rep. CERTCC-2013-39. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

14 Leigh B Metcalf and Jonathan M Spring (12th Oct. 2015). 'Blacklist Ecosystem Analysis: Spanning Jan 2012 to Jun 2014'. In: *The 2nd ACM Workshop on Information Sharing and Collaborative Security*. Denver, pp. 13–22.

15 Janos Szurdi et al. (Aug. 2014). 'The long "taile" of typosquatting domain names'. In: *23rd USENIX Security Symposium*. USENIX Association. San Diego, pp. 191–206.

16 Chris Hallenbeck et al. (7th Aug. 2014). 'Abuse of Customer Premise Equipment and Recommended Actions'. In: *Black Hat USA 2014*. Las Vegas, Nevada: UBM.

17 Jonathan M Spring (Sept. 2013a). 'A notation for describing the steps in indicator expansion'. In: *eCrime Researchers Summit (eCRS), 2013*. IEEE. San Francisco.

18 Jonathan M Spring and Edward Stoner (July 2015). *CND Equities Strategy*. Tech. rep. CERTCC-2015-40. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

19 Jonathan M Spring et al. (27th May 2015). 'Global adversarial capability modeling'. In: *APWG Symposium on Electronic Crime Research (eCrime)*. IEEE. Barcelona.

20 Mark Thomas et al. (1st July 2014). 'SiLK: A tool suite for unsampled network flow analysis at scale'. In: *IEEE BigData Congress*. Anchorage, pp. 184–191.

Prior publications also include some wider-ranging topics. I co-authored a textbook targeted at policy and management graduate students, to provide them with a strategic introduction to information security.[21] I took a short diversion into systems security work and wrote a well-cited analysis of cloud security.[22] Further, the initial philosophical contribution arguing for a mechanistic understanding of experimentation and generating knowledge within computer science was published in 2014.[23]

1.6 DISCLAIMER

I have come to this topic because of my prior work at the CERT® Coordination Center at Carnegie Mellon University's Software Engineering Institute. However, the SEI and Carnegie Mellon are unaffiliated with this thesis and none of its content necessarily represents the views of Carnegie Mellon or the SEI, or any other organization with which they are affiliated or that I have past affiliations with. All opinions expressed herein are my own (and perhaps those of my wonderful co-authors, where collaborations are noted).

21 Timothy Shimeall and Jonathan M Spring (Jan. 2014). *Introduction to Information Security: A Strategic-based Approach*. Waltham, MA: Elsevier.

22 Jonathan M Spring (2011a). 'Monitoring cloud computing by layer, part 1'. In: *Security & Privacy* 9.2, pp. 66–68; Jonathan M Spring (2011b). 'Monitoring cloud computing by layer, part 2'. In: *Security & Privacy* 9.3, pp. 52–55.

23 Eric Hatleback and Jonathan M Spring (2014). 'Exploring a mechanistic approach to experimentation in computing'. In: *Philosophy & Technology* 27.3, pp. 441–459.

Part I

BACKGROUND

This part is composed of two chapters. The first is an in-depth literature review on the main topic – human decision-making during Computer Security Incident Response (CSIR). Chapter 2 serves to lay out the scope of the thesis in detail by specifying what falls under human decision-making during CSIR and identifying the challenges the remaining chapters will work to solve. The second chapter in this part, Chapter 3, introduces in quite brief and broad strokes the field philosophy of science as well as reviewing the literature on a science of security. I take science of security to be the cybersecurity community's internal attempt to apply philosophy of science. I will explain why this science of security approach – and its focus on laws – has been misguided, and provide some context for the modern new-mechanist approach to generalised knowledge and knowledge generation which I will apply in the next part.

2

LITERATURE REVIEW – INCIDENT RESPONSE[1]

This literature review identifies the structure of human decision-making during Computer Security Incident Response (CSIR) (shortened as incident response if the usage is unambiguous) and the existing research programs that underpin the practical decision-making structure. The utility of such a literature review is to identify gaps in guidance for CSIR decision making that would benefit from further research.

Incident response is an attractive topic because it anchors the whole field of cybersecurity and information security. When information security is not responding to an incident, it is either preparing for one or learning from past incident responses.[2] Preparation and learning are each complex and independent activities. However, their crux is incident response. I restrict the focus to human-centric decision making as opposed to automated or machine-learning techniques. Such automation is clearly crucial to incident response. However, deciding what automation to build and use remains a human decision, as does how to interpret and act on results.

2.1 INTRODUCTION

Cybersecurity is important, yet breaches are usually detected months after they occur (Verizon, 2015). Professionals suggest judging whether an organization has a good security posture by their incident response capabilities (Shimeall and Spring, 2014, ch. 15).[3] The question addressed by this review is: *how do incident responders make decisions during an investigation?*

Successful guidance on this question improves incident response, and thereby all of cybersecurity. However, the question is deceptively simple, and

1 This chapter is based on the joint paper: Jonathan M Spring and Phyllis Illari (Apr. 2018b). 'Review of Human Decision-making during Incident Analysis'. In: *arXiv preprint* 1903.10080.

2 Prevention of future incidents may fruitfully be discussed as independent from incident response, but in practice prevention techniques are almost all adapted from the lessons learned after responding to an incident.

3 See also: "The question you must accept is not whether security incidents will occur, but rather how quickly they can be identified and resolved." https://www.gartner.com/smarterwithgartner/prepare-for-the-inevitable-security-incident/

the use of 'how' is ambiguous. More specifically, the goal is a satisfactory process by which incident responders generate and prioritize questions to ask, a satisfactory process of answering these questions under time constraints, and a satisfactory process for how these answers lead to decisions. In summary, there is ample advice on what questions to ask and how to answer a given specific question, but little on how to prioritize questions or interpret results. This is a significant lacuna in the existing literature on decision-making and evidence evaluation during incident response. I will propose possible directions for filling this gap using philosophy of science and mathematical modelling.

The immediate challenge of this literature review is how to determine an appropriate yet manageable scope. The scope expands for at least two reasons: the topic is of broad application with many sub-parts, and the academic literature is not the only relevant source. Practitioners are a necessary source if the review is to adequately capture the state of the art. Besides simply increasing the volume of publications to review, cybersecurity practitioners do not publish with the same reliability or norms as is customary for input to an academic literature review. This variability makes it difficult to evaluate contributions on a single appraisal strategy.

I will approach this problem in two parts. The first part covers scope and problem definition. Section 2.2.1 defines the term Computer Security Incident Response (CSIR), and which aspects are in scope. Section 2.2.2 defines the scope in terms of publication venues included in the review. Section 2.3 provides evidence that no literature review with similar scope and goals has been published. Section 2.4 defines our method for reviewing the state of decision-making recommendations during computer-security incident response.

The second part covers results and proposed research. Section 2.5 provides the results of the incident response decision-making literature review. The first four subparts cover each of the four venues in scope, and Section 2.5.5 reports the evaluation of documents cited by those documents from the search results that are evaluated to be relevant. Section 2.6 critically analyses the results of the reviews. Section 2.7 identifies the gaps in the literature and proposes a research question to make substantive progress on the identified gaps. Section 2.8 examines work from related disciplines.

2.2 SCOPE

This section limits the scope of the literature review in two distinct aspects: the definition of the topic and the publication venues. I will restrict the definition of CSIR to three subtasks during investigation: evidence collection, analysis, and reporting. I will restrict publication venues to relevant international standards and academic literature that is referenced therein. Specifically, the search venues will be the Internet Engineering Task Force (IETF), the International Organization for Standardization (ISO),[4] Forum of Incident Response and Security Teams (FIRST), and documents understood to represent the US intelligence community (IC).

As far as possible, I will use standard definitions for terms, preferring global, consensus, freely-available definitions. Explicitly, in order of preference, the IETF, ISO, and FIRST. This ordering is based on the extent of consensus (the IETF membership is broader than FIRST) and the openness of the definitions. Otherwise, the choice of established definitions for jargon is primarily for clarity, and to compress the discussion; I assume familiarity with the terms in the IETF Internet Security Glossary (Shirey, 2007).

The scope is not the traditional academic venues. The operational reason is to focus on what incident responders actually do. Ideally this would take the form of first-hand accounts; however, cybersecurity is a sensitive topic. Chatham House Rules[5] and non-disclosure agreements (NDAs) abound in the discipline; these norms within the security community further frustrate any usual academic publication expectations. It would be impossible to evaluate the selection bias or outright deception within studies. The incident response standards at least form an honest baseline of what is expected of a competent practitioner. The assumption is that this review applies to competent practitioners, where competent is defined by consensus among practitioners and codified in the standards. I do not empirically address the extent to which competence is common. Due to the community norms of secrecy, norms documented by Sundaramurthy et al. (2014), a comprehensive evaluation seems impractical.

4 Although ISO standards are only available for a fee, the terms and definitions as used in the relevant standards (the 27000 series) are freely available.

5 Chatham House Rules indicates a situation in which the information or content of a meeting, discussion, or presentation may be disclosed but the source of the information may not be identified, implicitly or explicitly. This request is made by the speaker prior to disclosing the information.

Section 2.3 provides evidence that the academic literature does not systematically cover my scope. The method to justify this is a search through two common sources of literature reviews, Association for Computing Machinery (ACM) Computing Surveys and the Institute of Electrical and Electronic Engineers (IEEE) Security and Privacy "systematization of knowledge" papers. In summary, no relevant surveys have been published on human decision-making during CSIR analysis. While the reason why is unclear, the task is not made easier by the natural secrecy of practitioners.

A further consideration in focusing on standards as the starting point is simply to prevent an explosion of documents to review. A cursory Google Scholar search for "computer security incident response" and "digital forensic investigation" each return about 2,000 results. Alternatively, searches in the ACM Guide to Computing Literature and IEEE Xplore databases for "computer security incident response" each return 20-25 results (both on Sep 1, 2017). Many of these results are obscure, and for those that are not it is challenging to evaluate their operational impact. Going straight to the standards bypasses this question of impact – the remit and authority of standards is explicit.

Finally, I draw on six years of personal experience working at a research center where I was able to interact with a variety of practicing incident responders.

2.2.1 *Scope—Topic*

The first task is to situate computer-security incident response within a context of what it excludes and what falls under it. Incident response is a subspecies of business continuity planning or continuity of operations. Continuity planning may be a response to man-made or natural events, and either physical or digital events. A military invasion is a man-made, physical event; a hurricane is a natural, physical event. Computer-security incident response is only in response to security incidents that are primarily digital, where a security incident is something "contrary to system policy" (Shirey, 2007). Thus, accidents of all kinds are out of scope; though distinguishing apparent accidents from malicious acts is included. Intentional physical destruction of computing resources is also out of scope of computer-security incident response (Brownlee and Guttman, 1998).

Narrowing the focus further, incident response is a task within incident management.[6] CERT/CC definitions of incident management (Alberts et al., 2004; Mundie et al., 2014) locate incident response as independent from activities such as preparation, improving defences, training, financing, and lessons learnt. Mundie et al. (2014) surveys practices including those by CERT/CC and ISO; six tasks are included as part of incident response: monitoring, detection, evidence collection, analysis, reporting, and recovery. These six tasks form the core topic of this survey of incident response.

The human-centric decisions that are elements of these six incident response tasks vary in importance. Analysis, reporting, and recovery are almost wholly human-driven. Monitoring is almost wholly automated, while detection and evidence collection are a mixture. Where detection is automated, say in an intrusion detection system (IDS),[7] it is out of scope. Decisions about what detection rules to implement in an IDS are part of the preparation or improving defences phases of incident management, as a result of lessons learnt, and thus are also out of scope. Actual human intrusion detection is rare, and when it occurs is usually the result of analysis during incident response to some other, automatically-detected incident. Therefore, the focus on human-driven incident response investigation excludes monitoring and detection.

The IETF (Brownlee and Guttman, 1998; Shirey, 2007) and CERT/CC define neither "investigation" nor "forensics" in relation to the incident management process. ISO/IEC (2015c) places investigation as the centrepiece of incident management, where the principles of incident management are to "give guidance on the investigation of, and preparation to investigate, information security incidents" ISO/IEC (2016, §0). In this way, ISO uses "investigation" as a near-synonym to "response" in the IETF and FIRST literature.

6 The term "incident management" does not appear in IETF documents consistently. Trammell (2012a) describes Incident Object Description Exchange Format (IODEF) (Danyliw et al., 2007) as a protocol for "exchange of incident management data," but the term "incident management" does not appear again in Trammell (2012a), and not even once in Danyliw et al. (2007). ISO/IEC (2016) defines "information security incident management" as "exercise of a consistent and effective approach to the handling of information security incidents." FIRST does not provide a definition itself, but FIRST (2017) recommends the CERT/CC documentation on incident management. Trammell and Danyliw both worked at CERT/CC, so this is probably the source of the informal reference in the IETF documents. The CERT/CC phases are consistent with the ISO/IEC (2016) phases of plan and prepare; detection and reporting; assessment and decision; responses; and lessons learnt. I prefer the CERT/CC definitions as they are public (vice the ISO standards), and recommended by FIRST (thus in scope of using global, consensus-driven definitions).

7 Shirey (2007) refers to Bace and Mell (2001) for the definition and guidelines for IDS, which has been superseded by Scarfone and Mell (2007).

The use of 'incident' emphasizes that the investigation is oriented towards the violation of some policy, possibly but not necessarily a law. Thus, modelling or analysing online crime is an investigation, and so is IT staff looking into a usage policy violation. Incident response or investigation is entwined with cybersecurity because one essential aspect of a defence strategy is feedback from investigation to 'preparation' and 'protection' (Alberts et al., 2004). However, detailed discussion of preparation and protection is placed out of scope.

Incident response, per IETF and FIRST, explicitly includes remediation. ISO (ISO/IEC, 2015c) treats remediation and response as separate from investigation. In determining scope, I follow ISO and exclude recovery. Note that both sets of standards agree that clear reporting is the proper output of incident analysis, and any recovery follows reporting. However, it does seem clear that recovery follows a different decision process than analysis, and the two should be treated separately. Within the six tasks identified within incident response, three are left in scope:

- evidence collection
- analysis
- reporting

These three seem too tightly coupled to separate, and are described consistently across the international standards organizations.

For each of these three topics, the concern is primarily with how an individual analyst makes decisions during these three phases. What tool or what language the analyst or investigator uses to make these choices is not germane and is out of scope. This is not a review of available security tools, platforms, or data exchange formats. The goal is to survey how analysts enumerate options, evaluate choices, generalize results, and justify these steps.

2.2.2 *Scope—Publication Venues*

Incident response and investigation includes professional and business aspects; therefore the scope of viable sources incident response practices cannot justifiably be limited to academic sources. As Chapter 3 will document, the science of security is an unsettled area of research rather than an area with anything like standards. In fact, traditional academic publication venues contain little if anything about day-to-day incident response practices; academics do not

do incident response themselves. Sundaramurthy et al. (2014) seems to mark the first anthropological study of a Computer Security Incident Response Team (CSIRT)[8] members and their attitudes, but this literature is not about the actual process of incident response; that is covered in the professional literature.

Therefore, to understand current incident response practices the scope of the review is internationally-relevant standards and whatever literature is referenced therein. The history of standards as its own industry is complex in its own right (Spring, 2011c). The Internet and IT standards are formed by heterogeneous processes by a wide variety of actors (Oksala et al., 1996). Security-relevant standards are beginning to be seen as having their own unique requirements, distinct from IT standards generally (Kuhlmann et al., 2016). However, it is a separate project to analyse how CSIR standards have come to be. The standards in this review are taken as-is, with the understanding that any interpretations should be made cautiously because the standards may not cleanly fit in to existing studies of how and why other IT standards are created. More than other IT standards, CSIR standards are likely a codification of tacit practitioner knowledge (Nightingale, 2009).

The scope of publication venues is limited to ISO, IETF, FIRST, and the US intelligence community (IC). This choice is based on what organizations are relevant in influencing or describing international incident response practices, which is in turn based in the history of the governance of the Internet. I mitigate potential over-restriction of focus by including any documents cited by standards publications. The reasoning for selecting these organizations specifically is as follows.

ISO and the International Telecommunications Union (ITU) are the authoritative technology standards makers (Oksala et al., 1996, p. 11). The US federal government plays a dominant role in Internet development and standards, through the original Advanced Research Projects Agency (ARPA) development under the US Department of Defense (DoD) and subsequent stewardship under the Department of Commerce.[9]

ISO is *de dicto* where one looks for international standards. Each nation-state is allowed to have one member is ISO, namely the official national standards body representing all the industry-based bodies in each country. It is a

8 CSIRT is the general term, and will be used unless referring to a specific organization.

9 Two important sub-parts of Commerce are Internet governance by the National Telecommunications and Information Administration and standards by National Institute of Standards and Technology (NIST).

federation of federations, representing a multitude of industries. ISO standardizes things like the two-letter country codes (which have been adopted as DNS top-level domains), paper sizes, and credit cards. The ITU and their CIRT program[10] seems promising in name; however, their website publishes little besides an events list. It appears that content is provided by FIRST members, companies, or other consultancies; the ITU does not produce its own incident response materials or standards. This leaves only ISO in scope of the potential authoritative international standards bodies.

On the other hand, the IETF is the *de facto* place to go for international Internet standards because, for all intents and purposes, its standards are the Internet. The IETF "doesn't recognize kings—only running code" and creates more pragmatic, open (freely-available) standards (Oksala et al., 1996, p. 12). Open standards happen to have won out on the Internet; IETF standards like Transmission Control Protocol / Internet Protocol (TCP/IP), DNS, and Border Gateway Protocol (BGP) underpin every Internet connection. For a background history of how the precursor to the IETF came to this dominant role, see Hafner and Lyon (1998). The other main open-standards body is the World Wide Web Consortium (W3C), which standardizes Hypertext Transfer Protocol (HTTP) and Extensible Markup Language (XML), for example. W3C stays narrowly focused on web standards, and although this includes important web security considerations, W3C does not work on incident management, so I mark the group as out of scope.

FIRST is not part of this longer information and communications technology (ICT) standards history. It was formed in 1990 specifically to coordinate among and represent the interests of CSIRTs globally. FIRST's mission includes developing and sharing best practices, as well as creating and expanding incident response teams (FIRST, 2003). FIRST is the one and only global organization representing those who do human-centric incident response tasks. FIRST's work with United Nations (UN) agencies like the ITU also testifies to its global influence. It is naturally included as in-scope.

There are three organizations one might consider naturally in scope that are excluded. These are EU Agency for Network and Information Security (ENISA), NIST, and the US DoD. However, within the gray area between NIST and the US intelligence community, I identify a fourth set of *de facto* standards.

ENISA is specifically focused on CSIRTs and information security. The European Union (EU) makes available an independent evaluation of ENISA's

10 http://www.itu.int/en/ITU-D/Cybersecurity/Pages/Organizational-Structures.aspx

limited activities.[11] Its function is coordination and capacity building among EU-member CSIRTs and to provide advice on some EU policies. While EU directive 2016/1148 will increase ENISA's statutory powers when it comes into effect in November 2018, at this point ENISA has little authority to force member states to take its advice. In the scheme of incident response practices, ENISA – founded in 2004 – is quite young. ENISA documents are informational, the one document interfacing with EU standards is an extended definition of the term "cybersecurity" and what EU work is done related to it (Brookson et al., 2015). Oddly, the document does not even mention incident response as an area within cybersecurity,[12] so it seems safe to leave ENISA out of scope.

NIST is a difficult organization to place in or out of scope. It is part of the Department of Commerce, and so has loose ties to the remaining Internet stewardship invested in the National Telecommunications and Information Administration. Strictly, NIST merely sets policies for how the US federal civilian government secures its IT infrastructure and responds to incidents (Cichonski et al., 2012). This Federal Information Security Management Act (FISMA) policy responsibility is a part of NIST's larger role of "advancing measurement science, standards, and technology in ways that enhance economic security and improve our quality of life" (U.S. Dept of Commerce, 2017). Through this role, NIST standardized AES, which is the *de facto* global standard encryption algorithm. NIST documents and standards are also cited by the IETF, ISO, and FIRST, elevating certain NIST work from a national to international status. I shall consider NIST generally out of scope; however, many NIST publications will be considered as works cited by the international standards organizations.

There are two US federal government units that do not fall under NIST's authority – the DoD and the Central Intelligence Agency (CIA). These two organizations have not published incident response standards as openly as NIST or these other standards organizations. The DoD does have other cybersecurity documents that are tangentially relevant. The Orange Book (Brand, 1985), which evaluates trusted computing platforms, is relevant background material for incident responders.

The questions the DoD and its sub-agency the National Security Agency (NSA) have raised around whether cybersecurity is, broadly, a science (see,

11 "Annual ex-post evaluation of ENISA activities" https://www.enisa.europa.eu/about-enisa/annual-ex-post-evaluation-of-enisa-activities

12 Despite the fact that ENISA sponsored an informational document on evidence gathering aimed at increasing understanding between CSIRTs and law enforcement (Anderson, 2015).

e.g., Galison (2012), Katz (2016) and MITRE (2010)) could inform evidence evaluation in incidence response because evaluating evidence properly is a primary scientific activity. While these DoD projects ask the right questions about science to help with incident response, they have generally concluded security is not (yet) a science, and so there is little advice. Chapter 3 will argue this conclusion is ill-founded and excessively pessimistic. However, the relevant point for this review is that the science of security literature does not advise CSIR.

While the main part of the DoD is out of scope, the intelligence community aspects of the US federal government do provide adequate documents. The DoD and CIA are generally not forthcoming with more conventional descriptions of their incident response practice. However, given that NIST is not authoritative over the IC, one would expect them to develop their own standard practices. Documents related to the practice of the US IC are occasionally published, with IC attribution either explicit or implicit. Three such documents are relevant to evidence collection and analysis in incident investigation, forming what is essentially a *de facto* standard. The first is a textbook published by the CIA and used to train intelligence analysts (Heuer, 1999) whose methods are applicable to CSIR. The second is a pair of documents, the kill chain model of computer network attacks (Hutchins et al., 2011) and the diamond model of intrusion analysis (Caltagirone et al., 2013). Unlike the textbook, these documents are not explicitly acknowledged as standard analysis methods within the defence and intelligence communities. However, the diamond model paper is published by the DoD publisher, the Defense Technical Information Center.[13] The diamond model builds on the kill chain. Given that Lockheed Martin, a US defence contractor, published the kill chain, it seems the papers are from overlapping communities. Although it is tenuous to term three documents a 'standard,' it is clear from the content that they come from a practitioner community and are one of the clearest available expressions of intrusion investigation methods publicly available. Therefore, it is necessary to place them in scope for discussion.

The US intelligence agencies exercise out-sized international influence. The US is part of an intelligence sharing alliance known as the five eyes, which includes Australia, Canada, New Zealand, and the United Kingdom. As the biggest partner in this alliance by far, what the US intelligence practitioners

13 See http://www.dtic.mil/docs/citations/ADA586960.

do is probably accommodated, if not directly copied, by the other countries' services.

US military influence goes beyond the five eyes. The North Atlantic Treaty Organization (NATO) is the biggest alliance the US leads, with 28 other countries. NATO intelligence is also presumably influenced by five eyes, as Canada and the UK also play a big role. The US tends to supply logistics and intelligence support in its alliances, so intelligence standards are likely to influence allies. Other locations which cooperate extensively with the US include Israel, South Korea, Japan, and the Philippines. By virtue of these alliances, it is reasonable to assume that intelligence professionals in all these places are relatively closely aligned with US intelligence standards. These alliances end up including most of the global military and intelligence spending. Essentially only China and Russia are excluded, and the two of them account for 15-20% of global military spending. Thus, although there are rather few IC documents, and they are focused on the US, they should provide information about how a large swath of such practitioners make decisions.

In summary, this review will include the IETF, ISO, FIRST, and available intelligence community documents as in-scope publication venues for incident investigation standards of practice for evidence collection, analysis, and reporting. The review will exclude the ITU, W3C, ENISA and US federal civilian government departments and agencies as out of scope due to either limited applicable content or limited jurisdiction. The most borderline organization is NIST, which occasionally has standards canonized by the in-scope venues; the review will only include those NIST standards cited or adopted explicitly by the four in-scope venues.

Section 2.4.1 describes the method for determining which standards are relevant within these venues.

2.3 RELATED LITERATURE REVIEWS

This section provides a structured survey of the literature to demonstrate that the intended scope, as defined in Section 2.2, has not been previously surveyed. Lack of related surveys in two academic venues provide evidence: IEEE Security and Privacy Systematization of Knowledge (SoK) papers and ACM Computing Surveys (CSUR) journal. I will appraise the 35 extant SoK papers (as of August 1, 2017) for relevance. For ACM CSUR I apply a keyword search to the corpus.

IEEE S&P has published 35 SoK papers since the venue initiated the SoK publication option in 2010. I evaluated relevance based on title and abstracts. The basic relevance criterion in this case is if the SoK is about designing or evaluating investigations of maliciousness. Of the 35 SoK papers, only Herley and van Oorschot (2017) and Rossow et al. (2012) are applicable. Chapter 3 will addresses the shortcomings in Herley and van Oorschot (2017) in some detail. In prior work (Hatleback and Spring, 2014), I expanded on the good work of Rossow et al. (2012); however, Rossow et al. (2012) is too narrow for my current scope as it focuses just on malicious software research. None of the SoK papers systematize knowledge of incident response, investigation, or analysis.

My CSUR keyword search uses Google Scholar, limiting to publications in "ACM Computing Surveys" between Jan 1, 2007 and Aug 1, 2017. I use the same keywords as the main study, described in Section 2.4.1. However, CSUR is a sufficiently different venue from the intended scope that I found some different keywords more useful. The surveys returned by the following search terms are included in Table 2.1. Quotes are applied to the search as listed.

1. "computer security incident response"
2. "incident investigation"
3. "incident management"
4. "computer security" & "evidence collection"
5. "incident response" & analysis
6. "security incident" & investigation
7. "computer security" & incident investigation
8. "computer security" & incident analysis

These eight searches within CSUR return 22 unique results. Note in particular that the two most precisely relevant searches return no matches. As the search terms are expanded to include more general, related terms, the results include a handful of possibly relevant papers.

The following search terms were tried on CSUR but returned too many clearly irrelevant results to be considered useful, with the total survey papers returned in brackets. The relevant papers appear to be already included in the 22 captured by the eight search terms used.

- "computer security" & analysis (79)
- "computer security" & reporting (25)

	Found in search #								Criteria		
Document	1	2	3	4	5	6	7	8	1	2	3
Li et al. (2017a)			■						✗	✗	✓
Li et al. (2017b)			■						✓	✗	✗
Jiang et al. (2016)				■					✗	✗	✗
Jhaveri et al. (2017)					■		■	■	✓	✗	✓
Pendleton et al. (2016)					■		■	■	✗	✗	✗
Khan et al. (2016)					■		■	■	✓	✓	✗
Laszka et al. (2014)						■	■	■	✗	✓	✓
Biddle et al. (2012a)							■	■	✗	✗	✗
Milenkoski et al. (2015)							■	■	✗	✓	✗
Tang et al. (2016)							■		✗	✗	✗
Meng et al. (2015)							■	■	✗	✗	✗
Calzavara et al. (2017)							■	■	✗	✓	✗
Labati et al. (2016)							■	■	✗	✓	✗
Ye et al. (2017)							■	■	✗	✓	✗
Edwards et al. (2015)*							■	■	✓	✓	✓
Avancha et al. (2012)							■	■	✗	✗	✗
Roy et al. (2015)							■	■	✗	✓	✗
Chandola et al. (2009)							■	■	✓	✓	✗
Pearce et al. (2013)								■	✓	✓	✗
Peng et al. (2007)								■	✗	✓	✗
Younan et al. (2012)								■	✗	✓	✗
Egele et al. (2008)*								■	✓	✓	✗

Table 2.1: Other literature reviews potentially related to computer security incident investigation from ACM CSUR. The three relevance criteria are (1) relates to forensics rather than prediction; (2) technical, investigative focus; (3) useful level of abstraction of incident investigation. Papers with an asterisk (*) are discussed in more detail in the text.

To determine whether any of these 22 surveys already adequately cover the topic of interest, I define three relevance criteria; namely, the survey must:

1. relate to reconstructing past events (i.e., forensics) rather than prediction;
2. focus on the technical- and knowledge-based decisions and processes, rather than management processes;
3. use my target level of abstraction to discuss the problem of incident response, investigation, or analysis (human decisions during the process),

rather than tool development, without being so abstract as to make implementation impractical.

These criteria are marked in Table 2.1, based on each paper's abstract. Some papers deserved a look beyond their abstracts.

Edwards et al. (2015) is the only survey that meets all three criteria, based on the abstract. However, their focus is quite different from my focus. They discuss automation of law-enforcement criminal investigation using computer science techniques. There may be overlap with computer-security incident response, in that some subset of law enforcement cases involve criminal action against computers. However, the focus of Edwards et al. (2015) is what Anderson et al. (2012, p. 3) call, quoting the European Commission, "traditional forms of crime... committed over electronic communication networks and information systems." Incident response and investigation focuses on a different category, "crimes unique to electronic networks," as well as organizational policy violations that are not illegal under the relevant jurisdiction. Finally, Edwards et al. (2015) focus on automation of police tasks, whereas my focus would be on the investigator's decision process in, among other things, choosing which automation techniques to use and how to evaluate the evidence they provide. These various differences make a clear case that my intended survey topic is sufficiently distinct from Edwards et al. (2015).

Egele et al. (2008, p. 1) aims to identify "techniques to assist human analysts in assessing ... whether a given sample deserves closer manual inspection." It is, in fact, a survey of software tools and their features, and does not discuss how an analyst should use them.

Many papers in Table 2.1 meet the technical criterion (#2) and fail the other two criteria. This pattern tends to be about some specific subset of network defence – for example, making better passwords, intrusion detection systems, or web browser defences. These tools are certainly used and evaluated as part of security management, and are important considerations. Nonetheless, these details are tangential to making decisions during CSIR.

These reviews of the available survey literature demonstrate a lacuna; the literature lacks a survey of CSIR practices. This omission matters. Most security research and security management requires, directly or indirectly, "ground truth" evidence from incident response teams. Research and management need this "ground truth" evidence to evaluate any other security infrastructure, plans, defences, or policy. However, it seems there is no systematic review of how this evidence should be collected, analysed, and reported. One must

understand these steps in order to properly interpret any such evidence. Therefore, although the topic of human decision-making during CSIR analysis is narrow, it has far-reaching impact on cybersecurity more generally.

2.4 METHODS

The scope here is restricted to evidence collection, analysis, and reporting in human-driven computer security incident response. It is further restricted to internationally-recognized standards. This choice maintains a pragmatic connection to actual professional practice without violating confidentiality around incident response, which organizations justifiably do not often disclose in detail. Section 2.3 demonstrated that, at least within the most prominent information security publication venues, no review overlapping this scope has been published previously.

This section explains the review methodology in three parts. First, search strategy. Secondly, the appraisal strategy for whether to include results in the synthesis. And finally, how to synthesize the results into a coherent statement of current practice.

2.4.1 *Search strategy*

The major determining factor in the literature search strategy is the scope of publication venues, as justified in Section 2.2.2. The search comes in two parts. First, I perform keyword searches in the relevant web archives. Secondly, I extract references from valid hits on these searches and include the referenced documents as sources to appraise.

The IC documents are selected by fiat, given the secretive nature of the community. Due to the idiosyncratic nature of the IC publication and publicity processes, there is no sense in a keyword search strategy. I arrived at the three core documents as the "standards" from this community essentially by word of mouth through interaction with practitioners.

Each of IETF,[14] FIRST,[15] and ISO[16] have dedicated web pages. For IETF and ISO, I use their site-based search engines that cover their respective corpora of standards. FIRST has a smaller, more focused corpus of work, such that it

14 https://www.rfc-editor.org/search/rfc_search.php
15 https://first.org/standards/
16 https://www.iso.org/standards.html

does not have a site specific search engine; I use Google and prepend the term "site:first.org" to focus the search.

Quotes are applied to the search as listed. The keywords employed are:

1. "computer security incident response"
2. "incident investigation"
3. "incident management"
4. "computer security" & "evidence collection"
5. "computer security" & analysis
6. "computer security" & reporting

I also added or modified terms slightly to accommodate each search venue. For example, the IETF RFC search tool does not accommodate mixing quoted phrases with other terms, so for terms 4, 5, and 6 the quotes were removed. I added the following terms to the IETF search:

- "incident response"

I added the following terms to the ISO search, after it became apparent from searches 2 and 3 that the ISO documents do not use the term "computer security" but rather "information security":

- "information security" & "evidence collection"
- "information security" & analysis
- "information security" & reporting

All the documents returned by this search strategy are appraised using the methods of Section 2.4.2. I then take a further search step and extract the references from those documents that pass the appraisal. I only include cited documents which are publicly available (or, in the case of ISO, readily available). All the documents extracted from the references are appraised using the same methods to determine whether they are included in the review, independent of the document that cited it.

2.4.2 *Appraisal strategy*

The purpose of the appraisal is to determine whether each document is within the scope of evidence collection, analysis, and reporting for incident investigation. The cut-off date for inclusion in the review is publication prior to July 1, 2017.

Standards may be superseded or amended by future work. Drafts are also commonly published for public comment before being finalized. I exclude any standard superseded as of July 1, 2017, and incorporate any amendments finalized by July 1, 2017. The existence of drafts on new topics is noted, but their content is excluded from the review.

Inclusion criteria for whether the content is in-scope are:

- Target audience as expressed by author includes security professionals
- Topic is within scope, namely it applies to one of the following parts of computer-security incident investigation (or some clear synonym thereof)
  - evidence collection
  - analysis
  - reporting
- Topic is on investigator practice (rather than implementation of software or managerial considerations related to CSIRTs)
- Document is finalized (not a draft) and not explicitly superseded as of July 1, 2017
- Document is available in English

A document must satisfy all of these criteria to be included.

2.4.3 *Synthesis methods*

The input at this stage of the method will be all documents that are in scope; they will be documents for security professionals about investigator practice during evidence collection, analysis, and reporting. My synthesis goal is to evaluate the nature and quality of advice that these documents provide about making decisions during these phases of incident response.

As a prelude to this synthesis, I will classify advice on these topics in several ways: to which phases the document applies, the directness with which the document applies to each phase, the type, investigative goals supported, broadness of scope, generalizability of advice, and formalism. I use this initial evaluation to identify groupings of documents and get an overview what the literature search has found to be available. The following describes each evaluation in more detail.

PHASES indicates what combination of evidence *collection*, *analysis*, and *reporting* the document covers.

DIRECTNESS has two possible values: *direct* and *constraints*. Direct commentary on incident investigation explicitly talks about what an investigator should or should not do. Constraints provide only requirements for outcomes or outputs, and do not indicate how these properties should be achieved. Constraint-based advice is common when situating investigation within the larger context of incident response, and situating response within incident management.

TYPE indicates what type and level of detail the document provides to decision making. Possible values are *case study*, *ontology*, *advice*, and *instructions*. At one end of the spectrum are case studies. Case studies report the facts of an individual case of investigation, without attempting to abstract up to general lessons. A categorization forms categories of useful actions (implicitly or explicitly from case studies), but gives no advice on how to apply these categories. Advice provides some ordering on what category of action should be taken, given certain conditions. Finally, instructions provide explicit decision-making instructions on how to evaluate options. Type also provides some rough guide on how much effort it will take to apply the document to practice, with case studies being the most difficult.

GOALS indicates what sort of investigation the advice targets. An investigator could have, for example, three goals: *fix* the system, gather *intelligence* on adversaries, and make *arrests*. Certainly, there may be other goals, but these cover a wide degree of practical differences. Investigators need quite different information between these goals. For example, to fix a system, one needs to know everything that has been accessed by the adversary, but you need to know rather little about them. Whereas to make arrests, one cares very much about the adversary, but also is bound by several practical matters of what counts as admissible legal evidence of attribution and loss. When gathering intelligence on what an adversary may do next, these legal considerations fall away, but one also focuses on quite different aspects than fixing a system. For example, to gather adequate intelligence one need not enumerate all compromised systems.

SCOPE reports how widely the document applies, as reported by the document. Options are *narrow*, *medium*, and *broad*. A narrowly-scoped

document targets only a small or unrepresentative group of people, and/or for a short period of time. Broad scopes are intended for most people within cybersecurity. Medium scope fits somewhere in between. Examples of medium scope are US-based law enforcement forensics specialists, or the operators of tier-three (that is, backbone) networks.

GENERALIZABILITY of advice indicates how likely the document can be relevant to contexts outside those for which it was specifically designed. Generalizability is explicitly level-set from the document's scope. Thus, a document with broad scope but no generalizability may still be applicable to more people than a narrowly-scoped document that is generalizable. Whereas scope is a measure taken directly from the document being evaluated, generalizability is an evaluation of potential not explicit in the document. Indicators of generalizability include use of models or methods from other disciplines with well-established other uses or evidence from sources other than the document itself that the advice from the document applies more widely. Options for this criterion are coarsely set as *unlikely* ($< 15\%, \pm 5\%$), *likely* (in between unlikely and highly likely), and *highly likely* ($> 85\%, \pm 5\%$); values represent essentially the evaluator's prior belief on the document being applicable outside its stated scope. A final value, *widely*, indicates the document is certainly generalizable beyond its scope and is likely to be generalizable to a much broader scope.

FORMALISM reports the degree of mathematical formalism present in the advice provided. Options are *none*, *qualitative*, *formal*, and *perfunctory*. Perfunctory indicates formalization is present, but essentially unused to advance the arguments or positions of the document. This rating does not mean the formalism is wrong; however, it does indicate it would take significant effort on the part of the reader to make use of the formalism beyond what qualitative models would provide. None only applies to narratives that make no attempts at abstraction. Both qualitative models and formal mathematical models have value in their own ways, and one should not be considered preferred over the other per se.

After this classification of the documents, I will undertake a more free form synthesis of the results. I focus on how analysts and investigators make decisions about evaluating the quality and importance of evidence, generalize from particular evidence to evidence of trends or patterns of behaviour, and

select what to report based on security constraints as well as what others find convincing. Although these align loosely with the three in-scope phases of CSIR, there is not a one-to-one connection between the three phases and the three focal points. For example, if an analyst or investigator knows some kind of evidence is particularly convincing to report, that should impact what they look for during the evidence collection phase. Section 2.6 reports the results of this classification, examination of focal points, and identification of preliminary gaps.

Section 2.7 identifies what gaps remain after this cursory but broad pass through potentially helpful fields. Section 2.7.1 suggests a research question to make progress on addressing these gaps. Section 2.8 identifies related work that may be useful to fill these gaps but of which the standards do not make use. Section 2.9 concludes the synthesis by suggesting possible research for moving forward, given the gaps and the related work.

2.4.4 *Limitations*

While these methods have much to recommend them, there are of course limitations. Some of these are practical, such as the restriction to publications available in English. Some limitations are a function of restricting the scope to standards. Perhaps the most dangerous limitation is a result of the subject matter – security practitioners tend to be secretive about their methodologies.

The restriction to English will naturally limit the results. For example, an internet search for '信息安全事件应对' (information security incident response) returns a couple dozen results on Google as well as Google Scholar. This seems to be the preferred term in mainland China. A search for '电脑安全事件应对' (computer security incident response) returns only a couple of Taiwanese sites.

The importance of this language choice on actually limiting available documents is less clear. EU and UN documents would be available in English as a matter of policy. The US government, which publishes in English, dominates this space, as do US companies. Countries that are not allied to the US and have developed computer security capabilities are relatively few; basically just the Russian Federation and the People's Republic of China (PRC). While it is possible these countries have published comprehensive decision making procedures for incident response, it seems unlike these regime's dispositions to publish such things widely. It also seems likely that, given how much attention

the US security establishment pays to Russia and the PRC, if such a thing were published it would be found and reported on, if not translated. For these reasons, I judge the impact of limiting the search to English documents is a low risk.

Focusing on standards, and what they cite, limits the scope but also creates other limitations. Indeed, reducing the scope to something manageable is one goal of focusing on standards, and this seems to function as intended. However, the type of information published in standards is different than that in academic journals and conferences, and this imposes some limits on the work. Specifically, standards are on a slower publication cycle than academic work. This delay would be a problem if the topic were covered much in the academic literature. However, as indicated by Section 2.3, academic publications do not appear to cover decision making during computer security incident response.

Creation of technology standards is itself a complex process, and as Section 2.2.2 discussed, the process has a complex history in its own right (Spring, 2011c). The way standards are made limits the findings as well. Standards are rarely made purely for the dissemination of information; rather, they usually solidify a dominant business position. Security standards appear to be an outlier from this norm, as they have unique concerns about correctness and non-subversion (Kuhlmann et al., 2016). Incident response standards are essentially unstudied within the academic standards literature; Kuhlmann et al. (2016) mostly focus on cryptographic standards. This situation means accepting a risk in that no one discusses openly what biases may be embedded in the creation of incident response standards. The standards literature provides evidence there will be a bias, but has not studied incident response standards in order to provide evidence for what that bias might be. One important question is whose interests are best served by the creation of incident response standards.

A closely related limitation of concern involves secrecy. Many incident response organizations may not wish to disclose their processes and procedures in detail lest the adversary learn how to subvert or avoid them. Other areas of cybersecurity experience similar publication restrictions. Incident responders likely have a legitimate concern in this regard, and may also have a legal or regulatory requirement to keep certain information or processes private. Therefore, this limitation imposes a significant risk that relevant information is not public. Lack of access obviously limits the review. My approach to reduce the impact of this limitation is to read between the lines of available documents

when plausible, expanding the interpretation of a document's contents with circumstantial evidence from the context surrounding its publication. However, I must accept that there is an amount of information about this topic which simply is not public and I cannot hope to access for a public literature review. One could perhaps use news articles and audit reports to attempt to evidence the extent to which organizations in fact implement the available standards; I leave such studies for future work. The community first needs a baseline understanding of the standards literature from which to begin.

2.5 RESULTS

Sections 2.5.1 through 2.5.4 present the results per search venue. Section 2.5.5 takes the results from these four venues, extracts the referenced documents from the results, and evaluates these cited documents.

Table 2.2 lists the relevant documents to analyse in depth. These 29 documents are the result of reducing from roughly 350 possible documents examining search results and references.

RFC 2196, §5.4 only (Fraser, 1997)	27035-1 ISO/IEC (2016)
RFC 6545 (Moriarty, 2012)	27037 ISO/IEC (2012)
RFC 7203 (Takahashi et al., 2014)	27041 ISO/IEC (2015a)
RFC 7970 (Danyliw, 2016)	27042 ISO/IEC (2015b)
RFC 8134 (Inacio and Miyamoto, 2017)	27043 ISO/IEC (2015c)
NIST SP 800-61 (Cichonski et al., 2012)	Kossakowski et al. (1999)
NIST SP 800-86 (Kent et al., 2006)	Alberts et al. (2004)
NIST SP 800-83 r.1, §4 only (Souppaya and Scarfone, 2013)	
Gorzelak et al. (2011)	Mundie et al. (2014)
Kill chain (Hutchins et al., 2011)	Heuer (1999)
Diamond model (Caltagirone et al., 2013)	Ciardhuáin (2004)
Carrier and Spafford (2004)	Casey (2010, ch. 2)
Osorno et al. (2011)	Cheswick (1992)
Joint Chiefs of Staff (2014c, ch. 5 only)	Stoll (1988)
Leigland and Krings (2004)	Mitropoulos et al. (2006)

Table 2.2: All documents found to be relevant through the search methodology

2.5.1 *IETF*

	Criteria				
Document	1	2	3	4	5
RFC 2350 (Brownlee and Guttman, 1998)	✓	✗	✓	✓	✓
RFC 3607 (Leech, 2003)	✓	✗	✗	✓	✓
RFC 5070 (Danyliw et al., 2007)	✓	✓	✓	✗	✓
RFC 6045 (Moriarty, 2010)	✓	✓	✓	✗	✓
RFC 6046 (Moriarty and Trammell, 2010)	✓	✓	✓	✗	✓
RFC 6545 (Moriarty, 2012)	✓	✓	✓	✓	✓
RFC 6546 (Trammell, 2012b)	✓	✗	✓	✓	✓
RFC 7203 (Takahashi et al., 2014)	✓	✓	✓	✓	✓
RFC 7970 (Danyliw, 2016)	✓	✓	✓	✓	✓
RFC 8134 (Inacio and Miyamoto, 2017)	✓	✓	✓	✓	✓

Table 2.3: IETF database search results. The criteria are (1) target audience is security professionals; (2) topic in scope, per Section 2.2.1; (3) focus is investigator practices; (4) document finalized and not obsoleted as of Aug 1, 2017; (5) available in English.

Table 2.3 evaluates the results of the search procedure. Four documents on meet criteria: RFCs 6545, 7203, 7970, and 8134.

The IETF documents break down into two clear broad categories, BCP 21 (Best Current Practice) on expectations for computer security incident response (Brownlee and Guttman, 1998), and all the others, which are to do with Incident Object Description Exchange Format (IODEF), its expansion, and usage.

As an expectations document, BCP 21 focuses primarily on the services and support a CSIRT should provide to its constituency, who that constituency should include, and so on. These considerations are vital to CSIRT operations; however they are not directly relevant to the research questions.

The IODEF projects in particular feature CERT/CC staff heavily. The authors Danyliw and Inacio worked there during their RFC authorship and still do, and Trammell contributed heavily to the SiLK (System for Internet-level Knowledge) tool suite at CERT/CC. Because CERT/CC is also heavily involved in FIRST, it is unsurprising to see a sort of division of labour between the IETF documents and the FIRST documents. In particular, the IODEF format focuses almost exclusively on technical issues of data exchange and reporting format.

The softer considerations, of how to collect, evaluate, and analyse the data contained within IODEF remain in the purview of FIRST.

As technical reporting formats are in scope as reporting results, all RFCs related to IODEF are relevant. There do not appear to be any other IETF documents within scope.

These IODEF documents may at first seem to be out of scope, as the scope as specified is how investigators make decisions, not what tools or formats they use to document them. This topic recurs in Section 2.5.5.1. IODEF is essentially a language for talking about computer security incidents. However, because data formats are out of scope, the language per se is out of scope. IODEF is in scope because as a constructed language it makes judgments about what aspects of incidents are important, necessary, or possible to communicate. These judgments, at least implicitly, bear on what an investigator should choose to report. I therefore judge IODEF as in-scope. However, the scope remains how to decide what information to report, not the language used to report it. Therefore, data formats and languages for anything else remain out of scope.

2.5.2 *ISO*

Nine search terms return 31 total results, with 23 unique results displayed in Table 2.4. Three standards meet the criteria to carry through to the citation-harvesting and synthesis stage:

- *Information security incident management – Part 1: Principles of incident management.* 27035-1 (ISO/IEC, 2016)
- *Guidance on assuring suitability and adequacy of incident investigative method.* 27041 (ISO/IEC, 2015a)
- *Incident investigation principles and processes.* 27043 (ISO/IEC, 2015c)

2.5.3 *FIRST*

FIRST is the smallest body surveyed, and it is not primarily a standards organization but rather a forum for organizations with a shared purpose – incident response.

On its "standards" web page, FIRST lists four standards it maintains:

	Criteria				
Document	1	2	3	4	5
IEC 31010:2009	✓	✗	✓	✓	✓
ISO 13485:2003	✗	✗	✗	✗	✓
ISO/IEC 17799:2005	✓	✗	✗	✗	✓
ISO/IEC 27000:2009	✓	✗	✗	✗	✓
ISO/IEC 27001:2005	✓	✗	✗	✗	✓
ISO/IEC 27002:2005	✓	✗	✗	✗	✓
ISO/IEC 27004:2016	✓	✗	✗	✓	✓
ISO/IEC 27005:2011	✓	✗	✗	✓	✓
ISO/IEC 27006:2011	✗	✗	✗	✗	✓
ISO/IEC 27006:2015	✗	✗	✗	✓	✓
ISO/IEC 27033-1:2009	✓	✗	✗	✗	✓
ISO/IEC 27033-4:2014	✓	✗	✗	✓	✓
ISO/IEC 27035:2011	✓	✓	✓	✗	✓
ISO/IEC 27035-1:2016	✓	✓	✓	✓	✓
ISO/IEC 27035-2:2016	✓	✗	✗	✓	✓
ISO/IEC 27041:2015	✓	✓	✓	✓	✓
ISO/IEC 27043:2015	✓	✓	✓	✓	✓
ISO/IEC TR 18044:2004	✗	✓	✓	✗	✓
ISO/IEC TR 20004:2015	✓	✗	✗	✓	✓
ISO/NP TS 11633-1	✓	✗	✗	✗	✓
ISO/TR 11633-1:2009	✓	✗	✗	✓	✓
ISO/TR 11633-2:2009	✓	✗	✗	✓	✓
ISO/TS 19299:2015	✗	✗	✗	✓	✓

Table 2.4: ISO database search results. The criteria are (1) target audience is security professionals; (2) topic in scope, per Section 2.2.1; (3) focus is investigator practices; (4) document finalized and not obsoleted as of Aug 1, 2017; (5) available in English.

Traffic Light Protocol (TLP) on agreed-upon levels for marking information sensitivity

Common Vulnerability Scoring System (CVSS) on describing the characteristics and severity of defects in software systems (not to be confused with Common Weakness Scoring System (CWSS) by the Mitre Corporation (MITRE))

Information Exchange Policy (IEP) is a reporting format; in this regard it is another language for reporting, similar to IODEF or those listed in Table 2.7. IEP's focus is on disseminating information responsibly and

	Criteria				
Document	1	2	3	4	5
Mundie et al. (2014)	✓	✓	✓	✓	✓
Alberts et al. (2004)	✓	✓	✓	✓	✓
OCTAVE (Caralli et al., 2007)	✓	✗	✗	✓	✓
ENISA (2006)	✗	✓	✓	✓	✓
Cormack (2015)	✓	✗	✗	✓	✓
Gorzelak et al. (2011)	✓	✓	✓	✓	✓
Cichonski et al. (2012)	✓	✓	✓	✓	✓
ETSI (2014)	✓	✗	✓	✓	✓
RFC 2350 (Brownlee and Guttman, 1998)	✓	✗	✓	✓	✓
RFC 2196 (Fraser, 1997, §5.4 only)	✓	✓	✓	✓	✓
RFC 2827 (Ferguson and Senie, 2000)	✓	✗	✗	✓	✓
RFC 2504 (Guttman et al., 1999)	✗	✓	✓	✓	✓

Table 2.5: FIRST results summary. The criteria are (1) target audience is security professionals; (2) topic in scope, per Section 2.2.1; (3) focus is investigator practices; (4) document finalized as of Aug 1, 2017; (5) available in English.

quickly during incident response, which gives it a different focus than these other formats.

passive Domain Name System (DNS) (pDNS) is a formatting standard for DNS traffic analysis; the FIRST group is working on an IETF standard.

None of these standards meet the relevance criteria, because none are about investigator practice. They all represent things a competent investigator should understand, but they do not discuss decisions an investigator should make in a given scenario. FIRST also notes it contributes to several ISO standards, which Section 2.5.2 covers (namely, 27010, 27032, 27035, 27037, and 29147).

More instructive than these standards are FIRST's "Security Reference Index", described as "helpful" to the FIRST community.[17] FIRST's members include many of the relevant professionals and practitioners. The documents Table 2.5 evaluates are listed as either best practices or standards in this reference index.[18] Five documents emerge as relevant: Alberts et al. (2004), Cichonski et al. (2012), Fraser (1997), Gorzelak et al. (2011) and Mundie et al. (2014).

17 https://first.org/resources/guides/reference

18 Strictly speaking, Mundie et al. (2014) and Alberts et al. (2004) are not linked directly; they are the most relevant part of a suite of publications linked to by FIRST as https://www.cert.org/incident-management/publications/index.cfm.

The security reference index also links to the home pages of other security organizations; however, I do not review all the content these organizations have produced in full. In large part, the information is more about solving specific technical problems than my target for a general problem solving method. Such specific problems make for instructive cases when thinking about generalized methods, and so these organizations do provide an integral relevant function. But they do not aim for the types of documents in scope of this review. The organizations identified are:

Center for Applied Internet Data Analysis (CAIDA), www.caida.org

CERT® Coordination Center (CERT/CC), www.cert.org

Center for Internet Security (CIS) Benchmarking, http://www.cisecurity.org/

TEAM CYMRU A security think tank, https://www.team-cymru.org/services.html

EU Agency for Network and Information Security (ENISA), https://www.enisa.europa.eu/, including CSIRT services https://www.enisa.europa.eu/topics/csirt-cert-services

Open Web Application Security Project (OWASP), https://www.owasp.org/index.php/OWASP_Guide_Project

MICROSOFT Security Guidance Center, https://technet.microsoft.com/en-us/library/cc184906.aspx

Sysadmin, Audit, Network, and Security Institute (SANS Institute) reading room, https://www.sans.org/reading-room/

Although ENISA (2006) targets management rather than practitioners, it provides links to training for practitioners. The two organizations the report lists are CERT/CC and the EU-funded TRANSITS.

2.5.4 *Intelligence community*

The canonical training course for CIA and other intelligence analysts is Heuer (1999). The book is essentially applied psychology. It covers topics such as analysing competing hypotheses, which includes evaluating whether evidence has been planted to deceive, as well as overcoming human cognitive biases such as anchoring, vividness, and oversensitivity to consistency. Such methods, especially for evaluating evidence in the face of deception, have clear relevance to incident investigation.

	Criteria				
Document	1	2	3	4	5
Heuer (1999)	✓	✓	✓	✓	✓
Hutchins et al. (2011)	✓	✓	✓	✓	✓
Caltagirone et al. (2013)	✓	✓	✓	✓	✓

Table 2.6: Intelligence community results summary. The criteria are (1) target audience is security professionals; (2) topic in scope, per Section 2.2.1; (3) focus is investigator practices; (4) document finalized as of Aug 1, 2017; (5) available in English.

The model of a computer attack as following a predictable 'kill chain' of steps from start to finish was published by Lockheed Martin incident responders (Hutchins et al., 2011). The seven steps are reconnaissance, weaponization, delivery, exploitation, installation, command-and-control, and actions on objectives. These are the steps in one single attack – a single phishing email, a single drive-by download with a malicious advert, etc. Adversaries almost always compose a campaign out of multiple attacks; the objectives of one attack may be to obtain a platform from which further attacks are possible. The purpose of this model "is to capture something useful about the pattern all, or at least nearly all, attacks follow," so the analyst can anticipate what to look for or expect next (Spring and Hatleback, 2017, p. 10).

Caltagirone et al. (2013) builds on attack ontologies, specifically the kill chain, and intelligence analysis to perform attribution in computer security incidents and analysis of whole campaigns. The method incorporates Bayesian statistics to model belief updates of the analyst. These statistical details are explicitly intended to help overcome analyst cognitive biases, such as those discussed in Heuer (1999).

2.5.5 *Referenced Documents*

In this section, I will harvest citations from the standards identified as relevant in the prior parts of Section 2.5. The evaluations are detailed in a subsection for each publication venue. The documents harvested from citations that are directly relevant to the review are:

- Carrier and Spafford (2004)
- Casey (2010, ch. 2)
- Ciardhuáin (2004)

- Leigland and Krings (2004)
- 27037 ISO/IEC (2012)
- 27042 ISO/IEC (2015b)
- NIST SP 800-83 rev 1, §4 only (Souppaya and Scarfone, 2013)
- NIST SP 800-86 (Kent et al., 2006)
- Osorno et al. (2011)
- Kossakowski et al. (1999)
- Cheswick (1992)
- Stoll (1988)
- Mitropoulos et al. (2006)
- Joint Chiefs of Staff (2014c, ch. 5 only)

2.5.5.1 *IETF documents*

The four relevant IETF standards reference 97 unique documents, excluding the IODEF-related standards already considered in Section 2.5.1. These documents fall into three broad categories: technical implementation requirements and dependencies; other related computer-security-incident report formats; and broader incident handling guidelines that describe the larger analysis and response context within which the reporting formats are used. This first category of implementation dependencies is not relevant. Therefore, other reporting documents and broader incident handling guidelines are the important results. There are 22 cited reporting-related documents and exactly one related to broader incident handling and use of the reporting formats.

Table 2.7 lists the reporting formats and data sources cited by the IETF results in Section 2.5.1. Other report formats are primarily produced by NIST and MITRE – with funding from the US government including NIST. These projects also include the only referenced documents that are continuously updated data archives. The data formats for CVE, CWE, CWSS, CCE, CCSS, CPE are also continuously populated and published by NIST and MITRE as new vulnerabilities, platforms, etc. are discovered or developed. Thus these projects provide not only a format, but a standard reference dictionary of the possible terms with which the format may be populated. CVSS is perhaps the most important of these metrics which provide data and scoring; for a survey that relates CVSS to other security metrics, see Pendleton et al. (2016).

These standard dictionaries are referenced by many of the other data formats which inherit the field essentially as a data type. For example, MAEC may indicate which vulnerability a malware targets using its CVE number.

Publisher	Type	Name
CERT/CC	Architecture	Automated Incident Reporting (AirCERT)
ICASI	Format	Common Vulnerability Reporting Framework (CVRF)
IEEE	Format	Malware Metadata Exchange Format (MMDEF)
IETF	Format	Intrusion Detection Message Exchange Format (IDMEF)
	Format	Incident Object Description Exchange Format Extensions (IODEF+)
	Format	RFC 5941, Sharing Transaction Fraud Data (extends IODEF)
ISO	Format	Software asset management: Software identification tag (ISO 19770)
FIRST	Data	Common Vulnerability Scoring System (CVSS)
MITRE	Format	Common Attack Pattern Enumeration and Classification (CAPEC)
	Format	Common Event Expression (CEE)
	Data	Common Vulnerabilities and Exposures (CVE)
	Data	Common Weakness Enumeration (CWE)
	Data	Common Weakness Scoring System (CWSS)
	Format	Malware Attribute Enumeration and Characterization (MAEC)
	Format	Open Vulnerability and Assessment Language (OVAL)
NIST	Data	Common Configuration Enumeration (CCE)
	Data	Common Configuration Scoring System (CCSS)
	Data	Common Platform Enumeration (CPE)
	Format	Open Checklist Interactive Language (OCIL)
	Format	Security Content Automation Protocol (SCAP)
	Format	Extensible Configuration Checklist Description Format (XCCDF)
XMPP	Format	XEP-0268 Incident Handling (using IODEF)

Table 2.7: Computer-security related reporting formats and data formats cited by IETF standards documents.

Such dictionaries are useful background knowledge during incident investigation, and help reduce confusion by providing common reference tags. However, following the pattern of other documents surveyed, these reference dictionaries do not provide agreed-upon evidence collection, field population, or analysis guidelines for their contents.

The next largest group of cited work from identified RFCs are three more IETF documents related to IODEF that did not appear in the original search. Other documents are also related to IODEF. CVRF and XEP-0268 extend and implement IODEF, respectively. AirCERT is a proposed implementation architecture that uses IODEF in automated indicator exchange. My conclusions about the IODEF results identified in Section 2.5.1 therefore apply equally to these other documents, and I will pass over them.

The remaining documents fall loosely into the NIST-MITRE orbit. ISO 19770 for asset management is developed separately from, but is related to, CPE and CCE, NIST's asset management for platforms and configurations, respectively. MMDEF is not directly related to MAEC; however, MAEC has adopted a significant component of the MMDEF schema.

The only citation related to actual decision making during an incident is NIST SP 800-61 (Cichonski et al., 2012). This document is already included from the FIRST results, see Section 2.5.3. Thus, the IETF citations do not add any new documents to the review.

2.5.5.2 *ISO documents*

There are three relevant ISO standards from which to extract references, namely ISO/IEC 27035, ISO/IEC 27041:2015, and ISO/IEC 27043:2015. ISO/IEC 27035 comes in two parts; although only the first part is in scope, I extract references from both parts. (Although ISO charges for access to its documents, all the bibliographies are freely available, so all documents are included in this step.)

There are 81 total references among the documents, with 64 unique references. Of these, 20 are elements of the ISO/IEC 27000 series of standards explicitly targeting information security. Four are the documents already referenced, and several others are already noted as not relevant in Table 2.4. However, the references add the following two 27000-series publications to the survey documents as relevant:

- 27037: *Guidelines for identification, collection, acquisition and preservation of digital evidence* (ISO/IEC, 2012)
- 27042: *Guidelines for the analysis and interpretation of digital evidence* (ISO/IEC, 2015b)

Of the remaining 44 referenced documents, 13 are further ISO standards. Specifically:

ISO 15489-1
ISO 8601
ISO 9000
ISO/IEC 10118-2
ISO/IEC 12207:2008
ISO/IEC 17024:2012
ISO/IEC 17025:2005
ISO/IEC 17043:2010
ISO/IEC 20000
ISO/IEC 29147
ISO/IEC 30111
ISO/IEC 30121
ISO/IEC/IEEE 29148:2011

All of these other ISO documents are out of scope. Further, the following are unavailable or already evaluated, and can be ignored: ILAC-G19, which directly follows from ISO 17020 and 17025; RFC 5070 (Danyliw et al., 2007), see Section 2.5.1; one by Valjarevic and Venter that is not available but appears by title and timing to be a working group presentation discussing the other two papers by these authors. I also will not consider the Daubert 1993 US Supreme Court case, to be jurisdiction neutral. Removing these leaves the documents listed and evaluated for relevance in Table 2.8. The following documents pass on to the next stage of analysis: Carrier and Spafford (2004), Casey (2010), Ciardhuáin (2004) and Leigland and Krings (2004). And, like the IETF, ISO cites NIST SP 800-61 (Cichonski et al., 2012).

Also cited are two further MITRE data formats not covered in Section 2.5.5.1: Structured Threat Information Expression (STIX) and Trusted Automated eXchange of Indicator Information (TAXII). These build on MAEC, CVE, and so on as formats for exchanging incident data. Like the other reporting formats already discussed, STIX and TAXII are not directly relevant to the incident response decision-making topic. While they are a language in which to report, and reporting is in scope, I judge these standards are focused mostly on the construction of an interoperable language system. Thus, they are out of scope, as the review is to discuss reporting in a language-independent way.

The Association of Chief Police Officers guidelines are representative of many of the documents in Table 2.8. Their target audience is "UK law enforcement personnel who may deal with digital evidence" (Williams, 2012,

	Criteria				
Document	1	2	3	4	5
Williams (2012)	✗	✓	✓	✓	✓
Valjarevic and Venter (2012b)	✓	✓	✓	✗	✓
Valjarevic and Venter (2012a)	✓	✓	✓	✗	✓
Carrier and Spafford (2003)	✓	✓	✓	✗	✓
Carrier and Spafford (2004)	✓	✓	✓	✓	✓
Edwards et al. (2009)	✗	✗	✗	✓	✓
Casey (2010)	✓	✓	✓	✓	✓
Cohen (2010)	✗	✗	✓	✓	✓
Cohen et al. (2011)	✗	✗	✓	✓	✓
Palmer (2001)	✗	✓	✗	✗	✓
Pollitt (2008)	✗	✓	✗	✓	✓
Reith et al. (2002)	✗	✓	✗	✓	✓
Beebe and Clark (2005)	✗	✓	✓	✓	✓
Ciardhuáin (2004)	✓	✓	✓	✓	✓
Leigland and Krings (2004)	✓	✓	✓	✓	✓
Rowlingson (2004)	✓	✗	✓	✓	✓
Hankins et al. (2009)	✗	✗	✓	✓	✓
SWGDE (2009)	✗	✗	✗	✓	✓
Garfinkel et al. (2009)	✓	✗	✗	✓	✓
Ballou et al. (2001)	✗	✗	✓	✓	✓
Alberts et al. (2014)	✗	✗	✗	✓	✓
Cichonski et al. (2012)	✓	✓	✓	✓	✓

Table 2.8: ISO database search results. The criteria are (1) target audience is security professionals; (2) topic in scope, per Section 2.2.1; (3) focus is investigator practices; (4) document finalized and not obsoleted as of Aug 1, 2017; (5) available in English.

p. 6). It is primarily about the legal chain of custody necessary to bring digital evidence to court. This topic is about evidence collection, so it is in scope. But the target audience is law enforcement, not security practitioners. The guidelines are not directly transferable to CSIR. The extent of comment on the actual work of understanding what the digital evidence means is constrained: "it is not practically possible to examine every item of digital data and clear tasking is needed to ensure that the digital forensic practitioner has the best chance of finding any evidence which is relevant to the investigation" (Williams, 2012, p. 10).

Some relevant work will not be carried through because it is obsoleted in a rather round-about way. Valjarevic and Venter (2012a, p. 1) notes "an effort to standardise the process has started within ISO, by the authors." Thus papers by these authors are obsolete because the authors directly subsumed their ideas into the ISO process. The process classes and activities used by ISO are clearly derived from Valjarevic and Venter (2012b, p. 6), which also contains a matrix of how these reference terms relate to other common forensic investigation ontologies. This set of works cited[19] matches the ISO work remarkably closely, as would be expected since the primary authors are the same. Unfortunately, Valjarevic and Venter (2012b) gives absolutely no methodology for how they arrived at this list of resources. Their analysis method is also not discussed, so it is unclear how or why they arrived at their categories and classification.

These omissions are particularly strange in that Valjarevic and Venter (2012b, p. 3) quotes Cohen et al. (2011) as rightly concluding the next steps in reaching consensus on and improving the field of digital forensics are a review of the literature that can be used to accurately drive consensus. This task is clearly what's been attempted, and as it has become an ISO standard it seems to have been accepted by a variety of practitioners. However, the lack of explanation of how these documents were selected as the correct set from which to drive consensus makes it hard to trace the authoritativeness of this source.

2.5.5.3 *FIRST Documents*

Five FIRST-related documents (Table 2.5) pass the evaluation of results and will have further citations harvested from them. Three of these documents do not have any citations ready to harvest. RFC 2196 (Fraser, 1997) does not have in-line citations, and only §5.4 is relevant, so the relevant citations to follow cannot be distinguished. Further, RFC 2196 is already 20 years old, and so following any citations would provide little modern benefit. On the other hand, Gorzelak et al. (2011) is a primary source – it is a survey of preventative practices at over 100 CSIRTs. Gorzelak et al. (2011) notes the tools that the respondents use, but it makes no citations to other incident investigation methodology documents. Alberts et al. (2004) is similarly a

19 Specifically, the overlapping works cited are Ballou et al. (2001), Beebe and Clark (2005), Carrier and Spafford (2003), Casey (2010), Ciardhuáin (2004), Cohen (2010), Leigland and Krings (2004), Reith et al. (2002) and Williams (2012)

primary source, though on incident management from CERT/CC. The only reference kept from Alberts et al. (2004) is where it explicitly indicates further information on incident analysis is contained in another CERT document, namely Kossakowski et al. (1999). Therefore, citations primarily come from Cichonski et al. (2012) and Mundie et al. (2014).

Cichonski et al. (2012) references three classes of resources. First is a list of incident response organizations, second is a list of NIST publications related to incident response, and finally a list of applicable data formats. The list of organizations includes many already discussed in Section 2.5.3. Those jointly listed by NIST and FIRST are CERT/CC, ENISA, and FIRST itself. NIST additionally lists the Anti-Phishing Working Group (APWG); Computer Crime and Intellectual Property Section (CCIPS); Government FIRST (GFIRST); High Technology Crime Investigation Association (HTCIA); InfraGuard; the Internet Storm Center (ISC); the National Council of ISACs; and US Computer Emergency Readiness Team (US-CERT). These organizations are certainly involved in various aspects of incident response. However, organizations as such are out of scope.[20]

	Criteria				
Document	1	2	3	4	5
SP 800-53 (Ross et al., 2013)	✓	✗	✗	✓	✓
SP 800-83 (§4) (Souppaya and Scarfone, 2013)	✓	✓	✓	✓	✓
SP 800-84 (Grance et al., 2006)	✗	✗	✓	✓	✓
SP 800-86 (Kent et al., 2006)	✓	✓	✓	✓	✓
SP 800-92 (Kent and Souppaya, 2006)	✓	✓	✗	✓	✓
SP 800-94 (Scarfone and Mell, 2007)	✓	✗	✗	✓	✓
SP 800-115 (Scarfone et al., 2008)	✓	✓	✗	✓	✓
SP 800-128 (Johnson et al., 2011)	✓	✗	✗	✓	✓

Table 2.9: NIST publications referenced by Cichonski et al. (2012). The criteria are (1) target audience is security professionals; (2) topic in scope, per Section 2.2.1; (3) focus is investigator practices; (4) document finalized and not obsoleted as of Aug 1, 2017; (5) available in English.

20 As a convenient sample, I have presented at APWG (Spring, 2013b; Spring et al., 2015) and attended InfraGuard meetings, and I do not expect there would be significant benefit in expanding the scope to include them. Likewise, I have interacted with several ISACs, and reviewed their available materials (Research and Education Networking Information Sharing and Analysis Center (ISAC) (REN-ISAC) and Financial Services Information Sharing and Analysis Center (ISAC) (FS-ISAC) especially) and do not believe they have any documents relevant to human decision-making in CSIR.

Table 2.9 lists and evaluates the relevance of the NIST publications referenced by Cichonski et al. (2012). All of these publications contribute to relevant background knowledge. For example, any incident response professional will need to know what a intrusion detection and prevention system is and how they are deployed (SP 800-94). But this topic is not about evidence collection, analysis, and reporting; it is merely necessary background knowledge. The two publications that are relevant are the guides to *Malware Incident Prevention and Handling for Desktops and Laptops* (Souppaya and Scarfone, 2013, §4 only) and *Integrating Forensic Techniques into Incident Response* (Kent et al., 2006).

The data exchange formats listed by Cichonski et al. (2012) are quite similar to those NIST, IODEF, and MITRE formats extracted from the IETF documents in Table 2.7. The only difference is the addition of Asset Identification, Asset Results Format, CVSS (from FIRST, see Section 2.5.3), and Cyber Observable Expression (CybOX). As discussed in Section 2.5.5.1, these formats are languages for reporting results, but do not directly discuss what to say. The research question here includes what to say in results, while being language agnostic, which puts these various formats and languages just outside the scope.

Mundie et al. (2014) cites 27 documents. They include several technical format documents for ontologies in the W3C Ontology Web Language (OWL), KL-ONE knowledge representation, knowledge graphs, process specification language, or the display tools used (Graphviz), which are out of scope. There are also psychology and ontology that are obviously out of scope (Baader, 2003; Miller, 1956). Further, four references have already been considered, namely ISO/IEC 27001, ISO/IEC 27002, Cichonski et al. (2012), and Beebe and Clark (2005). These exceptions leave the eight documents evaluated in Table 2.10. The only documents that pass the evaluation are Osorno et al. (2011) and Kossakowski et al. (1999).

MITRE (2010) on whether cybersecurity is a science is not within the current scope. However, since CSIR is an important subset of cybersecurity, whether security investigations are a kind of subcategory of scientific investigation clearly impacts the question of what incident investigation is and how to link it to knowledge generation and evidence evaluation more generally. Chapter 3 addresses the relationship between cybersecurity and science in detail; I will argue that cybersecurity as practised is a kind of science.

	Criteria				
Document	1	2	3	4	5
MITRE (2010)		✓		✓	✓
Mundie and Ruefle (2012)		✓	✓		✓
Fenz and Ekelhart (2009)		✓		✓	✓
Osorno et al. (2011)	✓	✓	✓	✓	✓
Magklaras and Furnell (2001)	✓		✓	✓	✓
Wang and Guo (2009)	✓			✓	✓
Chiang et al. (2009)	✓			✓	✓
Ekelhart et al. (2007)	✓		✓	✓	✓
Kossakowski et al. (1999)	✓	✓	✓	✓	✓

Table 2.10: Documents referenced by CERT/CC documents. The criteria are (1) target audience is security professionals; (2) topic in scope (Section 2.2.1); (3) focus is investigator practices; (4) document final and not obsolete as of Aug 1, 2017; (5) available in English.

Fenz and Ekelhart (2009) provides a difficult decision. It is one of the few attempts at formalization in the literature. However, its target is to formalize security knowledge, not security practice. This topic is closely allied to my hope to formalize CSIR analysis, as security knowledge would be instrumental to that project. So while not in scope for this review, this document may be useful for future related work.

2.5.5.4 *IC Documents*

Heuer (1999) presents some challenges to adequate reference harvesting. The book contains no collected list of references, written in a traditional humanities style in which references are in footnotes intermixed with commentary, but this is not the central problem. As essentially a military intelligence and psychology book, its sources are quite wide-ranging. References range from World War II Nazi-propaganda analysis to behavioural economics. It is only through Heuer's CIA experience that these disparate sources are converted into a useful guide on how to reason in adversarial situations. The other challenge is that Heuer (1999) makes only passing reference to computers as tabulating machines. The closest he seems to get to computer science is via Simon (1996), as he discusses decision-making and satisficing. For these three reasons I consider Heuer (1999) as essentially a primary source and do not trace citations from it. One should not be surprised it has many features of a primary source, as

surely its main value is summarizing CIA analytic experience not otherwise publicly available.

	Criteria				
Document	1	2	3	4	5
Stamos (2010)	✓			✓	✓
Amann et al. (2012)	✓			✓	✓
Cheswick (1992)	✓	✓	✓	✓	✓
Stoll (1988)	✓	✓	✓	✓	✓
Duran et al. (2009)	✓			✓	✓
Cohen (1995)	✓			✓	✓
Lewis (2008)				✓	✓
Tirpak (2000)		✓	✓	✓	✓
Hayes (2008)	✓			✓	✓
Willison and Siponen (2009)	✓	✓		✓	✓
Mitropoulos et al. (2006)	✓	✓	✓	✓	✓
Caltagirone and Frincke (2005)	✓		✓	✓	✓
Caltagirone (2005)	✓		✓	✓	✓
Bellovin (1992)	✓	✓		✓	✓
McClure et al. (2005)	✓		✓	✓	✓
Brenner (2002)				✓	✓
Van Eck (1985)	✓			✓	✓
John and Olovsson (2008)	✓		✓	✓	✓
Joint Chiefs of Staff (2014a)	✓			*	✓
Joint Chiefs of Staff (2013)	✓		✓	*	✓
Joint Chiefs of Staff (2014c, ch. 5 only)	✓	✓	✓	*	✓

Table 2.11: Documents referenced by Caltagirone et al. (2013) and Hutchins et al. (2011). The criteria are (1) target audience is security professionals; (2) topic in scope, per Section 2.2.1; (3) focus is investigator practices; (4) document finalized and not obsoleted as of Aug 1, 2017; (5) available in English. A (*) indicates the IC document cited the document current at the time, but it has since been updated and I cite the document current as of Aug 1, 2017.

Caltagirone et al. (2013) and Hutchins et al. (2011) are more straightforward. There are 79 citations between the two, with no overlap, though Caltagirone et al. (2013) cites both Hutchins et al. (2011) and Heuer (1999). The references in Caltagirone et al. (2013) are noticeably more strategy-focused over the tactically-focused Hutchins et al. (2011), as one would expect from their different topics. Hutchins et al. (2011) cites several vulnerability bulletins and company advisories as cases; it is more of a primary source, documenting

the analysis methods used by Lockheed Martin incident response staff. I do not consider such advisories, software tool documentation, and news items, as they are not within review scope. There are also several references already covered elsewhere: Cichonski et al. (2012), STIX, CVE, and SANS Institute. There is yet a new reporting and data exchange format: Vocabulary for Event Recording and Incident Sharing (VERIS). References that are definitions of terminology are excluded. These exceptions reduce the total referenced works to 50.

The kill chain (Hutchins et al., 2011) and the diamond model (Caltagirone et al., 2013) are both attack ontologies. They model the possible routes an adversary may take when executing an attack. I discuss attack ontologies as a kind of related foundational work in Section 2.8.2. One big class of cited work is other attack ontologies. I consider the kill chain and diamond model as *de facto* standard ontologies, but they likely reached that level of agreement within the IC because they also come with investigative norms for interpreting and filling in the ontologies. The referenced attack ontology works do not come with such guidance, and so I will not consider them directly in scope here. This decision removes 16 references from consideration because they are attack-ontology related.

There are also several references that are clearly for background or motivation, such as Hawkins (2004), Liu and Motoda (1998), Symantec's analysis of the Duqu malware, and assessments of Chinese attack capabilities. I removed how-to descriptions for conducting particular methods of technical analysis, namely on passive DNS analysis (Antonakakis et al., 2011), crime-pattern analysis (Palmiotto, 1988), and honeypots via the Honeynet Project. This leaves the 22 documents in Table 2.11. Of these, four pass all relevance requirements: Cheswick (1992),[21] Stoll (1988), Mitropoulos et al. (2006), and Joint Chiefs of Staff (2014c, ch. 5 only).

2.6 DISCUSSION

Given the purported importance of reliable incident response in the standards literature, it is rather surprising there is such a gap, without systematic advice for CSIR analysis on what questions to hypothesize and prioritize or how to reliably collect and interpret evidence. Table 2.12 classifies advice on CSIR

21 Technically the citation is to this article's republication in a popular textbook, Cheswick et al. (2003, ch. 16). However, as the rest of the textbook is not directly referenced, I reference just the original publication.

topics in several ways: to which phases the document applies, the directness with which the document applies to each phase, the applicability of the advice, investigative goals supported, broadness of scope, generalizability of advice, and formalism. This section is organised to make some commentary on all the documents in the Table.

This section begins with an analyst of what I view to be the primary gap in the literature. The evocative evidence for this claim will be the three NIST publications, as they make the point most clearly. After discussing these, the section moves on to the ISO, IETF, etc. documents in the order they appear in Table 2.12.

Despite recommending everyone use a methodical approach, NIST fails to provide one. This failure is symptomatic of the state of available practicable policy advice and practitioner training material. This is the central gap identified by the literature review: There may be adequate concrete tools and training available, but there is no general training or policy advice for strategic selection of tactics, that is, **which analysis heuristics or technical tools to employ in a particular situation and why**. A related gap is a failure to advise on **when the analyst or investigator is justified in generalizing**; that is, making a stronger, broader claim from singular pieces of evidence. Because there is no advice on which strategy to employ, or when broadening claims are justified, there is similarly a gap in **what information to report in order to convince a reader**.

Examining NIST SP 800-61 (Cichonski et al., 2012), it is obvious why all four standards venues reference it or use it as their standard. It is comprehensive and thorough without being overbearing. However, its focus is incident management, not investigation. The analysis phase receives about three pages of discussion (p 28-30), reporting one page (p 31),[22] evidence collection half a page (p 36), and general decision making and prioritization two pages (p 32-33). Cichonski et al. (2012, p 32) addresses the problem of scarce resources directly: "prioritizing the handling of the incident is perhaps the most critical decision point in the incident handling process." This is probably the best discussion of these topics in the documents found. Yet it is far from sufficient for a robust account of the nuances and difficulties of evaluating evidence, generalizing from particulars, and deciding how best to report findings.

22 SP 800-61 acknowledges its discussion of reporting is too brief, and refers the reader to RFC 5070. This document has since been obsoleted by RFC 7970 (Danyliw, 2016).

NIST SP 800-83 (Souppaya and Scarfone, 2013) §4.2 is titled "detection and analysis," yet I have labelled the document as having no bearing on the analysis phase. This decision is because the section's advice on analysis is entirely tool-focused pragmatics. For example, analysis should take place on an isolated or virtualized operating system to prevent spread of infection, and so on. The document mentions some fields that the investigator may want to collect, such as file names, service ports, and "how to remove the malware." There is no advice on how to obtain this information, why, or what it might be useful for. Therefore, these are best understood as reporting constraints, not analysis advice. This result is disappointing, considering §4 gives its opening motivation as "this section of the guide builds on the concepts of SP 800-61 by providing additional details about responding to malware incidents."

NIST SP 800-86 Kent et al. (2006) suffers similarly to SP 800-83; it consists of a stream of data formats and types and assumes what to do is known. These make up underlying technical skills necessary for an investigation, and so are not completely irrelevant to incident investigation. However, they do not help us understand how investigators make decisions. Advice on analysis again amounts to essentially constraints on reporting and collection: "the analysis should include identifying people, places, items, and events, and determining how these elements are related... often, this effort will include correlating data among multiple sources" (Kent et al., 2006, p 3-6).

These NIST documents seem to acknowledge at least a shadow of the gaps I identify, even if they do not fill it. Kent et al. (2006, p 3-8) recommends all organizations have an incident response capability and that analysts use "a methodical approach to a digital forensic investigation." The document clearly states the importance of digital forensic investigation and advises on terminology, analysis techniques, and pitfalls to avoid. Despite NIST's policy recommendation to do so, based on this survey NIST does not actually provide a methodical approach to analysing data during an investigation. SP 800-61 comes closest, but the discussion of analysis method there still amounts to an unordered collection of tips, tricks, and pitfalls to avoid. While these are all accurate and sound advice, they do not comprise a method. The NIST documents evaluated and surveyed here are one significant effort at providing practical advice to a wide audience on a complex topic. Their focus is practical guides on aspects of tools used in digital forensics. But the assumption is that once an analyst or investigator is taught how to use a tool, they will know

when and why to use it. This gap is a recurring assumption, and precisely my intended focus for improvement.

Many of those people writing these documents or making these tools are competent in CSIR analysis already, and so is the intended audience of their colleagues. Therefore, I imagine they do not write down some process that seems common sense to them. They seem especially likely to do so when it is expedient: security professionals tend to be under continuous time pressure. Skipping this step is doubly attractive if the intended audience seems to know already and there are more pressing issues. Finally, computing is known for a sort of rugged individualism as a profession (Ensmenger, 2015) that prefers self-taught struggle over asking for explicit details. Based on expediency, rather than slowing down the whole standards process, those who did manage to ask such a question were likely tutored privately – apprenticed in the trade-craft – either in formal training or informally. Finally, those offering formal training at high cost have little incentive to publicize their work.

Document	Directness by Phase			Scope by Goal			Gen	Type	Formal
	Col	Anl	Rep	Fix	Int	LE			
27035-1:2016 (ISO/IEC, 2016)	C	C	C	B			Un	Ont	Qual
27037 ISO/IEC (2012)	D					M	Un	Ont	Qual
27041 ISO/IEC (2015a)	C	C	C				Likely	Instr	∅
27042 ISO/IEC (2015b)		D	D			M	Un	Instr	∅
27043 ISO/IEC (2015c)	C	C	C	B		M	Un	Ont	Qual
RFC 2196, §5.4 only (Fraser, 1997)	C		D	M		B	Likely	Instr	∅
RFC 6545 (Moriarty, 2012)	C	C	D	N	B		Un	Ont	Formal
RFC 7203 (Takahashi et al., 2014)	C		C	N	N	N	Un	Ont	Formal
RFC 7970 (Danyliw, 2016)	C		D	M	M	M	Un	Ont	Formal
RFC 8134 (2017)	C			B	B		Un	Study	∅
NIST 800-61 (Cichonski et al., 2012)	C	D	C	M		M	Un	Adv	Qual
NIST 800-83 §4 (2013)	D		C	B			Un	Ont	Qual
NIST 800-86 (Kent et al., 2006)	C		C				High	Study	Qual
Gorzelak et al. (2011) (ENISA)	D				N		Un	Study	Qual
Alberts et al. (2004) (CERT/CC)			C	B			Un	Ont	Qual
Kossakowski et al. (1999) (CERT/CC)	C	D	C	M			Likely	Adv	Qual
Mundie et al. (2014) (CERT/CC)	C	C	C	B			High	Ont	Perf
Osorno et al. (2011) (US-CERT)		C	C	B	B		High	Ont	Qual
Hutchins et al. (2011) (IC)	C	C			M		Un	Adv	Qual
Caltagirone et al. (2013) (IC)	C	D			M		High	Adv	Perf
Heuer (1999) (CIA)		D		B	B	B	Wide	Instr	Qual
Joint Chiefs of Staff (2014c, ch. 5)		D			M		High	Instr	Qual
Casey (2010, ch. 2)	C	D	D			B	Wide	Ont	Qual
Mitropoulos et al. (2006)	C	C	C	M	N		Un	Study	Qual
Carrier and Spafford (2004)	D	C	D			M	Likely	Ont	Qual
Ciardhuáin (2004)	C	C	C			B	Un	Ont	Perf
Leigland and Krings (2004)	D			N	N	N	Likely	Instr	Formal
Stoll (1988)	C	D	C	N	M	N	Likely	Study	∅
Cheswick (1992)	C	D			B		Likely	Study	∅

Table 2.12: Categorization of relevant documents. Phases are collection, analysis, and reporting; a cross means phase is not addressed. Advice directness is direct (D) or constraints-based (C). Goals are: fix an infected system (fix), gather intelligence (int), and law enforcement action (LE). A document's intended scope is narrow (N), medium (M), or broad (B). The generalizability (Gen) of an approach is unlikely (Un), likely (Likely), highly likely (High), or already widely generalizable (Wide). Document types are case studies (Study), ontologies (Ont), advice on actions (Adv), or explicit instructions (Instr). Formalization is either not present (∅), qualitative (Qual), formal, or perfunctory (Perf).

2.6.1 *Comments and clarifications on Table 2.12*

Table 2.12 is complicated, and requires some unpacking. The following items explain some of my classification decisions captured in the table.

THE ISO 27000-SERIES is dedicated to information security. Five identified standards on information security are within the scope on the particular parts of CSIR. The relationship between these standards is documented by their Figure 1, reproduced in each of the ISO standards (which, due to copyrights, I cannot reproduce here). This figure states clearly that all the listed standards are applicable to "investigation process classes and activities" (ISO/IEC, 2015c, p. ix). Process classes are readiness, initialization, acquisitive, and investigative; activities overlap these classes, and are plan, prepare, respond, identify-collect-acquire-preserve, understand, report, and close. This taxonomy is essentially consistent with the taxonomies used by the IETF, NIST, and FIRST (Mundie et al., 2014).

However, where a NIST standard such as SP 800-61 is a single 70-page document, the ISO incident response standards are each 10-15 pages of unique content with 10-15 pages that are repeated in each document. The five ISO documents combined are comparable in scope and detail to SP 800-61. Unlike a NIST publication, the ISO documents do not present a clear investigative goal (e.g., fix the system, legal prosecution, etc.), even within documents, let alone among them.

27043, for example, alternates between incident response for fixing systems and analysis for providing evidence to a legal proceeding. Within 27043 ISO/IEC (2015c), §8 reads like advice from CERT/CC (Alberts et al., 2004), and §9 reads like advice from Casey (2010, ch. 2). The shift is abrupt and without explanation. The shift includes a shift in terminology and jargon for referring to essentially the same mental process by the investigator. This oddity does not build confidence that the ISO standards actually present a unified methodology for incident investigation as a series of disconnected vignettes.

27041 does little to dispel this sense of disconnectedness. This ISO document is disconnected from the other incident management documents in that it focuses on the client-contractor relationship. A process is validated in that "the work instruction fulfils the requirements agreed with the

client" (ISO/IEC, 2015a, p 9). 27041 ISO/IEC (2015a, p 12-13) states an investigation composed of validated examinations "can be considered to be validated" while defining a validated examination as one mode up of validated processes. Assuming composability of valid processes is a dangerous claim. Concurrent program verification has shown such claims cannot be assumed and are challenging to prove (Milner, 1989; O'Hearn, 2007); even though the technical sense of "valid" is slightly different.

For these reasons, the ISO standards would struggle to function well as a unified whole. There does not seem to be overarching editorial guidance to assure consistency or navigate conflicts. At best, if the reader already knows how to navigate the different, conflicting contexts, the ISO documents are useful expressions of each area of concern. The level of detail is appropriate for ensuring management ability to oversee a process, rather than to do the process itself. Even the most specific documents (27037 and 27042), to which the other, more general documents refer for details, are thin on anything that might help with actual decision-making. ISO/IEC 27042 provides a basic distinction between static and dynamic analysis of malware (it uses "live" for dynamic), but all that is really provided are a few descriptions of what distinguishes static and dynamic analysis. These descriptions do not provide information on how to actually do either kind of analysis, or even common pitfalls or errors to avoid.

RFC 2196 is quite old, and its advice shows its age. The steps are in general sound; however, they are from a time when it was reasonable to ask for "all system events (audit records)" to be recorded and evaluated by the investigator (Fraser, 1997, p 54). The text assumes that incident investigators will know what to do with these events once logged. This advice is not bad, such as it is; however it is best understood as historical rather than actionable advice.

RFC 6545 and RFC 6546 jointly detail Real-time Inter-network Defense (RID). RFC 6545 describes conceptual and formal details, whereas RFC 6546 provides technical communication and encryption details. RFC 6545 is an extension of IODEF (Danyliw, 2016), specifying methods and norms of communication using IODEF between organizations. As such, the document focuses on what to report, and how to use reports for mitigation. Policy of use and sensitivity of information is explicitly

integrated into the format. How analysis produces adequate data is out of scope. By providing such explicit standards on what should be reported and how those reports can expect to be used, RID does put constraints on analysis and evidence collection – those phases need to produce reporting with the specified types of fields.

RFC 7203 extends IODEF (Danyliw, 2016) "to embed and convey various types of structured information" Takahashi et al. (2014, p 2). Specifically, the various metrics and formats such as CVSS and CVE listed in Table 2.7. This extension serves to integrate two types of reporting format and constraint. This is useful, but is mostly programmatic. Therefore, it is not directly about reporting in the same way as RFC 7970. Although technically detailed, from a decision-making point of view RFC 7203 just suggests that these metrics are useful ways to describe an incident and report on it, and that investigators should do so. RFC 5901 makes similar suggestions specifically for reporting phishing (Cain and Jevans, 2010).

RFC 7970 is the heart of the IETF incident analysis standardization effort. It obsoletes RFC 5070, which is cited or used by most publication venues as the incident reporting format. The focus is on exchanging indicators of incidents for collective defence. Although IODEF is, strictly speaking, just an XML schema for document incidents, the available options and the ontology provided constrain the other phases up to reporting. For some fields, this provides only minimal collection requirements. However, consider the system impact type attribute, which is a required field. There are 24 options specified, ranging from "takeover-account" to "integrity-hardware" (Danyliw, 2016, p 46). Differentiating these various impacts would require a relatively sophisticated incident investigation and analysis capability; it is not so easy as logging an IP address and passing it along. Just within the assessment class, one of two dozen overarching classes, there are five types of impact to distinguish between with similar detail: system, business, time, money, and confidence. Such detail provides the most rigorous reporting requirements and guidance available.

RFC 8134 is informational, and not a standard. It provides a list of information exchanges, collaborations, and implementations that make use of IODEF. Because information exchanges are a source of evidence collection, the details about what information is available from what groups

provides evidence collection suggestions and introductions. Although this advice is at a rather abstract level, it is useful because it provides a discussion of network defence information sharing arrangements that is not commonly quite so public.

GORZELAK ET AL. (2011) is a study commissioned by ENISA and executed by the Polish CERT. The focus is on data sources – how do CSIRTs monitor their constituents. The method employed is a survey of over 100 CSIRTs. While this data is at best instructive of where to get data, it is an important resource for how respondents evaluate the quality of data sources. Such evaluation is directly relevant to evidence collection decisions. It is unlikely this study is instructive outside this relatively narrow context. However, it is directly relevant context for this work.

ALBERTS ET AL. (2004) is primarily about contextualizing incident management within a wider organizational context. In fact, Alberts et al. (2004, pp 24-26) is one of the best assessments of the relationship of investigation to preparation and protection from this review. However, my scope is narrower than its topic, namely how to situate the CSIR process within an organization. Alberts et al. (2004, p 128ff.) is an ambitious effort to organize a flow chart for incident response. Because their scope includes technical, management, and legal responses, the level of detail devoted to analysis amounts to "[d]esignated personnel analyze each event and plan, coordinate, and execute the appropriate technical response" (Alberts et al., 2004, p 136).

KOSSAKOWSKI ET AL. (1999, P 17FF.) provides classes of advice, like collect logs and isolate infected machines from the network. These perhaps come the closest to advice about how to collect evidence from computer incidents. However, it is silent on which logs to collect, or what to look for when examining network traces. While this advice is highly likely to be able to generalize to all cases of incident investigation, the level of detail is not operationalizable as decision-making instructions.

MUNDIE ET AL. (2014) is, in effect, a literature review of incident management. As such, it mostly constrains the inputs and outputs one would expect from incident investigation. The formalism provided is in a specification of an OWL ontology language of incident management. This language is a useful step in reconciling various incident management

processes. However, it is a few levels of abstraction above the current task.

Osorno et al. (2011) has done something similar to the scope here, in that they inventory various incident management processes, with two main differences. First, they focus on moving up a level to inter-organizational coordination during complicated incidents, rather than zooming in on individual analyst decision-making. Second, they focus on the US context. This different purpose leads to substantial differences in emphasis as to what is reviewed; for example, where this review has generally set aside data exchange formats (see Section 2.5.5.1), Osorno et al. (2011) spend considerable effort mapping these formats into each other. For this reason, the extent of their recommendation on incident investigation amounts to do an OODA-style loop (p 7).[23]

Hutchins et al. (2011) discusses courses of action for network defence. These are not direct advice on incident investigation steps. The level of advice is on the order of "to disrupt installation events, use anti-virus." This advice is sound, but it is not particularly concrete. However, matching this advice with an attack model (and perhaps a catchy name) has meant that the kill chain model of intrusion analysis proposed by Hutchins et al. (2011) has gained some traction outside the IC.

Caltagirone et al. (2013) provides some light formalization of their qualitative categories into both graph theory and subject probabilities and Bayesian statistics. 'Perfunctory' formalization indicates the extent of their documentation is to list the formal structures that are equally well-described by their prose. The document does not make any use of the formalism, and it is not central to their arguments. The structures appear to be constructed adequately; it is simply that any application of them is left as an exercise for the reader in much the same way as if they were not there.

Heuer (1999) is an instructive case for some advice being too broad for CSIR purposes. The book is comprised of explicit decision-making instructions for analysis of intelligence. However, the level of abstraction is so broad that it can be argued to be applicable to almost any adversarial decision-making environment. So while it is valuable, and it provides instructions for avoiding cognitive biases, it does not provide

23 OODA is a military decision-making term standing for observe, orient, decide, and act.

instructions at the level of detail that are directly useful for an investigation.

Joint Chiefs of Staff (2014c, ch. 5) is about how to think like your adversary. It is an extended treatment of developing and evaluating adversary action plans across multiple dimensions under constrained resources. The basic cycle is to identify objectives, enumerate courses of action, evaluate and rank the likelihood of following each action, and identify necessary intelligence collection requirements to determine adversary decisions. The document is about military intelligence operations generally, not computer-security incidents. However, one narrow but necessary aspect of any investigation is how to anticipate an adversary. This document covers the thought process behind the topic of anticipation in a way which should be easily applicable to computer security.

Casey (2010, ch. 2) is built around the claim that digital forensics is just another kind of scientific investigation. The basic ontology of the scientific method is represented simply as create and evaluate hypotheses dispassionately based on evidence. This description is supported by several case study examples working through the method. This pedagogical strategy does not quite amount to 'advice on decision making', but it is also more than merely an ontology. The end of the chapter includes advice on how to report conclusions convincingly to a jury or prosecuting attorney. The advice is simple, but direct and effective: be concise, let the evidence speak for itself, do not jump to conclusions, summarize key results up front. The target audience is law enforcement who will be using information technology to support general legal cases, not computer crimes. Despite this broad audience, the treatment of decision-making as part of the scientific method allows for easy and broad re-purposing for other scenarios more directly related to computer-security incidents.

Mitropoulos et al. (2006) is relevant to CSIR analysis for its series of small case studies on how to analyze different types of network logs. They provide a flow chart for analysis of IDS logs, intended to serve as an example of analysis of a particular kind of technical information. Although only represented at a relatively abstract level, this is a representation of reasoning and decision making during incident response. They also provide basic instructions on how to gather intelligence data on adversaries using various network protocols. While I have ranked

this as unlikely to generalize, it is also very specifically targeted to techniques that are commonly useful during incident investigation. So it is not particularly necessary that they are generalizable. These sorts of cases are difficult to teach and capture in a more abstract form, and are inflexible, and so this specificity has a cost. Mitropoulos et al. (2006) do not advise in which scenarios to employ these different types of forensic techniques.

Carrier and Spafford (2004) describes a evidence collection and hypothesis testing and reporting model for digital forensic investigation. They use forensic in its formal legal sense, and so target specifically the gathering of adequate legal evidence. Their treatment of analysis is particularly brief; it is rolled into the evidence collection process based on whether the data element is relevant to the defined target. As to target definition, it "is the most challenging of the search phase" and is done "from either experience or existing evidence" (Carrier and Spafford, 2004, p 8). Like the other documents, this one sidesteps most of the hard work on which I will focus.

Ciardhuáin (2004) attempts a novel formalization by incorporating information flow; although not cited, this is a term likely taken from Barwise and Seligman (1997). However, the application to forensic investigation (Ciardhuáin, 2004, p 21) bears little resemblance to formal information flow models. The discussion places vague constraints on evidence reporting such as "the investigators must construct a hypothesis of what occurred" and "The hypothesis must be presented to persons other than the investigators" (Ciardhuáin, 2004, p 7). But in practice these are of little use in making decisions during an investigation.

Leigland and Krings (2004) provides a formal specification of evidence collection methods that can be adapted to specific operating systems. The language associates collection goals with certain common attack patterns. The goal is narrowly practical: to speed collection of evidence by technicians during an investigation while reducing superfluous data collection to make analysis a bit easier. The language maps general actions to specific operating-system commands. The technician needs to specify file identifiers for specific attack campaigns; the main downfall of this method is that adversaries have learned to randomize certain identifiers within their attacks. Such randomization makes

keeping an adequate library of definitions for this language as defined essentially impossible.

2.6.2 *Note on case studies*

Case studies or collections of cases that have been analysed by others provide demonstrations of what sort of attacks are possible. Two of the earliest examples of this style of reporting are Stoll (1988) and Cheswick (1992). A survey of incident case studies may be a useful additional project. The practitioners who wrote these standards documents would be aware that many security vendors publish accounts of adversaries they have tracked in the course of their work. These are of varying quality, scope and importance. More recent impactful studies include Mandiant tracking an alleged unit of the Chinese military (Mandiant, 2013) and Google's self-report of compromise attributed to China (Drummond, 2010). Some case reports are official government commissioned, such as the Black Tulip report analysing the compromise of a TLS certificate authority critical to the operation of many Dutch government websites (Hoogstraaten, 2012).

The scope need not be an individual case. Some studies focus on trends rather than individual cases. Verizon's Data Breach Investigation Report is probably the best-known example (see, e.g., Verizon (2015, 2016)). United States federal civilian agencies must make annual breach reports to Congress per FISMA requirements; such detailed reports have been examined for trends (Pang and Tanriverdi, 2017).

One notable change in this style of report since Stoll (1988) and Cheswick (1992) is a trend away from discussing how exactly the investigators found what they found. In an environment where adversaries are likely to read any reports their targets publish, this shift towards withholding information is understandable. Paradoxically, this makes the old case studies more valuable, as they remain some of the better expressions of the investigator's thought process. Of course, the tools and networks the old case studies discuss are almost entirely irrelevant, which can make them hard to apply to today's systems. And the case studies do not do any of the work to make the necessary generalizations. Many of the modern expressions of the form of CSIR analysis cycles, as documented in this survey, are quite consistent with the mental process Stoll (1988) and Cheswick (1992) describe.

2.7 GAPS AND WORK PLAN

Section 2.6 suggests three primary gaps in the incident investigation standards around decision-making:

- strategic selection of tactics, that is, which analysis heuristic or technical tool to employ in a particular situation and why
- when the investigator is justified in generalizing; that is, making a stronger, broader claim from singular pieces of evidence
- what information to report and how to communicate it in order to convince someone that the investigator should be believed

2.7.1 *Research question*

In order to make progress towards providing decision-making in these gaps, I propose to examine the following research questions:

How to satisfactorily make clear and explicit the reasoning process used by individual CSIR analysts while they pursue technical details?

This breaks down into three component questions:

- What heuristics should an incident analyst use to construct general knowledge and analyse attacks?
- Is it possible to construct formal tools that enable automated decision support for the analyst with such heuristics and knowledge?
- How can an incident analyst apply general knowledge to improve analysis of specific incidents?

My answer to these questions will be within two expectations and scope limitations:

- It must include how to decide what information to report as results, but excluding decisions about actions in response (remediation, legal action, policy changes, etc.).
- The description and/or formalization of the reasoning process should support the evidence collection, analysis, and reporting phases of CSIR. Within each of these three phases, each of the three gaps need to be addressed. For example, what information to report is not just a problem for reporting, it is also a problem when one completes the evidence collection phase and needs to report results to the analysis phase.

2.8 RELATED WORK

Because standards are slow to evolve, one might expect more advanced work in related academic fields. This section will suggestively map out fields where one might expect to find related work, even though the result of this mapping will be that little of the work is relevant. Specifically, I will sketch the intersection between cybersecurity and the fields of economics, game theory, statistics, machine learning, modal logic, information theory, risk assessment, and systems engineering. Each of these fields has an established interdisciplinary overlap with security.

There is also potentially related work within computer science from fields that are allied with parts of cybersecurity. The subfields include attack ontologies, vulnerability analysis, intrusion detection, internet measurement, and science of security.

The discussion of related work in these fields is suggestive rather than exhaustive. A comprehensive review to codify the discussion could be done as future work, but probably would provide little additional insight. The results of the review in Section 2.5.5 indicate the standards do not integrate work from academic fields outside cybersecurity. The cited academic work is limited to digital forensics ontologies, attack ontologies, and classic case studies from around 1990. Better understanding of why practitioners seem to integrate so little academic work into these standards is perhaps an interesting sociology of science or science communication research question for future work. One probable reason, among others, is that available academic discussions are not at the correct level of abstraction to be useful in investigative decision-making.

The available levels of abstraction – attack ontologies, economics, etc. – have continually left gaps. My basic strategy is step away a bit further, to a higher level of abstraction about problem solving and decision making. From this perspective, the available tools can perhaps be adapted to fill the gaps.

The goal of this section is to identify areas that could be most easily adapted to the incident investigation context. There are likely useful lessons to be taken from all of these fields. The challenge is how to synthesize these lessons at a level of abstraction that is actionable during incident investigation. In general, none of the fields in this section can adequately inform the gaps of knowledge generation or reasoning in CSIR. I have left out of this section the two fields that I do find promising and will pursue through the rest of the

thesis. These are philosophy of science (see Chapter 3 and Section 4.2) and program verification (see Chapter 6).

2.8.1 *Interdisciplinary overlaps*

ECONOMICS does not tend to study individual decision making, but it does study average decision making of larger groups. The economics of information security, as its own field of study, is generally seen as initiated by Anderson (2001). At a minimum, economics adds constraints in two ways: what incident responders can do during an investigation, and expected reasonable motivations and economically-viable attacks of adversaries.

Economics has also been used in delimiting useful interventions to deter attacks. For example, Kanich et al. (2011) identify credit card payment processors as the main choke-point for generation of revenue from spam soliciting illicit pharmaceutical sales.

GAME THEORY began as a sub-field of economics; however, it has evolved into its own area of mathematical study of adversarial decision making. A game is any situation in which two or more intelligent decision makers knowingly compete. Game theory became famous around the question of how nuclear-armed superpowers could rationally interact without destroying the world (Ross, 2018).

Game theory seems a reasonable place to search for standards to guide decision making during incident investigation. There are well-developed theories for game theory and security, for example used to decide patrol schedules at Los Angeles Airport (Tambe, 2011).

Alpcan and Başar (2011) specifically focuses on computer-network security, including attack detection, optimal allocation of security investments, and joint responses to malware epidemics. These are all related to incident management – detection, preparation, and recovery, respectively. These are all management topics, not analysis topics, that were placed out of scope for this review.

More recent work has developed these topics further. For example, Anderson et al. (2016) takes a more realistic, qualitative preference-based view of defender investment priorities. Spring (2014) highlights the complexity of the relationship between adversary, victim, and system-owner is likely not reducible to a two-player game. These topics are refinements of those followed by

Alpcan and Başar (2011). The proceedings of WEIS, the pre-eminent venue for game theory and computer security, also focus on management and resource allocation questions, true to the focus on the economics of information security. Game theory provides rather well-developed support of decisions about allocating resources to parts of the incident management cycle, but is silent on how to make decisions about evaluating evidence during an incident investigation.

STATISTICS is often used to express confidence or belief in what aspects of an event have occurred. Although they do not use the formalism, Caltagirone et al. (2013) specifically provide a field for Bayesian statistics (Kadane, 2011) within their analysis model. Of course, experiment design is itself intimately tied to design of statistical methods (Fisher, 1971). With a focus on causal analysis, Pearl's work has had a big impact (Pearl, 2009). Causal modelling has been applied to software engineering problems (Letier et al., 2014), but not security. Any application would likely require much thought into what exactly the causal relation represents, as Pearl's model is not without valid critics (Dawid, 2010). Computational complexity also presents a challenge at scale (Chockler and Halpern, 2004). Scalability is necessary for viable proof theories and models in computer science, as I will demonstrate in Chapter 6.

MACHINE LEARNING uses statistical models to classify objects or identify trends. The field's integration with incident management is advanced. Darktrace sells a security product that is designed to be hands-off by the analyst and to simply develop and deploy defences according to its statistical models (Palmer, 2016). The main challenge is that my goal is to develop a model of decision-making. Machine learning makes decisions, but how and why something is or is not malicious is generally opaque. This technique works well when outcomes are clearly definable, success and failure are clear, and scope is clearly defined. However, incident investigation is full of yet-to-be delineated outcomes, poorly defined success criteria, and problems with indeterminate scope. Therefore, while an important tool, machine learning is not the solution to the research question.[24]

MODAL LOGIC is a branch of formal logic that includes modal operators such as necessity, being informed, and so on. As investigators need to

24 My thinking on this point was clarified by a talk by Heather Douglas on "How the public can assess expertise" at UCL on Jan 31, 2018. See https://www.youtube.com/watch?v=cuBo6iZ8-sM.

track what the adversary knows, it seems like an attractive formalization of such ideas. Ditmarsch et al. (2015) provides good coverage of such epistemic issues, and Floridi (2011) develops a logic of information intended to capture similar relations including machines as well as humans. Anderson and Pym (2015) adapts such modal structures to reasoning under and about constrained resources.

INFORMATION THEORY and the sub-area information flow are referenced in passing within the reviewed documents. Process theory (Milner, 1989) does indeed seem to be a promising area for defining decision processes around state changes, which is related to my research question. However, it would take significant effort to take a theory intended to reason about atomic operations inside a computer and bring it to bear on human decisions. If the atomic units of process theory are too small, atomic units in other works are too big or underspecified. A logic of distributed systems (Barwise and Seligman, 1997) and information-based logic (Devlin, 1995) both work primarily with information in an abstract sense, and the work to bring it to bear on human decisions during incident investigation seems likewise huge. If anyone has attempted to make such information logic more concrete, it is Floridi (2011), and the attempt remains a long way away from my practical concerns.

RISK ASSESSMENT is about determining how much one should spend on a project or various protections based on measured expectation of loss. Quantitative risk assessment works well in situations such as determining how much rain to prepare for when building a dam. The canonical reference for risk assessment methods comes from the Dutch government (Uijt de Haag and Ale, 1999).

Quantitative risk assessment is closely related to decision theory. Decision theory is mathematically similar to game theory but with the modified interpretation that only one player is actually making decisions (anything else is Nature).

In principle, these fields could provide decision support to incident investigation. However, the detailed information necessary to accurately calculate expected impacts is not generally available. This statistical approach works for building dams based on 150 years of reliable data on water levels coupled with a justified expectation that this past weather is reasonably representative of future weather. Incident responders have neither this wealth of historical

data nor the naive expectation that past adversary behaviour can be at all representative of future behaviour. Past adversary tactics do inform expectations of future behaviour, but not in such a simple 1-to-1 manner as with the weather that would justify simple statistical projection into the future (Spring and Stoner, 2015).

SYSTEMS ENGINEERING includes adequate incident management as an essential part of engineering secure systems – responding to incidents is an important impetus for modifications to systems (Ross et al., 2016, p. 3). In general, when designing a system it is important to know what attacks it will plausibly face. Incident management supplies this data on attacks. This informs what reporting should contain, insofar as it defines a common use of reporting. Part OP-3 of Ross et al. (2016, p. 142ff) gives high-level guidance that security anomalies need to be measured, tracked, and mitigated. The references for how this is to be done are ISO and NIST documents on incident management, which Section 2.6 notes are inadequate for CSIR analysis purposes.

Incident responders also benefit from a knowledge of security engineering. The canonical work by Anderson (2008), for example, sets expectations about common system design errors that might be the weak link exploited by an adversary in an incident. Such information is broadly useful for setting expectations of incident investigators, but does not provide concrete decision support.

2.8.2 *Parts of information and computer science*

ATTACK ONTOLOGIES are long-standing line of research within security that aims to understand attacker behaviours. Attack ontologies break down into several sub-categories, such as attack graphs, "a succinct representation of all paths through a system that end in a state where an intruder has successfully achieved his goal" (Jha et al., 2002, p. 1). Lippmann and Ingols (2005) provides an early, canonical review. A canonical starting point is Schneier (1999).

More broadly than attack graphs, ontologies of attacker methods abound (Bejtlich, 2004; Howard and Longstaff, 1998). These are certainly useful, and should inform incident response decision making. More recent efforts have included improved visualization of such graphs (Homer et al., 2008). The

kill chain model (Hutchins et al., 2011), discussed as a standard view within the intelligence community, is one specific type of attack ontology. There are clear influences on the *de facto* standard ontologies. For example, the diamond model aspect of Caltagirone et al. (2013) comes from Obrst et al. (2012).

At a higher level of abstraction than attack graphs, Spring et al. (2015) models the development of capability by the global community of adversaries against types of systems.

VULNERABILITY ANALYSIS includes several sub-parts, such as a standardized process for listing vulnerabilities (MITRE, 2012), assessing severity of vulnerabilities (Abraham et al., 2015; Scarfone and Mell, 2009), enumerating software coding weaknesses (MITRE, 2015), and secure coding practices to avoid all these problems (Seacord, 2005). Tools that scan a computer network for vulnerable or misconfigured hosts are free (Lyon, 2011). Tools for finding vulnerabilities in software via formal methods are in industrial application (Newcombe et al., 2015; O'Hearn, 2015), including free ones (Calcagno et al., 2015b). Vulnerability discovery can also be performed via black-box testing, or fuzzing, where the tester does not have access to the source code of the program to be tested (CERT/CC, 2017). Vulnerabilities can be automatically turned into exploits, as well, to decide which deserve priority to be fixed (Avgerinos et al., 2014). Creating exploits also transitions naturally into using them, which when done within the bounds of the law is called penetration testing.

During a penetration test trusted investigators, either contracted third parties or internal employees, "attempt to attack [identified] vulnerabilities in the same manner as a malicious hacker to verify which vulnerabilities are genuine" (Muniz and Lakhani, 2013, p. 1). There are open source tools, such as Metasploit, and whole ready-to-go tool suites, namely Kali Linux, for this purpose (Pritchett and De Smet, 2013).

Vulnerabilities can be considered from a more theoretical or comprehensive level as well. The term for this kind of analysis is the attack surface of software or a computer. Attack surface aims to estimate "the set of ways in which an adversary can enter the system and potentially cause damage" (Manadhata and Wing, 2011, p. 371). The main motivation for attack surface work is for security engineering and risk mitigation, not incident response. However, an incident responder might use the attack surface metric to prioritize investigating the weakest targets during an incident. Unfortunately, this is impractical

as existing attack surface metric calculation requires source code, often unavailable, and large code bases, "such as a Linux distribution," are too large to compute a value for anyway (Manadhata and Wing, 2011, p. 385).

INTRUSION DETECTION is the automated detection of security incidents; detection may be either signature- or anomaly-based (Scarfone and Mell, 2007). Intrusion detection is a mature field in its own right. Signature-based detection has a mature industry around supplying signatures for detection (Hopkins, 2009; Roesch et al., 2013). Applied statistics and machine learning has traditionally been more difficult, due to low base-rates of occurrence making reliable alerts difficult (Axelsson, 2000).

While any decent security installation includes an IDS, the primary function is detection, not analysis. However sophisticated automated detection may be, it does not directly tell the incident investigator what the adversary's goal is. Attack graph research has attempted to bridge this gap by classifying intrusion detection alerts to aid interpretation, but fails to scale in real-world problems (Lippmann and Ingols, 2005).

INTERNET MEASUREMENT includes both academic and professional communities. For incident response, the relevant measures are those that inform prior beliefs about the common attacks, common attack source locations, and reputation of operators regards responsiveness to reports of abuse on their infrastructure. The Anti-Phishing Working Group (APWG) produces multiple reports a year detailing patterns in domains that host phishing, which contain information related to this particular type of attack (see, e.g., Anti-Phishing Working Group (2016) and Rasmussen and Aaron (2012)). In the related but broader space of email anti-abuse, the Messaging, Malware, and Mobile Anti-Abuse Working Group (M3AAWG) reports levels of unwanted bulk email, commonly termed spam (M3AAWG, 2014).

There are also several efforts that measure the ecosystems supporting abuse and fraud. For example, several measurements indicate that attacks have gotten cheaper and easier to execute over time, such as crimeware-as-a-service, as detailed by Sood and Enbody (2013), and exploit-as-a-service, as detailed by Grier et al. (2012). Measurements of blacklists recommend using all possible lists with good conceptual fit, because each list shows little overlap with others (Metcalf and Spring, 2015). These measurements align with predictions that the cost of domains to attackers would approach their marginal

cost of zero (Spring, 2013b). Such measurements are examples of background information relevant to how incident responders should collect and analyse malware samples, exploits, blacklist information or internet identifiers (i.e., various types of evidence) during an incident.

SCIENCE OF SECURITY Whether or not cybersecurity is a science was first widely raised by MITRE (2010) as part of a US DoD contract. The result has been general consternation over the question of whether cybersecurity is a science with a general consensus (among those asking the question, at least) that the correct answer is "not yet." Chapter 3 argues against this pessimistic view and surveys the science of security landscape. That chapter will serve to argue that one can use scientific methods to better understand CSIR as well as a wider science of security. Chapter 3 is a separate literature survey, with a scope around the science of security and philosophy of science.

Although not generally discussed within the science of security community, the relationship between science and forensics is also potentially of interest. Casey (2010) states clearly that digital forensics uses the scientific method, though the author does not explore how it may differ in the digital context. There is relatively little work on how forensic science methods differ from their more academic counterparts; however, see, for example, Morgan and Bull (2007). CSIR is similar to forensics, in that when conducting CSIR the analyst has a policy violation they wish to explain. Unfortunately, there is not much in the way of a philosophy of forensics that is readily applicable to the scope of CSIR analysis.

2.9 CONCLUSION

This chapter has established a gap in the published advice available for how to perform CSIR analysis. My research question is how to formalise the reasoning process within CSIR analysis. Formalise here has both a qualitative and mathematical sense, and I will pursue both. The opportunistic pass through various fields in Section 2.8 suggested the available formalisation frameworks and tools are inadequate.

Based on this literature review, I claim that CSIR analysis lacks a strong intellectual foundation on which to build. In day-to-day practice, incident responders do good work. But strong intellectual foundations would help both improve this work and scale it up. Chapter 3 will map out which aspects

of the intellectual foundation of other sciences can be readily borrowed and which need to be constructed for CSIR specifically and information and cybersecurity generally. Following Chapter 3, I will be a position to discuss concretely what type of models of knowledge and reasoning in CSIR need to be constructed to address the research question.

3

LITERATURE REVIEW – SCIENCE AND SECURITY[1]

Although not framed as specific to incident response, there has been a prominent call to improve the research and practice of cybersecurity by making it more 'scientific'. Its proponents claim a science of security is needed for ongoing progress. Per the historian of science Dear, I use 'scientific' here as "a very prestigious label that we apply to those bodies of knowledge reckoned to be most solidly grounded in evidence, critical experimentation and observation, and rigorous reasoning" (Dear, 2006, p. 1). To scope out the security aspect of the discussion, I take the definition of security from RFC 4949: "measures taken to protect a system" (Shirey, 2007); see the RFC for the meaning of measures, protect, and system. As a starting point, then, I will consider a science of security to be the label we should apply to the most solidly grounded bodies of knowledge about measures one can take to protect a system.

The available literature is about a science of security in general, and not of incident response. In the prior chapter, I argued that incident analysis is valuable to study because it can have broad impact across the field of cybersecurity. In this chapter, I will rely on the idea that because incident analysis is a central subset of cybersecurity, any arguments about the field of science of cybersecurity also apply to a scientific approach to incident analysis. This review of the science of security literature does not discuss incident response explicitly. When seeking a scientific approach to human decision-making in incident analysis, it is sufficient to sketch it in a science of security in general.

This chapter will broadly outline philosophy of science before reviewing the science of security literature. I will find that there are no established methods from science of security to readily apply to incident response. I will find ready solutions from the modern philosophy of science literature for many stated challenges to cybersecurity (and thus to incident response). However, philosophy of science is not a ready solution for all challenges in security. Chapter 4 will directly tackle some of these as-yet unsolved challenges, con-

1 This chapter is based on joint work, the paper: Jonathan M Spring et al. (2nd Oct. 2017). 'Practicing a Science of Security: A philosophy of science perspective'. In: *New Security Paradigms Workshop*. Santa Cruz, CA, USA.

tributing to both security and philosophy. Let's start by introducing these stated challenges.

3.1 PURPORTED IMPEDIMENTS TO A SCIENCE OF SECURITY

The following items have resonated as serious obstacles to the practice of a science of security:

- Experiments are impossible in practice, because they are unethical or too risky;
- Reproducibility is impossible;
- There are no laws of nature in security;
- Cybersecurity will not be a science until everyone agrees on a common language or ontology of terms;
- Computer science is 'just engineering' and not science at all: questions of science of security are misplaced.

A philosophy of science perspective will show these obstacles are either misguided or can be overcome. The purported obstacles are frequently not genuine challenges because they rely on outdated conceptions of science, which yields a simplistic idea of evaluating evidence for claims (namely, falsification) and a naïve reductionism to universal laws that supposedly underpin all scientific endeavours. Alternatively, modern philosophy of science tends to describe, if applied adequately to security, what good security practitioners already do. Security is, as practised, already a kind of science. Table 3.1 summarizes our positive perspective on executing a science of security.

Section 3.2 provides a brief background on the logical empiricist movement within philosophy of science. Section 3.3 examines prior statements to detail the obstacles to practising a science of security. Section 3.4 explains how philosophy of science informs the scientific process already taking place in cybersecurity research, and Section 3.5 suggests a main gap is improving the reliability and growth of general, shareable knowledge. Chapter 4 will address this gap directly; the logic I will define in Chapter 7 also enables codifying and sharing knowledge.

Experiments are untenable	Scientists make structured observations of the empirical world
Reproducibility is impossible	Evaluate evidence by repetition, replication, variation, reproduction, and/or corroboration
No laws of nature	Scientists employ mechanistic explanation of phenomena to make nature intelligible
No single ontology	Specialization necessitates translation
'Just' engineering	Both science and engineering are necessary

Table 3.1: Five common complaints raised by the science of cybersecurity community and positive re-framing from the philosophy of science literature.

3.2 PHILOSOPHY OF SCIENCE – HISTORICAL BACKGROUND

Philosophy of science is a field that has developed as a discourse on top of science: a second-order reflection upon the operation of the sciences (Uebel, 2016). For three centuries, the scholars now recognized as scientists were called 'natural philosophers'; there were no separate philosophers of science. In inter-war Vienna, a group of thinkers who identified as 'the Vienna Circle' challenged both the prevailing metaphysics and political Romanticism (i.e., the Church and European fascism).[2] This movement emphasized themes of observation of the world, trust in science, high value on math and logic, and modernism. A key movement of the Circle has come to be called *logical empiricism*, for its reliance on logical rules based on empirical observations.[3]

Presently, I will briefly introduce two of the main tenets of logical empiricism: (i) empiricism and verification and (ii) unity or reduction of scientific fields (Creath, 2014). These tenets coalesced in the 1930s, were refined through the 50s, and by 1970 had suffered ample critiques to be changed beyond recognition. This historical trajectory makes it intellectually dangerous to rely upon logical empiricist arguments or concepts uncritically. Yet, Section 3.3 finds much logical-empiricist work uncritically assimilated in current statements on a science of cybersecurity.

2 The Economist (2016) argues that 1880–1930 Vienna produced the most vital intellectual movements in science, art, and philosophy of the 20th century.

3 Logical empiricism is closely related to logical positivism and neopositivism; I will not distinguish these at my level of analysis (Creath, 2014; Uebel, 2016).

EMPIRICISM AND VERIFICATION Statements testable by observation were considered to be the only "cognitively meaningful" statements (Uebel, 2016). Although logic and mathematics are the most reliable forms of reasoning, logical empiricists did not take them to rely on observation but instead accepted them as true by definition, following Russell and early Wittgenstein. Therefore, according to the logical empiricist view, the key scientific challenges are how to verify a statement is in fact about the world, and how to meaningfully integrate observations into logic and mathematics. Such integration is necessary for science to be useful. Integrating observations into deductive logical statements is also a response to Hume, two centuries earlier, and his famous problem of induction. Hume, in broad strokes, argues that no matter how many times everyone observes the sun to rise, we cannot prove (in the sense of deductive proof) that the sun will rise tomorrow based on the observations.

Consistent with logical empiricism, Carnap proposed a method for *verification* by working on atomic elements of logical sentences, and expanding observational sentences out based on rules from atomic observations (Creath, 2014). The goal of empiricism is to be grounded in observations. The goal of verification is to integrate those observations into a framework of general knowledge, in the form of statements in first-order logic, that can justify predictions. Carnap thus links induction and deduction, bypassing Hume's complaint.

Yet it became clear that verification might not always be achievable. It is against this backdrop that Popper proposed the more limited objective of falsification (Popper, 1959), which claims scientists cannot verify logical statements at all. Instead, Popper asserts that the best anyone can do is hope to falsify them.[4]

Even the more limited goal of falsification was shown to be untenable with Kuhn's challenge to Popper in 1962 (Kuhn, 2012). Kuhn refutes the premise that scientists operate solely on logical statements. Rather, he argues that key examples, literally 'paradigms', are scientists' operative cognitive model. Later work in philosophy of science has refined the shape of these cognitive models – one prominent method is as *mechanistic explanations* (see, e.g., Glennan and Illari (2017)) – and improved understanding of how data are processed to provide evidence for phenomena (see, e.g., Bogen and Woodward (1988)).

4 Popper published this in German in 1935, though the popular English translation appeared in 1959. Thus Carnap's 1956 work is actually done in knowledge of and contrary to Popper. Earlier verificationists, stricter than Carnap and against whom Popper reacted, include Wittgenstein as early as 1929 (Creath, 2014).

Even ignoring Kuhn's socio-scientific critique, falsification is about mapping observations into logic. Popper is silent on designing reliable observations and choosing what logic or conceptual framework in which a person should reason. These two problems are more important, and would provide more actionable advice, than whether something is falsifiable. More useful than falsification are modern discussions of investigative heuristics for scientists (Bechtel and Richardson, 1993), models of when a conclusion from observations is warranted (Norton, 2010), and accounts of causation that make use of intervention and statistics rather than logical implication (Woodward, 2003).

REDUCTION OF SCIENCE TO FIRST PRINCIPLES The other tenet of logical empiricism often unwittingly inherited by debates in science of security regards the unity of science or the reduction of science to single first principles. There are two senses of unity here that are often not properly distinguished: methodological unity and unity of content by reduction to a single set of models. A unity of methods would mean that, although individual sciences have distinctive approaches, there is some unifying rational observation and evaluation of evidence among all sciences. This view was deemphasized within logical empiricism. With confusing terminology, modern arguments often return to this idea under mosaic unity or pluralism: the sciences are about widely different subjects, but there are important shared social and methodological outlooks that unify science as an enterprise.

The traditional idea of reductionism is that the set of laws of one science can be logically reduced to that of another (Nagel, 1979). This notion requires the conception of laws as logical rules of deduction. As famously critiqued by Cartwright (1983), the laws of physics are not true explanations of the world, but rather of the models of the world. If laws are about models, and models can be diagrams or small-scale physical replicas, it is unclear how reduction could be defined. Bickle (2008) defines reductionism (in neuroscience) as when a lower-level mechanism contains all the explanatory power necessary to intervene on a higher-level mechanism. Merits of Bickle's view aside, he has disposed of all logical-empiricist ideas of laws, deduction, and verification and uses the modern concepts of mechanistic explanation and intervention.

Reductionism is dangerous because it tends to blind researchers from using the appropriate tool for the job. If everything reduces to physics, then a physics-hammer solves all problems, and everything looks like a nail. But I claim a more diversified toolbox is needed in a field such as cybersecurity.

Social sciences play an equally important role as technical sciences (Anderson and Moore, 2006). The modern terms in philosophy of science are *integrative pluralism* (Mitchell, 2003) or *mosaic unity* (Craver, 2007). The core of these terms is that fields cooperate on adding constraints to coherent explanations according to their particular tools and expertise. Such interfield explanations are what is valuable, not reductions (Darden and Maull, 1977). I explore challenges due to reductionism and its alternatives further in Section 3.4.3 and Section 3.4.4.

SCIENCE AS A PROCESS A common pitfall treats the terms 'scientific' and 'correct' as synonyms. Science is a process; it yields answers. Answers are correct or not based on facts of the world. However, one calls a process 'correct' if it follows an agreed, human-defined form. The question about a process should be whether it is satisfactory in efficiently producing adequate answers. One should not assume answers are reducible to one 'correct' answer; many answers may adequately satisfy a purpose (Simon, 1996). Conflating 'scientific' with 'correct' and 'correct answer' with 'adequate' results from logical-empiricist assumptions in the common complaints.

Removing these faulty assumptions is not a panacea. A sound scientific process may produce unsatisfactory results if the subject matter is difficult to study for undiagnosed reasons. One may construct a model of a system or phenomenon using scientifically rigorous methods. The model constructed will have certain properties that are considered correct if it adequately captures the properties of the system or phenomenon that are required to address the questions that the model is intended to explore. One key goal of this chapter is to refocus the question from 'is this process scientific' to 'why is this scientific process producing unsatisfactory results'.

Section 3.3 will evidence how logical empiricist threads pervade existing discussions of science of security and continue to tie in critical reflection from modern philosophy of science.

3.3 EXISTING STATEMENTS OF SCIENCE AND SECURITY

Many organizations have proposed problem statements and solutions regarding the state of cybersecurity research. Since 2008, these statements are frequently phrased using the language of a science of security. The motivation and

goals are complex, but one important consideration is policy makers asking for intelligible explanations that can inform their decisions. This section will first survey the problem and then the proposed solutions.

There is broad agreement that there is a problem with the state of cybersecurity. That sense predates the arguments that science is the answer; for example, education and standardization efforts predate the focus on science. More accurately, developing a science of security is part of a multi-pronged approach by the US government, later picked up by others, to respond to threats to ICT infrastructure. As early as 2001, the National Science Foundation (NSF) funded both student scholarships and academic capacity building (e.g., designing courses) to universities designated by the NSA as a Center of Academic Excellence (CAE) in Information Assurance Education (NSF, 2001). The NSA CAE program began in 1998. The National Institute of Standards and Technology (NIST) has been involved in information security standards for decades. Unlike what Chapter 2 found for incident response, in security generally the IETF and IEEE are at least as prominent as NIST. The study of how security standards require different features than usual information technology standards has only just begun (Kuhlmann et al., 2016). However, around 2008, there seems to have been a shift emanating from the US DoD that brought the question of science to bear on cybersecurity problems.[5]

The DoD expresses the motivation for its scientific shift in its tasking to MITRE, quoted by MITRE's JASON office in its final report. The reason for the timing is unclear, but the concern that precipitates the scientific shift is clear. This concern is worth quoting at length:

> "The Department of Defense, the Intelligence Community, and the planet have become critically dependent on the Internet for services, transactions, social interactions, communications, medical treatments, warfare; virtually everything. Cybersecurity is now critical to our survival but as a field of research does not have a firm scientific basis. Clearly, there is a scientific basis for the infrastructure of the internet such as computation, communications, and integrated circuits but its security attributes are

5 The first use of 'science of cybersecurity' or 'science of information security' is elusive. Google Scholar searches for these terms (with quotes) on Mar 1, 2017, restricted to 1990-2007, yield exactly one plausible result: a 2004 work on critical infrastructure policy (Winn, 2004). Perhaps (Winn, 2004) borrowed from the 2004 U.S. House Subcommittee on Cybersecurity, Science, and Research & Development, which she cites. However, besides in the subcommittee title, its report does not mention 'science of cybersecurity' or security science.

> often vague or un-measurable. . . . There are concerns that future damage could be catastrophic to major components of our core infrastructures such as power and water distribution, finance and banking, even the ability to deploy forces to defend the country.
>
> Our current security approaches have had limited success and have become an arms race with our adversaries. In order to achieve security breakthroughs we need a more fundamental understanding of the science of cyber-security. However, we do not even have the fundamental concepts, principles, mathematical constructs, or tools to reliably predict or even measure cyber-security. It is currently difficult to determine the qualitative impact of changing the cyber infrastructure (more secure now or less secure?) much less quantify the improvement on some specific scale. We do not have the tools to do experiments that can produce results that could be compared to theory, models, or simulations. Request the JASONs consider whether cyber-security can become or should be a science. If so, identify what is needed to create a science of cyber-security and recommend specific ways in which scientific methods can be applied to cyber-security. If not, what can we learn from the practice of science that would enable us to improve the security of our cyber infrastructure and assure the integrity of information that resides in the information technology infrastructure?" (MITRE, 2010, p. 9-10)

Note three key aspects of the problem statement. First, cybersecurity is of critical societal importance. Secondly, a desire to predict and measure security via "concepts, principles, mathematical constructs, or tools." Third, the purpose of this prediction is to prevent future catastrophic infrastructure damage. Science is positioned as a possible answer, but is not presumed. The real question is not whether security is a science, but "what can we learn from the practice of science that would enable us to improve the security of our cyber infrastructure."

Much government-centric or government-funded science of security work seems to accept this problem statement, including the Air Force MURI project. There is one recent voice with which to compare. The inaugural event for the working conference "Art into Science: A Conference for Defense (ACoD)" in early 2017 held the goal:

> "*Push the art to a science: Creating a professional discipline.* To mature our practice, we need to be able to share our methodologies in a systematic and consistent way. We have many professionals in the security industry, but do not have a professional discipline. We'd like to philosophically discuss concepts in security, push them forward, and model them" (Evron, 2017, emphasis original).

Interpreting this goal as a problem statement, it is a claim that security practitioners cannot share methods satisfactorily. Science is taken as a way to systematize knowledge at the appropriate level of generality that it can both be shared and remain useful. Sharing generalized knowledge would support the prediction and measurement of security, as identified in the DoD statement. The two statements do not disagree, but the modified focus may lead to different kinds of solutions. However, the ACoD community is too new to evaluate the results of this different perspective.

Having covered the available problem statements, I will switch to six statements of the current status of the science of security, each positioned as a method to solve the DoD problem statement. First, the direct response in MITRE (2010). Second, a DoD agency, the NSA, in its funding priorities. Third, a speech by Dan Geer, head of the public venture capital firm of the CIA, In-Q-Tel. Fourth, I will inspect the mission statement of the UK Research Institute in Science of Cyber Security. Fifth, the 2016 cybersecurity strategy laid out by the President of the United States. Finally, I consider a systematization of academic knowledge by Herley and van Oorschot (2017).

MITRE—JASON Although the DoD does not presuppose science as the answer to their challenge statement around cybersecurity, the problem statement does presuppose a conception of science. This received conception directly impacts the answers possible. For example, the report concludes "There are no intrinsic 'laws of nature' for cyber-security as there are, for example, in physics, chemistry or biology" (MITRE, 2010, p. 79). As Section 3.4 will demonstrate, the claim that there are laws in biology is highly contested, and the notion of unqualified, universal laws anywhere has been challenged with general success.

The implicit goal of science as putting forward unifying theories perhaps leads to the recommendation that "the most important attributes would be the construction of a common language and a set of basic concepts about

which the security community can develop a shared understanding" (MITRE, 2010, p. 3, cf. p. 15). MITRE is funded to curate several language ontologies, including MAEC at the time of the JASON report.

JASON does not provide a concise statement of what science means within the report, or what the report is hoping to provide. The report searches for "guidance" from other sciences, namely economics, meteorology, medicine, astronomy, agriculture, model checking, and immunology. Thus it seems the authors judge all these fields to be sciences worthy of emulation. The report notes that sciences need not conduct experiments and may be observational. The "crucial feature" is that data be "generalizable" (MITRE, 2010, p. 34). Unfortunately, the report is silent on how to achieve this. The closest JASON gets to a formulation of what a science of security would contain is to say "it is not simple to define what the 'security' in cyber-security means" and to call for precise definitions (MITRE, 2010, p. 22). One gets the sense that the authors explained what existing sciences may contribute to security, rather than how scientific methodology could be adapted to cybersecurity as an independent field.

NSA The NSA uses a definition of 'Security Science' to guide research funding considerations. Although this was posted rather obscurely to an online community forum of security researchers, the NSA operates the forum, and the description is by the technical director emeritus:

> "Security Science – is taken to mean a body of knowledge containing laws, axioms and provable theories relating to some aspect of system security. Security science should give us an understanding of the limits of what is possible in some security domain, by providing objective and qualitative or quantifiable descriptions of security properties and behaviors. The notions embodied in security science should have broad applicability – transcending specific systems, attacks, and defensive mechanisms. The individual elements contained within security science should contribute to a general framework that supports the principled design of systems that are trustworthy, they do what people expect it to do – and not something else – despite environmental disruption, human user, and operator errors, and attacks by hostile parties. Trustworthy system design may include contributions from a diverse set of disciplines including computer science, systems sci-

> ence, behavioral science, economics, biology, physics, and others" (Meushaw, 2012b).

This definition of science of security seems to be what the Symposium on the Science of Security (HotSoS) CFP has in mind when it refers to building "a foundational science of security" (Katz, 2016). The CFP does not otherwise define science of security, but the conference is funded largely by NSA. The definition has also influenced academic work, such as that summarized in a special issue of S&P Magazine, Evans and Stolfo (2011).

The NSA definition defines the characteristics of an answer, not a process. The science is a "body of knowledge," it provides "objective...descriptions," and should support "principled design of systems that are trustworthy." Such goals describe the outputs of a process, not how to conduct the process so as to achieve these results. Advice would be more actionable for practitioners if it guided how to act in order to bring about an end goal. This observation does not make the NSA statement wrong, just less useful.

Security academics informed the NSA definition. The NSA collected its influential thinkers for a special issue in "The Next Wave" on a "blueprint for a science of cybersecurity." The magazine highlights emerging trends salient to the NSA. The issue entrenches a logical empiricist position on a science of security, especially Schneider's article (Meushaw, 2012a). Like the JASON report, some articles mention biology or engineering, but nothing systematic and with no true departure from the inherited logical empiricist world view.

DAN GEER Related to HotSoS, the NSA also runs an annual 'science of security' award for best paper. One of the distinguished experts that review the nominations, Dan Geer, provides the following insight into the reviewers' decision making:

> "Amongst the reviewers our views of what constitutes a, or the, Science of Security vary rather a lot. Some of us would prioritize purpose... Some of us view aspects of methodology as paramount, especially reproducibility and the clarity of communication on which it depends. Some of us are ever on the lookout for what a physicist would call a unifying field theory. Some of us insist on the classic process of hypothesis generation followed by designed experiments" (Geer, 2015).

The disagreement highlighted by this statement is that cybersecurity experts may view the area with which they are familiar as the area that makes security into science. This bias is natural, any expert has likely pursued what they view as the most important topics. Why would any of Geer's listed possible priorities take primacy? All contribute to a wider understanding of the world and its mechanisms via different methods. There are unifying aspects, and important differences, between different biological sciences, as one example. However, reducing the decision to one or another feature, as Geer (2015) indicates the reviewers of the best 'science of security' paper are disposed to do, collapses away the nuance necessary for a constructive perspective on a science of cybersecurity.

RISCS Research Institute in Science of Cyber Security provides a perspective outside North America. Research Institute in Science of Cyber Security (RISCS) was established in 2012 and funded by the UK Engineering and Physical Sciences Research Council (EPSRC), Government Communications Headquarters (GCHQ), and Department for Business, Innovation, and Skills (BIS). Further, RISCS is cited by The Royal Society in their strategic plan for cybersecurity research in the UK generally (Royal Society, 2016). The statement of purpose summarizes the Institute's mission:

> "RISCS is focused on giving organisations more evidence, to allow them to make better decisions, aiding to the development of cybersecurity as a science. [RISCS] collects evidence about what degree of risk mitigation can be achieved through a particular method – not just the costs of its introduction, but ongoing costs such as the impact on productivity – so that the total cost of ownership can be balanced against the risk mitigation that's been achieved. [RISCS]'s main goal is to move security from common, established practice to an evidence base, the same way it happened in medicine" (UCL, 2017).

The emphasis on science is much more pragmatic in this British context. The primary goal of the Institute is to provide evidence for improved decision making; this in itself is taken to advance a science of cybersecurity. This approach neatly sidesteps many of the questions about what makes a science. RISCS issues annual reports about its work, and this may be leading by example, but it is not self-reflective about how the work advances a science of

cybersecurity (RISCS, 2016). It is not enough to say we need to use evidence, like in medicine. That statement is true, and the work done by RISCS certainly is scientific and does advance a science of cybersecurity. Similarly, the work presented at HotSoS and otherwise supported by DoD has a positive impact on the field. The community should want to extract the reasons why. For this explanatory task, the pragmatic statement of RISCS is not enough.

WHITE HOUSE The White House has developed a 'cybersecurity research and development (R&D) strategic plan' (Shannon et al., 2016, p. 2). To a large extent, the plan takes the definition of science for granted. The plan is much more about what types of work the agencies should prioritize for funding. However, these priorities explicitly include reinforcing a scientific foundation and a research infrastructure that are deemed to be lacking.

> "Cybersecurity...needs sound mathematical and scientific foundations with clear objectives, comprehensive theories (e.g., of defense, systems, and adversaries), principled design methodologies, models of complex and dynamic systems at multiple scales, and metrics for evaluating success or failure. ...[Currently,] most techniques are domain- and context-specific, often not validated as mathematically and empirically sound, and rarely take into account efficacy and efficiency. Thus, the state of the practice consists of heuristic techniques, informal principles and models of presumed adversary behavior, and process-oriented metrics" (Shannon et al., 2016, p. 30).

> "Research Infrastructure: Sound science in cybersecurity research must have a basis in controlled and well-executed experiments with operational relevance and realism. That requires tools and test environments that provide access to datasets at the right scale and fidelity, ensure integrity of the experimental process, and support a broad range of interactions, analysis, and validation methods" (Shannon et al., 2016, p. 13).

The strategic plan emphasizes certain aspects of security to focus on, for example defense should focus on deter, detect, protect, and adapt. This is unobjectionable as far as practical direction goes. However, these are the subject-areas about which to do science, but not related to any definition of science. The scientific foundations and research infrastructure are cross-cutting issues

of methodology to be applied to the subject matter of priority. In this way, the strategic plan plausibly separates the desire for answers from the method by which reliable answers are derived. At the same time, the prescriptions will come to seem overly rigid in our analysis. Why are domain-specific techniques not scientific? The statement "most techniques are domain-[specific]" above seems to imply that this state of affairs is unacceptable compared to "comprehensive theories," but this argument is unclear. Does the fact that specially-designed radio telescopes for finding pulsars in the centres of distant galaxies cannot be used to analyse toxicity in marsh ecosystems make finding pulsars unscientific somehow? Clearly not.

ACADEMIC SURVEY Herley and van Oorschot provide a comprehensive academic perspective (Herley and van Oorschot, 2017). Their main thesis takes a pessimistic outlook:

> "[T]he security community is not learning from history lessons well-known in other sciences. ... What is [hard] to accept is [The security community's] apparent unawareness or inability to better leverage such lessons" (Herley and van Oorschot, 2017, p. 16).
>
> "...practices on which the rest of science has reached consensus appear little used or recognized in security, and a pattern of methodological errors continues unaddressed" (Herley and van Oorschot, 2017, p. 1).
>
> "...The failure to validate the mapping of models and assumptions onto environments and systems in the real world has resulted in losing the connections needed to meet [the goal of improving outcomes in the real world]" (Herley and van Oorschot, 2017, p. 16).

The belief underlying this assessment seems to be that security researchers are unfamiliar with scientific methods or philosophy of science. This claim motivates an argument that the science of security initiative is essentially too vague to be useful. The solution Herley and van Oorschot advocate is to introduce the philosophical literature and draw practical lessons for security researchers. I concur with this general assessment, and the direction of the discussion and solutions. However, the solution does not focus on the most relevant or helpful segments of the philosophical literature. Instead, they rely on a historical framing emphasizing logical deduction and the problem of

induction. This framing inherits a world view from logical empiricism, Hume, and Kant. While historically important in philosophy of science, this perspective does not provide adequate tools for solving the modern challenges of a science of security. Modern philosophy of science judges the science of security less harshly than the failings identified by Herley and van Oorschot, while providing more actionable positive advice. Here, I focus on the philosophical summary and argue why the proposed solutions in the Systematization of Knowledge (SoK) are inadequate; Section 3.4 provides the positive advice from modern philosophy of science.

Supposed failures to apply lessons from science are the core argument. Some are novel compared to the government and industry positions explored above; for example, "failure to seek refutation rather than confirmation" and "reliance on unfalsifiable claims" (Herley and van Oorschot, 2017, p. 11,13). Unfortunately, these observations require logical empiricist views, notably Popper and Ayer. As explained in Section 3.2, logical empiricism is an outdated view that has been supplanted by more recent scholarship in the philosophy of science. Relying upon logical empiricist arguments has unfortunately led the authors to draw conclusions that are often unhelpful or incorrect.

Consider the criticism of failure to seek refutation rather than confirmation. What do practitioners refute in security? If it could be logical sentences somehow implying authorization, perhaps this is a useful criticism. However, authorization is a policy decision; in the terms of logic, it is a semantic property. One must define the model structure, satisfaction condition, and domain before interpreting any sentences and checking semantic properties. Such definitions can be argued for or justified, but are always contextual and not something usually refuted or confirmed. Like all of security, it cannot be done according to absolutes. This logical-empiricist drive for logical absolutes confounds security just as quickly as it has confounded other sciences. A better expression of these worries is that generalizations from particulars should be properly evidenced and that reasoning from existing knowledge should be justified appropriately. As elaborated in Chapter 4, I take the mechanism discovery literature as a better framework in which to discuss generalization. While the philosophical aspects are not made explicit, this emphasis on evaluating evidence and warranted generalization is consistent with the arguments put forth by Shostack and Stewart (Shostack and Stewart, 2008).

Because (Herley and van Oorschot, 2017) takes falsification as central, it is silent on how to draw useful generalizations in the social sciences. Since

a social science perspective is needed to understand cybersecurity (Anderson and Moore, 2006), this is a significant shortcoming. To see this, consider the statement "refuting evidence is [always definitive]" (Herley and van Oorschot, 2017, p. 16). This statement assumes the item being refuted has certain properties; namely, it must be a decidable, valid logical sentence in a sound proof system with excluded middle that is satisfied in a relevant model structure. Common, useful biology and sociology statements do not have these properties. Instead, sociologists (Elster, 1989) and biologists (Glennan, 2015) tend to talk about mechanisms. I expand on such alternatives in Section 3.4.3.

Two other failures, "to bring theory into contact with observation" and "to make claims and assumptions explicit" (Herley and van Oorschot, 2017, p. 12), are already represented by complaints from other sources about reproducibility, challenges to experimentation, and a common language. Making claims explicit is generally good advice, though Herley and van Oorschot (2017, p. 12) wants explicitness so observations will fit into a common-language of a logical theory. Thus I wish to dispense with the common-language critique while retaining the recommendation of explicitness. Explicitness has been recommended in science of cybersecurity previously by Maxion (2015), under the term "structure" or argument clarity, and by Hatleback and Spring (2014) as "transparency".

Despite our criticism, many recommendations in Herley and van Oorschot (2017, §5) are good. I agree that physics-envy and crypto-envy are counterproductive, for example. So why does it matter that they rely on outdated philosophy – logical empiricism – to get there? For one, the reasons for these conclusions matter. Physics- and crypto-envy are counterproductive because there is nothing special about them to envy. Physics and crypto are especially well-suited to a logical-empiricist perspective of logical laws of nature that can be falsified by observation presenting contradiction. Rejecting crypto-envy would not make sense if it were actually well-suited to our definitions of science. It matters that one do not merely say 'do not worry you are not as good as physics' but instead 'physics is not as unique as you think it is'. Craver's conception of *mosaic unity* in the mechanisms literature (Craver, 2007) is a more useful framework to understand why crypto-envy is counterproductive. Each field participates in a multidisciplinary endeavour like security by contributing constraints on complex mechanistic explanations. Crypto is just one such field, and all fields produce constraints that are, *a priori*, of equal value to the overall explanation.

3.3.1 *Prepositions: 'of' or 'for' security*

This is a small digression on the two phrases 'science *of* security' and 'science *for* security'. The choice of preposition makes a large difference in mental orientation. Those that use 'for' include the DoD problem statement and RISCS. Many of the other statements use 'of'.

These small words (for vs. of) unpack into some big differences. Science for security seems to indicate taking any scientific discipline or results and using that to make decisions about cybersecurity. Thus, 'for' is agnostic as to whether there is any work within security that looks like science. Science *for* security would simply advocate for evidence-based security decisions.

On the other hand, a science *of* security looks for a security science to establish itself as an independent, peer discipline of other sciences, with idiosyncratic methods, concerns, and venues. As this section highlighted, almost all the published work on a science of security is pessimistic about its existence or contributions. I suggest that distinguishing a science *of* security from science *for* security will yield more actionable advice on what to do moving forwards, and how to make both science of and for security better.

For example, NASEM (2017) follows the above pessimism on a science *of* security. However, 'for' or 'of' may matter a lot. Is cryptography part of a science of security, or is it mathematics for security? It seems plausible to treat cryptography as mathematics for security. But if that's the case, then lessons from cryptography have little bearing on methods for a science of security. For example, one would not expect lessons from cryptography to apply to usable security any more than expect lessons from mathematics to apply to sociology. Such transmissible lessons can be found, such as with statistics in experiment design, but they are quite context-specific.

This reorientation naturally presents the question of what topics are in a science *of* security. Perhaps, topics that cannot be studied independent of security concerns. Or perhaps it is just areas where the challenges specific to cybersecurity dominate the challenges of other disciplines. Either way, CSIR would be one example of a science of security. This is not to say that a science *of* security is more important than any sciences for security. To some extent, the core science of security may primarily be a translator between and among the other sciences for security. Certainly, if science is understood following Dear (2006, p. 1) ("a very prestigious label that we apply to those bodies of knowledge reckoned to be most solidly grounded in evidence, critical

experimentation and observation, and rigorous reasoning") all these various disciplinary perspectives on contributing to such knowledge in security are worth encouraging and nurturing.

Summary

These six broad statements highlight common themes. All agree there is a problem: cybersecurity is vitally important to society, yet not sufficiently understood to produce reliable systems. Each account on the source of that problem directly informs proposed solutions. The academic SoK uses historical philosophy of science to suggest what security needs. The other North American statements, from DoD and MITRE especially, implicitly use this logical-empiricist view heavily. Dan Geer's view highlights a naïve reductionist philosophy of science closely related to logical empiricism. RISCS's pragmatic statement carries little philosophical baggage, but provides little advice on how to adequately gather and evaluate evidence. A common mistake is to confuse evaluation of the process of doing science with the evaluation of the answers the process provides.

The common thread is to look to biology, physics, or other sciences for advice. This search may be misplaced. Philosophy of science is the independent field that discusses how to execute other sciences and such issues. As Section 3.4 will demonstrate, I can disqualify all of the complaints extracted from the above that cybersecurity is not a science. Security is a science, and should look to philosophy of science to address the genuine challenges of a science of cybersecurity.

Some claim a science of security is not possible; some there is no science of security yet (see Hatleback (2017) for a recent expression of this view); some just that too few people practice it. By contrast, Section 3.4 will disassemble the argument that such a science is impossible by explaining how modern philosophy of science supports the practices of cybersecurity as a science in essentially its present form.

3.4 PRACTICING SCIENCE OF SECURITY

Section 3.3 covered a broad selection of complaints that security is not scientific enough. This section contrasts those complaints with alternatives based

Untenable experiments Structured observations more broadly, not just experiments, are necessary for science. Qualitative research methods (Given, 2008) such as case studies (Stake, 1995), and natural experiments (Morgan, 2013), provide usable intellectual structure. Privacy and ethical concerns are adequately addressed by the Menlo report (Dittrich and Kenneally, 2012). Rapid technological change makes generalization a genuine challenge, but existing tactics can be synthesized, as Chapter 4 will explore.
Reproduction is impossible Reproduction comes in many forms (corroboration, statistical power, repetition, etc.) and usually only some work (Feitelson, 2015; Stodden, 2015). The misconception is requiring all forms simultaneously. See Cartwright (1991) for a touch point. Scientific work may cover non-replicable events, e.g., the extinction of the dinosaurs (Glennan, 2010).
No laws of nature 'Law' interprets how scientists explain or generalize knowledge, but is too rigid even to describe physics (Cartwright, 1983). Causal explanation as intervention is well-developed (Halpern and Pearl, 2005a,b; Woodward, 2003). Philosophy of science provides access to a rich set of mechanism discovery heuristics used in other sciences (Bechtel and Richardson, 1993; Craver, 2007; Darden, 2006) that Chapter 5 will port to security. From 'laws', these heuristics for designing and interpreting observations are unavailable.
No single ontology A single language does not define a field. Within physics, the subfields communicate via trading zones in which exchanges between the their jargons occur (Galison, 1999). Trading zones apply in security as well (Galison, 2012). Neuroscience provides a better metaphor for demarcating a science of security: the mosaic unity coheres from multiple subfields providing constraints on multi-level mechanistic explanations (Craver, 2007).
'Just' engineering Engineering as usually practised depends on science (Vincenti, 1990), while at the same time science as usually practised depends on engineering (Dear, 2006). Chapter 6 will explore the extent of the overlap between logic, engineering, and science in a case study of program verification.

Table 3.2: Summary of common complaints raised by the science of cybersecurity community and recommendations on positive actions from the philosophy of science literature to counteract the complaints.

on a more comprehensive view of science according to the philosophy of science literature. The immediate aim is to clear away these unjustified complaints that security is unscientific. The larger purpose is to, thereby, make genu-

ine challenges more visible. Table 3.2 lists the five common complaints and summarizes how to defuse each with the positive advice in the history and philosophy of science literature.

3.4.1 *Scientific methods*

CLAIM: EXPERIMENTS ARE UNTENABLE The inability to conduct experiments, at least in many contexts, is held to be a major strike against a science of cybersecurity. The received view is an explicit evidence hierarchy, with Randomized Controlled Trials (RCTs) at the top (Cartwright and Hardie, 2012, p. 135ff). This view is rooted in medicine, influenced public policy generally, and in turn security. A critical study of proper evidence-based policy summarizes the received view succinctly: "You are told: use policies that work. And you are told: [RCTs] will show you what these are." And yet, they immediately follow with "[RCTs] do not do that for you" (Cartwright and Hardie, 2012, p. ix).

Nevertheless, experiments generally, and RCTs specifically, hold a high status. High enough status that many statements from Section 3.3 present lack of experiments as sufficient grounds to demonstrate security is not a science. Ultimately, this high status reaches back to a conception of science as falsifying logical statements inherited from logical empiricism, for which RCTs are well-suited. I counter three common reasons for the claim experiments are untenable in security: lack of suitable control, privacy constraints, and rapid technological change.

Untenable experiments, narrowly understood, is not the right complaint in the first place. Section 3.2 expanded our view of scientific explanation beyond falsifiable logical statements. Therefore, robust explanation needs broader methods of evaluation than experiments.

ALTERNATIVE: STRUCTURED OBSERVATIONS OF THE EMPIRICAL WORLD Experiments are not a necessary condition for a field to be a science. No one can induce controlled hurricanes, yet meteorology remains a science. Similarly for astrophysics – no one induces supernovae – and palaeontologists, as no one induces extinction via meteor impact. Social sciences abound that rely on case studies; an individual case is "a specific, a complex, functioning thing" (Stake, 1995, p. 2).

I will prefer the term 'structured observations' over experiment as a necessary feature of science. I mean structured observations to include both experiments and case studies. Robust research methods provide the structure, for example as described by (Given, 2008; Stake, 1995). Structured observations are empirical, and this includes both qualitative and quantitative studies. Let us rephrase the complaint, then, as structured observations are untenable in security. Nonetheless, I will clearly demonstrate how those practicing a science of security can and already have been overcoming objections in the context of structured observations.

OVERCOMING: LACK OF CONTROL GROUPS There are surely some situations in studying security in which RCTs are not possible. Such a trial involves dividing a group of subjects such that the only statistically-meaningful difference between the two groups should be the intervention of interest. This structure permits statistical determination of the intervention's impact, granted various properties about the design hold.

RCTs have come to be considered a cornerstone of evidence-based medicine, and there is prestige associated with RCTs. However, recent projects challenge this perception of RCTs as standing alone at the top of a strict evidence hierarchy. For example, without mechanistic evidence, one cannot decide the details of what RCT to design and conduct (Williamson, 2015). Such arguments do not cast doubt on the use of RCTs when they are plausible, but rather cast doubt on the undisputed primacy of RCTs as the *best* evidence. Various interlocking kinds of evidence are necessary for evidence-based medicine; I see no reason why security should differ in this regard. Case studies, natural experiments, model-based reasoning, and RCTs all have important, interdependent roles to play. This insight helps sharpen what is actually needed in security research to make it more like medicine, as called for by the UK's RISCS.

There are instances where RCTs have a role to play in security, particularly where security interfaces with psychology, e.g., in usable security. Examples are success in studying alternatives to passwords (Biddle et al., 2012b) or biometric uses of keystroke dynamics (Killourhy and Maxion, 2009). Usable security experiments do have pitfalls essentially unique to the field; for example, to make sure users have a realistic sense of risk in the lab environment (Krol et al., 2016). And like in medicine, evidence is needed to link the experimental result to cases outside the lab.

As an example of a variety of approaches to structured observations in action, consider the following brief review of research involving passwords. Case study methods yield insights into how people select passwords and what motivates their behaviour. An example of a qualitative case study comes from Wash (2010), who conducted in-person interviews to identify the mental models people use when thinking about security, including passwords. Wash found that while everyone agreed that selecting good passwords was important, articulating why or how was much harder. Gaw and Felten (2006) present a quantitative case study of 49 undergraduate students that documented widespread password reuse, along with incorporating non-private attributes such as phone numbers into passwords. These case studies are complemented by later observational studies carried out at a larger scale. For example, Das et al. (2014) analysed hundreds of thousands of leaked passwords to quantify the prevalence of password reuse and other insecure activities. This study corroborated earlier case studies. Motivated by this and other studies, researchers have proposed new mechanisms to enable better password selection, which can then be evaluated empirically. For example, Ur et al. (2012) ran an experiment in which users selected passwords with the aid of 14 deployed strength meters. While the most stringent meters did elicit stronger passwords, they also required longer interactions and stirred greater resentment among users. Egelman et al. (2013) proposed a strength meter, then evaluated it using both a laboratory experiment and field study conducted over a much longer period for selecting a lower value password. Interestingly, while in the experiment users selected better passwords using the meter, in the field experiment on low-value passwords, the meters had no discernible effect. Finally, governments are now basing recommendations for password selection and use informed by the academic literature (NCSC, 2017).

What lessons can one draw from this brief survey through the passwords literature? First, that many methodological approaches are in use. Second, that structural observations can improve our understanding of a problem and produce better technology.

OVERCOMING: ETHICAL CONSTRAINTS A further concern is that experiments are impracticable in security for privacy and ethical reasons, rather than simply being impossible to design properly as the foregoing argument held. The ethical considerations of security studies have been traced in detail by the Menlo Report (Dittrich and Kenneally, 2012). In the biological

and social sciences, the Belmont Report established the ethical guidelines for experiment design; in the US and UK some of these guidelines have been put into law. Universities enforce these policies via review boards that oversee experiment design before the experiment can be run. The Menlo Report updates the three classic considerations – respect for persons, beneficence, and justice – for an inter-networked world. A fourth, respect for law and public interest, is added. Privacy plays in all four of these organizing principles.

Ethical restrictions are a basic part of research in many scientific fields. Neuroscientists cannot open up the brains of human test subjects and apply electrical shocks. Physicists should not wantonly release radioactive particles to test atomic phenomena. Virologists cannot test the spread of disease by releasing a pathogen in an airport and tracking its progress. All these fields make do by designing ethical observations that get to the necessary explanatory insights. A thorough update to these ethical considerations for ICT is available in the Menlo Report (Dittrich and Kenneally, 2012). Engaging with ethical review boards may slow down security researchers at first. But ethical review has not stopped other sciences, and it should not stop security.

There is some nuance to this privacy challenge to experiments, which is that participant data being private means that experiments are not reproducible; the data cannot be shared with other researchers. Using the language I will develop in Section 3.4.2, this is only a problem for rerunning statistical tests. In the other seven senses, the data is re-collected. If all the other artefacts for experiments are available, including the code to collect and analyse data, repetition, reproduction, and so on should be possible without knowing the original participants. And even then, in some cases the original data may be anonymizable. Therefore, the cost in terms of reproducibility for the researcher to comply with ethics and privacy appears manageable.

OVERCOMING: RAPID CHANGE The third critique on the tenability of structured observations in security concerns the longevity or generalizability of results due to the pace of technological change. The critique runs, roughly, that although experiments are plausible, their results are not useful because the results are outdated by the time they are compiled and published.

This critique rests on a combination of selecting the phenomenon of interest and a conception of who culturally is doing science. Some phenomena are more ephemeral than others, in the sense that the phenomenon only exists for a short time. The extinction of the dinosaurs was an ephemeral event in this

sense (Glennan, 2010). Ephemeral is not to be conflated with quick. Chemical reactions may happen fast, but the phenomenon of gunpowder exploding is a robust chemical phenomenon, for example. If one selects an ephemeral phenomenon of interest in security science, do not be surprised that the resulting explanation has short-lived relevance. Forensics and CSIR are especially likely to be interested in ephemeral phenomena because to investigate the specific details of past events is almost always to investigate ephemeral mechanisms. This focus is analogous to the difference between palaeontology and contemporary biology. Furthermore, since forensics is usually defined as applying science in the service of the law, it would be odd to claim there is no science of cybersecurity which is being applied in the course of digital forensics (Morgan and Bull, 2007).

The selection of ephemeral phenomena interacts strongly with who is considered a scientist in the security landscape. Beyond universities, many government organizations and private companies have security research labs. Even beyond staff with 'research' in their job description, the notion of engineering as satisficing to create artefacts and science as crafting generalized knowledge induces some interesting perspectives. A software developer writing code is building something, therefore engineering. But such classification blurs when the developer is trying to find the point in the code responsible for a bug, or conducting code reviews generally. Surely, the developer is following established practice, not inventing new modes of experiment (Oram and Wilson, 2010). But one does not say that a high-school student following textbook steps for a chemistry experiment is not participating in the scientific enterprise, just because the steps are known. The line blurs further for older students, following known procedures but with slightly varied chemicals to measure potential differences. Likewise, a developer may follow textbook steps for establishing a hypothesized source of a bug, intervene on the system, and measure to establish evidence for whether the intervention confirmed the hypothesis. But the code is new, so the answer is not known *a priori*.

I claim developers, or more likely in security, malware reverse engineers investigating the functionality of unknown code based on hypotheses, who operate in this mode participate in science. Therefore, the longevity-of-results picture changes. Experiments may have a short window of application, but cybersecurity experiments may also be quite quick to execute and apply. An experiment might be automated malware reverse engineering that runs the malware in a sandbox, extracts suspicious domain names and IP addresses

from connection attempts, and publishes those network elements to a blacklist. The time frame between the beginning of the experiment and operational impact on defender networks may be 15 minutes. Just because it takes the Food and Drug Administration (FDA) 15 years to approve a drug does not mean anything scientific takes years. The network elements may have a short lifetime of usefulness, but it is at least proportional to the duration of the experiment.

The critique that experiments in security are untenable because of rapid technological change takes an unacceptable combination of options – that these miniature experiments are not science, but that science must cope with highly ephemeral phenomena without such experiments. There is no reason to conceptualize science in this contradictory way. In either other conception of science's relation to automation and the expectations for its results, the challenge that rapid technological change makes it impossible simply evaporates. There may be methodological or generalization-based challenges to a science of cybersecurity so conceived. Chapter 4 will recognize such challenges and confront them directly.

STRUCTURED OBSERVATIONS VIA MATHEMATICAL MODELLING Another important purpose of structured observation is to support the construction of mathematical models of systems, perhaps incorporating representations of policies. The use of mathematical modelling in supporting the practice of science and engineering is well established. The process by which mathematical models are constructed is summarized by Figure 3.1. Observations of the world are performed; by a process of induction, the construction of a model is commenced; using deductive reasoning, the properties of the model are derived; those properties are interpreted in the observed world; and further (structured) observation of the world is used to assess the accuracy or utility of the model as constructed so far. This process of observation, construction, deduction, and interpretation is iterated until the modeller is satisfied that the constructed model is fit for purpose. This process by which mathematical models are constructed makes use of both inductive reasoning, for conclusions about the empirical world using observations and inferences from those observations, and deductive reasoning about the mathematical model itself. The process seeks to constrain each type of reasoning by reference to the other.

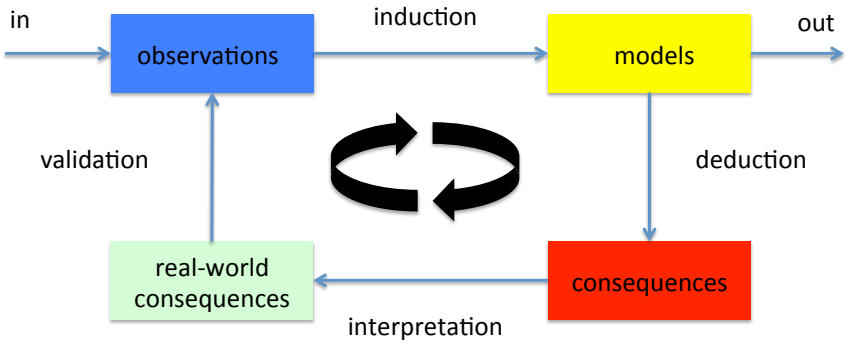

Figure 3.1: The mathematical modeling cycle

Both during the construction of a model and during the use of a completed model in an engineering process, two kinds of reasoning about the properties of models are used. First, models can be explored intensionally, in the style of experimental mathematics; that is, the space of evolutions of the model is explored systematically using simulation methods, such as Monte Carlo (Collinson et al., 2012a; Jain, 1991), and absolute and expected properties can be observed and deduced. For example, Caulfield and Pym (2015a) explores the consequences of different levels of resourcing for the effectiveness of access control to buildings and consequent information security breaches. Second, properties of the model can be explored extensionally; that is, the full range of logical and mathematical tools can be used to reason about the properties of the model considered as an object of mathematical study. This latter form of reasoning can be illustrated in the context of program verification by the set-up of Separation Logic, as I will describe in Chapter 6.

The mathematical modeling method applies in the science and engineering of security just as it applies, for example, in civil, electrical, and mechanical engineering and their supporting sciences.

3.4.2 *Evaluating Results*

CLAIM: REPRODUCIBILITY IS IMPOSSIBLE This complaint requires considerable unpacking. There are some intuitive challenges; for example, if reproduction of phenomena under controlled conditions is an absolute requirement, then astrophysicists and palaeontologists are not scientists. This complaint inherits a flavour from logical empiricism; reproducible experiments are necessary to repeatedly test and increase confidence in falsifiable

(but not yet falsified) statements of universal law. Fragile or contextual conclusions in biology – conclusions that were not readily reproducible – historically led to serious claims that biology was not a science (Cartwright, 1983).

Complaints of irreproducibility, at heart, strike out at the genuine observation that conclusions in cybersecurity research are often fragile or contextual. Philosophy of biology countered similar logical-empiricist attacks by creating a more nuanced idea of evaluating explanations and results. I will leverage this work about biology to do the same. Reproducibility is a complex term in itself; I will present no fewer than eight different senses of the term that discuss different aspects of evaluating evidence from structured observations.

ALTERNATIVE: EVALUATION TAKES MANY FORMS Although the distinction between replication and repetition is not new (Cartwright, 1991), recent work provides actionable advice to scientists. I focus on the family of five terms suggested by Feitelson (2015), plus the notion of statistical reproducibility from Stodden (2015). I will discuss three distinct senses of statistical reproducibility, for a total of eight distinct methods to support the robustness of evidence. When one complains that cybersecurity lacks reproducibility, usually what is meant is that one or two of these eight senses is impossible. All sciences similarly struggle to achieve reproducibility in all these senses at once. Thus, cybersecurity is no worse off than other sciences.

Feitelson (2015) suggests five distinct terms for a computer science discussion of evaluating results:

Repetition – to rerun exactly, using original artefacts

Replication – to rerun exactly, but recreate artefacts

Variation – repetition or replication, but with a measured intentional modification of a parameter

Reproduction – to recreate the spirit, in a similar setting with similar but recreated artefacts

Corroboration – to aim to obtain the same or consistent result as another study via different means

Each of these strategies has its uses, one is not strictly preferred over the others. This subsection uses an extended example of evaluation of Network Intrusion Detection and Prevention Systems (NIDPSs). One may question whether evaluating an NIDPS rule is scientific, in the sense desired. NIDPS rules may be very specific and not widely generalizable, but the same could be said for

determining whether a particular enzyme in the blood of some rare Amazonian fish actually selectively binds to some specific parasite. Chapter 7 will pick up on the example of NIDPS rules and how to integrate them into more general knowledge using the strategies from Chapter 4.

REPETITION Even in the restricted context of evaluating a single NIDPS rule, these strategies all have a sensible interpretation. Given a recorded stream of traffic, if one plays the stream back through the same NIDPS with the same network configuration, the same rules should fire. A failure of repetition would be indicative of race conditions or performance bottlenecks, or perhaps an architecture with multiple distinct NIDPS systems and a network switch that randomly assigned packets, meaning any rule that required correlation between packets would not reliably fire across repetitions. It is because of repetition experiments such as this that Bro, Snort, or Surricata work on flows, not packets. When load-balancing any of these tools, traffic is balanced based on flows, so that all packets in one conversation go to the same thread. Flow-based balancing is needed because a NIDPS works to reassemble application-level, or at least transport-layer, information (Scarfone and Mell, 2007). Any network architecture that randomized packets among instances, as opposed to maintaining flow-based balancing, would lead to unrepeatable observations because application-level reconstruction would fail to be the same, and the NIDPS tests would be applied to different information.

REPLICATION Replication might mean to use different NIDPS software and re-write the rule in its language to get identical detection. Replication might also mean using the same NIDPS architecture and rules on traffic that is artificially generated to have the same malicious features as a prior experiment; or perhaps it is simply to recreate the same network architecture at a different physical location and replicate that the NIDPS works on a pre-recorded traffic stream to provide evidence the newly setup sensor architecture has been installed correctly.

I have claimed that creating a new NIDPS rule and testing it on a pre-recorded traffic stream is an experiment. Am I abusing the use of experiment here, in a way that is not consistent with other sciences? No. Perhaps strangely, the objects of the experiment are artefacts, and software artefacts at that. But if cybersecurity is a science of anything, certainly it is of software (and how people interact with it). Therefore, the NIDPS analyst making a rule has

an untested specimen (the NIDPS rule), a hypothesis about how it should behave (what she designed it to do, in this case), and establishes a controlled environment in which to test the properties of the specimen of interest. This matches all the usual hallmarks of an experiment.

From repetition and replication, variation is straightforward and I will not discuss it in detail. For an NIDPS it is basically making a small change to a signature and observing how the results change.

REPRODUCTION AND CORROBORATION Reproduction of NIDPS alert efficacy is something practitioners measure often; with the same rule in a different network architecture, they evaluate the outcomes. Corroboration is similarly useful. Perhaps against different traffic patterns, do different NIDPS systems see similar but distinct alerts about exploitation of Heartbleed, for example, that allow corroboration of wider claims about internet-wide abuse of the vulnerability.

STATISTICAL PROPERTIES Stodden (2015) distinguishes three failures of statistical reproducibility. Lack of statistical reproducibility may result from poor design of observations, namely issues with low statistical power or sampling bias. Even if the sample design is appropriate, the statistical tests applied may be inappropriate. For example, a test may be applied despite requiring assumptions that the situation does not meet. Finally, results may be generalized beyond what statistics justifies. A rough translation of these three problems is:

- design of observations
- analysis of observations
- interpretation of observations

If an observation is poorly designed, it will not be able to be reproduced. Poorly designed means either there is a consistent, unknown confounding factor in the design (sample bias) or that the number of elements is not large enough to produce results independent of natural random variation. Sampling bias happens quite naturally across various organizations that might deploy the same NIDPS rule – different adversaries attack banks, militaries, and universities. In security, this reduces to a challenge of comparing like with like. In medicine, it is understood that patients from different age groups or income brackets have different health outcomes. But these differences are measured,

and then can be controlled for when observing the impact of a treatment on a varied population. The sampling bias is controlled by knowing the shape of bias with sufficient precision. Security suffers from an additional layer of evidence-gathering bias. Organizations may not report or disclose vulnerabilities, for example, due to ignorance, fear of financial risk, legal obligations, or to improve the reputation of their brand (Shostack and Stewart, 2008, p. 52ff). Such social and cultural biases apply to many types of security evidence. Adequately understanding these biases, and how to mitigate them, remains an area for further work.

The statistical analysis and tests performed after data are collected can also impact whether a compatible result can be obtained in any of the five strategies for confirming results. One common problem is for researchers to selectively report results, and even tests performed. If a researcher runs "many tests until one of them yields a p-value of less than 0.05 and then report[s] this result as if the other tests had never been run" then the result is actually a statistical outlier, even though it is being reported as a reliable result (Stodden, 2015, p. 6). The researcher essentially misrepresents the robustness of the result, which of course impacts confirmation attempts. Such statistical manipulations are well-documented to result in publication bias in psychology journals, for example (OSC, 2015). A science of cybersecurity must guard against such misapplications of statistics just like any other science.

The final area of statistical reproducibility to discuss is the data collection and generation process. Problems with data generation lead to a lack of generalizability of the results. For example, studies commonly report a linear regression establishing a relationship between two measurements, such as deploying an NIDPS rule and the number of intrusions. Unless the data generation is very strictly constrained, one may not safely use that relationship anywhere outside the data collected – the result may not be generalized (Stodden, 2015, p. 6). Generalizability is a problem in cybersecurity. However, to the extent that this challenge is due to failure to meet known statistical constraints on data collection, cybersecurity is not distinct from other sciences. Following this line of reasoning, Chapter 4 will draw on other sciences for strategies on how to build generalised knowledge in cybersecurity.

FORENSICS AND REPRODUCIBILITY One intuitive objection on reproducibility stems from the idea that security is forensic, and so necessarily considers single events. By definition, if some security event only happened

once, it cannot be repeated. Forensics or history may be different from science (see Section 3.4.5); however, there is not a sharp distinction. Establishing evidence for a particular series of events at a particular time in the past is forensic. Sometimes sciences require such evidence as part of knowledge creation, generalization, and application to new scenarios. It seems implausible to claim that astrophysics, palaeontology, and macroeconomics are unscientific. Yet they are largely historical, in a way similar to CSIR.

A science of security can integrate forensics in an analogous way to how biology integrates palaeontology and physics integrates astrophysics. Palaeontologists build evidence for the mechanism of what killed the dinosaurs, for example. Glennan (2010) refers to these one-time events of interest as being driven by "ephemeral" mechanisms, as opposed to say chemistry where mechanisms have a more robust, repetitious nature. The reason for unifying ephemeral mechanisms, as studied in palaeontology, with other sciences is because mechanism discovery strategies and mechanistic explanation provide a coherent account of scientific activity. Bringing palaeontology into that fold brings the philosophical account of scientific explanation via mechanisms in line with the obvious similarities between astrophysics and palaeontology on one hand and physics and biology on the other. Biology is not unscientific because it contains a sub-field focused on single events in the past – palaeontology; similarly, a science of cybersecurity is not totally scuttled simply because it contains a sub-field that focuses on forensics.

SUMMARY The thrust of this argument is that observations in cybersecurity appear, to a large extent, to be amenable to the various senses of reproduction. I frame this issue under the task of evidence evaluation. However, I did not attempt to refute the fragility of conclusions in cybersecurity, as this fragility appears genuine. The question can be more productively answered by selecting the correct level of analysis than naïvely insisting against reproducibility.

Questions of reproduction skip the important issue of what phenomenon is to be reproduced. Which phenomenon is of interest will impact which evidence evaluation strategies (repetition, statistical tests, etc.) are most valuable. Scientists are often interested in discovering the mechanism responsible for a phenomenon in order to better explain the phenomenon. Defining the phenomenon differently will change the mechanism of interest. For example, most models of computer network attacks have a step for exploitation of the target.

Attacks are a high-level phenomenon and exploitation is one activity within the mechanism. At a lower level, there are various mechanisms by which exploitation could occur, for example drive-by downloads or social engineering to run untrusted code, as I will discuss in Chapter 5.

3.4.3 *The nature of scientific inquiry*

CLAIM: NO LAWS OF NATURE The critique that there is no science of cybersecurity until there are laws of security, or mathematical rules which permit deductions from observations to consequences, comes presumably from the received view that this is how physics works. The importance of laws in the received view is to provide predictive power. However, to claim laws are necessary to make reliable predictions is unsupportable. Many sciences make reliable predictions without laws; in fact, many philosophers argue physics does not have laws in the sense commonly understood. Yet, many predictions of physics have a reliability that a security researcher would envy. This section introduces philosophical perspectives on how to create and evaluate predictive models.

Some science of cybersecurity work has noted that not all sciences have laws. The USArmy Research Lab is more thorough, pragmatically defining a science of security by specifying its subject matter, and taking models generally as central (Kott, 2014). However, no prior work goes nearly far enough in breaking the preconception about laws nor providing alternatives. First, biology has faced and refuted this conception of laws being a necessary criterion for science in detail; I will provide a brief summary. Second, I adapt to security a modern conception of explanation that has supplanted that of laws – mechanistic explanation.

First, let us summarize what a laws-based explanation is intended to mean. The DoD takes laws as "the basis of scientific inquiry" (MITRE, 2010, p. 4). Hempel provided a concise definition that is a useful historical basis: a law is "a statement of universal conditional form which is capable of being confirmed or disconfirmed by suitable empirical findings" and a law, as opposed to a hypothesis, refers to such a statement that "is actually well confirmed by the relevant evidence available" (Hempel, 1942, p. 35). This 1942 definition has all the seminal features; for example, Popper's falsifiability criterion in different words ("capable of being disconfirmed"). I will push hard on this logical-empiricist, laws-based conception of explanation with a detailed unpacking of

the meaning of the term followed by critiques by the modern philosophers of science Mitchell, Cartwright, Bogen and Woodward, and Woodward following Pearl.

To understand 'law' in this sense one must focus on the technical-philosophical meaning of three terms: universal, conditional, and capable of being confirmed. Universal means that the statement of law applies to all things, at all times, without exception or additional precondition. Conditional means an if-then statement in classical first-order logic. For a statement to be capable of being confirmed or refuted, it needs to have semantic content, or meaning, and be about the universe in which we live. One challenge to this view is captured by the aphorism 'all models are wrong, some are useful.' While models and statements may be semantic, they are also necessarily simplifications of our universe and there are always conditions in which that simplification is wrong. But a simplification cannot be confirmed or refuted. Simplifications may be useful or useless, in various contexts, but that is a far different thing than absolutely refuted, absolutely true or false. Many logical empiricists side-stepped such questions by saying that laws were true universally, so they must be independent of human artifice or language. This history makes it particularly strange to ask whether there are 'laws' of man-made system security.

Mitchell, a philosopher of science, has deconstructed a laws-based unity of science. For example, she argues that "nature is complex and so, too, should be our representations of it" and that complexity and the resulting pluralism of models "is not an embarrassment of an immature science, but the mark of a science of complexity" (Mitchell, 2003, p. 115). She advocates that the relevant aspect is not how theories are defined, but used. Thus, any theory that functions as an effective generalization might pragmatically be called a 'law' in physics, or a 'model' in biology. What is important is effective methods for generating and scoping generalizations so we know where and to what they apply.

Cartwright identifies various problems with laws-based conceptions. These include that usual laws-based conceptions cannot make sense of causal statements, and that laws explain the behaviour of mathematical models of the world, rather than the world directly (Cartwright, 1983). She calls this a simulacrum account of explanation.

Further deconstructing the logical positivist influence on philosophy of science, Bogen and Woodward (1988) identify the mediating influence of data

and observation on our ability to make theories. People tend to care about the phenomena in the world, and theories apply to phenomena. However, people observe data, and data collection is mediated by tools and their norms of use. The canonical example is the melting point of lead. No one observes lead melting at 327°C. Observers note many thermometer readings, with both known and unknown equipment errors and limitations, from which they statistically derive the value 327°C with some acceptably small margin of error and assign it to the phenomenon of lead melting. The argument goes on, with some nuance, that there is no law or even single theory that explains lead melting at this temperature, much less any of the individually-observed thermometer readings. The whole apparatus of experiment, including the engineering of the tools, the statistics, the metallurgical purity of the sample, and so on, are all necessary to explain the phenomenon. This perspective accords with Mitchell's proposition that complicated theories are a sign of maturity, not immaturity; as well as Cartwright's simulacrum account of explanation.

Woodward provides an alternative account of causal explanation to supplant a laws-based account. This account is known as an interventionist account because, roughly, it is based on the idea that the only way to determine causation is by an intervention that could, in practice or in a thought experiment, make a change (Woodward, 2003). Woodward relies on Pearl's statistical account of causality (Pearl, 2009), which has been updated since Woodward's treatment but with some modification is a sound statistical approach and language for discussing causation (Dawid, 2010). Causal explanation is not about laws of nature, but about building an explanation of the organization of elements of a phenomenon in such a way that one may intervene reliably on the outcome by changing the elements. Again, like Mitchell, there is a theme of what makes an adequate generalization of a system.

ALTERNATIVE: MECHANISTIC EXPLANATIONS OF PHENOMENA Herley and van Oorschot (2017, p. 14) recommend "physics-envy is counterproductive; seeking 'laws of cybersecurity' similar to physics is likely to be a fruitless search". This statement is true, but also naïve in two senses. First, as discussed above, physics does not have strict laws any more than biology, security, or economics. Second, it does not provide any viable alternative goal of scientific enterprise, and therefore no expression of what sort of generalization of knowledge I am seeking for security. I will sketch out

what tools philosophy of science provides here and Chapter 4 will focus on this question in more detail.

The most convincing account within the philosophy of science community is that of a mechanism (Glennan, 2015). However, this word must be divorced from a mechanical, Victorian interpretation as the world analogous to a machine. The current consensus definition is that "a mechanism for a phenomenon consists of entities (or parts) whose activities and interactions are organized so as to be responsible for the phenomenon" (Glennan and Illari, 2017, p. 2).[6] This mechanistic conception of scientific reasoning is useful because it provides a structure to build on and borrow mechanism discovery strategies.

The literature on mechanism discovery strategies examines how scientists develop hypotheses to test, and the constraints they build into experiments and observations in order to test them. Mechanistic thinking is more helpful than a laws-based approach because it provides hints as to what to do and what to look for in order to build a useful explanation. Bechtel and Richardson base their initial strategy of decomposition and localization on Herb Simon's work. Their work is specifically positioned as a strategy for scientific discovery "well suited to problems that are relatively ill defined, problems in which neither the criteria for a correct solution nor the means for attaining it are clear" (Bechtel and Richardson, 1993, p. 7). This description is encouraging for security practitioners beset with ill-defined problems.

The other main contributor to the mechanism discovery literature is Darden. She provides strategies used on slightly better-defined problems. If a general mechanism is known, but details of a particular entity or activity are hazy, a reasonable hypothesis is to attempt to resolve the hazy element more clearly ('schema instantiation'). If a mechanism is understood, but not its set-up or termination conditions, the investigator can chain their reasoning either backwards or forwards from the known mechanism to constrain our hypotheses about what must exist either before or after the known mechanism (Darden, 2006, ch. 12).

Hatleback and Spring (2014) discuss and resolve the apparent difficulty of discussing mechanisms that have been engineered, or created by humans, such as computer code. This different origin does not present any conceptual difficulty in understanding the function of the code as a mechanism. Like with

6 Compare to Bechtel and Richardson (1993), Illari and Williamson (2012) and Machamer et al. (2000).

any differing fields the exact tools used to examine mechanisms in computing would not be identical to biology, any more than radio telescopes for stars are useful in investigating mechanisms of frog ecology. Chapter 5 presents incident response and intrusion analysis as a kind of mechanism discovery task. I will show the heuristic of schema instantiation, from Darden (2006), to be analogous to incident analysis heuristics. Specifically, the heuristic an incident responder uses when resolving the 'exploitation' step in the kill chain to a drive-by download, for example, and then again when the drive-by download mechanism is instantiated to clarify what particular domains and web services participated in the exploitation and could be blocked.

Science of cybersecurity stands to benefit from refusing to consider explanation as laws-based and instead focusing on scientific investigation as mechanism discovery. The question of whether there are laws of cybersecurity is fundamentally the wrong question to ask. Both philosophy of science and mathematics have better perspectives on generating intelligible explanations of phenomena than a laws-based explanation. The modern philosophy of science literature provides heuristics for mechanism discovery that should be helpful in orienting scientific investigations in security. The mathematical modelling cycle described in Figure 3.1 provides a complementary heuristic process.

When using mathematical models, the situation is clear. Practitioners do not identify laws but rather properties of models. The properties of a model are used to assess its value and to support its refinement.

3.4.4 *Scientific Language(s)*

CLAIM: NO SCIENCE WITHOUT A COMMON LANGUAGE In this section I will argue against the idea that a single language or ontology of security would be a defining feature of a science of cybersecurity, although clarity of expression is necessary. Our main departure point is the insistence on unification into a single language, and advocate instead for a kind of integrative pluralism – which is what actually exists in other sciences anyway.

JASON identifies a "common language" as the "most important" attribute necessary for a successful science of cybersecurity (MITRE, 2010, p. 3). The other statements I reviewed are less direct, but there is a similar drive towards unification. Phrases that appear sympathetic to unification of language and explanations include "contribute to a general framework" from the NSA defin-

ition, "comprehensive theories" as opposed to "domain- and context-specific" ones in the White House plan, and the negative tone in which Geer relays the diversity of opinion among the best paper judges.

The implicit assumption amongst these statements is that, at least within security, a variety of disparate phenomena can be reduced to a relatively compact set of definitions and statements. Oltramari et al. (2014) cite the JASON reasoning and make this desire to compact the semantic space explicit. The traditional statement of unification into a common language was 'reductionism,' as discussed in Section 3.2 to be logical deduction of one field's laws to those of another (Nagel, 1979).

This reductionist idea of a common language implies an explanatory hierarchy. It seems more realistic to admit that explanations are about different topics, and each topic develops its own specialized language. A variety of fields contribute understanding to security mechanisms, from economics to electrical engineering to elliptic curves. Translations between these fields will be vital. But that does not create a common language any more than translating between German and French creates a combined language; it just creates a translation.

ALTERNATIVE: TRADING ZONES, PLURALISM, AND MOSAIC UNITY A more convincing account of interdisciplinary collaboration comes from physics. Galison, a historian of physics, borrows the anthropological term *trading zone* to describe the contact between subfields of physics, such as experimental and theoretical particle physicists (Galison, 1999, 2010). The analogue is in merchant towns, cultural anthropologists observe people from different cultures coming together and creating just enough of a shared language such that commerce can happen. As commerce grows and the communities benefit, the trading zone becomes more robust. This occurs linguistically as well as materially.

Galison's insight is that the same process happens between subfields of physics. There are members of each community who specialize as traders, go to places where the communities interface, and develop specialized languages (pidgins, creoles, etc.) that are incomplete mash-ups of each traders' home language. Theoretical physicists and experimental physicists do not, in fact, speak one common language. They speak importantly different languages, with their own jargons and evaluations of what features of explanation and evidence are most important. However, the two subfields can productively

exchange ideas because there is a developed trading zone where ideas can be translated into a shared language from both directions and then re-translated back out to the respective communities.

Galison's insights on trading zones and subcultures seem to have been misunderstood by JASON. In his presentation about science in cybersecurity, Galison writes "in these choices of basic objects and their relation lie the basis of separation of subcultures and the robust power that that division offers" (Galison, 2012, p. 20). Partiality of language is how Galison envisions these various subcultures of security communicating, just as in physics. The sense in which security needs a common language is that it need various trading zones in which security researchers can communicate. The idea is emphatically not to wait for a single common language to emerge with which all then speak unambiguously. It is not that such a goal may be slow or arduous to achieve; more importantly, it fundamentally undermines the robust power that division in specialities offers.

In biology, Mitchell (2003) has argued against reductionism for what she calls *integrative pluralism*. For Mitchell, the units of science here are theories, understood as idealized models of various phenomena. There are many models, hence 'pluralism,' and models are neither totally isolated from each other nor usually reducible so that one supplants another (Mitchell, 2003, p.192). Since each model comes with its own terms and specific usage of terms, if anyone could produce a common language that would be tantamount to unifying all our theories within security. Integrative pluralism, as a model of biology at least, indicates this unification of terminology is not possible except in localized, purpose-built contexts – that is, trading zones.

The most convincing analogue for defining a field of science of cybersecurity is the *mosaic unity* of neuroscience (Craver, 2007). The subfields making up neuroscience collaborate by adding constraints, based on their own individual methods and viewpoints, on mechanistic explanations. Like the stones in a mosaic, each subfield has its own unique structure and identity, but if one steps back each stone contributes to a bigger picture.

Security has a similar arrangement of diverse fields contributing constraints on explanations. Economics constrains explanations of what users can be asked to do based on how they spend their money in situations of information asymmetry. Usable security constrains explanations of what users can be asked to do based on how they spend their attention. Craver is more helpful than Mitchell for security in that, for Craver, the explanations are

mechanistic explanations, and that structure allows him to elaborate on how the subfields are interrelated and provide constraints. Different fields work on different parts of a mechanism or on a different level of a mechanism. There are a plurality of mechanisms, which are idealizations as theories are for Mitchell, and mechanisms are integrated via levels of explanation. I will return to Craver's concepts in Chapter 4.

These three related conceptions of communication and explanation in science all go against the need for a common language. All three also support the importance of clarity of expression. If context is important, because there are a plurality of theories and contexts, it is vital for researchers to make their assumptions clear and use terms consistently. Pluralism is not an excuse for sloppiness of explanation. If anything, it militates for the importance of thorough, careful explanation. Perhaps the push for the importance of a common language in security is actually a push for clarity of expression. Certainly, methodology sections are vital. Maxion (2015), for example, has argued strongly for structure and clarity as an aid to good science. However, structure and clear expression are a separate problem that should not be conflated with the idea of creating a common language before science can commence.

3.4.5 *Engineering or Science?*

CLAIM: SECURITY IS 'JUST' ENGINEERING One might ask, if the goal is to produce reliable systems, why discuss science at all? Producing systems to a certain standard of usefulness is engineering. While engineering certainly leverages scientific knowledge, it also uses other kinds of knowledge (Vincenti, 1990, p. 229). Indeed, the government call for a science of security looks similar to Anderson's description of security engineering:

> "Security engineering is about building systems to remain dependable in the face of malice, error, or mischance. As a discipline, it focuses on the tools, processes, and methods needed to design, implement, and test complete systems, and to adapt existing systems as their environment evolves" (Anderson, 2008, p. 3).
>
> "[Security engineers] need to be able to put risks and threats in [context], make realistic assessments of what might go wrong, and give our clients good advice. That depends on a wide understanding of what has gone wrong over time with various systems; what

> sort of attacks have worked, what their consequences were, and how they were stopped (if it was worthwhile to do so)" (Anderson, 2008, p. 5).

I resist both the assertion that there is no science to be done in cybersecurity as well as the pejorative connotation of placing engineering somehow below science. Sciences that study technology or human artefacts do have their own nuances and distinct concerns (Meijers, 2009), but this is no reason to collapse security entirely under engineering any more than it would make sense to collapse medicine entirely under engineering just because it is not pure biology.

Models, mathematical or conceptual, are a key way science is transmitted to engineering practice. Chapter 6 will explore the extent to which engineering, logico-mathematical, and mechanistic models overlap in computer science. Explaining an incident likely considers a model of the computer system and explores what sequence of events, as represented in that model, led to a certain observed outcome, as represented in that model. But in order to fix the impacted system, the model and explanation need to be transmittable to and usable by the security engineers. Chapters 4 and 5 will contribute to how to make conceptual models more clear, transmittable, and usable. Chapter 7 will contribute to a logico-mathematical model of CSIR. Thus, in many ways, this thesis is centred around enabling multiple modes of communication between scientific explanation and security engineering – arranged around the understanding in the following 'alternative' that there are no sharp dividing lines between these disciplines.

ALTERNATIVE: SCIENCE, ENGINEERING, AND FORENSICS

Efforts have been made to draw boundaries and classify the relationship between engineering and science. In the two extreme viewpoints, Simon (1996) tries to subsume engineering under science as a science of design, and Koen (2003) tries to subsume science under engineering as a special case of heuristic problem solving. There are also more nuanced views. Engineering as usually practised generates understanding and depends, in part, on science (Vincenti, 1990). At the same time science as usually practised generates artefacts and depends, in part, on engineering (Dear, 2006). Therefore, it seems unjustified to place science over engineering or vice versa. If people are going to engineer secure systems, in the sense of Anderson (2008), we will need a science of cybersecurity to extract and generalize the knowledge with which to build. Both building artefacts and extracting knowledge have their

own challenges, making it sensible to distinguish the tasks. Security should continue the traditional, close interplay between science and engineering.

Dear (2006) exposes a duality within science: between natural philosophy and instrumentality as twin, mutually indispensable, explanatory strategies for making nature intelligible. This duality blurs a divide between science and engineering. A more detailed exposition of the relationship between science, engineering, and forensics is left as future work. But as an example, I am sympathetic to Leonelli's conception of scientific understanding, which embraces this blurred duality between science and engineering:

> "Understanding can only be qualified as 'scientific' when obtained through the skilful and consistent use of tools, instruments, methods, theories and/or models: these are the means through which researchers can effectively understand a phenomenon as well as communicate their understanding to others" (Leonelli, 2009).

Clearly science, forensics, and engineering interact tightly. When systems break, practitioners conduct forensics to learn why and how. They then employ science to update knowledge, improve models, or document edge-cases based on this why and how. Adequate updates may include further, purpose-designed structured observations. Practitioners then employ engineering to adapt this new knowledge to build a better system, less likely to break. Thus, a feedback loop from engineering to forensics and science back to engineering which contains no sharp distinction between a science of cybersecurity and security engineering. This thesis will focus on the scientific enterprise,where science is understood as generalized-knowledge, evidence-based, explanation-generation activities.

3.5 A SCIENCE OF SECURITY VERY MUCH EXISTS

My argument has been that security is, as practised, a science with its own unique challenges. This statement contrasts with the surveyed views, which posit that whatever makes security hard also makes it a qualitatively different sort of enterprise than science. These detractors often accidentally overemphasize some scientific field in conceiving science generally. Of course security is not particle physics, nor molecular biology. This conception of science is too narrow. This overly-narrow view can, in many cases, be traced back to outdated views related to logical empiricism.

The common complaints against a science of security are: experiments are impossible, reproducibility is impossible, there are no laws of nature, there is no common ontology of terms, and it is 'just' engineering. I forwarded alternative perspectives on all these complaints that already accommodate security: structured observations of the empirical world, multiple methods for evaluating evidence, mechanistic explanation of phenomena, specialization necessitates scientific translation, and the interplay between science, engineering, and forensics.

Cybersecurity suffers from the same sorts of challenges as other sciences. It is not qualitatively different. However, different fields of science are defined, largely, by the characteristic challenges of their subject matter and how those challenges are approached. Cybersecurity must learn from challenges common with other sciences while at the same time pushing forward with novel solutions to those challenges and approaches unique to cybersecurity. Where this chapter suggested existing solutions to challenges shared with other sciences, Chapter 4 will attempt to overcome some challenges specific to security.

Also like other sciences, a science of security faces important social questions. Three possible directions are the gap between research and practice; the customers or recipients of knowledge produced by a science of security; and how the secretive nature its practice alters the science being done. Dykstra (2015) and Metcalf and Casey (2016) attempt to narrow the knowledge gap between practitioners and scientists; but the nature and social function of the gap should also be studied. Some customers are policy makers; future work would likely build on Jasanoff (1990). Perhaps some knowledge customers are practitioners, but as Vincenti (1990) argues, academia also receives knowledge from practitioners. Systemic secrecy has caused different scientific practice in the case of biological weapons development (Balmer, 2013); something related may happen in cybersecurity. An example question is how students with a classified PhD thesis might differ from those with a publicly published thesis.

It is less important to quibble over whether cybersecurity is a science than it is to lay out a satisfactory decision-making process for studying cybersecurity. It is certainly the case that cybersecurity has moved to the forefront of societal concerns. I seek to move past the debate over science or non-science. A better concern is to identify robust decision-making and evidence-gathering tools that enable satisfactory results within a topic of crucial social importance.

I view challenges in security as challenges to building generalized, shareable knowledge. In many cases, the science of cybersecurity community has hinted

at this conception of the challenge. Generalization is woven through the discussions of the difficulty of confirming observations, designing experiments, and developing a common language. Generalization is implicit in the discussion of laws because, traditionally, laws are a formal vehicle for expressing generalizations. These descriptions may accurately identify that generalization is hard in a science of cybersecurity, but the diagnosis of the cause of this challenge misses the mark, as Section 3.4 demonstrated. This conception of generalization as the core problem of a science of cybersecurity works with my tentative framework of engineering-science-forensics. Engineering and forensics are about applying knowledge or discovering particulars, whereas science is the aspect of security concerned with abstracting knowledge. Justified generalization is also the key to accurate prediction.

A community can tackle the problem of building general, shareable knowledge better if the problem is seen clearly for what it is: a problem shared by all sciences, with particular strategies more or less transferable between fields depending on the details of what a field studies. Part of the solution is to integrate other fields into security, as advocated by security practitioners such as Shostack and Stewart (2008). But simply bringing in new perspectives is not enough. Morgan (2014) argues that generalization, which she handles under the umbrella of resituating knowledge, is hard because knowledge is always produced locally. Knowledge transfers must be painstakingly warranted.

3.6 RESEARCH PLAN

Chapter 2 identified "how to satisfactorily make clear and explicit the reasoning process used by individual CSIR analysts" as the research question. Given the state of the art in the science of/for security literature, I propose the following research plan. In order to improve CSIR analysis I need to know what knowledge ought to look like and what reasoning ought to look like. At present, there is not a satisfying answer that is applicable to CSIR. The main task of the thesis will be to build credible accounts of knowledge and reasoning applicable to CSIR (which may apply in security more generally). Throughout, I will accompany this construction task with examples and demonstrations of cases where the construction is an aid to good knowledge creation or reasoning.

These research tasks will be localised in the following chapters. Chapter 4 will build an account the structure of general knowledge in security and some examples of building it successfully. Chapter 7 will build a logico-mathematical

model for reasoning in incident analysis. Chapter 5 will work through several examples of incident analysis to both demonstrate the use of my account of the structure of knowledge as well as gather case studies of reasoning in CSIR to inform the logic definition. To further inform my logic specification, Chapter 6 will investigate how heuristic and hypothetical reasoning has been formalised in program verification and what features of a logic make it more likely to succeed at scale.

Part II

GENERALISING, APPLYING, AND FORMALISING KNOWLEDGE

The goal of the next part, in three chapters, is to elucidate qualitative processes for incident analysis as well as understand the design goals and constraints for a logic to describe incident analysis. Chapter 4 reviews the present understanding of generalised knowledge in philosophy of science and takes three case studies of building such knowledge at different scales in cybersecurity. The result will allow a structured account of what to look for when building general knowledge. This structure provides various heuristic hooks for hypothesis generation, filling in gaps, and noticing errors that will be useful both to the current incident analyst and to logic design. Chapter 5 will apply this structured knowledge to some case studies of incident analysis to show how general knowledge is improved and used. These case studies also elicit some components and content that any incident analysis logic is going to need to be able to incorporate. Chapter 6 is an historical analysis of a logical system used in program verification – Separation Logic. This chapter will draw out the technical design and formal requirements that make a logic usable for reasoning at scale, and the way hypothesis generation heuristics can be incorporated in such formalizations. These technical features are what I will adapt to work with the incident analysis content identified in prior chapters.

4

GENERAL KNOWLEDGE OF MECHANISMS[1]

Scientists from many disciplines explain phenomena mechanistically. Different accounts of mechanistic explanation have been offered in the philosophy of science literature, leading to the emergence of something like a core consensus view referred to as 'minimal mechanism' in Glennan and Illari (2017): "A mechanism for a phenomenon consists of entities (or parts) whose activities and interactions are organized so as to be responsible for the phenomenon."[2]

Within philosophy of science the mechanisms literature actually exists as two largely parallel literatures, one studying mechanistic explanation in the life sciences, broadly construed (Bechtel and Richardson, 1993; Glennan, 2005; Machamer et al., 2000), while the other studies the social sciences (Elster, 1983; Kincaid, 2011; Steel, 2008). This chapter will begin to explore how to extend this interesting work to a major branch of science that has been largely neglected by the mechanisms literature: computer science.[3]

There are some exceptions to this general neglect. Such papers as Piccinini (2007) examine what it means for a process or mechanism to be characterized as a computation, Floridi et al. (2015) consider the notion of malfunction of a computation, and Angius and Tamburrini (2017) discuss explanation of the behaviour of computing systems. The impact of information security practices on discovery in medicine is discussed by Tempini and Leonelli (2018), but the focus is data curation in medicine. Galison (2012) has broadly stated

1 This chapter is based on joint work, the paper: Jonathan M Spring and Phyllis Illari (2018a). 'Building General Knowledge of Mechanisms in Information Security'. In: *Philosophy & Technology*. DOI: 10.1007/s13347-018-0329-z.

2 Compare Craver (2007), Glennan (2017) and Illari and Williamson (2012).

3 There has been a heated debate on the status of computer science as a science, in addition to the debate over status of security documented by Chapter 3. Tedre (2011) concludes in his survey of the discussion of the disciplinary status of computing by stating "there is nothing wrong either in considering computing to be a science, or considering it to be something else. Each of those arguments just assumes a specific viewpoint about both science and computing" (p. 382). Similarly, Tedre and Moisseinen (2014, p. 8) argue from their survey on experimentation in computing that philosophy of science should attempt to understand computing in its own right, rather than inflict idealized views of scientific methods from other fields on computing. Chapter 3 argued specifically that cybersecurity is a science. In moving forward, this thesis will follow the conclusions of these surveys and treat computer science, and cybersecurity in particular as a sub-field, as scientific disciplines from which philosophy of science can draw and apply lessons.

knowledge in "Manichaean Sciences" such as cybersecurity should be local and intercalated, and result from trading zones. Quite different from these, the focus here will be how computer scientists build more general mechanistic knowledge.

4.1 INTRODUCTION

For concreteness, I restrict the work on mechanisms to examples from cybersecurity,[4] which is the subdiscipline of computer science concerned with the "measures that implement and assure security services in information systems, including in computer systems and in communication systems" (Shirey, 2007).[5] This scope is, as in Chapter 3, broader than Computer Security Incident Response (CSIR). One case study will be a CSIR example, so I am confident that demonstrating how general knowledge can be built in cybersecurity also demonstrates how it can be built in CSIR. Furthermore, the knowledge that an incident analyst must apply may well be broader cybersecurity knowledge about system or adversary operations, so a broader account of general knowledge in cybersecurity is needed to support human decision-making in CSIR.

This chapter demonstrates how general knowledge is built using three case studies, which are interrelated in interesting ways. Three cases cannot illustrate how knowledge is built in all of cybersecurity, much less all of computer science. Indeed, there may be different ways of building more general knowledge, and there may be cases which are too difficult to build anything very general. Nevertheless, there are three contributions to the philosophical literature. These are cases where general knowledge is built via methods wildly different from the majority of cases studied to establish previous philosophical theories of knowledge. Therefore, the well-studied philosophical methods Chapter 3 identified as often useful do not appear to be adequate in these cases. In this philosophically unstudied domain, knowledge can be built in a way that is not well accounted for in existing philosophical approaches to knowledge.

4 Recall that my use of cybersecurity emphasizes that human users and the physical world are part of the socio-technical system under study, alongside computers or cyberspace. This definition is given in Chapter 1, and is a superset of the RFC 4949 definition of information security (Shirey, 2007).

5 Example services are confidentiality, integrity, and availability. Information security works entirely in the space between confidentiality and integrity on the one hand and availability on the other. One common quip is that the best confidentiality and integrity protection for your computer is to turn it off, unplug all the wires, encase it in concrete, and drop it to the bottom of the ocean. Availability is the necessary counterbalance.

Further, the mechanisms literature provides positive insights into how general knowledge is built in these difficult cases. Whether the mechanisms approach will generalise to further cases in computing is an area for future work.

Cybersecurity faces a novel constellation of challenges that make it particularly worth philosophical study. I select three interlocking challenges with significant impact on both research and methodology. First, the immediate object of study, namely software, can change behaviour during observations or between them; second, practitioners face active adversaries that respond to, and try to confound, observations; and, third, secrecy even among friendly practitioners is common because successful strategies need to be hidden from attackers, and participants need to protect their other interests.[6]

Computer science has a theoretical basis, and the logical basis of computers and computation may suggest that there should be a logical, a priori answer as to what a program does and whether that behaviour is secure. However, this is true neither in principle nor in practice. Turing (1936) famously proved that it is in principle not possible to calculate a priori whether a computer program will halt.[7] If one cannot determine if a program will halt, then one cannot determine how many resources it will use, and therefore how many resources to allocate to it. There are many exhaustible resources, including RAM, processor cycles, disk space, processor thread identifiers, and file system identifiers. If a computer runs out of a resource, at best it stops responding to new requests. At worst, a computer may run out of the resources to remain stable and crash in a way that makes it vulnerable to adversary attacks.

For extremely small systems, in-practice heuristics and cautious overestimation can overcome this principled hurdle. However, any computer system actually in use has a complex supply chain for both hardware and software that cannot be considered a small system. Furthermore, security in-practice is a risk assessment: balancing costs and benefits, and balancing availability with confidentiality and integrity. Appetite for risk impacts productivity and profits; there is no uniquely correct answer to how much risk is the correct

6 This list is not exhaustive. Pfleeger and Cunningham (2010) list nine challenges to measurement in security. Additional challenges I will not discuss include the practical difficulty of detecting rare events while suppressing alarm errors (Axelsson, 2000), economic incentives that work against secure systems (Anderson, 2001), and navigating a changing international legal landscape.

7 This famous "halting problem" is closely related to Church's Thesis on what mathematical functions are calculable and to Gödel's incompleteness theorem (Boolos et al., 2002). There is also a subfield of computing, complexity theory, dedicated to determining equivalence classes of how difficult a computable result is to derive in practice.

amount. The goal of cybersecurity is merely to find a satisfactory level of risk (of failure), both in a defensive posture and in the design of observations or experiments to detect adversaries.

Instead of building scientific theories, cybersecurity practitioners model modes of attack and of defence in ways that can usefully be thought of as modelling mechanisms of attack and of defence.[8] Chapter 3 cleared away the idea that general knowledge is built in the form of laws of nature. Further, building general knowledge in domains of this kind does not come by derivation from theory, as will gradually become clear in Sections 4.2 and 4.3. Cybersecurity practice is the basis for my account of building knowledge, in accord with the Philosophy of Science in Practice approach (SPSP, 2017), rather than attempting to focus exclusively on theories, as has been more common in the history of philosophy of science. Indeed, rather than assuming general knowledge is there to find, a focal point is how difficult it is to build.

This chapter's argument is organized in three parts. Section 4.2 briefly examines how the challenge of gaining general knowledge has been treated in philosophy of science. Section 4.3 explains the practice of building what I treat as knowledge of mechanisms in cybersecurity. Section 4.3.1 begins by developing the three major challenges of cybersecurity mentioned above. Then Section 4.3.2 will initiate detailed casework with three examples. The first is the 'intrusion kill chain' (Hutchins et al., 2011), one de facto standard model (per Chapter 2) of the categories of steps an adversary must take in order to penetrate and control a system. Second, at one level below the kill chain, practitioners analyse the malicious software (or malware) that accomplishes a particular step in the chain. This task is called 'malware reverse engineering'. Third, at one level up from the intrusion kill chain mechanism, the example is the model of the UK National Crime Agency (NCA) about the mechanism of money theft and laundering by the internet criminal ecosystem (Addis and Garrick, 2014). These three examples of discovery show how cybersecurity work in these areas can broadly be thought of as the discovery and modelling of three interrelated mechanisms.

8 Note that this chapter will not address metaphysical issues at all; instead, see Illari and Williamson (2013). Epistemology is the primary concern. The following chapters will, however, write about both mechanisms and models of mechanisms. There are some debates that might suggest this is controversial, particularly the ontic-epistemic debate; for example, see Illari (2013). This chapter aims to be in accord with the views even of the most ontic of major mechanists, in particular Craver (2006) and Glennan (2017), who, in discussing modelling extensively, both recognise the epistemic aspects of mechanism discovery far more than is usually recognised in discussions of their work.

The third part of the argument is located in Section 4.4, which uses this casework to show how fragmented work can still be seen as building, in a patchwork way, considerably more general knowledge despite the three cybersecurity challenges. Section 4.5 summarizes the positive impacts of understanding general knowledge as built up by discovering and modelling clusters of multifield mechanism schemas related along four dimensions.

4.2 GENERALITY IN PHILOSOPHY OF SCIENCE

Approaches to general knowledge in the history of philosophy of science were for a long time focused on scientific theory and laws. Section 4.2.1 examines that history, building on the argument from Chapter 3 that cybersecurity should follow lessons from the more recent philosophy of the life and social sciences. To develop this recommended path, Section 4.2.2 extracts from the philosophical mechanisms literature a coherent thread of how mechanism discovery builds general knowledge in the life sciences.

Note that this chapter will not directly address philosophical views of knowledge and generality, instead studying how the focus of such work has moved away from laws and theories as primary scientific knowledge, and this carries with it changes in how scientific knowledge is and should be seen. Given Radder's survey of the ways in which a justified true belief account of knowledge is not fitting for scientific knowledge, I take a broadly pragmatist approach to knowledge and understanding, current proponents of which include (Leonelli, 2009; Radder, 2017).

4.2.1 *Turning away from laws*

There has been a long-standing philosophical interest in understanding what unified or general knowledge is, and how to build it. Unity of knowledge was traditionally understood in terms of unifying theory, driven by important theoretical unifications in science. An important example attempt at describing general knowledge in this way was logical laws in the logical empiricist tradition introduced in Chapter 3. A representative description of unification is given by Bogen and Woodward (1988, p. 325), when it was already being criticised:

> 'A characteristic kind of advance in scientific understanding occurs when one sees how what previously seemed to be a number of independent, unrelated facts can be accounted for in terms of a small set of common mechanisms or laws. Nineteenth-century optical theories represented an important explanatory achievement because they provided a unified, systematic account of a wide range of optical phenomena involving reflection, refraction, diffraction, stellar aberration, and polarization in terms of a few basic assumptions regarding the transverse wave character of light. Similarly, Maxwell's theory provided a unified treatment of an apparently diverse set of electromagnetic phenomena.'

Bogen and Woodward (1988) already give one influential argument for the view that focusing exclusively on theory and laws was not adequate. With the growth of philosophy of life sciences, this questioning accelerated. Philosophers noticed that general knowledge in the life sciences cannot be understood in this way. There are relatively few unifying theories in the life sciences, and very few laws in the traditional sense. One possible exception is evolutionary theory and its various mathematical expressions, but even this, on its own, is not going to be adequate to capture everything that is known. Many philosophers of life sciences rejected a philosophy of science based on traditional laws, following Cartwright (1983), recognising the plurality and diversity of the life sciences that makes laws rare (Dupré, 2012; Mitchell, 2003).[9]

Cybersecurity seems to face a similar problem to the life sciences in understanding what counts as general knowledge and how to build it. This challenge has likewise manifested as a kind of wrestling with the problem of laws. Recall from Chapter 3 the following question from the DoD.

> Are there "laws of nature" in cyberspace that can form the basis of scientific inquiry in the field of cyber security? Are there mathematical abstractions or theoretical constructs that should be considered?

Which was answered with:

> There are no intrinsic "laws of nature" for cyber-security as there are, for example, in physics, chemistry or biology. Cyber-security

9 Some philosophers sought to re-characterise laws to be friendlier to the life sciences, such as Mitchell's pragmatic laws (Mitchell, 1997), or Woodward's invariant generalisations (Woodward, 2003). As this chapter's focus is mechanisms, I do not discuss these.

> is essentially an applied science that is informed by the mathematical constructs of computer science such as theory of automata, complexity, and mathematical logic (MITRE, 2010, p. 4).

So it seems there are parallel problems of understanding general knowledge in the life sciences and in cybersecurity. This similarity means work on the life sciences may well help address cybersecurity challenges. In philosophy of life sciences, attention has turned away from laws, to the search for mechanisms. However, the initial focus of the mechanisms literature was to give an account of scientific explanation without using laws, which means there has not been a lot of work directly on the question of building general knowledge by discovering mechanisms. There has been work on singular versus general mechanisms (Glennan, 1997, 2011), on how people should think about really fragile mechanisms (Glennan, 2010) and on regularity (Andersen, 2017), but comparatively little on how general knowledge is built, analogous to the idea of theory-building. This focus means that, in philosophy of science, and in spite of the turn away from laws in philosophy of life sciences, current efforts to understand unification or generality are still largely dependent on this history of focusing on laws or arguments, developing the traditional Hempelian framework (Friedman, 1974; Hempel, 1965; Kitcher, 1981). These developments also inform normative constraints for what CSIR analysts should seek when they seek to know about an incident. Therefore, an alternative account of general knowledge is needed.

4.2.2 *Generality and mechanisms*

This chapter focuses on how general knowledge can be built in cybersecurity using the philosophical mechanisms literature. However, the philosophical literature does not have an account of building general mechanistic knowledge ready to hand. This section will build such an account. The main goal is to help build knowledge in cybersecurity; however, I expect future work should show this account is broadly applicable across sciences.

This section will weave together Darden's work on clusters of mechanism schemas, Craver's discussion of the way multifield mechanisms can form what he calls a 'mosaic unity', and Glennan's very recent work on the dimensions of variation of mechanisms. This section will show that threads can be pulled out of this work and can be woven into a picture of general mechanistic knowledge

being painstakingly built as some mechanisms become well known, alongside related ones, while various interesting relationships among those mechanisms gradually become better established. The picture that emerges is rather than generality being something given with laws or theory, it is something painstakingly built up in the mechanism discovery process. In line with this, generality is something much less than universal, which was the original assumption which went hand-in-hand with the study of supposedly universal laws. Generality turns out to be hard to find, and highly valued when found.

Lindley Darden (2006) suggests that if there is such a thing as biological theory, it consists of clusters of mechanism schemas. Mechanism schemas are, in crude terms, abstractly specified mechanisms, lacking concrete detail. Mechanism schemas apply to far more things than concretely specified mechanisms, which always include detail that is particular to specific situations.[10]

For example, protein synthesis can be described at an extremely abstract level as the process by which genetic material is used by living things to create the proteins essential for life. This is often understood as involving two paradigm mechanism schemas: one for cells with a nucleus and one for cells without, each transcribing DNA to mRNA, which then moves to ribosomes to translate the mRNA code into amino acid chains. These are slightly different schemas, but each schema applies to many different kinds of cells, so each captures something quite general about protein synthesis. Together, knowledge of both schemas captures something even more general about protein synthesis—which includes the divergence between eukaryotes and prokaryotes. Going further, one more fact about protein synthesis is that lots of organisms use non-standard methods. One important example is protein synthesis as performed by HIV. HIV holds its genetic material as RNA, which it reverse transcribes into DNA, inserting that DNA into the genome of its host cell, to borrow the protein synthesis apparatus of the cell. But what was first discovered for a particular virus is now understood as a standard retroviral protein synthesis mechanism schema. So, one can put this schema alongside eukaryotic and prokaryotic schemas to understand something even more general about protein synthesis.

In this way, Darden's suggestion is that general knowledge in biology is built by clustering related mechanism schemas in this way, where the example has two paradigm cases, closely related and covering many different kinds of

10 See Glennan's work on ephemeral mechanisms for an account of one-off mechanisms, i.e. historical mechanisms that may occur only once (Glennan, 2010).

cells. Additionally, there are many non-paradigm cases, like retroviral protein synthesis, and probably many more quirky and particular cases to discover. The cluster cannot be reduced to a laws-description, nor can the local quirks be collapsed into an overall schema. Biochemistry students build knowledge by learning about the paradigm cases – and about the quirky cases. In spite of such variation, the collection of these schemas yields what general knowledge is actually available concerning protein synthesis. It is at least clear that understanding the cluster gives scientists something far more general than they gain from understanding the protein synthesis of one particular cell, or even one paradigm mechanism schema. So Darden's suggestion seems to offer an insightful beginning to an account of generality in mechanistic knowledge.

The second element of the account of general knowledge is the work of Carl Craver (2007) on what he calls "mosaic unity", which develops both his joint work with Darden, and her early work such as Darden and Maull (1977).[11] The traditional view of theory unification, summarised above, tended to assume that unified theories would be restricted to a single domain, and until relatively recently the dominant domain examined was physics. I explained Darden's suggestion above with reference to the life sciences as she studied them extensively, but restricted the discussion to the single domain of biochemistry. However, Craver notes that scientists increasingly collaborate across domains to explain phenomena, discovering what he calls "multifield mechanisms" (Craver, 2007). He is interested in how work based in different scientific fields is *integrated*. Craver studies the integration of various sciences of the mind and brain, which offers excellent exemplars of what he has in mind. I am interested in his insights into how unity can be made that crosses scientific fields, rather than his more particular studies of neuroscience. Nevertheless, his choice of case is best explained in his own summary of an influential 1973 paper:

> '[Bliss, Gardner-Medwin, and Lømo's] introduction is an extended argument for the relevance of LTP [Long-Term Potentiation] to learning and memory. Their argument, not coincidentally, appeals to results from multiple fields. They appeal to experimental psychologists' ablation studies ..., biochemists' assays of the mo-

11 This part of Craver's influential book is not much discussed, although the fact that the idea of mosaic unity is the topic of the final chapter, and appears in the book's title, suggests that Craver considered it very important, at least in 2007. Note that while Craver creates his account by studying neuroscience, the relevant item is his theoretical results, the account of mosaic unity.

> lecular constituents of the hippocampus ..., physiologists' EEG recordings during memory tasks ..., psychiatrists' evaluations of patients with brain damage ..., electrophysiologists' theoretical considerations ..., and computer scientists' models ... . Results from these different fields constrain the possibilities for situating LTP within a multilevel mechanism.' (Craver, 2007, p. 243, citations deleted).

I will assume, in accordance with minimal mechanism, that mechanisms are typically hierarchical in this kind of sense. The relevant interrelationships among the three cybersecurity cases will become a clear example of it in Section 4.3 and Section 4.4, showing a similar significant heterogeneity in relevant kinds of evidence. Craver argues that scientists' understanding of memory has advanced, not by reducing psychological phenomena to neuroscientific law, but by understanding the relationships between many entities, activities and their organisation, drawing on all the disciplines listed above.

Craver suggests that one can understand this integration as multiple fields all exploring a space of all the mechanisms that could possibly explain the phenomenon of memory. Scientists never explore that whole vast space. Instead, knowledge from different fields suggests "plausible" mechanisms. The discovery of new entities and activities, and forms of organisation, can make some parts of that space implausible, or open up new areas as plausible: 'A constraint is a finding that either shapes the boundaries of the space of plausible mechanisms or changes the probability distribution over that space...' (Craver, 2007, p. 247). Craver discusses many such constraints, but one of the easiest illustrations comes from spatial and temporal constraints (which Craver takes to operate both intra-level and inter-level). Time is important for memory, as it was crucial that what was possible at the cellular level could last long enough to be a plausible part of the mechanism for something that, at the psychological level, could last a long time – human memory. In this way, knowledge gained from multiple fields is slowly, carefully, integrated to provide an understanding that crosses fields. Generality is a slow, and often painful, achievement.

This allies very naturally with Darden's work. Memory is not a unified phenomenon (Bechtel, 2007), and one should think instead of a cluster of multifield mechanisms of memory – using Darden's terms alongside Craver's. Cybersecurity similarly crosses levels, from the sociotechnical system mapped by the NCA, to the very technical areas of malware analysis, and can draw

on knowledge from multiple disciplines, techniques and technologies, so that beginning from the combination of Craver's and Darden's work should be useful to apply to cybersecurity.

Very recent work by Stuart Glennan suggests further insights that can be interwoven with those of Darden and Craver. His current view accords well with the view here so far. He writes: 'But scientific fields are not islands. For one thing, they are integrated by what Darden and Maull (1977) once called interfield theories. This integration does not come via a grand theoretical reduction, but rather by exploring localized relations of mechanism dependence, where entities or activities assumed in one field are located, filled in and explained in other fields ...' (Glennan, 2017, p. 143).

I will focus on drawing out some insights of Glennan's very recent work about kinds of mechanisms and ways of comparing them – his 'taxonomy' of mechanisms. He uses the account of "minimal mechanism" introduced in Section 4.1 to build it. The heart of his view is simple: each of the parts of the characterisation of mechanism direct scientists, somewhat independently, to how mechanisms are more or less similar to each other. Three parts are obvious: phenomenon, entities and activities, and organization.[12] Glennan adds a fourth, etiology, which is history, or how the mechanism comes about.[13] He writes first of entities and activities:

> 'The picture I have given of mechanism kinds gives us some understanding about the nature of disciplinarity and of the extent and limits of the unity of science. Scientific fields are largely defined by what I have called material similarities—similarities in what (material) kinds of phenomena they seek to explain, as well as the set of entities, activities and interactions that they take to be (potentially) responsible for these phenomena. ... Disciplines grow around the material and theoretical resources, technologies, and experimental techniques used to explore these phenomena and the mechanisms responsible for them.' (Glennan, 2017, p. 143)

Entities and activities are often the most striking similarities and differences among mechanisms. Mechanisms that share common entities, like neurons, are obviously similar in sharing neurons, and fields form around the sharing of common technologies used to study those entities and their activities. This

12 The ideas of these as important dimension is already in the discussion of multifield mechanisms in Craver, 2007, but Glennan develops this considerably.

13 This is related to a skeletal account in Illari and Williamson (2012), and is summarised in Glennan and Illari (2017).

seems fairly obviously true, and indeed one way of thinking about the insight Glennan offers is that phenomena, organization and etiology are *also* very important to understanding and comparing mechanisms. Glennan offers an account of these and argues that these dimensions of comparison are to an important extent independent: for example, mechanisms for the same phenomenon might include quite different activities and entities, and mechanisms with similar entities and activities might have quite different forms of organisation.

Let us examine how Glennan's point can illuminate the idea of clusters of multifield mechanisms built from Darden and Craver's work. Multifield mechanisms are painstakingly built up by scientists collaborating on understanding a particular phenomenon, and, given that mechanisms are not unified, there are likely to be multiple related mechanism schemas for a particular phenomenon. Glennan's work, then, illuminates *four* different places to look for clustering among related mechanisms.

Of these four, relations between activities and entities in related mechanisms will be obvious. The shared technologies created to study them are likely striking. The nature of the phenomenon, as this is often the initial focus of work, is also likely striking. Forms of organization and etiology will be much less obvious. But if Glennan is right, one should expect them to be present and particularly important to cross-field mechanisms.

This section has woven together threads of insight from Darden, Craver and Glennan into an initial picture of the building of general knowledge by discovering clusters of related multifield mechanism schemas, that vary along four dimensions: activities and entities, phenomena, organization, and etiology. In this array of ideas, the mechanisms literature offers ways of thinking about what counts as such general knowledge as it is possible to get, and where to look for it within sciences which discover mechanisms. The following chapters will apply these theoretical insights to cybersecurity. Section 4.3 begins this application with a more detailed exploration of the challenges to building general knowledge in cybersecurity, and how practitioners respond.

4.3 BUILDING MECHANISTIC KNOWLEDGE IN CYBERSECURITY

Building mechanistic knowledge in cybersecurity faces many challenges. Some are similar to those addressed in the existing philosophical literature. I focus on three challenges that, while individually not unique to cybersecurity,

jointly produce distinctive difficulties in building general knowledge, making this a fascinating domain for philosophers interested in general knowledge. Section 4.3.2 explores three interrelated examples of active research problems in cybersecurity that each demonstrate the triad of challenges that Section 4.3.1 establishes. Each example overcomes the triad of challenges differently, and yet each can be illuminated by the picture of building mechanistic knowledge provided by the philosophical literature.

4.3.1 *The three challenges for cybersecurity*

Any problem domain has its own quirks that give practitioners difficulty. In experimental physics, building precise measurement devices to detect rare events is a challenge. In virology, pathogens evolve and change year-to-year, thwarting vaccines. In macroeconomics, one has to rely on natural, rather than controlled, experiments and cope with the fact that many of one's subjects are people, who may read and respond to research results and policy announcements. This section will introduce three aspects of cybersecurity that have heavily shaped research design and methodology in the practice. While none of these three aspects is unique to cybersecurity, each exacerbates the other two. This subsection will provide a rough introduction to the cybersecurity problem space and its problem-solving methods. This triad of challenges has forced cybersecurity practitioners to refine their methods beyond what is expected in disciplines where each aspect or challenge arises alone. This triad does not cover all the challenges in cybersecurity, but its combination provides an instructive set of examples.

The three selected challenges in cybersecurity are: the immediate object of study, namely software, can change behaviour during or between observations; active adversaries respond to, and try to confound, observations; and there is often justified secrecy among friendly parties. The combination of these aspects of cybersecurity pose notable methodological challenges. The next section illuminates how they pose a challenge to the building of general knowledge, by showing how some success has been achieved, using the three examples of mechanism schemas drawn from active research problems in cybersecurity.

CHANGEABLE SOFTWARE: That software is changeable is a property of computer science generally, not just security.[14] Code is easily and often changed by human software developers, and running programs may change their own behaviour during execution (Thompson, 1984).

This challenge includes at least two closely related issues. First, the fact that humans frequently adapt code – and can design, reuse and redesign code to behave differently based on input parameters – will be relevant when discussing of malware. Second, the fact that software environments are complex, such that code may behave differently in the presence of other code, combinations of input values, or on certain hardware and not others. This second issue is more relevant to studies of reliability, vulnerability detection, and debugging and might be considered the dynamic nature of software, rather than its changeability per se. However, malware authors tend to leverage the ambiguity afforded by the dynamic nature of the software environment to their advantage. The salient aspect in which software is changeable is that argued by Hatleback and Spring (2014); not that the source code is editable, but that the behaviour is both dynamic and responsive to the environment in arbitrary ways. Both the arbitrariness and having-been-designed make studying the dynamism of software a distinct challenge from dynamism in other fields such as chemistry and biology. For this reason, I use 'changeability' to capture both aspects simultaneously.

In computer science practice, one may want to verify that code as written has certain properties or meets particular requirements. That software can change dynamically during a test or experiment is one major challenge in this endeavour. Different run-time results can be purposeful (for example, if today is payday, then pay employees), accidental (if I trip on the network cable while the program is talking to the bank, and disconnect it, it fails because the environment changed), or stochastic (for example, the program generates a random number to start from). One impact of these various changeabilities is that a lot of effort in software development is put towards testing whether a patch or update has re-broken an old fix of an old flaw. Such 'regression testing' only approximates the ideal of testing each change against each past fix over each possible task the program might perform (Brooks Jr, 1995). In practice

14 For a review of formal semantics attempting to cope with changeability, see Winskel (1993, p. 297ff). Difficulties related to the changeability of software figure prominently in the historical development of the internet (Hafner and Lyon, 1998). To account for such changeability during mechanism discovery in security, Hatleback and Spring (2014) argue for heuristically dividing mechanisms into those that are engineered and those that are physical or natural.

the software changes too much to test all those possibilities exhaustively, both in that programmers make edits more quickly than are practical for tests and in that potential software run-time deviations based on task and environment are more numerous than are practical to test.

DECEPTIVE ADVERSARIES: The first challenge becomes particularly pernicious when combined with the second challenge: active adversaries deliberately exploit the changeability of software, re-writing it to make it harder for defenders to detect and repel. To be an adversary, something or someone should be what is known in game theory as a bona fide player; that is it must "(1) make choices and (2) receive payoffs" (Rapoport, 1966, p. 20). A *deceptive* adversary takes actions in response to the cybersecurity researcher to try to change the researcher's conclusions.

Some methodological problems involving the target system altering during study are already known in philosophy of science. For example, experimental manipulations made on a target system alter the causal structure of the system itself. Such experimental manipulations are known as 'structure-altering interventions' in the literature on Woodward's interventionist theory. The problem is discussed beyond this literature; Mitchell (2009, p. 67ff) applies this to gene knockout experiments. These experiments aim to find out what the knocked-out gene normally does, but face the methodological challenge that genes are strongly interactive, and backup mechanisms exist for many essential cellular processes. So if you knock out one gene, another set of genes often activates to fulfil the task. This means practitioners need to find other ways to establish the role of the knocked out gene.[15] However, the following will demonstrate that the combination of the problems of changeable software and deceptive adversaries goes far beyond that of structure-altering interventions.

Other domains also study targets that in some sense actively resist. For example, immunology faces adversaries in the pathogens they study. Pathogens may change in response to anti-biotics, for example. It would be hard to argue the pathogens make choices, as adversaries do in cybersecurity. Even if so, pathogens do not read the immunologist's research papers and figure out ways in which to subvert them, as adversaries do in cybersecurity. This is a qualitative difference with noticeable impact on building general knowledge in cybersecurity.

15 See extensive discussion by Steel (2008) and Cartwright, primarily with respect to social sciences. As Cartwright (2012) points out, within economics this problem is known as the 'Lucas Critique,' following Lucas Jr. (1976).

WELL MOTIVATED SECRECY: The third challenge of the triad pushes the overall problem further beyond challenges discussed with respect to other domains. Cybersecurity practitioners must hide knowledge and successful strategies from adversaries, and so cannot freely share knowledge, strategies, and successes. Not only does this need for secrecy lead to repeated work, but each cybersecurity practitioner is not in sole control of what knowledge or successes are leaked to the adversaries, who then use that knowledge to instigate changes to their deception.

These three challenges are averred by practitioners, including Kaspersky, an anti-virus and security-consulting firm. For example, on the challenges of secrecy among friends and active adversaries, consider:

> ...we remain bound by corporate realities, respect for the research methods of collaborators, and, most of all, legal constraints. As such, we may not always be able to provide full disclosure of indicators involved in certain findings. ...we feel these are not vital to convey the main thrust of our argument, which is that intermediate-to-advanced threat actors are aware of attribution methods and are already attempting to manipulate researchers to expend limited resources chasing ghost leads. Where gaps arise, let us relegate these accounts to camp fire re-tellings among friends. (Bartholomew and Guerrero-Saade, 2016, p. 3.)

They also discuss the need for, yet difficulty in, constructing general knowledge:

> An often ignored facet of the [cybersecurity knowledge] production cycle is the role of the analyst whose purpose is to coalesce various sources of information, arrive at various conclusions, and vet the overall logic of the finished product. Sadly, at this stage in the rise of the threat intelligence industry, deficient hiring practices overemphasize specialized technical knowledge and eschew generalist broad-thinking capabilities, often assuming technical candidates will bring these in tow. This is seldom the case... (Bartholomew and Guerrero-Saade, 2016, p. 9)

This challenge of secrecy goes along with changeability and deception to create a particularly serious barrier to the building of general knowledge. Ultimately, if general knowledge is to help improve cybersecurity practice, then

it needs to be in a form that can be *shared*, as general knowledge is shared in many scientific fields. The need for some kind of shareability that meets these challenges becomes an integral part of the problem of building general knowledge in cybersecurity.

One obvious way of sharing knowledge is to publish it in the standard academic, peer-reviewed venues. However, there is a spectrum of sharing between telling no one and publication, and multiple options are important to cybersecurity. The spectrum ranges from fully-formed government classification networks with strict military-legal guidelines, to contractual non-disclosure agreements among corporations, to informal networks among peer individuals. Indeed, current attempts to reduce barriers imposed by secrecy predominantly involve painstaking networking among professionals in the field to build personal relationships that support sharing. There is not much research into this phenomenon, but Sundaramurthy et al. (2014) anthropologically documents a case study of the difficulty of gaining trust among computer-security incident-response staff.

Information sharing may self-organize or be mandated. Two examples of self-organized or self-selected constituencies are the Anti-Phishing Working Group and Shadowserver. An example of mandated sharing is the US Presidential Decision Directive 63 which, in 1998, formed information sharing and analysis centres for each of the national critical infrastructure sectors. Game-theoretic analysis of information sharing suggests firms best voluntarily share information in the implausible scenario of highly competitive markets with firms both large and equally matched – and even then the results fall short of what would "maximize social welfare" (Gal-Or and Ghose, 2005, p. 200). Modern operational data agrees that sharing is disjointed and visibility partial.[16] Further, cybersecurity contains what economists call a market for lemons, where a consumer cannot distinguish quality products from bad ones (lemons), though the vendor has the information to make the distinction (Anderson, 2001).[17]

16 Dozens of lists of malicious computer locations are broadly disjoint, even across wide spans of time (Kührer et al., 2014; Metcalf and Spring, 2015). Limited sharing also appears to perform worse than public sharing on website compromise recidivism (Moore and Clayton, 2011).

17 For further reading see the long-running Workshop on the Economics of Information Security (WEIS) or the Workshop on Information Sharing and Collaborative Security (WISCS).

I will consider multiple possibilities for sharing beyond academic publication, but only the forms of sharing that are relatively wide. That is, what can be shared beyond painstakingly built trusted private networks.

4.3.2 *Three examples of cybersecurity mechanisms*

This subsection will illustrate how practitioners approach these common challenges through three examples of active research problems in cybersecurity. These three examples serve to justify and deepen the various assertions I have made above about the nature of cybersecurity practice, particularly indicating the range of applications with which cybersecurity contends. The three cases are (1) research to track and reduce the harm caused by the myriad attacks that steal money; (2) the intrusion kill chain model of an individual attack, which models the entities, activities, and their organization by which an adversary initiates, executes, and makes use of an attack; and (3) tools for reverse engineering a single piece of malicious software (malware), which is a particularly important entity in many individual attacks. These three examples form what is in some sense a hierarchy. In reverse order, malware is almost always used in an attack, but it is only one part of the mechanism modelled by the intrusion kill chain. Likewise, various attacks are used in the mechanism of electronic crime (e-crime), but they are in turn only a part of e-crime considered more broadly, from the perspective of national agencies. In this way, the three examples are hierarchically related. Taken together, they demonstrate the scope of cybersecurity, from social and economic systems through the technical minutiae of how malicious software takes control of a computer.

It will also become clear that cybersecurity practice does still manage to achieve considerable success in spite of the three challenges, and this subsection will evidence how thinking of the building of general knowledge in the field as the building of mechanism schemas – *shareable* ones – is a reasonable and useful way of conceptualising the achievement. This prepares Section 4.4 to indicate how fruitful this conceptualisation could be for practitioners.

BOTNETS—THE NCA'S BANKING TROJAN MODEL: The first example comes from the UK's National Crime Agency (NCA)[18] and their de-

18 Thanks to Stewart Garrick for this example and permission to use his diagram.

scription of how networks of compromised computers (botnets) are created, monetized, and how the money obtained is laundered. The NCA may seem an unlikely source for academic lessons on mechanism discovery. However, cybersecurity concerns are endemic to all sectors of society, and much research activity happens outside academia. Figure 4.1 displays the "banking trojan business model" for internet criminals who steal money from consumer banking accounts, as described by Addis and Garrick (2014).

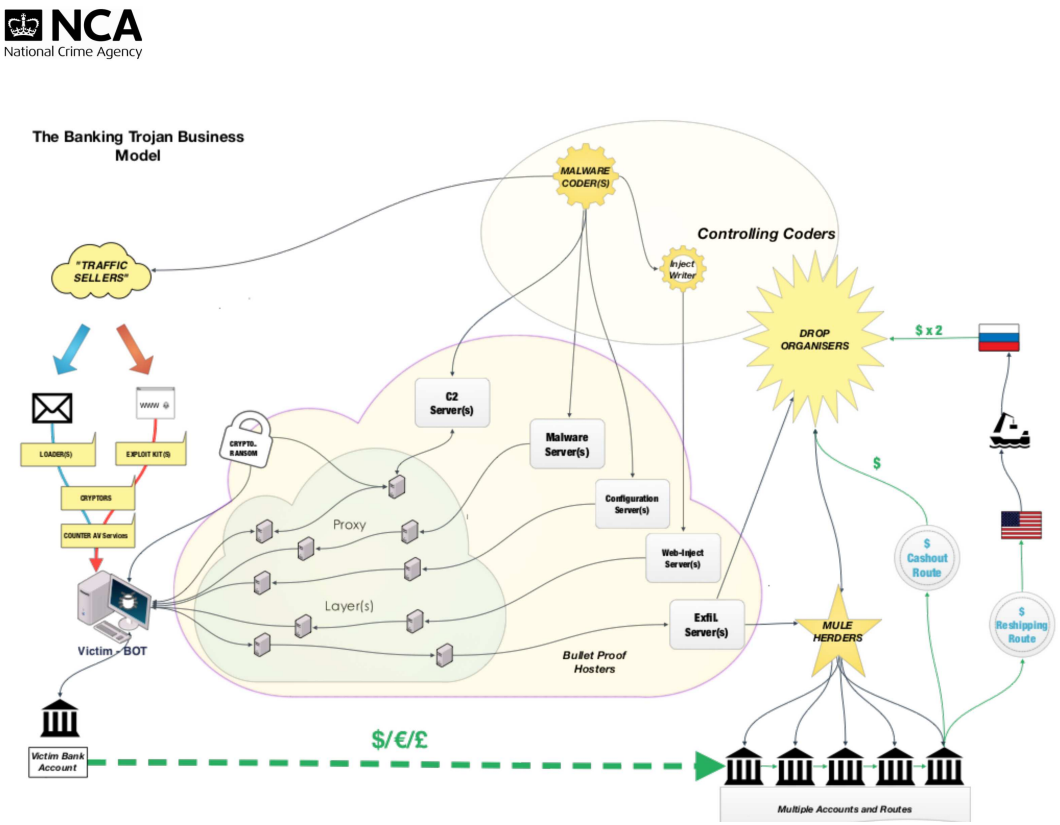

Figure 4.1: Botnet theft and money laundering mechanism as described by the NCA in Addis and Garrick (2014). Reprinted with permission.

This criminal business mechanism is complex. It starts in the top-center of the diagram with the controlling coders, the software developers who create the software necessary to manage a diverse infrastructure of loosely coordinated compromised machines. This infrastructure is unreliable to the criminals, because the machines' owners may turn them off, move them, change their network addresses, or notice and fix the malware infection. The NCA is not concerned with individual computers in the network, but with the network itself: that it is unreliable to the criminals changes how they behave and what they build, and so what the NCA should look for. Like any other savvy

project manager, the criminals outsource various aspects of their business. Grier et al. (2012) and Sood and Enbody (2013) survey these exploitation-as-a-service and crimeware-as-a-service businesses. The various entities to which services are outsourced are listed in the diagram, for example "traffic sellers," "malware servers," and "proxy layers." The NCA's mechanism discovery has decomposed the criminal's task into these entities. A security expert would also understand the activity localized to each entity. For example, traffic sellers use various tricks such as compromising popular web sites and sending unsolicited bulk email (spam) to direct potential victims to the controlling coders' malware. These two activities produce malicious emails and websites, entities represented underneath the "traffic sellers" entity. Similar things happen with the other elements of the mechanism, leading counter-clockwise eventually to the criminals receiving money.

In this way, the NCA mechanistic model conveys a great deal of information. But on closer inspection, the NCA are also safeguarding against some of the challenges of cybersecurity; specifically, the challenges of the changeability of software and the presence of active adversaries who will respond to published observations. The goal of understanding the mechanism of the criminals' business is to interrupt that business, and this goal could be impeded by publishing too much. Given this, the mechanism description focuses on essential functionalities. Although software is changeable, the internet's basic rules of operation do not change quickly. The Internet Engineering Task Force and Internet Architecture Board oversee change proposals, and key services may be updated only once a decade. The criminals must accomplish certain tasks within this framework, because all their potential victims are on the internet. Therefore it is relatively safe to publish to the criminals that the NCA knows traffic is delivered via email, websites, and proxy layers. There may be myriad ways to create software that performs these activities, but each activity itself cannot be easily abandoned if the criminals still want to accomplish their goal.

When legal authorities plan to intervene on a criminal mechanism of this kind, they must also respect the challenges of cybersecurity. In examining the planning of successful interventions, one starts to feel the pressure of justified secrecy among friendly parties. As one example, imagine that Internet Service Providers (ISPs) detect indicators of compromise among their customers and notify the banks to freeze the account if one of their mutual customer's computers is compromised, thus limiting theft. However, privacy laws generally prevent ISPs from providing information about their customers' traffic to any-

one, including banks. The ISP may not even be allowed to know the customer's bank. And where the victim's traffic is encrypted the ISP may not be able to detect when a customer is victimized at all. Encryption is mathematical secrecy between two parties. Encryption is a highly recommended protection against, among other things, criminals stealing your banking credentials during online banking. But encryption works just as well for the criminal to hide their attacks. If encryption is actually to provide privacy, intermediate parties like ISPs must not be able to distinguish between any two encrypted items, even if one is encrypted banking and the other encrypted attacks. So privacy, both legal and technical (provided by encryption), limits the possible cybersecurity interventions.

For a crime prevention agency, the end goal is usually to arrest the criminals. This seemingly straightforward goal is further hampered by a combination of the internet's global reach and international politics, which creates justified secrecy in another form. Everyone accesses (essentially) the same internet, whether it is from London, Moscow, or Tierra del Fuego. Arrest warrants do not have such an immediate global reach. Although mutual legal assistance treaties often succeed eventually, and there have been successful arrests, national legal processes do not allow broad sharing of suspects of investigations with just anyone. Further, the internet is dominated by pseudonyms, and arrest warrants for "xxCrimeBossxx" or "192.168.6.1" are not terribly effective. Although private companies may know the identities of their customers behind these pseudonyms, for legal or contractual reasons private companies may not be able to share these with law enforcement, especially foreign law enforcement. This all means that mechanisms of intervention tend to focus effort on protection and prevention activities.

The NCA navigates the triad of challenges of changeable behaviour of software, reactive adversaries, and justified secrecy. First, they diagram the relatively unchangeable constraints criminals *have* to operate within, and second, they publicise only constraints already known to criminals, and not alterable by them. Of course, this does not eliminate attempted deceit by criminals, and many issues of secrecy even among those attempting to preserve security still remain.

A NOTE ON THE RELATIONSHIP BETWEEN EXAMPLES I will shortly turn to the second example, computer network attacks, but first note the relations among the examples. The three cases form a mechanistic hier-

archy in the sense described by Craver (2007). At the heart of the internet banking crimes described above are the computer network attacks described next. These are attacks which convert a healthy computer controlled by its owner to an infected victim controlled by an adversary. Attacks occupy the left-hand side of Figure 4.1 above, from the traffic sellers through taking control of the victim computer, known as the 'bot', and ending at the objective of access to the victim's bank account. However, Figure 4.1 does not detail how the attacks happen, what methods the criminals use, or who is targeted. In part, this is because the attacks used are diverse, and changeable, and so are hard to model. More importantly, for the level of the explanation of the criminal business model the details of how the attacks occur are not important. However, from a different perspective, of computer owners who would like to protect themselves, the details of how each attack happens are crucial to detecting and preventing attacks.

Descending a level further hits the third example, malware analysis. Note that malware is not placed at a lower level merely because it explains physically smaller items, or merely because a part is spatially contained within a whole (two popular views). Instead I follow Craver (2007, ch. 4-5) in holding that levels of explanation are relative to levels of mechanisms for the phenomenon of interest where the elements are indeed parts of wholes, but they are also mutually manipulable in the sense that changes at the level of the part will at least sometimes make detectable changes at the level of the whole, and changes at the level of the whole will at least sometimes make changes detectable at the level of the part. So these examples form a hierarchy because one of the components of the mechanism describing the criminal business model shared by the NCA is computer network attacks. And one of the elements of computer network attacks in turn is malware. This is not strictly a part-whole relationship; attacks can happen outside of crime, for example during nation-state espionage. And to explain even one malware sample used in an attack, one must explain not only its attack role but also its historical relation to other malware as well as how it hides itself. Yet in this way, a mechanistic explanation of attack adds to the higher-level explanation of the criminal business model, and vice versa, and so on with malware related to these two examples.

COMPUTER NETWORK ATTACKS – LOCKHEED MARTIN'S INTRUSION KILL CHAIN: With that loose approach to mechanistic hier-

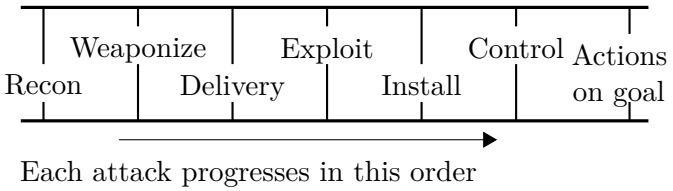

Figure 4.2: A simple kill-chain diagram showing the steps defined by Hutchins et al. (2011), with slightly simplified names for the steps.

archy in place, this example moves down a mechanistic level. Understanding models of attack is the second example of an active research problem in cybersecurity. One de facto standard model of the steps any adversary must take in a successful attack is the intrusion kill chain (Hutchins et al., 2011). Chapter 5 will develop this example further and argue that the kill chain can be considered a mechanistic explanation of an attack.

The kill chain model decomposes an attack into seven steps. For an individual attack, where an attack is defined with a quite small scope of targeting exactly one computer, these steps occur in a linear sequence. The seven steps are: (1) *reconnaissance*, gathering necessary details on a target; (2) *weaponization*, creating a malicious file suitable for the target; (3) *delivery*, sending the weaponized file, usually via email, web traffic, or USB drive; (4) *exploitation*, initial attempt to take over a computer once the file is delivered and opened; (5) *installation*, adding additional software to maintain a robust covert presence; (6) *command and control*, any communication between the installed malicious software and the human adversary for reporting, updates, or direction; and (7) *actions on objectives*, where adversaries finally move to complete their material goals. Adversary goals may include stealing files, corrupting essential data, or starting back at reconnaissance (1) to conduct new attacks which are only viable from a computer which the defender still trusts (Hutchins et al., 2011, p. 4-5). This describes a single attack, but note that an adversary almost always coordinates multiple attacks to achieve an effective campaign (which is accounted for in the de facto standard campaign modelling framework, the diamond model). Figure 4.2 captures the organization of the seven steps of the kill chain.[19]

The kill chain model avoids the challenge of the changeability of software and adversary responsiveness with a strategy similar in some respects to the model of criminal business methods. The kill chain model contains somewhat

19 Chapter 5 will improve this diagram to make the relevant entities explicit, see Figure 5.1.

abstractly specified activities that are necessary steps in a successful attack. There is extraordinary variability in the entities that perform these activities, where they are performed from, and the exact details of how to accomplish the activities, but the activities themselves are fairly stable. For an individual attack, the organization is also fairly simple: activities occur in linear order. That is slightly more complex for the usual case, which involves multiple interrelated attacks, which often run simultaneously. Nevertheless, the kill chain mechanism is a rare statement of a stable organization of stable activities even in the face of considerable variability and changeability in entities. Adversaries still attempt to hide their activities and status along the chain, for example by conducting many simultaneous attacks at asynchronous stages of progress. For example, adversaries are known to confound attack response by quietly hiding an important attack within a noisy but merely annoying attack. Although they can cover their tracks in such ways, the kill chain remains a stable organization of activities per attack.

The triad of challenges in cybersecurity all impact modelling at the level of the kill chain. The kill chain model was published by Lockheed Martin, one of the largest US defence contractors, an organization attacked by all manner of adversaries. The Lockheed researchers who created it based the kill chain model on their own experience investigating and responding to attacks over eight years.

One common criticism of the kill chain model is that it is too abstract. This criticism directly relates to how few commonalities there are among these eight years of attacks. The changeability of software forces the level of analysis up to where the activities and their organization are both stable. But this level of analysis is too high, too abstract, for much day-to-day cybersecurity because reasoning with the kill chain model is not automatable. The kill chain research resists the active response of adversaries by selecting elements which the adversary cannot change and remain effective, but this resistance comes at a cost.

Defenders benefit from using the kill-chain model of the mechanism of attack because it is a focus for orienting a defensive posture and incident response based on what steps of the kill chain the adversary has accomplished. Although the kill chain alone is too abstract to actually be capable of detecting malicious software, it directs how the defender responds after detecting malicious software. Such models speed communication among defenders, who are almost always organized into teams, specifically, a computer security in-

cident response team (CSIRT). Alberts et al. (2004) describe a detailed set of processes for incident management by CSIRTs. These include establishing or coordinating clear communication, and educating constituents. Established models play an important role.[20]

Publishing the kill chain model helps to diminish secrecy among friendly parties by providing a common language of discourse and instructing defenders what to look for. There is a small risk that unskilled adversaries could use the kill chain as a how-to guide. However, this risk of improving the skills of the least effective adversaries is weighed against the potential collateral damage of secrecy. Broad use by allies is also weighed against consulting profits of maintaining a proprietary model. Lockheed Martin is not a university; rather, they are publishing the kill chain as industry experts, to try to help their allies improve their defences.

MALWARE ANALYSIS: Security cannot be achieved on the basis of the kill chain alone, though, because automation is a key aspect of effective cybersecurity. The actual detection or prevention procedures are handled by computers. Even a small computer network handles billions of decisions every minute; no human can be directly involved at such a speed. Thus, the end goal of cybersecurity work is often a pattern or indicator of malicious activity which is highly reliable and simple enough to be quickly checked by a computer. Failing to provide direct progress towards this goal is one criticism of the kill chain. Automation involves moving down a level of mechanism, even though this sacrifices some of the stability achieved by the kill chain model. Malware analysis, the third example of an active research problem, is one method of generating such patterns or indicators.

In general, to find a pattern or indicator that will work as an adequate detection procedure requires a person. There is no hard and fast reason for requiring a person, but one important factor is the fact that the adversaries are people. Computers can sometimes develop adequate detection indicators; this is a common application of machine learning. This example, malware analysis, is driven by a person. Malware analysis relates to the other two examples

20 There is debate about whether functional explanations (which in a sense provide only a breakdown of tasks) are distinct from mechanistic explanations (Craver and Tabery, 2017). An important distinction in that debate is whether the breakdown of tasks is considered complete, or as a stage on the way to something more distinctively mechanistic. This is a tricky case, because the kill chain, as publishable and shareable, stops at describing activities. Nevertheless, as it is generally used as a device for helping to identify the entities used in a particular attack, it is best thought of as a model of a mechanism.

in this chapter in that both have malware as a component entity, and the malware analyst is attempting to discover the lower-level mechanisms of how the malware functions. Research on these lower-level mechanisms contends directly with the challenges injected by the changeability of software and the adversaries' ability to respond to and interfere with research results.

Malware analysis is one example of many possible processes of building a method for understanding and detecting a specific attack or campaign of attacks. Roughly, a representative process is that a human malware analyst receives an unknown computer program that has been deemed suspicious or likely malicious.[21] The malware analyst then behaves much like a scientist. She will put the unknown sample in a specially designed, controlled environment. She then attempts to determine some relevant properties of the malware, and trigger the malware within the safe environment to exhibit characteristic malicious behaviour or divulge information about its author or controller. Yakdan et al. (2016), Lin et al. (2015), and Lawrence Livermore National Laboratory (2016) describe some of the available technical tools for these tasks.

It is key to understand one fact about computers that makes this task difficult. Recall that Section 4.1 explained that in practice and in principle, one cannot know *a priori* what a computer program will do. This is true even if you write it yourself, and the problem is far worse if an adversary wrote the program to be covert or confusing. The in-principle side of this argument is provided by a suite of formal results, including Turing's halting problem, Church's thesis, and Gödel's incompleteness theorem (Boolos et al., 2002). More practically, when a malware analyst receives a program for analysis, it is just a blob of uninterpreted ones and zeros. You may as well be asked to determine, *a priori*, whether a sentence in a foreign language mentions a cat or not. Even if you can determine the usual word for cat, the speaker may use any number of metaphors, synonyms, cultural references, or proper names to refer to a cat without using the word (such as 'Felix' or 'witch's familiar'). Computer languages can be similarly evasive. Colloquialisms or oblique references can conceal the subject of conversation—namely, malicious actions—from the software analyst. Because a computer tracks references with a precision impossible for a human, computer languages can also be arbitrarily long-winded and round-about while performing these oblique concealments.

21 The determination of what programs are suspicious is an independent topic. At a high level, a program is suspicious if it has properties similar to other malicious programs (attached to similar emails, for example) or if incident response staff unexpectedly find it on a misbehaving machine.

Malware analysis comprises multiple specialized sub-tasks for defeating the deception methods that adversaries employ. A common deception is to hide, or obscure, the true purpose of the malicious software. There are many techniques adversaries use for hiding or obscuring, collectively called 'obfuscation'. Obfuscation takes advantage of the changeability of software to enable a broad class of activities, such as those described in the kill chain and the criminal business models, while attempting to avoid notice. The main task of a malware analyst amounts to seeing through these obfuscation techniques to discover the malicious mechanism that is the intended purpose of the malware. O'Meara et al. (2016) describe two case studies of malware development patterns over several years. The overarching pattern is that defenders defeat the common obfuscation technique at the time and publish a preventative measure, and then the criminals change their software to re-hide their larger-level activity of stealing money so that their thefts are successful again.

An illustrative example of an aspect of the cat-and-mouse game is to play with time. The malware analyst has thousands or millions of suspicious files to analyse. The malware authors know this. The authors also know that their actual targets likely will not know they have been targeted, and tend to be on their computers for a while after the initial infection. So one of the early tactics the malware authors implemented was to make their software sleep, or incubate, for two minutes before doing anything. This defeated malware analysts who opened a file and expected malicious behaviour immediately. Some analysts figured this out and realized they could wait. Then the malware authors increased the sleep period to an hour, far more than any analyst has time to wait, even in mass-automation of analysis. However, the malware analyst totally controls the environment, so they can move the computer environment's clock forward 12 hours and trick the malware. The malware authors realized this and started using arithmetic instead, basically telling their malware to count to a trillion by ones before acting. While counting is notionally benign, malware analysts soon realized that there are not any benign programs that start and just count for a while, so this in itself becomes suspicious. And so on, strategy and counter-strategy.

Under these difficult circumstances, provenance, or the historical sources and similarities among malware files, is often the most useful guide. Groups of attackers tend to reuse their past work rather than start from scratch. The similarity of a malware sample to past samples tends to be important for understanding it. The history is the most stable part of the target mechanism.

In these examples, cybersecurity practitioners find ways to overcome the joint challenges of the changeability of software, justified secrecy, and active adversaries. As the malware analysis example exemplifies, the combination of these challenges is particularly pernicious. If the adversaries could not make malware changeable in a way reactive to practitioners' analysis attempts, understanding adversaries' activities would be less daunting. If practitioners could share detailed results widely without tipping off their adversaries, this daunting burden could be shared and made easier. Alas, this is not the case.

SUMMARY In all three examples, cybersecurity practitioners build and use knowledge of stable activities, stable organization, and the properties of entities—recognisably mechanism discovery strategies. These mechanism discovery strategies overcome the three challenges and build general knowledge, though practitioners rarely use these words. These three examples each focus on a different aspect of what could be thought of as a multifield mechanism which collects relatively stable and shareable knowledge. Within the banking trojan example of money laundering, the organization of the parts of the mechanism is the focus; details of the entities and activities are secondary and remain abstractly specified. The intrusion kill chain provides a schema of activities that attacks contain, roughly organized in linear sequence, largely independent of the highly changeable entities involved. When studying malware, analysts are often interested in provenance, which equates to etiology or the historical sources of a malware file. Groups of attackers tend to reuse their past work rather than start over; similarity to past malware samples provides important understanding. While each of these examples builds its evidence through different strategies, they also mutually reinforce each other as part of a multifield mechanistic explanation.

4.4 BUILDING GENERAL KNOWLEDGE IN CYBERSECURITY

This section explores how the philosophical threads drawn from the mechanisms literature in Section 4.2 can illuminate the field of cybersecurity. I view the various strategies used by cybersecurity practitioners as mechanism discovery strategies used in the face of the triad of challenges. This mechanistic view extracts some coherent purpose behind what at first sight are wildly different actions by practitioners.

The examples elaborated in Section 4.3 demonstrate the depth of the triad of challenges for building general shareable knowledge in cybersecurity. There is no stable system into which one can make surgical interventions in the way of Woodward. The challenge is far more difficult. Even further, what cybersecurity practitioners are facing spans from the technical, in malware, to the social in the NCA diagram and international legal systems. Flechais et al. (2005) and Anderson and Moore (2006) claim that only by considering the socio-technical system as indivisible can practitioners make adequate security evaluations. This, along with the fact that many parts of that socio-technical system are highly responsive, means that cybersecurity is in a constantly evolving arms race between defenders and adversaries.

The examples in Section 4.3.2 also contain myriad successful responses to that challenge, and draw attention to how success was achieved and then publicised. This section will extract what successful responses share. In particular, the strategy in all three cases was to find what is relatively fixed in the midst of a changeable social and technological system, and, of that, what is publicised is what need not remain secret. What remains fixed varies in the different cases, but I will show that they can be seen as elements of mechanisms, in the sense discussed by Glennan. Further, just as in other scientific and technical fields, mechanisms in cybersecurity do not stand isolated and alone. Mechanisms both cluster within fields and are interrelated across fields, such that the three examples can be seen as parts of a multifield mechanism, as recognised by Darden and by Craver.

One can productively see synthesizing general knowledge as linking up mechanisms along these complex and highly variable lines. Indeed thinking of this as modelling mechanisms illuminates the coherence of that search for shareable general knowledge. Finally, cybersecurity both shares enough features with other fields which perform mechanism discovery that security researchers can use the mechanisms literature; while it has enough peculiarities to help develop that philosophical literature in interesting ways.

FOUR ELEMENTS OF MECHANISMS YIELD FOUR DIMENSIONS OF VARIATION *(Glennan)*: First, if one takes seriously the four dimensions of similarity of mechanisms that Glennan draws attention to then each of these very different examples can be seen as cases where practitioners search for some stability in the mechanism. Glennan's four dimensions for searching for similarity among mechanisms are the entities and activit-

ies involved, the organization of the components, the phenomenon for which the mechanisms are responsible, and the etiology or history leading to the mechanisms (Glennan, 2017).

These three cybersecurity examples each focus on a different aspect of mechanisms. Tracing banking trojan delivery follows the organization of the parts; modelling the adversary's kill chain follows the common activities involved across incidents, and also shows a reasonably stable organization; and finally malware analysis focuses on the history (which Glennan calls etiology) of how the file and its mechanism came to be. In all three cases, the practitioners use these as foci to work on other things they need to know. Note that none of the three cases focuses on entities. This might be accidental, but I suspect that, at the least, it reflects the changeability of the entities. To deal with this, practitioners look elsewhere to coordinate a response, and find out how to stop a particular piece of malware, attack in progress, or criminal organization.

This helps to explain why the surface actions of practitioners can be so very different, although there is a sense in which they are all working on the same very general problem. Notice also that it helps to explain why each example coalesces under a recognizable discipline within cybersecurity: internet architecture (banking trojan delivery), incident response (kill chain), and reverse engineering (malware analysis). Practitioners acquire considerable expertise within their fields. Nevertheless, each can be seen as focusing on an aspect of the highly changeable and reactive mechanism they are facing. And each makes the sensible choice of focusing on what remains most fixed, and of sharing information about what does not need to remain secret—because it is already known by, or cannot easily be changed by, the criminals.

THESE MECHANISMS ARE MULTIFIELD *(Craver)*: Another important way of seeing the coherence here is to understand the models of mechanisms cybersecurity deals with as hierarchically related, specifically multifield in Craver's sense. Craver (2007, ch. 7) argues that in the multifield research program of neuroscience, explanation of memory is best understood as a mosaic of interlocking evidence of a mechanism spanning multiple levels. The different fields locally depend upon each others' evidence. Each field provides support to the other; one is not reduced to the other (Kaiser, 2011). Seeing the three examples as combining into a multifield mechanism in this way is also useful for understanding how cybersecurity succeeds.

The examples of attack and crime modelling provide one example of this interlocking support. The kill chain model describes activities carried out over these distribution channels to which the adversary is constrained. Kill chain terms such as *delivery* and *exploit* describe steps in the banking trojan ecosystem at a finer level of detail. On the other hand, the banking trojan model expounds on the kill chain's final activity, *action on objectives*, to fill in what the objectives are (steal banking credentials) and explains how criminals use short-term objectives as stepping stones to accomplish their overarching mission. In this way each field supports the other; neither has primacy.

So the interrelationship of these three examples occurs in several ways. The criminal business model is a higher-level mechanism because the kill chain mechanism is contained within the criminal process. In another sense, the kill chain represents a mechanism schema which is partially instantiated by the criminal business model. The crime model instantiates the kill chain because it restricts certain kill chain activities, such as delivery, to the more specific methods of malicious web sites and phishing emails. The kill chain activity of command and control is also instantiated. The NCA criminal business model is specific to a particular botnet; in this specificity it makes sense as an instantiation of the purposefully-general kill chain model.

None of these relationships are strictly part-whole, nor is malware solely deployed to steal money. Nevertheless, for understanding this particular criminal activity – and stopping it – responders have to understand this multifield mechanism, forming a loose hierarchy where what is known about each level constrains how the whole can operate; that is, a hierarchy in just the way Craver describes.

CLUSTERS OF RELATED MECHANISMS, NOT UNITARY MECHANISMS *(Darden)*: While I selected three mechanisms that were interrelated in a way that illuminates the multifield and hierarchical nature of cybersecurity practice, it would be a mistake to think these are the only mechanisms in their domain. Instead, there is a great deal of clustering of related mechanisms, in accord with Darden's work. The models of the NCA mechanism, the kill chain mechanism, and malware reverse engineering mechanism are each quite abstracted, capable of being elaborated in different ways. So each is better thought of as an exemplar, allowing the practitioner to understand a cluster of related mechanisms.

Multifield mechanism models are a focal point for collecting and anchoring general knowledge. They do not describe one unified mechanism, but a cluster, or exemplar of related clusters. Alongside the other two mechanisms, the NCA's model of the mechanism of computer network attacks form part of a cluster that illuminates the phenomenon of a criminal campaign to steal money using the internet. I elide the technical details, but the law enforcement action included deceiving the criminals' communication, public awareness to help prevent and detect attacks, and seizing key physical computers in 11 countries simultaneously (Addis and Garrick, 2014). The successful execution of these interventions indicates practitioners developed shareable, general knowledge; I conceptualize these three examples as a loose hierarchy, and also within the cluster of multifield mechanisms forming this general knowledge.

The kill chain model provides a schema around which to cluster other mechanisms of other specific botnets. Both the original kill-chain work, and further work building on it, use the kill chain as a schema about which to cluster malicious activity for attribution of similar attacks to similar actors (Caltagirone et al., 2013). cybersecurity practitioners doing attribution of attacks cluster on targets, techniques & procedures, and malicious infrastructure. There is clear resonance here with the clustering features described by Darden, along the dimensions explored by Glennan.

Glennan (2017) can be used to illuminate how clustering works. Clustering mechanisms requires a feature on which to cluster. Darden and Craver make the case for hierarchy, but do not elaborate *what* about a mechanism is similar to another mechanism that permits building general understanding. Hierarchy is not enough to explain on what dimensions mechanisms are similar. Glennan's dimensions provide features on which to cluster, or features to guide or assess multiple fields investigating similar mechanisms.

There are interesting relationships within the picture of cybersecurity. These include the four elements of mechanism, clustering of mechanisms within a field, and hierarchical relationships across fields. These differences are all present in cybersecurity. However, the distinctions are not sharp. Perhaps they never are, but with a domain as fluid, flexible, and reactive as the changeable technologies and social systems of cybersecurity, one should not expect to find sharp distinctions and rigid boundaries. Whether a particular difference counts as variation in, for example, an activity, a different activity, or a schema instantiation, may often be indeterminate. This does not mean one cannot usually say something useful about relationships between

neighbouring activities, or between an activity and organization, within a specific context and purpose.

4.4.1 *Constraining: On improving coordination in cybersecurity*

The resemblance of mechanism discovery in cybersecurity to that in other disciplines is very useful. This section will indicate how this work might improve coordination in cybersecurity by seeing that the disciplines collaborate to generate understanding by adding *constraints* on the overall mechanism of online crime. As noted, Craver describes this happening in neuroscience (Craver, 2007; Darden, 2006; Darden and Craver, 2002). So, broadly, this development of Craver's view is that the discovery of new entities, activities, forms of organization, and etiologies can open or close space in the overall space of plausible mechanisms. This is how discoveries in one case study can impact on others. Seeing these relations across very different cases can show how knowledge in cybersecurity is available that is more general even than the knowledge that can be built of each case on its own.

First consider malware analysis, which relies on constraints from the other two examples for building general knowledge. For those families of malware that steal banking credentials, there is a remarkable coupling between defences by the financial services industry and malware capabilities added to circumvent those defences (O'Meara et al., 2016). Financial institutions add social defences that interrupt the business model. For example, sending PINs to customers' phones; malware authors quickly learn how to infect the phones and forward the PINs. Malware is also often categorized based on what stage of the kill chain it is involved in: initial exploit, permanent presence, command-and-control infrastructure, or achieving objectives. The very name banking trojan uses such a categorization: trojan malware is for a quiet, permanent presence, and 'banking' indicates what sort of objective it is used to achieve. Other malware might be categorized based on what vulnerability it exploits. So if you know what must remain stable on one of the other levels of the hierarchy, that constrains where you should look in your efforts to combat malware. Knowledge at one level is a guide to help to build knowledge at another level.

Malware analysis likewise supports and constrains kill chain analysis. A particular malware file can only run on a specific system, say a Windows PC or a Linux server. By indicating what the malware could possibly target, the mal-

ware analyst constrains what the potential *delivery* and *action on objectives* activities are. The kill chain model constrains where a practitioner might look to find malware; if a computer has been the recent target of reconnaissance, it is more likely malware has been delivered to try to exploit the computer. Glennan's work can help illuminate how constraints work, by finding the feature which is constrained, and in particular where that constraint will travel to related mechanisms. For example, if one knows what software vulnerability some malware exploits, then constrain the search for infected computers to those running vulnerable versions of that software.

The fixedness practitioners seek in the midst of the triad of challenges tends to occur at the boundary of the phenomenon of interest and another field or system that provides constraints. The sociological mechanisms[22] of how humans interact with the technology available through the internet are email and web browsing; thus these are the constraints on the distribution channels of banking trojans. The challenge in the case of the banking trojan business model is to determine how the various mechanisms of theft are organized. It turns out the organization of the mechanism tends to be similar even for different criminal organizations, especially if each criminal organization is under similar constraints. The criminals must distribute via the channels their victims are used to. The kill chain provides constraints on the activities: delivery before exploitation, for example. Internet architecture provides constraints on the entities for delivery, web and email. Criminological work such as the NCA model constrains the organization and can localize the elements of the mechanism: both web and email are used simultaneously in this case, and the task is outsourced to specialized individuals by the criminals. In this way, understanding how the three mechanism schemas (or clusters of schemas) above relate clearly yields much more general knowledge than understanding one alone.

At the other end of the multifield mechanism, constraints also operate in malware analysis. Here the changeability of entities is very marked, and a technological arms race exists between criminals and cybersecurity practition-

22 Social systems fall under what Hatleback and Spring (2014) categorise as stable in that they tend not to change purposefully to disrupt observation within the time from of usual observation studies. Social systems have been discussed as mechanisms for some time (Elster, 1989). Social and legal systems seem a liminal case between changeable mechanisms, such as software, and the more fixed nature of mechanisms in fields like chemistry. However, the salient feature is that at least some of the relevant social systems are much less changeable than malware. The local dependence of the malware on the social provides a dependable constraint on the changeability of the malware.

ers. Nevertheless there are important constraints. Hierarchy matters: malware is an entity within the kill chain; the activities of the kill chain are activities that the criminals undertake in their business. However, this overview does not adequately capture the nuanced similarities between these mechanisms on other dimensions. The criminal business model is specific to a criminal network named for the malware used to make the thefts: GameOver Zeus. The malware analysis that provides this name, specifically this family name of Zeus, is based on a etiological cluster of many malware samples, all of which have a common progenitor malware (O'Meara et al., 2016). Attackers automate and outsource where they can, but they are ultimately people, and can only build on what they have done before. And so the etiology of the attack, what is known of that source, is a useful guide to defenders.

The three examples are, in some sense, grouped around similar entities or activities investigated through different lenses. Constraints learned about one activity or entity in one example can impose constraints on the whole. Not only entities and activities, but also organization, phenomenon and etiology are important.

If this is right, then the impact of constraints hinges on being able to home in on one of the dimensions of similarity identified by Glennan (2017). When similarities among mechanisms extend to four dimensions of variation, one can see how the constraints work. There is no simple link between types of similarity among mechanisms and relationship between mechanisms, either within the same level or at different levels. Nor is there an easy link between aspects of interfield mechanistic explanation, i.e. mosaic unity, and similarity among mechanisms. However, for two mechanisms to be related, or two fields to interrelate, they must be related by something. These four dimensions of similarity provide a plausible starting point.

Ultimately, in the face of these challenges, cybersecurity practitioners have achieved a great deal. General cybersecurity knowledge supports practitioners, when building a security architecture or responding to an ongoing intrusion, because general knowledge indicates courses of action that will plausibly be successful. The boundary between general knowledge and case-specific knowledge is not perfectly sharp. Both chief information security officers at big companies and malware analysts are practitioners who use general knowledge, but what counts as general knowledge to the malware analyst probably seems very case-specific to the other. Nevertheless, the field as a whole knows rather a lot.

4.5 CONCLUSION

Returning to the chapter's initial philosophical questions, it should be clear that there are nothing like general laws in cybersecurity, and far from anything like a scientific theory in the traditional sense. General knowledge in cybersecurity is not gained by finding general laws or theories. General knowledge is as hard to win in cybersecurity as it is anywhere, due to the triad of challenges. Each of these—changeability of software, active adversaries, and justified secrecy—alone could frustrate generality.

Yet the case is not hopeless; cybersecurity has seen progress, however fitful. The initial strategy of looking to mechanism discovery from philosophy of science to illuminate this question has shown to be fruitful. One can see general knowledge built up by discovering clusters of related multifield mechanism schemas, expected to vary along four dimensions: activities and entities, phenomena, organization, and etiology. I demonstrated how this can help conceptualise work in cybersecurity by studying in detail three cases of active research problems within cybersecurity that can be seen as discovering—and sharing—mechanisms. The cases of mechanisms in fact vary along these dimensions. Although each cybersecurity case comes from a different field of expertise, and each focuses on a different element of mechanisms, the three cases are nevertheless strongly interdependent. This approach also happens to indicate something about what makes cybersecurity a coherent field. Just as Craver (2007) describes the 'mosaic unity' of neuroscience, built up by interleaving constraints applied by various fields on multifield mechanisms, I describe a 'mosaic unity' of cybersecurity. There is little else that joins applied logicians verifying software correctness with criminologists interviewing malware authors into a coherent field of practice.

Future work should study many further interrelated cases. The three examples show what is possible, but fall far short of defining a whole field. This chapter provided the beginning of a framework for understanding the way general knowledge works for other problems within cybersecurity. For example, one might apply such a framework to (in the language of mechanisms) other phenomena, such as skimming credit card details at point-of-sale devices. These are constrained by how the criminals monetize their gains, attack the point-of-sale device, and the malware's development history and how it hides itself. Practitioners build explanations and models in ways similar to these examples when, for example, examining how the point-of-sale devices at the

retailer Target, which has stores across the US, were leveraged to steal tens of millions of credit card details before Christmas 2013 (Krebs, 2014). Chapter 5 will study the kill chain example in more detail, examine other levels above and below it, and how CSIR analysts apply such general knowledge.

These explanatory models of mechanisms are not developed in isolation. For example, the authors of the NCA's criminal business model would have been aware of an attack model similar to the kill chain, if not the kill chain itself. The kill chain constrains the hypothesized organization of the criminals' activities. Delivery happens before exploitation. This matches the criminal business model. And still higher-level mechanisms are also relevant. For example, mechanisms understood from the fields of internet architecture and international finance also constrain the criminal business model. Via examples like this criminal business model, one can see how international finance places constraints on the kill chain. Cybersecurity practitioners have used such lessons to notice that, in some cases, the Visa payment system was the weakest point in a criminal's mechanism (Kanich et al., 2011). This example demonstrates one way in which constraints help lead to general knowledge. If an inter-field constraint applies to a part of a mechanism, and that mechanism is related to a second based on one of Glennan's similarities (in this case, the activity of criminal action on objectives via the kill chain), then other mechanisms similar along the same dimension to the same part of the mechanism may also be subject to the same constraint. Similarity among mechanisms provides a path for generalizing knowledge to other contexts.

This chapter has not touched on the sociological practicalities of how practitioners go about collaborating to build general shareable knowledge. The focus here has been on what to build, rather than how to build it, as having a target for what general shareable knowledge might look like seemed the priority. The most difficult communications will likely be across the clusters – between practitioners focused on different topics or with very different research methods. In this way, building general knowledge perhaps resembles building what Galison (1999) calls trading zones within the history of physics. People move between specialised fields, learning enough of the language of each to translate between them, in the trading zone driven by common need, such as the creation of a common tool. Indeed, Galison has applied this trading zone analogy to cybersecurity as part of the JASON report (MITRE, 2010) and at the first science of security symposium (Galison, 2012). Galison acknowledges that this kind of work is often under-appreciated, given its important role.

Galison's trading zones provide a description of what structures are needed (trading zones), but not how to build an effective trading zone. The description of general shareable knowledge as clusters of multifield mechanism schemas is likewise silent on how to build trading zones. However, at least with a description of what cybersecurity practitioners may want (an effective trading zone) and what it should produce (general shareable knowledge) one could evaluate whether a given social situation is effective. Boundary objects (Star and Griesemer, 1989), as well as conceptual translation, would likely be exchanged, negotiated, and re-interpreted in trading zones.[23]

There is another way in which the general knowledge of cybersecurity is not like building a theory in the usual sense, such as building the theory of relativity—at least not as that has been traditionally treated in philosophy of science. The knowledge of cybersecurity is very practically oriented, aimed at security, which is a very dynamic matter of constantly preventing and analysing attacks. This practical aspect still needs more attention within philosophy of science. Future work could integrate the ideas here with ideas developing the nature of evidence of mechanism in medicine (Clarke et al., 2014; Illari, 2011). Ultimately the goal would be to show how a particular entity, activity, form of organization or etiology in one place may be well evidenced in cybersecurity, and that evidence communicated effectively so that it may be made use of at other levels throughout the sociotechnical system. This line of questioning elicits a constructive research program and builds on where the summary of current philosophy of science application to security in Chapter 3 leaves off.

Even in the absence of laws, in a domain that is about as diverse and changeable as exists, and which has the special problem of secrecy, general shareable knowledge is still possible. This can be seen as the painstaking building of clusters of multifield mechanism schemas which vary along at least four dimensions of similarity: phenomena, activities and entities, organization, and etiology. Cybersecurity provides cases in which practitioners successfully build general shareable knowledge along these dimensions.

CSIR analysts should aim to learn such general shareable knowledge. It constrains what attacks are likely or feasible. It also conditions beliefs on what investigations are likely to succeed. That is, this account of general shareable knowledge bears on two aspects of the research question (see Section 2.7). The structure of general knowledge and the mechanism discovery literature inform

23 MITRE has connected the concepts of boundary objects in this sense to security. CCE specifically is called a boundary object in Mann (2008), but also presumably CWE, CVE, and so on are viewed this way. See Table 2.7 for other such data formats.

how to construct general knowledge about attacks. This account also suggests how to apply general knowledge, as a set of constraints on expectations and priors. Chapter 5 will elaborate on how to use general mechanistic knowledge further.

5

THE INTRUSION KILL CHAIN AS A CASE STUDY[1]

This chapter will provide case studies of how reasoning in incident analysis can apply mechanistic thinking. It primarily answers the research question: "How can an incident analyst apply general knowledge to improve analysis of specific incidents?"

The majority of the chapter will be dedicated to an extended translation and example manipulation of the kill chain model introduced in Section 4.3.2. This chapter bridges the more military language used in practice with more academic language. I will present two further case studies of good decision-making in CSIR and how that maps to mechanistic explanation and heuristics for problem solving discussed in Chapters 3 and 4.

The chapter will thus answer the research question in two parts. First, it shows how to interpret important practitioner-generated incident models as mechanistic explanations. Second, it presents two diverse examples of incident analysis to point out how and where good incident analysts make use of general knowledge such as incident models.

5.1 INTRODUCTION

Per Chapter 4, models are important to science and to rational inquiry more generally. The purposefulness of modelling is critically important. Computer Network Operations (CNO) is the general term that encompasses attack, defence, and exploitation using computer networks (Joint Chiefs of Staff, 2014b). Our purpose in modelling CNO by incorporating mechanistic thinking is to understand intrusions more thoroughly and, ultimately, to reduce the damage and disruption caused by CNO. Having such a model enables both better CSIR and better CND because responders and defenders can more adeptly understand the situation by interpreting it via the model, despite the occasional oversimplification. Incident responders may not in practice think of themselves as scientists. However, as argued in Chapter 3 as well as previously in Hatle-

1 This chapter is based on joint work, the paper: Jonathan M Spring and Eric Hatleback (Jan. 2017). 'Thinking about intrusion kill chains as mechanisms'. In: *Journal of Cybersecurity* 3.3, pp. 185–197.

back and Spring (2014), analysts stand to benefit from adopting techniques honed by scientists. The intent of this chapter is to demonstrate how analysts can integrate mechanistic thinking as an example of, and gateway to, scientific techniques.

As this chapter is targeted more at the CSIR practitioner, it may be necessary to motivate why the modelling choice from philosophy of science is worth the effort above and beyond the modelling languages to which computer scientists are more accustomed. For example, it may seem more natural to use a modelling language like Unified Modeling Language (UML) since it is a common model to design computer systems (Larman, 2004). UML and other software engineering models are not incompatible with scientific modelling via mechanisms. However, unlike a systems engineer, and like a scientist, the security practitioner attempting to understand an incident must build a model that includes physical, human, and engineered elements. Also like a scientist, the security analyst must form a descriptive model of how the world is working, unlike an engineer, i.e. a designer, whose model goal is to satisfice particular desired features of the design (Simon, 1996, p. 119ff). Chapter 6 will explore the interplay between a model of how the system works, a design of how the system should work, and mathematical models. However, here we focus only on elaborating the explanatory or scientific aspect of modelling for incident analysis.

The kill chain model (Hutchins et al., 2011) is a *de facto* standard for CSIR (see Chapter 2) and this chapter takes it as an extended example. The kill chain model is not complete or perfect, but it is a shared starting point. The model has enough detail to enable an instructive elaboration of what we might describe as the cluster of mechanism schema describing an attack. I will take the structured advice created in Chapter 4 and now apply it to improving the kill chain model.

Recall that mechanisms are comprised of entities and activities. I will describe the seven elements of the intrusion kill chain as activities. Thus, one material improvement provided by this chapter will be to identify the entities the kill chain leaves implicit. Thinking about the attack process mechanistically will allow me to import the detailed modelling and design of scientific observations to enrich incident analyst's understanding of adversary CNO. That is, this chapter is the hook by which I will demonstrate how the prior discussion of science and general knowledge has direct practical impact for how an incident analyst should reason. The impact of merging how incident

analysts think with mechanistic modelling is to make CSIR more 'scientific' (Recall from Chapter 3 I mean scientific as "a very prestigious label that we apply to those bodies of knowledge reckoned to be most solidly grounded in evidence, critical experimentation and observation, and rigorous reasoning" (Dear, 2006, p. 1)).

There are various parts of an attack. Understanding how these parts interact, or potentially could interact, is a practical challenge. I propose clusters of multi-field mechanisms help organize analysts' understanding. For concreteness, consider the exploitation mechanism of a drive-by download, an example I use later in Figure 5.3. Within CSIR, a practitioner doing each of prepare, protect, detect, triage, and respond (Alberts et al., 2004) will emphasize different questions given the mechanism for a drive-by download. Recall that incident analysis occurs within both triage and respond phases. Within the detection phase, one may ask what aspects of the mechanism distinguish it from benign traffic. Within the preparation phase, one may ask how commonly, and what sorts of, adversaries tend to use drive-by downloads to determine resource allocation to protection and detection. During the response phase, an analyst might want to determine whether the exploitation mechanism is well-known, in which case response may be easier or more reliable. Information such as mechanism of exploitation also contributes to wider campaign analysis. And so on.

As Chapter 4 selected examples demonstrating the inter-related levels of mechanisms in security (the kill chain is lower-level than the GameOver Zeus money-laundering mechanism), this chapter continues to elaborate inter-related examples. The incident analyst naturally takes advantage of this feature; an analyst asks about lower-level details that are distinguishing for detection, and about higher-level details of actors for preparation or attribution (for example, Caltagirone et al. (2013)). This example also highlights the way in which a single mechanism is not a stand-alone explanation, but rather contributes to knowledge by interrelation with other mechanisms.

Although I primarily elaborate the kill chain, Section 5.2.1 situates attacks (i.e., the kill chain) within the broader context of campaign analysis. The standard campaign analysis model identified in Chapter 2 is Caltagirone et al. (2013). However, I will ease into the concept by introducing the simpler campaign model of Howard and Longstaff (1998), which situates attacks in the context of adversaries and their objectives.

When translating models into mechanistic models I will find some explanatory gaps using the heuristics described in Chapter 4. I will propose refinements to both Hutchins et al. (2011) and Howard and Longstaff (1998) that bridge these explanatory gaps and provide a more robust model of CNO.

The rest of the chapter is organized as follows. In Section 5.2, I move forward with expressing the kill chain as a mechanistic model. Subsections demonstrate knitting together the general knowledge of attacks expressed in the kill chain with both higher- and lower-level knowledge. Section 5.2.1 provides an example of incorporating the kill chain model of attacks into a more coarsely-grained model of CNO, and Section 5.2.2 presents an example member of a mechanism schema cluster that explains a specific activity of the kill chain. In Section 5.3, I highlight two historical examples of incident analysis where the decision-making has been well-documented. The purpose of these cases will be to suggest similarity between mechanistic reasoning and good analysis cases as well as to provide suggestive examples of what sorts of processes the logic in Chapter 7 will be expected to capture.

5.2 REASONING WITH THE KILL CHAIN AS A MECHANISM

This section will demonstrate that (some) general knowledge in cybersecurity can be represented as a mechanisms and the process of using heuristics from the mechanisms literature to refine an item of general knowledge. This process will yield an incremental improvement to the kill chain. But the focus is to demonstrate the process, rather than the resulting improvement.

Attack ontologies are not the only type of general knowledge in cybersecurity that could benefit from mechanistic modelling. For example, the Bell-Lapadula (BLP) model (Bell and LaPadula, 1973) for multi-level secure systems could cleanly be cast as mechanisms: subjects and objects are two types of entities, and activities are the classic actions initiated by subjects such as read and write. BLP then describes what set of mechanisms lead to the desirable behaviour of a secure system. Incident management processes (Alberts et al., 2004) are easily cast as mechanisms, with activities such as detect, triage, and mitigate. Cyber incident attribution (Caltagirone et al., 2013) and cyber threat intelligence (Chismon and Ruks, 2015) also use models that could be translated to a mechanistic model. The kill chain model is a good starting place, however, because attacks are complex enough to provide interesting challenges while remaining tractable.

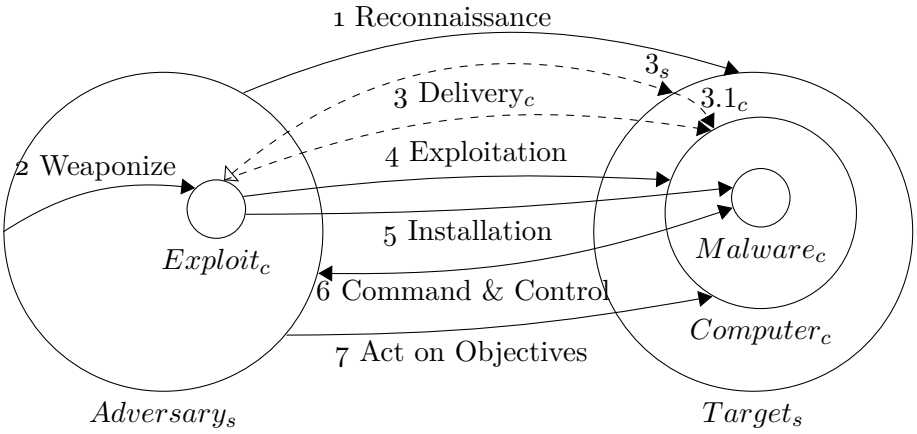

Figure 5.1: Improved Kill Chain mechanistic diagram, where delivery (3) is replaced by two options, an software-defined changeable activity directly to the target (3_c) and a more stable activity through human deception (3_s) with an optional secondary software delivery step (3.1_c).

At a coarse level of granularity, one can simply cast the kill chain as a mechanism. The entity acting in each case is an "adversary." The activities are the seven steps introduced in Section 4.3.2. These activities each require an entity as an object as well as an actor, and so at a coarse granularity, the object of the activities is the "target," or defender.

But this coarse-grained description abstracts away too much information to be useful. For example, the adversary and the target are not the only entities. There are finer-grained entities that are necessary to map the kill chain model accurately as a mechanism, such as "remote access trojan" (Shirey, 2007, see "backdoor" and "Trojan"), "exploit" (Seacord, 2005), and "victim system" (Shirey, 2007, see "system component," "system entity"). These entities have coherent definitions in the works cited, despite the fact that the kill chain paper does not define them or reference definitions (Hutchins et al., 2011). Employing a mechanistic understanding helps make clear the relationship between these entities in the kill chain. Figure 5.1 provides a conceptualization of the seven steps in the kill chain mechanism.

Hatleback and Spring (2014) labelled entities and activities with subscript e or p to indicate whether the element is changeable software (i.e. 'engineered') or not (i.e., 'physical'). This prior work makes a useful heuristic distinction between mechanisms that can change at the will of an adversary in time comparable to the expected period of observation. A practical example is malware during reverse engineering. In retrospect, perhaps Galison's terms 'Au-

gustinian' (no intelligent adversary) and 'Manichaean' (with an intelligent adversary) would be less prone to confusion or semantic collision (Galison, 2012). I will rename the heuristic from Hatleback and Spring (2014) as 'changeable within the time period of observation', c, or 'stable within the time period', s, to make a clear connection to the challenge of changeable software discussed in Chapter 4. Whatever the term, the distinction is pragmatic for incident analysts. If there is a mechanism under observation that can be altered by an adversary during observation, the design and interpretation of the observations need to be much more careful.

Some elements, in this case delivery, have multiple paths, some Augustinian and some Manichaean. One important contribution of Hatleback and Spring (2014) is to note that these different paths should be teased apart for more reliable analysis. Figure 5.1 incorporates this change by splitting delivery in two, refining the kill chain model.

The activities, as arrows labelled 1 through 7 in Figure 5.1, represent the seven steps of the kill chain.The diagram provides a richer interaction describing the phenomenon by including the entities and their organization (in that some entities are sub-parts of others). First, the adversary performs reconnaissance on target. Second, the adversary weaponizes exploit delivery code. Third, the exploit is delivered to the target. Fourth, the weaponized $Exploit_c$ compromises the victim computer. Fifth, the exploit installs malware of some kind on the victim system. Sixth, the malware communicates over a command and control channel with the adversary. Seventh and lastly, the adversary completes actions on objectives against or with the victim system.

Some activities have aspects that are changeable software elements mixed with elements that are stable during the time period of an attack. To say this confidently, one must be able to understand the activity as a mechanism – that is, move to a lower level – and have an explanation for the activity that includes changeable and non-changeable components on the relevant time scale. The activities having both types of component are reconnaissance (1), command and control (6), and actions on objectives (7). An incident analyst may intuitively learn these distinctions, because whether something is a result of changeable software or not rather intuitively changes what sort of observations one can reliably make. So, for example, analysts should learn about the different aspects of reconnaissance with different techniques.

I will walk through the parts of recon in some detail to demonstrate this differential study. Reconnaissance is defined as "research, identification, and

selection of targets" (Hutchins et al., 2011, p. 4). Research on and selection of targets are human activities. Adequate explanations are available using the methods of psychology or economics. For example, the GameOver Zeus botnet, discussed in Chapter 4, tends to select as targets owners of bank accounts in wealthier countries. Another factor likely influencing this targeting is that Russian citizens are discouraged from targeting other Russian citizens (it is illegal in Russia), but not discouraged from computer fraud generally. On the other hand, identification of targets is often accomplished by software scanners or other computer tools, such as the open-source Nmap (Lyon, 2011). If an analyst wants to explain reconnaissance against her network, it would be wise to separate these aspects of reconnaissance for separate inquiry. My intuition is that the study design for stable and changeable mechanisms should be different. Similar considerations around changeability are important for understanding command and control and actions on objectives, which I elide here.

Constructing a more detailed, mechanistic model of the kill chain helps makes clear to an analyst both areas of high detail and the areas in which more detail is needed. A target likely has multiple types of systems and components arranged in a system architecture. The target should know its own architecture well enough to list these entities; if it does not, it should prepare better.[2] Then one could define exploitation in detail against each of these components, as I will do in Chapter 7.4 for delivery. For another example, databases such as CWE (MITRE, 2015) and CVE (MITRE, 2012) are essentially collections of such finer details of exploitation; they could be understood as clusters of lower-level mechanism schema elaborating the exploitation activity (schema because they represent vulnerabilities; actual exploit code would instantiate the schema of a CVE, yet another level down).

Mechanistic modelling also helps identify what is not yet known. If the defender needs to complete an assessment of an incident, a mechanistic diagram helps identify gaps. Given a mechanism schema for an attack such as the kill chain, the analyst can compare the schema to the current assessment of what is known. If, for example, installation and actions on objectives have been identified by the analyst, this comparison should immediately provoke the question of how command and control happened. More generally, the comparison to relevant general knowledge about how adversaries proceed will

2 Preparation is out of scope for incident analysis, but these sorts of practices are recommended as part of incident management standards noted in Chapter 2, such as Alberts et al. (2004).

produce a list of leads for further investigation. In Chapter 7, I will discuss this reasoning process more formally as abduction.

The information linkages across different aspects of the kill chain may be complex. For example, certain types of adversaries may use a certain control channel; knowing this activity helps identify the adversary and potential goals. This line of reasoning is precisely the kind of disciplined thinking encouraged by the IC de facto standard attack models (Caltagirone et al., 2013; Hutchins et al., 2011). I am suggesting to enrich their thinking with heuristics for hypothesis generation and gap identification along clusters of multi-level mechanism schema, as Chapter 4 described.

A complete diagram of every possible computer network attack could not be readily comprehended by a human. Similarly, a maximally detailed view of how the human body works cannot be readily comprehended and used by a single doctor. Computer security and intrusion analysis should encourage a way of studying its complex system at different levels as needed. Mechanistic knowledge facilitates this goal.

Different levels of granularity already are recognized in computer security. For example, there are large-scale network analysts, and host-based analysts. The OSI layers are a form of granularity levels with abstraction and encapsulation (ISO/IEC, 1996). But the difficulty of communicating important information across levels of abstraction and among professionals with different specializations has not yet been overcome. By importing mechanistic thinking and attendant good scientific practices, I believe these communication deficiencies can be overcome.

A final benefit of structured mechanistic knowledge is to improve identification of areas for improvement. For example, delivery (3) is defined as "Transmission of the weapon to the targeted environment" whereas exploitation (4) is "after the weapon is delivered to victim host, exploitation triggers [malicious] code" (Hutchins et al., 2011, p. 4). In the kill chain, a gap exists between 3 and 4, since the target is different, though the entity acting appears to be the same. Thus, the approach permits us to question whether the definition of delivery is accurate or whether there is an additional activity describing how the weaponized code transits the target environment to get to the victim system. Gaps such as these are more easily identified when thinking mechanistically; one goal in a complete mechanistic description is identifying explanatory gaps (Machamer et al., 2000, p. 3).

I propose that delivery (3) is better understood as one of two activities, one for computer targets and one for human targets. In a phishing email, there is a malicious link or file delivered to the human's email inbox. However, the exploitation (4) does not occur unless the human is tricked into opening the malicious content. For a clear example of delivery to an purely computer target, consider the old ping-of-death vulnerability: as soon as the target computer received the malicious packet, it automatically processed it and crashed. Therefore I split delivery into delivery_s, where a human is in the loop as the target, is a distinctly different activity than delivery_c, where a machine will be exploited automatically without human action. This stable type is readily studied by existing experiment design techniques in psychology, in deception for example. But researchers may or may not find that the deception techniques have a stable psychological explanation year-on-year, the length of robust psychology studies, even though they do tend to be stable for the duration of a computer-security incident. It is an important feature of this stability / changeability label that it is contextual for the period of observation and study.

This example of identifying a gap in the explanation warrants further exploration. One may ask how much the mechanistic approach to modelling has actually enabled this gap-finding. The modelling language brings this gap to the fore quite naturally. By doing this translation, the gap is readily apparent, when it was not before: there is no activity linking the effect of delivery of an exploit and the installation of the malicious code. The mechanistic language also provides a ready heuristic for improving the model. To ask what fits in that gap; namely, the user clicking a link, bypassing a warning, viewing an email with a malicious font, or perhaps automated system action. While the content of the model and improving the model both require subject matter expertise, translation into a mechanistic model provided a check on the explanatory soundness of our models by allowing ready inspection for gaps. Thus I argue it is the structure of thinking mechanistically that is the key enabler here, not any sort of cleverness of the analyst.

This mechanistic approach to understanding incident analysis does not rely on the validity of the kill chain per se. The kill chain is my example because it was identified as a de facto standard in Chapter 2. Other attack ontologies can be similarly understood as mechanism schema. Some such attack models take a different view of what is important or what to model than the kill chain. For example, Bejtlich (2004) models an attack as "Reconnaissance, Exploitation,

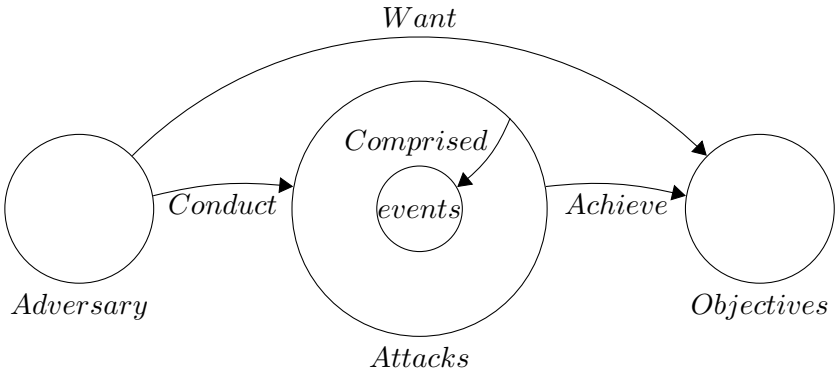

Figure 5.2: Visualization of the common language for computer security incidents (Howard and Longstaff, 1998) as a mechanism, where "attack" is simplified to one entity rather than the finer granularity of the kill chain's details.

Reinforcement, Consolidation, Pillage". This model nearly matches the kill chain, but has fewer steps. The three steps in the kill chain of weaponization, delivery, and exploitation are subsumed into just exploitation. Reinforcement is synonymous with installation; pillage implies a smaller scope than actions on objectives. Consolidation includes the control of compromised assets, and so largely mirrors command and control.

One final question may be whether explanatory soundness is valuable in CSIR, specifically the analysis step (per Chapter 2). The aim of incident analysis is a robust explanation of what happened. By treating incident analysis like a science (per Chapter 3), and applying structured knowledge via mechanisms (per Chapter 4), I have demonstrated how a practitioner can materially improve the models they use for individual attacks. In the following, I will explain how to further link such improvements across levels of mechanistic explanation.

5.2.1 *Higher-level mechanisms*

Other models help put attacks in context. The diamond model (Caltagirone et al., 2013) explicitly puts the kill chain model of attacks in the context of campaigns – a coherent, long-running operation by an adversary. The diamond model is one of the de facto IC standards identified in Chapter 2. The Howard and Longstaff (1998) model is less complex than the diamond model; I use it as my initial illustrative example. The Howard and Longstaff model involves 'incidents', which are comprised of one or more attacks, whereas the

kill chain models only single attacks. The natural approach models a single attack as a lower-level mechanism within the larger mechanism of an incident or campaign.

The taxonomy of Howard and Longstaff (1998, p. 16) translates naturally to a mechanistic way of thinking. The taxonomy presents a temporal ordering of items for an incident, namely "Attackers -> Tool -> Vulnerability -> Action -> Target -> Unauthorized Result -> Objectives". The taxonomy groups these items usefully; "events" are made up of just the "Action -> Target" sequence. If events are understood as part of a mechanism cluster about attacks, we can switch perspective to the kill chain and naturally ask which of the seven steps of the kill chain an event represents (if any). This sort of perspective will be captured in logic in Chapter 7. "Attacks" (that is, what the kill chain models) stretch from the "tool" to the "unauthorized result."

Figure 5.2 presents a coarse visualization of the common language for security incidents (Howard and Longstaff, 1998). Since there are more entities than activities specified, mechanistic modelling quickly identifies some gaps. For example, an adversary must *conduct* an attack. What that activity entails should be specified. Likewise, with respect to how "attacks" relate to "objectives," attacks *achieve* objectives, although exactly how this unfolds would benefit from further specification. Howard and Longstaff (1998) discusses "success and failure" of objectives related to whether the objective was achieved, but they provide no robust description of what this means. The mechanistic diagram highlights the importance of understanding this activity in order to understand the whole incident. Mechanism discovery strategies such as identifying gaps (Darden, 2006), as surveyed in Section 3.4, should complement CSIR analysis nicely.

Mechanistic modelling also eases integration of other security research that can inform CSIR. For example, Liu et al. (2005, fig 2) use a primitive attack model but graft a sophisticated game-theoretic model of attacker intent (Liu et al., 2005, p. 89ff) onto it. By seeing these two elements as separate but related mechanisms, the analyst can upgrade the primitive attack model with the more appropriate elements from Howard and Longstaff (1998) or Hutchins et al. (2011). At the same time, these attack models can benefit because of readier access to a more detailed model of attacker intent. As Figure 5.5 will demonstrate, game-theoretic models are still compatible with mechanistic understanding of phenomena.

I also find it natural to explain campaigns mechanistically. Caltagirone et al. (2013) suggest targets, attackers, infrastructure, and tools & tactics as the four important analytic categories. These form the four points of the eponymous diamond. The entities that our mechanistic understanding of the kill chain model brought out, such as the adversary's computer, are also explicit in the diamond model; in the case of the adversary's computer, that is 'infrastructure'. The various dimensions of similarity in Chapter 4 – entities and activities, phenomena, organization, and etiology – can complement these pragmatic categories and perhaps guide analysis of how the campaign is achieved. In some sense, the attack types in Howard and Longstaff (1998) could be understood as a subset of Tools, tactics, and procedures (TTPs) within Caltagirone et al. (2013). So these various models need not be seen as competing, or one strictly better than another. As Chapter 4 described, understanding the complexities of how the cluster of explanations interrelate is a sign of successful knowledge building.

5.2.2 *On lower-level mechanisms*

To be practically useful, an item of general knowledge will have to be able to explain real attacks and help investigations of them. Figure 5.3 models an obfuscated drive-by-download delivered via a malicious ad network, as described by Segura (2014). The mechanism is an example of the *delivery* activity from the kill chain examined at a finer granularity. This diagram focuses on the technical aspects of the delivery to the user's browser. The mechanism for how the ad networks select an ad for a user is left at a coarse grain, but could be modelled in more detail if more data becomes available (Preimesberger, 2015). Identifying such an item that requires more research because it is not clearly understood is a benefit of mechanistic modelling.

Most entities and activities in Figure 5.3 have their common English meaning and the same modelling norms as Figure 5.2. A special case is "fetches," which is transitive. That is, if the user's browser fetches a web page, which fetches an ad, which fetches a Uniform Resource Locator (URL), the browser has made an HTTP request to fetch all three of these things. Modelling it this way preserves which resource is redirected to which other resource, while an arrow from the browser to each resource would not.

It would be sensible to model the activities 3, 7 and 10 in Figure 5.3 as yet finer-grained mechanisms. Activity 3 is a sort of automated target selection,

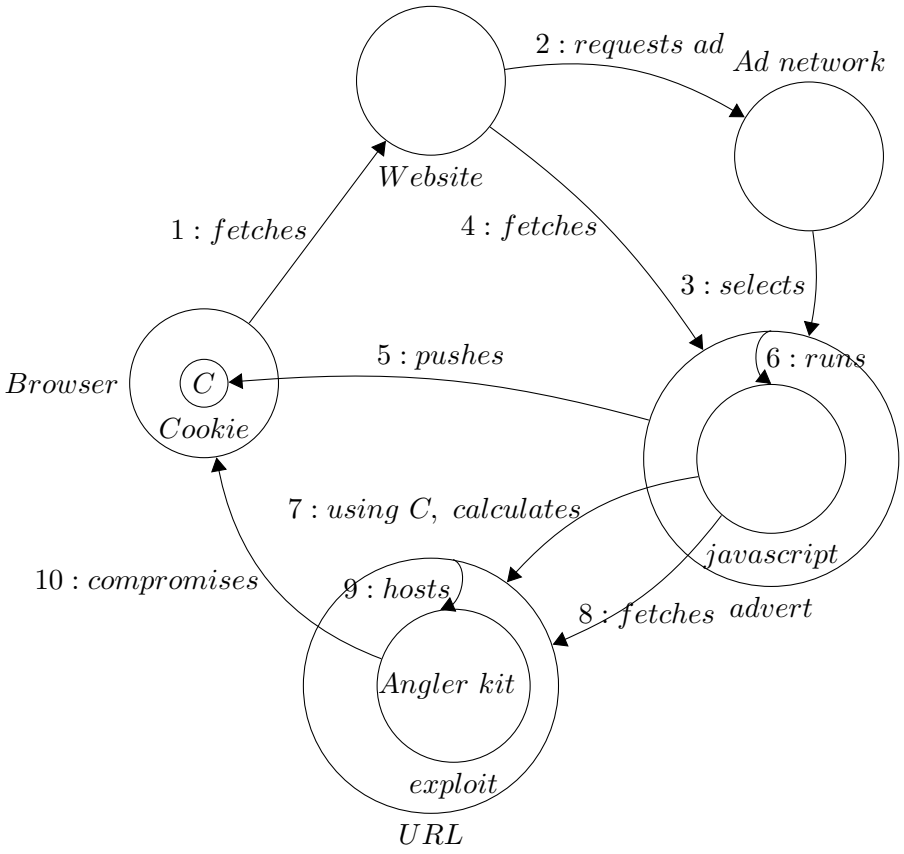

Figure 5.3: Diagram of an obfuscated drive-by-download delivered via a malicious web advertisement

which if an analyst could expand might provide insight into adversary targeting interests. Activity 10 indicates the end of *delivery* and the beginning of *exploitation*. Material to our interest in deepening understanding of this particular *delivery* mechanism, consider activity 7; it is the calculation by which the JavaScript in the advert de-obfuscates the malicious URL. Per Segura (2014), the mechanism is to use a regex to extract a string from the cookie and then unescape, split, and reverse that string, which yields a JavaScript HTTP request to a URL to be fetched (step 8). This mechanism defines a class of obfuscation techniques using cookies; the report by Malwarebytes (Segura, 2014) goes one level of granularity finer and specifies items like the specific cookie and regex used.

Thus the analyst naturally moves to three levels of mechanistic explanation below the kill chain – delivery as a drive-by download, URL calculation and obfuscation within drive-by downloads, and specific obfuscated URLs in a specific instance of an attack. By understanding the relationship between the levels of explanation, the analyst can orient themselves within its structure to

provide context to each mechanism cluster. The creation of Figure 5.3 demonstrated how to use mechanistic modelling to move fluidly between levels to make sense of detailed cybersecurity events. An explanation can link specific obfuscation techniques, to how such techniques relate to delivery of exploits via Figure 5.3, to how delivery fits into attacks via the kill chain, to how attacks fit into adversary's human objectives via Howard and Longstaff.

5.3 EXAMPLES OF INCIDENT ANALYSIS

So far, this chapter has been about how existing models of security incidents – attacks and campaigns – can be presented as mechanistic explanations. That is, the kill chain and the diamond model, for example, are a kind of general knowledge related to CSIR. These models are general knowledge painstakingly built up over time, despite the challenges described in Chapter 4, to explain something about how security incidents tend to happen. The chapter so far has also demonstrated the multi-level context switching in the context of mechanisms and how that connects these models. Next, I plan to demonstrate how such general knowledge can be applied during CSIR analysis. Because these are models from the intelligence community, there are not canonical public descriptions of their application. However, the diamond model cites Stoll (1989) as an inspiration, and so it seems plausible to take that as one example.

In this section I will use two exemplary instances of analysis to highlight how incident analysts should make use of general knowledge and structured reasoning. General knowledge serves two primary purposes in these examples. The first, familiar from other scientific endeavours, is to assist in the generation of hypotheses. Many of the strategies of Bechtel, Darden, and Glennan touched on in Chapter 3 and Chapter 4 are recognizable in the following examples. The second purpose of general knowledge, more specific to security, is to identify when something seems wrong; that is, when something may indicate a weakness, a flaw, or a violation of security policy. In this interpretation, security policy understood by the analyst as general knowledge about how the system should behave. No human can memorize thousands of lines of router configurations and access control lists. At a human level, any security policy must be understood in broad strokes.

I will discuss structured reasoning in more detail in the following chapters. My goal is to create a logic to express structured reasoning in incident analysis

in Chapter 7. The following examples put important constraints on this logic to come. Broadly, these constraints will be uncontroversial. Incident analysts reason about events in the past; CSIR analysts reason about events on computers; and they care about composing different incidents together to explain campaigns of malicious activity. Other features of structured reasoning have important details, such as how to integrate a(n) (in)validated hypothesis into one's model of the attack. I will bring such details out in the examples.

This section contains two examples. The first, obviously applicable to CSIR, is from the famous Cuckoo's Egg (Stoll, 1989). The second, less obviously applicable to CSIR, is the famous application of game theory to deployed decision-making at Los Angeles International Airport (LAX) via the Assistant for Randomized Monitoring Over Routes (ARMOR) tool (Tambe, 2011). We use this non-computerized example to analyse a security situation with explicitly structured, formalized decision-making. Access to results of applying such formal decision-making to CSIR are not publicly available,[3] so I make do with an analogous physical-world case.

5.3.1 *Example 1: The Cuckoo's Egg*

Stoll (1989) recounts a laborious intrusion detection investigation by the computing staff at Lawrence Berkeley National Laboratory (LBNL) in the late 1980s. The tale represents a comprehensive example because the investigation evidences many of the good practices currently recommended during intrusion analysis per Caltagirone et al. (2013) and all of the canonical stages of a computer security incident identified in Howard and Longstaff (1998) and Hutchins et al. (2011). This similarity is despite predating these models by a decade or two. This example is a case study of an investigator's thought process through hypothesis generation guided by general knowledge, seeking data, integrating this data into her models, and iterating until the investigator is satisfied. Stoll makes for a good case study, even retrospectively, because his training as an astrophysicist lead to suitable notes about his process. The format, a book targeting a lay audience, also provided space and motivation for introspection in a detailed way that modern forensics specialists rarely include.

3 Decision-making systems in cybersecurity are predominantly naive pattern-matching rules, such as with NIDPS, or based on machine learning. These approaches have their successes. However, in either case, the decision-making is not formalized and accessible in the sense meant in the logic or game-theoretic communities.

The story opens with the investigation of a small accounting error. Stoll gathers qualitative data on what entities within the whole-lab system are involved in calculating charges for computer usage. The initial hypothesized mechanisms cannot distinguish between the error resulting from the accounting system that collects usage data, or the billing system that charges the academic departments for their usage.

From this basic model the investigator asks several questions about how to best resource further inquiry. Since there are two likely candidate error sources, the clear approach is to attempt to eliminate one. To this end, Stoll performs what one might now call a code review of the accounting system. The most important features of a code review are that it is accurate, clearly documented, exercises all parts of the system, and does not damage the system. These four features track remarkably well to the four primary features desired of experiments in computing more generally: internal validity, transparency, external validity, and containment (Hatleback and Spring, 2014; Rossow et al., 2012), respectively.

The process of the code review impacts the investigator's confidence in his results. Confidence in a technique is specific to the relevant domain and probably can only be precisely expressed in its own jargon. The value assigned to confidence in a process may change as a domain develops more refined techniques. This discussion of incident analysis generally abstracts away from the technique specifics, although psychology of intelligence analysis may provide a framework for robustly assessing confidence (Heuer, 1999).

Abstraction from specific techniques cannot lose the importance of specifics in assigning confidence and belief. In this example, the code review results in the investigator eliminating the accounting system as the source of the error. As Figure 5.4 demonstrates, this is because the review provides reliable evidence that each of the component entities and activities of the custom system mechanism function as expected. The UNIX OS accounting mechanism stands as the only remaining source, which now becomes the hypothesis. This pattern of model-building, abducing likely outcomes, testing predicted outcomes, and revising the model is extremely common. Both the scientific method and mathematical modelling follow it (Collinson et al., 2012b), recall also Chapter 3. Any adequate decision support for incident analysis must address how to assign and warrant such abduction and model changes adequately.

This is one example of how Stoll warrants his belief. The whole investigation continues on this loop, through a combination of extrapolating on his

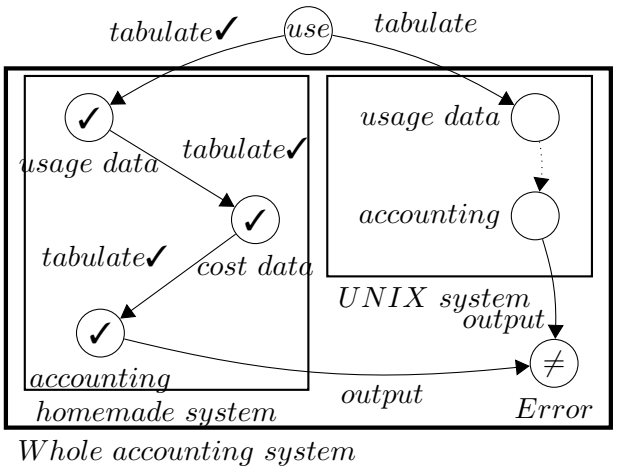

Figure 5.4: Stoll's model of the two concurrent accounting systems, including verification efforts on the home-made system (Stoll, 1989, p. 5). I am representing his belief as a mechanistic model. Circles are entities, arrows are activities, and boxes surround phenomena of interest. Check marks represent an experimental result indicating the element is not malfunctioning. The dashed arrow is assumed activity. This experimentation provides evidence that the error is not due to a flaw in the home-made system.

physicists' experiment-design training and inventing new cybersecurity tools, such as his own NIDPS. Figure 5.4 represents the investigator's belief at a time. On page 5 of the tale, Figure 5.4 is a reasonable assessment of the situation on the computer system. On page 100, it is not.

Stoll abduces the cause of the user's insertion into the billing system as either a human paperwork error or a system data-handling error. He sets about to test this new model. The human error is given more weight based on the investigator's prior knowledge. Bayesian statistics gives a useful account of prior knowledge (Kadane, 2011). Some accounts of incident analysis rely on Bayesian reasoning heavily (Caltagirone et al., 2013). My approach will be that all that really matters is the preferred order of prior knowledge, and not their relative "distance". Thus, logic is an adequate tool, and Bayesian reasoning would be overkill. Humans do not reason with statistical precision about preferences, and humans are famously non-statistical, especially with very high and very low probabilities (Kahneman et al., 1982). One feature my logic should have is to preserve the ability to prefer certain heuristics or beliefs over others, while dispensing with the computational overhead of the precise statistics.

The model in Figure 5.4 demonstrates a shift in focus on a lower-level, or more detailed, mechanism. Initially Stoll thought of the accounting system

as one single black box. Figure 5.4 represents his model after examining the system in detail. I present his belief as a mechanistic model; this is a small interpretive leap from of what Stoll reports himself, but it is clear from the text that he views himself as behaving scientifically and making a scientific model of the incident. Circles represent entities. Lines represent activities. Arrowheads represent the organization of the activities — which entity is acting on which along the line. Boxes surround discrete phenomena of interest. I use these elements of a mechanism — entity, activity, organization, phenomena — in the sense defined by (Glennan and Illari, 2017). The dotted line represents an activity that is assumed to exist but is not measured. The check marks represent that Stoll performed some test to warrant his belief that the entity or activity is performing as expected and is not responsible for errors.

This ability to change focus highlights an important benefit of mechanistic modelling. Specifically, it is a benefit laid out in the mechanism discovery literature; I have touched on that briefly in relation to alternatives to scientific laws (Section 3.4.3). The most brief explanation is that when seeking an explanation for a phenomenon, the structure of mechanisms helps by constraining plausible explanations that are considered or sought. More particularly, the mechanism in question is *decomposed* into parts, and the different properties of the system are *localized* within certain components (Bechtel and Richardson, 1993). Decomposition and localization are the two key mental heuristics that Bechtel and Richardson (1993) identify in scientists' work that make mechanistic modelling an effective method. Stoll is doing it naturally, as this heuristic has suffused scientific problem solving. In this example, he decomposed the accounting system, localizing tasks to subsystems, and checked that each subsystem performs the task accurately. I seek to preserve this compositional reasoning as much as possible; I will take advantage of Separation Logic for this feature in Chapter 7.

The process of warranting a belief in an individual component of a mechanism may be simple or quite involved. In this case, Stoll (1989, p. 5) recounts it took a day to write a test suite to verify the in-house accounting system. In a contrasting modern example, the security firm Mandiant spent more than a year warranting their belief that a particular PRC army unit in Shanghai was an explanation of security incidents against over 100 of their clients (Mandiant, 2013). Much like in science generally, there is no hard and fast rule about when data is adequately warranted (Norton, 2015); however, analysts follow and continually refine guidelines.

The investigator's action following building the satisfactory model of Figure 5.4 reflects his goal. Common goals in incident analysis include attributing attacks to a particular adversary or group, legal prosecution of a person, or fixing the impacted system. At different stages in the narrative, Stoll (1989) manifests each of these goals. However, at this early stage his goal is to eliminate a small accounting error, that is, fix the system. So his action, accordingly, is to delete the offending user (Stoll, 1989, p. 7). This goal and subsequent action is entangled with the belief that the modification was not malicious, but rather an error. Stoll's belief of root cause later changes as additional evidence arises.

Investigators should learn this lesson early: it is important to keep an open mind for new evidence and avoid the dangers of confirmation bias (Heuer, 1999), although confirmation bias is far from the only danger for human analysis of sparse data (Puvathingal and Hantula, 2012). Confirmation bias is one of many human cognitive biases that have been established in psychological study of decision-making (Kahneman et al., 1982). Specifically, confirmation bias is the (unconscious) behaviour of favouring existing beliefs, including unfairly discounting contrary evidence or alternative possibilities to the existing belief. Scientific methodology (see Section 3.4) provides advice for mitigating these biases. But, per the de facto IC standard analytic methodology of Heuer (1999), in the presence of an adversary the analyst needs additional safeguards against cognitive bias the adversary can intentionally abuse. The case studies on building general knowledge in cybersecurity (see Chapter 4) indicate that mechanism discovery and mechanistic explanation can be, and to some extent have unwittingly been, adapted to mitigate biases even in this adversarial case.

Deleting the user account is an intervention in a system that is only partially understood – that is, a scientific experiment. Therefore the belief that the error will not recur is conditioned on the assumption the analyst's present model is correct. This set-up provides a natural test. If the error recurs, the model is likely incomplete or incorrect, so the analyst iterates over the evidence collection and model update process again. This happens to Stoll repeatedly. Eventually, after many iterations, it leads to the model of a Soviet spy using the LBNL computers to attack US military computers. This discussion covered only the first cycle in the course of the example. To avoid belabouring the point, I will not map these iterations but rather switch to a second example.

5.3.2 *Example 2: Airport Security*

I would like a widely applicable model of incident analysis. Therefore, the next example is a physical-world investigation example with a different goal than Stoll (1989). This section examines the use of game theory to provide decision-support for security patrols at Los Angeles International Airport (LAX) airport using the Assistant for Randomized Monitoring Over Routes (ARMOR) software agent. This example is helpful because it explicitly has a model of behaviour, unlike Stoll. On the other hand, security patrols at LAX are a very different kind of analysis than determining the root cause of an accounting error.

But ARMOR represents a kind of ongoing incident analysis. The divergent aspects of this case study are helpful because they help establish the extent of generalizability of my representation of CSIR. By selecting rather widely varied examples, I aim to establish a wide space of applicability and demonstrate significant commonality between information and physical security.

My main source for LAX security are the papers collected by Tambe (2011). This section will focus on the chapters that provide motivation (Southers, 2011), development (Pita et al., 2011), and evaluation (Taylor et al., 2011) of the game-theoretic security solution at LAX.

Southers (2011) motivates securing LAX with a combination of shifting global risk factors and past targeting of LAX. Because of increasingly strict security measures post-9/11, the author predicts increased threat at pre-security-checkpoint areas. LAX is the largest point-of-origin and point-of-destination airport in the world (Southers, 2011, p. 35) and it spends roughly 23% of its operating budget on police resources (Southers, 2011, p. 41). Furthermore, the terminal's pre-screening area suffered six terrorist attacks between 2001 and 2011, the most of any US-airport during this time frame (Southers, 2011, p. 38).

Pita et al. (2011) provide the details of ARMOR's implementation at LAX. The heart of the implementation is a Bayesian Stackleberg game, a type of game in which the first player makes a decision to commit to a strategy and the second player decides knowing the first player's choice (Pita et al., 2011, p. 72). The mathematical model here is clear. The link from math to the conceptual model of the phenomenon, terrorist attacks at LAX terminals, is also clear. However, one may reasonably question the sense in which this

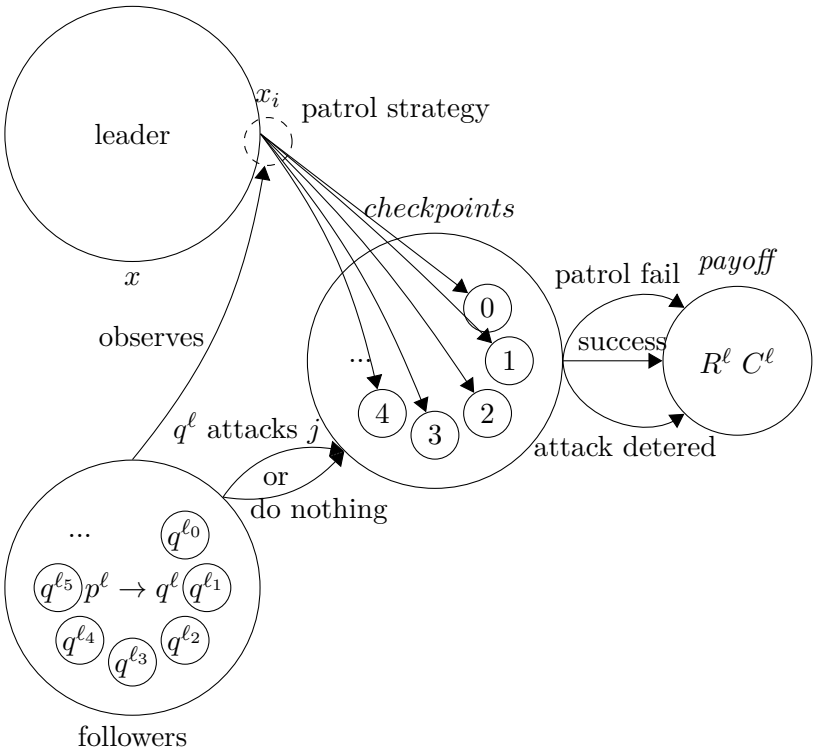

Figure 5.5: Translation of ARMOR (Pita et al., 2011) to a mechanistic diagram. The diagram does not have the precision of the mathematical representation, but it captures the qualitative flow of the patrol mechanisms defined by ARMOR. The leader selects a (mixed) patrol strategy x_i of the checkpoints, then nature selects which adversary q^ℓ the leader will face using p^ℓ. The adversary q^ℓ observes the leader's strategy x_i and then makes her decision of which checkpoint to attack, if at all. Utility payouts are calculated from the matrices R^ℓ and C^ℓ to the leader and follower, respectively, based on which of the three possible results obtained, where patrol fail means the adversary successfully executed an attack, patrol success means the adversary was caught in an attack attempt, and attack deterred means the adversary did not attack.

is incident analysis, or at least in which the example bears on day-to-day computer network operations.

The conception of incident analysis for LAX physical security is that it is the multi-year ongoing patrolling and searching of the LAX premises. ARMOR informs scheduling decisions within the active incident analysis for where to most profitably search for evidence; it is only one of many resources available in the multi-layered approach to LAX defence (Southers, 2011, p. 37). Not all layers are active, for example hardened cockpit doors are a static defence strategy. The whole process of analysis and investigation which ARMOR informs includes the officers actually at the checkpoints, their routine for screen-

ing vehicles, response plans when contraband is discovered, etc. Thus while the ARMOR model is a key decision-support resource, it is not the whole of the investigation.

Let's explicitly note the instantiations of the challenges listed in Chapter 4 to verify this example is applicable to incident analysis. The Stackleberg game within ARMOR explicitly assumes the adversary will observe defender strategy choice and adjust. The justified secrecy among friends is implicit in the intelligence sharing arrangement; LAX police receive an unclassified subset of their Federal Bureau of Investigation (FBI) liaison's classified intelligence. Since it is a physical security case, the changeability of software is not present in the same way. The ARMOR case does share some other features with cybersecurity. For example, both deal with detection of rare events.[4] ARMOR is also dealing with the challenge familiar to incident management of being a cost center for its parent organization where the desired outcome is that nothing happens. LAX spends 23% of its operating budget on security.

In providing background, Southers (2011) demonstrates two important features of models. First, models are situated in the world, and the facts of the world determine how well a model fits. A poor understanding of the facts of the world usually leads to an ineffective model, and thus an ineffective incident analysis. Secondly, the data ingested by the analyst when forming her model may actually be reports of other analyst's models. That is, the process is recursive. Concretely, Southers (2011, p. 47) clearly situates the ARMOR model as an appropriate response to a RAND Corporation model of most dangerous threats to LAX, joint vulnerability assessments by the US Transport Security Administration (TSA) and FBI, the intelligence community's model of the terrorist planning cycle, and the police divisions report on its available material resources. Incident analysis rarely occur in a vacuum, unrelated to any other incidents.

Taking the relevant scope as the large-scale patrolling of LAX contrasts nicely with the small-scale analysis example provided by the initial 10 or so pages of Stoll (1989). Some elements, such as the process, resources, model, decisions, and value of information are more obvious in this larger scope (Pasman, 2011); for precise definitions of how these terms interact see Caulfield and Pym (2015b). For example, the purpose of randomizing patrol strategies is explicitly to reduce the predictively-valuable information available to adversar-

4 For the canonical description of this problem in NIDPS systems, see Axelsson (2000). LAX experienced six attacks in ten years with 165,000 people passing through the terminal daily (Southers, 2011, p. 47).

ies (Pita et al., 2011, p. 69). This fact highlights the strategic and adversarial nature of incident analysis and the value of information to both parties; this nature is less clear in smaller investigations which consist of one adversary action, then analyst response.

It may not appear that the game theory model used at LAX fits with mechanistic explanation. The game theory literature does not use the word mechanism or mechanistic, but that is more a difference of terms than of substance. Recall from Chapter 4 that "a mechanism for a phenomenon consists of entities (or parts) whose activities and interactions are organized so as to be responsible for the phenomenon" (Glennan and Illari, 2017).[5] In the LAX game, the phenomenon is optimal patrol of the airport, the entities are the two sets of agents, the activities are the actions the agents can select, and "organized so as to be responsible for" is the mathematical structure of the Stackleberg game and its rules. I do not imply that either discipline should change its vocabulary or jargon. However, the disciplines should benefit from more or better trading zones, a term sociology of science borrows from anthropology to refer to the places where useful interchange of ideas can occur between specialized disciplines (Galison, 2010). I utilize mechanistic modelling language in this spirit, to help translate the insights of this analysis, culminated as a game-theoretic model, beyond the discipline of game theory.

Figure 5.5 puts game-theoretic mathematics into a mechanistic form. The value in this exercise is to demonstrate how detailed mathematical models are compatible with qualitative investigative methods and representations. Both are necessary for full explication of a phenomenon; diagrams cannot replace mathematical models. A savvy investigator uses the best-suited language of the various logical, mathematical and conceptual jargons available in order to express the different aspects of her model. Indeed, one intended benefit is to help the analyst realize what parts of her model would most benefit from additional (mathematical) detail. The devil is in the details for elaborating such mathematical models; I have selected ARMOR as an example to demonstrate that it is nonetheless possible.

Finally, Taylor et al. (2011) report on the value of the ARMOR model relative to the goals of the LAX security forces. The primary goal is feedback into the effectiveness of the model (Taylor et al., 2011, p. 274) and with the explicit secondary goal of providing data for evidence-based policy decisions (Taylor et al., 2011, p. 282). Taylor et al. divide the security force's goals into direct

5 Compare Craver (2007) and Illari and Williamson (2012).

benefits and indirect benefits, such that direct benefits are practicably measurable whereas indirect benefits are generally not. These goals are for security decision-support systems, and so should overlap with the my goals for logical tools. The direct goals suggested are: "reduced security costs; attacks prevented or mitigated during execution; increased numbers of attackers caught; or reduced damage from successful attacks" (Taylor et al., 2011, p. 272). The suggested indirect goals are: "attacks prevented through deterrence; increased attacker planning time; increased requirements for a successful attack; improved public perceptions of security; or improved (qualitative) assessments of security by experts" (Taylor et al., 2011, p. 272). These goals apply to cyber-defence generally as well.

5.4 CONCLUSIONS

The intrusion kill chain is a model of attacks. As such, CSIR analysts use it to understand the steps that have led to an incident. This chapter has demonstrated how such attack models can be understood as mechaistic models, or, more accurately, multi-field clusters of mechanism schema. This view provides analysts access to the heuristics for hypothesis generation discussed in Chapter 4. I have also demonstrated how the structure of general mechanistic knowledge provides heuristics improving models of attacks by filling in gaps.

Chapter 4 located the kill chain in a mechanism hierarchy related to botnets and malware. In this chapter, I have located the kill chain related to specific exploitation steps and within campaigns of attacks. Chapter 7 will further enrich this web by located the kill chain model relative to NIDPS alerts on the delivery phase of the kill chain and a natural method of composing the results of multiple kill chain instances. Thus, through the repeated location of clusters of mechanisms, we are building up general knowledge for incident analysts.

Cybersecurity is now regarded by respected practitioners like Geer (2014) as outside the comprehension of any single expert. The cybersecurity community needs more structured communication between experts for the field to progress with the necessary speed and accuracy to continue to be effective. Mechanistic models are one way to provide this richer and more clear language to expand on the models practitioners already use. Chapter 6 provides the necessary foundation to move forward in Chapter 7 with another way to provide such

precise and clear language that is needed by the CSIR community, complementary to and supported by the mechanistic example from this chapter.

6

SEPARATION LOGIC AS A CASE STUDY[1]

This chapter focuses on logic as a technology by reflecting on the achievements in verification of computer programs using logics as tools. Specifically, I will trace the history of Separation Logic, a development within theoretical computer science firmly established by Ishtiaq and O'Hearn (2001), O'Hearn and Pym (1999) and Reynolds (2002). The result is a case study with multiple uses. The paper on which this chapter is based argued for several uses, to both logicians and philosophers. However, within this thesis, the primary purpose of a historical case study on Separation Logic is to extract the features of a formal system that make it work, in the sense that it is deployable and usable in industrial ICT settings. In Chapter 7, I will seek to emulate and build with these features. Additionally, this chapter introduces the formal definitions of Separation Logic on which Chapter 7 will build.

6.1 INTRODUCTION TO SEPARATION LOGIC

Separation Logic adds a connective to standard logic called 'and, separately' that solves a problem of reasoning about the resources a computer program will need when it executes. This chapter will lay out what makes reasoning about computer resources hard and explain Separation Logic's special arrangement of properties that enable its effective use in program verification problems.[2] This chapter focuses on the current form of Separation Logic; for

1 This chapter is based on joint work, the paper: David Pym et al. (2018). 'Why separation logic works'. In: *Philosophy & Technology*. DOI: 10.1007/s13347-018-0312-8.

2 The sub-discipline of logic and verification of computer programs has flourished within wider computer science since at latest 1970 with the activity surrounding Floyd–Hoare logic (Apt, 1981). The first academic conference dedicated to studying programming languages, including the verification of languages using logic as a tool, took place in 1973 (Principles of of Programming Languages, or 'POPL', http://www.sigplan.org/Conferences/POPL/) and a dedicated journal appeared in 1979 (ACM Transactions on Programming Languages and Systems, or 'TOPLAS', http://toplas.acm.org). Independent publication venues help mark where an academic community forges its own identity, characteristic problems, and norms. Program verification may be some mix of computer science and logic, but it is also independent. I will focus on the technical aspects of program verification. For a wider sociological view of how program verification fits into the history of mechanizing proofs, see MacKenzie (2004).

an account of its development history, see Calcagno et al. (2015a) and O'Hearn (2015).

There are two main reasons why Separation Logic works. First, it merges with the scientific-engineering model the programmer uses to understand and build software. This feature mirrors in some ways the dual abstract-physical nature of computer code in general (Turner and Angius, 2017). Close interaction between scientific models (the topic of prior chapters) and logical models will be an important feature of the logic of incident analysis in Chapter 7. Secondly, the proof theory developed to check software using Separation Logic is based on rules for scaling the reasoning task, and has has been deployed in numerous tools for formal reasoning about programs. These tools range from 'proof assistants' that are used by humans to automatic software analysis tools that provide pragmatic and usable advice to software engineers during development.

Program verification holds growing importance as a problem. Assigning the appropriate resources to a computer program is important for efficiency, accuracy, reliability, and security. The 2016 Gödel prize, awarded to Brookes and O'Hearn for resolving these resource management problems with Concurrent Separation Logic, puts the problem in context:

> "For the last thirty years experts have regarded pointer manipulation as an unsolved challenge for program verification. . . "

Program verification as a discipline is focused by the practical challenges of making computer software function reliably, such as by assuring the program claims, uses, and releases the correct resources at the correct times. Resource allocation decisions are hard both in principle and in practice. In principle, recall from Section 4.1 that given computer code (i.e., software), one cannot determine *a priori* when or if the program will halt (Turing, 1936). In practice, for small programs this impossibility-in-principle is overcome by estimates based on past experience with similar programs. Modern software projects at companies like Microsoft, Facebook, and Google have many millions of lines of code, written by thousands of people. With such a fragile, distributed, and complex system in which changing fives lines of code out of a million can drastically change behaviour, estimates based on experience are inadequate. Therefore, although software is a human artefact, one does not in general know surely what any given computer code will do when executed. To overcome these challenges, companies including Spotify and Facebook use Separation

Logic to verify their mobile app software (Calcagno et al., 2015b) using a tool called 'Infer'. Separation Logic is not limited to one use; extensions of it are used, for example, to verify operating-system scheduling (Xu et al., 2016), a crash-tolerant file system (Chen et al., 2015), and an open-source cryptographic operation (Appel, 2015). The scope of this case study is the development of Separation Logic from fundamentals in logic through to real-world application.

The resource about which Separation Logic can best reason is computer memory, specifically Random Access Memory (RAM) (hereafter, simply 'memory'). Appropriately resourcing memory for a computer program is an important task within computer science. Memory errors are not easily handled during program execution, and adversaries can use errors to remotely take control of computers using open-source attacks.

Separation Logic was developed at a crossroads of two problems, a logical–theoretical issue of reasoning about resources generally, and a technological–computer-science problem of preventing errors in memory usage. This confluence led to a special constellation of properties. Under the right conditions, interpreted the right way, the logical model can be at once a logic model and a scientific model. To be a genuine joint logic-scientific model is to take on the features and norms for use of models both in a logic and in a scientific discipline.[3] In practice, not just the model but also the proof theory adapts to satisfice the needs of the engineering problem at hand ('satisfice' as per Simon (1996)). I will not propose anything surprising about the features of models. The surprise has come forward in powerful results for reasoning about computer programs once Separation Logic was built with the constellation of properties that genuinely make for features of both types of model.

Separation Logic is one example case out of several projects in computer science that exhibit this simultaneity of logic and scientific models. In the 1970s, Floyd–Hoare logic merged the engineer's notion of program execution into first-order logic. However, Hoare logic by itself does not adequately adapt to the task of efficiently reasoning about pointers and mutable data; proofs are not practically tractable (Bornat, 2000). A primary feature of an engineering model is to be satisfactorily useful, not merely to include considerations from

3 In the journal paper, we used 'engineering' here instead of 'scientific'. But given my arguments that computer security is a kind of science in Chapter 3, as well as my prior work arguing computer science generally conducts experiments (Hatleback and Spring, 2014), I will treat scientific and engineering models as interchangeable for the purposes of this discussion.

the engineered world. This further step is more rare. It requires adapting the logic model, model structure, and proof theory to suit the engineering model or task and to test that suitability empirically. For example, temporal logic, as used in tools built by Amazon to manage its infrastructure, also seems to have achieved this special merging of logic and engineering models (Newcombe et al., 2015). This chapter will survey how Separation Logic has adapted its features to the task of verifying a program's use of memory. The logic is adapted through its syntax and proof theory as well as its model structure – the engineering model does not merely supply semantics to a logical syntax.

As Chapter 3 touched on, there is a historical connection between scientific laws and logic models. The logical positivists in the mid-20th century held that a scientific theory was a set of sentences in first order logic. The physical world and its laws of nature are interpreted as a model of true scientific theories (Frigg and Hartmann, 2012, §1.3). Logical positivism and this usage of model have fallen out of favour. Practitioners use scientific or engineering models to represent phenomena or data (Frigg and Hartmann, 2012, §1). To say here that Separation Logic merges logic and engineering models, I do not mean a by-definition (*de dicto*) merging reminiscent of logical positivism. I mean a logic built and empirically tested to usefully reason about a phenomenon.

Although the computer scientists building tools such as Separation Logic have not explicitly intended it, the result is logic as a technology that is compatible with the idea of general knowledge as clusters of related mechanisms as put forward in Chapter 4. Each logic model is useful within its specific, empirically-tested domain. If one switches to a different domain of computer science problems, a different model likely applies. But these logics naturally form related clusters of how practitioners use and understand them, such as Separation Logic.

This case study focuses on practical questions of the efficient use of models for reliable reasoning, not ontological questions of what processes are computation. The important part of Infer is that it predicts what a salient complex object will do, not questions of whether physical objects compute (Piccinini, 2007) or whether a computation is a miscomputation or dysfunctional (Floridi et al., 2015). This distinction gives a sense of the extent to which tools like Infer are pragmatic, engineering projects. Yet, testing for mundane properties like stability still requires a novel development of a logical technology. As Swoyer might say, Separation Logic represents computer programs 'in a medium that facilitates inference' (Swoyer, 1991) about them.

Another lens of interpretation for the case of Separation Logic is that of scientific representation. Suárez (2010) distinguishes between analytical and practical inquiries into the nature of representation. This chapter considers Separation Logic tools to be a case study in pragmatic representation. The case has value within the analytic–practical distinction because within these verification tools one logical model (Separation Logic) is used as a pragmatic representation of another system (computer code). Tools such as Infer have both representational force and inferential capacities, as defined by Suárez (2010, p. 97). This case study will describe the details of Separation Logic that give it these two properties, thus making it both a model in the scientific sense and in the logical sense. The following sections will support the claim that this merging is a vital feature of what makes Separation Logic successful, and is worth emulating. A logic model that is also a scientific model poses an interesting case for analytical inquiry in future work. However, this chapter will focus on the practical description of how programmers use Separation Logic to solve problems using this form of model building.

The two categories of Separation Logic's properties this chapter will elaborate are its semantics, which has a clear interpretation in the mathematical model of computer memory, and its proof theory for composing reasoning about resources, which is both automatable and modular so as to scale to real-world problems. Both of these features are related to the properties of the connective 'and, separately', represented in symbols as $*$. The primary insight is to learn to recognize situations in which a logic model, by coincidence or by design, usefully overlaps with a model of a practical problem. When this coincidence is recognized and pursued, the development of both the logic and the practical solution benefit. Pursuing this confluence will be my goal in Chapter 7.

Separation Logic mirrors the computers in the physical world in a deep and important way that first-order logic does not. Both atoms of Separation Logic and computer parts are composable in a natural way. In some sense, the other beneficial properties of Separation Logic derive from pursuing and refining the benefits of a logical primitive ($*$) that directly and cleanly captures the compositionality of resources in the physical world.

Section 6.2 describes the context of the application, including details about what challenges make program verification of allocation of computer memory resources a hard and important problem to solve. Section 6.3 introduces the properties of Separation Logic that meet the relevant challenges. Section 6.4

surveys the logical properties (semantics, syntax, etc.) of Separation Logic, focusing on its novel connective 'and, separately' ($*$). Section 6.5 describes how the Frame Rule and automated abduction make Separation Logic a solution for reasoning about computer memory resources that is practical for large software development firms to deploy. Section 6.6 concludes the chapter by extracting advice from this case study for Chapter 7: that a simultaneous logic-engineering model is a good start, but to succeed the logic model's structure must be exploited to give some measurable benefit.

6.2 SOLVING A HARD PROBLEM

Memory management is challenging, and errors potentially lead to unstable behaviour, resource exhaustion, or security threats. Management of the computer's memory falls directly to the programmer in languages like C. This section will introduce the importance of C-like languages and the task of memory management. This description amounts to the programmer's model of what the computer does, a model very much like any other scientific model. The main features of the model are pointers and memory locations, to which this section will give a minimal introduction. This section will motivate why problems in memory management matter, and why some other methods of finding such problems are unsatisfactory. These gaps help understand some reasons why Separation Logic works.

The C programming language was first developed in 1972, to implement the UNIX operating system. Every major computer operating system is now written using C. C and languages derived from it — such as Java, C++, C#, and Python — may account for as much as half the computer code written yearly.[4] The impact of C is perhaps even more than this ratio indicates. As part of the operating system, C code is in the critical path for almost any computer task.

As introduced above, the programmer cannot in general know what her program will do once written. The popularity of C has to do with its expressivity, speed, and effectiveness. Unfortunately, its benefits do not include easy identification, or tolerance, of errors. The computer can check that it understands the syntax the programmer wrote, which catches some errors. Beyond this,

4 There is no precise way to count code written; however, analysis of publicly available web sites indicate C and descendant languages account for about half. See http://www.tiobe.com/tiobe_index.

one cannot readily predict errors[5] that will occur during execution of the program, known as run-time errors. There are various technical types of run-time errors, but for present purposes let's partition them into 'annoying' and 'catastrophic'. Annoying errors lead to incorrect results, for example dividing by zero. The program can catch annoying errors and recover. Catastrophic errors lead to the program fatally failing, such as by crashing or exhausting available resources. Recovering intermediate progress is not generally possible after a fatal failure. If the program is able to exhaust all system resources, such an error may bring down the rest of the computer system as well. Memory management errors are one common type of catastrophic run-time error.

Memory management is not the only task for which Separation Logic provides a suitable logical substrate. Task scheduling within an operating system is another source of catastrophic run-time errors (Xu et al., 2016). Section 6.5.2 touches on the use of Separation Logic to address this second example; however, the memory management context is the primary example. To see what makes memory management errors catastrophic, consider the basics of computer memory.

The programmer's model of memory management abstracts away from the hardware. Computer memory is a slab of silicon electronics. Conventionally, the smallest elements are binary digits, or bits, interpreted as 1 or 0 based on whether the local voltage is high or low. The hardware is designed such that any location can be read or written equally quickly (thus 'random access' memory). Eight bits are usually grouped into a byte for the basic unit of memory with which humans interact. The programmer thus models the memory as a list of individual bytes, like houses on a very long street. In 64-bit operating systems, these bytes in memory are given an address from one to $2^{64} - 1$.

Strictly, a pointer is the address of some object in memory. A pointer-variable (usually, unhelpfully, just 'pointer') is a kind of variable that contains an address; in particular, the address where some other variable's value is stored (Kernighan and Ritchie, 1988, p. 93). Pointers are well-known in computer science to be both 'extremely powerful' and 'extremely dangerous' (Ishtiaq and O'Hearn, 2001, p. 1). Pointers are powerful because they allow quick calculation over items in memory. Figure 6.1 demonstrates pointer basics. Each variable is represented by a square. Its name is above the square;

5 A programmer might take different dispositions towards errors, as summarized by Petricek (2017).

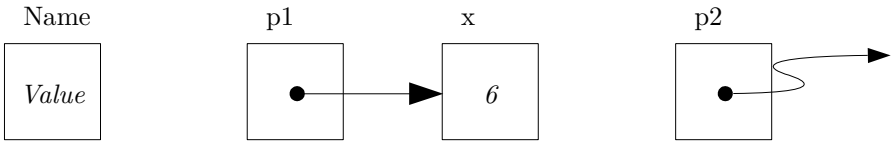

Figure 6.1: Anatomy of a pointer. Pointer $p1$ points to the variable x, whose value is 6. The name for pointer $p2$ is reserved, but it does not point anywhere; such a pointer has a special value called `NULL`.

contents are inside. If the content is a pointer, it is represented as a arrow to its target. A pointer may be declared, to reserve its name, without a target, which is represented by a wavy arrow without a target. A pointer with no target has the special value `NULL`, and is called a null pointer. One common memory management error which one can find with Separation Logic is when a program attempts to use a null pointer in a situation that requires a pointer with a valid value.

A more subtle pointer error is to lose track of items in memory. An item in memory is only accessible if there is a pointer to it. Garbage is the official term for memory which is allocated but not accessible. If memory garbage is not cleaned up by the programmer, memory eventually gets clogged by allocated but inaccessible data. This slow exhaustion of reserved memory by failure to clean up is called a memory leak. Figure 6.2 demonstrates a method by which a memory leak may occur. The term for cleaning up memory is to free it; that is, release reservation on its use. Unfortunately, it is not so simple as to just ensure the program frees all memory eventually. Errors when freeing memory also lead to dangerous behaviour. If the program maintains and uses a pointer to a memory location after freeing the memory, the location could have been used by another program to store different data.

A program with a memory management error may behave erratically. Whether this behaviour is catastrophic in a human sense depends on the importance of the program. A word processor with a memory leak which means it cannot run for more than four hours is probably fine. If the software is for a continuously-operating air traffic control radar facility, it is more severe.[6]

Memory management errors are also security problems. As Chapter 2 described, CWE tracks canonical types of flaws that lead to security vulnerabilities. The entries for null pointer exceptions, resource leaks, and memory

6 A 2015 Federal Aviation Administration (FAA) press release description of such a failure suggests that the cause was a memory leak (FAA, 2015).

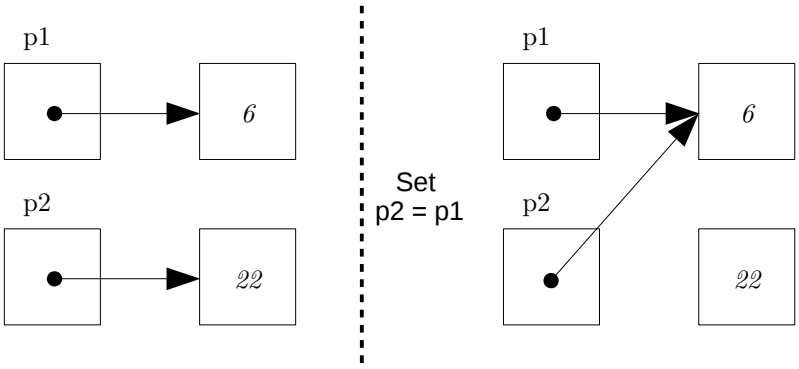

Figure 6.2: One example error involving pointers. The data element 22 is no longer accessible, because there is no pointer to it. That memory space has 'leaked' and cannot be freed (released back to the computer) or accessed. Memory leaks eventually lead to resource exhaustion.

leaks (which are CWE-476, CWE-402, and CWE-401, respectively) provide a long list of software that has been vulnerable to a hostile takeover due to these memory management errors (MITRE, 2015). Again, the amount of harm depends on the importance of the victimized computer. However, criminals can use and resell the electricity and network connection of any computer, to either hide more sinister attacks or rent as infrastructure for less technologically capable criminals (Sood and Enbody, 2013). Thus, it is important to prevent vulnerabilities such as memory management errors in all computers.

These are two major reasons memory management errors are problematic. They cause instability and make a program crash, which is bad for functionality and usability. They also frequently lead to exploitable security vulnerabilities.

I will now switch from a survey of the challenges to a survey of the solutions. There are two classes of methods to find flaws in computer software: static and dynamic. In static analysis, one analyses symbolic and structural features but does not run the code.In dynamic analysis, one runs the code and measures what happens. With Separation Logic, one finds errors statically.

Success in program verification can be measured in at least four ways: reduced software flaws, accuracy of findings, speed of analysis, or reduced human time to fix software flaws. In practice, measuring how many flaws a technique finds is easy to do but hard to interpret. The total number of flaws remains unknown, because of Turing's halting result discussed in Section 6.1. Other, complementary explanations for why program verification might fail include difficulty in communicating how one's understanding of the program

should change based on the verification (De Millo et al., 1979). A more fundamental historical contention by Fetzer (1988) is that abstract formalisms such as proofs are categorically about different types of things than physical computer operations, and so the former cannot, strictly speaking, deductively demonstrate properties of the latter. It shall become clear that Separation Logic works because it supports reasoning about physical systems using formal systems specifically by designing the two types of systems with overlap in important and reliable ways.

Regardless of exactly why, it is evident that one cannot know for certain how many flaws remain undetected after a verification attempt. Therefore, calculating accuracy or relative reduction of flaws is impracticable. The problem is that finding 99 flaws could equally well be 99% effective or 2% effective. A more interpretable measure for the software industry is the rate at which found flaws are able to be fixed. This measure relates to analysis speed, because humans fix software better if given fast feedback. These desirable engineering outcomes suggest static analysis.

To make progress with static analysis, one must take a defined subset of the general problem of all software flaws. To target memory in particular, elements of Separation Logic faithfully incorporate the engineer's model of how the computer manages memory; that is, pointers as previously described. Within verification software such as Infer, the logic model and the engineering model will coincide. This confluence overcomes several of the challenges described in this section to more effectively prevent memory management errors.

Pure technical logical matters are not sufficient for success. Brooks' essay (Brooks Jr, 1995) is an example starting point for general software engineering considerations. For a program analysis tool such as Infer, integration with an organization's programming culture and process is significant work; see O'Hearn (2015). For logic to impact software engineering practice of artefacts such as operating systems or cryptography software, in which proofs of properties closer to functional correctness are important, effective integration with a powerful proof assistant is critical (Appel et al., 2014). While these contextual questions are important, this chapter will focus on the features of the logic and its model that contribute to its success, not on the additional contextual factors which are nonetheless important.

This section introduced pointer management in computer memory. Pointer mismanagement can lead to serious stability and security flaws. Such flaws are hard to find dynamically during run-time. However, finding such flaws

statically, based on the program's source code, has historically been too hard to do well enough to be useful. The following sections describe how Separation Logic succeeds at this task. Section 6.3 introduces all the properties of the logic that contribute to success. Separation Logic's properties that make it useful can be tackled in two broad categories: semantics (Section 6.4) and proof theory (Section 6.5). I will argue that the logic undergoes a holistic adaptation to meet the practicalities of the engineering task.

6.3 WHY SEPARATION LOGIC WORKS

Separation Logic works for solving this problem of reasoning about memory allocation because of a group of features:

- A useful engineering model of computer memory;
- A logical model and language that are grounded, respectively, in an interpretation of and semantics for exhaustible resources;
- The productive overlap of these two (types of) models;
- The use of the connective $*$ for 'and, separately', from the logic of bunched implications (BI), to facilitate the formulationof a 'Frame Rule' to support compositional local reasoning; and
- Scalable pre- and post-conditions.

The first three elements are modelling choices that provide a powerful capacity for prediction of computer memory usage which is not otherwise available. The latter two elements support a scalable algorithm for calculating and proving these predictions for a given computer program.

BI can be interpreted as a logic of exhaustible resources (Galmiche et al., 2005). For example, if one has 10 coins, it is certainly true that one has the capacity to buy a tea that costs 4 coins. It is also true that one has the capacity to buy a lemonade that costs 5 coins and the capacity to buy a beer that costs 7 coins. It is not, however, true that one has the capacity to buy both a lemonade and, separately, a beer – 12 being more than 10. 10 coins does provide the capacity to buy both a tea and and a lemonade, or two lemonades. The resource-interpretation of BI's semantics provides a precise interpretation of formal logical statements of such cases. More specifically, BI makes a distinction between the usual logical 'sharing' conjunction, for example,

> '10 coins is enough for a lemonade and is enough for a beer'

which is true only if the resource of 10 coins can be shared or 'reused' by the two parts of the statement, and the separating conjunction, for example,

'10 coins is enough for a tea and, separately, a lemonade'

where the resource of 10 coins must be divided, or separated, into those required for each part of the statement.

Computer memory, like money, is an example of an exhaustible resource. Though a computer is, basically, electricity and magnetism in silicon, computer programmers[7] do not write software as if they were individually manipulating millions of tiny magnets. Like most engineers, a programmer works with a model of what she is building. To the programmer, the model of computer's memory is provided by the stack and heap. In a violently simplified analogy, the stack is what you're doing, and the heap is what you're working on. These metaphorical names are evocative of their function. The stack is an ordered array of data elements, and the computer can only put elements on the top, and take them off the top. This structure ensures an orderliness and efficiency good for sequential recursive instructions but not good for big chunks of data. In the heap elements can be accessed in any order but only if the program's stack has a pointer to the information's location in memory. Note that the stack maps variables into values, whereas the heap maps addresses into values (Reynolds, 2002).

Though the programmer's model abstracts away from it, location here has a physical interpretation. Computer memory is an apparatus with a numerical address for each microscopic bit in its vast silicon plane. Like the structural engineer who has mathematical equations that inform her choices for bridge design, the programmer uses the model of the stack and heap to inform software development. In both cases, the engineer's model's prediction is not perfect, and the bridge or the program could collapse despite best efforts.

One success of Separation Logic is to merge the logic-model and the engineering-model. The stack and the heap have formal representations in Separation Logic, with the heap as a resource. A programmer's models of memory can be expressed as sentences within Separation Logic without unacceptable loss of applicability to the real world. Sentences in Separation Logic have deductible consequences, and can be proved. In the computing context, this amounts to proving properties of future behaviour of possible program

7 Instead of 'programmer', one may find '(software) developer', 'coder', or 'software engineer'. These terms have differing connotations across various communities, which are not relevant here. I just mean anyone who writes software.

executions. Such proofs enhance the engineering-model of the program directly. If the logic deduces a section of the program code will make a memory-usage error, the code can be tested empirically to verify the error and gather information about the mechanism by which the error is committed. These logical and empirical results update the programmer's model of the software, and she fixes the code accordingly. Although the engineering-model is of a built system, the programmer updates this model similarly to how natural scientists might update their model of a phenomenon after using a tool to observe it. In this case, logic is a primary tool, as opposed to telescopes or microscopes in other sciences.

An engineer's model commonly merges mathematical modelling with some subject-matter expertise to make predictions. For example, a structural engineer can use mathematical models to predict when stress on a bridge element will exceed its shear strength because of established accurate physical measurement of each material's properties, gravity, etc. But computers are devices made up of logic more-so than metal. However, just as with bridges, when building a computer and software, one does not know everything that the computer will do just because an engineer designed it. There are interactions with the world that are unpredictable. Logic is one of the subject-matter expertise areas programmers use, as a structural engineer uses materials science. Also similar to other engineering or science disciplines, using the correct logic is important. The correct logic for a programming task is determined empirically; in our[8] experience with Separation Logic, the process seems similar to the usual scientific model-building.

There are four distinct logical elements which have made separation logic successful: the connective $*$, Hoare triples, the Frame Rule, and the specification of automatable abduction rules.

6.3.1 *The separating conjunction*

The connective 'and, separately' ($*$) is related to the familiar connective for conjunction ($\wedge$). One writes $\phi \wedge \psi$ for the situation $w \models \phi \wedge \psi$ (read $w \models \ldots$ as 'the world w "supports" or "satisfies" ...') if and only if (abbreviated iff) $w \models \phi$ and $w \models \psi$. One can make a different conjunction, 'and, separately' to capture the resource interpretation for reasoning about exhaustible resources.

8 This attestation is David's and Peter's experience, filtered through our discussions

This construction requires a little more knowledge about the world w that supports $\phi * \psi$ to say when $w \models \phi * \psi$. The world needs to be able to be broken up into disjoint parts, which is usually represented as $w_1 \cdot w_2 = w$ to say w_1 composed with w_2 is w. Given this decomposition, then $w \models \phi * \psi$ iff there are $w_1 \cdot w_2 = w$ such that $w_1 \models \phi$ and $w_2 \models \psi$ (Galmiche et al., 2005).[9]

The difference between $w \models \phi \wedge \psi$ and $w \models \phi * \psi$ is just that aspects of the world can be reused to satisfy conjunction, but not with the separating conjunction. This difference is most obvious in that if $w \models \phi$, then $w \models \phi \wedge \phi$ is always true, but $w \models \phi * \phi$ need not be true, because it may be the case that there is one part of the world that satisfies ϕ ($w_1 \models \phi$), but the rest of the world does not ($w_2 \not\models \phi$). If ϕ is 'I have enough money to buy a drink', then $w \models \phi \wedge \phi$ says nothing new, but $w \models \phi * \phi$ says I have enough money to buy two drinks.

6.3.2 *Hoare triples*

The second technical element that enables the success of Separation Logic is the Frame Rule. Separation Logic builds on Floyd-Hoare logic (Apt, 1981) (henceforth, 'Hoare logic'). Hoare logic developed through the 1970s specifically to reason about the execution of computer programs. The intuition is straightforward: proving the relevant properties of a program C amounts to proving that whenever a certain precondition holds before executing C, a certain postcondition holds afterwards. This statement, known as a Hoare triple, is written formally as

$$\{\phi\}\, C\, \{\psi\},$$

where ϕ is the precondition and ψ is the postcondition. Hoare logic provides various proof rules for manipulating triples. For example, composing two program fragments if the postcondition of the first is the precondition of the second. Such deductions are written as

$$\frac{\{\phi\}\, C_1\, \{\chi\} \quad \{\chi\}\, C_2\, \{\psi\}}{\{\phi\}\, C_1\, ;\, C_2\, \{\psi\}}$$

with the given statements on top and the deduced statement below.

9 This treatment of $*$ is for Boolean BI, where the set of worlds is not ordered. An intuitionistic definition (see Section 6.4.2) requires that $w_1 \cdot w_2 \sqsubseteq w$, where $\sqsubseteq$ is a preorder that is defined on the set of worlds and which satisfies *monotonicity*.

6.3.3 *Frame Rule*

The Frame Rule permits one to combine a Hoare triple with $*$ – for 'and, separately' – to reason about just the local context of a program fragment. This support for local reasoning is critical, supporting *compositional* reasoning about large programs by facilitating their *decomposition* into many smaller programs that can be analysed and verified *independently*. This analysis relies on the compliance of resource semantics with Frege's principle that the meaning of a composite expression be determined by the meanings of its constituent parts.

Logicians write the Frame Rule as

$$\frac{\{\phi\}\, C\, \{\psi\}}{\{\phi * \chi\}\, C\, \{\psi * \chi\}}$$

provided χ does not include any free variables modified by the program C (that is, formally, $\text{Modifies}(C) \cap \text{Free}(\chi) = \emptyset$). The Frame Rule is powerful because it lets analysts ignore context they have not changed. Reasoning locally, as opposed to globally, is vital when analysing large programs. Normally, a program verification technique would have to re-evaluate a whole program if one line changes. When the program has even ten thousand lines of code this is prohibitively inefficient. Industrial scale program analysis must analyse millions of lines of code, and so without the ability to reason locally any approach will fail.

The sorts of preconditions and postconditions that are interesting are directed by the programmer's goals for modelling. Abstracting away from the details of a computer, the precondition may be something like 'there exists an available resource not currently in use' and the postcondition may specify the details of 'nothing bad happened' or 'the program worked'. The Frame Rule is powerful because now one can break the program up into disjoint parts. Having proved $\{\phi\}\, C\, \{\psi\}$ for one parts, one can take χ to be the union of all other pre- and post-conditions of disjoint parts of the program and know $\{\phi * \chi\}\, C\, \{\psi * \chi\}$ without having to re-prove the statement in the new context. Thus, if a million lines of code can be broken up in to ten thousand disjoint fragments, then after a code modification in one it is only needful to prove $\{\phi\}\, C\, \{\psi\}$ for that fragment and not the 9,999 others.

6.3.4 *Automatable abduction*

With these two design choices in place, I'll introduce the last – automatable abduction. Local reasoning is helpful for reasoning about programs at scale, but a human still has to be rather clever and expert to choose exactly the right pre- and post-conditions. There are simply not enough clever experts to do this at scale. Large software development companies have many hundreds of developers who each need their code checked and analysed within a few hours of making complex changes. A human logician might take days to figure out the right conditions for the Hoare triples for each code change. Even if that many experts could be trained, no company is likely to pay for that sort of increased labour cost. Separation Logic works because of abducing potential pre- and post-conditions to test.

Industrial-scale use of logic for proving program properties requires a deployable proof theory. The combination of local reasoning and abduction support a deployable proof theory for Separation Logic. Abduction, as introduced by Peirce (Bergman and Paavola, 2016), is akin to hypothesis generation. Initial implementations of Separation Logic to analyse programs required a human analyst to provide the pre- and post-conditions. However, logicians have been able to automate abduction within the scope of well-defined and well-structured problems (O'Hearn, 2015). One such automatic analysis program, Infer, is published freely as open-source for anyone to use (Calcagno et al., 2015b).

Computer code is not arranged into Hoare triples, so verification tools must create that logical structure as they read and analyse the program. A pre- or post-condition may be established in a segment of the code far distant from where they are needed or checked. One cannot build a table of all possible combinations of legal Hoare triples to solve this problem; the number is astronomically large for even modest programs. Abduction makes this problem manageable by determining at analysis-time what conditions a segment of code might expect. Pre-determining the full extent of such conditions, as is necessary for many other analyses, is prohibitively expensive. The trade off for this benefit is that each abductive hypothesis is not perfect. But each hypothesis can be tested quickly and the results reused. The analysis program can quickly and soundly check each hypothesized pre- and post-condition; because reasoning is local (per the Frame Rule) the result can be stored and reused easily.

Separation logic is useful because it calculates predictions of hard-to-handle computer program execution errors; this is well known. Why separation logic is so effective at this useful task was not deeply questioned. Yet, understanding successful tactics will help reproduce success in different areas of inquiry, of which there are many in logic and computer science. The thesis I will argue for is that the useful predictions are generated by the convergence of two senses of the word model and an effective proof theory. Namely, the logic model designed specifically for the task and the programmatic-engineering model of what the computer and its software actually do. A proof theory is effective only if it meets the engineer's goals, which in this case are timely and explanatory prediction of errors.[10]

This section has introduced the features of Separation Logic that are adapted for it to become an adequate engineering model. Section 6.4 introduces the semantics of Separation Logic; the exposition reinforces the integration of the logic and engineering models. Section 6.4 can be skipped without loss of continuity. Section 6.5 discusses two features that contribute to making the proof theory 'deployable' – local reasoning, as supported by the Frame Rule, and automated abduction. These details will demonstrate directly that the logic model and the engineering model are inescapably and intricately intertwined. This confluence is a source of Separation Logic's success in analysing programs.

6.4 THE SEMANTICS OF SEPARATION LOGIC

The development of Separation Logic has been influenced by many ideas and systems, from program verification developed in the 1970s through to how tech giants produce computer programs today. One formative idea is Hoare's development of assertion programs, with the insight that valid program execution can be interpreted as a logical proof from the preconditions to the postconditions (Apt, 1981). However, the familiar classical logical connectives — $\neg$, $\vee$, $\wedge$, and $\rightarrow$ — and quantifiers — $\exists$ and $\forall$ — did not capture the resource management problems that computer science then found intractable.

10 That results must be timely is straightforward. That the result also provides satisfactory explanation of the error is equally important. Practically, the programmer must receive enough detail to locate and fix the error. Psychologically, programmers are less likely to trust an arcane or unintelligible report than a transparent documentation of the entities and activities responsible for the error. This transparency merges a sense of adequate scientific explanation with the logical community's sense of when a proof is both convincing and elegant.

Linear Logic (Girard, 1987), originally developed as a tool in proof theory, introduced an explicit single-use resource interpretation and a modality (!) to mark resources as being usable as many times as needed. Although linear logic has enjoyed much success, the resource-management problem remained out of its reach.

With the benefit of hindsight, what was necessary was a logic of resources with a structure that was composable and decomposable in a way that mirrors the composability of resources in the physical world. Resources in linear logic are usable once, or infinitely many times. This pattern does not match real-world resources like sandwiches or money. How many hungry people a sandwich satisfies depends on how many parts it can be decomposed into that independently satisfy a hungry person. This number is often more than one but less than 'as many as needed'. Those working to develop Separation Logic into a verification tool wanted a logical structure that mirrors this behaviour. This section will present three historical stages of exposition to reach this structure: first, bunched logic, then the semantics of bunched logic, and finally the semantics for resources in separation logic. Thus, the history will be a technical history, traced through the formal developments. I highlight how these logical tools continued to be developed until the logical model and tools met the needs of program verification practice.

6.4.1 *Bunched Logic*

O'Hearn and Pym (1999) introduced BI. In its initial form, BI can be understood as freely combining the intuitionistic propositional connectives (additive connectives in BI) with the multiplicative fragment of intuitionistic linear logic (multiplicative connectives in BI).

The idea of bunching is used to formulate natural deduction and sequent calculus proof systems for BI. Bunching is an older idea from relevant logic; see, for example, Dunn and Restall (2002) and Read (1988). The key point is that proof-theoretic contexts are constructed using two operations, one corresponding to the additive conjunction, $\wedge$, and one corresponding to the multiplicative conjunction, $*$.

To see how this works, consider the natural deduction rules for introducing the additive and multiplicative conjunctions, $\wedge$ and $*$, respectively. If $\Gamma \vdash \phi$ is read as 'ϕ is provable from assumptions Γ', then the introduction rules are:

$$\frac{\Gamma \vdash \phi \qquad \Delta \vdash \psi}{\Gamma; \Delta \vdash \phi \wedge \psi} \wedge \mathrm{I} \quad \text{and} \quad \frac{\Gamma \vdash \phi \qquad \Delta \vdash \psi}{\Gamma, \Delta \vdash \phi * \psi} * \mathrm{I}.$$

Notice that the $\wedge$I rule combines the contexts Γ and Δ using semi-colon, corresponding to $\wedge$, whereas the $*$I rule combines them using the comma. Semi-colon admits the contraction and weakening rules,

$$\frac{\Theta(\Gamma\,;\,\Gamma) \vdash \phi}{\Theta(\Gamma) \vdash \phi}\,\mathrm{C} \quad \text{and} \quad \frac{\Theta(\Gamma) \vdash \phi}{\Theta(\Gamma\,;\,\Delta) \vdash \phi}\,\mathrm{W},$$

respectively, whereas the comma does not. The form of these rules draws attention to a key point; bunches are trees, with leaves labelled by propositions and internal vertices labelled with ' ; ' and ' , '.

Contraction is what permits the simple additive form of the $\wedge$I rule. What is meant when one says the context Γ is shared between the two components of the conjunction is just $\Delta = \Gamma$ in $\wedge$I.

6.4.2 *The Semantics of Bunched Logic*

The semantics of BI can be seen as being based on the notion of resource. Intuitively, the builders of Separation Logic sought to bridge a scientific-engineering concept of resources with a logical model that allows reasoning about resources. An English description of the properties of resources was translated to a mathematical description of these properties in the form of a monoid, which was in turn translated to a structure that permits interpretation of logical sentences and the determination of logical satisfaction (i.e., a model). Specifically, two central properties of an engineering model of resource, bearing in mind the examples of interest as discussed above, are as follows.

- Given two elements of a given type of resource, it should be possible, subject to an observation spelled out below, to combine them to form a new element of that type of resource. In the example of coins in Section 6.3, combination was the addition of numbers of coins.
- Given two elements of a given type of resource, it should be possible to compare them. In the coin example from Section 6.3, the usual mathematical less-than relationship is a meaningful comparison between the

number of coins available (10) with the number required to buy *both* a lemonade and a beer (12).

Mathematically, these 'axioms' for resource were captured conveniently by requiring that a given type of resource carry the structure of a preordered partial commutative monoid.[11] That is, a (type of) resource $\mathbf{R}$ is given as

$$\mathbf{R} = (R, \cdot, e, \sqsubseteq),$$

where R is the set of resource elements of the given type, $\cdot : R \times R \rightharpoonup R$ is a partial function, e is a unit (or identity) element for $\cdot$ such that, for all $r \in R$, $r \cdot e = r = e \cdot r$, and $\sqsubseteq$ is a preorder on R. In coin example case, the monoid of resources can be taken to be the ordered monoid of natural numbers,

$$(\mathbb{N}, +, 0, \leq).$$

The partiality of $\cdot$ reflects that in many natural examples of resource, such as computer memory, not all combinations of resource elements will be defined. In this coin example, addition of natural numbers happens to be total, but this is not necessary. Where it helps clarity, I write $r\!\downarrow$ to denote that a resource r is defined.

Finally, for technical mathematical reasons, logicians required that the combination $\cdot$ and comparison $\sqsubseteq$ of resources should interact conveniently. Specifically, the following functoriality condition is required: for all r_1, r_2, s_1, s_2,

$$r_1 \sqsubseteq r_2 \text{ and } s_1 \sqsubseteq s_2 \text{ implies } r_1 \cdot s_1 \sqsubseteq r_2 \cdot s_2.$$

For example, in the ordered monoid of natural numbers, $(\mathbb{N}, +, 0, \leq)$, if $m_1 \leq m_2$ and $n_1 \leq n_2$ implies $m_1 + n_1 \leq m_2 + n_2$.

This set-up is known as *resource semantics*. The technical choices made in setting up resource semantics provide a mathematical representation of the engineer's model of the two desired properties about resources. This mathematical structure is a necessary step to translate between the English-language description of resources and a formal logical model one can reason with automatically. For a more detailed history and sociology of the way the mathematical structure of monoids have been used in computer science, see Petricek

11 A preorder $\sqsubseteq$ on a set S is required to be reflexive and transitive. It is not a total order.

(2018). This mathematical structure is exactly what is required to define a formal logical model of BI.

The starting point for this is intuitionistic logic (Kripke, 1965) and its Kripke semantics in which an implication $\phi \to \psi$ is interpreted as a function, or procedure, that converts evidence for the truth of ϕ into evidence for the truth of ψ. Technically, this is achieved using a preorder on the set of possible worlds, or states of knowledge (Van Dalen, 2004): if an observer can establish the truth of ψ from the truth of ϕ at its current state of knowledge, then it must also be able to do so at any greater state of knowledge; this is called *monotonicity*.

A similar interpretation can be applied to the separating conjunction, $*$, described in Section 6.3. Where $r \models \phi * \phi$ says I have enough money to buy two drinks, if $r \sqsubseteq s$, then it will always be the case that $s \models \phi * \phi$. Intuitively, if r is a world in which you have 10 coins, and that is enough for two lemonades, then in any world where you have at least 10 coins (i.e., s) will also satisfice for two lemonades.

Monotonicity is defined formally below.

With these interpretations in mind, and assuming (i) a 'resource monoid' $\mathbf{R} = (R, \cdot, e, \sqsubseteq)$, (ii) that $r \models \phi$ is understood as 'the resource r is sufficient for ϕ to be true', and (iii) for each atomic proposition p, a set $\mathcal{V}(\mathrm{p})$ of resource elements that are sufficient for $\mathcal{V}(\mathrm{p})$ to be true, consider a formal semantics to BI as follows, where $r \models \phi$ is read, as before, as 'the world r supports, or satisfies, the proposition phi':

$$
\begin{array}{rcl}
r \models \mathrm{p} & \text{iff} & r \in \mathcal{V}(\mathrm{p}) \\
r \models \bot & \text{never} & \\
r \models \top & \text{always} & \\
r \models \phi \vee \psi & \text{iff} & r \models \phi \text{ or } r \models \psi \\
r \models \phi \wedge \psi & \text{iff} & r \models \phi \text{ and } r \models \psi \\
r \models \phi \to \psi & \text{iff} & \text{for all } r \sqsubseteq s,\ s \models \phi \text{ implies } s \models \psi \\
r \models \mathrm{I} & \text{iff} & r \sqsubseteq e \\
r \models \phi * \psi & \text{iff} & \text{there are worlds } s \text{ and } t \text{ such that } (s \cdot t)\!\downarrow \sqsubseteq r \\
 & & \text{and } s \models \phi \text{ and } t \models \psi \\
r \models \phi \mathrel{-\!\!*} \psi & \text{iff} & \text{for all } s \text{ such that } s \models \phi \text{ and } (r \cdot s)\downarrow, \\
 & & r \cdot s \models \psi.
\end{array}
$$

All propositions ϕ are required to satisfy *monotonicity*: if $r \models \phi$ and $r \sqsubseteq r'$, then $r' \models \phi$.

With this semantics and with a system of rules of inference along the lines of the ones sketched above, Galmiche et al. (2005) and Pym et al. (2004) obtain soundness and completeness theorems for BI: the propositions that are provable using the inference rules correspond exactly to the ones that are true according to the semantics.

In the context of this semantics, the significance of the contraction and weakening rules can now be seen: they explain how the semi-colon combines properties of resources that may be shared whereas the comma combines properties of resources that must be separated.

This description is the original, intuitionistic formulation of BI. However, Separation Logic in fact builds on the classical or 'Boolean' variant (Ishtiaq and O'Hearn, 2001; Reynolds, 2002). Boolean BI is based on classical logic, so that the implication $\phi \to \psi$ is defined to be $(\neg\phi) \vee \psi$, where the negation satisfies the classical 'law of the excluded middle'. Technically, Boolean BI works with a resource semantics based on simply partial commutative monoids, without including a preorder. That is,

$$\mathbf{R} = (R, \cdot, e),$$

where R is the set of resource elements of the given type, $\cdot : R \times R \rightharpoonup R$ is a partial function, e is a unit (or identity) element for $\cdot$ such that, for all $r \in R$, $r \cdot e = r = e \cdot r$.

With models of this form, the semantics of Boolean BI is given as above, but with the following variations:

$$\begin{aligned}
r \models \phi \to \psi \quad & \text{iff} \quad r \models \phi \text{ implies } r \models \psi \\
r \models \mathrm{I} \quad & \text{iff} \quad r = e \\
r \models \phi * \psi \quad & \text{iff} \quad \text{there are worlds } s \text{ and } t \text{ such that } (s \cdot t) \downarrow = r \text{ and} \\
& \qquad\quad s \models \phi \text{ and } t \models \psi.
\end{aligned}$$

Notice that in Boolean BI the separating conjunction divides the resources exactly.

6.4.3 *The Resource Semantics of Separation Logic*

The resource semantics described above, much richer than that in linear logic (Girard, 1987), allowed the construction of specific logical models for a characterization of computer memory. Characterizing memory addressed challenging problems in program verification (Ishtiaq and O'Hearn, 2001). Over the next 15 years, this logic acquired the name Separation Logic (O'Hearn, 2007; Reynolds, 2002) and developed into a reasoning tool successfully deployed at large technology firms like Facebook (O'Hearn, 2015) and Spotify (Vuillard, 2016). This section will explain how the semantics of (Boolean) BI forms the basis of Separation Logic.

Ishtiaq and O'Hearn (2001) introduced 'BI Pointer Logic', based on a specific example of Boolean BI's resource semantics. Three points about BI Pointer Logic are important to the historical development of a logical model adapting sufficiently closely to the engineer's model to be useful.

- First, its resource semantics is constructed using the stack, used for static, compile-time memory allocation, and the heap, used for dynamic, run-time memory allocation.
- Second, the semantics of the separating conjunction, $*$, splits the heap, but not the stack. The stack contains the allocations required to define the program, which are unchanged at run-time; the heap contains the allocations made during computation.
- Third, it employs a special class of atomic propositions constructed using the 'points to' relation, $\mapsto$: $E \mapsto E_1, E_2$ means that expression E points to a cons cell E_1 and E_2. (It also employs a class of atomic propositions which assert the equality of program expressions, but this is a standard formulation.)

These factors combine to give an expressive and convenient tool for making statements about the contexts of heap (cons) cells. For example, the separating conjunction

$$(x \mapsto 3, y) * (y \mapsto 4, x)$$

says that x and y denote distinct locations. Further, x is a structured variable with two data types; the first, an integer, is 3, and the second is a pointer to y. The variable y denotes a location with a similar two-part structure in which the first part, also called the car, contains 4 and the second part, sometimes

called the cdr ('could-er'), contains a pointer back to x (Ishtiaq and O'Hearn, 2001). Note that the pointers identify the whole two-part variable, not just the car. Figure 6.3 displays this linked list relationship.

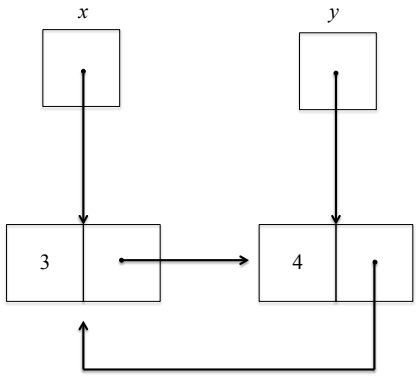

Figure 6.3: As in Figure 6.1, variable names are listed above their square, and contents of the variable are inside the square. The diagram represents the logical statement $(x \mapsto 3, y) * (y \mapsto 4, x)$.

Separation Logic can usefully and safely be seen, per O'Hearn and Yang (2002), as a presentation of BI Pointer Logic. The semantics of BI Pointer Logic is an instance of the resource semantics of first-order Boolean BI in which the monoid of resources is constructed from the program's heap. In detail, this model has two components, the stack and the heap. The stack is a partial function mapping from variables to values, $a \in \mathrm{Val}$, such as integers, and the heap is a partial function from natural numbers to values. In logic, the stack is often called the valuation, and the heap is a possible world. In programming languages, the stack is sometimes called the environment.

Within this set-up, the atomic formulae of BI Pointer Logic include equality between expressions, $E = E'$, and, crucially, the points-to relation, $E \mapsto F$. This set up requires some additional notation for the components of the computer and how the logic handles them. $dom(h)$ denotes the domain of definition of a heap h and $dom(s)$ is the domain of a stack s; $h\#h'$ denotes that $dom(h) \cap dom(h') = \emptyset$; $h \cdot h'$ denotes the union of functions with disjoint domains, which is undefined if the domains overlap; $[f \mid v \mapsto a]$ is the partial function that is equal to f except that v maps to a; expressions E are built up from variables and constants, and so determine denotations $[\![E]\!]s \in \mathrm{Val}$. With this notation, the satisfaction relation for BI Pointer Logic is defined as in Figure 6.4.

$$
\begin{array}{rll}
s, h \models E = E' & \text{iff} & [\![E]\!]s = [\![E']\!]s \\
s, h \models E \mapsto (E_1, E_2) & \text{iff} & [\![E]\!]s = \mathrm{dom}(h) \text{ and} \\
 & & h([\![E]\!]s) = \langle [\![E_1]\!]s, [\![E_2]\!]s \rangle \\
s, h \models \mathtt{emp} & \text{iff} & h = [\,] \text{ (the empty heap)} \\
s, h \models \phi * \psi & \text{iff} & \text{there are } h_0,\ h_1 \text{ s.t. } h_0 \# h_1, \\
 & & h_0 \cdot h_1 = h,\ s, h_0 \models \phi \text{ and } s, h_1 \models \psi \\
s, h \models \phi \mathrel{-\!\!*} \psi & \text{iff} & \text{for all } h', \text{ if } h' \# h \text{ and } s, h' \models \phi, \\
 & & \text{then } s, h \cdot h' \models \psi \\
s, h \models \bot & \text{never} & \\
s, h \models \phi \to \psi & \text{iff} & s, h \models \phi \text{ implies } s, h \models \psi \\
s, h \models \exists x \,.\, \phi & \text{iff} & \text{for some } v \in \mathrm{Val},\ [s \mid x \mapsto v], h \models \phi
\end{array}
$$

Figure 6.4: The satisfaction relation for BI Pointer Logic (Ishtiaq and O'Hearn, 2001).

The judgment $s, h \vDash \phi$ says that the assertion ϕ holds for a given stack and heap, assuming that the free variables of ϕ are in the $dom(s)$.

The remaining classical connectives are defined in the usual way: $\neg\phi = \phi \to \bot$; $\top = \neg\bot$; $\phi \vee \psi = (\neg\phi) \to \psi$; $\phi \wedge \psi = \neg(\neg\phi \vee \neg\psi)$; and $\forall x \,.\, \phi = \neg\exists x \,.\, \neg\phi$.

The definition of truth for BI Pointer Logic – that is, its satisfaction relation – provides a clear illustration of the merging of logic-models and engineering-models. The stack and the heap and how they are manipulated by programs are considered directly by working programmers; indeed, memory management at this level of abstraction is a key aspect of the C programming language (see Kernighan and Ritchie (1988) for descriptions of the history, definition, and usage of C). BI Pointer Logic, with its truth-functional semantics of the form

$$s, h \models \phi$$

provides elegant semantics for reasoning about the correctness of programs that manipulate computer memory.

However, this is not yet the full story of how Separation Logic was implemented. Hoare logic, based on triples $\{\phi\}\, C\, \{\psi\}$, is both more natural and convenient for reasoning directly about the behaviour of programs. The main reason why Hoare triples are so convenient is that they directly include code, C, whereas BI Pointer Logic is formulated wholly in terms of properties of the contents of memory. These two points of view were connected by providing

a semantics of Hoare triples in terms of BI Pointer Logic (Calcagno et al., 2007). There are essentially two ways of connecting them, depending upon on the strength of requirements on the behaviour of the code. The behaviour of code is expressed in terms of the evaluation of a program C, using stack s and heap h, with respect to sequences of steps defined by its operational semantics ($\rightsquigarrow$). Code behaviour is essentially denoted by $C, s, h \rightsquigarrow^* s', h'$, read as 'the program C transforms the memory configuration s, h into the memory configuration s', h'. There is a special configuration, *fault*, indicating a memory fault or abnormality.

The first semantics for Hoare triples, partial correctness, relies on the notion of safety,

$$C, s, h \text{ is safe if } C, s, h \not\rightsquigarrow^* \mathit{fault}$$

and is the 'fault-avoiding' interpretation, per O'Hearn and Yang (2002).

> *Partial correctness semantics*: $\{\phi\}\, C\, \{\psi\}$ is true in a model of Pointer Logic if, for all s, h, it is the case that $s, h \models \phi$ implies
>
> – C, s, h is safe, and
> – if $C, s, h \rightsquigarrow^* s', h'$, then $s', h' \models \psi$.

The second semantics, called total correctness (O'Hearn and Yang, 2002), does not require the safety condition because it requires the 'stronger' property of 'normal' termination, which requires the program returns a value that lies within its intended range of outputs:

> *Total correctness semantics*: $\{\phi\}\, C\, \{\psi\}$ is true in a model of Pointer Logic if, for all s, h, it is the case that $s, h \models \phi$ implies
>
> – C, s, h must terminate normally, and
> – if $C, s, h \rightsquigarrow^* s', h'$, then $s', h' \models \psi$.

These definitions took the development of a Frame Rule for Separation Logic most of the way. The further necessary elements were some non-trivial technical development as well as soundness (that the rule transforms true properties into true properties) and completeness (that the rule derives one specification statement from another just when this inference holds semantically) theorems for the Frame Rule. The formal statement of the Frame Rule for Separation Logic is (O'Hearn and Yang, 2002):

$$\frac{\{\phi\}\, C\, \{\psi\}}{\{\phi * \chi\}\, C\, \{\psi * \chi\}} \text{ Modifies}(C) \cap \text{Free}(\chi) = \emptyset,$$

This theorem says, roughly, that conclusions about what a set of instructions (C) does can ignore the activity in other parts of the program, so long as those other parts do not interact with the variables used by the set of instructions.

Together, the above theorems give precise mathematical expression to the coincidence of the logical and engineering models of computer memory allocation. More explicitly, given soundness and completeness properties for the Frame Rule (O'Hearn and Yang, 2002), it exactly characterizes logical truth for local reasoning about memory allocation.

This section provided some detail on the novel aspects of Separation Logic's semantics, and how they support reasoning about computer memory as a resource. At heart, the atoms of the logic are composable in a way that mirrors the way that the physical substrate is composable. The physical transistors come apart, and one can make meaningful claims about affixing or pulling apart bits of silicon that have reliable impacts on the changes to the electrical and computational properties of the physical system. The structure of the logical model using partial commutative monoids and $*$ allows claims made using Separation Logic to naturally mirror this physical fact.

Section 6.5 will detail the cluster of properties surrounding the proof theory of Separation Logic that make it a successful engineering tool. These details are related to the composability of $*$ through the Frame Rule, as it is leveraged for efficient computation of results. Equally important to the deployability of the proof theory is the automation of bi-abduction for generating hypothetical pre- and post-conditions to drive proof solutions. The abductive rules in use are essentially encodings of engineer's heuristics about computer memory usage, further demonstrating the deep ways in which the logical and engineering aspects of the task merge in Separation Logic.

6.5 DEPLOYABLE PROOF THEORY FOR SEPARATION LOGIC

An important consequence of a system of logic having a completeness theorem is that its proof system can be used as a basis for formal reasoning within it. Consequently, the study of the automation of proof systems — that is, the provision of computationally feasible presentations of proof systems — is a widely studied topic in modern logic. Perhaps the most famous example is the provision of resolution systems for classical logic (Robinson, 1965). Resolution is deployed in the programming language Prolog (Hodgson, 1999; Van Emden and Kowalski, 1976) and within SAT solvers, with good results (Heule and

Kullmann, 2017). Not every proof system supports mechanized reasoning as well as it does resolution; for instance, a natural deduction or a Hilbert-style system sees many choices of applicable proof rules at any time, and this leads to a blow up of the search space.

Let us call a proof system that can support effective mechanized reasoning *deployable*. That is, the search for, and construction of, proofs in the system is computationally tractable. For a problem to be tractable, the compute resources required are acceptable for their intended use. This section discusses the intellectual choices that make Separation Logic deployable. Actual deployment requires integration with software engineering practices. Integration is not trivial; the deployment challenges associated with Infer are described by O'Hearn (2015).

The setting of Separation Logic presents a deployable proof system for a semantics that simultaneously captures the engineering model of computer memory allocation and its logical interpretation. There are two key properties that a proof theory ought to have to be deployable: scalability and automation. Separation Logic achieves these engineering-type implementation goals through features built in to the logic. Scalability comes mainly from access to the Frame Rule, and the parallel computation that it enables. Automation of proofs is a large topic on its own, with tools such as Coq. However, Separation Logic has also been used to make systems that can automate reasoning about what is relevant to attempt to prove — that is, abduction. Automating abduction in this context means formalising heuristics engineers use to diagnose errors. The logical system must be tailored to accomplish this task.

6.5.1 *Separation Logic and the Frame Rule*

The formal definition of the Frame Rule for Separation Logic was introduced in Section 6.4.3. The 'frame' in the Frame Rule is essentially a context, formally a set of logical statements In software engineering, the frame is the variables and memory resources that a program modifies. The Frame Rule lets the analyst break a program into disjoint fragments, analyse them separately, and cleanly and quickly conjoin the results. The Frame Rule allows the following: as long as the frame and the program do not modify each other's variables, one can freely conjoin the frame to the pre- and post-conditions for the program.

Consider a drinks-as-resources analogy. If the 'program' of interest is I drink my drink, a sensible pre-condition is that I have a full drink. The post-

condition is, let's say, that I have an empty glass. The frame then is all the other drinks in the restaurant, as well as the food, and the sunshine outside, as long as there is no joker in the place going about pouring people's drinks into one another's glasses. In computer programming, logicians can check rigorously for such jokers because they can check what variables (in this example, the glasses) different programs can access.

The benefits for scalability, and therefore deployability, are immediate. Imagine if one had to re-analyse one's 'program' for drinking a glass of water every time another patron entered or exited the restaurant, or any time any other patron refilled or finished their own drink. In program verification, this is a serious threat to viable tools. Programs change often, and are expensive to analyse wholesale. It is not plausible to reanalyse a whole program for each minor change. The Frame Rule gives Separation Logic a deployable proof theory for two reasons. First is the facility it provides for the saving of results from past analyses of unchanged program fragments and applying them quickly to analyse small, changed fragments. The second reason is perhaps more subtle, but more powerful. Modern computing is done largely in clouds owned by giant tech companies. The benefit of cloud computing is that hundreds of thousands of processors can work on a computation in parallel and merge their results. Without the Frame Rule, Separation Logic would not be able to take advantage of the massive computational resources of cloud computing; parallelization requires fragmentation of a problem into smaller parts and sound merging of results.

6.5.2 *Deployability via Contextual Refinement*

The Frame Rule is not the only path to a deployable proof theory for Separation Logic. Xu et al. (2016) describe an extension of Concurrent Separation Logic (O'Hearn, 2007) that uses contextual refinement between implementation and specification of a program to prove the correctness of the program. Contextual refinement is a formal specification of the following property: the implementation – that is, the actual written computer code – does not have any observable behaviours that the abstract design specification of the system does not have.

Xu et al. (2016) deploy Separation Logic to verify the scheduling behaviour of operating system kernels.[12] This application of Separation Logic treats computer memory as a resource. However, the relevant memory property is unique ownership by a task, rather than unique identification of a pointer location. The main difficulty in scheduler design is ensuring that two programs that both hold the same pointer do not interfere with each other. The technical details are out of scope; however, this is a common and challenging computer science problem. Complex scheduling has been common since the mid 1990s to improve hardware usage efficiency. Deployable verification of scheduling for a real-world (pre-emptive) operating system kernel uses Separation Logic (Xu et al., 2016).

The design of a logic to verify operating system scheduling is tailored to that problem, to the extent that 'the interrupt mechanism in our operational semantics is modelled specifically based on the Intel 8259 A interrupt controller, and the program logic rules for interrupts are designed accordingly' (Xu et al., 2016, p. 77). To arrive at a satisfactory semantics, the authors modelled the behaviour of a specific processor on specific Intel hardware. This clearly demonstrates the merging of the logical model and the engineering model. The inference rules Xu et al. (2016) uses are different from those used by Calcagno et al. (2011). The following subsection focuses on inference rules over memory allocation via Calcagno et al. (2011); there are analogous rules for operating system scheduling which I elide (Xu et al., 2016, p. 72).

6.5.3 *Bi-abduction*

Section 6.3 briefly discussed the importance for the effectiveness of Separation Logic of the concept of abduction. This section introduces how it is integrated into the logic. Abduction was introduced by Charles Peirce around 1900, when writing about the scientific process, and explained by Peirce as follows:

> 'Abduction is the process of forming an explanatory hypothesis. It is the only logical operation which introduces any new idea' (Bergman and Paavola, 2016, CP 5.171).

Consider a non-technical example. A baby is crying for no obvious reason. Approaching the problem like an engineer, you should like to know the source

12 The kernel is the most trusted part of the operating system; it coordinates all other applications' access to the physical hardware.

of the baby's distress, in order to devise a method to allay it. But as no one saw the child begin to cry, you must guess at, or abduce, the source. Perhaps you abduce that a malicious Cartesian demon is making the baby believe it is being viciously pinched. Or perhaps you guess hunger is the source. Neither are entirely new ideas, both suggested by past experience with, and structure of, the world. Yet the abduction of hunger is preferable, if for no other reason than there is a ready method to allay hunger, and none such for demons. That is, whether the baby is hungry is testable. One can guess at the expected post-condition after intervening if the precondition is true: if the baby is hungry, and then fed, then the baby will stop crying. If once fed, the baby does not stop, we surmise our guess failed and move to abduce something else. Thus, even though there are incalculably many conceivable causes of the baby's crying, the structure of the situation suggests certain abductions. Knowing, or abducing, what should or might be true of conditions after a process or intervention puts constructive constraints on the abductions of prior conditions.[13]

There is not a general process by which one generates useful new ideas. However, if one has both a precise language and a detailed conception of the mechanisms of interest in the system, abduction becomes more tractable. The logic and the engineering model of computer memory, respectively, provide these features. Further, the proof theory provides a fast and composable method for soundly checking the correctness of the guesses from abduction. Infer can automate abduction in the case of looking for pre-conditions and post-conditions that lead to memory errors in computer code.

Abduction in classical logic is formalised, deceptively simply, as:

Given: assumption ϕ and goal ψ;
Find: additional assumptions χ such that $\phi \wedge \chi \vdash \psi$.

In this expression it is customary to disallow trivial solutions, such as $\phi \to \psi$. Reasoning about computer memory and pointers uses the separating conjunction in the obvious analogue:

Given: assumption ϕ and goal ψ;
Find: additional assumptions χ such that $\phi * \chi \vdash \psi$.

13 This process is not solely abduction. Such hypothesis generation is also constrained by general knowledge of the mechanisms that often lead to a baby crying; in this way Chapter 4 also helps guide abduction. The ability to manipulate models, in general, is also needed. Structural reasoning (Swoyer, 1991) thus perhaps plays a role in addition to mechanistic knowledge. However, this chapter focuses only on abduction as that is the feature that Separation Logic identifies as automated.

Because the problem domain is program analysis and specifically the program's use of memory, χ is constrained to formula representing a heap. This constraint disallows trivial solutions such as $\phi \mathrel{-\!\!*} \psi$ (Calcagno et al., 2011, p. 6).

To contribute genuinely to a deployable proof theory, one needs to know both the pre-conditions necessary for the piece of code to run safely and also all the logical conditions that will be true after the piece of code finishes. Post-conditions for a single piece of code do not help to verify that particular piece of code. However, computer programs are complex arrangements of separable but interrelated pieces of code. The post-conditions of one segment are good candidate guesses for pre-conditions of other segments. Calcagno et al. (2011) coin the term bi-abduction for finding both pre- and post-conditions. In program analysis, the pre-conditions are the anti-frame and the post-conditions are the frame, so bi-abduction is formalized as follows:

Given: assumption ϕ and goal ψ;
Find: additional assumptions ?anti-frame and ?frame such that
$\phi * ?\text{anti-frame} \vdash \psi * ?\text{frame}$.

The statement's specific logical form, the model of the mechanism of computer-memory use, and the machine-readable nature of computer programs, all combine to allow automatic generation of potential solutions to the frame and anti-frame. The result of this synthesis makes bi-abduction 'an inference technique to realize the principle of local reasoning' (Calcagno et al., 2011, p. 8).

Consider some step-by-step bi-abduction examples. First, consider ascertaining pre-conditions in some detail. The following does not assume any familiarity with C or with programming, so I will also explain the target program segment in English detail.

Calcagno et al. (2011, p. 8) use this example to explain abduction:

```
void free_list(struct node *x){
  while (x!=o) {
   t=x;
   x=x->tl;
   free(t);
  }
}
```

This example program steps through or traverses all the elements of a list and removes them. Literally, it frees the memory used to store each element.

Let's use the example of a shopping list, for concreteness. Traversing a list is just to read all the elements in order. For a paper list, this ordering is handled by the physical layout on the paper. Eggs are first if they are on the first line. In computer memory, a directly analogous physical layout is difficult and inefficient for technical reasons. Instead each list element contains two parts. First, its contents, say 'eggs', and second, a pointer to the next element. Pointers, as discussed in Section 6.2, can cause myriad problems during a program's execution. Such linked lists are a common place to find pointers, and so a common place to find memory management errors.

When reasoning tools encounter a program like `free_list`, they start off assuming an empty heap (*emp*) and that the variable x has some value X. However, at the line '`x=x->tl`' the reasoning stalls. There needs to be some X' to which X points. Using abduction, the tool guesses that such an X' exists. Another step is required. In the general case, there is an infinite regress of assuming ever-more X'', X''', and so on. Separation Logic requires an abstraction step, which Calcagno et al. (2011, p. 9) link to a scientific induction step. The abstraction step is to posit a list of arbitrary length from X to X' and to assert or abduce that a program that works on lists of length 4 probably works on lists of length 6. The trick is to encode these heuristics, such as the guess that an X' exists, into formal proof rules that can be applied automatically. Abduction and abstraction potentially weaken preconditions. Weakening may be unsound, and must be tested. But such tests can also be automated in Separation Logic. Calcagno et al. (2011, p. 10) describe perhaps 50 pages of their article as 'devoted to filling out this basic idea [of using abduction to guess what good preconditions might be]'. Consider one further example to illustrate some of the complications that can arise in the task.

Lists can get more complicated. For example, the last element can link back to the first. Imagine taping a shopping list into a loop, so that 'eggs', the first element, came on the line after the last element, 'chocolate'. The C syntax of a program to handle such a circular list is (Calcagno et al., 2011, p. 53):

```
void traverse-circ(struct node *c) {
  struct node *h;
  h=c; c=c->tl;
  while (c!=h) { c=c->t1;}
}
```

Human shoppers would not start over and traverse the list again, picking up a second copy of everything on the list. And then a third, looping through the list until their cart overflowed. However, `free_list` would naïvely enter such an infinite loop. So `traverse-circ` not only reads an element and goes to the next one, but remembers where it started so that it can stop after going through once. Since the program is designed to read circular lists, the logic produces a circular list as a pre-condition. Specifically, one abduces the precondition (Calcagno et al., 2011, p. 52)

$$c \mapsto c\_ * list(c\_, c)$$

That is, for the program to run safely, the input (c) must be a pointer to a valid element of memory ($c\_$), and separately there must be a linked list going from that valid element back to the initial element.

Let us explore in more detail the formal form of this abduction, which is Algorithm 4 in (Calcagno et al., 2011, p. 37). The algorithm is run (by another computer program) with the small program of interest as input, along with a guess at the starting state. The first steps of the algorithm build a logical model of the program's interaction with memory. The logical model takes the form of Hoare triples. How exactly a computer program is soundly converted into Hoare triples is a matter of shape analysis, or 'determining "shape invariants" for programs that perform destructive updating on dynamically allocated storage' (Sagiv et al., 2002). There are technical details about converting the program to a logical model that are out of scope here, but note that the logical model and language are purpose-built tools for this task. Going back to Hoare's explicit axiomatization of programs (Hoare, 1969) through to the definition of $\mapsto$ for the function of a stack element pointing to a heap location, both broad strokes and finer details of the logic are responsive to the problem at hand.

After constructing the logical model, Algorithm 4 iterates through all of the Hoare triples and calls AbduceAndAdapt (Calcagno et al., 2011, p. 43). This function has two main purposes: to do bi-abduction, and to take any successful results from bi-abduction and 'perform essential but intricate trickery with variables' to maintain precise results. The abduction aspect of the algorithm is specified in Algorithm 1. This algorithm, in turn, depends upon a set of

proof rules used in reverse as abduction heuristics (Calcagno et al., 2011, p. 15-17). The rules are all of a special form,

$$\frac{H_1' * [M'] \triangleright H_2' \quad Cond}{H_1 * [M] \triangleright H_2}$$

Here *Cond* is a condition on the application of the rule based on parts of H_1 and H_2. The proof rules can thus be read backwards to create a recursive algorithm that will eventually abduce pre- and post-conditions. To read them in this manner, the algorithm checks that the condition holds. If so, instead of answering the (harder) question $H_1 * ?? \vdash H_2$, the algorithm goes on to search for the answer to the (simpler) abduction question $H_1' * ?? \vdash H_2'$ (Calcagno et al., 2011, p. 17).

The example at hand, traverse-circ, will hit the heuristic 'ls-right' until the list loops, generating the precondition that there is a list from $c\_$. The other precondition is generated by the heuristic '$\mapsto$-match'. These are linked in the 'intricate trickery' done in the algorithmic step to keep results precise.

The details of which proof rules are chosen as abduction heuristics is important and non-trivial. The choice is based on decades of prior experience and empirical results on the effectiveness of different modelling choices. The logic has been shaped to be a tool to solve the engineering problem at hand such that the proof rules are chosen empirically.

The postconditions of this example seem less exciting. The program only reads the list, it does not output any contents nor change it. Therefore, the abduced post-conditions will be the same as the preconditions. While this initially seems unenlightening, remember that bi-abduction is on program segments, not whole stand-alone programs. So if a larger, more realistic program runs this traverse-circ process successfully, and it had the necessary preconditions, then there is a circular linked list in memory. This information may be very helpful for determining whether another program segment runs safely. For example, a process that deletes elements of a list one at a time often has the flaw that it will not check for circular lists. When such a delete process cycles, it will try to delete the now non-existent first list-element, causing a memory error that can result in a crash. In such a situation, this post-condition of a circular linked list would be informative. For more details on how to abduce postconditions, see Algorithm 6 in Calcagno et al. (2011).

Abduction is automatable in this situation because the problem space investigated by the engineering/scientific model is quite precisely defined. In-

stead one might say that abduction is automatable here because the logical model sufficiently accurately represents the behaviour of real computer programs. These two assessments are both true, and amount to the same thing: effective merging of the features of the logical model and the conceptual model. Automated abduction is a striking example of the benefits of such a confluence.

The best measure of whether a proof theory is deployable for finding errors in software is whether programmers in fact fix the errors it finds. For programmers to fix errors, the tool must provide a combination of timely results, precise results, and clear explanations. These are part of usefulness requirements within the industrial software engineering setting that are essentially social or organizational (Calcagno et al., 2015a). Therefore, what counts as a satisfactory fix-rate may change from one organization to another. Infer is open-source and used by many organizations. Separation Logic is measured as deployable in some sense because it is deployed in these contexts. For an account of the social and practical environment necessary to shepherd Infer to deployment, see Calcagno et al. (2015a).

Section 6.2 detailed why finding memory usage flaws is an important task in computer programming. Programmers make these errors, and in products that are widely used. Further, these kinds of errors impact stability and security in costly ways that are hard to catch and handle during execution. Separation Logic has been tailored to this problem specifically, through adaptations to both its semantics (detailed in Section 6.4) and proof theory. This section detailed how the proof theory has been made deployable, to meet the needs of industrial application. It is deployable because (1) its reasoning is scalable and fast, using the compositionality of the Frame Rule; and (2) its generation of hypothetical pre- and post-conditions is automated using encoded discovery heuristics and bi-abduction.

While the Infer tool originally grew out of Separation Logic and bi-abduction, it has evolved to encompass other reasoning methods using a technique known as abstract interpretation (Cousot and Cousot, 1977). Some of the ideas from Separation Logic have been transposed into other reasoning methods, without adopting the syntax of Separation Logic. For instance, a reasoning technique for finding errors due to conflicts in concurrently executing activities calculates the accesses or resources used by a program component, and then composes these with the resources for other program parts (Blackshear and O'Hearn, 2017). Thus, the ideas of resource-based reasoning and composition of this reasoning (as in the Frame rule), which

were formulated in BI and Separation Logic, appear to be fundamental, formalism-independent concepts in reasoning about computer programs.

6.6 CONCLUSION

This chapter introduced Separation Logic as a tool for reasoning about computer programs. This case study identified that Separation Logic works because the logic model overlaps with the conceptual model of a practical problem and the proof theory is usefully deployable. I view this convergence as a tactic for model building. Although I have mostly referred to the programmer's model of memory as an engineering model, it is also clearly identifiable as a mechanistic explanation of memory and thus a sort of general scientific knowledge. Therefore, the lessons from this case study should be transferable to any CSIR reasoning task using scientific or engineering knowledge that can be precisely logically specified.

The type of errors that Separation Logic is currently used to find are constrained to a specific, though important, type of catastrophic run-time error. There are at least two types of run-time errors – memory allocation (Calcagno et al., 2011) and task scheduling (Xu et al., 2016) – that have been addressed with Separation Logic. These types of errors arise in a variety of applications, from hardware synthesis (Winterstein et al., 2016) to computer security (Appel, 2015) to popular phone apps. Separation Logic is not the solution to all computer science problems, but it is not so specific as to be uninteresting.

Other specific problems will likely require logics tailored to them. As one example, Lamport (2002) details temporal logic which is used by Amazon for its network architecture (Newcombe et al., 2015). Another aspect of assuring memory, called shared memory consistency, used yet a different logic model to address its programming problem (Adve and Gharachorloo, 1996). These other examples of bringing a programming/engineering model into close contact with an adequately designed logic model strengthen my conclusion. The history of Separation Logic, through to its implementation in deployed verification tools, demonstrates that such overlap is an effective strategy for reasoning about the behaviour of computer systems. See O'Hearn (2015) and Calcagno et al. (2015a) for accounts of the software-engineering effort involved in deploying one such tool.

It is important to understand the extent to which the case of Separation Logic is relevant to both computer-science models and science more generally.

Model-based reasoning in computer science seems to come in at least two flavours. Some parts of computer science, like human-computer interaction and usable security, have methodologies that are closely adapted from established fields like psychology (Krol et al., 2016). However, in other parts of the field, computer-science methods are distinctly developed within the discipline. Hatleback and Spring (2014) argued that experiments and model-building in computing are not so different from other sciences, after accounting for the unique challenges of the fields. Separation Logic provides a good example of this second type; the above examples of temporal logic and shared memory consistency indicate it is not alone. Both Hatleback and Spring (2014) and Chapter 4 argued that reasoning about objects that can purposefully change at the same pace as the observer can interact with them, namely software, is a particular challenge in computer science. Separation Logic is an example of how computer scientists overcome this problem. Reasoning at the appropriate level of abstraction produces stable representations of the phenomenon so that conclusions are reliable. This stability issue was a common complaint identified by Chapter 3 in the science of security literature; Separation Logic is thus a plausible tool for tackling this aspect of challenges to stable knowledge in incident analysis.

This approach would not get off the ground without a deployable proof theory, no matter how nice the overlap between the model of computer memory and the logical interpretation of exhaustible resources. In fact, exploiting the model structure for some practical benefit, such as timely parallel computation, is perhaps more rare and important than devising a model that is simultaneously a logic and an engineering model. Verification tools using Separation Logic reach a deployable proof theory due to a constrained domain that permits the automation of abduction combined with a composable logic that permits reuse of results. The main points of this development are (1) the introduction of the logic of bunched implications, which admits the usual conjunction with contraction and weakening rules and a different conjunction that does not; (2) the semantics of a resource as a preordered partial commutative monoid; (3) a full definition of the connectives $*$ and $-\!\!*$.

Philosophical logic has a long tradition of analysis of arguments and meaning. One message for logicians is that it can have more clarity and impact when the model theory is grounded in concrete engineering or scientific problems; that is, where the elements of the model have a clear reading or interpretation apart from their role in defining the semantics of sentences. For example,

relevant-logicians have admitted to struggles in interpreting the meaning of the elements in their formal semantics based on ternary relations (Beall et al., 2012). Their semantics enjoys completeness theorems with respect to their proof theories, but the subject matter of the models themselves is not evident. In contrast, this chapter has shown a nearby semantics, not identical, where the model elements are understood in terms of the structure of computer memory – and more generally of resources (Pym et al., 2004). In general, when the semantics of logics meets independently-existing science and engineering, a feedback cycle can be set up which impacts both to mutual benefit.

Logic, like any other technology, must be designed to specifications for the task at hand. In concert with design, the logic employed should be empirically tested as to whether it meets specifications. This sort of feedback loop is not so different from the tool-building and scientific modelling interaction in other fields. However, unlike, say, biology whose tools are often of glass and metal, the tools in computer science are often conceptual or logical tools. Considering computer science as the field that explores the human-created abstractions of mathematics and logic, this tooling change makes sense. Moreover, understanding that when humans build or define some system there is no automatic guarantee of the properties and behaviours of said system perhaps elucidates why, as Tedre and Moisseinen (2014) argues, computer science can often usefully be considered an experimental science.

Separation Logic is also a useful case for communicating salient aspects of computer science to the broader philosophy of science community. For the debate on model-based reasoning, Separation Logic is an automatable system for model-based reasoning, albeit in a tightly constrained environment. Perhaps such extensive context constraints are necessary to formalize reasoning to the level of detail necessary for automation. In addition, the case study provides a starting point from which philosophers may be able to generalize broader lessons for model-based reasoning.

Part III

A LOGIC OF REASONING IN INCIDENT ANALYSIS

This part, in one chapter, constructs a logic suitable for reasoning about Computer Security Incident Response (CSIR). Chapter 7 begins with a short summary of the design requirements developed in the previous two parts. It then will define and elaborate the logic. The definition will include the technical elements of the syntax, semantics, and model. These elements will include resource and temporal elements, suitable for reasoning about what an adversary has done in the past to a system of interest. I will elaborate the logic via a worked example of how it can be used to reason about an attack and suggest reasonable predicates. Finally, to speed along construction of rules for the logic, I will demonstrate how existing detailed attack knowledge, specifically Network Intrusion Detection and Prevention System (NIDPS) rules, can be represented in the logic and integrated into analysis via attack models.

7

LOGIC DEFINITION[1]

The previous chapters have drawn out threads from multiple disciplines: the status of Computer Security Incident Response (CSIR) standards, the debate on the scientific status of security, how scientists and security practitioners build general knowledge, examples of reasoning in incident response, and a history of a successful logic as a scalable reasoning tool. Although these topics are diverse, and have intrinsic value as pursuits in-themselves, they inform the decisions on building a logic to capture reasoning in CSIR analysis.

This chapter proceeds as follows. Section 7.1 will distil the preceding philosophical and historical analyses to the key points that need to inform the design of a logic for incident analysis. Section 7.2 will review some of the design choices that did not work or that I deemed inadequate. Section 7.3 will define the body of my CSIR logic – its syntax, model, and semantics. Section 7.4 will elaborate this logic and demonstrate how it can be used for abduction of attack details through an expression of the kill chain in the logic. Finally, Section 7.5 makes some benefits and potential impacts of using this CSIR logic, before Section 7.6 finishes.

7.1 PHILOSOPHICAL AND HISTORICAL ANALYSES

I will summarize the contributions to constraints on the logic from each of the prior chapters individually. The Chapter 2 constrains the question. The next three chapters constrain what it is the logical tool needs to do to answer that question. Chapter 6 constrains the logical tools with which to build a solution.

1 This chapter is based on joint work, the paper: Jonathan M Spring and David Pym (31st Oct. 2018). 'Towards Scientific Incident Response'. In: *GameSec*. LNCS 11199. Seattle, WA: Springer.

Incident response literature review

Chapter 2 mapped out the gap I want to fill. Specifically, the research question is: "how to satisfactorily make clear and explicit the reasoning process used by individual CSIR analysts while they pursue technical details?" The philosophical contributions and analysis in Chapters 3, 4, and 5 have already narrowed this gap. Two related gaps that remain are a distinct lack of any sort of formally-precise description of reasoning in incident analysis and any sort of automatable decision support. This remaining gap is the motivation for designing a logic, in order to provide a language for reasoning in which analysts could be more precise and more explicit.

Science and security literature review

Chapter 3 placed cybersecurity squarely within a modern understanding of scientific practice. Of course, what happened in a particular instance in the past, the topic for incident response, is a matter of forensics or of historiography. But criminal forensics is generally understood as using science in the service of the law. And historiography, and especially historical sciences like palaeontology and physical anthropology, very much use general knowledge and findings from present scientific research to inform and constrain historical accounts.

Similarly, the type of reasoning in incident analysis is what can be broadly understood as scientific reasoning. Of course, it will have to be adapted to the specific challenges and interests of incident analysis. However, conforming the basic shape and norms of broadly scientific reasoning still provides useful constraints and guidelines. In general, this means the logic must support learning from structured observations of the empirical world, evaluation of evidence, and intelligible explanations of phenomena. The intellectual system the logic is a part of should support integrative pluralism, translation of terms, and integration with engineering needs – this requirement suggests the logic should avoid dogmatic definitions of security phenomena. Several of these constraints depend on an adequate representation and use of generalized knowledge, which was not readily available in the literature.

Building generalized knowledge

Chapter 4 synthesized several different philosophical accounts of modelling and knowledge to fill this gap by constructing a new account of generalized knowledge in cybersecurity. The method of building general knowledge is important in its own right; however, the structure of general knowledge also provided some heuristics for reasoning with it. For example, Darden (2006) presented heuristics such as backward-chaining and instantiating entities based on the structure of mechanistic explanation. We have enriched that structure to identify generalized knowledge as painstakingly built up from clusters of multi-field mechanism schema on the dimensions of entities & activities, phenomena, organization, and etiology. The design of the logic will need to be able to interface with all the parts of this structure in order to make use of the general knowledge practitioners have built. I will demonstrate how such general knowledge might be presented within the logic by encoding one example discussed in Chapter 4, the intrusion kill chain, and representing it with the CSIR logic.

Examples of incident analysis

Chapter 5 made use of examples to connect scientific models and scientific reasoning to incident analysis. First, I showed how to improve the kill chain by considering it as a mechanistic explanation of intrusions and looking for gaps. Thus, not only is the structure of knowledge put forward by Chapter 4 compatible with incident analysis, but the heuristics (in this case, forward- and backward-chaining from Darden (2006)) provide an immediate benefit.

The other examples used in Chapter 5 were of the analytic process, rather than stable models. Stoll (1989) provides a clear case of good incident analysis that makes use of discovery heuristics from science. Just as clearly, this convergence was largely an accident of the analyst having trained as a physicist. The logic I intend to build should allow an analyst to capture their reasoning processes that mitigate the typical challenges of incident analysis (as described in Chapter 4). Another clear feature of both Stoll and the case of ARMOR, the second example process, is that the logic will need to support iterative improvement of hypotheses and explanations.

History of Separation Logic

Chapter 6 demonstrated that Separation Logic is able to capture important aspects of decision support for software engineering in the way it automates abduction. There are other, more common, approaches to decision support in cybersecurity – largely, machine learning. The choice of building a logic to fill the remaining gap may be non-obvious. However, as Pearl (2016) argues, machine learning approaches cannot, in principle, answer "why" or "what if" questions. The mathematical formalisms of current machine learning approaches do not permit such reasoning. In order to fix a real-life system, an incident analyst needs to know why it was compromised; that is, they need an explanation. Testing whether a proposed fix will be robust given the known method of compromise is a "what if" question. Pearl (2016) suggests his own causal reasoning tools as an answer. However, his tools have not demonstrated the other feature Chapter 6 identifies as necessary: operational deployability.

Additionally, Chapter 6 recounted deployed, scalable implementations of Separation Logic. The computational properties of Pearl's causal reasoning are much less optimistic (Chockler and Halpern, 2004), and it lacks a sufficiently concrete interpretation (Dawid, 2010). So I narrowed down the goal from designing a logic to fill the gap identified in Chapter 2 to designing a Separation Logic.

7.1.1 *Summary of requirements*

The logic design goals are, in no particular order, summarized as follows:

- Fill the gap in incident response by describing and (perhaps) supporting the decision process of incident analysis.
- Use Separation Logic as a scalable and deployable framework.
- Support hypothesis generation and testing (i.e., abduction).
- Support scientific heuristics and norms – evaluation of evidence, intelligible explanations, and translation among pluralistic fields.
- Support generalized knowledge, conceptualised as clusters of multi-field mechanism schema on four dimensions.
- Capture traditional instances of attack models (I will use the kill chain as an example).

- Enable sufficiently precise expression of reasoning so that analysts can exchange information about how to overcome the challenges typical of cybersecurity and incident response.
- Capture iterative improvement in the analyst's model of an incident.

7.2 LOGIC DESIGN CHOICES

This section will briefly explain the key logic design choices as well as what other choices were considered but deemed unsatisfactory. There are a suite of decisions related to how the temporal and spatial aspects of the logic interact that focused the work.

In an early effort I attempted a epistemic perspective that captured the investigator's beliefs and information directly. However, although my goal is to help the investigator, epistemic logic or other related modal logics tend to be about modelling communication among agents. I focus here is on a single agent's process in order to simplify the situation. With that decision, agent-based modal logic just over-complicates the model. The lone investigator implicitly knows everything captured by the history and the deductions and abductions based on it.

Related to this was a data-based or information-based approach. Roughly inspired by Floridi (2011) and guided by Floridi and Illari (2014), I considered a logic of how the investigator's model of the incident could or should change. This approach would have required wading into a debate on what data and information are, which is not the core concern in CSIR. This thread of thought led to useful questions about how information relates to both scientific and logical models. However, just as the investigator is implicit in the logic as designed, so too is evidence- or information-based learning. However, this is all implicit in how the history monoid (i.e., the logic's model) is changed during the course of an incident to reflect data gathered by the investigator prompted by abductive heuristics.

Thus, the most important early choice was to have the logical model represent information systems, not information. This choice follows how Shirey (2007) define information security (recall I defined cybersecurity as a superset of information security in Chapter 1). Information has proved notoriously difficult to define (Zins, 2007) and, in any case, I judge the logic of information by Floridi (2011) is not designed to be pragmatic or scalable. Therefore I have

avoided any attempt to model information, and instead focused on modelling the information systems (as defined by Shirey (2007)).

Another, later, critical decision is using a linear temporal logic model. A desirable feature seems to flow from this decision, namely the ability to interleave temporal and spatial operators in formulae arbitrarily. Prior work, such as Espinosa and Brotherston (2017), which all seems to use Concurrent Time Logic (CTL), was not able to achieve this interleaving. With linear time, a sensible separation of the history into two disjoint parts is possible. A few other decisions about handling time and separation are important to ensure sensibility; namely, the separation of the history must be at every time, from the start to end of the history. I also considered only separating from "now" forward, but because formulae might depend on earlier points in time, that decision fails. This dependence should not be due to including past-tense operators; as Manna and Pnueli (1992) note, including past-tense operators does not make a logic more expressive. With branching or concurrent time, there does not seem to be an appropriate way to handle how to define a separation into two disjoint histories that also respects branches.

Although the temporal and spatial operators can interleave, other relationships are not likely well defined. For example, $(\phi_1 * \phi_2)\mathcal{U}\phi_3$ and $(\phi_1\mathcal{U}\phi_3) * (\phi_2\mathcal{U}\phi_3)$ are not equivalent. The difference is synchronicity. In the former, ϕ_1 and ϕ_2 must hold simultaneously until ϕ_3. In the latter, the time at which they give way to ϕ_3 may be different. I cannot think of any way to enforce preservation of this information between the two formulae, and so accept this as a limitation of the logic as defined.

I also considered defining a process-based temporal transformation, based on Milner (1989). The idea would have been to define what possible reads and writes could have occurred within the system, and then reason with that as the model instead of the history monoid defined in this chapter. Although this might have the benefit of being more explicit about system transformations, it has two major drawbacks. First, acquiring that level of detail about system processes is not generally practical for an incident responder. Information generally must be collected prior to the incident, and fine-grained traces of every system are just too voluminous. Second, it would require a notion of absolute or global time, because transitions need to be tightly sequenced. As mentioned, I follow Lamport (1983) in the modelling choice that a 'next' operator, or tight sequencing, is unwise to build into a temporal logic. In logics where the goal is to prove system properties, such conditions on 'next' are generally what

needs to be proven. However, here is an important departure point for my CSIR logic – I am not trying to prove the properties of a program. In incident response, one just needs to know whether key system states or events were reached. Therefore, a process-based or transition system is inappropriately detailed. This realization also guides the use of Hoare triples not to prove the pre- or post-conditions of a software element $\mathcal{C}$, but rather to reason about the relevant impacts of these conditions.

Other paths are not reflected in the final product at all. For example, I briefly considered a defeasible separation logic. In some sense, a defeasible logic is intuitive because conclusions in incident response are tentative and subject to revision. However, embedding defeasibility into the valuation of sentences seemed needlessly complex. This intuition is adequately captured by the use of bi-abduction for generating heuristic explanations. Given a novel combination of temporal operators with a separation logic, I wanted to use reliable or well-understood logical tools where possible.

7.3 LOGIC DEFINITIONS

This section builds a logical system as a tool for expressing paradigmatic features of incident analysis. Section 7.4 will use these logical tools to further elaborate the logic through an example.

A necessary part of a logical system is its model. A model, in this logic sense, is a mathematical structure with which one can interpret a proposition, and then determine whether it is satisfied or not. This sense is quite far from a scientific model. However, as Chapter 6 argues, an effective logic will align its logic model with the salient features of a scientific model of the represented phenomenon. Therefore, I develop logical tools with the purpose of incident analysis in mind at every step. The phenomena of interest are violations of security policy; that is, a resultant state of a computer system. The logical model will represent these as histories, where a history is a series of states of the computer.

I make a variety of choices to adapt the logic to incident analysis. Some are simple: incident analysis is largely about past events, so I include both past-tense and future-tense temporal operators. Others are more subtle. For example, the model has a separation of network, storage, and processor resources at a basic level because practitioners think about, monitor, and defend these things quite differently. I want the logic to reflect this reality. Some choices

have an eye towards pragmatics of usability and deployable decision-making. As O'Hearn (2015) describes, the road from formal logic to operational implementation is long. However, I include the 'and, separately' operator to support composable reasoning and keep an eye towards scalability.

7.3.1 *Expressions*

The definition of expressions is essentially the same as Ishtiaq and O'Hearn (2001) and Calcagno et al. (2011). An expression can be an integer, an atom, or a variable.

$$
\begin{array}{rcll}
E & ::= & x & \text{Variable} \\
& | & 37 & \text{Integer} \\
& | & \texttt{nil} & \text{nil} \\
& | & \texttt{a} & \text{atom} \\
& | & \ldots &
\end{array}
$$

The open-ended definition of expressions allows additional expressions so long as they can be interpreted in the semantic domain specified.

The semantic domains are values, addresses, and content; this domain is analogous to, and slightly more general than, the values, stacks, and heaps used in Calcagno et al. (2011):

$$
\begin{array}{rcl}
Val & = & Int \cup Atoms \cup Loc \\
A & = & Var \rightharpoonup_{fin} Val \\
C & = & Loc \rightharpoonup_{fin} Val \times Val
\end{array}
$$

where $Loc = \{\ell, \ldots\}$ is an infinite set of locations, $Var = \{x, y, \ldots\}$ is a set of variables, $Atoms = \{\texttt{nil}, \texttt{a}, \ldots\}$ is the set of atoms, and finite partial functions are represented by $\rightharpoonup_{fin}$. Elements of addresses and content are $a \in A$ and $c \in C$, respectively. As customary for stack variables, I do not provide an explicit operation for allocating address variables.

The domain of an element of addresses is $dom\,(a)$ for $a \in A$. Similarly, $dom\,(c)$ is the domain for an element of contents. Note that English grammar here may be confusing. An address a is a set of mappings from variables to values, not a singleton. Likewise, c is a set of content mappings, not a singleton.

Interpretation is independent of the particular computer represented, analogous to heap-independent interpretations in Calcagno et al. (2011): $[\![E]\!]\, a \in Val$, so long as $dom\,(a)$ includes the free variables of E.

7.3.2 *Basics and syntax*

I will make use of some familiar classical propositional connectives, some perhaps less-familiar temporal connectives, and a 'spatial' connective from a more recent logic – Separation Logic introduced in Chapter 6. The classical connectives are 'if, then', 'and', 'or', and 'not' and the first-order quantifiers are 'there exists' and 'for all'.

The operators 'until' and 'since' are both temporal, following the definition from Manna and Pnueli (1992). 'Until' is about the future, and 'since' is about the past, but otherwise they are similar. It is the case that 'ϕ until ψ' when the first formula ϕ is true now and into the future, for at least enough time such that the second formula becomes true at some time later. It is what one might expect when asking "Hold this cup until I get back." Though one would need to be explicit about the social assumption, in classical logic, of "If I return, then give me the cup." 'Since' is similar. It is the case that 'ϕ since ψ' when at some point in the past ψ occurred, and ϕ has been occurring from then up through to the present. As one might expect from "I have been sad since my cup broke."

The final connective is $*$ for 'and, separately'. Recall from Chapter 6 that classical 'and' is collapsible. That is, "I have five coins and I have five coins" is, in classical logic, the same as "I have five coins." The connective 'and, separately' is not collapsible.

Computers, like coins, are resources. I use Separation Logic to be able to express "A computer is compromised and, separately, a computer is compromised" with the natural meaning, for example. The classical 'and' would lose this information that two computers are compromised, because the formula would collapse.

Following these intuitions, construct logical formulae inductively:

ϕ, ψ	$::=$	α	Atomic formulae
	$\mid$	$\bot$	Falsity
	$\mid$	$\phi \Rightarrow \psi$	Material implication
	$\mid$	$\texttt{emp}$	Empty content
	$\mid$	$\exists x.\phi$	Existential quantification
	$\mid$	$\phi \mathcal{U} \psi$	Temporal Until
	$\mid$	$\phi \mathcal{S} \psi$	Temporal Since
	$\mid$	$\phi * \psi$	Spatial conjunction

Atomic formulae include equality, points-to relations, and predicates.

α	$:=$	$E = E'$	Equality
	$\mid$	$E \mapsto E_1, E_2$	Points to
	$\mid$	$P\left(\left(Val_1, E_1\right), \left(Val_2, E_2\right)\right)$	Relational predicate
	$\mid$	$\ldots$	

In Ishtiaq and O'Hearn (2001), points-to is defined as a three-place relation, $E \mapsto E_1, E_2$. Calcagno et al. (2011) contains both a simple points-to relation, $E \mapsto E'$ and a higher-order concept of lists that treats the properties of lists as primary, rather than their contents. My goal is not to analyse details of doubly-linked lists or higher-order lists. Therefore, the syntax does not treat lists directly. However, this three-place syntax provides a way to separate a large data element into arbitrary chunks while preserving ordering. This works for memory, files on disk, and network packets. This is useful because, for example, it can represent malware analysis techniques, such as segment hashing, by representing properties of a connected series of expressions. However, the intention is not to be exhaustively faithful to the file-system representation. If the segments of a large file are not of interest, then elide the details of the file system block size and the linked list that actually composes the file contents.

The usual classical and temporal symbols are defined from available formulae:

- negation; i.e., 'not', is $\neg\phi \stackrel{\text{def}}{=} \phi \Rightarrow \bot$
- truth is simply not false; i.e., $\top \stackrel{\text{def}}{=} \neg\bot$

- disjunction; i.e., 'or' is customarily $\phi \vee \psi \stackrel{\text{def}}{=} (\neg\phi) \Rightarrow \psi$
- conjunction; i.e., 'and' is thus $\phi \wedge \psi \stackrel{\text{def}}{=} \neg\,(\neg\phi \vee \neg\psi)$
- 'for all' is in terms of the existential, $\forall x.\phi \stackrel{\text{def}}{=} \neg\exists x.\neg\phi$
- 'at least once in the future' relates to until, $\Diamond\,\phi \stackrel{\text{def}}{=} \top\mathcal{U}\phi$
- 'henceforth' is $\Box\,\phi \stackrel{\text{def}}{=} \neg\,\Diamond\,\neg\phi$
- analogously, 'at least once in the past' is $\Diamond\!\!\!-\,\phi \stackrel{\text{def}}{=} \top\mathcal{S}\phi$
- and 'has always been' is $\boxminus\phi \stackrel{\text{def}}{=} \neg\,\Diamond\!\!\!-\,\neg\phi$.

Following Lamport (1983), the definition does not have a simple 'next' temporal operator. For various reasons Lamport (1983) lays out, and a choice that is validated by how incident analysts reason in the case studies of prior chapters, the logic should care about observable changes, not the precise sequence that brings those changes about.

7·3·3 *Model*

My model is designed to support incident response reasoning by embedding the most important objects of analysis as the basis of the model. The model keeps the three salient types of computing resources separate, and the series of resources is indexed by time as a history. Each resource is a partial monoid with composition operator and unit.

$(R_M, \cdot_M, e_M)$ for processor and RAM (M for memory)
$(R_D, \cdot_D, e_D)$ for file storage (D for disk)
$(R_N, \cdot_N, e_N)$ for network bandwidth (N for network)

where, for $i \in \{M, D, N\}$, R_i is a set of resource elements of the given type, $\cdot_i : R_i \times R_i \rightharpoonup R_i$ is a partial function operating on resources of the given type, and e_i is the unit element of $\cdot_i$ such that for all $r \in R_i$ it is the case that $r \cdot_i e_i = r = e_i \cdot_i r$.

More concretely, each R_M, R_D, R_N is composed of (address, content) pairs analogous to (stack, heap) pairs. I define $\mathbf{m} ::= s, h$ for $\mathbf{m} \in R_M$, $\mathbf{d} ::= \delta, \beta$ for $\mathbf{d} \in R_D$, and $\mathbf{n} ::= \kappa, \upsilon$ for $\mathbf{n} \in R_N$. These sub-parts of the resources are proper subsets of the address and content defined above. The fact that $s \in \mathsf{S}$ with $\mathsf{S} \subset A$ and $h \in \mathsf{H}$ with $\mathsf{H} \subset C$ makes the usual stack-heap model of Separation Logic somehow contained in the address-content model. Further, define $\delta \in \mathsf{N}$ for $\mathsf{N} \subset A$ and $\beta \in \mathsf{B}$ for $\mathsf{B} \subset C$ (for inodes and file blocks). For network host addresses and data units (i.e., packets), $\kappa \in \mathsf{K}$ for $\mathsf{K} \subset A$ and $\upsilon \in \mathsf{U}$ for $\mathsf{U} \subset C$.

Formally, these three resource monoids could be considered as one monoid $\mathbf{R} = (R, \cdot, \mathbf{E})$ where $R = R_M \uplus R_D \uplus R_N$ (the disjoint union of the resources), composition $\cdot, \cdot : \mathbf{R} \times \mathbf{R} \to \mathbf{R}$ such that

$$\cdot (r_1, r_2) ::= \begin{cases} r_1 \cdot_i r_2 & \text{if } r_1, r_2 \in R_i \\ \text{undefined} & \text{otherwise} \end{cases}$$

and $\mathbf{E} = \{e_M, e_D, e_N\}$

$$\text{where } \mathbf{E} \cdot r ::= \begin{cases} \bigcup_{e \in \mathbf{E}} r \cdot_i e = \{r\} = \bigcup_{e \in \mathbf{E}} e \cdot_i r & \text{if } r \in R_i \\ \text{undefined} & \text{otherwise} \end{cases}$$

The definitions of $\cdot$ and a set of units are adapted from Brotherston and Villard (2015, def 2.3). These definitions will be used to describe the state of a computer or a computer network at a given time as a composition of different programs, files, and network activity.

Incident analysis needs a notion of time and changes. Therefore, I adopt a linear time model composed of a sequence of states. Each state is represented by an element $r \in \mathbf{R}$. Define a history $H \in \mathcal{H}$ as a ordered finite set

$$H ::= \left\{ r^1, r^2, \ldots, r^t, \ldots, r^T \right\},$$

with $T \in \mathbb{N}$. The pair (H, t) uniquely identifies the state $r^t \in \mathbf{R}$. The length of a history is $|H| = T$. There is no notion of absolute time or a "wall clock." The time T indicates a sequence of states without any claims about the time between transitions.

HISTORY MONOID Define a monoid, $\mathbf{H} = (\mathcal{H}(\mathbf{R}), \circ, e)$ where $\mathcal{H}$ is the set of histories H (defined above) that can be constructed using a given resource monoid $\mathbf{R}$; $\circ : \mathcal{H} \times \mathcal{H} \rightharpoonup \mathcal{H}$; unit e to be the empty history with $|e| = 0$.

More specifically, define $\circ$ as:

$$(H_1 \circ H_2, t) ::= \begin{cases} (r_1^t \cdot r_2^t, t) \text{ for } r_1^t \in H_1 \text{ and } r_2^t \in H_2 & \text{if } |H_1| = |H_2| \\ (H_2, t) & \text{if } H_1 \prec H_2 \\ (H_1, t) & \text{if } H_2 \prec H_1 \\ \text{undefined} & \textit{otherwise} \end{cases}$$

Here $H_1 \prec H_2$ indicates that one history is contained in the other. I define four conditions that must all be met for this to hold. Specifically, $H_1 \prec H_2$ iff

1. $|H_1| < |H_2|$, where $|H_1| = T$, $|H_2| = T'$; and
2. for all $r_1^t \in H_1$, with $t \in T$, there exists some $r_2^{t'} \in H_2$ with $t' \in T'$ such that $r_1^t = r_2^{t'}$ and $t \leq t'$; and
3. for all $r_2^{t'} \in H_2$ and given any r_1^t, r_1^x in H_1 with $t, x \in T$, it is the case that $r_2^{t'} = r_1^t$ and $r_2^{t'} = r_1^x$ iff $t = x$; and
4. for all r_1^t, r_1^x in H_1 with $t, x \in T$ such that $t < x$, it is the case that, for $r_2^{t'}, r_2^{x'} \in H_2$ with $t', x' \in T'$, we have $r_1^t = r_2^{t'}$ and $r_1^x = r_2^{x'}$ iff $t' < x'$.

The intuition for these requirements as expressing the concept of "contained in" is as follows. A smaller history is contained in a larger one. All the events of the smaller history appear in the larger one, in the same relative ordering. The only change permitted is that new events are inserted into the larger history; such inserted events can be interleaved in any way.

The unit e as the empty history behaves as expected.

$$H \circ e = H = e \circ H$$

To demonstrate this is true, consider a proof of identity by cases. It is given that $|e| = 0$, so either

1. $|H| = 0$, that is H is e, thus need to prove $e \circ e = e$

    a) This is true, following $r^t \coloneqq r_1^t \cdot r_2^t$. However, $T = 0$ so there are no elements to compose. The result is the history of length 0, namely, e.

2. $|H| \geq 1$

    a) Requirement 1 for $\prec$ holds $(0 < 1)$.
    b) Requirement 2 holds vacuously (all $r_1^t \in e$ is $\emptyset$).
    c) Requirement 3 holds vacuously, without r_1^t, r_1^x to compare.
    d) Requirement 4 holds vacuously, without r_1^t, r_1^x to compare.

One might think the unit for $\circ$ could be the history of length 1, containing just the unit element $\mathbf{E}$ (recall $\mathbf{E} = \{e_M, e_D, e_N\}$). However, if defined thus, requirement 2 for $\prec$ might fail if there is no element of H in $H \circ e$ such that $(H, t) = \mathbf{E}$. Then $H \circ e$ could be undefined for $|H| > 1$, in which case $H \circ e = H = e \circ H$ would not hold as required. Every history could start with the unit element to make this true by construction, but that seems unnatural. Therefore the unit of $\circ$ should be the empty history $|e| = 0$.

Briefly, the intuition behind this definition is that a history will represent a hypothesis for the series of events and changes to the resources of a computer system during the course of the incident. Combining histories needs to be defined in order for $*$ to behave properly when the hypothesis includes two parts that happen separately. By adapting proof rules from program verification and the style of biabduction introduced in Chapter 6, an analyst should be able to formally check whether a hypothetical model supports the conclusion that an incident happened. If the model does not, then either more evidence is needed (adding or changing states in the history H) or the representation of CSIR reasoning heuristics is inadequate. The next step in enabling this reasoning chain is defining semantics and satisfaction relations.

7·3·4 *Semantics*

The semantics of the atomic expressions are many-sorted operations. To unfold the truth value of an expression, recall
$(H,t) \stackrel{\text{def}}{=} [(s^t,h^t),(\delta^t,\beta^t),(\kappa^t,\upsilon^t)]$. Therefore we have

$$[(s^t,h^t),(\delta^t,\beta^t),(\kappa^t,\upsilon^t)] \models E = E' \quad \text{iff} \quad \begin{cases} [\![E]\!]\, s^t = [\![E']\!]\, s^t \\ [\![E]\!]\, \delta^t = [\![E']\!]\, \delta^t \\ [\![E]\!]\, \kappa^t = [\![E']\!]\, \kappa^t \end{cases}$$

Abbreviate this as

$$H,t \models E = E' \quad \text{iff} \quad [\![E]\!]\, a^t = [\![E']\!]\, a^t$$

Because these three types of resource are disjoint (namely $\mathsf{S} \subset A$; $\mathsf{N} \subset A$; $\mathsf{K} \subset A$ and $\mathsf{S} \cap \mathsf{N} = \mathsf{S} \cap \mathsf{K} = \mathsf{N} \cap \mathsf{K} = \emptyset$), only one of the three interpretations can be valid. Namely, only one of $[\![E]\!]\, s$ or $[\![E]\!]\, \delta$ or $[\![E]\!]\, \kappa$ can hold for any E, or they are equivalent. Only one exists because for $[\![E]\!]\, a$ to be interpretable, $dom\,(a)$ must include the free variables of E. The domains of s, δ, κ are disjoint by definition. If there are no free variables in E, then $[\![E]\!]\, s = [\![E]\!]\, \delta = [\![E]\!]\, \kappa$.

Similarly, points-to can be defined over the three disjoint parts of the model at a given time, and then abbreviated in terms of elements of A and C:

$$
\begin{aligned}
&\left[\left(s^t, h^t\right), \left(\delta^t, \beta^t\right), \left(\kappa^t, \upsilon^t\right)\right] \models E \mapsto E_1, E_2 \\
&\quad \text{iff} \begin{cases}
h^t\left(\llbracket E \rrbracket\, s^t\right) = \langle \llbracket E_1 \rrbracket\, s^t, \llbracket E_2 \rrbracket\, s^t \rangle & \left\{\llbracket E \rrbracket\, s^t\right\} = dom\left(h^t\right) \\
\beta^t\left(\llbracket E \rrbracket\, \delta^t\right) = \langle \llbracket E_1 \rrbracket\, \delta^t, \llbracket E_2 \rrbracket\, \delta^t \rangle & \left\{\llbracket E \rrbracket\, \delta^t\right\} = dom\left(\beta^t\right) \\
\upsilon^t\left(\llbracket E \rrbracket\, \kappa^t\right) = \langle \llbracket E_1 \rrbracket\, \kappa^t, \llbracket E_2 \rrbracket\, \kappa^t \rangle & \left\{\llbracket E \rrbracket\, \kappa^t\right\} = dom\left(\upsilon^t\right)
\end{cases}
\end{aligned}
$$

Abbreviate this as

$$
\begin{aligned}
&H, t \models E \mapsto E_1, E_2 \quad \text{iff} \quad \left\{\llbracket E \rrbracket\, a^t\right\} = dom\left(c^t\right) \\
&\quad \text{and} \quad c^t\left(\llbracket E \rrbracket\, a^t\right) = \langle \llbracket E_1 \rrbracket\, a^t, \llbracket E_2 \rrbracket\, a^t \rangle
\end{aligned}
$$

The element `emp` actually represents a set of three related elements: $\left\{\overset{M}{\texttt{emp}}, \overset{D}{\texttt{emp}}, \overset{N}{\texttt{emp}}\right\}$. The semantics for `emp` is defined as

$$
\begin{aligned}
\left[\left(s^t, h^t\right), \left(\delta^t, \beta^t\right), \left(\kappa^t, \upsilon^t\right)\right] &\models \overset{M}{\texttt{emp}} \quad \text{iff} \quad h^t = [] \\
\left[\left(s^t, h^t\right), \left(\delta^t, \beta^t\right), \left(\kappa^t, \upsilon^t\right)\right] &\models \overset{D}{\texttt{emp}} \quad \text{iff} \quad \beta^t = [] \\
\left[\left(s^t, h^t\right), \left(\delta^t, \beta^t\right), \left(\kappa^t, \upsilon^t\right)\right] &\models \overset{N}{\texttt{emp}} \quad \text{iff} \quad \upsilon^t = [] \\
H, t &\models \texttt{emp} \quad \text{iff} \quad \overset{M}{\texttt{emp}} \text{ and } \overset{D}{\texttt{emp}} \text{ and } \overset{N}{\texttt{emp}}
\end{aligned}
$$

Here, $h^t = []$, $\beta^t = []$, and $\upsilon^t = []$, represent the empty heap, empty file system, and empty network, respectively.

The semantics for a relational predicate, P, is given by

$$
\begin{aligned}
&H, t \models P\left(\left(Val_1, E_1\right), \left(Val_2, E_2\right)\right) \\
&\quad \text{iff} (H, t) \in \mathbb{V}\left[P\left(\left(Val_1, E_1\right), \left(Val_2, E_2\right)\right)\right]
\end{aligned}
$$

Here $\mathbb{V} : \mathbb{A} \rightarrow \mathcal{P}\left(States\right)$ is the valuation function from the set $\mathbb{A}$ of atoms of $P\left(\left(Val_1, E_1\right), \left(Val_2, E_2\right)\right)$ to the powerset of possible states of the form (H, t).

The other semantic clauses are as follows:

$$
\begin{array}{lll}
H,t \models \phi \Rightarrow \psi & \text{iff} & \text{if } H,t \models \phi \text{ then } H,t \models \psi \\
H,t \models \exists x.\phi & \text{iff} & \text{for some } v \in Val.\, [a|x \mapsto v], c \models \phi \\
H,t \models \phi \mathcal{U} \psi & \text{iff} & \text{for some } i \in T \text{ with } i \geq t \text{ and } (H,i) \models \psi \text{ such that} \\
 & & \text{for all } j \in T \text{ with } t \leq j < i \text{ it is the case } (H,j) \models \phi \\
H,t \models \phi \mathcal{S} \psi & \text{iff} & \text{for some } i \in T \text{ with } i \leq t \text{ and } (H,i) \models \psi \text{ such that} \\
 & & \text{for all } j \in T \text{ with } i < j \leq t \text{ it is the case } (H,j) \models \phi \\
H,t \models \phi * \psi & \text{iff} & \text{for some } H_1, H_2 \text{ such that } H_1 \# H_2 \text{ and} \\
 & & H_1 \circ H_2 = H, \text{ where } H_1,t \models \phi \text{ and } H_2,t \models \psi
\end{array}
$$

Here $H_1 \# H_2$ indicates the histories are point-wise disjoint. $H_1 \# H_2$ is the case if and only if the following conditions hold:

1. $|H_1| = |H_2| = T$; and
2. For all $[(s_1^t, h_1^t), (\delta_1^t, \beta_1^t), (\kappa_1^t, \upsilon_1^t)] \in H_1$ and $[(s_2^t, h_2^t), (\delta_2^t, \beta_2^t), (\kappa_2^t, \upsilon_2^t)] \in H_2$ it is the case that, for all $t \in T$:
   a) $dom\,(h_1^t) \cap dom\,(h_2^t) = \emptyset$ and
   b) $dom\,(\beta_1^t) \cap dom\,(\beta_2^t) = \emptyset$ and
   c) $dom\,(\upsilon_1^t) \cap dom\,(\upsilon_2^t) = \emptyset$.

These semantics are novel in that they allow temporal and spatial operators to interleave freely. Prior attempts to combine spatial and temporal operators used CTL with its parallel branching time semantics (Espinosa and Brotherston, 2017). These attempts did not achieve the ability to interleave operators I have achieved here. Otherwise, these semantics are closely related to how separation logics have usually been defined, as Chapter 6 reviewed, and so should readily support reasoning with abduction and deduction.

7.3.5 *Abduction*

As introduced in Chapter 6, this logic model can assist in automating abduction. Recall that Charles Peirce defines it as:

> "Abduction is the process of forming an explanatory hypothesis.
> It is the only logical operation which introduces any new idea."
> (Bergman and Paavola, 2016, CP 5.171)

Abduction would naturally be grouped into a trio with deduction and induction. These terms have long, problematic histories of usage. Deduction requires a proof theory, and because one can justifiably define different proof theories for different purposes (von Plato, 2016), 'deduction' is not just one thing. But generally 'deduction' captures the reasoning from premises to conclusions following explicit rules. 'Induction' has received voluminous attention, since Hume in the 1740s (Henderson, 2018). It roughly means concluding that because something has been the case before, it will be again. A more fruitful discussion might be had under the topic of how to generalize from what is known. Generalization methods and general knowledge, as defined in Chapter 4, are one example of the heuristics that should be encoded into the logic.

Abduction is neither deduction nor induction. Abduction is the generation of an explanation, which can then be evaluated against available evidence.More formally, abduction asks what (minimal) formula needs to be added to a proposition such that it will be satisfied. As Chapter 6 discussed, abduction is automatable as long as the problem space is constrained, checking the validity of hypothetical additions is scalable, and human heuristics for generating additions can be encoded in the logic. Attack ontologies (those identified in Chapter 2) will serve as the heuristics for incident analysis.

Section 7.4 will endeavour to represent the intrusion kill chain (Hutchins et al., 2011) in the logic. That section will also demonstrate how to link existing knowledge bases, such as Snort rules, into this structure. This should allow a system to be built that instruments a computer network, ingests security-relevant information, and, given a security incident, uses the logic to assist in the process of abducing explanations of how an adversary penetrated the network. Given this decision support, the natural future work is to test and improve different abduction rules in a scientific manner, recalling the broad scientific status of CSIR as argued for in Chapter 3. Thus I do not claim the kill chain as represented is the heuristic, but just a starting point for study and improvement.

7.3.6 *On the metatheory of the security incident analysis logic*

Generally, when setting up and explaining a system of logic, one gives a language (of propositions) and a semantics specified by a class of models together with a satisfaction relation which specifies which propositions are true in which

parts of an arbitrary model. Typically, one also gives a proof system – that is, a collection of inference rules – which determines which propositions are provable. The first meta-theoretic challenge is then to establish soundness and completeness. Soundness means that the provable things are also true in the models. Completeness means that there is a model for which the notion of truth specified in the semantics coincides with the notion of provability specified by the inference rules. This, together with other metatheoretic analyses, is what assures logicians that a logic makes good sense.

This section has described a logic for analysing security incidents. I have defined the logic by giving its propositional language together with a semantics given by a specific model together with a satisfaction relation which determines which propositions are true in which parts of the given model.

The defined logic is based on a combination of some well-understood constructions together with a specific concrete model. In this respect, its definition resembles that of Separation Logic (see Chapter 6) enough that these important methatheoretical properties, such as soundness and completeness, should be the same in both logics. Since Separation Logic is known to be sound and complete, this is good evidence my logic is on solid footing.

7.4 A WORKED EXAMPLE

In this section, I return to the 'kill chain' model as an example. As situated in Chapter 2 and touched on in Chapter 4, the kill chain model was introduced by Lockheed Martin to explain an abstract pattern they observed in attacks targeting their organization (Hutchins et al., 2011). As elaborated in Chapter 5, it is a useful model of computer network attacks because it helps inform the incident analyst about expected sorts of activities, targeting which entities, and how the attack elements are organised so as to be responsible for the outcome. The abstract nature of the kill chain makes it a good example to be expressed in this incident analysis logic. It also models a useful unit of incident analysis: a single attack. Multiple attacks are almost always sequenced to achieve an adversary's overall goal during a campaign. Also, most attacks do not succeed, so usually many attacks occur in parallel. Therefore, modelling a single attack should be a fruitful example because analysts can compositionally build on it to analyse security incidents.

The goal is to turn this conceptual model of the kill chain into a set of logical statements of pre- and post-conditions that express useful abduction

heuristics. This translation requires a realignment of the components of the model. The logic talks about observable computer activity, and as such the humans in the kill chain have no explicit representation. Their interests are represented in the definition of predicates. For example, the truth values of *compromised*() will depend on the security policy of the defender.

What counts as malware or an exploit is also dependent on the point of view of the agents. The logic models only software instructions, computer systems, and bit-strings. These categories are intention-neutral. Malware, like all software, is simply an instruction set – what the logic will model as a *computation*. I shall not dictate how malware is classified. One benefit of the logic is to express precisely how an analyst determines how to differentiate malware from benign computations. Descriptions of what behaviours are indicative of malicious versions of those elements will be contingent.

To define the representation of a computation (i.e., software, functions, etc.), I adapt Hoare-Floyd logic. Hoare logic is a mainstay of program verification. Recall from Chapter 6 that Hoare logic is primarily concerned with statements of the form $\{\phi\}\,\mathcal{C}\,\{\psi\}$, where ϕ is pre-conditions, ψ is post-conditions, and $\mathcal{C}$ is some specific computation. The goal of Hoare logic is to verify that ψ can be guaranteed to be satisfied if $\mathcal{C}$ executes in an environment that satisfies ϕ.

The construction of Hoare logic is about the details of $\mathcal{C}$ and trying to demonstrate post-conditions given pre-conditions. I will use Hoare triples differently. The incident responder knows a post-condition, usually some security violation, and wants to understand more about the pre-conditions and software. The computation $\mathcal{C}$ can be described in various levels of detail. This is an important benefit. The logic, so defined, permits description of programmatic details. Malware reverse engineering tries to construct details of an unknown $\mathcal{C}$. Incident analysis is primarily involved in a higher level task, merely constraining the observable traces in the system, not how some $\mathcal{C}$ made these changes. Knowing malware details is helpful because it narrows the potential pre- and post-conditions. I have jointly made a start at how to describe a logic of malware in Primiero et al. (2018). However, I leave discussion of how specifically $\mathcal{C}$ works in malware for future work. What incident analysts need to know about malicious logic is simply a Hoare triple $\{\phi\}\,\mathcal{C}\,\{\psi\}$ where ϕ and ψ are known. This approach to knowledge is essentially the programming

principle of encapsulation. If one know what goes in and what comes out, how it works is irrelevant; only the impacts on the system matter.[2]

Let's represent a computer system as σ, taken from the systems known to the analyst. The full complement of systems is represented by $\mathbf{S}$. At a given time t, the system is σ^t. The system σ^t is shorthand for a cluster of resources $[(s^t, h^t), (\delta^t, \beta^t), (\kappa^t, \upsilon^t)]$. Therefore, at any given time t, the state of the world (H, t) might be decomposed into one or more systems $\sigma_1^t \cdot \sigma_2^t \cdot ... \cdot \sigma_n^t$. The concept of system is therefore merely a shorthand for a cluster of resources that the incident analyst is interested to treat as a unit of analysis.

The third and final entity, bit-strings are a type of expression E. Usually programmers represent strings in human-readable form. Human-readable strings can be represented as integers, so the syntax for E remains unchanged. I elide the details of local encodings (ASCII vs. unicode vs. hexadecimal, big- vs. little-endian, etc.) that complicate mapping between strings and integers. Notating strings as strings instead of expressions is merely a syntactic convenience.

Given computations and systems, all the necessary predicates can be defined as follows:

- $compromised\,(\sigma^t)$
- $hostile\,(\sigma^t)$
- $malicious\,(\mathcal{C})$
- $match\,(string_1, string_2)$
- $trusts\left(\sigma_1^t, \sigma_2^{t'}\right)$ (often $t = t'$)
- $vulnerable\,(\sigma^t, \mathcal{C})$
- $exploited\,(\sigma^t, \mathcal{C})$

Compromised, hostile, and malicious have the intuitive meanings. In the current set of definitions, these have binary truth values. Analysts may be interested in intermediate values. I leave an extension of the logical definitions to a many-valued logic as future work.

Note that my intention here is that the compromised system is internal, under defender ownership, whereas a hostile system is on the Internet, not owned by the defender. Therefore, a different reasonable definition would be to define an ownership predicate, and define *compromised* () in terms of hostile and owned. There are multiple compatible ways to represent relevant concepts. I select the above as a viable definition, not the only one.

2 Any given $\{\phi\}\,\mathcal{C}\,\{\psi\}$ for a program will be treated as a hypothesis, and one that given sufficient evidence might be overturned and modified.

The predicate $trusts\left(\sigma_1^t, \sigma_2^{t'}\right)$ is a relationship between two systems. Although it is an oversimplification, for the time the predicate reduces trust to the ability to communicate. More specifically, receive information. That is, given an address $a_1 \in A$ such that $a_1 \subset \sigma_1^t$ and address $a_2 \in A$ such that $a_2 \subset \sigma_2^t$ and any expression E, it is the case that $trusts\left(\sigma_1^t, \sigma_2^{t'}\right)$ just in case that $(a_1 \mapsto E) \Rightarrow \Diamond(a_2 \mapsto E)$. I adopt the convention that any similar predication over a σ must be similarly reducible to a logical formula.

This is an abstract concept of communication. It just says that if some address in system one points to an expression, somehow eventually an address in system two comes to point to the same expression. The reason this is trust, and not chance, is that this relationship holds for any expression. This definition abstracts away from how that communication is executed. A real security policy may restrict which expressions are permitted or disallowed. I leave such definitions of a trust predicate as future work.

The predicate $match\,()$ represents a common use case in incident analysis and computer network defence: pattern matching. Tools such as intrusion detection systems, firewall ACLs, and spam email detection all rely on matching incoming communications to known patterns of interest. These patterns are signatures or blacklists of malicious behaviour.

Let's define the semantics of $match\,(string_1, string_2)$ such that:

$$\left[\left(s^t, h^t\right), \left(\delta^t, \beta^t\right), \left(\kappa^t, \upsilon^t\right)\right] \models match\,(string_1, string_2)$$

just in case

$$in\left(\left[\left(s^t, h^t\right), \left(\delta^t, \beta^t\right), \left(\kappa^t, \upsilon^t\right)\right], string_2\right) \wedge string_1 = string_2$$

The $in\,()$ predicate holds just in case

$$\begin{aligned}
&[\![string_2]\!] \in dom\,(s^t) \quad \vee \quad [\![string_2]\!] \in dom\,(h^t) \quad \vee \\
&[\![string_2]\!] \in dom\,(\delta^t) \quad \vee \quad [\![string_2]\!] \in dom\,(\beta^t) \quad \vee \\
&[\![string_2]\!] \in dom\,(\kappa^t) \quad \vee \quad [\![string_2]\!] \in dom\,(\nu^t)
\end{aligned}$$

One may abbreviate this as $in\,(\sigma^t, string)$ or $in\,((H, t), string)$. If you wish to emphasize a certain type of string only occurs in the contents of files, for example, then elide the other variables and write $in\,(\beta^t, string)$.

The equality operator is expression equality as defined in Section 7·3·4, since strings are expressions. Specifically, if strings are understood as integers,

the expressions will have no free variables and so it becomes the usual integer equality.

Write $\sigma^t \models \{\phi\}\,\mathcal{C}\,\{\psi\}$ just in the case that there is some content $c \in C$ and $\sigma^t \models \mathit{match}\,(\mathcal{C}, c) \Rightarrow (\phi \Rightarrow \Diamond\psi)$. This assumes that the computation $\mathcal{C}$ terminates. But analysts are primarily concerned with malware that has successfully run, so this termination condition should not cause great trouble. Furthermore, time is defined to be finite, so termination can always be defined as the state at (H, t) when $t = T$.

I then propose to define $\mathit{vulnerable}\,(\sigma^t, \mathcal{C})$ to hold iff $\sigma^t \models (\{\phi\}\,\mathcal{C}\,\{\psi\}) \wedge \phi \wedge \mathit{malicious}\,(\mathcal{C})$. If $\mathit{vulnerable}\,(\sigma^t, \mathcal{C})$ holds, the real-world impact is a bad security situation. Such a system can be exploited at the will of the adversary.

To differentiate from the less severe situation where a system is vulnerable but exploit code is not present, define $\sigma^t \models \mathit{vul}\,(\phi)$. This is a syntactic convenience; it means only that $\sigma^t \models \phi$ and that ϕ is the precondition for the execution of some malware.

Vulnerability is not the same as exploitation (in the traditional terminology of computer security). Exploit also requires access, which can be defined in terms of trusting, execution, etc. However, simply the state of having been exploited, $\mathit{exploited}\,(\sigma^t, \mathcal{C})$, I define as: $\sigma^t \models \diamondsuit\,(\{\phi\}\,\mathcal{C}\,\{\psi\}) \wedge \mathit{malicious}\,(\mathcal{C}) \wedge \psi$.

7.4.1 *A logic of the kill chain*

The kill chain provides the incident analyst with heuristics for the pre-conditions that lead to observed post-conditions. It is straightforward to define expected pre- and post-conditions from each of the seven steps of the kill chain. If an analyst observes the post-conditions of one, she can abduce its pre-conditions. To complement this, Section 7.4.2 specifies more specific conditions for kill chain steps at a level of detail that is compatible with tools available to practicing incident analysts. The kill chain provides the basic structure of a single attack. Once I have defined how to reason with single attacks, Section 7.4.3 groups attacks together into campaigns.

The last step in the kill chain is the first that an incident analyst is likely to observe. Thus the measure of time starts with $t = T$, the end of the history, and works backwards to $t = 0$. Because the logic has no absolute notion of time, each discrete phase moves time back one step. In this way, analysis of

a single attack will continue to step backwards through the attack from the end to the beginning:

- Action on Objectives, the final state: the system is under adversary control
  - *Post-condition* (observed): $H, t \models \mathit{Compromised}(\sigma_1^t)$ for $t = T$.
  - *Pre-condition*: C&C, defined as: there is some σ_2 such that $H, t \models \mathit{trusts}(\sigma_1^t, \sigma_2^t) \wedge \mathit{hostile}(\sigma_2^t)$ for $t = T - 1$.

This does not tell the analyst much, but it importantly identifies that there must be some hostile system that the defender's system has come to trust. Unwinding the next steps backwards would shed light on how.

- Command and control
  - *Post-condition* (observed): C&C, as defined above
  - *Pre-condition*: Installation of malware, that is $\sigma_1^t \models \Box\left(\left\{\widehat{\phi_{\mathcal{C}_1}}\right\}\mathcal{C}_1\left\{\widehat{\psi_{\mathcal{C}_1}}\right\} \wedge \widehat{\phi_{\mathcal{C}_1}}\right)$, for $t = T - 2$. The $\Box$ indicates that the malware will be able to execute indefinitely into the future, not just once.

Where $\left\{\widehat{\phi_{\mathcal{C}_1}}\right\}\mathcal{C}_1\left\{\widehat{\psi_{\mathcal{C}_1}}\right\}$ as follows:

$\widehat{\psi_{\mathcal{C}_1}}$ is a post-condition for the adversary's objectives, including at minimum establishing a communication channel; i.e., $H, t \models \mathit{trusts}(\sigma_1^t, \sigma_2^t)$ for $t = T - 1$. Discovery of further unobserved objectives is likely one investigative goal.

$\widehat{\phi_{\mathcal{C}_1}}$ is the pre-conditions for the malware to run. These may simply be the post-conditions of the installer (i.e., $\widehat{\psi_{\mathcal{C}_2}}$, defined below), but may include what type of system the adversary can or wants to target.

A more flexible definition of the pre-conditions for command and control would be $\left(\left\{\widehat{\phi_{\mathcal{C}_1}}\right\}\mathcal{C}_1\left\{\widehat{\psi_{\mathcal{C}_1}}\right\}\right)\mathcal{U}\phi$, for some ϕ, instead of $\Box\left(\left\{\widehat{\phi_{\mathcal{C}_1}}\right\}\mathcal{C}_1\left\{\widehat{\psi_{\mathcal{C}_1}}\right\}\right)$.

- Installation of $\mathcal{C}_1$ (the main malware) by $\mathcal{C}_2$ (a downloader, installer, etc.)
  - *Post-condition* (observed): Installation, captured by $\sigma_1^t \models \left\{\widehat{\phi_{\mathcal{C}_1}}\right\}\mathcal{C}_1\left\{\widehat{\psi_{\mathcal{C}_1}}\right\} \wedge \widehat{\phi_{\mathcal{C}_1}}$, for $t = T - 2$.
  - *Pre-condition*: Exploitation; i.e., $\sigma_1^t \models \mathit{exploited}(\sigma_1^t, \mathcal{C}_2)$, for $t = T - 3$.

Note that the installation post-condition is weaker than the command and control pre-condition. The post-condition is what can be observed, but the pre-condition is abduced. In this context, the analyst should not assume the malware will stop, but rather that it will continue running indefinitely. Of course, like all abductions, this hypothesis might be changed by further observations.

Here $\left\{\widehat{\phi_{\mathcal{C}_2}}\right\}\mathcal{C}_2\left\{\widehat{\psi_{\mathcal{C}_2}}\right\}$ is as follows.

$\widehat{\psi_{\mathcal{C}_2}}$ contains at least that $\sigma_1^t \models \left(\left\{\widehat{\phi_{\mathcal{C}_1}}\right\}\mathcal{C}_1\left\{\widehat{\psi_{\mathcal{C}_1}}\right\}\right) \wedge \widehat{\phi_{\mathcal{C}_1}}$, for $t = T-2$; i.e., system one both stores the malware and is configured such that it can run.

$\widehat{\phi_{\mathcal{C}_2}}$ is a pre-condition containing at least the transfer of data necessary for installation; i.e., there is some σ_3 such that $H, t \models \mathit{trusts}\,(\sigma_1^t, \sigma_3^t)$, for $t = T-4$.

- Exploitation of system σ_1 by an exploit $\mathcal{C}_3$.
  - *Post-condition* (observed): $\sigma_1^t \models \left\{\widehat{\phi_{\mathcal{C}_2}}\right\}\mathcal{C}_2\left\{\widehat{\psi_{\mathcal{C}_2}}\right\} \wedge \widehat{\psi_{\mathcal{C}_2}}$, for $t = T-3$.
  - *Pre-condition*: $\sigma_1^t \models \mathit{vulnerable}\,(\sigma_1^t, \mathcal{C}_3)$, for $t = T-5$.

Here $\left\{\widehat{\phi_{\mathcal{C}_3}}\right\}\mathcal{C}_3\left\{\widehat{\psi_{\mathcal{C}_3}}\right\}$ is as follows:

$\widehat{\psi_{\mathcal{C}_3}}$ contains at least $\sigma_1^t \models \left(\left\{\widehat{\phi_{\mathcal{C}_2}}\right\}\mathcal{C}_2\left\{\widehat{\psi_{\mathcal{C}_2}}\right\}\right) \wedge \widehat{\phi_{\mathcal{C}_2}}$, for $t = T-4$. I say "at least" here because the exploit may or may not delete itself, for example, so in general additional traces on the system cannot be specified.

$\widehat{\phi_{\mathcal{C}_3}}$ represents the vulnerability to be exploited and targeting conditions designed by the adversary.

- Delivery of an exploit
  - *Post-condition* (observed): There exists content $c, c' \in C$ such that $\sigma_1^t \models \mathit{match}\,(\mathcal{C}_2, c) * \mathit{match}\,(\mathcal{C}_3, c')$, for $t = T-6$.
  - *Pre-condition*: There is σ_4 such that $(H, t) \models \mathit{trusts}\,(\sigma_1^t, \sigma_4^t)$, for $t = T-7$.

The delivery phase does not assume the exploit runs, just that it reaches the defender's system from somewhere. The rules abduced the existence of that system, here called σ_4.

- Weaponization against an observed vulnerability

  - *Post-condition* (explicitly unobserved): This is the creation of the malware. It also might include all the work the adversary did to discover the vulnerability, etc.
  - *Pre-condition*: The reconnaissance was successful and the adversary learns that the system $\sigma_1^t \models vul(\phi)$ for some ϕ, for $t = T - 8$.

Weaponization is an abduced step. Because it occurs local to the adversary, the defender almost never observes it, but knows that it must happen.

- Reconnaissance on target systems
  - *Post-condition*: Observable communication between σ_5 and σ_1. That is, $(H, t) \models trusts(\sigma_1^t, \sigma_5^t) \wedge \psi$, for $t = T - 9$, where ψ represents the information communicated. In some situations, it may be possible to learn what vulnerability is likely communicated, that is $\psi \Rightarrow vul(\phi)$.
  - *Pre-condition*: There exists σ_5 with $(H, t) \models trusts(\sigma_1^t, \sigma_5^t)$, for $t = 0$. Depending on the communication, it may be possible to put constraints on what cluster of resources represent σ_5.

The adversary-controlled systems $\sigma_5, \sigma_4, \sigma_3, \sigma_2$ may or may not be the same real-world system, sharing some combination of resources.

7.4.2 *Using more granular knowledge*

Chapter 5 used the kill chain as an example of mechanistic explanation and incorporated a lower (mechanistic) level of explanation via a type of exploitation: drive-by download. In this logic, I can similarly refine expressions. For example, known exploits would put constraints on $\left\{\widehat{\phi_{\mathcal{C}_3}}\right\} \mathcal{C}_3 \left\{\widehat{\psi_{\mathcal{C}_3}}\right\}$.

Integrating specific rules should enable automating the process of finding likely explanations. The incident analyst might have many thousands of potential specifications of various phases of the kill chain, derived from anti-virus signatures, blacklists, and so on. The MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) framework is an example attempt at such structured knowledge. Translating such knowledge into the logic would enable mechanization of inspection of which details are more likely to be at play in a given incident based on observations.

This subsection will demonstrate how existing knowledge bases can be leveraged in this way via a Snort rule. An IDS rule, such as Snort rules, is a structured statement of suspicious network activity. Consider an old, but representative, example rule from Roesch (1999), which introduced Snort. Translations from anti-virus rules, etc., should be similarly easy.

```
alert tcp any any -> 192.168.1.0/24 143 (content:"|E8C0 FFFF
FF|/bin/sh"; msg:"IMAP Buffer Overflow detected!";)
```

This rule is a specification of the kill chain "Delivery" phase. Some parts are responses, such as "alert", which need not be represented in the logic because response is out of the scope defined in Chapter 2. Similarly, annotation such as the "msg" field is useful, but would be implemented elsewhere.

This leaves essentially two elements of the rule. The header, which specifies the matching rules for packet headers, and the payload detection options (here, "content"). In this case, these aspects map to statements about the network, namely κ, ν. Specifically, header rules are about κ and content rules are about ν. This makes translation of such Snort rules relatively straightforward.

The network headers are simply communication between some external system, σ_4, and the defender's system σ_1. System σ_4 remains unconstrained, represented by `any any` for IP and port matches. However, there are two constraints on σ_1. Firstly, the system is 255 IP addresses, namely `192.168.1.0/24`. Represent this as the claim that there exists some $\kappa \in A$ such that

$$\sigma_1^t \models dip\,(192.168.1.0, \kappa) \vee ... \vee dip\,(192.168.1.255, \kappa) \wedge dport\,(143, \kappa)$$

The predicates $dip\,()$ and $dport\,()$ use the $match\,()$ predicate, defined to match specific parts of an IP packet header (destination IP and port, respectively).

The content is a string-matching constraint on the communication between σ_4 and σ_1. I change the notation for hexadecimal content from $|FF|$, as Snort uses, to FF. Then this half of the Snort rule is easily translated; simply assert there exists some $\nu \in C$ such that

$$\sigma_1^t \models match\,(\underline{\texttt{E8C0FFFFFF}}/\texttt{bin/sh}, \nu)\,.$$

This matches with an exploit, represented as $\mathcal{C}_3$ in this formulation. The Snort rule is the conjunction of these two statements.

Recall the broad statement of delivery in the kill chain. Transfer of data, including $\mathcal{C}_3$, from some σ_4 to σ_1. This example demonstrated how one can specify greater detail of these aspects. Specifying the specifics of all such

attacks is a huge undertaking. For that reason, I have chosen an example – Snort rules – where much of this undertaking has already been collected and curated in machine-readable form. Such existing data bases of attack patterns should be readily leveraged by this incident analysis logic.

Analysts should also propagate specifics forward in the kill chain. This example finds an attack against email servers. Therefore, the analyst knows more accurate preconditions for $\mathcal{C}_3$. Particularly, whether $\mathit{vulnerable}(\sigma_1, \mathcal{C}_3)$ holds. If σ_1 is not an email server, then it is not vulnerable. This sort of reasoning should allow the analyst to reduce the number of systems that need to be investigated as to whether the exploitation step was successful.

7.4.3 *Composition of attacks into a campaign*

To model a campaign of many attacks, join attacks together by $*$. This is particularly important because the compromised system σ_1 might be used to conduct further attacks locally. The postconditions of one attack might be preconditions for other attacks. It's important that this is $*$ and not $\wedge$, to count compromised machines and attacks as individuals.

There are prior attempts at organising general knowledge systematically out of smaller elements such as attacks. Indeed, Caltagirone et al. (2013) accomplish something similar by stitching together kill chain instances with Bayesian belief statements. However, especially in cases where attribution is to be pursued legally or in public diplomacy, a systematic and explicit campaign analysis via logic would be preferable. Such logical explanations should also be easier for incident responders to work with. As Křetínský (2018) discussed, machine learning may be best used to select among a complicated set of rules or heuristics in formal methods. I leave such optimization for future work.

Data input is another area in which I will only give examples. Section 7.4.2 provides an example of how existing network tools can be leveraged for data input. Higher level data, such as device inventories, can similarly be leveraged. MulVal (Ou et al., 2005), which helps with vulnerability management by logical discovery of impactful attack graphs, is an example open-source tool. I will focus here on how to stitch such information together into the logical analysis of an incident.

A logical description of botnet operations, such as Zeus, should be possible by composing aspects and instances of the kill chain. Chapter 4 provided a framework – clusters of mechanism schemas – that provides a workable concep-

tual definition of how more general knowledge can be built up in cybersecurity. I will not pursue such an involved example here, but rather will express the reasoning provided by Stoll (1989), which was introduced in Section 5.3.1. Stoll's example is useful because it is more simple and explicit, while still suggestive of all the steps in modern incident response reasoning.

The first steps in Stoll's response to the incident of the accounting shortage are to inventory the system. In the logic, this would be represented as populating the history monoid. Proofs and abductive rules operate on a given history, but the process of incident response also includes adding more information to what is known about the incident. As Stoll accumulates information about the home-made and UNIX-default accounting systems, this logic would capture that as disk contents (δ, β) or memory contents (s, h) at a time t. These additions can follow the defined monoid operations. Recall $\mathbf{H} = (\mathcal{H}(\mathbf{R}), \circ, e)$. A history can change by changing the resources known to be in the system or a history can change by interleaving newly discovered events into the known sequence. Both of these options are defined for $\circ$.

As Stoll builds out the details of the existing accounting system, the programs can be represented as variables, values, and pointers, in the usual way of program verification. There is a lot of work in how to do that well, as Calcagno et al. (2011) explains clearly. However, it is well understood that such structured representations of computer systems can be built well and automated. I leave such specification for histories as future work. One way or another, this is the type of information Stoll is gathering. The important insight is that a given history is essentially a hypothesis about how the system behaved. If the analyst knows a compromise occurred, but the current history does not support any explanation for how, more details will need to be added to the history. The abduction heuristics provided by the kill chain lead to some structure as to what to look to add.

Stoll knows from the very beginning that something went wrong. He may not know $H, t \models \mathit{Compromised}(\sigma_1^t)$, but perhaps a slightly more general concept is needed such as $H, t \models \mathit{Error}(\sigma_1^t)$ where Error is defined as either a compromise or a benign fault, but either way it needs to be resolved. The description in Section 5.3.1 ended when Stoll satisfied himself the system was in fact compromised.

From here, the analysis process could be captured as follows. Given a history, run an abduction and verification process with the available incident analysis heuristics. Stoll (1989) goes on to describe how he discovers the com-

mand and control system of remote communications. Within the logic, this is adding more resources and events to the model of the system history. Note that my use of model here is in both the formal logical sense and the mechanistic sense. The process for Stoll was quite human; in a modern case some aspects are well established and can be automated. Network sensors, for example, have various open-source options, including Snort. But even so, a mechanistic model should help guide the human interaction and inform what information is most valuable to seek out. The logical model can inspect the collected data and see if it matches any heuristics for attack patterns, including by abducing some steps to fill in minor gaps that are rarely observed. Stoll goes on to do this, tracing back from the C2 structure to what program was exploited, and so on. Here, the analyst has used the kill chain to analyse a single attack, and has a fair amount of structured information.

After Stoll understood the single attack on his system, he turns to a bigger question. He sought to understand the campaign the adversaries conducted. He seeks to understand their overall objectives by who else they are attacking, what their tactics and operating procedures are, and what other infrastructure they use to achieve these goals. This is captured in modern clarity by the diamond model of intrusion analysis (Caltagirone et al., 2013), the complementary de facto standard to the kill chain situated by Chapter 2.

The diamond model takes the kill chain as a model of individual attacks and stitches multiple attacks into an intrusion campaign on four dimensions. I propose to capture these connections logically. The four points of connection proposed by the diamond model are:

- Adversary
- Infrastructure
- Tools, tactics, and procedures (TTPs)
- Victim

The goal of CSIR may vary in relation to these four elements of the campaign. For example, the analyst's goal may be attribution of the adversary's identity. However, any of the four points may be the goal. For example, Stoll enumerates other victims in order to notify them as well as to understand what adversary might want to attack that collection of victims. As another example, infrastructure and TTPs are good candidates to share with other defenders to improve collective defence, or to create IDS rules.

One example of a logical abduction heuristic would be to look for other systems within defender control that are also compromised. One simple but often effective method here is to look for other systems communicating with a known C2 server. In the kill chain example, the C2 server was σ_2^t and the defender system known to be compromised was σ_1^t. So, given $H, t \models trusts\,(\sigma_1^t, \sigma_2^t)$, the investigator might abduce that there exists at least one defender systems σ_x^t such that $H, t \models trusts\,(\sigma_x^t, \sigma_2^t)$. If this abduction can be verified within the network traffic, then an additional compromised system has been found to remediate in addition to σ_1^t. There are various other elements of an attack as noted by the kill chain that an analyst can pivot on to find additional attacks within defender systems. For example, malicious software ($\mathcal{C}_1$, $\mathcal{C}_2$, or $\mathcal{C}_3$), adversary-controlled systems for exploit delivery or installation (σ_3^t or σ_4^t), or system vulnerabilities ($vul\,(\phi)$).

Expanding on attacks internally, on defender-owned systems, is one method. An incident responder can also use what I have called indicator expansion (Spring, 2013a) to find systems believed to be controlled by the adversary. This technique looks to connect up different elements of adversary infrastructure. One example application is finding fast-flux networks (Metcalf et al., 2017). However, logic may be better for capturing rules-based relations among adversary infrastructure. For example, if the system used to conduct reconnaissance is not the same as that used to deliver the exploit, then there must be a communication channel between those systems.

Capturing TTPs at a useful level of detail is a challenge. Other attack ontologies have also struggled with this. There is not a ready source to translate into logical statements. But I expect categories of attacks such as ransomware can be adequately described by their pre- and post-conditions $\left\{\widehat{\phi_{\mathcal{C}}}\right\}\mathcal{C}\left\{\widehat{\psi_{\mathcal{C}}}\right\}$. Specifically, $\widehat{\psi_{\mathcal{C}}}$ would describe a system where, for all files, the file is encrypted. These predicates could be ground out by system observables rather straightforwardly. The benefit would be that, if the tactic of ransomware is detected or known, it limits both what sorts of malicious software are likely, as well as what adversaries are likely and what their goals are.

Creating an adequate repository of heuristics and definitions for matching the elements of the diamond model will be a challenge. However, incident responders know much of this information, albeit less precisely (see, for example, ATT&CK). I leave capturing the catalogue of adversary TTPs for future work. My goal in this subsection has been to demonstrate it is possible to take

granular knowledge about specific attacks and plausibly inform campaign-level analysis.

7.5 BENEFITS OF THIS LOGIC

I believe that this logic could represent the reasoning in Section 5.3.1 and Section 5.3.2. With some work, the logic can even represent the reasoning in all of Stoll (1989), which is a large – but finite – collection of observations, reasoning heuristics, hypothetical explanations, and deduced conclusions. Stoll's reasoning is cited by CSIR standards as a model of good reasoning, so if it can be represented there is reason to believe that my logic is applicable to practical CSIR reasoning tasks. I have only sketched these usage patterns, but this logic should work similarly to Separation Logic, which has these features.

This logic satisfies the task within the research plan laid out in Section 2.7.1 to describe the reasoning with CSIR. Formal reasoning begins to fill two gaps identified in Section 2.7: strategic selection of tactics and what information to report. The logic is just a beginning to a full solution. The logical tools make headway in these gaps in the following areas:

- Communication
- Clarification
- Decision-support potential
  - Explanations
  - Automation

A logic such as this aids communication between analysts. In general, logical tools aid communication by reducing ambiguity. If one analyst describes their process in this logic, it will help other analysts understand and reproduce that process. This clarity and abstraction should assist analysts to overcome the challenge of justified secrecy among allies that Chapter 4 discussed. A logic allows the analyst to abstract out some sensitive system details, thus increasing shareability. These aspects both provide guidance on what to report.

Clarification of an analyst's own thinking is another benefit. Expressing one's reasoning in such a logical language forces an analyst to be precise. The process of clarification likely will surface analysis errors. Such errors are common; as Heuer (1999) identifies, human cognitive biases often subtly insert themselves into analytic thinking. By specifying reasoning explicitly, one can examine the reasoning process for such instances of bias and work to reduce it.

This aspect improves report clarity as well as providing guidance on analytic strategy.

Decision-support is an ultimate aim that would require significant software development. Logics are a better tool for explanations than machine learning. And explanations are ultimately what scientists seek to make the unknown intelligible (Dear, 2006). The components of a scientific explanation are outlined in Chapter 3 and fleshed out in Chapter 4. Logical tools move us towards a scientific incident analysis in part because they can represent such explanations and the justifications for them. The point of going through the pain of specifying a logic, rather than remaining in the realm of philosophy of science and natural language descriptions of incident analysis, is that logics are automatable. Automation is a clear prerequisite for any decision-support in a field like incident analysis, where data volumes are so large. At the same time, I have adapted logical tools that have demonstrated scalable reasoning in other contexts, temporal (Lamport, 2002) and Separation Logic (Chapter 6). The logic design is not just tailored to incident analysis, but, insofar as is possible at this stage, tailored to a scalable automated support for analytic strategy.

7.6 CONCLUSIONS

Based on analyst accounts and case studies, I have developed logical tools for incident analysis. My goal is both descriptive and prescriptive. On the one hand, this chapter enhanced useful and accurate descriptions of what analysts do. At the same time, analysts should emulate these descriptions and build on them, to express their process methodically. This process will be gradual. Logical tools provide a new paradigm which helps enable this gradual advancement, alongside existing incident management and forensics practices.

This work lays out the beginning of an approach to decision support for incident analysts. It also serves to highlight where additional formal definitions are appropriate (e.g., see Section 7.3.6). And of course, as with Separation Logic, the devil will be in the details of implementing such formal definitions (O'Hearn, 2015). Although the core sense-making and goal-setting aspects likely will remain a distinctly human endeavour, these logical developments provide hope that tools tailored to incident analysis could reduce the analyst's workload.

Part IV

CONSOLIDATION

In the following part, I will review the arguments made so far, consolidate my position, and conclude. Chapter 8 highlights the important aspects of the argument. Chapter 9 indicates the impacts and potential uses of my results.

8

SUMMARY

Chapter 2 argued that the incident response literature and standards do not contain a usable description of decision-making during evidence collection, analysis, and reporting stages of incident analysis.

Chapter 3 introduced the basics of relevant philosophy of science and surveyed the science of security literature. Contrary to the prevailing literature, I argue strongly that security is a science. This conclusion supports my strategy to adapt decision-making systems from other scientific fields and apply them to incident analysis, via philosophy of science.

Chapter 4 attacked the problem of building general knowledge head-on. It explored the current state of understanding about generality in philosophy of science. I found the state-of-the-art here to be lacking a coherent presentation. However, I presented a novel synthesis of the philosophical literature, developing a coherent picture of building general knowledge in cybersecurity. I identified three significant barriers to applying the synthesis to cybersecurity, namely the cybersecurity challenges of:

- Conflict with intelligent adversaries
- Changeable nature of computer software
- Justified secrecy among friendly parties

I argued that despite these challenges, cybersecurity professionals manage to build general knowledge. Although cybersecurity professionals do not call it mechanistic knowledge, cybersecurity is readily recognizable as mechanism discovery—just as described in the philosophical literature on building general knowledge. I suggested incident analysis would therefore benefit by applying the generalization strategies synthesized from the philosophical literature.

Chapter 5 applied mechanism discovery, mechanistic explanation, and generalization to a key incident analysis framework, the intrusion kill chain. There are no obvious barriers to casting the kill chain as a form of mechanistic knowledge; specifically general knowledge as hard-fought descriptions of important and importantly-stable features of adversary activity. I saw how such knowledge is qualitatively applied to evidence collection, analysis, and reporting.

Chapter 6 examined what makes for a good logical formalism in computer science. My literature review identified evidence collection, analysis, and reporting as the most human-intensive aspects of incident response. Scalability is one barrier to incident analysis. To the end of automating some decision support, Chapter 6 examined Separation Logic as a case study of successfully deployed large-scale automated decision support. I focused on the technical aspects that enabled interpretable, scalable, and useful logical support. Three features seem to be critical:

- A logical model that substantively matches the scientific model of the phenomenon of interest
- Scalability via parallelization thanks to the Frame Rule
- Automated reasoning through formalized abduction heuristics

The final part took these various insights and used them to construct a logic to support incident analysis decision-making. Chapter 7 provided the formal definition of the logic. Returning to the example of the kill chain, I then translated the reasoning demonstrated qualitatively in Chapter 5 into formal logical reasoning. The other examples of mechanistic explanation touched on should be similarly translatable. Chapter 9 will describe the various benefits of this result, how it might be used, and where future work would be best directed.

9

CONCLUSIONS

I have achieved some success in answering the research question "how to satisfactorily make clear and explicit the reasoning process used by individual Computer Security Incident Response (CSIR) analysts while they pursue technical details?"

Specifically, in response to the three component questions of the research question, I can answer:

- What heuristics should an incident analyst use to construct general knowledge and analyse attacks?
  - Where cybersecurity is like other sciences (per Chapter 3), use similar heuristics.
  - Where cybersecurity faces a unique combination of challenges, build according to the norms Chapter 4 established.
- Is it possible to construct formal tools that enable automated decision support for the analyst with such heuristics and knowledge?
  - It seems likely, given the successful construction of a logic to capture CSIR analysis in Chapter 7.
- How can an incident analyst apply general knowledge to improve analysis of specific incidents?
  - Chapter 5 demonstrated how the kill chain can be applied by an analyst, and how common examples of analysis (Stoll, Assistant for Randomized Monitoring Over Routes (ARMOR)) applied general knowledge.
  - Chapter 6 demonstrated how abduction can be automated to apply general knowledge, properly codified, to program verification. Chapter 7 suggests automated abduction may be successful in CSIR as well.

There are three areas where I expect the tools and ideas developed will have impact: benefits to practising CSIR analysts; enabling future research directions; and training for new incident analysts.

Improvements to practising incident response will take some significant effort to transition, not least because the attention of such personnel is a scarce resource. The learning curve on scientific knowledge generation heuristics will also be steep. Nonetheless, I have provided two specific improvements that can impact CSIR presently:

- norms for what general knowledge in cybersecurity should look like
- a logical language to communicate reasoning steps without revealing sensitive data

The first item is intimately linked to the argument that cybersecurity is a type of science. Creating general knowledge requires adequate design of structured observations, as touched on in Chapter 3.[1] Chapter 4 and Chapter 5 expanded on what form the resulting general knowledge should take. These advances also open up potential benefits by applying the heuristics identified from the philosophy of science literature in Chapter 3. One could also view this contribution as adapting how Heuer (1999) recommends analysts avoid cognitive biases to CSIR, and perhaps Computer Network Defense (CND) more generally.

Although future development of the logic defined in Chapter 7 will expand its usefulness, it can provide some practical benefit even in its current form. Based on Chapter 7, a practitioner could write out an analytic process by hand. This might be justified for sensitive but important processes in which communicating the details is vital but when sharing a script or code that performs the analysis includes system details and would reveal too much sensitive data to be permissible.

The work presented here will enable further research and development. I have charted out new fields of philosophy of cybersecurity in general and of computer security incident response in particular, providing a framework for this research. The three research or development areas that appear most fruitful going forwards are: automated decision support for incident response; meta-logical analysis of the properties of the logic defined; and to extend the account of explanation and mechanism discovery both into cybersecurity and back into the philosophical literature on scientific explanation generally.

The most pragmatic area of future work is how the logic presented here can provide decision support via some automation. To do this, one would likely need to build prototype systems that can ingest both large databases of

1 I have discussed structured-observation design in other work (Hatleback and Spring, 2014; Krol et al., 2016).

attacks, such as intrusion detection system (IDS) signatures, as well as a set of more human-driven heuristics such as attack ontologies like the kill chain. The Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) framework may be useful in this effort. Such a prototype would also need to ingest incident data, such as packet capture (pcap), IPFIX (i.e., netflow), or anti-virus alerts. By leveraging well-supported open-source tools on both the network forensics (e.g., System for Internet-level Knowledge (SiLK) and Bro) and program verification (e.g., Infer) halves of this direction, it may be possible to jump start progress in this area.

The trick will be to figure out whether the discovery heuristics can reliably find new suspicious behaviour. With a suitable language fixed, sociological research into cataloguing different incident analyst's heuristics and models would also be enabled. If a suitable prototype can be built and such heuristics gathered, then the interesting work of evaluating which proof rules and heuristics lead to better outcomes can begin. However, the success of this endeavour is related to further work on the logic itself.

Meta-logical analysis was touched on in Section 7.3.6. A formal proof that this CSIR logic is sound and complete is one area of future work. This work would include some other useful steps, such as specifying the proof theory in detail, which would help the decision support project. Other meta-logical studies are also available, such as formalizing the expressivity and complexity of the incident analysis logic.

Perhaps the most wide-ranging future work I have opened up is in the history and philosophy of science. My contributions have been varied and so are the areas opened for future work. Two of these contributions can be seen as historical. The historical account of the science of security community's interaction with logical empiricism in Section 3.3 opens future work to understand why that community gravitated towards those beliefs. This work also contributed to the debate on demarcation criteria of science (Chapter 3). This invites comparison between cybersecurity sciences and other sciences. The historical analysis of Separation Logic to extract why the logic work so well for program verification (Chapter 6) opens future work in logic, program verification, and philosophy. For example, whether or not Separation Logic works for the same reasons as another successful logic, Temporal Logic of Actions (TLA) and its implementation within TLA+.

Chapter 4 provided a nuanced account of how cybersecurity practitioners create generalized knowledge. However, this account is largely derived from

the philosophy of life sciences literature. The natural question for future work is what similarities there are between cybersecurity and life-science practitioners. My hypothesis is that other scientists, from biology to astronomy, actually construct general knowledge analogously to the description provided in Chapter 4. Testing the extent to which this account of generalized knowledge generalizes to other disciplines is thus promising future work. Related to the discrediting of laws in logical empiricism, philosophers currently lack a good account of how knowledge becomes general. If the account does in fact generalize, it would fill a gap that has been sitting uncomfortably in the philosophy of science literature.

The third area for impact for future work is in how to train new incident analysts. There are at least four ways this impact might be realized. First, standards bodies could work to fill the gap Chapter 2 identifies in existing standards. Secondly, both Chapter 3 and Chapter 4 identify likely challenges; future work on pedagogy could focus on how to best enable analysts to overcome them. Chapters 4, 5, and 7 offered qualitative progress on some norms and expectations for analytic steps. Integrating these ideas into training and measuring results is future work. Finally, further work is necessary on the structure for hypothesis generation via mechanism discovery, based on mechanistic general knowledge. Both theoretical work to integrate mechanism discovery heuristics with incident analysis thought processes and empirical work to measure what improvement to quality of hypotheses a structured set of heuristics may provide.

I have brought together several fields to help fill the gaps within CSIR. In order to align these fields to the problems at hand, I have contributed to those fields as well as incident analysis. Philosophy of science did not have an account of how general knowledge can be built up out of mechanistic understanding of particulars; cybersecurity may prove a better arena for that discussion than biology, since general knowledge is so hard-fought in cybersecurity. Logic and program verification did not have a perspective on why certain logics might work better than others; I have provided a plausible case. The logic I built in Chapter 7 is not just a logic suited to incident analysis, but a new kind of logic: a separable linear-time logic (SLTL, let's say) in which the temporal and spatial operators are fully commutable.

The goal of improving incident analysis and charting out a philosophy of cybersecurity is far from over. This work has laid foundations for a scientific incident analysis.

Part V

BACKMATTER

Mostly just the bibliography, this part also contains acknowledgements and the colophon.

ACKNOWLEDGEMENTS

I gratefully acknowledge the support of University College London's Overseas Research Scholarship and Graduate Research Scholarship for funding my PhD studies.

I have a lot of people to thank for their help with improving this document. Any remaining errors are my own.

Thanks to David Pym for his foresight and for taking a chance with me and and carrying through on this endeavour together.

Thanks to Phyllis Illari for her intellectual curiosity and agreeing to help me in this endeavour.

Thanks to Emiliano De Cristofaro for his support in this project.

Thanks to Eric Hatleback for his curiosity, patience, and collaboration on prior projects that inspired this work.

Thanks to Tyler Moore and Peter O'Hearn for their generous time in collaborating on work that because Chapters 3 and 6, respectively.

Various people have provided feedback on prior versions of the papers that have been adapted for the thesis. Besides my co-authors and supervisors, I'd like to thank Wolter Pieters, Karl Levitt, Marie Vasek, and all the NSPW 2017 attendees for comments on what became Chapter 3; Inge de Bal, Giuseppe Primiero, Dingmar van Eck, Stuart Glennan, and Stewart Garrick for discussions about what would become Chapter 4; my colleagues at the CERT Division of the SEI for feedback on what became Chapter 5; Simon Docherty for comments on what would become Chapter 6 and Chapter 7; Pavle Subotic for advice on Chapter 7; and finally to Claudia Cristalli for her advice on Peirce.

The various groups at UCL have been welcoming places to grow intellectually. Thanks to the STS department for adopting me; thanks to the information security and usable security groups for engaging with me; and thanks to the PPLV group for being patient with me while I learned logics.

Other co-authors during my time at UCL, even though our shared work is not adapted into this thesis, have shaped my work and my thinking. Thanks to Angela Sasse, Simon Parkin, Albesa Demjaha, Ingolf Becker, Giuseppe Primiero, Leigh Metcalf, and Eric Hatleback (again).

Thanks to Michael Spring for his generous advice on IT standards as well as generous and thorough proof reading.

Thanks to Sebastian Meiser and Alex Mittos for general feedback and engagement with this project.

Thanks to Paul Vixie for his initial encouragement and support.

Thanks to anonymous reviewers from WEIS 2016, HotSoS 2016, LASER 2016, USEC 2018, Data Science for Cybersecurity 2017, Security Protocols Workshop 2017, A Conference on Defense 2017, Philosophy & Technology (repeatedly), the European Journal for Philosophy of Science, the Journal of Cybersecurity, and NSPW – 2016, 2017, and 2018. Comments from these various people who volunteered their time have improved my thinking and my expression of ideas in the various papers that have been adapted for this thesis.

A final thank you to all my friends and loved ones, for various kinds of vital support.

Abraham, Renchie et al. (June 2015). *Common Vulnerability Scoring System v3.0: Specification Document*. Tech. rep. Forum of Incident Response and Security Teams.

Addis, Benedict and Stewart Garrick (3rd Dec. 2014). 'Botnet takedowns – our GameOver Zeus experience'. In: *Botconf*. AILB-IBFA. Nancy, France.

Adve, Sarita V. and Kourosh Gharachorloo (1996). 'Shared memory consistency models: A tutorial'. In: *Computer* 29.12, pp. 66–76.

Alberts, Chris, Audrey Dorofee, Georgia Killcrece, Robin Ruefle and Mark Zajicek (2004). *Defining Incident Management Processes for CSIRTs: A Work in Progress*. Tech. rep. CMU/SEI-2004-TR-015. Software Engineering Institute, Carnegie Mellon University.

Alberts, Christopher, Audrey Dorofee, Robin Ruefle and Mark Zajicek (2014). *An Introduction to the Mission Risk Diagnostic for Incident Management Capabilities (MRD-IMC)*. Tech. rep. CMU/SEI-2014-TN-005. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Alpcan, Tansu and Tamer Başar (2011). *Network security: A decision and game-theoretic approach*. Cambridge, UK: Cambridge University Press.

Amann, Bernhard, Robin Sommer, Aashish Sharma and Seth Hall (2012). 'A lone wolf no more: supporting network intrusion detection with real-time intelligence'. In: *Research in Attacks, Intrusions, and Defenses*, pp. 314–333.

Andersen, Holly (2017). 'What Would Hume Say? Regularities, Laws, and Mechanisms'. In: *Handbook of Mechanisms and the Mechanical Philosophy*. Ed. by Stuart Glennan and Phyllis Illari. London: Routledge.

Anderson, Gabrielle and David Pym (21st Sept. 2015). 'Substructural modal logic for optimal resource allocation'. In: *3rd International Workshop on Strategic Reasoning*. Ed. by Julian Gutierrez, Fabio Mogavero, Aniello Murano and Michael Wooldridge, pp. 1–11.

Anderson, Gabrielle, Guy McCusker and David Pym (Nov. 2016). 'A logic for the compliance budget'. In: *International Conference on Decision and Game Theory for Security*. Vol. 9996. LNCS. Springer. New York, pp. 370–381.

Anderson, Philip (25th Mar. 2015). *Electronic evidence – A basic guide for first responders*. Tech. rep. 10.2824/068545. Heraklion, GR: European Union Agency for Network and Information Security. URL: https://www.enisa.europa.eu/publications/electronic-evidence-a-basic-guide-for-first-responders/at_download/fullReport.

Anderson, R. J. (2008). *Security Engineering: A guide to building dependable distributed systems*. second. Indianapolis: Wiley.

Anderson, Ross J. (Dec. 2001). 'Why information security is hard: an economic perspective'. In: *Computer Security Applications Conference*. IEEE. New Orleans, LA, pp. 358–365.

Anderson, Ross J. and Tyler Moore (2006). 'The Economics of Information Security'. In: *Science* 314.5799, pp. 610–613. DOI: 10.1126/science.1130992.

Anderson, Ross, Chris Barton, Rainer Böhme, Richard Clayton, Michel JG Van Eeten, Michael Levi, Tyler Moore and Stefan Savage (26th June 2012). 'Measuring the cost of cybercrime'. In: *Workshop on the Economics of Information Security*. Berlin.

Angius, Nicola and Guglielmo Tamburrini (2017). 'Explaining Engineered Computing Systems' Behaviour: the Role of Abstraction and Idealization'. In: *Philosophy & Technology* 30.2, pp. 239–258.

Anti-Phishing Working Group (2016). *APWG Phishing Attack Trends Reports*. accessed Jan 2016. URL: https://apwg.org/resources/apwg-reports/.

Antonakakis, M., R. Perdisci, W. Lee, N. Vasiloglou II and D. Dagon (2011). 'Detecting Malware Domains at the Upper DNS Hierarchy'. In: *20th Usenix Security Symposium*. San Francisco, CA.

Appel, Andrew W (2015). 'Verification of a cryptographic primitive: SHA-256'. In: *Transactions on Programming Languages and Systems (TOPLAS)* 37.2, p. 7.

Appel, Andrew W, Robert Dockins, Aquinas Hobor, Lennart Beringer, Josiah Dodds, Gordon Stewart, Sandrine Blazy and Xavier Leroy (2014). *Program logics for certified compilers*. New York: Cambridge University Press.

Apt, Krzysztof R. (1981). 'Ten years of Hoare's logic: a survey—Part I'. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 3.4, pp. 431–483.

Avancha, Sasikanth, Amit Baxi and David Kotz (Dec. 2012). 'Privacy in Mobile Technology for Personal Healthcare'. In: *ACM Comput. Surv.* 45.1, 3:1–3:54.

Avgerinos, Thanassis, Sang Kil Cha, Alexandre Rebert, Edward J Schwartz, Maverick Woo and David Brumley (2014). 'Automatic exploit generation'. In: *Communications of the ACM* 57.2, pp. 74–84.

Axelsson, Stefan (2000). 'The base-rate fallacy and the difficulty of intrusion detection'. In: *ACM Transactions on Information and System Security (TISSEC)* 3.3, pp. 186–205.

Baader, Franz (2003). *The description logic handbook: Theory, implementation and applications*. Cambridge, UK: Cambridge University Press.

Bace, Rebecca and Peter Mell (2001). *Intrusion detection systems*. Tech. rep. SP 800-31. Gaithersburg, MD: U.S. National Institute of Standards and Technology.

Ballou, Susan et al. (2001). *Electronic Crime Scene Investigation: a Guide for First Responders*. Ed. by Technical Working Group for Electronic Crime Scene Investigation. Washington, D.C.: US Department of Justice, Office of Justice Programs, National Instit. of Justice.

Balmer, Brian (2013). *Secrecy and science: A historical sociology of biological and chemical warfare*. Ashgate Publishing, Ltd.

Bartholomew, Brian and Juan Andres Guerrero-Saade (5th Oct. 2016). *Wave your false flags! Deception tactics muddying attribution in targeted attacks*. Tech. rep. Presented at Virus Bulletin 2016. Woburn, MA: Kaspersky Lab USA.

Barwise, Jon and Jerry Seligman (1997). *Information flow: the logic of distributed systems*. Cambridge University Press. ISBN: 9780511895968.

Beall, Jc et al. (2012). 'On the ternary relation and conditionality'. In: *Journal of Philosophical Logic* 41.3, pp. 595–612.

Bechtel, William (2007). *Mental mechanisms: Philosophical perspectives on cognitive neuroscience*. 1st ed. London: Routledge.

Bechtel, William and Robert C. Richardson (1993). *Discovering complexity: Decomposition and localization as strategies in scientific research*. 1st. Princeton, NJ: Princeton University Press.

Beebe, Nicole Lang and Jan Guynes Clark (2005). 'A hierarchical, objectives-based framework for the digital investigations process'. In: *Digital Investigation* 2.2, pp. 147–167.

Bejtlich, Richard (2004). *The Tao of network security monitoring: beyond intrusion detection*. Pearson Education.

Bell, D.E. and L.J. LaPadula (Nov. 1973). *Secure Computer Systems: Mathematical Foundations*. Tech. rep. ESD-TR-73-278. Bedford, MA: MITRE Corporation. URL: http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=AD0770768.

Bellovin, Steve (Sept. 1992). 'There Be Dragons'. In: *USENIX Security Symposium*. Baltimore, MD.

Bergman, Mats and Sami Paavola (14th July 2016). *'Abduction': term in The Commens Dictionary: Peirce's Terms in His Own Words. New Edition*. http://www.commens.org/dictionary/term/abduction.

Bickle, John (2008). 'Real reduction in real neuroscience: metascience, not philosophy of science (and certainly not metaphysics!)' In: *Being reduced: New essays on reduction, explanation, and causation*. Ed. by Jakob Hohwy and Jesper Kalestrup. Oxford University Press, pp. 34–51.

Biddle, Robert, Sonia Chiasson and P.C. Van Oorschot (Sept. 2012a). 'Graphical Passwords: Learning from the First Twelve Years'. In: *ACM Comput. Surv.* 44.4, 19:1–19:41.

– (Sept. 2012b). 'Graphical Passwords: Learning from the First Twelve Years'. In: *ACM Comput. Surv.* 44.4, 19:1–19:41.

Blackshear, Sam and Peter W. O'Hearn (19th Oct. 2017). *Open-sourcing RacerD: Fast static race detection at scale*. URL: https://code.facebook.com/posts/293371094514305/open-sourcing-racerd-fast-static-race-detection-at-scale/.

Bogen, James and James Woodward (1988). 'Saving the phenomena'. In: *The Philosophical Review* XCVII.3, pp. 303–352.

Boolos, George S., John P. Burgess and Richard C. Jeffrey (2002). *Computability and logic*. 4th. Cambridge: Cambridge University Press.

Bornat, Richard (2000). 'Proving pointer programs in Hoare logic'. In: *Mathematics of Program Construction*. LNCS 1837. Springer, pp. 102–126.

Brand, S.L. (1985). *DoD 5200.28-STD Department of Defense Trusted Computer System Evaluation Criteria (Orange Book)*. Tech. rep.

Brenner, Susan W (2002). 'Organized cybercrime: How cyberspace may affect the structure of criminal relationships'. In: *North Carolina Journal of Law & Technology* 4, p. 1.

Brooks Jr, Frederick P (1995). *The Mythical Man-Month: Essays on Software Engineering*. 2nd. Boston, MA: Addison Wesley.

Brookson, Charles et al. (Dec. 2015). *Definition of Cybersecurity: Gaps and overlaps in standardisation*. Tech. rep. v1.0. Heraklion, GR: ENISA.

Brotherston, James and Jules Villard (2015). 'Sub-Classical Boolean Bunched Logics and the Meaning of Par'. In: *Proceedings of CSL-24*. LIPIcs. Dagstuhl, pp. 325–342.

Brownlee, N. and E. Guttman (June 1998). *Expectations for Computer Security Incident Response*. RFC 2350 (Best Current Practice). RFC. Fremont, CA, USA: RFC Editor.

CERT/CC (2017). *"Basic Fuzzing Framework (BFF)"*. https://www.cert.org/vulnerability-analysis/tools/bff.cfm. accessed Feb 6, 2017.

Cain, P. and D. Jevans (July 2010). *Extensions to the IODEF-Document Class for Reporting Phishing*. RFC 5901 (Proposed Standard). RFC. Fremont, CA, USA: RFC Editor.

Calcagno, Cristiano, Peter W. O'Hearn and Hongseok Yang (2007). 'Local action and abstract separation logic'. In: *Logic in Computer Science*. IEEE, pp. 366–378.

Calcagno, Cristiano, Dino Distefano, Peter W. O'Hearn and Hongseok Yang (2011). 'Compositional shape analysis by means of bi-abduction'. In: *J. ACM* 58.6, 26:1–26:66.

Calcagno, Cristiano et al. (2015a). 'Moving fast with software verification'. In: *NASA Formal Methods*. LNCS 9058. Springer, pp. 3–11.

Calcagno, Cristiano, Dino Distefano and Peter W. O'Hearn (11th June 2015b). *Open-sourcing Facebook Infer: Identify bugs before you ship*. https://code.facebook.com/posts/1648953042007882/open-sourcing-facebook-infer-identify-bugs-before-you-ship/.

Caltagirone, Sergio (2005). *Evolving active defense strategies*. Tech. rep. CSDS-DF-TR-05-27. Moscow, ID, USA: University of Idaho.

Caltagirone, Sergio and Deborah Frincke (2005). 'ADAM: Active defense algorithm and model'. In: *Aggressive Network Self-Defense*, pp. 287–311.

Caltagirone, Sergio, Andrew Pendergast and Christopher Betz (2013). *The Diamond Model of Intrusion Analysis*. Tech. rep. http://www.threatconnect.com/methodology/diamond_model_of_intrusion_analysis. Center for Cyber Intelligence Analysis and Threat Research.

Calzavara, Stefano, Riccardo Focardi, Marco Squarcina and Mauro Tempesta (Mar. 2017). 'Surviving the Web: A Journey into Web Session Security'. In: *ACM Comput. Surv.* 50.1, 13:1–13:34.

Caralli, Richard, James Stevens, Lisa Young and William Wilson (2007). *Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process*. Tech. rep. CMU/SEI-2007-TR-012. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Carrier, Brian and Eugene H Spafford (2003). 'Getting physical with the digital investigation process'. In: *International Journal of digital evidence* 2.2, pp. 1–20.

– (Aug. 2004). 'An event-based digital forensic investigation framework'. In: *Digital forensic research workshop*. Baltimore, MD, pp. 1–12.

Cartwright, Nancy (1983). *How the Laws of Physics Lie*. Oxford: Clarendon Press.

– (1991). 'Replicability, reproducibility, and robustness: Comments on Harry Collins'. In: *History of Political Economy* 23.1, pp. 143–155.

– (2012). 'RCTs, Evidence, and Predicting Policy Effectiveness'. In: ed. by Harold Kincaid. Oxford: Oxford University Press, pp. 298–318.

Cartwright, Nancy and Jeremy Hardie (2012). *Evidence-based policy: a practical guide to doing it better*. New York: Oxford University Press.

Casey, Eoghan (2010). *Handbook of digital forensics and investigation*. Elsevier.

Caulfield, Tristan and David Pym (2015a). 'Improving Security Policy Decisions with Models'. In: *IEEE Security & Privacy* 13.5, pp. 34–41. ISSN: 1540-7993. DOI: doi.ieeecomputersociety.org/10.1109/MSP.2015.97.

– (2015b). 'Modelling and simulating systems security policy'. In: *Proceedings of the 8th International Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 9–18.

Chandola, Varun, Arindam Banerjee and Vipin Kumar (July 2009). 'Anomaly Detection: A Survey'. In: *ACM Comput. Surv.* 41.3, 15:1–15:58.

Chen, Haogang, Daniel Ziegler, Tej Chajed, Adam Chlipala, M Frans Kaashoek and Nickolai Zeldovich (4th Oct. 2015). 'Using Crash Hoare logic for certifying the FSCQ file system'. In: *25th Symposium on Operating Systems Principles*. ACM. Monterey, CA, pp. 18–37.

Cheswick, Bill (Jan. 1992). 'An Evening with Berferd: In which a cracker is Lured, Endured, and Studied'. In: *USENIX Winter Technical Conference*. San Francisco, pp. 20–24.

Cheswick, W.R., S.M. Bellovin and A.D. Rubin (2003). *Firewalls and Internet security: repelling the wily hacker*. 2nd ed. Addison-Wesley Professional.

Chiang, Tung Ju, Jen Shiang Kouh and Ray-I Chang (2009). 'Ontology-based risk control for the incident management'. In: *International Journal of Computer Science and Network Security* 9.11, p. 181.

Chismon, David and Martyn Ruks (2015). *Threat Intelligence: Collecting, Analysing, Evaluating*. Tech. rep. London: MWR InfoSecurity.

Chockler, Hana and Joseph Y. Halpern (2004). 'Responsibility and blame: A structural-model approach'. In: *Journal of Artificial Intelligence Research* 22, pp. 93–115.

Ciardhuáin, Séamus Ó (2004). 'An extended model of cybercrime investigations'. In: *International Journal of Digital Evidence* 3.1, pp. 1–22.

Cichonski, Paul, Tom Millar, Tim Grance and Karen Scarfone (Aug. 2012). *Computer Security Incident Handling Guide*. Tech. rep. SP 800-61r2. Gaithersburg, MD: US Dept of Commerce, National Institute of Standards and Technology.

Clarke, Brendan, Donald Gillies, Phyllis Illari, Federica Russo and Jon Williamson (2014). 'Mechanisms and the evidence hierarchy'. In: *Topoi* 33.2, pp. 339–360.

Cohen, Fred, Julie Lowrie and Charles Preston (Feb. 2011). 'The state of the science of digital evidence examination'. In: ed. by Gilbert Peterson and Sujeet Shenoi. Orlando, FL: IFIP, pp. 3–21.

Cohen, Frederick B (1995). *Protection and security on the information superhighway*. John Wiley & Sons, Inc.

Cohen, Frederick B. (2010). 'Fundamentals of Digital Forensic Evidence'. In: *Handbook of information and communication security*. Ed. by Peter Stavroulakis and Mark Stamp. New York: Springer, pp. 790–808.

Collinson, M., B. Monahan and D. Pym (2012a). *A Discipline of Math.Systems Modelling*. College Publns.

Collinson, Matthew, Brian Monahan and David J. Pym (2012b). *A discipline of mathematical systems modelling*. Vol. 2. Systems Thinking and Systems Engineering. London: College Publications.

Cormack, Andrew (2015). *JANET Suggested Charter for System Administrators*. Tech. rep. Bristol, UK: Jisc.

Cousot, Patrick and Radhia Cousot (Jan. 1977). 'Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints'. In: *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages*. Los Angeles, CA, pp. 238–252. DOI: 10.1145/512950.512973.

Craver, Carl F. (2006). 'When mechanistic models explain'. In: *Synthese* 153.3, pp. 355–376.

– (2007). *Explaining the brain: mechanisms and the mosaic of unity of neuroscience*. Oxford: Oxford University Press.

Craver, Carl and James Tabery (2017). 'Mechanisms in Science'. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Spring 2017. Metaphysics Research Lab, Stanford University.

Creath, Richard (2014). 'Logical Empiricism'. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Spring 2014. Metaphysics Research Lab, Stanford University.

Danyliw, R. (Nov. 2016). *The Incident Object Description Exchange Format Version 2*. RFC 7970 (Proposed Standard). RFC. Fremont, CA, USA: RFC Editor.

Danyliw, R., J. Meijer and Y. Demchenko (Dec. 2007). *The Incident Object Description Exchange Format*. RFC 5070 (Proposed Stand-

ard). RFC. Obsoleted by RFC 7970, updated by RFC 6685. Fremont, CA, USA: RFC Editor. URL: https://www.rfc-editor.org/rfc/rfc5070.txt.

Darden, Lindley (2006). *Reasoning in Biological Discoveries: Essays on Mechanisms, Interfield Relations, and Anomaly Resolution*. Cambridge, UK: Cambridge University Press.

Darden, Lindley and Carl Craver (2002). 'Strategies in the interfield discovery of the mechanism of protein synthesis'. In: *Studies in History and Philosophy of Biological and Biomedical Sciences* 33.1, pp. 1–28.

Darden, Lindley and Nancy Maull (1977). 'Interfield theories'. In: *Philosophy of science* 44, pp. 43–64.

Das, Anupam, Joseph Bonneau, Matthew Caesar, Nikita Borisov and XiaoFeng Wang (2014). 'The Tangled Web of Password Reuse'. In: *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society. URL: http://www.internetsociety.org/doc/tangled-web-password-reuse.

Dawid, A Philip (2010). 'Beware of the DAG!' In: *NIPS Causality: Objectives and Assessment* 6, pp. 59–86.

De Millo, Richard A, Richard J Upton and Alan J Perlis (1979). 'Social processes and proofs of theorems and programs'. In: *Communications* 22.5, pp. 271–280. DOI: 10.1145/359104.359106.

Dear, Peter (2006). *The intelligibility of nature: How science makes sense of the world*. Chicago and London: University of Chicago Press.

Demjaha, Albese, Jonathan M Spring, Ingolf F Becker, Simon Parkin and M Angela Sasse (18th Feb. 2018). 'Metaphors considered harmful? An exploratory study of the effectiveness of functional metaphors for end-to-end encryption'. In: *Workshop on Usable Security (USEC)*. San Diego, CA: ISOC.

Devlin, Keith (1995). *Logic and information*. Cambridge University Press.

Ditmarsch, Hans van, Joeseph Y. Halpern, Wiebe van der Hoek and Barteld Kooi, eds. (2015). *Handbook of epistemic logic*. London: College Publications.

Dittrich, David and Erin Kenneally (Aug. 2012). *The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research*. Tech. rep. U.S. Department of Homeland Security. URL: http://www.caida.org/publications/papers/2012/menlo_report_actual_formatted/.

Drummond, David (12th Jan. 2010). *A new approach to China*. http://googleblog.blogspot.com/2010/01/new-approach-to-china.html.

Dunn, J. Michael and Greg Restall (2002). 'Relevance Logic'. In: *Handbook of philosophical logic*. Ed. by Dov M. Gabbay and F. Guenther. Vol. 6. Dordrecht: Springer Netherlands, pp. 1–128.

Dupré, John (2012). *Processes of life: essays in the philosophy of biology*. Oxford: Oxford University Press.

Duran, Felicia, Stephen H Conrad, Gregory N Conrad, David P Duggan and Edward Bruce Held (2009). 'Building a system for insider security'. In: *IEEE Security & Privacy* 7.6, pp. 30–38.

Dykstra, Josiah (2015). *Essential cybersecurity science: build, test, and evaluate secure systems*. "O'Reilly Media, Inc."

ENISA (2006). *A Step-by-step Approach on How to Set Up a CSIRT*. Tech. rep. WP2006/5.1. Heraklion, Greece.

ETSI (June 2014). *Key Performance Security Indicators (KPSI) to evaluate the maturity of security event detection*. Tech. rep. GS ISI 003 V1.1.2. Cedex, France: ETSI Information Security Indicators (ISI).

Edwards, Harry T. et al. (2009). *Strengthening forensic science in the United States: a path forward*. Washington, D.C.: National Academies Press.

Edwards, Matthew, Awais Rashid and Paul Rayson (Sept. 2015). 'A Systematic Survey of Online Data Mining Technology Intended for Law Enforcement'. In: *ACM Comput. Surv.* 48.1, 15:1–15:54.

Egele, Manuel, Theodoor Scholte, Engin Kirda and Christopher Kruegel (Mar. 2008). 'A Survey on Automated Dynamic Malware-analysis Techniques and Tools'. In: *ACM Comput. Surv.* 44.2, 6:1–6:42.

Egelman, Serge, Andreas Sotirakopoulos, Ildar Muslukhov, Konstantin Beznosov and Cormac Herley (2013). 'Does My Password Go Up to Eleven?: The Impact of Password Meters on Password Selection'. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Paris, France: ACM, pp. 2379–2388. DOI: 10.1145/2470654.2481329.

Ekelhart, Andreas, Stefan Fenz, Markus Klemen and Edgar Weippl (6th Jan. 2007). 'Security ontologies: Improving quantitative risk analysis'. In: *Hawaii International Conference on System Sciences*. IEEE. Waikoloa, HI, 156a.

Elster, Jon (1983). *Explaining technical change: A case study in the philosophy of science*. Cambridge, UK: Cambridge Univ Press.

– (1989). *Nuts and bolts for the social sciences*. Cambridge, UK: Cambridge Univ Press.

Ensmenger, Nathan (2015). 'Beards, Sandals, and Other Signs of Rugged Individualism: Masculine Culture within the Computing Professions'. In: *Osiris* 30, pp. 38–65.

Espinosa, Gadi Tellez and James Brotherston (2017). 'Automatically Verifying Temporal Properties of Programs with Cyclic Proof'. In: *CADE-26*. Vol. 10395. LNAI. Springer, pp. 491–508.

Evans, David and Sal Stolfo (2011). 'The Science of Security: Guest editors' introduction'. In: *Security & Privacy* 9.3, pp. 16–17.

Evron, Gadi (25th Jan. 2017). *Art into Science: A conference on defense*. http://artintoscience.com/. accessed Apr 2017.

FIRST (26th June 2003). *FIRST Vision and Mission Statement*. https://first.org/about/mission. accessed Jun 2017.

– (2017). *Security Reference Index*. https://first.org/resources/guides/reference. accessed Feb 4, 2017.

Federal Aviation Administration (17th Aug. 2015). *FAA Statement on Automation Problems at Washington Center*. https://www.faa.gov/news/press_releases/news_story.cfm?newsId=19354.

Feitelson, Dror G (2015). 'From repeatability to reproducibility and corroboration'. In: *ACM SIGOPS Operating Systems Review* 49.1, pp. 3–11.

Fenz, Stefan and Andreas Ekelhart (12th Mar. 2009). 'Formalizing information security knowledge'. In: *Symposium on information, Computer, and Communications Security*. ACM. Sydney, Australia, pp. 183–194.

Ferguson, P. and D. Senie (May 2000). *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*. RFC 2827 (Best Current Practice). RFC. Updated by RFC 3704. Fremont, CA, USA: RFC Editor. URL: https://www.rfc-editor.org/rfc/rfc2827.txt.

Fetzer, James H (1988). 'Program verification: the very idea'. In: *Communications of the ACM* 31.9, pp. 1048–1063.

Fisher, Sir Ronald Aylmer (1971). *The design of experiments*. 8th. First published 1935. New York: Hafner Publishing Company.

Flechais, Ivan, Jens Riegelsberger and M. Angela Sasse (2005). 'Divide and Conquer: The Role of Trust and Assurance in the Design of Secure Socio-technical Systems'. In: *Workshop on New Security Paradigms*. NSPW. Lake Arrowhead, California: ACM, pp. 33–41.

Floridi, Luciano (2011). *The philosophy of information*. Oxford: Oxford University Press.

Floridi, Luciano and Phyllis Illari (2014). *The Philosophy of Information Quality*. Vol. 358. Synthese Library. Springer. ISBN: 978-3-319-07121-3.

Floridi, Luciano, Nir Fresco and Giuseppe Primiero (2015). 'On malfunctioning software'. In: *Synthese* 192.4, pp. 1199–1220.

Fraser, B. (Sept. 1997). *Site Security Handbook*. RFC 2196 (Informational). RFC. Fremont, CA, USA: RFC Editor.

Friedman, Michael (1974). 'Explanation and scientific understanding'. In: *Journal of Philosophy* 71.1, pp. 5–19.

Frigg, Roman and Stephan Hartmann (2012). 'Models in science'. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Fall 2012. Metaphysics Research Lab, Stanford University.

Gal-Or, Esther and Anindya Ghose (2005). 'The Economic Incentives for Sharing Security Information'. In: *Information Systems Research* 16.2, pp. 186–208.

Galison, Peter (1999). 'Trading zone: Coordinating action and belief'. In: *The Science Studies Reader*, pp. 137–160.

– (2010). 'Trading with the enemy'. In: *Trading zones and interactional expertise. Creating new kinds of collaboration*. Ed. by Michael E. Gorman. Cambridge, MA: MIT Press. Chap. 3.

– (29th Nov. 2012). 'Augustinian and Manichaean Science'. In: *Symposium on the Science of Security*. National Harbor, MD: CPS-VO.

Galmiche, Didier, Daniel Méry and David Pym (2005). 'The semantics of BI and resource tableaux'. In: *Mathematical Structures in Computer Science* 15.06, pp. 1033–1088.

Garfinkel, Simson, Paul Farrell, Vassil Roussev and George Dinolt (2009). 'Bringing science to digital forensics with standardized forensic corpora'. In: *digital investigation* 6, S2–S11.

Gaw, Shirley and Edward W. Felten (2006). 'Password Management Strategies for Online Accounts'. In: *Symposium on Usable Privacy and Security*. Pittsburgh, PA, USA: ACM, pp. 44–55. DOI: 10.1145/1143120.1143127.

Geer, Dan (6th Aug. 2014). 'Cybersecurity as Realpolitik'. In: *Black Hat USA 2014*. Las Vegas, Nevada: UBM. URL: http://geer.tinho.net/geer.blackhat.6viii14.txt.

– (6th Jan. 2015). 'T.S. Kuhn revisited'. In: *NSF Secure and Trustworthy Cyberspace Principal Investigators' Meeting*. Arlington, VA. URL: http://geer.tinho.net/geer.nsf.6i15.txt.

Girard, Jean-Yves (1987). 'Linear logic'. In: *Theoretical Computer Science* 50.1, pp. 1–101.

Given, Lisa M, ed. (2008). *The Sage encyclopedia of qualitative research methods*. Thousand Oaks, CA: Sage.

Glennan, Stuart (1997). 'Capacities, Universality, and Singularity'. In: *Philosophy of Science* 64.4, pp. 605–626.

– (2005). 'Modeling mechanisms'. In: *Studies in History and Philosophy of Biological and Biomedical Sciences* 36.2, pp. 443–464.

– (2010). 'Ephemeral Mechanisms and Historical Explanation'. In: *Erkenntnis* 72, pp. 251–266.

– (2011). 'Singular and general causal relations: A mechanist perspective'. In: *Causality in the Sciences*. Ed. by Phyllis Illari, Federica Russo and Jon Williamson. Oxford: Oxford University Press, pp. 789–817.

– (Aug. 2015). 'Mechanisms and Mechanical Philosophy'. In: *The Oxford Handbook of Philosophy of Science*. Ed. by Paul Humphreys. Oxford University Press.

– (2017). *The new mechanical philosophy*. Oxford, UK: Oxford University Press. ISBN: 9780198779711.

Glennan, Stuart and Phyllis Illari, eds. (2017). *The Routledge Handbook of Mechanisms and Mechanical Philosophy*. Handbooks in Philosophy. London, UK: Routledge.

Gorzelak, Katarzyna, Tomasz Grudziecki, Paweł Jacewicz, Przemysław Jaroszewski, Łukasz Juszczyk and Piotr Kijewski (2011). *Proactive Detection of Network Security Incidents*. Tech. rep. 2011-12-07. Heraklion, Greece: CERT Polska / NASK.

Grance, Tim, Tamara Nolan, Kristin Burke, Rich Dudley, Gregory White and Travis Good (Sept. 2006). *Guide to Test, Training, and Exercise Programs for IT Plans and Capabilities*. Tech. rep. SP 800-84. Gaithersburg, MD: US Dept of Commerce, National Institute of Standards and Technology.

Grier, Chris et al. (2012). 'Manufacturing Compromise: The Emergence of Exploit-as-a-service'. In: *ACM Conference on Computer and*

Communications Security. CCS '12. Raleigh, North Carolina, USA, pp. 821–832.

Guttman, E., L. Leong and G. Malkin (Feb. 1999). *Users' Security Handbook*. RFC 2504 (Informational). RFC. Fremont, CA, USA: RFC Editor.

Hafner, Katie and Matthew Lyon (1998). *Where wizards stay up late: The origins of the Internet*. Simon and Schuster.

Hallenbeck, Chris, Chris King, Jonathan M Spring and Paul Vixie (7th Aug. 2014). 'Abuse of Customer Premise Equipment and Recommended Actions'. In: *Black Hat USA 2014*. Las Vegas, Nevada: UBM.

Halpern, Joseph Y. and Judea Pearl (2005a). 'Causes and explanations: A structural-model approach. Part I: Causes'. In: *The British Journal for the Philosophy of Science* 56.4, pp. 843–887.

– (2005b). 'Causes and explanations: A structural-model approach. Part II: Explanations'. In: *The British Journal for the Philosophy of Science* 56.4, pp. 889–911.

Hankins, Ryan, Tetsutaroh Uehara and Jigang Liu (July 2009). 'A comparative study of forensic science and computer forensics'. In: *Secure Software Integration and Reliability Improvement*. IEEE. Shanghai, pp. 230–239.

Hatleback, Eric N (2017). 'The protoscience of cybersecurity'. In: *The Journal of Defense Modeling and Simulation*, pp. 1–8.

Hatleback, Eric and Jonathan M Spring (2014). 'Exploring a mechanistic approach to experimentation in computing'. In: *Philosophy & Technology* 27.3, pp. 441–459.

– (2018). 'A Refinement to the General Mechanistic Account'. In: *European Journal for Philosophy of Science*.

Hawkins, Douglas M (2004). 'The problem of overfitting'. In: *Journal of chemical information and computer sciences* 44.1, pp. 1–12.

Hayes, LTC Ashton (2008). 'Defending Against the Unknown: Anti-terrorism and the Terrorist Planning Cycle'. In: *The Guardian* 10.1, pp. 32–36.

Hempel, Carl G. (1942). 'The function of general laws in history'. In: *Journal of Philosophy* 39, pp. 35–48.

– (1965). *Aspects of Scientific Explanation*. New York: Free Press.

Henderson, Leah (2018). 'The Problem of Induction'. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Summer 2018. Metaphysics Research Lab, Stanford University.

Herley, Cormac and P.C. van Oorschot (23rd May 2017). 'SoK: Science, Security, and the Elusive Goal of Security as a Scientific Pursuit'. In: *Symposium on Security and Privacy (Oakland)*. San Jose, CA: IEEE.

Heuer Jr., Richards J (1999). *Psychology of intelligence analysis*. US Central Intelligence Agency.

Heule, Marijn J. H. and Oliver Kullmann (2017). 'The science of brute force'. In: *Commun. ACM* 60.8, pp. 70–79. DOI: 10.1145/3107239.

Hoare, Charles Antony Richard (1969). 'An axiomatic basis for computer programming'. In: *Communications of the ACM* 12.10, pp. 576–580.

Hodgson, J.P.E. (1st Mar. 1999). *Project "Contraintes" Prolog Web Pages: The ISO Standard*. URL: http://http://www.deransart.fr//prolog/overview.html.

Homer, John, Ashok Varikuti, Xinming Ou and Miles A McQueen (2008). 'Improving attack graph visualization through data reduction and attack grouping'. In: *Visualization for computer security*. Springer, pp. 68–79.

Hoogstraaten, Hans (13th Aug. 2012). *Black Tulip: Report of the investigation into the DigiNotar Certificate Authority breach*. Tech. rep. Fox-IT.

Hopkins, Stefan (2009). 'From Snort to Sourcefire to Nasdaq'. In: *Clarkson University Magazine* Summer. URL: http://www.clarkson.edu/alumni_magazine/summer2009/cybersecurity_roesch.html.

Howard, John D and Thomas A Longstaff (Oct. 1998). *A common language for computer security incidents*. Tech. rep. SAND98-8667. Sandia National Laboratories.

Hutchins, Eric M, Michael J Cloppert and Rohan M Amin (2011). 'Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains'. In: *Leading Issues in Information Warfare & Security Research* 1, p. 80.

ISO/IEC (June 1996). *Open Systems Interconnection – Basic Reference Model: The Basic Model*. Tech. rep. 7498-1:1994(E). International Organization for Standardization and International Electrotechnical Commission.

ISO/IEC (Oct. 2012). *Information technology – Security techniques – Guidelines for identification, collection, acquisition and preservation of digital evidence*. Tech. rep. 27037:2012. Geneva: International Organization for Standardization and International Electrotechnical Commission.

– (June 2015a). *Information technology – Security techniques – Guidance on assuring suitability and adequacy of incident investigative method*. Tech. rep. 27041:2015. Geneva: International Organization for Standardization and International Electrotechnical Commission.

– (June 2015b). *Information technology – Security techniques – Guidelines for the analysis and interpretation of digital evidence*. Tech. rep. 27042:2015. Geneva: International Organization for Standardization and International Electrotechnical Commission.

– (Mar. 2015c). *Information technology – Security techniques – Incident investigation principles and processes*. Tech. rep. 27043:2015. Geneva: International Organization for Standardization and International Electrotechnical Commission.

– (Nov. 2016). *Information technology – Security techniques – Information security incident management – Part 1: Principles of incident management*. Tech. rep. 27035-1:2016. Geneva: International Organization for Standardization and International Electrotechnical Commission.

Illari, Phyllis McKay (2011). 'Mechanistic evidence: disambiguating the Russo–Williamson thesis'. In: *International Studies in the Philosophy of Science* 25.2, pp. 139–157.

Illari, Phyllis McKay and Jon Williamson (2012). 'What is a mechanism? Thinking about mechanisms across the sciences'. In: *European Journal for Philosophy of Science* 2.1, pp. 119–135.

Illari, Phyllis (2013). 'Mechanistic explanation: Integrating the ontic and epistemic'. In: *Erkenntnis* 78.2, pp. 237–255.

Illari, Phyllis and Jon Williamson (2013). 'In Defence of Activities'. In: *Journal for General Philosophy of Science* 44.1, pp. 69–83.

Inacio, C. and D. Miyamoto (May 2017). *Management Incident Lightweight Exchange (MILE) Implementation Report*. RFC 8134 (Informational). RFC. Fremont, CA, USA: RFC Editor.

Ishtiaq, Samin S. and Peter W. O'Hearn (2001). 'BI As an Assertion Language for Mutable Data Structures'. In: *Principles of Programming Languages*. London, UK: ACM, pp. 14–26. ISBN: 1-58113-336-7.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis*. Wiley & Sons.

Jasanoff, Sheila (1990). *The fifth branch: Science advisers as policymakers*. Cambridge, MA, USA: Harvard University Press.

Jha, Somesh, Oleg Sheyner and Jeannette Wing (June 2002). 'Two formal analyses of attack graphs'. In: *Computer Security Foundations Workshop*. IEEE. Cape Brenton, Nova Scotia, pp. 49–63.

Jhaveri, Mohammad Hanif, Orcun Cetin, Carlos Gañán, Tyler Moore and Michel Van Eeten (Jan. 2017). 'Abuse Reporting and the Fight Against Cybercrime'. In: *ACM Comput. Surv.* 49.4, 68:1–68:27.

Jiang, Wenjun, Guojun Wang, Md Zakirul Alam Bhuiyan and Jie Wu (May 2016). 'Understanding Graph-Based Trust Evaluation in Online Social Networks: Methodologies and Challenges'. In: *ACM Comput. Surv.* 49.1, 10:1–10:35.

John, Wolfgang and Tomas Olovsson (2008). 'Detection of malicious traffic on back-bone links via packet header analysis'. In: *Campus-Wide Information Systems* 25.5, pp. 342–358.

Johnson, Arnold, Kelley Dempsey, Ron Ross, Sarbari Gupta and Dennis Bailey (Aug. 2011). *Guide for Security-Focused Configuration Management of Information Systems*. Tech. rep. SP 800-128. Gaithersburg, MD: US Dept of Commerce, National Institute of Standards and Technology.

Joint Chiefs of Staff (31st Jan. 2013). *Joint Targeting*. Tech. rep. JP 3-60. Washington, D.C.: U.S. Dept of Defense.

– (20th Nov. 2014a). *Information Operations*. Tech. rep. JP 3-13. Washington, D.C.: U.S. Dept of Defense.

– (2014b). *Information Operations*. Tech. rep. JP 3-13. United States Armed Forces. URL: www.dtic.mil/doctrine/new_pubs/jp3_13.pdf.

– (21st May 2014c). *Joint Intelligence Preparation of the Operational Environment*. Tech. rep. JP 2-01.3. Washington, D.C.: U.S. Dept of Defense.

Kadane, Joseph B (2011). *Principles of uncertainty*. Chapman & Hall.

Kahneman, Daniel, Paul Slovic and Amos Tversky, eds. (1982). *Judgment under Uncertainty*. Cambridge University Press.

Kaiser, Marie I (2011). 'The limits of reductionism in the life sciences'. In: *History and philosophy of the life sciences* 33.4, pp. 453–476.

Kanich, C. et al. (2011). 'Show Me the Money: Characterizing Spam-advertised Revenue'. In: *20th USENIX Security Symposium*. San Francisco, CA. URL: https://www.usenix.org/legacy/event/sec11/tech/full_papers/Kanich.pdf.

Katz, Jonathan (Dec. 2016). *Call for Papers: Hot Topics in the Science of Security (HoTSoS)*. http://cps-vo.org/group/hotsos/cfp.

Kent, Karen and Murugiah Souppaya (Sept. 2006). *Guide to Computer Security Log Management*. Tech. rep. SP 800-92. Gaithersburg, MD: US Dept of Commerce, National Institute of Standards and Technology.

Kent, Karen, Suzanne Chevalier, Tim Grance and Hung Dang (Aug. 2006). *Guide to Integrating Forensic Techniques into Incident Response*. Tech. rep. SP 800-86. Gaithersburg, MD: US Dept of Commerce, National Institute of Standards and Technology.

Kernighan, Brian W. and Dennis M. Ritchie (1988). *The C Programming Language*. 2nd. Upper Saddle River, NJ: Prentice Hall.

Khan, Suleman, Abdullah Gani, Ainuddin Wahid Abdul Wahab, Mustapha Aminu Bagiwa, Muhammad Shiraz, Samee U. Khan, Rajkumar Buyya and Albert Y. Zomaya (May 2016). 'Cloud Log Forensics: Foundations, State of the Art, and Future Directions'. In: *ACM Comput. Surv.* 49.1, 7:1–7:42.

Killourhy, K.S. and R.A. Maxion (2009). 'Comparing anomaly-detection algorithms for keystroke dynamics'. In: *Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on*. IEEE, pp. 125–134.

Kincaid, Harold (2011). 'Causal modelling, mechanism, and probability in epidemiology'. In: *Causality in the Sciences*. Ed. by Phyllis Illari, Federica Russo and Jon Williamson. Oxford: Oxford University Press, pp. 70–90.

Kitcher, Phillip (1981). 'Explanatory unification'. In: *Philosophy of Science* 48.4, pp. 507–531.

Koen, Billy Vaughn (2003). *Discussion of the method: Conducting the engineer's approach to problem solving*. New York: Oxford University Press.

Kossakowski, Klaus-Peter et al. (Feb. 1999). *Responding to Intrusions*. Tech. rep. CMU/SEI-99-SIM-006. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Kott, Alexander (2014). 'Towards Fundamental Science of Cyber Security'. In: *Network Science and Cybersecurity*. Ed. by Robinson E. Pino. New York, NY: Springer, pp. 1–13.

Krebs, Brian (5th Feb. 2014). *Target Hackers Broke in Via HVAC Company*. http://krebsonsecurity.com/2014/02/target-hackers-broke-in-via-hvac-company/. accessed Mar 2017.

Křetínský, Jan (12th Jan. 2018). 'Learning small strategies fast'. In: *Logic and Learning*. London, UK. URL: https://logic-data-science.github.io/Slides/Kretinsky.pdf.

Kripke, Saul A. (1965). 'Semantical analysis of intuitionistic logic I'. In: *Studies in Logic and the Foundations of Mathematics* 40, pp. 92–130.

Krol, Kat, Jonathan M Spring, Simon Parkin and M. Angela Sasse (26th May 2016). 'Towards robust experimental design for user studies in security and privacy'. In: *Learning from Authoritative Security Experiment Results (LASER)*. IEEE. San Jose, CA, pp. 21–31.

Kuhlmann, Dirk, Liqun Chen and Christopher J. Mitchell (29th Mar.–1st Apr. 2016). 'Trust and Legitimacy in Security Standardization – a new Management Issue?' In: *Interoperability for Enterprise Systems and Applications (I-ESA 16)*. Guimaraes, Portugal: ISTE Publications.

Kuhn, Thomas S. (2012). *The structure of scientific revolutions*. 4th. Introductory essay by Ian Hacking. Chicago and London: University of Chicago Press.

Kührer, Marc, Christian Rossow and Thorsten Holz (June 2014). *Paint it Black: Evaluating the Effectiveness of Malware Blacklists*. Tech. rep. TR-HGI-2014-002. Ruhr-Universität Bochum, Horst Görtz Institute for IT Security.

Labati, Ruggero Donida, Angelo Genovese, Enrique Muñoz, Vincenzo Piuri, Fabio Scotti and Gianluca Sforza (June 2016). 'Biometric Recognition in Automated Border Control: A Survey'. In: *ACM Comput. Surv.* 49.2, 24:1–24:39.

Lamport, Leslie (1983). 'What good is temporal logic?' In: *IFIP Congress*. Ed. by R.E.A. Mason. Elsevier, pp. 657–668.

– (2002). *Specifying systems: the TLA+ language and tools for hardware and software engineers*. Boston, MA, USA: Addison-Wesley.

Larman, Craig (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*. 3rd. Upper Saddle River, NJ: Prentice Hall.

Laszka, Aron, Mark Felegyhazi and Levente Buttyan (Aug. 2014). 'A Survey of Interdependent Information Security Games'. In: *ACM Comput. Surv.* 47.2, 23:1–23:38.

Lawrence Livermore National Laboratory (2016). *ROSE compiler infrastructure*. http://rosecompiler.org/.

Leech, M. (Sept. 2003). *Chinese Lottery Cryptanalysis Revisited: The Internet as a Codebreaking Tool*. RFC 3607 (Informational). RFC. Fremont, CA, USA: RFC Editor.

Leigland, Ryan and Axel W Krings (2004). 'A formalization of digital forensics'. In: *International Journal of Digital Evidence* 3.2, pp. 1–32.

Leonelli, Sabina (2009). 'Understanding in biology: The impure nature of biological knowledge'. In: *Scientific understanding: Philosophical perspectives*. Ed. by Henk W De Regt, Sabina Leonelli and Kai Eigner. Pittsburgh, PA, USA: University of Pittsburgh Press, pp. 189–209.

Letier, Emmanuel, David Stefan and Earl T. Barr (2014). 'Uncertainty, Risk, and Information Value in Software Requirements and Architecture'. In: *Proceedings of the 36th International Conference on Software Engineering*. ICSE 2014. Hyderabad, India: ACM, pp. 883–894. ISBN: 978-1-4503-2756-5. URL: http://doi.acm.org/10.1145/2568225.2568239.

Lewis, James A (1st Apr. 2008). 'Holistic Approaches to Cybersecurity to Enable Network Centric Operations'. In: *Statement before*

Armed Services Committee, Subcommittee on Terrorism, Unconventional Threats and Capabilities, 110th Cong., 2nd sess. Vol. 1. Center for Strategic and International Studies.

Li, Tao, Chunqiu Zeng, Yexi Jiang, Wubai Zhou, Liang Tang, Zheng Liu and Yue Huang (July 2017a). 'Data-Driven Techniques in Computing System Management'. In: *ACM Comput. Surv.* 50.3, 45:1–45:43.

Li, Tao et al. (Mar. 2017b). 'Data-Driven Techniques in Disaster Information Management'. In: *ACM Comput. Surv.* 50.1, 1:1–1:45.

Lin, Pei-Hung, Chunhua Liao, Daniel J. Quinlan and Stephen Guzik (Oct. 2015). 'Experiences of Using the OpenMP Accelerator Model to Port DOE Stencil Applications'. In: *11th International Workshop on OpenMP (IWOMP)*. Aachen, Germany, pp. 45–59.

Lippmann, Richard Paul and Kyle William Ingols (31st Mar. 2005). *An annotated review of past papers on attack graphs*. Tech. rep. ESC-TR-2005-054. Lexington, MA: MIT/LL.

Liu, Huan and Hiroshi Motoda (1998). *Feature selection for knowledge discovery and data mining*. New York: Springer Science & Business Media.

Liu, Peng, Wanyu Zang and Meng Yu (2005). 'Incentive-based modeling and inference of attacker intent, objectives, and strategies'. In: *ACM Transactions on Information and System Security (TISSEC)* 8.1, pp. 78–118.

Lucas Jr., Robert E. (1976). 'Econometric policy evaluation: A critique'. In: *Carnegie-Rochester conference series on public policy*. Vol. 1. Elsevier, pp. 19–46.

Lyon, G.F. (2011). *Nmap Network Scanning: The Official Nmap Project Guide To Network Discovery And Security Scanning*. Nmap Project.

M3AAWG (Nov. 2014). *M3AAWG email metrics report*. https://www.m3aawg.org/for-the-industry/email-metrics-report. accessed Jun 2017.

MITRE Corporation (19th Nov. 2010). *Science of Cyber-Security*. Tech. rep. JSR-10-102. McLean, VA: JASON Office.

– (2012). *Common Vulnerability Enumeration*. http://cve.mitre.org. last access Apr 2, 2012.

– (Dec. 2015). *Common Weakness Enumeration: A community-developed dictionary of software weakness types v2.9*. http://cwe.mitre.org.

MacKenzie, Donald A (2004). *Mechanizing proof: computing, risk, and trust*. MIT Press.

Machamer, Peter, Lindley Darden and Carl F. Craver (Mar. 2000). 'Thinking about mechanisms'. In: *Philosophy of science* 67, pp. 1–25.

Magklaras, GB and SM Furnell (2001). 'Insider threat prediction tool: Evaluating the probability of IT misuse'. In: *Computers & Security* 21.1, pp. 62–73.

Manadhata, Pratyusa K. and Jeannette M. Wing (2011). 'An attack surface metric'. In: *Transactions on Software Engineering* 37.3, pp. 371–386.

Mandiant (2013). *APT1: Exposing One of China's Cyber Espionage Units*. Tech. rep.

Mann, David (24th July 2008). *An introduction to the Common Configuration Enumeration*. Tech. rep. v1.7. McLean, VA: MITRE Corporation.

Manna, Zohar and Amir Pnueli (1992). *The temporal logic of reactive and concurrent systems*. New York: Springer-Verlag.

Maxion, Roy (Jan. 2015). 'Structure as an Aid to Good Science'. In: *Workshop on the Science of Cyber Security*. IFIP Working Group 10.4. Bristol, UK. URL: http://webhost.laas.fr/TSF/IFIPWG/Workshops&Meetings/67/Workshop-regularPapers/Maxion-Bristol_012315.pdf.

McClure, Stuart, Joel Scambray and George Kurtz (2005). *Hacking Exposed: Network Security Secrets & Solutions*. 5th. McGraw-Hill Osborne.

Meijers, Anthonie, ed. (2009). *Philosophy of Technology and Engineering Sciences*. Vol. 9. Handbook of the Philosophy of Science. Amsterdam: North-Holland.

Meng, Guozhu, Yang Liu, Jie Zhang, Alexander Pokluda and Raouf Boutaba (July 2015). 'Collaborative Security: A Survey and Taxonomy'. In: *ACM Comput. Surv.* 48.1, 1:1–1:42.

Metcalf, Leigh B and Jonathan M Spring (Sept. 2013). *Everything you wanted to know about blacklists but were afraid to ask*. Tech. rep. CERTCC-2013-39. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

– (12th Oct. 2015). 'Blacklist Ecosystem Analysis: Spanning Jan 2012 to Jun 2014'. In: *The 2nd ACM Workshop on Information Sharing and Collaborative Security*. Denver, pp. 13–22.

Metcalf, Leigh B, Dan Ruef and Jonathan M Spring (18th Oct. 2017). 'Open-source measurement of fast-flux networks while considering domain-name parking'. In: *Learning from Authoritative Security Experiment Results (LASER)*. USENIX. Arlington, VA, USA, pp. 13–24.

Metcalf, Leigh and William Casey (2016). *Cybersecurity and Applied Mathematics*. Cambridge, MA, USA: Syngress.

Meushaw, Robert, ed. (2012a). *Developing a blueprint for a science of cybersecurity*. Vol. 19:2. The Next Wave. Fort Meade, MD: U.S. National Security Agency. URL: https://www.nsa.gov/resources/everyone/digital-media-center/publications/the-next-wave/assets/files/TNW-19-2.pdf.

– (19th Dec. 2012b). *What is Security Science?* http://cps-vo.org/node/6041.

Milenkoski, Aleksandar, Marco Vieira, Samuel Kounev, Alberto Avritzer and Bryan D. Payne (Sept. 2015). 'Evaluating Computer Intrusion Detection Systems: A Survey of Common Practices'. In: *ACM Comput. Surv.* 48.1, 12:1–12:41.

Miller, George A (1956). 'The magical number seven, plus or minus two: some limits on our capacity for processing information.' In: *Psychological review* 63.2, p. 81.

Milner, Robin (1989). *Communication and concurrency*. New York: Prentice hall.

Mitchell, Sandra D. (1997). 'Pragmatic laws'. In: *Philosophy of Science* 64, S468–S479.
– (2003). *Biological Complexity and Integrative Pluralism*. Cambridge, UK: Cambridge University Press.
– (2009). *Unsimple truths: Science, complexity, and policy*. Chicago, IL: University of Chicago Press.
Mitropoulos, Sarandis, Dimitrios Patsos and Christos Douligeris (2006). 'On Incident Handling and Response: A state-of-the-art approach'. In: *Computers & Security* 25.5, pp. 351–370.
Moore, Tyler W. and Richard Clayton (Mar. 2011). 'The impact of public information on phishing attack and defense'. In: *Communications and Strategies* 81, pp. 45–68.
Morgan, Mary S. (2013). 'Nature's experiments and natural experiments in the social sciences'. In: *Philosophy of the Social Sciences* 43.3, pp. 341–357.
– (2014). 'Resituating Knowledge: Generic Strategies and Case Studies'. In: *Philosophy of Science* 81.5, pp. 1012–1024.
Morgan, Ruth M and Peter A Bull (2007). 'The philosophy, nature and practice of forensic sediment analysis'. In: *Progress in Physical Geography* 31.1, pp. 43–58.
Moriarty, K. (Nov. 2010). *Real-time Inter-network Defense (RID)*. RFC 6045 (Informational). RFC. Obsoleted by RFC 6545. Fremont, CA, USA: RFC Editor. URL: https://www.rfc-editor.org/rfc/rfc6045.txt.
– (Apr. 2012). *Real-time Inter-network Defense (RID)*. RFC 6545 (Proposed Standard). RFC. Fremont, CA, USA: RFC Editor.
Moriarty, K. and B. Trammell (Nov. 2010). *Transport of Real-time Inter-network Defense (RID) Messages*. RFC 6046 (Informational). RFC. Obsoleted by RFC 6546. Fremont, CA, USA: RFC Editor. URL: https://www.rfc-editor.org/rfc/rfc6046.txt.
Mundie, David A and Robin Ruefle (24th Aug. 2012). 'Building an incident management body of knowledge'. In: *Availability, Reliability and Security (ARES)*. IEEE. Prague, pp. 507–513.
Mundie, David A, Robin Ruefle, Audrey J Dorofee, Samuel J Perl, John McCloud and Matthew Collins (20th Nov. 2014). 'An Incident Management Ontology'. In: *Semantic Technology for Intelligence, Defense, and Security*. C4I. Fairfax, VA, pp. 62–71.
Muniz, Joseph and Aamir Lakhani (2013). *Web Penetration Testing with Kali Linux*. Birmingham, UK: Packt Publishing Ltd.
Nagel, Ernest (1979). *The structure of science: Problems in the logic of scientific explanation*. 2nd. London: Routledge & Kegan Paul.
National Academies of Sciences, Engineering, and Medicine (2017). *Foundational Cybersecurity Research: Improving Science, Engineering, and Institutions*. Ed. by Lynette I. Millett, Baruch Fischhoff and Peter J. Weinberger. Washington, DC: The National Academies Press. ISBN: 978-0-309-45529-9. DOI: 10.17226/24676.
National Cyber Security Centre (UK) (2017). *Password Guidance: Simplifying Your Approach*. https://www.ncsc.gov.uk/guidance/password-guidance-simplifying-your-approach.

National Science Foundation (2001). *Federal Cyber Service: Scholarship for Service (SFS). A Federal Cyber Service Training and Education Initiative*. Tech. rep. NSF 01-167. Arlington, VA: NSF, Directorate for education and human resources, Division of undergraduate education.

Newcombe, Chris, Tim Rath, Fan Zhang, Bogdan Munteanu, Marc Brooker and Michael Deardeuff (2015). 'How Amazon web services uses formal methods'. In: *Communications of the ACM* 58.4, pp. 66–73.

Nightingale, Paul (2009). 'Tacit Knowledge and Engineering Design'. In: *Philosophy of Technology and Engineering Sciences*. Ed. by Anthonie Meijers. Handbook of the Philosophy of Science. Amsterdam: North-Holland, pp. 351 –374.

Norton, John D. (Dec. 2010). 'There Are No Universal Rules for Induction'. In: *Philosophy of Science* 77.5, pp. 765–777.

Norton, John D (2015). 'Replicability of Experiment'. In: *Theoria* 30.2, pp. 229–248.

O'Hearn, Peter W. (2007). 'Resources, concurrency, and local reasoning'. In: *Theoretical Computer Science* 375.1, pp. 271–307.

– (2015). 'From Categorical Logic to Facebook Engineering'. In: *Logic in Computer Science (LICS)*. IEEE, pp. 17–20.

O'Hearn, Peter W. and David J. Pym (1999). 'The Logic of Bunched Implications'. In: *Bulletin of Symbolic Logic* 5.2, pp. 215–244.

O'Hearn, Peter W. and Hongseok Yang (2002). 'A Semantic Basis for Local Reasoning'. In: *Proceedings of the 5th FoSSaCS*. LNCS 2303. Springer, pp. 402–416.

O'Meara, Kyle, Deana Shick, Jonathan M Spring and Edward Stoner (Feb. 2016). *Malware Capability Development Patterns Respond to Defenses: Two Case Studies*. Tech. rep. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Obrst, Leo, Penny Chase and Richard Markeloff (25th Oct. 2012). 'Developing an Ontology of the Cyber Security Domain.' In: *Semantic Technologies for Intelligence, Defense, and Security*. Fairfax, VA: George Mason University, pp. 49–56.

Oksala, Steven, Anthony Rutkowski, Michael Spring and Jon O'Donnell (Mar. 1996). 'The structure of IT standardization'. In: *StandardView* 4.1, pp. 9–22.

Oltramari, Alessandro, Lorrie Faith Cranor, Robert J Walls and Patrick D McDaniel (Nov. 2014). 'Building an Ontology of Cyber Security'. In: *Semantic Technology for Intelligence, Defense, and Security*. Fairfax, VA, USA, pp. 54–61.

Open Science Collaboration (2015). 'Estimating the reproducibility of psychological science'. In: *Science* 349.6251, aac4716.

Oram, Andy and Greg Wilson (2010). *Making software: What really works, and why we believe it*. "O'Reilly Media, Inc."

Osorno, Marcos, Thomas Millar and Danielle Rager (June 2011). *Coordinated Cybersecurity Incident Handling: Roles, Processes, and Coordination Networks for Crosscutting Incidents*. Tech. rep. Laurel, MD: Johns Hopkins Univ, Applied Physics Laboratory.

Ou, Xinming, Sudhakar Govindavajhala and Andrew W Appel (4th Aug. 2005). 'MulVAL: A Logic-based Network Security Analyzer.' In: *USENIX Security Symposium*. Baltimore, MD.

Palmer, Dave (10th Mar. 2016). 'Self learning immune systems in theory and in practice'. In: *Statistical Aspects of Cyber-Security*. Royal Statistical Society. London.

Palmer, Gary (2001). 'A road map for digital forensic research'. In: *First Digital Forensic Research Workshop*. Utica, NY, pp. 27–30.

'Crime pattern analysis: an investigative tool' (1988). In: *Critical issues in criminal investigation*. Ed. by Michael J Palmiotto. 2nd. Pilgrimage. Chap. 2, pp. 59–69.

Pang, Min-Seok and Huseyin Tanriverdi (26th June 2017). 'Security Breaches in the U.S. Federal Government'. In: *WEIS*. La Jolla, CA.

Pasman, Hans J (2011). 'History of Dutch process equipment failure frequencies and the Purple Book'. In: *Journal of Loss Prevention in the Process Industries* 24.3, pp. 208–213.

Pearce, Michael, Sherali Zeadally and Ray Hunt (Mar. 2013). 'Virtualization: Issues, Security Threats, and Solutions'. In: *ACM Comput. Surv.* 45.2, 17:1–17:39.

Pearl, Judea (2009). *Causality*. Cambridge, UK: Cambridge University Press.

– (Nov. 2016). 'Theoretical Impediments to Machine Learning: A position paper'. In:

Pendleton, Marcus, Richard Garcia-Lebron, Jin-Hee Cho and Shouhuai Xu (Dec. 2016). 'A Survey on Systems Security Metrics'. In: *ACM Comput. Surv.* 49.4, 62:1–62:35.

Peng, Tao, Christopher Leckie and Kotagiri Ramamohanarao (Apr. 2007). 'Survey of Network-based Defense Mechanisms Countering the DoS and DDoS Problems'. In: *ACM Comput. Surv.* 39.1.

Petricek, Tomas (2017). 'Miscomputation in software: Learning to live with errors'. In: *The Art, Science, and Engineering of Programming* 1.2, p. 14.

– (Mar. 2018). 'What we talk about when we talk about monads'. In: *The Art, Science, and Engineering of Programming* 2.3.

Pfleeger, Shari and Robert Cunningham (2010). 'Why measuring security is hard'. In: *IEEE Security & Privacy* 8.4, pp. 46–54.

Piccinini, Gualtiero (2007). 'Computing Mechanisms'. In: *Philosophy of Science* 74.4, pp. 501–526.

Pita, James et al. (2011). 'Deployed ARMOR Protection: The Application of a Game-Theoretic Model for Security at the Los Angeles International Airport'. In: *Security and game theory: algorithms, deployed systems, lessons learned*. Ed. by Milind Tambe. Cambridge University Press. Chap. 4.

Pollitt, Mark (2008). 'Applying traditional forensic taxonomy to digital forensics'. In: *Advances in Digital Forensics IV*. Ed. by Indrajit Ray and Sujeet Shenoi. IFIP TC 11.9. Boston, MA, pp. 17–26.

Popper, Karl R. (1959). *The logic of scientific discovery*. London: Hutchinson.

Preimesberger, Chris (22nd Mar. 2015). *Why 'Malvertising' Has Become a Pervasive Security Risk*. URL: http://www.eweek.com/security/why-malvertising-has-become-a-pervasive-security-risk.html.

Primiero, Giuseppe, Frida J Solheim and Jonathan M Spring (2018). 'On Malfunction, Mechanisms, and Malware Classification'. In: *Philosophy & Technology* Online First.

Pritchett, Willie L and David De Smet (2013). *Kali Linux Cookbook*. Birmingham, UK: Packt Publishing Ltd.

Puvathingal, Bess J and Donald A Hantula (2012). 'Revisiting the psychology of intelligence analysis: From rational actors to adaptive thinkers'. In: *American Psychologist* 67.3, pp. 199–210.

Pym, David J., Peter W. O'Hearn and Hongseok Yang (2004). 'Possible worlds and resources: The semantics of BI'. In: *Theoretical Computer Science* 315.1, pp. 257–305.

Pym, David. 'The origins of cyberspace'. In: *Oxford handbook of cyber security*.

Pym, David, Jonathan M Spring and Peter O'Hearn (2018). 'Why separation logic works'. In: *Philosophy & Technology*. DOI: 10.1007/s13347-018-0312-8.

Radder, Hans (Aug. 2017). 'Which Scientific Knowledge is a Common Good?' In: *Social Epistemology* 31, pp. 431–450.

Rapoport, Anatol (1966). *Two-person game theory: The essential ideas*. Mineola, New York: Courier Dover Publications.

Rasmussen, R. and G. Aaron (Sept. 2012). *Global phishing survey: trends and domain name use in 2Q2012*. Tech. rep. Anti-Phishing Working Group.

Read, Stephen (1988). *Relevant Logic: A Philosophical Examination of Inference*. Basil Blackwells. URL: https://www.st-andrews.ac.uk/~slr/Relevant_Logic.pdf.

Reith, Mark, Clint Carr and Gregg Gunsch (2002). 'An examination of digital forensic models'. In: *Int. J. of Digital Evidence* 1.3, pp. 1–12.

Research Institute in Science of Cyber Security (2016). *Annual Report*. Tech. rep. London, UK: University College London.

Reynolds, John C. (2002). 'Separation Logic: A Logic for Shared Mutable Data Structures'. In: *Logic in Computer Science*. IEEE, pp. 55–74.

Robinson, J. A. (1965). 'A Machine-Oriented Logic Based on the Resolution Principle'. In: *J. ACM* 12.1, pp. 23–41.

Roesch, Martin (Nov. 1999). 'Snort: Lightweight intrusion detection for networks'. In: *Large Installation Systems Admin*. Seattle, WA: USENIX, pp. 229–238.

Roesch, Martin, Chris Green and Sourcefire (2013). 'SNORT Users Manual 2.9.4'. In: Sourcefire, Inc. Chap. 3 – Writing Snort Rules. URL: http://manual.snort.org/node27.html.

Ross, Don (2018). 'Game Theory'. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Fall 2018. Metaphysics Research Lab, Stanford University.

Ross, Ron et al. (Apr. 2013). *Security and Privacy Controls for Federal Information Systems and Organizations*. Tech. rep. SP 800-53r4.

Gaithersburg, MD: US Dept of Commerce, National Institute of Standards and Technology.

Ross, Ron, Michael McEvilley and Janet Carrier Oren (Nov. 2016). *Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems*. Tech. rep. SP 800-160. Gaithersburg, MD: U.S. National Institute of Standards and Technology.

Rossow, Christian, Christian J Dietrich, Chris Grier, Christian Kreibich, Vern Paxson, Norbert Pohlmann, Herbert Bos and Maarten Van Steen (2012). 'Prudent practices for designing malware experiments: Status quo and outlook'. In: *Security and Privacy (S&P), IEEE Symposium on*, pp. 65–79.

Rowlingson, Robert (2004). 'A ten step process for forensic readiness'. In: *International Journal of Digital Evidence* 2.3, pp. 1–28.

Roy, Arpan, Santonu Sarkar, Rajeshwari Ganesan and Geetika Goel (Feb. 2015). 'Secure the Cloud: From the Perspective of a Service-Oriented Organization'. In: *ACM Comput. Surv.* 47.3, 41:1–41:30.

Royal Society (July 2016). *Progress and research in cybersecurity: Supporting a resilient and trustworthy system for the UK*. Tech. rep. ISBN: 978-1-78252-215-7. London, UK. URL: royalsociety.org/cybersecurity.

SPSP (2017). *Society for Philosophy of Science in Practice: Mission statement*. accessed Jul 2017. URL: http://www.philosophy-science-practice.org/en/mission-statement/.

SWGDE (17th Sept. 2009). *Position on the National Research Council Report to Congress Strengthening Forensic Science in the United States: A Path Forward*. https://www.swgde.org/documents/Current%20Documents/SWGDE%20Position%20on%20the%20NAS%20Report.

Sagiv, Mooly, Thomas Reps and Reinhard Wilhelm (2002). 'Parametric shape analysis via 3-valued logic'. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 24.3, pp. 217–298.

Scarfone, Karen and Peter Mell (Feb. 2007). *Guide to Intrusion Detection and Prevention Systems (IDPS)*. Tech. rep. SP 800-94. Gaithersburg, MD: US Dept of Commerce, National Institute of Standards and Technology.

– (Oct. 2009). 'An analysis of CVSS version 2 vulnerability scoring'. In: *International Symposium on Empirical Software Engineering and Measurement*. IEEE. Lake Buena Vista, FL, pp. 516–525.

Scarfone, Karen, Murugiah Souppaya, Amanda Cody and Angela Orebaugh (Sept. 2008). *Technical Guide to Information Security Testing and Assessment*. Tech. rep. SP 800-115. Gaithersburg, MD: US Dept of Commerce, National Institute of Standards and Technology.

Schneier, Bruce (1999). 'Attack trees'. In: *Dr. Dobb's journal* 24.12, pp. 21–29.

Seacord, Robert C (2005). *Secure Coding in C and C++*. Upper Saddle Ridge, NJ: Pearson Education.

Segura, Jérôme (5th Nov. 2014). *The Proof is in the Cookie*. https://blog.malwarebytes.org/malvertising-2/2014/11/the-proof-is-in-the-cookie/.

Shannon, Gregory et al. (Feb. 2016). *Federal cybersecurity research and development strategic plan: Ensuring prosperity and national security*. Tech. rep. Washington, DC: National Science and Technology Council.

Shimeall, Timothy and Jonathan M Spring (Jan. 2014). *Introduction to Information Security: A Strategic-based Approach*. Waltham, MA: Elsevier.

Shirey, R. (Aug. 2007). *Internet Security Glossary, Version 2*. RFC 4949 (Informational). RFC. Fremont, CA, USA: RFC Editor.

Shostack, Adam and Andrew Stewart (2008). *The new school of information security*. Pearson Education.

Simon, Herbert A. (1996). *The sciences of the artificial*. 3rd. Cambridge, MA: MIT press.

Sood, Aditya K and Richard J Enbody (2013). 'Crimeware-as-a-service: A survey of commoditized crimeware in the underground market'. In: *International Journal of Critical Infrastructure Protection* 6.1, pp. 28–38.

Souppaya, Muragiah and Karen Scarfone (July 2013). *Guide to Malware Incident Prevention and Handling for Desktops and Laptops*. Tech. rep. SP 800-83r1. Gaithersburg, MD: US Dept of Commerce, National Institute of Standards and Technology.

Southers, Erroll G. (2011). 'LAX – Terror Target: The History, the Reason, the Countermeasure'. In: *Security and game theory: algorithms, deployed systems, lessons learned*. Ed. by Milind Tambe. Cambridge University Press. Chap. 2.

Spring, Jonathan M (2011a). 'Monitoring cloud computing by layer, part 1'. In: *Security & Privacy* 9.2, pp. 66–68.

– (2011b). 'Monitoring cloud computing by layer, part 2'. In: *Security & Privacy* 9.3, pp. 52–55.

– (Sept. 2013a). 'A notation for describing the steps in indicator expansion'. In: *eCrime Researchers Summit (eCRS), 2013*. IEEE. San Francisco.

– (Sept. 2013b). 'Modeling malicious domain name take-down dynamics: Why eCrime pays'. In: *eCrime Researchers Summit (eCRS)*. IEEE. San Francisco.

– (Sept. 2014). 'Toward realistic modeling criteria of games in internet security'. In: *Journal of Cyber Security & Information Systems* 2.2, pp. 2–11.

Spring, Jonathan M and Eric Hatleback (Jan. 2017). 'Thinking about intrusion kill chains as mechanisms'. In: *Journal of Cybersecurity* 3.3, pp. 185–197.

Spring, Jonathan M and Phyllis Illari (2018a). 'Building General Knowledge of Mechanisms in Information Security'. In: *Philosophy & Technology*. DOI: 10.1007/s13347-018-0329-z.

Spring, Jonathan M and Phyllis Illari (Apr. 2018b). 'Review of Human Decision-making during Incident Analysis'. In: *arXiv preprint* 1903.10080.

Spring, Jonathan M and David Pym (31st Oct. 2018). 'Towards Scientific Incident Response'. In: *GameSec*. LNCS 11199. Seattle, WA: Springer.

Spring, Jonathan M and Edward Stoner (July 2015). *CND Equities Strategy*. Tech. rep. CERTCC-2015-40. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Spring, Jonathan M, Sarah Kern and Alec Summers (27th May 2015). 'Global adversarial capability modeling'. In: *APWG Symposium on Electronic Crime Research (eCrime)*. IEEE. Barcelona.

Spring, Jonathan M, Tyler Moore and David Pym (2nd Oct. 2017). 'Practicing a Science of Security: A philosophy of science perspective'. In: *New Security Paradigms Workshop*. Santa Cruz, CA, USA.

Spring, Michael B. (2011c). 'What Have We Learned about Standards and Standardization?' In: *Homo Oeconomicus* 27.4, pp. 501–517.

Stake, Robert E (1995). *The art of case study research*. Thousand Oaks, CA: Sage.

Stamos, Alex (17th Feb. 2010). *"Aurora" Response Recommendations*. Tech. rep. iSec Partners.

Star, Susan Leigh and James R. Griesemer (1989). 'Institutional Ecology, 'Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39'. In: *Social Studies of Science* 19.3, pp. 387–420. DOI: 10.1177/030631289019003001.

Steel, Daniel (2008). *Across the boundaries: Extrapolation in biology and social science*. Oxford: Oxford University Press.

Stodden, Victoria (2015). 'Reproducing statistical results'. In: *Annual Review of Statistics and Its Application* 2, pp. 1–19.

Stoll, Clifford (1988). 'Stalking the wily hacker'. In: *Communications of the ACM* 31.5, pp. 484–497.

– (1989). *The cuckoo's egg: tracking a spy through the maze of computer espionage*. London: Pan Books.

Suárez, Mauricio (2010). 'Scientific representation'. In: *Philosophy Compass* 5.1, pp. 91–101.

Sundaramurthy, Sathya Chandran, John McHugh, Xinming Simon Ou, S Raj Rajagopalan and Michael Wesch (2014). 'An anthropological approach to studying CSIRTs'. In: *IEEE Security & Privacy* 5, pp. 52–60.

Swoyer, Chris (June 1991). 'Structural representation and surrogative reasoning'. In: *Synthese* 87.3, pp. 449–508.

Szurdi, Janos, Balazs Kocso, Gabor Cseh, Jonathan M Spring, Mark Felegyhazi and Chris Kanich (Aug. 2014). 'The long "taile" of typosquatting domain names'. In: *23rd USENIX Security Symposium*. USENIX Association. San Diego, pp. 191–206.

Takahashi, T., K. Landfield and Y. Kadobayashi (Apr. 2014). *An Incident Object Description Exchange Format (IODEF) Extension for*

Structured Cybersecurity Information. RFC 7203 (Proposed Standard). RFC. Fremont, CA, USA: RFC Editor.

Tambe, Milind (2011). *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press.

Tang, Jun, Yong Cui, Qi Li, Kui Ren, Jiangchuan Liu and Rajkumar Buyya (June 2016). 'Ensuring Security and Privacy Preservation for Cloud Data Services'. In: *ACM Comput. Surv.* 49.1, 13:1–13:39.

Taylor, Matthew E., Christopher Kiekintveld and Milind Tambe (2011). 'Evaluating Deployed Decision-Support Systems for Security: Challenges, Analysis, and Approaches'. In: *Security and game theory: algorithms, deployed systems, lessons learned*. Ed. by Milind Tambe. Cambridge University Press. Chap. 13.

Tedre, Matti (2011). 'Computing as a science: A survey of competing viewpoints'. In: *Minds and Machines* 21.3, pp. 361–387.

Tedre, Matti and Nella Moisseinen (2014). 'Experiments in computing: A survey'. In: *The Scientific World Journal* 2014, pp. 1–11.

Tempini, Niccol'o and Sabina Leonelli (May 2018). 'Concealment and discovery: the role of information security in biomedical data re-use'. In: *Social Studies of Science* In press.

The Economist (24th Dec. 2016). *The City of the Century: How Vienna produced ideas that shaped the West*. http://www.economist.com/news/christmas-specials/21712044-city-century-how-vienna-produced-ideas-shaped-west.

Thomas, Mark, Leigh Metcalf, Jonathan M Spring, Paul Krystosek and Katherine Prevost (1st July 2014). 'SiLK: A tool suite for unsampled network flow analysis at scale'. In: *IEEE BigData Congress*. Anchorage, pp. 184–191.

Thompson, Ken (Aug. 1984). 'Reflections on Trusting Trust'. In: *Commun. ACM* 27.8, pp. 761–763.

Tirpak, John A (2000). 'Find, fix, track, target, engage, assess: F2T2EA is shorthand for the operational goal the Air Force will pursue into the 21st century'. In: *Air Force Magazine* 83.7, pp. 24–29.

Trammell, B. (July 2012a). *Guidelines and Template for Defining Extensions to the Incident Object Description Exchange Format (IODEF)*. RFC 6684 (Informational). RFC. Fremont, CA, USA: RFC Editor.

– (Apr. 2012b). *Transport of Real-time Inter-network Defense (RID) Messages over HTTP/TLS*. RFC 6546 (Proposed Standard). RFC. Fremont, CA, USA: RFC Editor.

Trost, Ryan (7th Aug. 2014). 'Threat Intelligence Library - A New Revolutionary Technology to Enhance the SOC Battle Rhythm!' In: *Black Hat USA 2014*. Las Vegas, Nevada: UBM.

Turing, Alan Mathison (Nov. 1936). 'On computable numbers, with an application to the Entscheidungsproblem'. In: *Proceedings of the London mathematical society* 2.1, pp. 230–265.

Turner, Raymond and Nicola Angius (2017). 'The Philosophy of Computer Science'. In: *The Stanford Encyclopedia of Philosophy*. Ed. by

Edward N. Zalta. Spring 2017. Metaphysics Research Lab, Stanford University.

U.S. Dept of Commerce (26th Jan. 2017). *NIST Mission, Vision, Core Competencies, and Core Values*. https://www.nist.gov/about-nist/our-organization/mission-vision-values. accessed Jun 2017.

Uebel, Thomas (2016). 'Vienna Circle'. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Spring 2016. Metaphysics Research Lab, Stanford University.

Uijt de Haag, P. A. M. and B. J. M. Ale (1999). *The 'Purple Book': Guideline for Quantitative Risk Assessment*. Tech. rep. PGS3. Amsterdam: Netherlands Ministerie van Binnenlandse Zaken en Koninkrijksrelaties.

University College London (2017). *"The Research Institute in Science of Cyber Security (RISCS)"*. https://www.riscs.org.uk/. accessed Mar 6, 2017.

Ur, Blase et al. (2012). 'How does your password measure up? The effect of strength meters on password creation'. In: *USENIX Conference on Security Symposium*. Bellevue, WA: USENIX Association, pp. 65–80.

Valjarevic, Aleksandar and Hein S. Venter (Aug. 2012a). 'Harmonised digital forensic investigation process model'. In: *Information Security for South Africa (ISSA)*. IEEE, pp. 1–10.

Valjarevic, Aleksandar and Heini S. Venter (2nd–5th Sept. 2012b). 'Analyses of the State-of-the-art Digital Forensic Investigation Process Models'. In: *Southern Africa Telecommunication Networks and Applications Conference (SATNAC)*. Ed. by Stefan Scriba. 11. Fancourt, South Africa.

Van Dalen, Dirk (2004). *Logic and structure*. 4th. Springer-Verlag.

Van Eck, Wim (1985). 'Electromagnetic radiation from video display units: An eavesdropping risk?' In: *Computers & Security* 4.4, pp. 269–286.

Van Emden, Maarten H. and Robert A. Kowalski (1976). 'The semantics of predicate logic as a programming language'. In: *J. ACM* 23.4, pp. 733–742.

Verizon (2015). *2015 Data Breach Investigations Report (DBIR)*. Tech. rep. URL: http://www.verizonenterprise.com/DBIR/2015/.

– (2016). *2016 Data Breach Investigations Report (DBIR)*. Tech. rep. URL: http://www.verizonenterprise.com/verizon-insights-lab/dbir/2016/.

Vincenti, Walter G. (1990). *What engineers know and how they know it: Analytical studies from aeronautical history*. Ed. by Merritt Roe Smith. Johns Hopkins Studies in the History of Technlogy. Baltimore and London: Johns Hopkins University Press.

Vuillard, Jules (17th Mar. 2016). *Blog post on Infer-Spotify Collaboration*. http://fbinfer.com/blog/2016/03/17/collaboration-with-spotify.html. Facebook.

Wang, Ju An and Minzhe Guo (15th Apr. 2009). 'OVM: An ontology for vulnerability management'. In: *Workshop on Cyber Security and Information Intelligence Research*. 34. ACM. Oak Ridge, TN.

Wash, Rick (2010). 'Folk Models of Home Computer Security'. In: *Symposium on Usable Privacy and Security*. Redmond, WA, USA: ACM, 11:1–11:16.

Williams, Janet (Mar. 2012). *Good Practice Guide for Digital Evidence*. Tech. rep. London: Association of Chief Police Officers.

Williamson, Jon (1st June 2015). *Evaluating evidence in medicine*. https://blogs.kent.ac.uk/jonw/projects/evaluating-evidence-in-medicine/.

Willison, Robert and Mikko Siponen (2009). 'Overcoming the insider: reducing employee computer crime through Situational Crime Prevention'. In: *Communications of the ACM* 52.9, pp. 133–137.

Winn, Jane K. (2004). 'Should vulnerability be actionable? Improving critical infrastructure Computer security with trade practices law'. In: *George Mason Univ. Critical Infrastructure Protection Project Papers Vol. II*.

Winskel, Glynn (1993). *The formal semantics of programming languages: an introduction*. Cambridge, MA: MIT press.

Winterstein, Felix J, Samuel R Bayliss and George A Constantinides (2016). 'Separation logic for high-level synthesis'. In: *Transactions on Reconfigurable Technology and Systems (TRETS)* 9.2, p. 10.

Woodward, James (2003). *Making things happen: A theory of causal explanation*. Oxford, UK: Oxford University Press.

Xu, Fengwei, Ming Fu, Xinyu Feng, Xiaoran Zhang, Hui Zhang and Zhaohui Li (July 2016). 'A Practical Verification Framework for Preemptive OS Kernels'. In: *Computer Aided Verification (CAV)*. LNCS 9780. Springer. Toronto, Ontario, pp. 59–79.

Yakdan, Khaled, Sergej Dechand, Elmar Gerhards-Padilla and Matthew Smith (23rd May 2016). 'Helping Johnny to Analyze Malware'. In: *IEEE Security & Privacy (Oakland)*. San Jose, CA.

Ye, Yanfang, Tao Li, Donald Adjeroh and S. Sitharama Iyengar (June 2017). 'A Survey on Malware Detection Using Data Mining Techniques'. In: *ACM Comput. Surv.* 50.3, 41:1–41:40.

Younan, Yves, Wouter Joosen and Frank Piessens (June 2012). 'Runtime Countermeasures for Code Injection Attacks Against C and C++ Programs'. In: *ACM Comput. Surv.* 44.3, 17:1–17:28.

Zins, Chaim (2007). 'Conceptual approaches for defining data, information, and knowledge'. In: *Journal of the American society for information science and technology* 58.4, pp. 479–493.

von Plato, Jan (2016). 'The Development of Proof Theory'. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Winter 2016. Metaphysics Research Lab, Stanford University.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Thank you very much for your feedback and contribution.

Final Version as of 10th May 2019 (`classicthesis` version 1.1).