Kendrick, P, Criado, N, Hussain, A and Randles, M

A Self-Organising Multi-Agent System For Decentralised Forensic Investigations

http://researchonline.ljmu.ac.uk/8036/

Article

Citation (please note it is advisable to refer to the publisher's version if you intend to cite from this work)

Kendrick, P, Criado, N, Hussain, A and Randles, M (2018) A Self-Organising Multi-Agent System For Decentralised Forensic Investigations. Expert Systems with Applications, 102. pp. 12-26. ISSN 0957-4174

A Self-Organising Multi-Agent System For Decentralised Forensic Investigations

Phillip Kendrick[a], Natalia Criado[b], Abir Hussain[a], Martin Randles[a]

[a]*John Moores University, Liverpool, L3 3AF, United Kingdom*
[b]*King's College, London, WC2R 2LS, United Kingdom*

Abstract

As network-based threats continue to evolve more rapidly, detecting and responding to intrusion attempts in real-time requires an increasingly automated and intelligent response. This paper provides an agent-based framework for the analysis of cyber events within networks of varying sizes to detect complex multi-stage attacks. Agents are used as intelligent systems to explore domain specific and situational information showing the benefit of adaptive technologies that proactively analyse security events in real time. We introduce several algorithms to encapsulate and manage the traditional detection technologies and provide agent-based performance introspection as a mechanism to identify poorly performing systems. Our evaluation shows that the algorithms can reduce the amount of processing needed to analyse a security event by over 50% and improve the detection rate by up to 20% by introducing corrective systems to reduce false alarm rates in error-prone environments.

Keywords: Multi-Agent Systems, Cyber Security, Network Forensics.

1. Introduction

With the increasing size of networks and the requirement for organisations to share business-critical information, current cyber security solutions, such as Intrusion Detection Systems (IDS) (Mukherjee et al. (1994); Verwoerd & Hunt (2002)) and manual network forensics (Clint et al. (2002)), have been unable to adapt to modern requirements. The increasing use of mobile and wireless technologies has expanded the boundaries of the traditional network by introducing a dynamic component wherein users and devices may come and go as needed. In addition to this, the pervasive adoption of the Software As A Service paradigm, characteristic of cloud-based software that can be updated or changed with ease, can alter the network's shape by enabling or disabling services and

Email addresses: P.G.Kendrick@2012.ljmu.ac.uk (Phillip Kendrick), Natalia.Criado_Pacheco@kcl.ac.uk (Natalia Criado), A.Hussain@ljmu.ac.uk (Abir Hussain), M.J.Randles@ljmu.ac.uk (Martin Randles)

protocols. Furthermore, specific structures such as supply chain networks can increase the digital attack surface if not well protected by scalable security models (Zolfpour-Arokhlo et al. (2013)). Within this context, traditional security technologies have been unable to scale to the necessary levels due to their centralised nature and expensive hardware limiting their application to a single fixed network model (Liao et al. (2012)).

IDSs are most commonly deployed as either network-based IDSs (NIDS) or host-based IDSs (HIDS). The network-based variation has access to raw packet data (Mahoney & Chan (2001)) collected directly off the wire which provides insight into how the endpoints[1] within the network communicate with each other, while NetFlow data (Galtsev & Sukhov (2011)) consists of aggregated statistics about the packet data. The host-based variant is installed on individual machines and has access to user-specific data such as the contents of decrypted packets and biometric data (e.g., keyboard typing speed and system calls) (Rudrapal et al. (2013)). IDSs can further be categorised into signature-based, misuse-based and anomaly-based detection (Carvalho et al. (2016)) depending upon the model used for analysis. Signature-based detection uses a database of predefined examples of malicious activity to identify attacks. Signatures are defined by domain experts after a new attack has been detected to match all future instances of that attack. The manual process of defining signatures prevents the detection of previously unseen zero-day attacks making them a reactive technology. Misuse and anomaly-based detection provide an alternative by attempting to detect deviations from normal behaviour, as in the case of anomaly detection, or by learning the characteristics of abnormal behaviours and pattern matching future instances as in the case of misuse detection (Tsai et al. (2009)). Misuse and anomaly-based detection are seen as more flexible and scalable than signature-based approaches but may result in a higher number of incorrectly classified instances due to the lack of grounded knowledge (i.e., known signatures) (Zuech et al. (2015)).

In this paper, we propose a Decentralised Multi-Agent Security System (DMASS) as a scalable solution for the collection and analysis of cyber security and network forensic data (Kendrick et al. (2016)) . The proposed DMASS model adapts to changing network architectures through the introduction of new agents and is far more scalable than current IDS solutions, which will typically require expensive high-end hardware to avoid performance bottlenecks (Verwoerd & Hunt (2002)).

The proposed multi-agent approach uses a collection of agents, which are distinguished from traditional software by their autonomous implementation, to perform a variety of roles in the network security environment. In addition to performing network monitoring and attack detection, currently carried out by IDSs, this research focuses on bestowing agents with the tools to replicate

[1] An endpoint is defined as any networked device within the internal network that has an IP address and is capable of communication; examples include computers, mobile devices and networked services.

the manual forensic process, currently conducted by trained practitioners, to examining the security environment pragmatically. Bestowing agents with the ability to react to environmental changes, consider the performance of other agents and to work proactively to follow one line of investigation over another, when there is evidence to support it, is the fundamental principle included in the proposed model. This approach to digital evidence collection and cyber security is different from the traditional IDS approaches that typically use either signature, anomaly or misuse detection (Zuech et al. (2015)). The DMASS approach of using automated forensic processes increases the agent's adaptability by enabling it to respond to unforeseen circumstances where the attacker can evade traditional signature or anomaly detection.

The remainder of this paper is organised as follows. Section 2 contains an analysis of the current research in the area. Section 3 contains an outline of our DMASS model. Section 4 contains a case study to illustrate the benefits of using our approach. Section 5 formalises the concept of domains by describing how information is obtained by the agents. Section 6 describes the agent simulator developed for testing the proposed systems. Section 7 describes algorithms to aggregate agent decisions and information collected by agents to improve the efficiency and detection performance. Finally, Section 8 contains conclusions and a discussion on future work.

2. Related Research

In this section, we review existing agent-based architectures used for assessing network security. Ideally, agents should take advantage of the scalability and deployability improvements offered by multi-agent architectures and avoid common problems experienced with centralised processing, expensive hardware and rigid operating structures.

Shakarian et al. (2015) use an agent-based cyber attribution system with agent reasoning to consider multiple sources of information. Agents use information derived from various military sources to reason about factors such as the geographical location, political landscape and possible motives of an attack. This system has the advantage of using high-quality information sources which are used to make conclusions about cyber attacks. The system classifies data as either a fact or presumption, treating presumptions as unverified facts. The DMASS architecture presented in this paper is designed for use in non-military fields and recognises that data may be incorrect or missing requiring strategies to seek out information proactively. Furthermore, agents perform live data collection to gather the most up-to-date information available to avoid problems of data degradation often experienced with central information repositories.

Haack et al. (2010) use a hierarchical Multi-Agent System (MAS) for monitoring and reporting policy violations within the security environment. The system is composed of various agent types each with a particular task to perform (e.g., event monitor, alert generator, report builder). The network administrator defines a network policy for the agents to implement in a hierarchical model with instructions passed down from higher to lower-tier agents. This model is inherently centralised and suffers from many highlighted disadvantages experienced by IDSs. The top-layer agents classify security events[2] from their fixed position using data collected by the mobile lower-tier agents. The layered structure of having one class of agents to collect and another class of agents to analyse produces a static information flow which can suffer from availability downtime and performance bottlenecks. A more scalable approach, advocated in this paper, is to allow each agent to make decisions about the security events from their local viewpoints which are then brought together to classify the event as a whole.

Jahanbin et al. (2013) introduce an agent framework for forensic information gathering using three types of agents for data collection, analysis and alert generation. The authors remark that the MAS paradigm is well suited to the task of forensic data collection since agents can be dispatched to areas of the network to perform evidence gathering, a feature lacking in many IDSs that just monitor the visible network connections. Structural similarities exist with the system proposed by Haack et al. (2010), i.e., with three layers of agents forming an information pipeline from the lower layers to a higher layer agent. The decision-making process used in this model is similar to an IDS because the security decisions are made based upon the available data without consideration of possible missing data. Our proposed DMASS evaluates security events based on what data is found as well as what is missing; accounting for the possibility that the attacker may have obfuscated evidence during the attack.

Shanmugasundaram et al. (2003) develop a distributed forensics system using a hierarchical approach with multiple configurable sensors placed on the network. The system uses a variety of sensors and servers to collect and aggregate the information to derive the nature of the security event from the observable data. The system identifies the attack type based on which pieces of evidence are missing during the search. In the complex and changing cyber security environment, this approach is desirable since the lack of information does not necessarily conclude no attack has taken place.

Baig (2012) survey the current applications of MASs in critical infrastructure fields including intrusion detection. System resilience is highlighted as an important factor for multi-agent architectures where attackers could force agents offline through denial of service attacks. Agents should be able to adapt to changes in the network structure by taking into account agents which may not be able to communicate or devices that cannot be reached. Hierarchical models cannot function unless their communications pipeline is unimpaired which is unrealistic in modern expanded networks. Our proposed system uses heterogeneous agents that operate without a central control structure to withstand attacks on the availability of the system. To further improve the resilience of agents, communication paths and the performance of agents are determined at run-time based on the attack characteristics, for example, to avoid communicating over areas of the network currently under attack.

[2] A security event is defined as one or more series of suspected attacks.

Mees (2012) uses a MAS to detect Advanced Persistent Threats (APTs) (Chen et al. (2014)) by using external data sources to look up the origins of suspicious connections. Within the framework three agents were described: a consultation agent to evaluate the location of IP addresses, an analysis agent to compare suspect connections with previously seen traffic patterns, and a third agent to attempt to distinguish between human and robot connections by performing a task-specific analysis of the data. By using agents in this way, the agents were able to gather extra information that might not have been available to traditional IDSs which, for security, do not usually make external connections. This system utilises agents capable of performing multiple tasks which do not take advantage of having a greater number of specialised agents for improving scalability. Our model uses a larger number of specialised agents to encourage competition among agents to improve the overall performance.

3. Decentralised Multi-Agent Security System

In this section, an overview of the proposed decentralised agent model is provided. The system is composed of several agents $G = \{g_1, ..., g_i\}$, each capable of performing one data collection and analysis task for a particular software service[3]. A set of features F formalises information collected from services, representing information about an activity, e.g., the IP address of a connection, VPN usage, etc. To encourage agent specialisation, agents perform only one data collection task with additional agents created to interact with other services. Agents are placed close to the source they monitor (i.e., on the same network, subnet or device) to give them access to the required data streams (e.g., decrypted network data on the monitored device). In addition to increasing the observable network, this approach offloads the computational workload from a single device.

Each feature ($f \in F$) describes the type of information while the value-set V describes the range of possible values the feature may take. The heterogeneity of technologies found on the modern network is vast, and so agents are created as self-contained entities capable of processing one particular service to improve deployability. New agents introduced to the system do not require knowledge of other agents minimising exploitable dependencies. Furthermore, sensitive information is kept locally within each agent with only the data analysis of the security event shared to reduce the network footprint and possibility of leaked information. Network-layer detection often uses sources of threat intelligence to check the reputation of users, for example, by checking email senders against a list of known malicious IP addresses. The mobility of agents makes it easier for dedicated agents to perform threat intelligence monitoring without exposing the whole security solution to the external world.

[3] A service describes any system that an agent interacts with, this ranges from low-level network protocols to application services.

Each agent has a set of constraints placed upon it which must be satisfied before the agent can perform its collection and analysis task. Constraints, hereby termed *conditions*, are defined as feature-value pairs (f, v) representing information about the environment. For example, an agent performing fingerprinting of a Virtual Private Network (VPN) device could hold the two conditions that the user is located remotely (locationRemote) and that the connection is flowing over a VPN (isVPN). The results of the agent's data collection task is termed the effect and may be used to satisfy another agent's condition.

Definition 1. *A data collection action is defined as a tuple $\langle C, e\rangle$:*

- *C is the action conditions; i.e., a set of pairs (f, v) where feature $f \in F$ and value $v \in f_v$;*
- *$e \in F$ is the action effect; i.e., a feature whose value will be determined by the action.*

Definition 2. *Given a set of pairs (f, v) representing the available information about a suspicious activity, we define a data analysis action as a function returning a value between* $[0, 1]$ *representing the probability of the suspicious activity being malicious.*

Definition 3. *Given a set I formed by pairs (f, v) representing the available information about a suspicious activity, and a data collection action $\langle C, e\rangle$ we define that action conditions are satisfied if for all $(f, v) \in C$, $(f, v) \in I$; and not satisfied otherwise.*

An *extended data collection task* describes the process of several agents performing independent data collection and analysis tasks in conjunction with each other to analyse more of the security event. The mapping between an agent's effect (output) and another agent's condition (input) enables agents to discover each other during the extended data collection task. With each additional agent included in the extended data collection task, more information is gathered and analysed emulating the manual forensic process of gathering information based on what is already known. Typically, at the start of this process little is known about the attack, but as agents collect more data which will satisfy more conditions, a greater number of agents will be able to participate since their conditions will become satisfied.

A communication module allows for the transfer of information between agents, the main use of which is to send the *report*, which is a grouping of the agent's ID, effect and the local decision about the maliciousness of the data, between agents. The report is generated and then sent to the next agent whose conditions have been satisfied by the effects already known. Each agent may add to the collective information (i.e., the report) by aggregating data with the current knowledge before passing it to the next agent. The transfer of the aggregated set of reports facilitates the build-up and propagation of information within the agent network. Figure 1 illustrates the information flow with data collected from information sources during stages (1) and (3) by two separate agents. The newly collected data is broadcast to other agents at stage (2). Agents whose conditions are satisfied by the effect (Agent-2) respond by requesting the full report of all previous data, following this, the current agent must select one of the responding agents to receive the report. If multiple agents request the report, only one agent will receive it, but the previous requests will be carried over to the next agent. In the case of multiple requests for the report, the agent with the highest reputation (based on previous conformity in voting with the group decision) is used to decide which agent should receive it. Where previous requests are carried over, they are pushed into a last-in-first-out stack structure. This promotes a depth-first search of the network where new requests are handled first with the reasoning that a series of successful agent investigations into a particular domain is more likely to discover the source of a network breach whereas a breadth-first search is a less specific search for information. Figure 2 shows a similar extended data collection process occurring within our multi-agent simulator; this process can be viewed as connections made between the agent nodes. To improve privacy and reduce the communications overhead, the feature type (f) of the currently known effects can be broadcast during the report propagation stage rather than effect values $(f, v) \in e$. Agents whose condition types are fulfilled by the currently known effect types would then be able to request the report containing both the feature type and the values. Using this strategy, agents whose condition types do not match the current effect types are not exposed to the information which increases the privacy of the system.

Decisions are made by individual agents based on their local view of the network and role to monitor a specific feature or technology. The security community has developed a variety of detection mechanisms for specific attacks, ranging from signatures of malicious behaviour to anomaly and misuse detection systems to distinguish between normal and abnormal activity. However, the scope of these individual detection systems is often small and provides the attacker with an opportunity to evade detection and gain access through an alternate route. The local decision modules within each agent make use of these traditional security mechanisms but together define a broader view of the network by combing the local views into a more comprehensive global view.

Definition 4. *Given a security event and agent identity, a local report R_{Local} is defined as a tuple consisting of $\langle eid, ts, g, (f, v), p\rangle$ where:*

- *eid is a unique event identifier;*
- *ts is the events timestamp;*
- *$g \in G$ is the agent's identity;*

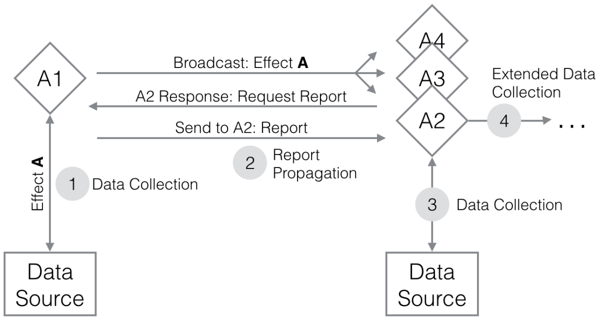

Figure 1
Flow diagram for the extended data collection task using agents (A1-A4) and data sources.

- (f, v) *is a feature-value pair corresponding to the output of the data collection action performed by agent g;*
- $p \in [0, 1]$ *is the agent's analysis of the suspicious activity; i.e., the probability of the suspicious activity being malicious.*

During the extended data collection process, once no more agents can participate because of unsatisfied conditions, the aggregated set of reports is analysed by the last agent to receive them using one of the voting algorithms discussed in Section 7. The result of this analysis, termed the *final global decision*, is where the final classification for the security event as a whole is made. Following the classification, the final global decision is sent to all participating agents so that they may compare their performance to that of the group's decision.

Definition 5. *A global report* R_{Global} *is defined as a set of local reports* $\{R_{Local_1},$ $..., R_{Local_n}\}$ *containing the information collected by different agents participating in the same extended data collection process as well as a unique event ID and timestamp.*

Definition 6. *Given a global report* R_{Global} *representing the local decisions made by the agents participating in an extend data collection process, the global decision is a function returning a value between* $[0, 1]$ *representing the collective judgement about the maliciousness of the investigated activity.*

The local network can be viewed as a time-varying network DeLellis et al. (2017); de Lellis et al. (2017); da Gama Batista et al. (2015) that experiences changes in its topology as devices are added and removed. With the growth of mobile technologies and Bring Your Own Device To Work (BYODTW) networks

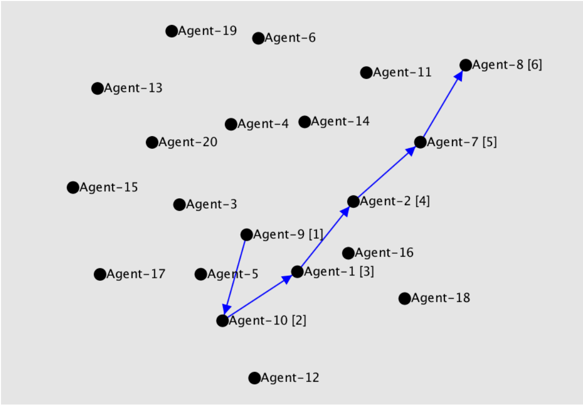

Figure 2
A smulated data collection task showing the propagation of information between agents. Agents are selected based on the currently known information about the attack and condition information.

can change shape and size on a regular basis. While the size variability of the core network may not change as often as the auxiliary mobile subnet, our agent-based approach adapts to the changing shape of a network by penalising the use of agents that perform poorly over time. Note that it is possible for an agent to recover from non-selection over time as their specialisation is required regardless of performance. Algorithm 4, Section 5.1 details the algorithm for weighing agent decisions by moving the participation requests to the end of the queue. This mechanism is used so that if a network changes and has the effect of making a particular agent perform poorly (because the source of information has changed), the agent's participation in future extended data collection tasks will be diminished over time.

4. Case Study

To illustrate the advantages of using our model, several agents were deployed in a live networking environment to detect and respond to cyber reconnaissance activities (refer to Table 1). Typically there are several stages to a network breach, of which the first is information gathering and reconnaissance (Herzog (2010)). Port scanning is a commonly undertaken activity during a cyber attack with specifically crafted packets sent to hosts to discover information about the underlying technologies (Kikuchi et al. (2009)). While port scanning is associated with the early stages of a penetration attempt, it can also be utilised legitimately by the network's administrator for housekeeping activities. The current solution for detecting port scanning activities is to use a firewall or centralised IDS to monitor the network traffic, specifically looking for the specially crafted packets that match various IDS signatures (Roesch & Green (2016)). The methodology of monitoring a network and making judgements about the

Table 1
Agents used within the case study.

Agent	Condition	Effect	Location
Agent-1	Multiple connections on different ports	IP address of offending host	Host-A (192.168.56.101)
Agent-2	IP address belongs to Host-B	Process owner identity	Host-B (192.168.56.102)
Agent-3	External IP address	Information about external IP	Host-C (192.168.56.103)
Agent-4	Any IP address	Information about IP location	Host-C (192.168.56.103)

event as a whole is used commonly throughout all areas of cyber security but ignores additional pieces of evidence that could be collected and used to consider the attack more intelligently. Furthermore, it does not take into consideration the situational data about the attacker or attack which could be used to analyse the event more accurately. To illustrate the benefits of using an agent-based approach, a port scanning scenario is detailed.

Four agents $\{g_1, ...g_4\} \in G$ are implemented and installed on three hosts joined by ethernet connection (shown in Figure 6). The functions of the deployed agents are (1) monitoring port scan attempts; (2) monitoring system privileges and process owners; (3) checking IP addresses against a known blacklist; and (4) monitoring the origin of connections (refer to Table 1). Using the system of conditions and effects outlined in Section 3, the agents collectively perform an extended data collection task to gather more information about the event.

Event 1 (Port Scan Initiated). A port scan attack, initiated from Host-B against Host-A, is detected by Agent-1 whose condition is satisfied by the presence of multiple connections made on different ports from the same host. Whereas a signature-based approach may immediately detect and block the connections made, the DMASS agent-based approach begins an extended data collection task to learn more about the event. The monitored information about this event (stored on a per agent basis as R_{Local}) is captured and stored within the global report (R_{Global}) and the effect (the IP address of the initiating host) is broadcast to find additional agents that can work with the data. Figure 3 shows the communications stack after Agent-1 has broadcast its effect to the network[4].

Event 2 (second broadcast). Of the agents whose condition is satisfied and that responded to the broadcast (i.e., Agents 2 and 4), Agent-4 is selected by Agent-1 to receive the global report next. In this example, the selection of Agent-4

[4] In Kendrick et al. (2016) we proposed an agent-based interaction model for the sending of messages between agents that would be utilised in this example.

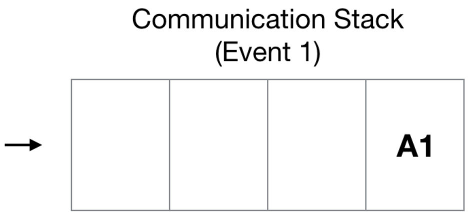

Figure 3
The communications stack after event 1 showing Agent-1 (A1) is the only agent to have communicated.

over Agent-2 was made based on the reputation of both agents (See Figure 4). Agent-4's functions are more broad than Agent-2 and so is more likely to be involved in a greater number of extended data collection tasks and thus have an increased reputation. In this way, agents whose functions are based around general information gathering are typically prioritised over rarely used agents. This prioritisation is preferred as it gives alternate agents an opportunity to search for conflicting evidence rather than making classifications based on minimal evidence. The function of Agent-4 is to determine the location (i.e., local or remote) of a host given an IP address. Upon finding the location of the IP address is local, Agent-4 broadcasts this information as its effect. Note that the request made by Agent-2 is carried forward as part of the report structure passed between selected agents.

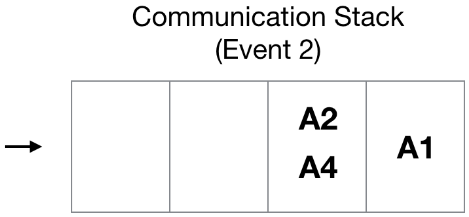

Figure 4
The communications stack after event 2 showing Agents 2 and 4 have responded to Agent-1's broadcast.

Event 3 (event classification). In this example with few agents, there are no more responses the the broadcast made by Agent-4 in the previous step (note that Agent-2 does not respond as its request has been carried forward). Agent-3, whose function is to externally gather information about remote IP addresses, does not respond to the broadcast in this event as its condition (an external IP address) is not fulfilled. Instead, the request carried forward by Agent-2, which is used to analyse the owner of suspicious processes where the IP address of the suspected host matches the IP address of Host-B (the same host it is located on), is selected as the next agent to receive the report. Upon analysis, Agent-2 finds that the port scan did originate from Host-B, however, the process is owned by an administrative account and so classified as routine maintenance rather than a malicious event.

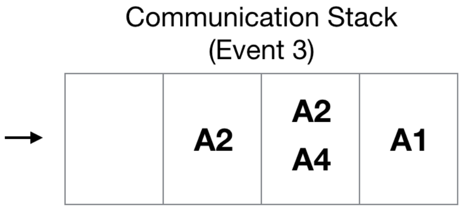

Figure 5
The communications stack after event 3 showing Agents 2 being selected to participate in the extended data collection task from its previous request made in event 2.

In this example, the IP address effect information is required by two agents (Agent-2 and Agent-3) for the fulfilment of their conditions. The function of Agent-3 is to check remote IP addresses against known blacklists requiring the IP address to be remote. The function of Agent-2 is to monitor the process privileges on Host-B specifically, and as such, requires that the IP address is local and belong to Host-B (stored as a local report R_{Local}). Following the IP address being identified as belonging to the local network and referencing Host-B, the information was passed to Agent-2 rather than Agent-3 whose conditions were satisfied by the available information. The information collected by each agent, in the form of their individual local reports, is joined to form the global report (R_{Global}) for communication between agents. Upon analysing the process information for Host-B, Agent-2 found that the popular port scanning tool, Nmap, had been recently used by an administrator account, this contextual information led to the event being classified as non-malicious and no further action against hosts involved was taken. While this limited scale example includes few agents per host, the architecture supports many agents per device capable of performing a variety of functions to allow more complex data collection and analysis tasks to take place. Furthermore, the application example highlights the need for more in-depth analysis of security events rather than the surface detection and prevention of network connections when they appear to match illegal signatures. Traditional approaches that would associate the port scan activity with being malicious would have blocked the activity immediately rather than performing extended data collection to discover more about the service owner as in the case of our DMASS. The strength and novelty of the proposed system are the agents that can gather contextual information surrounding a security event, prioritise search, and evaluate the performance of other agents. These mechanisms aid the agents in analysing the security event in a more informed manner.

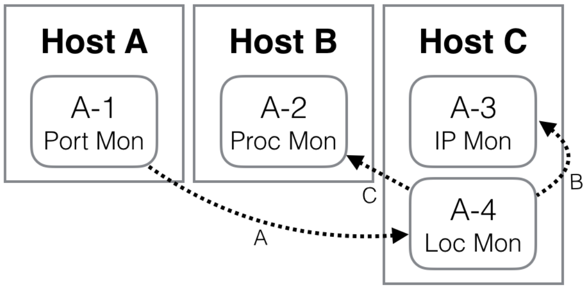

Figure 6
Example using agents (A-1,2,3,4) distributed across three devices (Hosts-A,B,C) to detct and investigate the cause of a port scan attempt.

5. Environment Modelling using Domains

The concept of domains is introduced to describe the complexities found within the cyber security environment more accurately. Domains modelling enables agents to weigh the progress of the extended data collection task against expected attack patterns.

By using specialised agents, with each one capable of performing one data collection task, it is possible to explore the underlying network structure by examining the agent's relationships with each other. Consider that each agent has a set of conditions and one effect, with the effects fulfilling the conditions for other agents; when modelled, this produces a graph of connections showing the relationship between the agents and by extension the relationship between the underlying services (Obes et al. (2010)). Figure 7 shows this graph with nodes representing agents, the colouring representing groups of agents belonging to the same service, and edges representing which effect satisfies which condition.

The choice to use specialised agents that perform only a single data collection action results in a system where there may be multiple agents that work with a single technology. Complex technologies will justify the use of many agents performing different types of tasks, for example, an email server may have several agents for performing incoming, outgoing, and spam monitoring. At the beginning of an extended data collection task when little is known about an event, the search will begin with general data collection. As more information is gathered, the agents will collect more attack-specific information as the agents find evidence about the attack.

Hence, the purpose of domains is to use the underlying network structure information to more accurately and efficiently evaluate the collected information by considering whether the detected attack is plausible. Plausibility is a measure of whether the agent's analysis of an event makes sense given common attack patterns. Typically an attacker will attempt to gain access to a system through the path of least resistance (Fernandez et al. (2007)), if agents from multiple domains monitoring several technologies detect the presence of an attack, the event is recognised as having an unusually wide scope. Furthermore, we expect the attack to spread through connected neighbouring technologies since network links exist between them, this is grounded in empirical evidence and is often used in attack graph generation (Durkota & Kiekintveld (2015)). Agents use these predictable and well-defined movement patterns to determine the plausibility of monitored events.

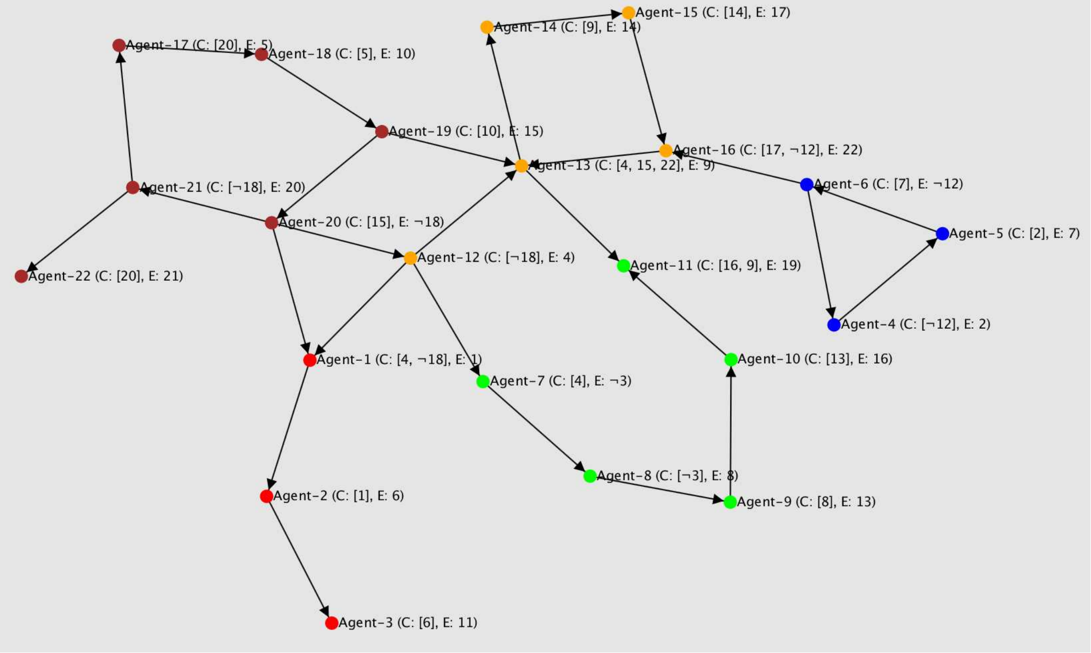
Agent−15 (C: [14], E: 17)
Agent−14 (C: [9], E: 14)
Agent−17 (C: [20], E: 5)
Agent−18 (C: [5], E: 10)
Agent−19 (C: [10], E: 15)
Agent−16 (C: [17, ¬12], E: 22)
Agent−13 (C: [4, 15, 22], E: 9)
Agent−6 (C: [7], E: ¬12)
Agent−21 (C: [¬18], E: 20)
Agent−20 (C: [15], E: ¬18)
Agent−5 (C: [2], E: 7)
Agent−12 (C: [¬18], E: 4)
Agent−11 (C: [16, 9], E: 19)
Agent−22 (C: [20], E: 21)
Agent−4 (C: [¬12], E: 2)
Agent−10 (C: [13], E: 16)
Agent−1 (C: [4, ¬18], E: 1)
Agent−7 (C: [4], E: ¬3)
Agent−8 (C: [¬3], E: 8)
Agent−9 (C: [8], E: 13)
Agent−2 (C: [1], E: 6)
Agent−3 (C: [6], E: 11)

Figure 7
The simulated network environment containing several technological domains (represented by colours). Conditions (C) and Effects (E) are numerically represented to abstract the data they represent.

Given the example of a supply chain network, used for facilitating communication between multiple businesses (Smith et al. (2007)), the typical IDS solution would offer limited protection against more complex multi-stage cyber attacks (Tobergte & Curtis (2013)). By participating in the supply chain network, the larger corporation, which may have a more robust cyber security solution, puts itself at risk since the attacker is more likely to penetrate through a less protected supply networks and pivot to the real target of the attack. The concept of domains can accurately model this threat with the proposed DMASS to monitor the attacker attempts to penetrate specific technologies as they pivot to the real target. The scalability of the DMASS supports the placement of multiple agents on the expanded network allowing the extended data collection task to take place on the network as a whole.

The example of the supply chain network is descriptive of an Advanced

Persistent Threat (APT) which is more likely to see the use of novel zero day exploits that currently have no matching signature for detection. The proposed system of using domains to analyse the context of agent alerts is used as a mechanism to detect zero day attacks by analysing the movements of an attacker by the footprints that are left behind. An attacker employing APT techniques and zero day exploits may still leave detectable footprints that can only be properly analysed in the global context where the evidence is brought together. To this end, the goal of the domains approach is to facilitate the context-aware processing of information to detect stealthy and novel attacks.

5.1. Proposed Algorithms

Using the concept of domains, we provide several algorithms for the analysis of the network environment. The algorithms make significant improvements to the agent's efficiency in analysing the network by introducing concepts such as agent memory, agent performance analysis and evaluating the plausibility of attacks. Additionally, improvements to the detection accuracy are made by avoiding the inclusion of poorly performing agents in the extended data collection task. An evaluation of the proposed algorithms is provided in Section 7.

Baseline (Refer to Algorithm 1). The baseline algorithm iteratively processes the available information without making use of optimisation techniques and is presented for comparison. To decide the global decision for the group, the algorithm tallies the local decisions from each agent and takes the highest number of votes for either malicious or innocuous as the final event classification. As a result, the algorithm performance is contingent on the individual agent's performance in evaluating the collected information. In cases where the agents cannot accurately collect and analyse the digital evidence, the algorithm will incorrectly classify the event as no corrective mechanisms are used to counter poor performance. This model is similar to the current generation of security technologies that use a variety of detection mechanisms but do not consider the detection alerts as a whole to evaluate whether the monitored attacks form a plausible attack pattern.

Series Weighting (Refer to Algorithm 2). The first proposed algorithm is introduced to give weighted bonuses to malicious votes that appear within an unbroken series of agent decisions. Given that the product of an extended data collection task can be viewed as a tuple of local decisions ($R_{Global} = \langle eid, ts, \{g_n, (f_n, v_n), p_n\}\rangle$), the decisions must be aggregated to produce the final event classification. The proposed algorithm uses the corrective measure of giving extra weighting to series of malicious decisions that appear in sequence when modelled using the domains graph. Group cohesion is measured by the number of similar decisions made by agents from the same or neighbouring domains. High group cohesion during the decision-making process indicates an abundance of evidence which can be relied on more. This algorithm uses these concepts to favour groups of agents that vote in the same way. By default, each

Algorithm 1 Baseline Algorithm

```
Require:
    $R_{Global}$ the global report containing a set of local reports $R_{Local}$ corresponding to a security event.
    $\alpha \in [0,1]$ the minimum threshold required for a classification of malicious.
Define:
    $T_{malicious} \leftarrow 0$ a tally of malicious votes.
    $T_{innocuous} \leftarrow 0$ a tally of innocuous votes.

for $\langle eid, ts, g, (f, v), p \rangle \in R_{Global}$ do        ▷ Iterate and tally local decisions
    if $p \geq \alpha$ then
        $T_{malicious} \leftarrow (T_{malicious} + 1)$
    else
        $T_{innocuous} \leftarrow (T_{innocuous} + 1)$
if $T_{malicious} \geq T_{innocuous}$ then        ▷ Return global decision
    return $\langle eid, decision : malicious \rangle$
else
    return $\langle eid, decision : innocuous \rangle$
```

decision has a value of 1, but for each additional vote of *malicious* after the first, an additional weighting is given. Additional weighting for *malicious* but not *innocuous* votes is given because it is expected that the majority of agents will vote innocuous as attacks typically target only a subset of network domains. Within compromised networks, the volume of legitimate non-malicious traffic will typically outweigh the volume of attack traffic resulting in a high false negative rate when most agents correctly identify no attack. This algorithm corrects the problem of agents being outweighed by providing additional weights to the decisions of cohesive groups.

Series Weighting with Cut-off (Refer to Algorithm 3). To increase the efficiency of the Series Weighting voting algorithm the algorithm was further extended to improve the efficiency by allowing agents to autonomously decide the point at which enough information to make the global decision had been collected. If during the extended data collection task, a sufficient amount of evidence is found supporting one decision over the other, agents can decide to make the global decision earlier without consulting all agents that can participate[5]. Making quicker global decisions improves real-time detection by reducing the number of agents involved. To search the entire domains model for indicators of compromise, involving all agents in the analysis, would ensure that the event classification is made using all of the available information, however, would be operationally

[5] Currently a value δ is used as a static value for the amount of local reports to be processed. In future work we aim to implement an adaptive threshold for this value based on the network size.

Algorithm 2 Series Weighting Algorithm

```
Require:
    $R_{Global}$ the global report containing a set of local reports $R_{Local}$ belonging to $g \in G$.
    $\alpha \in [0, 1]$ the minimum threshold required for a classification of malicious.
Define:
    $\beta \leftarrow null$ the last processed decision.
    $\gamma \leftarrow 1$ a counter ranging from 1 to 5.
    $T_{malicious} \leftarrow 0$ a tally of malicious. votes
    $T_{innocuous} \leftarrow 0$ a tally of innocuous votes.

for $\langle eid, ts, \{g, (f, v), p\}\rangle \in R_{Global}$ do      ▷ Iterate and tally local decisions
    if $p \geq \alpha$ then
        $T_{malicious} \leftarrow \gamma$
        if $\beta = decision : malicious$ then
            if $\gamma < 5$ then
                $\gamma \leftarrow (\gamma + 1)$      ▷ Increase series weighting bonus when additional votes are cast
        $\beta \leftarrow decision : malicious$      ▷ Store the last counted decision in $\beta$
    else
        $\gamma \leftarrow 1$      ▷ Reset the counter when the series is broken
        $T_{innocuous} \leftarrow (T_{innocuous} + \gamma)$
        $\beta \leftarrow decision : innocuous$
if $T_{malicious} \geq T_{innocuous}$ then      ▷ Return global decision
    return $\langle eid, decision : malicious\rangle$
else
    return $\langle eid, decision : innocuous\rangle$
```

inefficient. Alternatively, if the decision to end the search for information is made too early, the classification will be made on an unrepresentative subset of the available information leading to inaccurate results. The domains model is used to allow agents to find the most favourable point to end the search for evidence by taking into consideration the plausibility of the data already collected. By considering the origin of evidence in relation to the domain graph, agents can decide whether a branch of the network has been sufficiently explored or whether further evidence collection is required.

Algorithm 3 Series Weighting with Cut-off Algorithm

```
Require:
    $R_{Global}$ the global report containing a set of local reports $R_{Local}$.
    $\alpha \in [0,1]$ the minimum threshold required for a classification of malicious.
    $\delta \in [0,1]$ the amount of local reports that will be processed.
Define:
    $\beta \leftarrow null$ the last processed decision.
    $\gamma \leftarrow 1$ a counter ranging from 1 to 5.
    $T_{malicious} \leftarrow 0$ a tally of malicious votes.
    $T_{innocuous} \leftarrow 0$ a tally of innocuous votes.

while $increment(R_{Global}) \leq (\delta * |R_{Global}|)$ do    ▷ Iterate over the global reports
    for $\langle eid, ts, \{g, (f, v), p\}\rangle \in R_{Global}$ do    ▷ Iterate and tally local decisions
        if $p \geq \alpha$ then
            $T_{malicious} \leftarrow \gamma$
            if $\beta = decision : malicious$ then
                if $\gamma < 5$ then
                    $\gamma \leftarrow (\gamma + 1)$    ▷ Increase series weighting bonus when additional votes are cast
            $\beta \leftarrow decision : malicious$
        else
            $\gamma \leftarrow 1$    ▷ Reset the counter when the series is broken
            $T_{innocuous} \leftarrow (T_{innocuous} + \gamma)$
            $\beta \leftarrow decision : innocuous$
if $T_{malicious} \geq T_{innocuous}$ then    ▷ Return global decision
    return $\langle eid, decision : malicious\rangle$
else
    return $\langle eid, decision : innocuous\rangle$
```

Series Weighting with Self-Selected Groups (Refer to Algorithm 4). To increase the adaptability of the system, agent preference is introduced in the form of self-selected groups to allow agents to autonomously measure the effectiveness of cooperating agents to prioritise their participation in future extended data collection tasks. Following the collection and analysis of some information, the agent must decide the general direction of the extended data collection task by choosing which agent can participate next. Over time, agents will find groups of high-performance agents that it prefers to work with, defining the self-selected group. This postpones the invocation of poorly performing agents effectively preventing their participation in the data collection task thus improving the overall performance. Agent performance measures the individual agent's accuracy compared to the groups. The corrective measures proposed in these algorithms attempt to improve the decision accuracy by minimising the effect poorly performing agents have on the overall classification by forcing poorly performing agents to participate later in the extended data collection task and thus increasing the chance that they will be cut-off and not be given a chance to participate (refer to Table 3 for detection rate improvements). Self-selected groups require inter-agent communication to inform participating agents of the event classification following the final global decision. Information about how other agents voted is also compared and the performance measure for each is locally updated for use in future events. Furthermore, this system makes the agents adaptable to changing network circumstances. If a particular technology becomes unavailable, the agent will quickly be removed from the self-selected group of preferred agents until the technology is restored and the agent can once again contribute to the extended data collection task.

6. Simulator

In this section, a discussion about the implementation of the DMASS simulator, including how domains are simulated, as well as an explanation of the underlying variables is provided. The simulator's configurable parameters fall into three categories (refer to Table 2): (1) those that control the agents (Gv), for example, decision accuracy and choice of voting algorithm; (2) those that control the environment (Ev), for example, the ratio of security events that are malicious and the size of the network; and (3) those that control the attack (Av), for example, the size of the affected region and detectability of the attack. We briefly describe the main variables in these three categories below.

Several variables are introduced to control aspects of the simulated network. The number of domains ($Ev_{amount} \in \mathbb{N}$) is set according to the size of the simulated network. Smaller networks with few services are simulated as having a limited number of domains, while larger expanded networks are simulated with more. Services with more complex operations require more agents to perform data collection. Agent membership to the individual domains is controlled by the *Domain Size Variability* variable ($Ev_{size} \in [0, 1]$). If set to zero, agents are distributed uniformly between the domains. If increased, domains will be created with a varied number of agents, with agents distributed at random if set to 1. The *Domain Association Factor* ($Ev_{association} \in [0, 1]$) variable controls the logical connections between neighbouring domains. If set to zero, all domains will be disjoint. If increased, links between domains, in the form of shared conditions and effects, are made which represents similar technologies that are closely related (refer to Figure 7). Over time these variables may

Algorithm 4 Self-Selected Groups Algorithm

Require:

R_{Global} the global report containing a set of local reports R_{Local}.
$R_{Decision}$ the final event classification from R_{Global}.

Define:

$g_{req} \subseteq G$ a subset of agents whose conditions c are satisfied and request the global report R_{Global}
g_{log} is a set formed by pairs (g, g_{score}), where $g \in G$ is an agent that previously has participated in an extended data collection task and $g_{score} \in \mathbb{N}$ is a tally of correct number of decisions.

```
procedure AGENT SELECTION($g_{req}, glog$)   ▷ Find the highest performing agent from $g_{req}$ using the previous performances of agents in $g_{log}$
    for $g \in g_{req}$ do
        return Highest($(g, g_{score}) \in g_{req}$)   ▷ Return highest performing agent listed in both $g_{req}$ and $g_{log}$ using the $g_{score}$.

procedure LOG UPDATE   ▷ Update the log with details of previous extended data collection tasks after each extended data collection task
    for $\langle eid, ts, \{g, (f, v), p\}\rangle \in R_{Global}$ do
        if $\exists g_{score} : (g, g_{score}) \in g_{log}$ then
            if $p == R_{Decision}$ then   ▷ Adjust the tally of correct decisions agent has made compared to the final group decision.
                $g_{score} \leftarrow (g_{score} + 1)$
            else
                $g_{score} \leftarrow (g_{score} - 1)$
```

Table 2
Agent and environment variables.

Variable	**Value**
Runs	100
Iterations	1000
Repetitions[1]	11
No. Agents	[30,100]
Preferred Agent Threshold[1]	[0,1]
No. conditions	1: 80%, 2: 15%, 3: 5%
Analysis (p)[2]	[0,1]
No. Domains (Ev_{amount})	[5,25]
False Alarm Rate	[0,1]
Attack Penetration ($Av_{penetration}$)	[0,1]
Attack Detectability ($Av_{detectability}$)	[0,1]
Domain Size Variability (Ev_{size})	[0,1]
Domain Association Factor ($Ev_{association}$)	[0,1]

[1] The preferred agent threshold is increased by 0.1 for each repetition. The repetition is cycled for the specified amount every iteration.
[2] Refer to Definition 4.

change as the network evolves with new devices added and others removed. However, for the purpose of the simulations, the network remains constant once initialised, we believe this is reflective of the typical local network that may experience infrequent changes over time but not necessarily during the operation of a particular extended data collection task.

Attacks are also simulated using the concept of domains. The variable *Attack Penetration* ($Av_{penetration} \in [0, 1]$) controls how far the attack will spread through the simulated network. Attacks primarily spread through connected nodes (defined by the Domain Association Factor). *Attack Detectability* ($Av_{detectability} \in [0, 1]$) controls the stealthiness of the attack. A low value would simulate an APT that is harder to detect. These variables are used to model the broad nature of cyber security attacks, for example, a Distributed Denial of Service (DDoS) attack would rank low on the attack penetration but high on attack detectability.

Table 2 lists the agent and environment variables used during the performance tests. Many of the variables including the false alarm rate, domain size and spread of the attack were randomised to verify the system under a wide variety of network conditions. The experimental setup to obtain these results modelled the network environment as closely as possible. Agent detection performance (Analysis) was controlled using a random distribution rather than a normal distribution to reflect the diversity of detection technologies that may perform differently depending on the accuracy of the individual sensor.

Table 3
Results and comparison with the system baseline using Detection Rate (DR) and False Alarm Rate (FAR) for the agents local analysis.

Evaluation	DR	FAR
Highest Votes (System baseline)	0.595	0.102
Series Weighting Basic	0.727	0.225
Series Weighting with Preferred Agents	0.792	0.231
Series Weighting with Cut-off and Preferred Agents	0.804	0.228

7. Evaluation

To evaluate the algorithms, 100,000 simulations consisting of 1000 *runs* with 100 security events per run are performed. For each run, a new domain network (See Figure 7) is generated, and 100 simulated security events consisting of both attacks and false alarms are initiated to assess the agent's performance.

Table 3 shows a comparison of the detection rate (DR) and false alarm rate (FAR) of the four algorithms during the individual evidence evaluation stage with a 20% DR improvement made over baseline. During the evidence evaluation stage, the agents individually collect and analyse evidence and decide whether it indicates normal or malicious activity. These individual analyses are then combined using one of the voting algorithms to classify the security event as a whole, as such the overall performance of the classifier must be judged based on both the individual performance (Table 3) and the final classification made by the group (Figure 9).

$$DR = \frac{TP}{TP + FN}$$

$$FAR = \frac{FP}{FP + TN}$$

This DR improvement can be attributed to the corrective measures introduced by the proposed algorithms that avoid the inclusion of poorly performing agents as well as the increased intelligence used to analyse and weigh the reliability of decisions. Furthermore, this improvement is made without introducing any extra detection mechanisms but instead is made by intelligently considering the plausibility of the information gathered through the concept of domains and preventing unreliable agents from damaging the integrity of the event classification. As a result of the algorithms, the FAR is also increased, however, we consider this to be by an acceptable amount given the current industry standards Alsubhi et al. (2011) and improvement made to the DR, furthermore this result is later improved during the application of the voting algorithm to decide the final classification. In many business environments, the detection rate is given priority over the false alarm rate to protect networked assets at the cost of possible availability disruption. Significant efficiency improvements were also made with a reduction in the number of local decisions needed to analyse an event. Figure 8 shows the total number of local decisions reduced by over 50% while maintaining the same ratio of correct global decisions. The result shows that less processing was needed to come to the same conclusion about the security event. This performance improvement is significant as it shows the agent's ability to select the most relevant agents needed for analysing the security event while avoiding the use of irrelevant agents outside of the attacked domain. This result shows the opposite of the brute-force approach often adopted by IDSs and instead avoids irrelevant computation during the detection process. Figure 9 shows a comparison of the algorithms in both low and high false alarm environments with the improved algorithms performing better in environments with a high false alarm rate. This improvement from the system baseline is again attributed to the agent's ability to identify the poorly performing agents using the domains model. Whereas the system baseline performance is directly linked to the agent's ability to analyse an individual piece of information, high false alarm environments provide the improved algorithms with an increased opportunity to identify the poorly performing agents and optimise the extended data collection task around them.

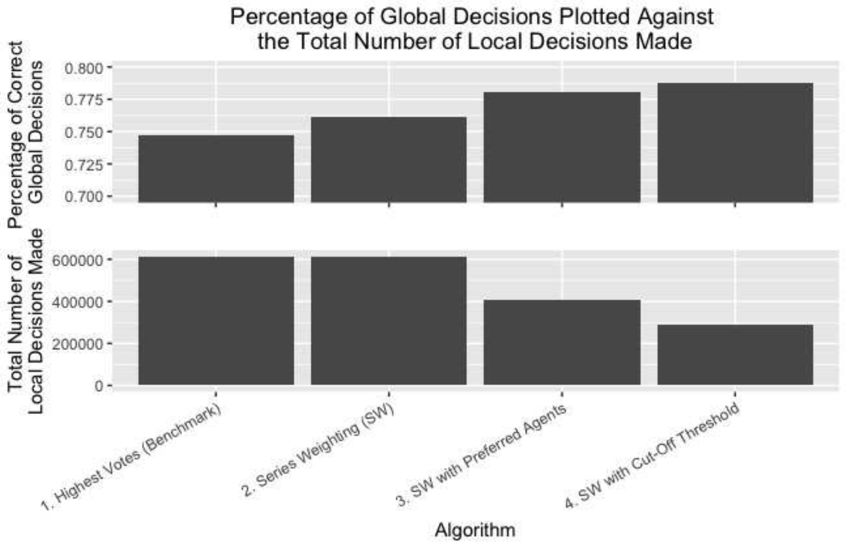

Figure 8
The total number of local decisions (i.e., cost associated actions) using each of the four algorithms described in Table 3 plotted against the percentage of correct global decisions.

This system is entirely decentralised with each agent maintaining a local copy of how well an agent performed in the past. Different types of attacks elicit data collection tasks that explore different parts of the network, and since each agent maintains a local database of preferred agents, they are sensitive to the directionality of various attacks. In addition to the preferred agent, any agents that continually perform poorly will be avoided during the data collection

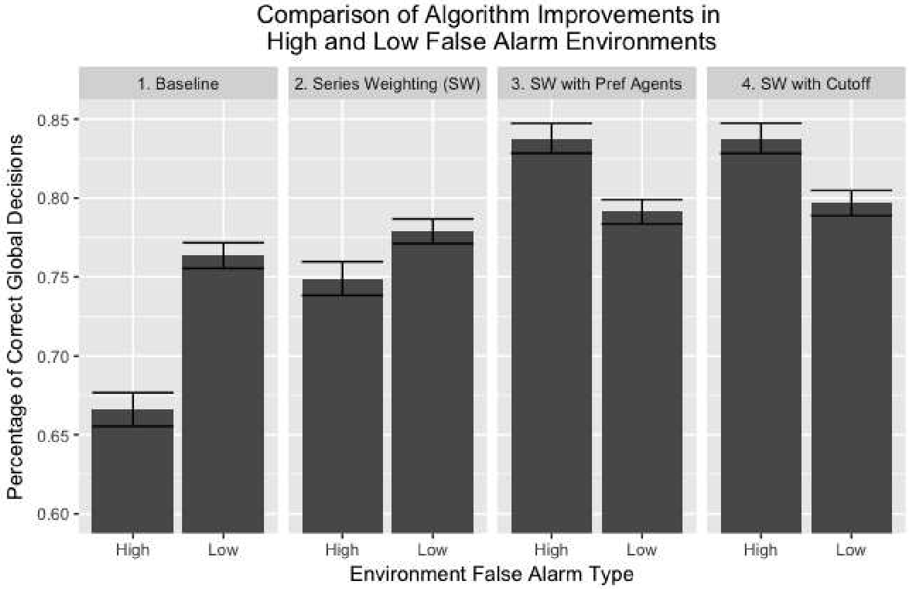

Figure 9
A comparison of the 4 algorithms featured in Table 3 in both a low and high false alarm environment (95% confidence intervals).

process. This is more desirable than centrally managing the reputation of agents which does not reflect the performance diversity under different situations, but instead, assigns generic labels which may be unrepresentative. Whereas an agent may perform badly within one domain, it may perform well in another. Under the current system, this is recognised and selected for during the preferred agent process. Coupled with the cut-off system, this has the effect of invoking the most reliable agents earlier on in the collection process and not providing the poorly performing agents with an opportunity to participate.

Figures 10 and 11 show the change in amount of correct local decisions (Total LD correct) and correct global decisions (Total GD correct) when changing the simulator parameters for the number of agents and attack detectability. The number of agents has the effect where fewer agents (in the 15-100 range) causes the system as a whole to perform poorly due to ineffective corrective measures that would otherwise penalise poorly performing agents. The performance of the agent preference corrective strategy, which penalises poorly performing agents in future extended data collection tasks, is limited by the lack of agent choice in the 15-100 range resulting in poorly performing agents used out of necessity to continue the analysis (where no preferable agents are available, less preferred agents are chosen to fulfil the extended data collection task). Where more choice (in terms of number of agents) is available, the more reliable and highly performing agents are selected for participation. The performance gain of adding

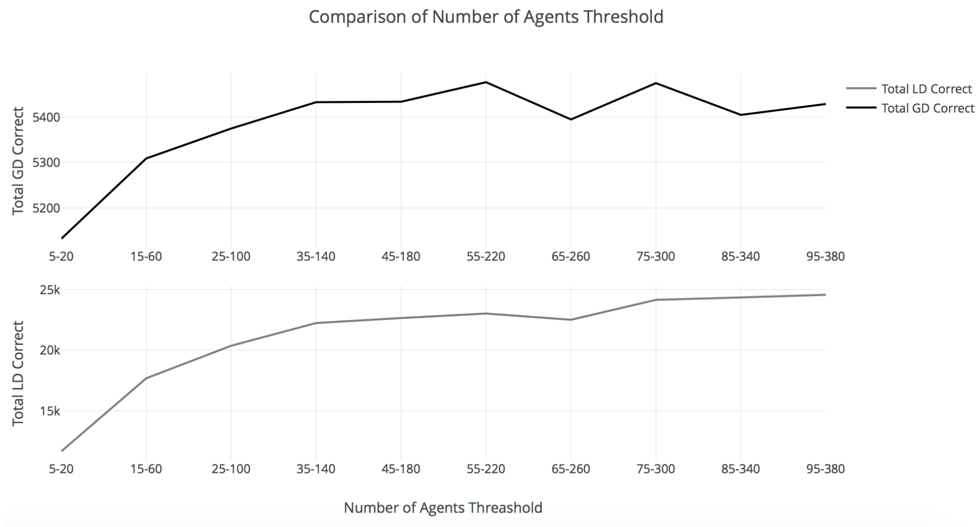

Figure 10
Comparison of Number of Agents (No. Agents) Threshold.

agents to the network stabilises after a point owing to the cut-off algorithm that prioritises the high performing agents and ends the search for evidence before every agent has had an opportunity to participate. While there is no clear disadvantage to adding more agents to the network, the amount of messages sent between agents for participation increases with each additional agent. The attack detectability measure which represents the stealthiness of the attack has the predictable effect of reducing the amount of correct global and local decisions when it is harder to detect, i.e., when this value is low, and the attacker is harder to detect, agents are more likely to analyse the event incorrectly. The corrective measures of penalising poorly performing agents go some way to improving the agent performance, however, stealthy attacks are still a challenge to the agent-based system as well as to the IDS.

7.1. Snort IDS Evaluation

To further show the need for the proposed system, an evaluation using the Snort IDS Roesch & Green (2016) was performed on the UNB ISCX Intrusion Detection Evaluation Dataset Shiravi et al. (2012) to show increased efficiency in detecting malicious traffic. The dataset contains a series of labelled network flows from within a local network. The dataset was first analysed with Snort IDS's signatures and then processed by the agents to find the remaining connection flows that were not detected. Figure 12 shows an example of the network graph built from the UNB ISCX dataset where nodes are IP addressable hosts and edges represent network connections made between them. The graph is used to model connections between nodes so that agents may follow the spread

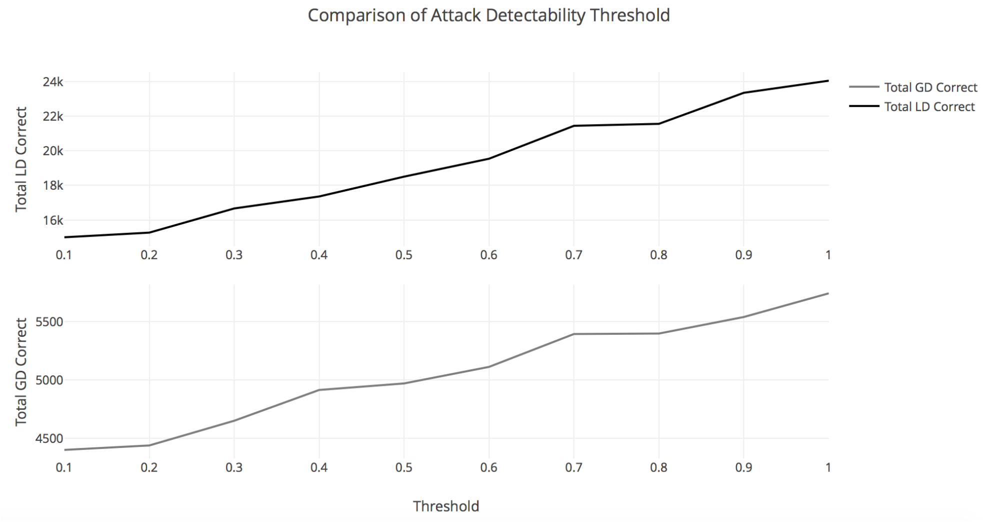

Figure 11
Comparison of Attack Detectability ($Av_{detectability}$ threshold).

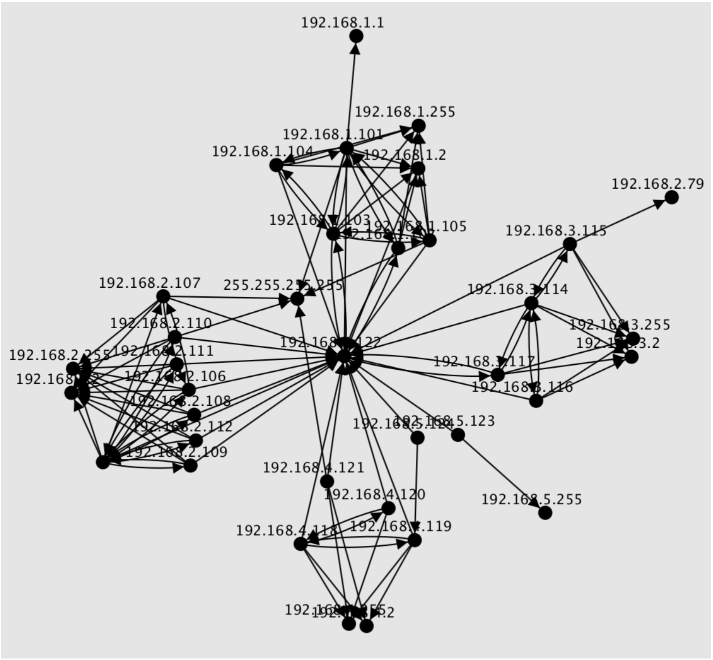

Figure 12
An example of the USB ISCX dataset network.

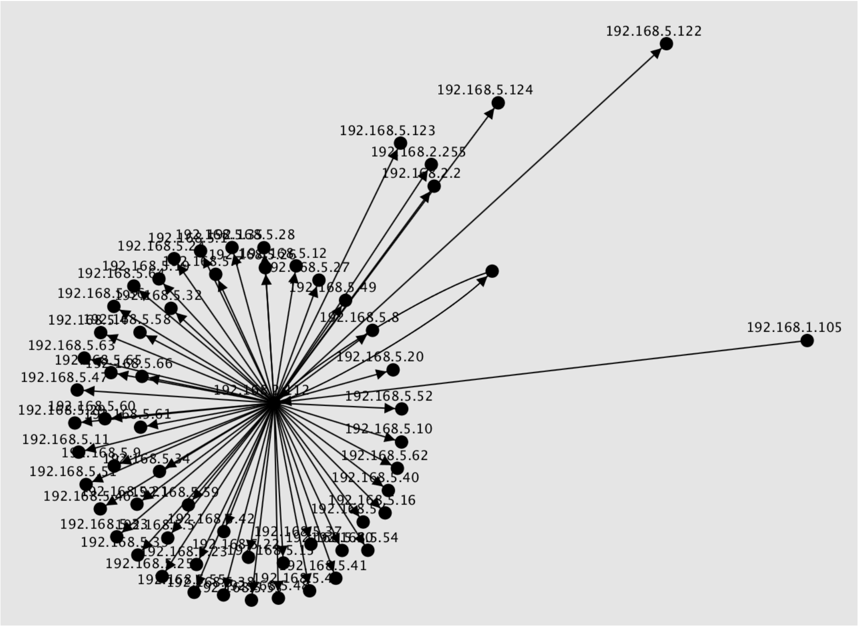

Figure 13
An example of probe activity not detected by snort IDS.

of connection through the domains to search for undetected nodes.

Using the Snort IDS analysis as a starting point, the agents search subsections of the network to discover additional edges that are malicious but were not detected by the IDS. Snort takes on the role as the data gathering/analysis module by using its signatures to classify activity as malicious or innocuous. Typically an IDS has many signatures that can be enabled or disabled depending on the threat model with fewer signatures being used to improve the FAR and more signatures used to increase the DR but at the cost of increasing the FAR.

The dataset model contains a total of 236 edges with 148 malicious. Snort IDS detected a total of 25 edges. The majority of the edges not detected by Snort were attempts to scan the network where the attacker scanned an IP address not in use. Typically, this behaviour is considered non-malicious by most IDS's as its cause often occurs normally within local networks as a result of typical probing and non-malicious networking problems. Figure 13 shows the connection model for probe activity with many connections made to IP addressable nodes that do not exist on the network and so get no response.

The following agent based search of the data is measured by the detection rate increase as well as the computational cost of performing additional search. The detection rate is measured by the number of additional edges investigated that were malicious while the computational cost is measured by the total number of edges investigated. While the detection accuracy is dependent on the quality of the signatures a search based on the series weighting algorithm (see Algorithm 2) can be used to search sub-domains of the graph network based on the results of Snort's initial analysis of the data. By prioritising search in the areas that Snort previously identified as containing malicious traffic, the search can more efficiently find the remaining malicious edges without having to search unaffected areas of the network. While Snort parsed all 236 edges to find 25 malicious events, the following search of the data parsed only 64 edges to find an additional 9 edges that were undetected by Snort. The agent search was both more efficient than Snort (based on the amount of searched nodes that contained malicious traces) and detected additional instances of malicious activity that it could not. The benefit of our proposed system is that it can use the concept of domains to find undetected locations of compromise. Following the agent's identification of an undetected location, the area can be analysed with additional IDS signatures under increased suspicious while avoiding unnecessary analysis of unaffected areas.

7.2. System Scalability

During an IDS's operation it will process all available data and make comparisons against known signatures of known malicious activity. This action provides a $\mathcal{O}(n)$ processing time where n is the number of signatures for be processed. This is functionally similar to the system benchmark shown in Figure 8 that has a large number of local decisions must be processed to analyse the global decision. Our proposed algorithms improve on this by requiring less local decisions (i.e., pieces of information) to be analysed for a similar overall result. In particular, algorithm 4 titled Series Weighting with Cut-Off Threshold reduces the amount of processing required by over 50% which significantly improves the scalability of the system as less work must be undertaken to achieve the same result.

Furthermore, the communication module to allow agents to exchange information and communicate is kept as lightweight as possible by separating the system functions. The function to find agents whose conditions are satisfied by the current information is a small message with a minimal network footprint, while the larger global report is only sent specifically to the chosen agent and not broadcast to the network. The algorithm results in Figure 8 that show the amount of local decisions required is substantially reduced is in proportion with the amount of network messages that are sent as every unique local decision requires the global report to be sent.

7.3. Existing Approaches

Given that the proposed system uses existing detection technologies within each agent to process the collected information, average agent performance modelled the current state of the art for IDSs (detection rate of 70-100% depending on the type of attack). Agents were tested under a number of simulated environment types (controlled by the Ev variables) to make the results as generalisable as possible. A wide range of attacks were performed to simulate the diversity of footprints that may be left behind by the attacker (controlled by the Av variables). Using this model, improvements were made to the performance of traditional technologies through the inclusion of agent-based mechanisms, in particular, the proposed algorithms are shown to be effective in high false alarm environments where traditional technologies relying on only detection signatures without corrective measures perform poorly. With the control variables (Ev and Av) randomly initialised between simulated runs, simulations were performed under a diverse range of conditions representative of modern networks. From the results in Table 3, we show that an improvement to the detection rate of up to 20% can be made over traditional approaches by introducing corrective measures to detect poorly performing agents in high false alarm situations. We show that traditional systems that attempt to match as many signatures as possible results in worse performance than the extended data collection approach because irrelevant classifications can dilute the final event analysis.

Throughout this paper, we study intrusion detection using a graph-theoretic approach due to the lack of context that can be extracted from the current generation of datasets. The proposed model requires knowledge about technology domains to evaluate series of decisions, however, such information is rarely included in current signature-based systems. Currently, when network datasets are created, contextual information about the network and endpoint service are typically not collected increasing the difficulty of automatically defining conditions and effects. The authors agree with Grobler et al. (2010) that live forensic investigations are hampered by a lack of automatic forensic standards and procedures and hope the proposed system will go some way to providing an effective agent-based architecture for performing decentralised forensic investigations.

7.4. Model Vulnerabilities

Our model provides a robust solution to encompassing many detection technologies for use by an automated agent-based forensic framework. However, the use of agent-based technologies brings with it a set of considerations highlighting the limitations of the model. Agents rely on the transfer of information to share local views of the network to make the final global decision, as a result, under conditions of prolonged network congestion or purposeful denial of service, the extended data collection task could be disrupted. While the availability of agents is an important consideration, forensic investigations can occur after-the-fact, as such the process may continue once normal network conditions are restored.

In MAS, the communications overhead is often computationally expensive due to the decentralised nature of the system. Figure 8 shows the trade-off between the global decisions and the number of local decisions (which represents the number of communications made). While the figure shows a reduction in the overall number of communications between the four algorithms, this is still exponentially higher than a central approach that does not require any communication between its components. The trade-off is shown in Figure 9 where the baseline approach performs poorly in high false alarm environments because of the vast amount of endpoints to scan and an inability to distinguish between malicious and innocuous. However, the Series Weighting with Cut-off algorithm performs much better because it uses an agent performance measure to penalise poorly performing agents. Overall, while the communications overhead creates an increased burden on the network, the detection accuracy is increased.

8. Conclusion & Future Work

Performing automatic security and forensics within networked environments is a challenging problem due to the variety of data and unpredictable events. The Decentralised Multi-Agent Security System presented in this paper performs automatic data collection and analysis by using forensic-inspired processes to search for information useful during the analysis of a security event. The modelling concept of domains was introduced to capture both the underlying structure of the network security environment as well as expected attack patterns by allowing agents to weigh the importance of information based on the location and likelihood that the data is, in fact, a true positive. Several agent-based algorithms were developed to improve agent performance during the evaluation process by devaluing the importance of poorly performing agents and actively seeking the participation of high performing agents. By using the proposed algorithms, the efficiency of the system was increased by over 50% promoting shorter and more targeted automatic forensic investigations, and the accuracy of the system was improved by 20%. Future work will focus on the development of anomaly-based detection algorithms that make use of the proposed framework to consider attacks from both the local and the global views available to agents. We aim to bring the concepts of situational information and the plausibility of evidence discussed throughout this paper to improve anomaly detection through resulting targeted investigations to find supporting evidence.

References

Alsubhi, K., Bouabdallah, N., & Boutaba, R. (2011). Performance analysis in Intrusion Detection and Prevention Systems. *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, (pp. 369–376). doi:10.1109/INM.2011.5990713.

Baig, Z. a. (2012). Multi-agent systems for protecting critical infrastructures: A survey. *Journal of Network and Computer Applications*, *35*, 1151–1161. URL: `http://dx.doi.org/10.1016/j.jnca.2012.01.006`. doi:10.1016/j.jnca.2012.01.006.

Carvalho, L. F., Barbon, S., Mendes, L. D. S., & Proen??a, M. L. (2016). Unsupervised learning clustering and self-organized agents applied to help network management. *Expert Systems with Applications*, *54*, 29–47. doi:10.1016/j.eswa.2016.01.032.

Chen, P., Desmet, L., & Huygens, C. (2014). A study on advanced persistent threats. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (pp. 63–72). volume 8735 LNCS. doi:10.1007/978-3-662-44885-4-5.

Clint, M. R., Reith, M., Carr, C., & Gunsch, G. (2002). An Examination of Digital Forensic Models. *International Journal of Digital Evidence*, *1*, 1–12. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.9683. doi:10.1109/SADFE.2009.8.

da Gama Batista, J., Bouchaud, J. P., & Challet, D. (2015). Sudden trust collapse in networked societies. *European Physical Journal B*, *88*, 1–11. doi:10.1140/epjb/e2015-50645-1. arXiv:1409.8321.

DeLellis, P., DiMeglio, A., Garofalo, F., LoIudice, F., Franco, G., & Francesco, L. I. (2017). The evolving cobweb of relations among partially rational investors. *PLoS ONE*, *12*, 1--21. doi:10.1371/journal.pone.0171891.

Durkota, K., & Kiekintveld, C. (2015). Optimal Network Security Hardening Using Attack Graph Games. In *IJCAI International Joint Conference on Artificial Intelligence*.

Fernandez, E., Pelaez, J., & Larrondo-petrie, M. (2007). Attack Patterns : a New Forensic and Design Tool. In *IFIP International Conference on Digital Forensics* (pp. 345--357). volume 242.

Galtsev, A., & Sukhov, A. (2011). Network attack detection at flow level. *Smart Spaces and Next Generation Wired/ ...*, *6869 LNCS*, 326--334. URL: http://arxiv.org/abs/1104.1010 http://link.springer.com/chapter/10.1007/978-3-642-22875-9_30. arXiv:1104.1010.

Grobler, C. P., Louwrens, C. P., & Von Solms, S. H. (2010). A multi-component view of digital forensics. *ARES 2010 - 5th International Conference on Availability, Reliability, and Security*, (pp. 647--652). doi:10.1109/ARES.2010.61.

Haack, J. N., Fink, G. a., Maiden, W. M., McKinnon, a. D., Templeton, S. J., & Fulp, E. W. (2010). Ant-based cyber security. *Proceedings - 2011 8th International Conference on Information Technology: New Generations, ITNG 2011*, (pp. 918--926). doi:10.1109/ITNG.2011.159.

Herzog, P. (2010). *OSSTMM 3.0 - The Open Source Security Testing Methodology Manual*. Technical Report 3.0 ISECOM.

Jahanbin, A., Ghafarian, A., Seno, S. A. H., & Nikookar, S. (2013). A Computer Forensics Approach Based on

Autonomous Intelligent Multi-Agent System. *International Journal of Database Theory and Application*, *6*, 1--12. URL: http://www.sersc.org/journals/IJDTA/vol6_no5/1.pdf. doi:10.14257/ijdta.2013.6.5.01.

Kendrick, P., Hussain, A., & Natalia, C. (2016). Multi-Agent Systems for Dynamic Forensic Investigation. In *International Conference on Intelligent Computation (ICIC)* (pp. 796----807). Lanzhou: Springer.

Kikuchi, H., Kobori, T., & Terada, M. (2009). Orthogonal expansion of port-scanning packets. In *NBiS 2009 - 12th International Conference on Network-Based Information Systems* (pp. 321--326). Ieee. doi:10.1109/NBiS.2009.82.

de Lellis, P., Di Meglio, A., & Lo Iudice, F. (2017). Overconfident agents and evolving financial networks. *Nonlinear Dynamics*, (pp. 1--8). doi:10.1007/s11071-017-3780-y.

Liao, H.-J., Richard Lin, C.-H., Lin, Y.-C., & Tung, K.-Y. (2012). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, *36*, 16--24. URL: http://dx.doi.org/10.1016/j.jnca.2012.09.004. doi:10.1016/j.jnca.2012.09.004.

Mahoney, M., & Chan, P. (2001). PHAD: Packet header anomaly detection for identifying hostile network traffic. *Florida Institute of Technology technical report CS-2001-04*, (pp. 1--17). doi:citeulike-article-id:9927948.

Mees, W. (2012). Multi-agent anomaly-based APT detection. *Proceedings of the Information Systems Technology Panel Symposium (IST-111/RSY-026)*, (pp. 1--10).

Mukherjee, B., Heberlein, L. T., & Levitt, K. N. (1994). Network intrusion detection. *IEEE Network*, *8*, 26--41. doi:10.1109/65.283931.

Obes, J. L., Sarraute, C., & Richarte, G. (2010). Attack Planning in the Real World. In *AAAI Conference on Artificial Intelligence* (pp. 10--17). Atlanta. arXiv:1306.4044v1.

Roesch, M., & Green, C. (2016). *Snort User Manual*. Technical Report Sourcefire Inc. URL: https://www.snort.org/documents/snort-users-manual.

Rudrapal, D., Das, S., Debbarma, N., & Debbarma, S. (2013). Internal attacker detection by analyzing user keystroke credential. *Lecture Notes on Software Engineering*, *1*, 49. doi:10.7763/LNSE.2013.V1.11.

Shakarian, P., Simari, G. I., Moores, G., & Parsons, S. (2015). Cyber Attribution : An Argumentation-Based Approach. *Cyber Warefare, Springer International Publishing*, (pp. 151--171).

Shanmugasundaram, K., Memon, N., Savant, A., & Bronnimann, H. (2003). ForNet : A Distributed Forensics Network. *Computer Network Security. Springer*, (pp. 1--16).

Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers and Security*, *31*, 357--374. URL: http://dx.doi.org/10.1016/j.cose.2011.12.012. doi:10.1016/j.cose.2011.12.012.

Smith, G. E., Watson, K. J., Baker, W. H., & Pokorski II, J. a. (2007). A critical balance: collaboration and security in the IT-enabled supply chain. *International Journal of Production Research*, *45*, 2595--2613. doi:10.1080/00207540601020544.

Tobergte, D. R., & Curtis, S. (2013). Modeling Multistep Cyber Attacks for Scenario Recognition. *Journal of Chemical Information and Modeling*, *53*, 1689--1699. doi:10.1017/CBO9781107415324.004. arXiv:arXiv:1011.1669v3.

Tsai, C. F., Hsu, Y. F., Lin, C. Y., & Lin, W. Y. (2009). Intrusion detection by machine learning: A review. *Expert Systems with Applications*, *36*, 11994--12000. URL: http://dx.doi.org/10.1016/j.eswa.2009.05.029. doi:10.1016/j.eswa.2009.05.029.

Verwoerd, T., & Hunt, R. (2002). Intrusion detection techniques and approaches. *Computer Communications*, *25*, 1356--1365. doi:10.1016/S0140-3664(02)00037-3.

Zolfpour-Arokhlo, M., Selamat, A., & Hashim, S. Z. M. (2013). Route planning model of multi-Agent system for a supply chain management. *Expert Systems with Applications*, *40*, 1505--1518. URL: http://dx.doi.org/10.1016/j.eswa.2012.08.040. doi:10.1016/j.eswa.2012.08.040.

Zuech, R., Khoshgoftaar, T. M., & Wald, R. (2015). Intrusion detection and Big Heterogeneous Data: a Survey. *Journal of Big Data*, *2*, 1--41. URL: http://www.journalofbigdata.com/content/2/1/3. doi:10.1186/s40537-015-0013-4.