A Notation for Describing the Steps in Indicator Expansion

Jonathan M. Spring

CERT® Division; Software Engineering Institute
Carnegie Mellon University
Pittsburgh PA

***Abstract*—Indicator expansion is a process of using one or more data sources to obtain more indicators of malicious activity by identifying those related to currently known indicators. Due to the many variables in how the process is carried out, it quickly becomes difficult to capture the process that leads to an expanded set of data. Keeping track of this process is important for description to other analysts. A compact description of the process is even necessary just for the analysts doing the work to keep track of their own process and which paths have been investigated, particularly in naming files.**

This paper proposes a method of succinctly capturing the process of indicator expansion in a deterministic yet flexible and extensible manner. The target audience is analysts and investigators engaged in indicator expansion or directly consuming results therefrom.

I. Introduction

Tracking down malicious actors may involve associating indicators found to be malicious with indicators suspected to be malicious. "Indicator" is a general term for an information item that is available to the security researcher or a protection process that can be used an an identifier for a useful set of information. An indicator might be an IP address, domain name, file hash, file name, registry key, credit card number, etc. *Indicator expansion* is the process of taking in one type of indicator known to be associated with malicious activity and associating it with a new set of indicators.

Quality indicator expansion requires careful selection of various variables to limit spurious associations and maintain the utility of the process. What malware hashes attempt to contact google.com, or what domain names share an IP address with its companions on a virtual hosting service, are not useful. Such indicators with large numbers of connections often actively make the process useless. There are distinct methods for dealing with some of these difficulties. Some methods exclude a certain list of known values, others set thresholds for minimum or maximum results. Additionally, a variety of data types can be used for indicator expansion, such as IP addresses, domain names, zone files, and other logs.

The mathematical operations captured by the notation mostly include set operations and set naming conventions specific to the use case of indicator expansion. An elegant technical definition of the captured process is selective traversal of an undirected graph based on labeling of the edges and the data type of the vertices. Another benefit of the introduced notation is that it abstracts away from these definitions for the analyst not inclined toward these mathematical constructs.

Section 2 specifies a notation which can capture all of these aspects of quality indicator expansion and express them in a condensed form. Section 3 briefly describes how to use the specification to name computer files. Section 4 explains why this approach is relevant and useful to the information security and forensics communities.

The notation is agnostic to the method or tools used to carry out indicator expansion. Some tools may use set operations in python. Others may use manipulation of edges and vertices of a graph. Another may use database operations. Each of these tools could equivalently express an operation of the indicator expansion notation.

II. Specification

This section specifies the syntax for different aspects of the indicator expansion algebra. The order of operations is simply left to right. Expansions can be chained indefinitely using the principles proposed, however examples are limited to one or two expansions for the sake of simplicity.

A. Data types

Data types are identified by a capital letter from the Latin alphabet. The appropriate level of specificity is important. For example, DNS (Domain Name System) should be considered to have multiple data types, such as domain names, IP addresses, and name servers. The identifiers in network flow data are only source and destination IP address, but network flow can also contain patterns of activity which can be used for indicator expansion. While network flow and DNS are common data sources, this algebra can also be used to define custom or uncommon data sources. For example, consider a data set that links md5 hashes of malicious code samples to the domain names they resolve during dynamic analysis. These hashes are a new identifier, but these domain names captured during dynamic malware analysis can reuse the same symbol as domains from DNS analysis. The following letters are proposed for common data types:

I: IP address, such as the rdata (final) field in a DNS record of type A.

D: Domain name, such as the rname (initial) field in a DNS record. Note that this definition is particular to names that would appear in the rname or query field (initial) of a DNS record.

N: Name server, the domain name serving in its capacity as a name server. This comes from either the rdata of DNS messages of type NS (2), or from zone files.

M: MD5 hashes, usually hashes of samples of malicious code.

With just these letters, we can specify relatively complex indicator expansions. Some context is necessary to reproduce the precise results, however the method used should be clear without such context. In order to describe an expansion, the letters are juxtaposed, without any operators or other notation, such as DN or MD. The space in between letters is the operation performed. This is important later in specifying variations to that operation. The expansion DN takes a set of domain names and returns the name servers which are responsible for them. The initial set of domains, the limitations of scope and capture of the data source, and the duration of time investigated are not included in this shorthand, and would be necessary information items to precisely determine the resulting set of name servers. This context can be provided elsewhere. However, DN uniquely specifies an operation. The data source must be DNS NS records, where the rname matched one of the domains in the initial set D. The operations is not transposable; $DN \neq ND$.

The capital letters should be sufficiently specified to avoid confusion or ambiguity. For example, in the example of DN, the relation of names to name servers could come from zone files as well as passive DNS. The recommendation is to specify the source of the data as part of the data type. Consider the following further specifications:

N: Name server relation derived from passive DNS observation; refers to names in the rdata of NS (2) type records.

Z: Name server relation derived from zone files; relation is derived from the appropriate row the relevant zone's text file.

I: IP address which does not contain directionality, such as from a DNS A (1) record

T: Destination IP address, or the IP address *t*o which traffic is going, in directed traffic data sources such as flow and pcap.

F: Source IP address, or the IP address *f*rom which traffic originates, in directed traffic data sources such as flow and pcap.

T and F are used (for to and from, respectively) so that D for destination does not conflict with D for domain names.

One final data type requires the most specification on a per-use basis, due to it's changeable nature. That is a pattern of activity within a communication protocol or protocols. Patterns are generally discovered via flow, pcap, or some other passive network monitoring tool. If a particular expansion chain involves more than one pattern, each will have to be specified somehow, but in the simple case where there is only one pattern of interest, we can use the following:

P: Pattern of behavior; discovered in passive network monitoring, for example. Usually will require specification of what the pattern is.

Patterns are almost always the result of human-driven analysis or forensics. Possible expressions or specifications of a pattern may be a Snort rule on pcap files or a SiLK command-line query on netflow.

B. Limiting the expansion

It is often necessary to limit the acceptable responses from an expansion. Consider for example an operation DMD, which starts with domain names, checks which are used by malicious software, and then takes those malware samples and returns all the domains they attempt to contact. In many expansion operations a minimum and maximum number of MD5s that a domain contacts must be supplied for the operation to be useful. If only one malware sample resolved the name, it may be an uninteresting singleton. If the name was resolved by 500,000 samples, it is probably a connectivity test rather than something useful, and will result in many false positives.

There are two methods by which to express limits: absolute and relative. Absolute limits are represented by integers; they are absolute because they do not change based on the number of results. Relative limits are represented by a rational number between 0 and 1 (exclusive). The precise value of the limit changes with each expansion because the elements returned are a range of a sorted ordering of the complete results.

In either case, the precise best limits vary for each expansion situation.

1) Absolute limits: The following is an example using syntax to express absolute limits:

$$DM_2^{500}D \tag{1}$$

This equation describes two expansions. The first goes from domains to MD5s. It is unbounded, which is noted by the fact that the space between the first D and the M, which represents the expansion, has no markings. The second expansion, from MD5s back to domains, is bounded. Each hash must link to between 2-500 domains, inclusive, for any of the domains it links to to be included in the final set of domains. These limits are therefore shorthand for $2 \leq$domains-linked-to-MD5≤ 500. The operators are always $\leq$, and never $<$, to avoid possible confusion.

Numerals used in superscript or subscript always define limits to an expansion in this way. If limits are not written, the implied values are 1 and infinity. Therefore DM is equivalent to $D_1^\infty M$.

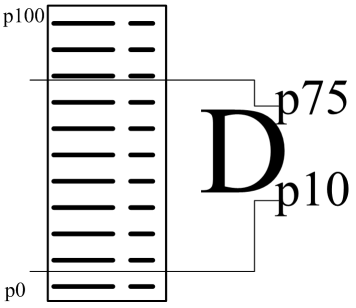
2) Relative limits: The following is an example using syntax to express relative limits:

$$D_{p1}^{p75}MD \tag{2}$$

This expansion is conceptually the same as the absolute example – it goes from domains, to MD5 hashes of malware, back to domains. The limit in this example is in the first operation—DM—rather than the second. The salient difference is that unlike absolute limits, the elements to be carried through the operation are chosen dynamically. Domains are passed through based on the number of associations per initial item relative to other initial items; in this example number of malware MD5s associated with each domain, relative to the value for all the other domains in question. We can call this property "popularity" or "relative popularity." Technically, the notation describes a *quantile* for the limitation. The p is notation for the decimal point. It is used instead of a literal "." both to call attention to the different process and to prevent confusion when naming computer files (see section III).

For example, let's say that this operation starts with 200 domains. First, find the number of hashes associated with each domain. Discard the bottom 10% of domains that are least popular (have the fewest hashes associated) and the 25% of domains that are most popular; i.e. we keep the middle section, just as in II-B1. Technically speaking, we keep a percentile range of the rank-ordered list; in this example, we discard the bottom decile and the top quartile. To determine which results to keep, we sort the list of 200 domains by popularity (number of hashes per domain). In this case, the top 50 and bottom 20 are discarded, and the hashes associated with the remaining 130 domains are carried through to the next operation. One way to visualize the process is Figure 1.

Figure 1. *Visualization of popularity (quantile) limitation*



It is not possible to determine the absolute limits before the operation. It depends on the values for each other domain in the operation and the number of domains in the initial set.

The notation represents inclusive limits, just as the absolute limits are. For absolute limits, this means the comparison operation is $\leq$ and never $<$. The percentile is inclusive in that rows that have the same value for the tested field, i.e. tie, are all included if any of them are included. Consider the operation $D^{p8}_{p2}M$ on the five domains in table I.

With 5 input domains, the operation should only keep 3 of them. The 80th percentile value for MD5s is 10; the 20th percentile is 3. Instead of arbitrarily choosing which

Domain	MD5s	
www.b.c.com	10	
www.a.b.com	10	←
q.com	6	
r.com	3	
t.com	3	←

Table I
Arrows delimit percentile limits on a sample of 5 domains, given $D^{p8}_{p2}M$.

domain to pass, the operation should use the ranks to create a numerical range of hash values and substitute this absolute range for the relative one. In this example, they would be $D^{10}_{3}M$. This operation is run with $\leq$ and not $<$, so will pass all the domains through. The result will thus be all the MD5s associated with any of the five domains.

The lowercase p, to mark that the value is a decimal and not an integer, is the only alphabetic character that may appear in limiting subscripts or superscripts. The next section defines the use of alphabetic subscripts to distinguish between different sets which are of the same type (e.g. name servers or IPs).

C. Whitelisting and naming sets

It is often useful to remove a set of known names or IPs from an expansion. The set to be subtracted usually represents a popular service or services which obscure more interesting results. For example, there may be a lot of links to domain names which are very common. One way to eliminate these names would be to subtract from the results the set of names in a list like the Alexa top 50 or 127 [1]. This process requires two features. First, it requires set subtraction, which is defined and notated in the standard set mathematics fashion. Secondly, it requires a method for defining particular sets. This is done by appending a subscript lowercase letter to describe the set to a capital letter based on the set's data type. For example:

D_a: Domains names indicated by www.alexa.com to be in the top 127 most popular sites on January 1, 2012.

See Section III for full computer file naming conventions, however note that in order to preserve a sensible convention, "e" and "p" may not be selected as the subscript when naming sets.

These specifications can also be slightly more complex. Building off the last example, we could define the following:

N_a: The name servers which result from the following expansion, using passive DNS as the source of name servers: D_aN

In order to use these in an expansion by white-listing the contents, we might subtract the set from the results. For example, $DN - N_a$. However, this is somewhat ambiguous if we want to continue expanding. The name server set that is being used should be clearly grouped as one entity if we continue. Parentheses make this easy. The following

line describes an expansion which takes a set of names, finds their name servers, removes the name servers of the most popular sites, and then finds the domains that use the remaining name servers:

$$D(N - N_a)D \tag{3}$$

Naming sets allows for a complex branching indicator expansion to be communicated. One can perform multiple expansions of arbitrary complexity and then combine them. Equation 3 is a simple example.

An investigator could also use named sets to incorporate indicators obtained from arbitrary sources. The Alexa domains are one example – the list itself is the result of a complex analysis. Another example would be the domain names released by Mandiant as part of its report on Chinese cyber espionage [2]: D_m for example. This flexibility in the algebra should allow for incorporating arbitrary indicator sets, including those obtained from sources other than network analysis.

D. Unions

Analogously to subtraction for whitelisting, two disparate analysis groups of the same base type can be unioned and operated on as a unit. This simply uses + instead of −. Thus if we have two sets of IPs, I_a and I_b, then an expansion from domains with A records to either IP would be written as equation 4.

$$DI(I_a + I_b) \tag{4}$$

First all the IP addresses are obtained, and then those IP addresses on either list are passed through as the second step. To keep the notation descriptive of the requisite process, DI is done first to obtain the IPs of the domains, and then those IPs are filtered by an operation of comparison to the named list $(I_a + I_b)$.

In some cases, it is useful to compare multiple sources of the same type of indicator. Say the analyst has three lists that are of mediocre quality. However, if any IP is on two of the three lists, it is much more likely malicious. This can be indicated by a subscript on the + operator itself. The sign $+_2$ means that an indicator is only carried through if it is on two of the sets in the operation. If there are only two sets, this means both. If there are three sets, it means any two of the three, as in equation 5. All + operators within one group of parentheses must have the same subscript.

$$D(I_a +_2 I_b +_2 I_c) \tag{5}$$

Intersection can be captured as long as the subscript of + is equal to the number of elements in the operation. For example, $(I_a +_2 I_b)$ is the intersection of two sets of IPs. Further, $(I_a +_3 I_b +_3 I_c)$ is the intersection of three sets of IPs, and so on.

E. Limiting expansion of groups

To limit the expansion when a white-list is involved, the edges of the parentheses are the space in which to put the superscripts and subscripts. These are the places where an expansion takes place — the subtraction operation is not an expansion, and no limiting notation should surround it. Therefore, including limits when parentheses are involved should be done as follows:

$$D_{10}^{1000}(N - N_a)_2^{333}D \tag{6}$$

F. Dealing with multiple instances of a data type[1]

In the previous sections, some analysts may consider using T and F for IP addresses excessively confusing. They may prefer to use lowercase subscripts instead, such as I_t or I_d as the destination IP address, I_f or I_s for source IP, and so on. This has some benefits and some draw backs. It provides clarity to data type and provides a larger name space for data sources, at the expense of clarity about some operations, clarity about specific data sets, and to a small extent clear naming in computer data files.

It provides some clarity to a canonical data type – IP address – which may be shared across data sources. The different data sources then must be indicated by lower case subscripts, and all canonical data types must be represented by upper case letters. IPs may show up in passive DNS (as rdata in A records), zone files (only for name servers the zone knows about), flow source and destination addresses, routing groupings by Autonomous System Number (ASN), and so on.

With a small number of items to be described, this problem is naturally deconflicted by the definition of the operation between two data types. There is only one data source that relates domain names to md5s in the above section, so it is not necessary to specify between a D_d (rdata in passive DNS) and a D_m (domains collected as run-time analysis from malicious code). The fact that DM and MD are unique functions is enough to avoid ambiguity. The functions DN and DZ are only unique because we have specified two distinct sources of information about name servers – passive DNS and zone files. The data sources N and Z could instead by specified as N_d and N_z. However, consider section II-C. These different data sources would not appear different than a named data set of a particular composition. Additionally, this makes an expansion function possibly difficult to read. For example, $DN_{dp10}^{p70}DN_{z5}^{250}D$ is not easy to read, as compared to $DN_{p10}^{p70}DZ_5^{250}D$. However, when naming computer files, some of this ambiguity may actually be less severe. If we make the convention that any (non 'p' or 'e') lowercase letter immediately following an uppercase letter is considered to be a subscript describing the canonical

[1]It is important to make a decision on one side or the other about the options presented here, and I would appreciate feedback as to which to go with, or strong reasons to make both options available (given the potential for possible confusion if the scheme choice is left unmarked).

data type, it does not need a '_' to mark that fact. We can thus have data elements separated more cleanly as either every capital letter or parentheses-grouped item. For example, `DNdep70_p10DNze250_5D` is not much better or worse than `DNep70_p10DZe250_5D`.

G. Lookahead assertions

The name of this technique is taken by analogy to regular expressions. In some cases, it is valuable to execute an expansion function not to keep its results, but rather to use the results to inform limits on the prior set. For example, let's say the desired output is a set of domain names. However, that final list should have benign names removed before it is passed on to an analyst. A static way to do this would be with a white-list, as above. In order to do it dynamically based on properties of the elements themselves, a lookup can be executed in order to provide a reference for comparison, even though that expansion operation is not carried through. That the final expansion is not returned is expressed by enclosing the target data type in square brackets, such as $[M]$. This operation is generally used as the final one in a series of expansions, such as:

$$D_2^{100} M D_{p15}^{p8} [M] \tag{7}$$

The first expansion goes from domains to malware with absolute limits on the number of hashes each domain may reference. Then, the resulting hashes are transformed back to domains without qualification. This produces a list of domains that we want to return to the analyst. But in order to prune it further first, we retrieve again the number of malware hashes that each domain is associated with, and remove the 20% most used and 15% least used. But instead of then expanding this reduced list into hashes, the list of domains is simply returned as is.

The lookahead assertion need not be the final element of an expansion. One could desire to expand from domain names to IP addresses, but only use domains that bear a certain relation to a set of MD5's. The expansions are from domains to MD5s and domains to IPs, and you look ahead to the MD5s to limit the set of domains used in the domain-IP expansion. The notation for this example is:

$$D_{p15}^{p8} [M]_2 \, I \tag{8}$$

The subscript 2 here represents a limit on the second expansion, between domains and IPs. By itself, that operation would be written $D_2 I$. Equation 8 is equivalent to the following series of definitions and operations:

D_m: $\quad D_{p15}^{p8} [M]$

$$D_{m\,2} I$$

III. Naming files

Computer directories do not have the same ability to manipulate text as document processors, and so some modifications to this proposal must be made for naming files with legal characters. With the simple syntax it is simple, because the proposal is only capital ASCII characters and does not include spaces between characters. In regards to the expanded meanings, it is customary to use '^' and '_' to represent superscript and subscript, respectively. Since '^' is not legal, lowercase e may be used (commonly used for "exponent" in scientific notation). The only limitation this imposes is that subscripts denoting specific sets, such as 'a' in N_a, should not be 'e' to avoid possible confusion. It shall be customary to type superscripts before subscripts. The dash, '-', can be used to indicate subtraction without a problem. Parentheses of all kinds are reserve characters. Dots, '.', can be substituted for both left and right parentheses without ambiguity. In the complex and rare case of nested parentheses that do cause ambiguity, they can be inserted in the file name escaped by '\'. Using these conventions, the most complex operation, $D_{10}^{1000}(N - N_a)_2^{333} D$, would be a file named `De1000_10.N-N_a.e333_2D`.

For lookahead assertions, a double underscore is used to represent both left and right square brackets. Lookahead assertions most commonly appear at the end of a series of expansions, and so the final double underscore may be omitted if desired without adding ambiguity. If not explicit, The brackets are assumed to close on the final element. Therefore, for the expression $D_2^{100} M D_{p1}^{p9} [M]$ there are two acceptable file names. Both `De100_2MDep9_p1__M__` and `De100_2MDep9_p1__M` are acceptable.

IV. Relevance

A common language permits clearer and faster communication. The question is then if indicator expansion is a sufficiently useful concept for security research to warrant a common parlance. Researchers often present a known set of indicators in one format or another, often IPs [3][4] or domain names [5][2]. There are also myriad publicly available block lists of domains and IPs, such as those provided by Spamhaus, SURBL, Phishtank, or Google Safe Browsing, for example, not to mention the security companies offering commercial lists and private communities developing indicators to exchange among peers, such as the REN-ISAC [6].

This example of sharing indicators among peers particularly requires each peer to be able to understand where various indicators originated from in order to make a decision on how much to trust that indicator. A method for concisely expressing the method for generating the indicator, such as that described in this paper, would facilitate this communication.

Likewise, black box machine learning algorithms could be modeled as just another indicator expansion function. For example, [7], [8], [9] all take in resource records from passive DNS and with a defined set of operations produce a list of domain names that are suspicious. These processes could also be easily assigned a marker in the expansion algebra presented above.

Acknowledgement

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

References

[1] Alexa, "Alexa internet, inc. - top sites." http://www.alexa.com/topsites, January 13, 2012.

[2] Mandiant, "Apt1: Exposing one of china's cyber espionage units," tech. rep., 2013.

[3] abuse.ch, "abuse.ch Palevo tracker," 2013.

[4] abuse.ch, "abuse.ch Zeus tracker," 2013.

[5] E. Rodionov and A. Matrosov, "The evolution of TDL: Conquering x64, revision 1.1," tech. rep., ESET, Bratislave, Slovakia, June 2011.

[6] W. Young, "Security message standardization: the beginning of the end.," in *5th Annual REN-ISAC Member Meeting*, April 9, 2011.

[7] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for DNS," in *19th Usenix Security Symposium*, 2010.

[8] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou II, and D. Dagon, "Detecting malware domains at the upper DNS hierarchy," in *20th Usenix Security Symposium*, (San Francisco, CA), 2011.

[9] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "EXPOSURE: Finding malicious domains using passive DNS analysis," *Proceedings of the Annual Network and Distributed System Security (NDSS)*, February 2011.