IET Wireless Sensor Systems

Research Article

IET Journals
The Institution of Engineering and Technology

Evolving attackers against wireless sensor networks using genetic programming

ISSN 2043-6386
Received on 28th September 2016
Revised 20th February 2017
Accepted on 23rd March 2017
E-First on 18th May 2017
doi: 10.1049/iet-wss.2016.0090
www.ietdl.org

Kinga Mrugala[1], Nilufer Tuptuk[1] ✉, Stephen Hailes[1]

[1]*Department of Computer Science, University College London, London, UK*

✉ *E-mail: nilufer.tuptuk.13@ucl.ac.uk*

Abstract: Recent hardware developments have made it possible for the Internet of Things (IoT) to be built. A wide variety of industry sectors, including manufacturing, utilities, agriculture, transportation, and healthcare are actively seeking to incorporate IoT technologies in their operations. The increased connectivity and data sharing that give IoT systems their advantages also increase their vulnerability to attack. In this study, the authors explore the automated generation of attacks using genetic programming (GP), so that defences can be tested objectively in advance of deployment. In the authors' system, the GP-generated attackers targeted publish–subscribe communications within a wireless sensor networks that was protected by an artificial immune intrusion detection system (IDS) taken from the literature. The GP attackers successfully suppressed more legitimate messages than the hand-coded attack used originally to test the IDS, whilst reducing the likelihood of detection. Based on the results, it was possible to reconfigure the IDS to improve its performance. Whilst the experiments were focussed on establishing a proof-of-principle rather than a turnkey solution, they indicate that GP-generated attackers have the potential to improve the protection of systems with large attack surfaces, in a way that is complementary to traditional testing and certification.

1 Introduction

Genetic programming (GP) has been explored and used successfully in many areas. However, there has been little published research on the use of GPs within the security field and our aim in this paper is to demonstrate that there is considerable potential in further developing this as an application area for GP and as a tool that is useful for identifying vulnerabilities.

One of the areas of current academic [1] and commercial [2] interest is the concept of Internet of Things (IoT) [3] in which objects of all kinds are networked together. The essential idea is simple: the availability of cheap integrated circuits that couple simple processors with wireless connectivity opens the possibility of networking large numbers of entities that would not form part of the traditional computing environment. This affords rich new potential marketplaces since it is possible to sense at a fine level of granularity, obtaining detailed information about environment and behaviour, and achieving more intelligent control by using actuators. However, this new technology comes with unique security risks: intimate information carries with it fears about the invasion of privacy [4], and misuse of actuators has the potential to cause physical harm [5]. At the same time, traditional security mechanisms, which rely on boundary protection (e.g. firewalls), are much less applicable since attack can be directed using the wireless link against individual (resource poor) devices by attackers who have the very considerable power of botnets at their disposal.

The sheer number of potential points of attack, the likely heterogeneity of devices, the broadcast nature of the transmission, and the mobility of devices and consequent rather unconstrained and unpredictable change in their security environment all make it hard to reason about the likely threats one might face in the IoT. The need to address the issue of security in IoT becomes even more pressing when one realises that IoT technologies are regarded as an exciting near-term prospect for use in the industrial IoT (so called Industry 4.0 [6]), healthcare, and autonomous vehicles, amongst other things. Attacks on all of these areas have the potential to impact people either individually or, through attacks on critical national infrastructure, on a regional or national basis.

At present, there have fortunately been rather few attacks on either domestic and workplace IoT systems or increasingly networked industrial automation. The reasons for this are primarily that this is a new technology, much of it bespoke, and specialist knowledge is required to launch effective attacks (which have, nevertheless, managed to cause damage to Iranian nuclear plants [7], an explosion in a blast furnace [8] and remove power from a city in the Ukraine [9], amongst others). This is changing and, whilst academics and companies are focused on the challenging problem of creating functional future systems, little is being done to explore the attack space – other than by attackers.

It is our contention that evolutionary computing techniques have the potential to revolutionise the way in we engineer future systems – from individual devices up to integrated systems – by facilitating the automated creation of attackers against which one can test (and so adapt) one's defensive strategies. Whilst an amount of work has been undertaken in using evolutionary computing in creating defensive systems for wired networks, to the best of our knowledge, this study is one of very few attempts to explore the attack space for wireless networks. Given the complexities of causing many heterogeneous systems to operate together, to say nothing of the sheer novelty of IoT and lack of prior information about attack, we believe that evolved attackers are a powerful tool that will augment more conventional forms of security analysis. Naturally they might also provide attackers with new routes to attack complex systems – in which case a failure to use them proactively becomes a failure to prepare effectively.

Here, we explore a limited problem to demonstrate the feasibility of evolving attackers. In this paper, we examine the problem of protecting wireless sensor networks (WSNs), a form of IoT system constrained not to have actuators, that use an artificial immune system (AIS) devised by Wallenta *et al.* [10]. We evolve attackers against it, and demonstrate their effectiveness by comparing our attackers to the hand-crafted attacker used to evaluate the original AIS.

The main contributions of this paper are:

- A novel approach to developing attackers for WSNs using GP.
- Improving the design of the IDS proposed in [10] to fit the proposed scenario.

- Defining a 'trustworthiness' metric in the context of the IDS proposed in [10] and using that metric to improve the overall results of the IDS.
- A full implementation and evaluation of the approach, which demonstrates that the proposed solution performs better than the original solution (in terms of both the attack and the defence) and provides the basis for further research on the topic.

In the rest of this work, we first introduce some concepts from sensor networking on which we later rely, along with a selection of other work that has been undertaken in the field (Section 2). We follow this with a statement of the hypotheses we explore in this paper (Section 3.1) and the experiments conducted to evaluate those hypotheses (Section 3). Finally, we summarise our findings (Section 4) and conclude (Section 5).

2 Background and related work

While networks are becoming very large and complex, with a very large attack surface, adversaries are becoming better educated and more persistent. As yet, there are few tools with which one can analyse such networks and search for vulnerabilities. In this paper, we describe an approach to vulnerability identification, based on existing strategies for searching large spaces. Our aim is to create a methodology that allows issues to be explored and addressed, preferably in advance of deployment but at least in advance of a live attack. Evolutionary algorithms were designed to solve problems in which there are elements of non-linearity and randomness for which it is too challenging to create closed-form solutions. They often reduce search time considerably in cases where the search space is large, and the methodology is general – it does not assume specific knowledge of the space.

The concept of evolution has been widely used in computer science as part of the subfield of evolutionary algorithms. These algorithms are implemented using the concepts in nature; the evolution usually starts from a population of randomly generated individuals, and generates a new population iteratively. At each generation, the fitness of every individual in the population is calculated using a pre-defined fitness function. During evolution, there is a selection process in which the individuals deemed to survive are chosen on the basis of their fitness level. Next, evolution operators such as crossover and mutation are used to simulate the natural reproduction and mutation of the species, restoring the population to its original level; this new population is then used for the next iteration of the algorithm. This process is repeated until the algorithm reaches a stopping condition (i.e. a satisfactory fitness level is reached indicating that an acceptable solution exists, or more than a certain number of generations have been created).

In nature, coevolution is the process of jointly evolving individuals or species that are linked at some evolutionarily important level (e.g. predator/prey): change in one population produces a selective pressure for change in the other. In biological processes these changes are genetic and are produced through natural variation, recombination, and selection: changes arise, those that are beneficial are rewarded with survival and those that are not die out. Similar forces can drive organisational change, with the pressures exerted by each side on the other punishing poor tactical choices.

There have been a number of studies proposing the use of evolutionary algorithms to achieve coevolution in security. Here we mention those related to our work. Kayacik *et al.* [11, 12] developed a framework called the evolutionary exploit generator that uses GP to evolve buffer overflow attacks, using these to modify intrusion detection systems (IDSs). The proposed model acts as a black box, and is limited to the feedback from the IDS on anomaly rates and delays. In [13], researchers introduce a coevolutionary agent-based lightweight event system for network defence, using this to coevolve attacker and defender agent strategies and evaluate potential solutions using a custom abstract computer network defence simulation. The study reports a qualitative analysis of the result data, and provides a proof of concept for the applicability of coevolution in planning for, and defending against, novel attacker strategies in computer network security. The study in [14] proposes the use of genetic algorithms to improve moving target defence, in which the defence changes the system's attack surface to limit the intelligence that can be gathered by the attacker (e.g. during the reconnaissance phase). The authors carried out prototype experiments on web servers to generate new security configurations based on the old security configurations. In [15], the authors showed how to use coevolution to evolve attackers and networks. The network was modelled as a graph-based network, in which the nodes were disconnected and re-wired based on the strategies of the defender and attacker. Coevolution has also been used to prevent faults and cascading blackouts in electric power transmission systems [16, 17]; automating red teaming for military scenarios [18]; defending against denial-of-service attacks [19]; and fault injection for smart cards [20].

Genetic algorithms have been applied to optimisation problems in WSNs, including routing protocols for energy harvesting-WSNs [21], optimal location of the sensor nodes to manage energy consumption [22], and an optimal topological balancing strategy to improve the performance of consensus-based clock synchronisation protocols [23]. However, to the best of our knowledge, there are no attempts to use evolutionary algorithms to explore the attack space for WSNs. Our work extends the existing work described above by exploring the feasibility of evolving attackers for use in WSNs that are built on both an appropriate communication paradigm, directed diffusion [24], and an intrusion detection mechanism, sensor network based AIS (SNAIS) [10], drawn from the literature. Our objective was to demonstrate the feasibility of evolving attackers for this specific domain, and we show there is good reason to suppose both that it is possible to use evolutionary algorithms in the context of security and also that there is a prospect of co-evolving attack and defence once it has been established that the generation of attackers is feasible.

In following sections, we describe both directed diffusion and SNAIS in more detail.

2.1 Directed diffusion

Directed diffusion [24] is a simple WSN routing algorithm that was specifically designed to take account of the fact that most of the traffic in sensor networks travels from a set of individual sensors to a sink. For this to occur, the complexity of an Internet Protocol stack is both unnecessary and challenging to resource as processing power, network bandwidth, and maximum packet size are severely constrained. A key element of directed diffusion is its use of data-centric (rather than identity-centric) forms of addressing. In this, the sink requests data from sensor nodes by sending an *interest* message describing the type of data it would like to receive to all other nodes. The nodes that possess that data respond to that interest using routes built through ant-like path reinforcement. More properly, directed diffusion consists of several elements:

Interest message: An interest message is sent by the sink, requesting some kind of information, for example the temperature of occupied rooms. An interest typically consists of name-value pairs, which indicate the type of data requested, the interval at which the data should be sent back to the sink, and the duration for which the interest is valid.
Data message: A data message is the response from sensor nodes, also known as sources, to the sink. A data message, like an interest, consists of name-value pairs that provide information about the particular event that triggered the data message.
Gradient: A gradient is constructed by each node when it receives an interest. It contains the information about the 'next hop' for a data message that matches this interest.
Reinforcement: A reinforcement is an interest with a higher interval sent by the sink. This interest is used to reinforce one (or a small number) of paths in the network that are the best routes for data messages.

Directed diffusion is built on two main phases: the exploratory phase and the reinforcement phase. In the exploratory phase, the

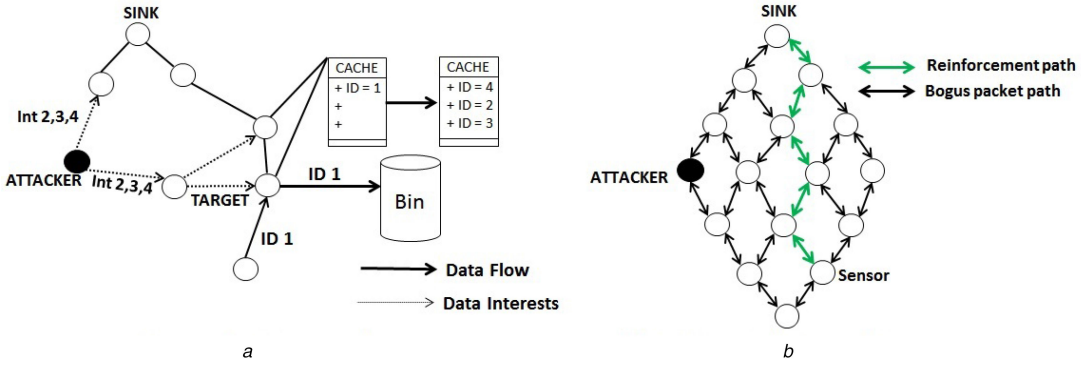

Fig. 1 *Interest cache poisoning attack [10]*
(a) Bogus cache poisoning attack, ***(b)*** Bogus interest packet propagation

sink sends an interest message to all its neighbours, who then send that interest on to their neighbours and so on, ultimately flooding the network. All nodes in the network either save that interest in their *interest cache* or update the entry of a matching interest. The nodes also create or update the gradient that is linked to the interest. The gradient stores the neighbour from which the interest was received and the data rate and expiry time specified by the interest. When the gradient expires, it is deleted from the cache. When all gradients expire, the entry is deleted from cache.

Once the gradients are established, the source starts sending data messages to the sink. In the initial phase, the source sends data to all its neighbours, who send it to all their neighbours – again flooding the network. Once the initial data message is delivered to the sink, the sink sends a reinforcement interest to the node from which it first heard the data message. This information is gathered from the data cache, which holds the information of the most recent data packets received and their origin. The next hop node uses the same criteria to forward the interest packet. It also updates its interest cache with the new gradient. Once the reinforced interest reaches the source, the reinforced path is established.

The sink can also negatively reinforce a path, which it might do if it failed to hear messages along a particular path for some period, or if a better path is found. This is done in an analogous way to positive reinforcement.

2.2 Sensor network based AIS

The SNAIS [10] is based on a dendritic cell algorithm [25, 26] and builds upon the AIS proposed in [27]. In the human immune system, a dendritic cell (DC) [28] carries antigen information to the immune system. An immature DC experiences different types of signals, which lead the DC to mature into a mature or semi-mature DC. A mature DC is considered a 'bad' cell, whereas a semi-mature DC is a 'good' cell. Chemical signals known as cytokines are produced by mature and semi-mature DCs and influence the differentiation of naïve forms of T-cells. These properties of DCs are used to design the functionality of the SNAIS.

In this paper, we consider attacks based on the interest cache poisoning attack described in [10]. In an interest cache poisoning attack, an attacker attempts to inject bogus interest entries into the interest cache of a node. Under the directed diffusion protocol, the node chooses to where to send a data message using the interest cache entries. If legitimate entries can be forcefully eliminated from the cache by an attacker, nodes will drop legitimate data packets. Fig. 1*a* shows the impact caused by the interest cache poisoning attack. The attacker sends bogus packets, and causes the cache belonging to packets on the path to fill up, thereby forcing the legitimate interests (ID 1) to be dropped to make space for bogus interests. As a result, when the requested data for ID 1 arrives, the target node drops them, since it does not have a matching gradient in the cache.

The components of SNAIS are shown in Fig. 2. In SNAIS, as in the human immune system, DCs act as detectors, and an antigen is represented by an interest packet that is stored in an interest cache. The data cache and the interest cache are sources of the signals received by the DCs. These signals indicate normal and dangerous situations, and are stored in a signal matrix. When a new interest packet (an antigen) arrives, a new immature DC is created. The immature DC copies the signals from the signal matrix into its own signal store and, when it has sufficiently many signals, it mutates into a mature (dangerous) or semi-mature (safe) DC.

Intrusion detection is used to filter out bogus packets before they enter the cache and so cause damage to the system. The filtering decision – whether a packet is believed to be dangerous (drop) or safe (pass) – is based on information from a content classifier. Fig. 3 shows an example of the content classifier and packet filter. The interests packets are classified by the content classifier, and partitioned into two possible classes A or B. As we will explain later, the classes used for this study are: benign (packets sent from the sink node) and malicious (packets sent from the attacker). During the initial stage of the IDS, the packet filter keeps all packets, and stores them in the interest cache. These packets are classified by a DC as either safe or dangerous. Then, the information about the class (A or B) and the DC decision (safe or dangerous) are used to update the packet filter, as illustrated in Fig. 3. The packet filter keeps the drop rate for packets classified into class A and class B. This reflects the history of the last N packets of each class, for example, if the 14 of the last 100 packets of class A have been classified as dangerous, the drop rate for class A would be 14%. Thus, new packets classified as class A have a probability of 14% to be dropped by the packet filter. The packet filter keeps a log of all packets that passed the filter and matured through the DCs. This information is then used to update the drop rate for each class and its DC classification.

The signal matrix stores signals used to mature the DCs. In this paper, the signal matrix has a row for each interest cache entry and a column for each signal. Signals persist until they are overwritten. The signals that are fed to the signal matrix are:

PAMP signal (PS) generated from data delivery failures.
Safe signal 1 (SS1) generated from data packet arrival. It shows that the data requested by the sink has been forwarded to the given node.
Safe signal 2 (SS2) generated when the interest cache entry expires.
Safe signal 3 (SS3) generated from normal cache update rate.
Danger signal 1 (DS1) generated from abnormal cache update rate.
Danger signal 2 (DS2) generated from cache entry overwriting.
Inflammatory cytokines (IC1) generated from the changes in gradient directions.
Inflammatory cytokines (IC2) generated from mismatches in the cache entry.

The inflammatory cytokines amplify the effects of the other three types of signals. Each signal has an assigned concentration, which varies according to the data from which the signal is generated. The exact details of how concentration is calculated for each signal can be found in [10].

Signals from the signal matrix are captured by DCs. A DC is created when a new interest cache entry is created, thus each DC is mapped to an interest cache entry. From then on, at a constant rate throughout its lifetime, a DC copies the relevant signals from the

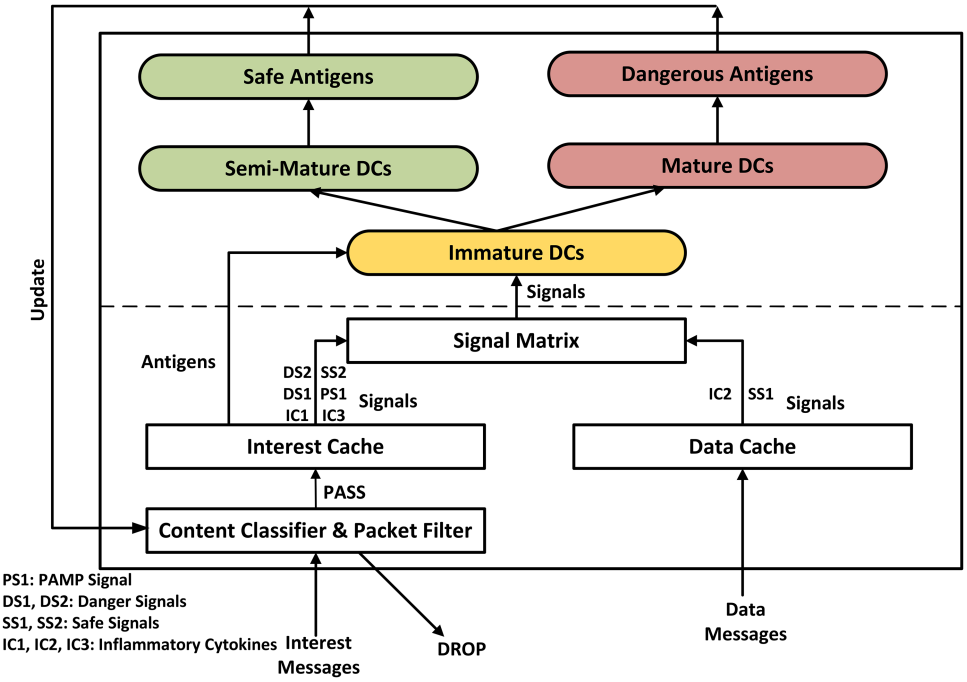

Fig. 2 *SNAIS system overview [10]*

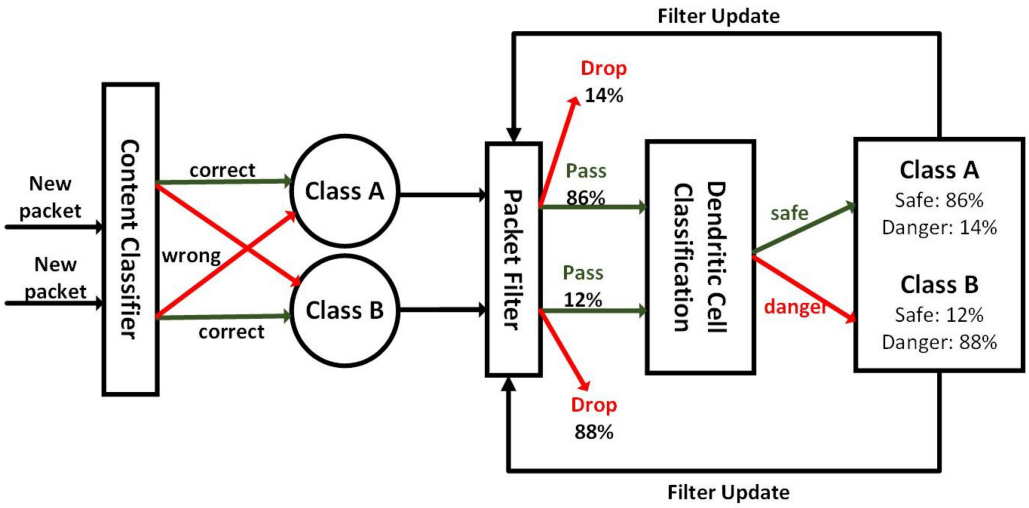

Fig. 3 *Content classifier and packet filter [10]*

signal matrix and calculates temporary output cytokines using a variation of the formula proposed in [29].

The DC matures when the interest cache entry is overwritten. When that happens, the mature (o_{mature}) and semi-mature ($o_{\text{semi-mature}}$) temporary output cytokines are compared to decide how to mature the DC. Specifically, if $o_{\text{mature}} > o_{\text{semi-mature}}$, the DC is considered mature, otherwise it is semi-mature.

The content classifier deals with the packets that have just been received. This classifies all interest packets into two classes: packets sent from the sink node(s) and packets sent from the attacker(s). In our case, the SNAIS implementation uses a simulated content classifier, at an accuracy of 80%.

As described earlier, the information from the DCs along with the information from the content classifier enables the packet filter to decide what to do with similar packets (packets that are considered the same class by the content classifier). The packet filter keeps the drop rate for each class, which reflects the history of the last N packets. That is, if from the last N packets x of them were classified as dangerous, the drop rate would be x/N.

3 Proposed model

The proposed model is illustrated in the block diagram in Fig. 4. In this section, we give an overview of the model, the details of which can be found in the following subsections.

WSN uses directed diffusion to forward packets, and SNAIS as a defence system to detect attacks. The objective of the GP attacker (malicious node in the network) is to generate interest cache poisoning attacks that evade the IDSs.

GP starts with a population of individuals (attackers) whose behaviour is randomly generated from functionality provided by a WSN node (so called terminals and functions). Subsequent populations are derived iteratively with the aim of improving the fitness of the population (effectiveness of the attack) in each generation. To achieve this, the fitness of every individual (each possible attack) in a population is calculated according to the objective of the search. So, for example, one might aim to reduce the number of packets delivered from source to sink, or to lower the precision and recall of SNAIS, or to do some combination of both. During evolution, the GP selects individuals based on their fitness level, and uses evolution operators (crossover and mutation) to simulate the natural reproduction and mutation of species. Crossover is used to generate offspring by combining randomly chosen parts of the two selected parent solutions. Mutation is used to create diversity of the population, by producing new offspring as a result of randomly altering a randomly chosen part of a selected

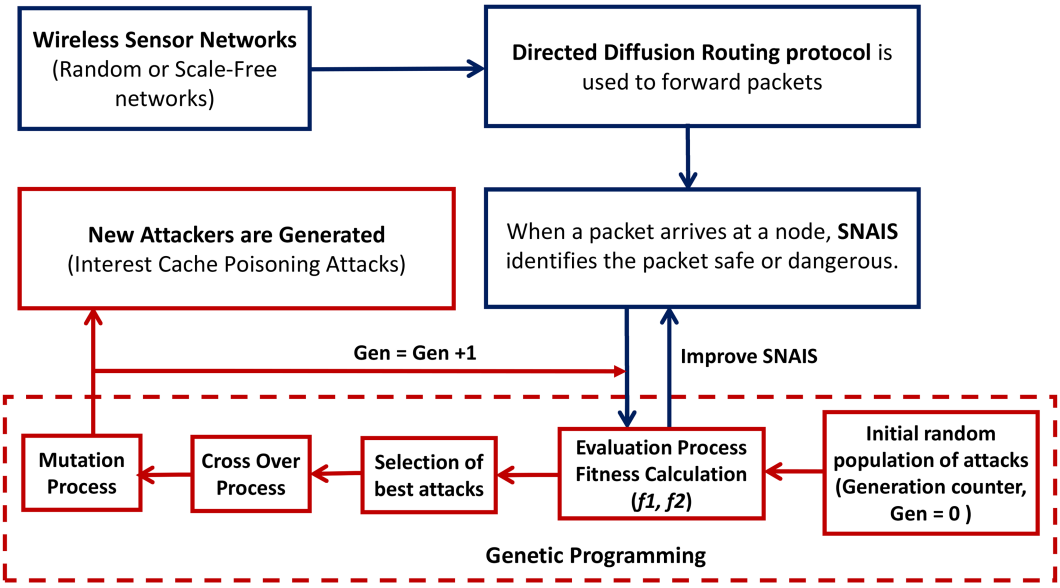

Fig. 4 *Block diagram of the proposed model*

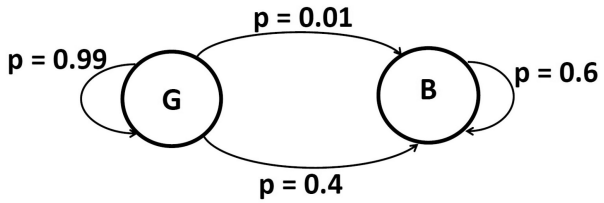

Fig. 5 *State transition diagram for Gilbert–Elliott model*

solution. The fittest individuals (best attackers) in a population are selected to form the basis of the next generation. This process is repeated until the GP reaches a pre-defined stopping condition (e.g. when a set of number of generations have been generated), and returns the best individual.

By examining the detailed behaviour of the evolved attackers, weaknesses can be determined in SNAIS and improvements put into effect to reduce the chances of successful attack.

3.1 Hypotheses

In this section, we introduce the hypotheses we explore later in the paper.

Hypothesis 1 (H_1): An evolved attacker using only the operations that are used in the interest cache poisoning attack can perform better than a basic burst attack in a variety of scenarios. A 'better' attacker is an attacker that can either:

- Trick the IDS into producing more false positives or false negatives. That is, the precision and recall of the IDS are lower. Or
- Suppress the flow of packets from the source to the sink. That is, the ratio of number of packets received by the sink to the number of packets generated by the source is lower.

Hypothesis 2 (H_2): By adding more operations that the attacker can perform, the attacker will perform better than in hypothesis 1 across a variety of scenarios.

Hypothesis 3 (H_3): It is possible to improve the AIS based on the evolutionary attacker. That is, the AIS can evolve with the attacker and become harder to break.

3.2 Wireless sensor network

In this project, the WSN was abstracted to a graph, in which nodes represent devices and edges represent link-layer connectivity. Whilst we simulate the system using the OMNET++ simulator, we chose not to use a standard CSMA/CA MAC layer protocol. Instead, we assume the existence of a slotted MAC for which slots have been pre-allocated so there is no possibility of collision. To that end, we model the delays inherent in communication by assuming a constant 100 ms delay between sending and receiving packets. Whilst this delay is large in synchronously connected networks, it is a reasonable approximation in duty-cycled (slotted) networks, in which the node is powered down much of the time and delays are relatively predictable because of the slotting. Packets can be lost as a result of corruption, a process that is modelled using a two-state Gilbert–Elliott model [29]. The Gilbert–Elliott model defines a Markov chain with two states: good state (G) and bad state (B). The packet is lost if the channel is in state B. Upon receiving a packet, the system can stay in a state or transit to the other state. The probabilities of packets transmitted while the system is in G and B states, and the transition probabilities used for the experiments are shown in Fig. 5.

To understand the impact of network connectivity, the attackers were tested on both random networks (Fig. 6*a*) with s^i nodes, where $i \in \{2\ldots7\}$, and four types of scale-free network (Fig. 6*b*) with 2^7 nodes, in two of which the attacker was positioned in the outskirts of the network and in the remainder of which the attacker was positioned in the centre. The nodes in the scale-free network follow a power law degree of distribution (Fig. 6*c*): most nodes have few links, but a small number of nodes have large number of links. Random networks have a Poisson type distribution (Fig. 6*c*): the majority of the nodes have the same number of links.

In both cases, 50 independent replicates were simulated for each network conformation, with variation in: the position of the sink, source, and attacker; the seeds used for random variables; both the number and placement of edges. Finally, we assessed the impact of misunderstanding the true network structure by evolving attackers on one type of network and assessing them on the other. The exact parameters for each simulation are defined in Table 1.

3.3 Genetic programming

The attacks were generated using GP and, in this section, we discuss the structure of the evolved attackers, details of which can be seen in Table 2.

3.3.1 Fitness functions: To explore the hypotheses in Section 3.1, two fitness functions were used. The first focuses on suppressing the number of packets delivered from source to sink (f_1); the second focuses on trying to lower the precision and recall of SNAIS (f_2)

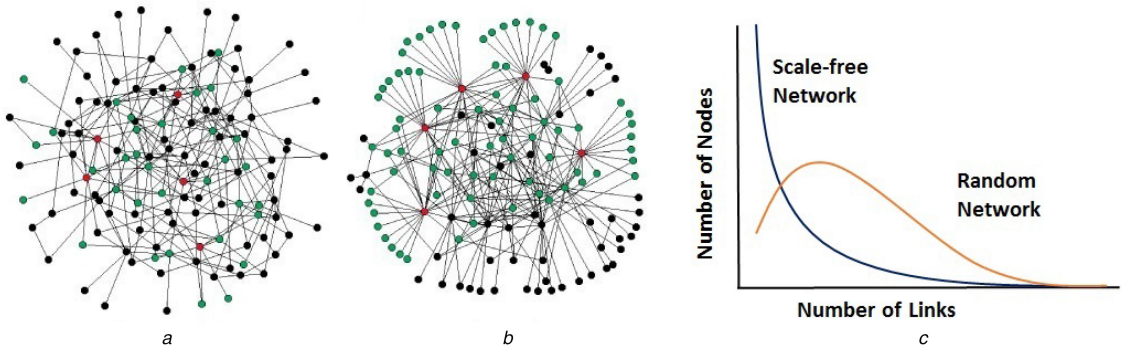

Fig. 6 *Example of*
(a) Random network [30], ***(b)*** Scale-free network [30], ***(c)*** Their degree probability distributions

Table 1 Simulation parameters

Variable	Value
size of interest cache	4
exploratory interest interval	10 s
reinforced interest interval	2 s
time between exploratory interest messages	20 s
time between reinforced interest messages	10 s
time until a negative reinforcement is sent	30 s
simulation time	1000 s
generations	50
population	100
mutation rate	0.25
crossover rate	0.7
selection	rank selector
tree depth	30
tree generation method	ramped

Table 2 GP results

Name	Network	Fitness	Hyp.	Result
suppressive	random	f_1	H_1	wait(broadcastInterest(broadcastInterest(broadcastInterest(wait(broadcastInterest(wait(broadcastInterest(wait(broadcastInterest(wait(broadcastInterest(broadcastInterest(wait(wait(0))))))))))))))))
suppressive-with-data	random	f_1	H_2	sendDataPacket(broadcastInterest(wait(wait(broadcastInterest(wait(wait(sendDataPacket(sendDataPacket(sendDataPacket(wait(wait(sendDataPacket(broadcastInterest(broadcastInterest(broadcastInterest(broadcastInterest(sendDataPacket(broadcastInterest(broadcastInterest(wait(broadcastInterest(sendDataPacket(broadcastInterest(sendDataPacket(broadcastInterest(wait(sendDataPacket(wait(sendDataPacket(0))))))))))))))))))))))))))))))))
stealthy	random	f_2	H_1	broadcastInterest(0)
stealthy-with-data	random	f_2	H_2	broadcastInterest(0)
suppressive	scale-free	f_1	H_1	broadcastInterest(broadcastInterest(broadcastInterest(broadcastInterest(wait(wait(wait(wait(wait(wait(wait(wait(broadcastInterest(broadcastInterest(broadcastInterest(broadcastInterest(0))))))))))))))))
suppressive-with-data	scale-free	f_1	H_2	sendDataPacket(wait(sendDataPacket(sendDataPacket(wait(sendDataPacket(sendDataPacket(wait(wait(sendDataPacket(sendDataPacket(sendDataPacket(wait(sendDataPacket(sendDataPacket(wait(broadcastInterest(wait(sendDataPacket(broadcastInterest(wait(broadcastInterest(broadcastInterest(sendDataPacket(broadcastInterest(sendDataPacket(wait(sendDataPacket(broadcastInterest(wait(0))))))))))))))))))))))))))))))))))
stealthy	scale-free	f_2	H_1	broadcastInterest(wait(wait(wait(wait(wait(0))))))
stealthy-with-data	scale-free	f_2	H_2	broadcastInterest(wait(wait(wait(wait(wait(0))))))

$$f_1 = \frac{\text{number of packets received by the sink}}{\text{number of packets sent by the source}},$$

$$f_2 = \frac{\text{precision} + \text{recall}}{2}$$

3.3.2 Primitive set: The primitive set consists of terminals and functions. There is only one terminal given to the GP, the integer 0. The functions given to the GP are:

- *broadcastInterest(int i)*: This function broadcasts an interest packet to all the neighbours of the attacker. The interest is broadcasted after i seconds and the type of the interest is a random 20-character alphanumeric string.
- *sendDataPacket(int i)*: This function sends a data packet to the neighbour from which it received an interest packet. The data packet has the same type as the interest it received, i.e. it is of the type that the sink requested in an interest. The data packet is sent after an i second delay.

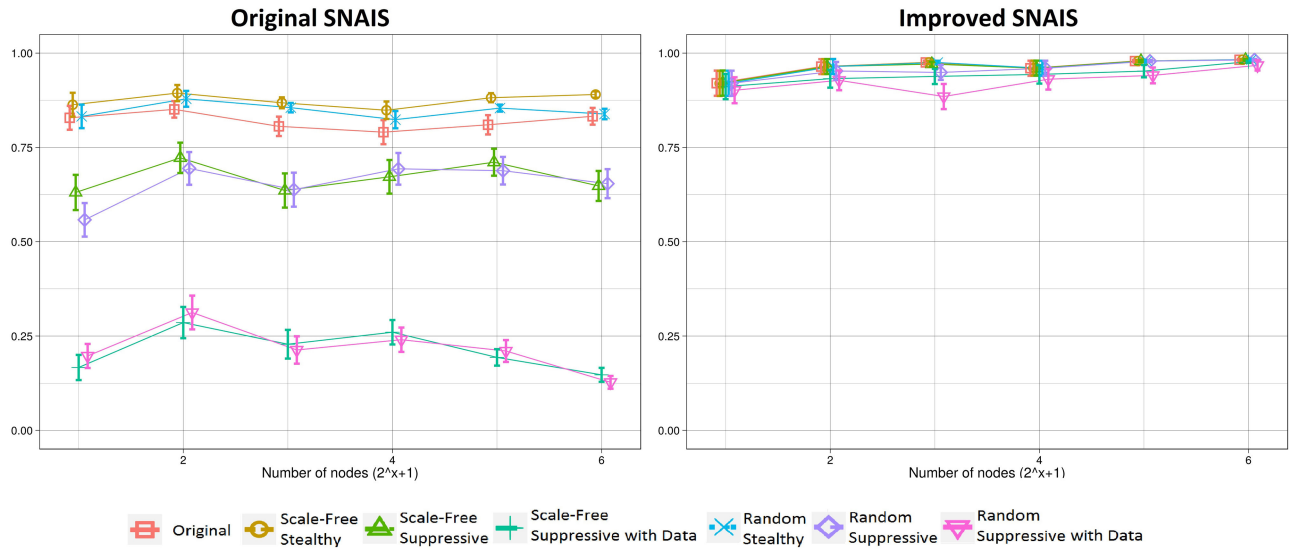

Fig. 7 *Ratio of received packets using original SNAIS and improved SNAIS for random networks*

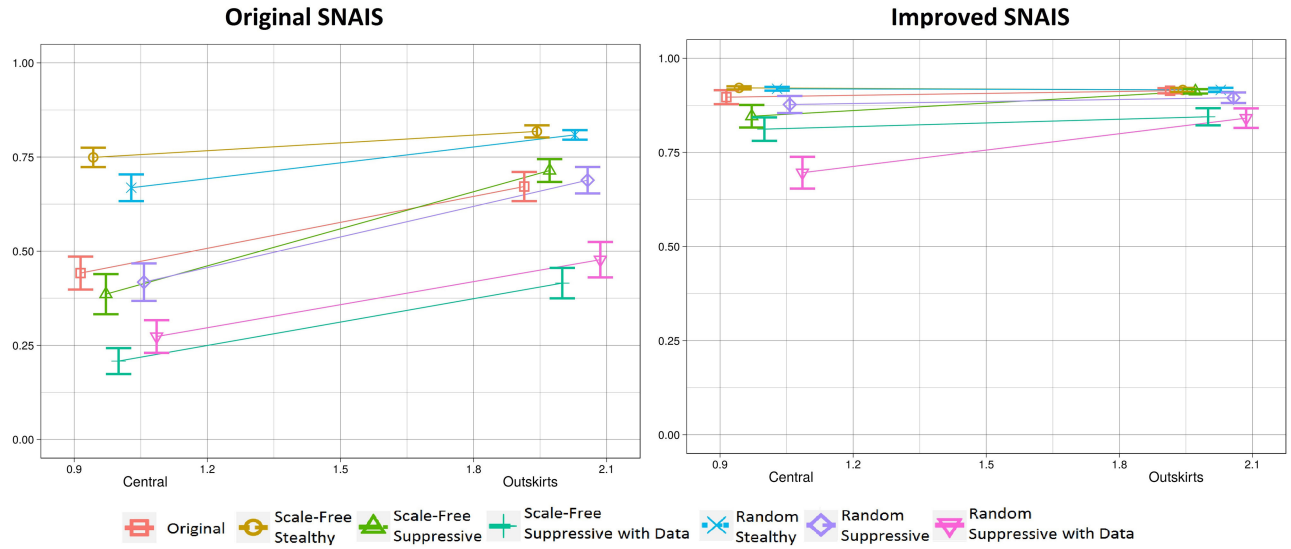

Fig. 8 *Ratio of received packets using original SNAIS and improved SNAIS for scale-free networks*

- *wait(int i)*: This function increments i by one and returns the new value. This is used to increase the amount by which to wait before sending an interest or data packet.

Not all GP runs have access to the same function set. The GP attackers used to evaluate H_1 are evolved using only *broadcastInterest* and *wait* functions, as those two are sufficient to implement an interest cache poisoning attack. For H_2, the function set is expanded to include the *sendDataPacket* function, as a result of which the attacker can pretend to be a source and thus receive reinforced interests. This can also confuse the forwarding of data packets, as SNAIS assumes that only interest packets can be malicious and that all data packets are treated equally.

4 Results

4.1 Attackers

Eight attackers were evolved, each with a different configuration of fitness functions, function sets, and networks. The attackers are described in Table 2. The goal of suppressive attacks is to block the legitimate messages being received by the sink (i.e. they use f_1). The goal of stealthy attackers is to lower the detection capabilities of SNAIS (i.e. they use f_2). The attackers labelled as 'with-data' have access to an additional function, *sendDataPacket*, and are exploring hypothesis H_2. The attacks without data do not have access to this function and thus are aimed at hypothesis H_1.

As we can see in the evolved programs shown in Table 2, the suppressive attackers send considerably more packets than the stealthy attackers, which behave very similar to a benign node. We can also see that both suppressive attackers (with and without the sendDataPacket function) evolved to the same program, which might indicate that this makes the attackers more detectable. We can also see that the stealthy attackers evolved on both networks are very similar, indicating that the structure of the network does not make a significant difference. The suppressive data attackers on both networks achieved max depth of the tree, which shows that they just try to flood the system with as many packets as possible and hope that some go through.

Figs. 7 and 8 illustrate the ratio of received packets obtained by running all attackers on both random and scale-free networks and on both original and improved SNAIS. We will discuss the results shown in those figures in more details, when we discuss the hypotheses in the later section. However, we can see that in both cases the stealthy attackers were less effective in stopping packets and the suppressive attackers performed better than original. We can also see both from Figs. 7 and 8 and from Table 3 that improvement to SNAIS resulted in a huge improvement, and none of the attackers managed to suppress a significant number of packages. Looking at Table 4 we can see that while the IDS is much better, precision and recall did not change much between the original and improved SNAIS.

4.2 Evaluating the hypotheses

4.2.1 H_1: On random networks, the suppressive attacker performed better than the original paper's attacker. It managed to suppressive on average 17% more packets than the original attacker. A Kolmogorov–Smirnov (KS) test shows that the distributions of the two sets are significantly different ($P = 1.117 \times 10 - 12$, $D = 0.3067$). On the other hand, the suppressive attacker became more detectable. The average precision is higher by 6% and the average recall is also higher by 5%. Again, the KS test indicates that the distributions of the attackers are significantly different (precision: $P < 2.2 \times 10^{-16}$, $D = 0.4333$, recall: $P < 2.2 \times 10^{-16}$, $D = 0.65$). The stealthy attacker also outperformed the original in terms of precision and recall. It reduced the recall of the system on average by 22% and the precision by 22%. The KS test for both indicates that the distributions are significantly different (precision: $P < 2.2 \times 10^{-16}$, $D = 0.6333$, recall: $P < 2.2 \times 10 - 16$, $D = 0.9833$). However, the stealthy attacker did not increase the suppression rate of the packets; instead, it decreased it by 2% ($P < 2.563 \times 10 - 7$, $D = 0.23$).

On scale-free networks, the suppressive attacker did not perform as well as it did on random networks. There is no real difference between the two attackers in terms of the suppression ($P = 0.3667$, $D = 0.13$). The suppressive attacker decreased the precision by 4% ($P = 0.07832$, $D = 0.18$), and the recall increased by 8% ($P < 2.2 \times 10 - 16$, $D = 0.71$), in comparison to the original attacker. The stealthy attacker also performed worse on scale-free networks. It reduced precision by 6% ($P = 0.002318$, $D = 0.26$), and it reduced recall by 17% ($P < 2.2 \times 10 - 16$, $D = 0.95$). The attacker seemed to perform better only when it was placed in the centre, where the precision was lower by 14% and the difference was significant ($P = 1.581 \times 10 - 6$, $D = 0.52$). However, when the attacker was in the outskirts position it performed very similar to the original attacker. As in random networks, the stealthy attacker decreased the suppression rate by 15% ($P = 0.0006709$, $D = 0.4$).

Table 3 Suppression ratio between original and improved SNAIS

Name	Network	KS test result	Improvement, %
suppressive	random	$P < 2.2 \times 10^{-16}, D = 0.8367$	30
suppressive-with-data	random	$P < 2.2 \times 10^{-16}, D = 0.88$	71
stealthy	random	$P < 2.2 \times 10^{-16}, D = 0.8067$	12
suppressive	scale-free	$P < 2.2 \times 10^{-16}, D = 0.72$	33
suppressive-with-data	scale-free	$P < 2.2 \times 10^{-16}, D = 0.76$	52
stealthy	scale-free	$P < 2.2 \times 10^{-16}, D = 0.69$	14

4.2.2 H_2: Hypothesis 2 is concerned with assessing whether the attackers from hypothesis 1 are improved by increasing the size of the function set of the GP. Interestingly, both the stealthy attacker and stealthy-with-data attacker evolved to the same program, indicating that the additional function has no effect on the detection rate. The suppressive-with-data attackers evolved using different network types performed in a very similar manner. These results are summarised in Table 3.

On the other hand, the additional function did have a significant effect on the suppression rate on the random network ($P < 2.2 \times 10 - 16$, $D = 0.7133$). Without a significant change in detectability, the suppressive-with-data attacker blocked 44% more packets than the attacker without data and 60% more packets than the original attacker.

On scale-free networks, the suppressive-with-data attacker managed to suppress 29% more packets than the suppressive attacker ($P = 4.929 \times 10 - 7$, $D = 0.54$). In addition, unlike in random networks, the suppressive-with-data attacker became more detectable than the suppressive attacker. The precision increased by 9% and recall by 1%. The differences in recall are marginal ($P = 0.1124$, $D = 0.24$); however, the differences in precision are significant ($P = 1.581 \times 10 - 6$, $D = 0.52$).

4.2.3 H_3: The focus of hypothesis 3 is to understand whether it is possible to use evolved attackers to improve defence. In this case, we examined two improvements. The first was to adjust the weights used to compute the output cytokines; the second was to introduce a trustworthiness metric for each source.

In spite of the fact that a key parameter of the AIS would appear to be the weights used to decide how a DC should mature, in this case, adjusting the weights using a basin hopping algorithm [31] proved to have a negligible effect on the overall performance of the system. However, before the evolution of attackers, this relative insensitivity was unknown and the result is therefore surprising.

The simple trustworthiness metric introduced assumed that each node had an unforgeable cryptographic identity, rendering the system immune to Sybil attacks [32, 33]. The metric, calculated by each node individually, was then implemented as follows:

i. When the DC matures a packet, the packet is added to a buffer. The buffer stores the ten most recent entries for each of the sources that the node heard from
ii. The metric for a source s is defined as

$$T(s) = 1 - \frac{\text{number of mature packets from } s \text{ in the buffer}}{10}$$

This implementation is a simplification of the kinds of metrics used in more elaborate trust-based systems [34], and, in the event, the trustworthiness metric did improve the system's performance.

Table 4 Differences between the original and improved SNAIS in terms of precision and recall

Network	Attacker	KS test		Improvement	
		Precision	Recall	Precision, %	Recall, %
random	suppressive	$P = 0.0226$	$P < 2.2 \times 10^{-16}$	4	0.5
		$D = 0.12$	$D = 0.35$		
random	suppressive-with-data	$P < 2.2 \times 10^{-16}$	$P < 2.2 \times 10^{-16}$	17	4
		$D = 0.5883$	$D = 0.54$		
random	stealthy	$P = 0.0767$	$P = 5.223 \times 10^{-9}$	0	−1
		$D = 0.34512$	$D = 0.2567$		
scale-free	suppressive	$P = 0.281$	$P = 0.1111$	3	−1
		$D = 0.14$	$D = 0.17$		
scale-free	suppressive-with-data	$P = 0.9062$	$P = 0.02431$	−2	2
		$D = 0.08$	$D = 0.21$		
scale-free	stealthy	$P = 0.0541$	$P = 5.099 \times 10^{-9}$	2	−4
		$D = 0.19$	$D = 0.47$		

With random networks, the attackers failed to suppress a notable number of the packets. As shown in Table 3, for a stealthy attacker, the effect was a 12% increase in the number of packets delivered to the sink ($P < 2.2 \times 10 - 16$, $D = 0.8067$); for a suppressive attacker, this rose to 30% ($P < 2.2 \times 10 - 16$, $D = 0.8367$); and for a suppressive-with-data attacker, this rose further to an impressive 71% ($P < 2.2 \times 10 - 16$, $D = 0.88$). Interestingly, in no case did the improved SNAIS result in a significant improvement in precision or recall. This requires further exploration, but our suggestion is that this may be because precision and recall are also dependent on the content classifier, which was constrained to an 80% accuracy.

For scale-free networks, the number of packets received by the sink was around 87%, which is slightly less than the 95% for random networks. There was a 14% increase improvement with the stealthy attacker in comparison to the original ($P < 2.2 \times 10 - 16$, $D = 0.69$); the suppressive attacker by 33% ($P < 2.2 \times 10 - 16$, $D = 0.72$); and the suppressive-with-data attacker by a still impressive 52% ($P < 2.2 \times 10 - 16$, $D = 0.76$).

As for random, there is no significant difference between the original improved SNAIS in terms of precision and recall. The differences between the original and improved SNAIS for both random and scale-free networks are summarised in Table 4.

5 Conclusion

This paper explored whether GP can be used to improve the overall security of WSNs. This was done in two stages: first an attacker was evolved using GP, then the IDS was improved by analysing the attack strategies of the evolved attackers. The results were analysed by running the system on a simulator using a variety of networks of different structures, gathering metrics from the simulation and undertaking a statistical analysis to understand whether there was any significance to observed differences. Whilst the WSN system under consideration is simple, neither it nor the AIS used in defence were specifically constructed for this research.

The results appear to be very promising. The GP attacker outperformed the original hand-crafted attacker in both suppression and detectability. As a consequence, the defending SNAIS mechanism was adapted, though with mixed and somewhat surprising results. Contrary to expectations, the precise parameterisation of the AIS algorithm appeared to matter rather little. However, the analysis of holes that the attackers were exploiting in the original SNAIS proved to have a tremendous impact on securing the system. Introducing a simple trustworthiness metric improved the packet flow from the source to the sink by 95% on random networks.

The network structure also appears to matter: the attackers on scale-free networks did not perform as well as on random networks, even when they were evolved on that topology. The dynamics of the two networks are clearly different, yet the GP attackers failed to discover this. Moreover, environmental factors over which the GP has no control also influenced the result. The position of the attacker in the network changed the detectability of the attacker without changing the suppression rate significantly. This suggests that a yet richer set of primitives, allowing the attacker to discover the form of the network and their position within it, could prove to be beneficial.

The value being mooted for IoT technologies deployed domestically, in the workplace and within infrastructure and industry is enormous. The complexity of such systems is such that even getting them to function effectively is proving challenging and, as a result, security is a concern receiving rather more belated attention than it deserves. Nowhere is this more so than for industrial control systems, in which the nature of threat and the security response to that is poorly understood even at present, let alone in the IoT-based systems under construction. In such complex environments, built from heterogeneous components that learn, adapt, and must protect themselves, there is a desperate need for tools that assist in reasoning about security. This research suggests that the generality of the evolutionary approach to the identification of attack vectors is a plausible candidate for such a tool and a strategy worthy of further exploration; indeed it is an avenue that we are actively exploring.

6 Acknowledgment

Nilufer Tuptuk acknowledges the financial support of the EPSRC under the Security Science Doctoral Training Centre at UCL, grant no. EP/G037264/1.

7 References

[1] Xu, L.D., He, Wa., Li, S.: 'Internet of things in industries: a survey', *IEEE Trans. Ind. Inf.*, 2014, **10**, (4), pp. 2233–2243

[2] Cisco: '*The Internet of Things: how the next evolution of the internet is changing everything*' (CISCO Internet Business Solutions Group (IBSG), 2011)

[3] Atzori, T., Iera, A., Morabito, G.: 'The internet of things: a survey', *Comput. Netw.*, 2010, **54**, (15), pp. 2787–2805

[4] Roman, R., Zhou, J., Lopez, J.: 'On the features and challenges of security and privacy in distributed internet of things', *Comput. Netw.*, 2013, **57**, (10), pp. 2266–2279

[5] Slay, J., Miller, M.: 'Lessons learned from the Maroochy Water Breach'. Proc. of the Critical Infrastructure Protection, 2007, pp. 73–82

[6] MacDougall, W.: 'Industrie 4.0 Smart manufacturing for the future', Germany Trade and Invest, 2014

[7] Falliere, N., Murchu, O.L., Chien, E.: 'W32.Stuxnet Dossier (version 1.4)', Symantec Security Response, 2011

[8] BSI: 'Die Lage der IT-Sicherheit in Deutschland 2014', Bundesamt fur Sicherheit in der Informationstechnik, 2014

[9] ICS-CERT: 'Alert (ICS-ALERT-14-281-01D): ongoing sophisticated malware campaign compromising ICS (update D)', Industrial Control Systems Cyber Emergency Response Team, 2016

[10] Wallenta, C., Kim, J., Bentley, P.J., *et al.*: 'Detecting interest cache poisoning in sensor networks using an artificial immune algorithm', *Appl. Intell.*, 2010, **32**, (1), pp. 1–26

[11] Kayacik, G.K., Zincir-Heywood, N.A., Heywood, M.I.: 'Can a good offense be a good defense? Vulnerability testing of anomaly detectors through an artificial arms race', *Appl. Soft Comput.*, 2011, **11**, (7), pp. 4366–4383

[12] Kayacik, G.K., Zincir-Heywood, N.A., Heywood, M.I.: 'Evolutionary computation as an artificial attacker: generating evasion attacks for detector vulnerability testing', *Evol. Intell.*, 2011, **4**, (4), pp. 243–266

[13] Rush, G., Tauritz, D.R., Kent, D.A.: 'Coevolutionary agent-based network defense lightweight event system (CANDLES)'. Proc. of the 17th Annual Conf. Companion on Genetic and Evolutionary Computation (GECCO'15), 2015, pp. 859–866

[14] John, D.J., Smith, R.W., Turkett, W.H., *et al.*: 'Evolutionary based moving target cyber defense'. Proc. of the Companion Publication of the 2014 Annual Conf. on Genetic and Evolutionary Computation (GECCO'14), 2011, pp. 1261–1268

[15] Arnold, H., Masad, D., Pagani, G.A., *et al.*: 'NetAttack: co-evolution of network and attacker'. Proc. of the Santa FeInstitute Complex Systems Summer School, 2013

[16] Service, T., Tauritz, D., Siever, W.: 'Infrastructure hardening: a competitive coevolutionary methodology inspired by neo-darwinian arms races'. Computer Software and Applications Conf., 2007, vol. **4**, pp. 101–104

[17] Tarvis, S., Tauritz, D.: 'Increasing infrastructure resilient through competitive', *New Math. Nat. Comput.*, 2009, **05**, pp. 441–457

[18] Decraene, J., Chandramohan, M., Low, M.Y.H., *et al.*: 'Evolvable simulations applied to automated red teaming: a preliminary study'. Simulation Conf. (WSC), Proc. of the 2010 Winter, 2010, pp. 1444–1455

[19] Colbaugh, R., Glass, K.: 'Predictability-oriented defense against adaptive adversaries'. IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC), 2012, pp. 2721–2727

[20] Bouffard, G., Thampi, B.N., Lanet, J.-L.: 'Detecting laser fault injection for smart cards using security automata'. Proc. Security in Computing and Communications: Int. Symp., SSCC 2013, 2013, pp. 18–29

[21] Wu, Y., Liu, W.: 'Routing protocol based on genetic algorithm for energy harvesting-wireless sensor networks', *IET Wirel. Sens. Syst.*, 2013, **3**, (2), pp. 112–118

[22] Bhondekar, A.P., Renu, V., Singla, M., *et al.*: 'Genetic algorithm based node placement methodology for wireless sensor networks'. Int. Multi Conf. of Engineers and Computer Scientists, 2009, pp. 106–112

[23] Panigrahi, N., Mohan, P.K.: 'Optimal topological balancing strategy for performance optimisation of consensus-based clock synchronisation protocols in wireless sensor networks: a genetic algorithm-based approach', *IET Wirel. Sens. Syst.*, 2014, **4**, (4), pp. 213–222

[24] Intanagonwiwat, C., Govindan, R., Estrin, D., *et al.*: 'Directed diffusion for wireless sensor networking', *IEEE/ACM Trans. Netw.*, 2002, **11**, (1), pp. 2–16

[25] Greensmith, J., Aickelin, U., Cayzer, S.: 'Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection'. Artificial Immune Systems, 2005 (LNCS, **3627**), pp. 153–167

[26] Greensmith, J., Aickelin, U., Twycross, J.: 'Articulation and clarification of the dendritic cell algorithm'. 5th Int. Conf. on Artificial Immune Systems (ICARIS), 2006, pp. 404–417

[27] Aickelin, U., Bentley, P., Cayzer, S., *et al.*: 'Danger theory: the link between AIS and IDS?'. Proc. of the Second Int. Conf. on Artificial Immune Systems, 2003 (LNCS, **2787**), pp. 147–155

[28] Steinman, R.M.: 'The dendritic cell system and its role in immunogenicity'. *Annu. Rev. Immunol.*, 1991

[29] Ebert, J.P., Willig, A.: 'A Gilbert–Elliot bit error model and the efficient use in packet level simulation'. Technical Report TKN-99-002, Telecommunication Networks Group, Technical University Berlin, 1999
[30] Albert, R., Jeong, H., Barabasi, A.-L.: 'Error and attack tolerance of complex networks', *Nature*, 2000, **406**, pp. 378–382
[31] Wales, D.J., Doye, J.P.K.: 'Global optimization by basin-hopping and the lowest energy structures of Lennard–Jones clusters containing up to 110 Atoms', *J. Phys. Chem. A*, 1997, **101**, (28), pp. 5111–5116
[32] Druschel, P., Kaashoek, M.F., Rowstron, A.I.T.: IPTPS '01: Revised Papers from the First Int. Workshop on Peer-to-Peer Syst., 2002
[33] Newsome, J., Shi, E., Song, D.*, et al.*: 'The Sybil attack in sensor networks: analysis and defenses'. Proc. of IEEE Conf. on Information Processing in Sensor Networks (IPSN), 2004
[34] Lopez, J., Roman, R., Agudo, I.*, et al.*: 'Trust management systems for wireless sensor networks: best practices', *Comput. Commun.*, 2010, **33**, (9), pp. 1086–1093