A Multi-Agent Approach to Advanced Persistent Threat Detection in Networked Systems

Phillip Kendrick BSc (Hons)

A thesis submitted in partial fulfilment of the requirements of

Liverpool John Moores University for the degree of Doctor of Philosophy

September 2018

Abstract

Advanced cyber threats that are well planned, funded and stealthy are an increasing issue facing secure networked systems. As our reliance on protected networked systems continues to grow, the motivation for developing new malicious techniques that cannot be easily detected by traditional signature-based systems, and that make use of previously unseen zero-day vulnerabilities, continues to grow. Lack of adaptivity, extended data-collection and generalised algorithms to detect stealthy attacks is contributing to the insecurity of modern networked systems. To protect these networks, new approaches that can monitor and respond to indicators of compromise in a reflective way that considers all of the available evidence rather than individual points of data is required.

This thesis presents a novel approach to intrusion detection and specifically focuses on detecting advanced persistent threats which are characteristically stealthy and evasive attacks. This approach offers a multi-agent model for automatically collecting, analysing and classifying data in a distributed way that considers the context in which the data was found. Using a context-based classification that considers the likelihood of a data-point being a false alarm or legitimate is used to decrease the prevalence of erroneous classifications and regulate continuation of the data collection process. Using this architecture, a detection rate increase of up to 20% is achieved in false alarm environments and an efficiency increase of up to 50% made over traditional monolithic intrusion detection systems. Additionally, the shortcomings of algorithms to detect stealthy attacks are addressed by providing a generalised anomaly detection algorithm for detecting the initial traces of an attack and deploying the proposed multi-agent model to investigate the attack further. The generalised algorithms can detect a wide variety of network-based attacks at an average detection rate of 85% providing an accurate and scalable way to detect the initial traces of compromise.

The main novelty of this thesis is providing systems for detecting attacks where the threat model is increasingly stealthy and assumed capable of bypassing traditional signature-based approaches. The multi-agent architecture is unique in its ability, and the generalised anomaly detection algorithm is novel in detecting a variety of different cyber attacks from the network-

flow layer. The evidence from this research suggests that context-based evidence gathering can provide a more efficient approach to analysing data and the generalised anomaly detection algorithm can be applied widely to detect attack indicators.

Acknowledgements

I would first like to express my gratitude to my advisers Dr. Natalia Criado, Prof. Abir Hussain and Dr. Martin Randles for their continuous support of my PhD study. In particular, I would like to thank them for their support, guidance and knowledge provided to me throughout the course of my research. I could not have imagined having a better advisory team. Special thanks go to the Department of Computer Science, Liverpool John Moores University for their funding of this project and the encouragement I received.

Finally, I must express my gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis.

Contents

Authored Papers

1. Kendrick, P., Criado, N., Hussain, A., and Randles, M. An Unsupervised Approach to Detecting Local Network Infiltrations. *IEEE Transactions on Sustainable Computing, Special Issue on Sustainable Cyber Forensics and Threat Intelligence* (2018), 1–12 (In Review)

2. Kendrick, P., Baker, T., Maamar, Z., Hussain, A., and Buyya, R. An Efficient Multi-Cloud Service Composition Via A Distributed Multiagent-based , Memory-driven Approach. *IEEE Transactions on Sustainable Computing* (2018) (In Review)

3. Kendrick, P., Criado, N., Hussain, A., and Randles, M. A self-organising multi-agent system for decentralised forensic investigations. *Expert Systems with Applications 102* (2018), 12–26

4. Kendrick, P., Hussain, A., Criado, N., and Randles, M. Selecting Scalable Network Features for Infiltration Detection. In *2017 10th International Conference on Developments in eSystems Engineering (DeSE)* (2017), pp. 88–93

5. Kendrick, P., Hussain, A., Criado, N., and Randles, M. Multi-agent systems for scalable internet of things security. In *ICC '17 Proceedings of the Second International Conference on Internet of things and Cloud Computing* (2017)

6. Kendrick, P., Hussain, A., and Criado, N. Multi-Agent Systems for Dynamic Forensic Investigation. In *International Conference on Intelligent Computation (ICIC)* (Lanzhou, 2016), Springer, pp. 796—-807

Glossary

AAFID Autonomous Agents for Intrusion Detection 35, 50

ABS Agent-Based System 48, 65, 68, 89

ACCESS Agents Channeling (or Conveying) ContExt Sensitive Services 69

ACCS Australian Centre for Cyber Security 147

AIS Artificial Immune System 54, 55

AM Analytical Model 38

ANN Artificial Neural Network 51, 53, 55, 58

APT Advanced Persistent Threat 23, 31, 40, 41, 43–47, 52, 55, 83, 85, 102, 103, 111, 130, 150, 151

ART Adaptive Resonance Theory 17, 51, 52, 63

BDI Belief, Desire, Intention 65, 74

BFS Breadth-First Search 80, 82, 83

CIKC Cyber Intrusion Kill Chain 45

CnC Command and Control 44

CNN Convolutional Neural Network 50

DARPA Defense Advanced Research Projects Agency 60, 146, 147, 149

DDoS Distributed Denial of Service 103, 149

DEC-POMDPs Decentralised Partially Observable Markov Decision Processes 81, 82

DFS Depth-First Search 76, 80

DHCP Dynamic Host Control Protocol 37, 41

DMASS Decentralised Multi-Agent Security System 9, 17, 18, 85–88, 98, 100, 102, 109, 111, 129–132, 137, 141, 144, 153, 154, 158

DNS Domain Name System 38, 41, 42, 47, 56, 73, 76, 77, 87, 89

DoS Denial of Service 34, 53, 58, 147, 149

DR Detection Rate 137, 138, 142

EC Evolutionary Computation 52, 53

EM Environmental Model 38

FAR False Alarm Rate 137–139, 142

FFP Fast Forward Planning 82

FIRE Fuzzy Intrusion Recognition Engine 17, 49, 50

HTN Hierarchical Task Network 83

IDA* Iterative Deepening A* Search 75

IDS Intrusion Detection System 17, 32, 34, 35, 38–40, 42, 46, 47, 49, 51, 52, 54–59, 68, 86, 87, 97, 102, 112, 113, 119, 137, 139, 141–143, 148, 153, 158

IoT Internet of Things 39, 45, 58

IP Internet Protocol 47, 90, 97–99, 114, 116

L2L Local-to-Local 113, 146

LDA Latent Dirichlet Allocation 59

LSTM Long Short-Term Memory 51

MAS Multi-Agent System 35–39, 47–50, 65–70, 72–74, 77–79, 83, 85, 96, 111, 131, 137, 145, 158

MDP Markov Decision Process 81

ML Machine Learning 48, 153

MPCP Multi-Agent Plan Coordination Problem 80

NIST National Institute of Standards and Technology 23, 31

PCA Principal Component Analysis 51, 63

PDDL Planning Domain Definition Language 76

PLC Programmable Logic Controller 43, 44

R2L Remote-to-Local 50, 53, 57, 58, 149

RAT Remote Administrator Toolkit 41, 42

SHOP Simple Hierarchical Ordered Planner 83

SI Swarm Intelligence 56

SQL Structured Query Language 42

SVM Support Vector Machine 58

U2R User-to-Root 50, 53, 57, 58, 149

VPN Virtual Private Network 90, 91

XSS Cross Site Scripting 42–44

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Secure network systems are essential for the functioning of modern computing in all sectors of society. Encapsulated within the concept of security are the individual goals of providing (i) confidentiality, (ii) integrity, and (iii) availability which together keep data known only to the individuals that should have access to it, ensure the data remains unchanged and allow the data to be accessed when needed. NIST (National Institute of Standards and Technology) further expands this definition to include the process of protection, detection, identification, response and recovery. The adversaries of secure systems attempt to perform data breaches which are described in a study by the Ponemon Institute [132] as an event in which an individual's name and a medical record and/or a financial record or debit card is potentially put at risk - either in electronic or paper format. A record is described as information that identifies the natural person (individual) whose information has been lost or stolen in a data breach. One example is a retail company's database with an individual's name associated with credit card information and other personally identifiable information. Another is a health insurer's record of the policyholder with physician and payment information.

Cybercrime affects society as a whole in addition to threatening the individual's privacy or businesses intellectual property. Breaches in privacy may compromise a countries critical infrastructure (i.e., energy, telecommunications, financial and transport) and cause lasting effects.

1.1 The Cause of Insecure Systems

The cause of insecure systems is a combination of poor detection systems and increasingly advanced adversaries. The APT (Advanced Persistent Threat) is an example of an attack model becoming increasingly prevalent in recent years, characterised by their well funded and targeted operations that prioritise stealth and attack longevity over immediate reward. These advanced

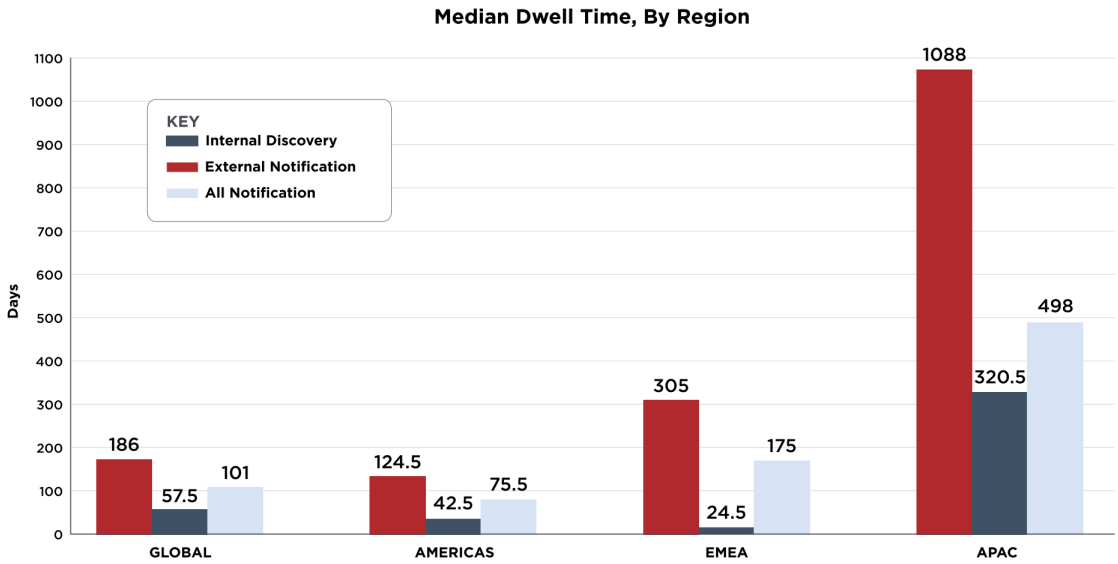

Figure 1.1
Median dwell time by region - Europe, the Middle East and Africa (EMEA), Asia-Pacific (APAC) [109]

attacks often include new and unseen approaches to evading current detection solutions making them increasingly difficult to monitor. In a 2018 study by Mandiant, the dwell times of attacks were surveyed to examine the extent of long-lasting cyber attacks (i.e., the number of days from the first evidence of compromise that an attacker is present on a victim network before detection). Figure 1.1 shows the global average to be 101 days while increasing to 498 days for the Asia-Pacific (APAC) region highlighting the issue that attackers are capable of spending prolonged periods of time within systems protected by the current generation of detection technologies.

Furthermore, the study found that the 44% of the attacks discovered were detected by an external source for the European and the Middle East and Africa (EMEA) regions (see Figure 1.2). This finding highlights the current state of cybersecurity with many intrusion detection and prevention systems failing to detect the attack at any point, and further highlights the need for an increasingly autonomous approach capable of investigating stealthy attacks that would otherwise not be detected by traditional technologies in a timely manner.

1.2 The Cost of Insecure Systems

The Ponemon Institute study collected global statistics on the average number of breached records by country (refer to Figure 1.3) as well as the average cost per capita compared to the four-year average (refer to Figure 1.4). The study concluded that the faster the data breach can be identified and contained, the lower the resulting cost. A separate study highlighted the

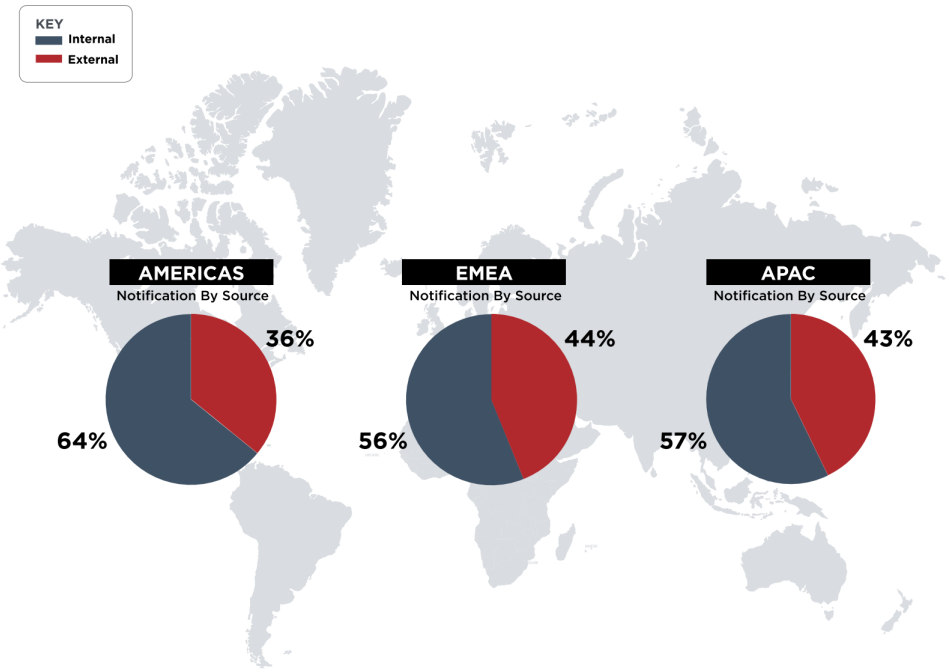

Figure 1.2
Organisation attack detection source [109]

need for improved detection within internal networks, citing that the median period between an attack happening and its detection was 205 days with only 31% of breaches detected internally by the target of the attack [59, 77]. The average size of the data breach concerning the number of breached records was 21,663 for the United Kingdom in 2017 and cost on average £74 ($123) per capita.

One possible approach to address the issues surrounding network security and real-time responses is to utilise the advances made within the multi-agent systems and machine learning communities. This thesis examines the use of agent-based systems for automated detection and response to network-based cyber attacks. This has been achieved by i) developing a multi-agent system named the Decentralised Multi-Agent Security System (DMASS) to encapsulate current detection mechanisms such as signature detection and targeted anomaly-based detection into a mobile and automated system to facilitate increased information traversal, and ii) the development of a Generalised Anomaly Detection Algorithm to be used in conjunction with the DMASS to detect the initial stages of an attack at an abstract layer and mobilise the DMASS to perform a targeted investigation of the potential security breach.

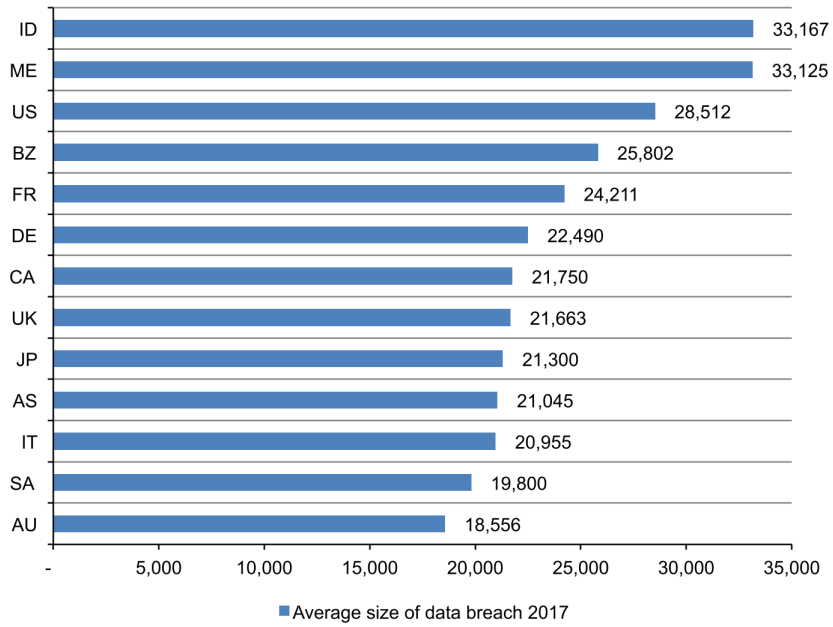

Figure 1.3
The average number of breached records by country or region [132].

1.3 Aims & Objectives

The aim of this thesis is to address the issue of advanced and stealthy threats facing modern networked systems. In particular, to increase the detection rate of a variety of attacks and where possible the efficiency. The issue of advanced and stealthy attackers has become more prominent in recent years due to advancements in detection technologies and the corresponding techniques developed to avoid detection. Below, several objectives are considered to address this aim:

Objective 1: Multi-Agent Systems. To investigate and demonstrate the use of Multi-Agent Systems to provide a more adaptive and real-time solution for network monitoring.

Objective 2: Datasets. To gain an in-depth understanding of network-layer security datasets. Any dataset used must be up to date, contain a variety of network attacks and provide a comprehensive overview of the network.

Objective 3: Anomaly Detection. To investigate the use of anomaly-based algorithms to guide the proposed multi-agent system during cyberattack investigations. While Multi-Agent Systems are useful in autonomously investigating events, a trigger is required to determine where and when the system should be deployed. To address this, generalised anomaly-based algorithms will be analysed as a possible solution.

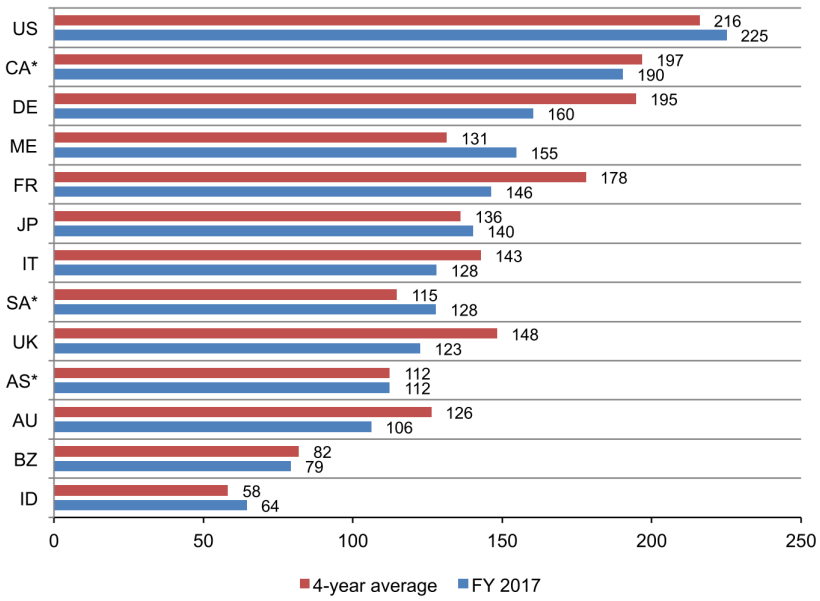

Figure 1.4
The 2017 per capita cost of data breach compared to the four-year average [132].

Objective 4: Overfitting. To investigate the use of machine learning within the network security environment. In particular, to investigate measures to avoid overfitting in this adversarial domain where attackers deliberately seek to deceive the network monitor and a lack of good datasets inhibits supervised training.

Objective 5: Network Forensics. To investigate the application of manually performed forensic processes for use within a Multi-Agent System. The forensic process of gathering information and tailoring the search response to consider evidence found is inherently efficient but not currently used in an automated way.

1.4 Novel Contributions

The research within this thesis provides a novel multi-agent approach to network evidence collection and analysis as well as algorithms to guide autonomous systems during the detection process.

This thesis makes the following novel contributions to the field of intrusion detection and multi-agent systems:

- An autonomous multi-agent architecture to collect and analyse information within the local network, featuring a system of conditions & effects which describes how information and technologies within a network relate to each other to provide a guiding mechanism

for evidence retrieval. No current adaptive architecture provides adequate information collection and analysis of networks or logically follows the evidence already collected to inform future search actions.

- A guidance system for the multi-agent architecture using agent preference (i.e., allowing agents to decide which other agents they prefer working with locally), voting algorithms and discretionary control over when to end the search for additional information and classify the event. Using these automated approaches, the multi-agent system may adapt to both the operating environment and the attack without restricting the system with hard-coded thresholds.

- An algorithm for detecting the traces of local network cyber attacks at the network flow layer. Currently, anomaly-based and machine learning techniques focus on individual domains within the network (i.e., to detect one particular attack) and cannot generally be applied to detect multiple attacks without suffering from overfitting. The generalised anomaly detection algorithm developed within this thesis is used in conjunction with the multi-agent system to detect and then investigate the early indications of an attack.

1.5 Organisation

The organisation of this thesis is as follows:

Chapter 2 provides an overview and literature review of the three main topics discussed in this thesis: (i) intrusion detection, (ii) advanced persistent threats, and (iii) multi-agent systems.

Chapter 3 presents the novel multi-agent approach to intrusion detection and the concept of domains which is a descriptive model that encapsulates the relationships between technologies within a network. A formal system definition and algorithms for performing an extended data collection task (i.e., the agent approach to evidence collection) are also provided.

Chapter 4 describes the novel generalised anomaly detection algorithm which aims to detect a wide variety of attacks during the early stages of a system penetration. In particular, the algorithm aims to address the problem of overfitting within cybersecurity research. Details of the simulator in which the algorithm was designed is also provided.

Chapter 5 contains the results and discussion of the multi-agent system and the anomaly detection algorithm.

Chapter 6 concludes this thesis with a discussion of the findings and possible future work.

Chapter 2

Literature Review

In this chapter, an overview of the current state of the art in related research is provided. Several topics including multi-agent systems, network security and machine learning are included among the discussed areas of research. Critical networks, such as large-scale utility, business critical and supply chain networks, require protection from increasingly stealthy and sophisticated security threats. While networks with many separate components, potentially spanning over a considerable geographical distance (in the case of utility and supply chain networks) provide many benefits to the users, they exponentially increase the attack surface and opportunity for system compromise. The importance of protecting networks from the APT has been highlighted in recent years with many high profile security breaches; including Stuxnet, Duqu, Flame and Red October, discussed further in this section. New techniques are required to provide sufficient protection of vulnerable networks as it is apparent that existing methods do not meet the security networks requirements.

2.1 Intrusion Detection

This thesis aims to improve the performance of intrusion detection, to facilitate this, an understanding of existing intrusion detection techniques is necessary. This section discusses intrusion detection concepts that are currently used throughout the security industry and research.

Many frameworks have been proposed to model and manage cybersecurity risk for critical infrastructure and organisations. NIST proposed an industry-backed framework for governing risk management processes, of which the framework core describes a set of cybersecurity activities and desired outcomes that are common across many sectors [120]. The framework core consists of five concurrent and continuous functions: (i) identify, (ii) protect, (iii) detect, (iv) respond, and (v) recover. Furthermore, the framework describes three implantation layers that security

should be performed on: (i) the executive level, (ii) the business process level, and (iii) the implementation/operations level. The framework recognises the multiple attack vectors that exist within every organisation, for example, those that exist at the executive level, the business process level or the implementation and operations level which may become the target of an attack. The framework is particularly applicable to modelling advanced attacks that target specific organisations. The approach taken in this thesis is to address the threats that exist within the technical operations layer; however, many of these threats may exist due to vulnerabilities that exist in the higher business process or executive layer.

Intrusion detection has branched out into several areas to address the increasing amount of threats facing modern networks. Figure 2.1 shows a high-level view of intrusion detection system architectures and detection methodologies.

System Deployment An IDS may either be deployed centrally or distributed across several devices. Host-based IDSs are deployed on a per-device basis to detect host-level attacks and some device-specific network-level attacks. Network-based IDSs, on the other hand, are deployed at a bottleneck point on the network, typically at the network's perimeter so they can evaluate all foreign traffic originating from outside of the network as well as outbound traffic from the protected devices. Network-based IDSs have arisen as the most popular choice in recent years owing to their ease of deployment, high visibility and easy administration. In comparison, host-based or decentralised systems are most costly to deploy and administrate due to the increased volume of individual devices that must be monitored. Other technology types, including wireless-based IDSs are used for specialised architectures but function similarly.

Detection Methodology Of the several approaches used to detect attacks signature-based detection has arisen as the most common. Using manually defined signatures created by domain experts, individual attack patterns are described and used to detect future occurrences. An example of a signature from a leading IDS named Snort is included below; in particular the "content" field describes a string of encoded data that is to be searched for within the network packets.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 139
flow:to_server,established
content:"|eb2f 5feb 4a5e 89fb 893e 89f2|"
msg:"EXPLOIT x86 linux samba overflow"
reference:bugtraq,1816
reference:cve,CVE-1999-0811
```

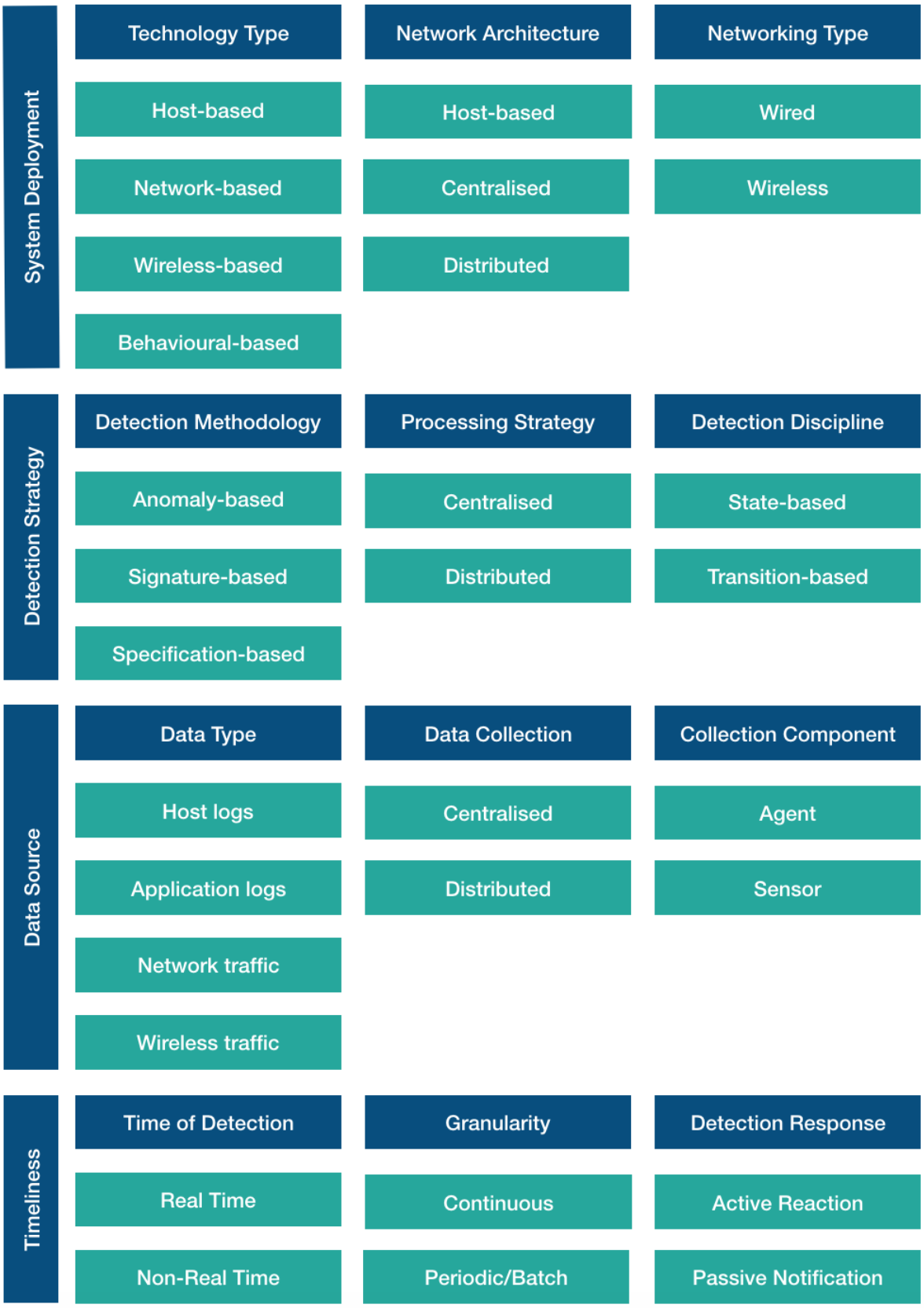

Figure 2.1
Intrusion detection attributes.

```
classtype:attempted-admin
```

Since signatures are manually defined to fit a particular attack, they are vulnerable to a number of evasion attempts including encrypting the data to bypass content filters and manually changing the attack in minor ways so it no longer matches the signature. To address these concerns, new methods including anomaly and specification-based methodologies have been used to improve scalability and detect attacks without relying on specific signatures. The specification-based approach uses examples of normal system behaviour and expected protocol interactions to detect deviations from predefined norms [154], whereas anomaly detection uses heuristics or rules that specify the form of malicious behaviour to detect attacks. The system is first taught to recognise regular activity and builds a behavioural model around it, deviations from this model are then processed as malicious. Typically, the IDS is parameterised by a threshold T that is adjusted to control the detection performance. Adjusting the threshold to make T higher would have the effect of potentially increasing the detection rate but at the cost of more false alarms.

Timeliness Anomaly-based IDSs require training time to build the behavioural model of the network. This overhead often hinders their usefulness as training can often be both a computationally and temporally expensive process. On-line/real-time detection systems are capable of learning and detecting while processing live networks events whereas off-line systems require prior training or extra time to process the collected data. Additionally, a system may process network events as a continuous stream of information or perform periodic batch processing after a certain number of events have been detected. A trade-off between the storage cost of remembering events and the computational cost of continually updating the system model or performing signature matching is made when deciding the granularity.

Data Source Processed data may come from a wide variety of sources within a network. Network traffic (e.g., packet flow data) is the most visible source and contains all information moving through the network. Host-based data such as system commands, system logs and security logs are often used for specification-based IDSs due to the detailed information regarding a wide variety of system events. Data may be collected either centrally or through a distributed network of systems, however, distributed host-based systems may be vulnerable to disruptive DoS (Denial of Service) attacks or network congestion while central systems may become vulnerable to bottlenecks when processing a large volume of data.

It has been argued [83, 63, 145] that the current state of IDSs falls short of effectively defending systems from compromise due to the lack of intelligence, adaptability and proactive

capabilities. Current IDSs[1] typically fall into two categories: detection by signature recognition or detection by anomaly recognition. Unfortunately, IDSs have not evolved in any significant way beyond the simple rule and anomaly-based systems that were seen over 20 years ago. Many authors [21, 60, 145, 83] have noted that the current IDS technologies do not go far enough in investigating potential attacks. Bass [21] called for IDSs to incorporate multi-sensor data fusion techniques to detect more of the environment through the use of targeted sensors. Frank [60] highlighted some of the problems around IDS data interpretation, noting that data reduction, a highly expensive computational process, is a major problem due to the high volume of data that the IDS will observe. Rather than performing bulk collection, a more targeted approach is needed whereby the data is intelligently collected and filtered out before being stored. Kautz et al. [83] called for intelligent systems to proactively monitor the environment and interpreting the data to infer the possible goals of an attacker. This type of inference and proactive technology has not yet been applied to the area of intrusion detection on any wide scale.

Spafford et al. [147] described desirable characteristics necessary for intrusion detection, including fault tolerance, resistance to subversion, minimal overhead, conceivability, adaptability, and autonomy. Many of the desirable properties overlap with those of a MAS (Multi-Agent System). Spafford details an example architecture, AAFID (Autonomous Agents for Intrusion Detection), consisting of four components that perform data collection, controlling transceivers, monitors and filters that format data for use. Agents operated within a hierarchical structure whereby agents report to transceivers, who in turn report to the monitors, which are overseen by human operators. Testing highlight a performance bottleneck that occurred during the dissemination of information and was improved upon by introducing a side channel for communication and the delivery of control instructions. The performance bottleneck is a common problem associated with top-down hierarchical control structures that fail to scale well to dynamically changing or large environments.

Helmer et al. [70] designed an agent-based IDS using lightweight agents that have a smaller computational footprint but are less proactive in their abilities. While lightweight agents are less proactive, they have a shorter development cycle and are less costly to transport in production environments due to their small footprint. Further consideration was given to the unreliable network security environment that may not provide agents with an active network connection at all times. Helmer uses a system of report buffering to aggregate reports made by the agents offline until such time that a connection can be made to the broader MAS. Availability in networked environments can be distributed by many scenarios including both unintentional network issues

[1]Note: The properties of an IDS also generally apply to IDSs as an Intrusion Prevention System is an extended IDS with preventative capabilities.

arising from faulty technology and the intentional disruption of services by an attacker to prevent communication from taking place.

Bass et al. [21] criticised the way that IDSs defend systems calling them "primitive" due to their lack of sensors capable of interpreting the operating environment. Stating that they are not able to detect sophisticated attacks, which has been proven numerous times in the media, he called for the integration of multiple sensors that go beyond the current paradigm of rule-based and anomaly-based detection. Examples of desirable sensors included performance sensors for analysing the false positive rate, temporal sensors for distinguishing between multiple cyber attacks over time, tracking models for following adversaries through a network, target revisitation sensors for tracking the interaction between targets and systems, and other various measurement sensors. Sensor-based systems necessarily involve a certain degree of inference when interpreting the collected data. Bass shows that threat analysis involves a high level of inference to defend a system but little to attack it. Bass calls for a more systematic approach to threat analysis stating the need for data gathering methods to eliminate unlikely associations, association methods to quantify the similarity between events, assignment methods to declare beliefs about events, deviation analysis to analyse statistical norms and suggested the use of machine learning to go beyond this.

Kothari et al. [96] highlighted the problem of security systems that are incompatible with the people that use them. Given the example of a security administrator that forces the use of long passwords, users may subvert the intended security controls by reusing passwords or using character repetition to ease the burden of complying with the new controls. Kothari proposed that a study looking at the differences in population groups, focusing on identifying users prone to using evasive techniques, and to find the "tipping point" that users would find acceptable. The proposed system monitors the user's activities on a system and attempts to infer their current state of mind as a way to pre-empt workaround attempts on behalf of the user. Kothari attempted to understand security needs from both a social and technical perspective which resulted in a better understanding of how users would react to security controls and improve the overall effectiveness of the systems.

Orfila et al. [122] proposed a MAS using three agents to detect and respond to intrusion attempts. A predicator agent makes judgements about attacks based on probabilistic models of the environment and sends them to the evaluator which aggregates the data and makes recommendations to the manager agent who takes action against the possible attacker based on the security policy. The system is organised hierarchically and could benefit from a more adaptive organisation of agents such as the use of weighted relationships that put more trust in

agents that have performed well in the past [174, 159].

Vecchiato et al. [157] have designed a modular MAS with three layers: the personal layer, the ambient layer and the cloud layer. Having the various sensor (data collection) agents constrained to one of these three domains and designing the system using secure software methodologies, allows the designer to better constrain the agent scope. Comparisons can be drawn to cloud computing where the personal agent hands-off any resource intensive computation to the cloud layer where the system resources can be scaled up to match demand.

MASs are particularly suited to the domain of legacy technologies due to their flexible nature. Given the example where two systems need to be connected to work correctly, coalitions of agents could be formed to bridge the gap between the systems. This also has applications within the security domain as networks of systems often comprise of a plethora of different systems and technologies. To effectively interact with all of them it becomes necessary to have a system capable of forming dynamic coalitions of agents that have interfaces to interact with the systems. Li et al. [101] used a MAS to encapsulate old legacy technologies which are no longer supported by the vendor or the company using them and could pose a security risk to the organisation because of the lack of software patches. The proposed system includes a collection of agents for performing various tasks, such as the kernel agent for processing the low-level system commands and a collection of local resource agents for managing input and output. Having agents capable of taking over the processes associated with systems to apply an extra layer of security could be a useful tool in the intrusion prevention effort. In a security context giving agents the ability to encapsulate sensitive data when suspicious conditions are met would be a useful automatic response feature.

2.1.1 Digital Forensics

Computer forensics is tightly coupled with intrusion detection often providing the backbone for informed improvements after an attack. Often the task is manually performed by a forensic investigator, who cannot provide actionable information fast enough for any real-time decisions to be made. One of the most important problems addressed by computer forensics is attacker attribution: the process of attempting to learn about the source of an attack and the identity of the person perpetrating it. Forensic investigators will often interact with services available on the Internet as well as local systems to learn more about the attacker. Some of the services and technologies that could be used to gather information about the attacker are: DHCP (Dynamic Host Control Protocol) logs, active directory events, web proxy logs, hardware tracking software logs, Wireless Access Point logs, email logs, Virtual Private Network logs, hard drive analysis,

peer-to-peer logs, anti-virus logs, router interrogation, DNS (Domain Name System) interrogation, authentication logs and more. Currently, when intrusion detection is performed, most of these services go unused, even though they could be used to learn more about the possible attacker. MASs can often benefit the forensic process by aiding in the data collection process. Whereas traditional IDSs collect data centrally, they often suffer from visibility problems because it is limited to monitoring the location that it has been placed in, for example, on a host device or a section of the network. With increased mobility, the MAS can more easily move between devices to gather data, as well as access the internet to perform extended collection.

Shakarian et al. [140] described a cyber attribution architecture [141, 142] that takes into consideration many data sources and use MASs to reason about the origin of an attack through the use of agent reasoning. The agents have access to two models: the EM (Environmental Model) and the AM (Analytical Model). The EM represents the world in which the agent operates in and the AM represents the model that it used to reason about the data collected from the EM. The EM is much broader than usually seen within MAS taking into consideration factors such as a given countries investment into technology and capability to perform a cyber attack. A predicate notation is used to represent statements with weights assigned to them, for example, *hasMseInvest(X)* is the predicate used to represent whether the actor 'X' has a significant investment in math-science-engineering education, for each predicate the agent assigns a probability distribution to represent beliefs about the model. The agents, when reasoning about the data, can take into consideration both facts and presumptions; a judgement consisting of mainly facts would have a higher weighting than one consisting of many presumptions. A noted problem with this is the dimensionality of the data which can multiply as more variables are considered (how to best analyse this data is an open problem within the literature). Additionally, it was noted that of contradictory facts, given a scenario where two agents report two contradictory facts, being able to reason about which is valid is a challenge. This is especially important in the intrusion detection context as the attacker may leave false data behind to fool forensic investigators (and agents).

Shanmugasundaram et al. [143] develop a distributed forensics system using a hierarchical approach with multiple configurable sensors placed on the network. Current approaches are criticised because (i) the large volume of data makes prolonged storage, processing and sharing of information infeasible, (ii) incomplete logs caused by security events without signatures (i.e., zero-day attacks), (iii) long response times due to the requirement of manual investigation, (iv) lack of mechanisms to share logs due to fears of information leakage, and (v) unreliable logging mechanisms on host systems that can be circumvented. The proposed architecture named ForNet

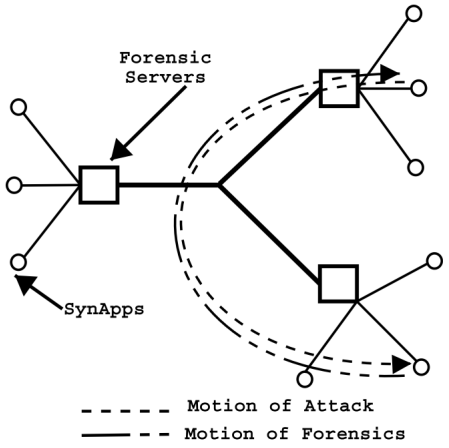

Figure 2.2
The ForNet heirarchical structure

is composed of two components: a system to summarise network events in its local vicinity (SynApps) and a centralised server for management (refer to Figure 2.2). The architecture provides a middle ground between long-term storage (through the server component) and a distributed net of worker agents that provide the detection capability (through the SynApps).

Oriwoh et al. [123] propose a forensics response model tailored for IoT (Internet of Things) environments taking into consideration the increased scope and complexity faced by forensic analysts. Performing manual forensics by trained practitioners can be a time-consuming task in an IoT environment where great distances can physically separate the devices, and the number of devices can be many. The proposed architecture uses evidence to inform future searches has been modelled using a MAS to automate the forensic process in IoT environments. This model is beneficial over the traditional IDS approach of analysing security events where data is analysed according to known signatures or anomaly analysis without the forensic feedback loop to help inform future analysis of relevant data.

Jahanbin et al. [78] introduce an agent architecture 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 [67], 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.

2.1.2 Advanced Persistent Threat Modelling

In recent years, the complexity and effectiveness of cyber threats and attackers have increased with advanced tools made available to the public. The APT is an example of this growing sophistication which sees well planned and targeted attacks with a view to compromise a business or industry completely. The term "persistent" is used to describe the use of advanced evasion tools for bypassing security mechanisms undetected which allow the attack to remain within the target network for a protracted amount of time. With attacks remaining undetected for a significant amount of time, attackers can fully explore the network and achieve their goals. Combined with the use of zero-day exploits, previously unknown to security vendors, the attacks are much harder to defend against and monitor for when no signature exists for detecting traces of the attack. Currently, solutions that merely combine traditional technologies such as IDSs and firewalls have proven unable to detect advanced attacks where the attacker is adaptive to the security solution. This calls for more adaptive and intelligent solutions to detect attackers in different ways without relying on static IDS signatures that can be evaded. While APTs may make use of zero-day exploits and stealthy mechanics to remain undetected, they must still traverse the internal network and often times make use of existing tools to achieve their goals. While directly detecting APTs that use zero-day exploits is difficult, the approach advocated in this thesis of detecting the network traversal movements can be used to infer the presence of attacks and signal further investigation for in-depth automated evidence gathering systems.

Attack Model

APTs are deployed in stages that follow the standard 6-stage attack cyber attack model of (i) reconnaissance, (ii) initial compromise, (iii) command & control, (iv) lateral movement, (v) target attainment, and (vi) exfiltration. While this process is well known to the security community, subtle differences in the deployment of APTs and regular attacks at various stages make the former much harder to detect.

Information Gathering The starting point of an attack is to gather as much information about the target as possible [24]. Many sources of information exist with a variety of open source tools available to perform intelligence gathering (known as open source intelligence gathering). Since APTs are persistent and aim to remain stealthy within the network, several approaches to gathering information exist. Passive collection refers to the monitoring of network communications without directly interacting with them. Passive information gathering only detects and does not leave a footprint within the network. Semi-passive information gathering performs

similar monitoring of the environment but accelerates the process by generating traffic that would not usually be considered suspicious within the network (e.g., DNS or DHCP queries). The introduction of these packets into the network invokes a response from the target device which can be interpreted to gain information about the system. The most popular example of this is network scanning using tools such as Nmap [108] which sends probe packets of different kinds to a device and interprets the packet features (e.g., window size, maximum payload size, and the type of response) to determine the operating system in use or versions of software operating on a particular port. Tools such as Nmap are also used for the most intrusive example of information gathering, active scanning, which sends a variety of packets to many different ports and devices to force the response of the system to gather as much information as possible. Many systems exist to detect this type of scanning behaviour and are often used to detect the initial stages of a larger cyber attack. As a result, APTs tend to make use of either passive or semi-passive information gathering to remain undetected and produce only a small detectable footprint. While the APT approach results in a slower compromise, the advantage of APTs are their persistent nature allowing them to remain within networks for an extended period.

Exploitation Once the initial compromise of the system's perimeter has been achieved, a RAT (Remote Administrator Toolkit) is typically used to retrieve additional software from locations on the internet for use within the local network. With APTs focusing on remaining undetected, additional software is downloaded slowly over time to avoid detection. Once an initial foothold is made within a local network, the compromised device is used as a platform for launching attacks against other, more valuable, targets within the network. Attackers often take the path of least resistance when compromising systems, i.e., they compromise tertiary devices such as outdated systems, utility systems and printers to gain the initial access and then spread throughout the network.

The initial attack may take many forms, including:

- **Spear Phishing**. A targeted attack directed at a particular host or entity to gain access to the wider network. Often network administrators or employees with a high degree of access such as management staff are the targets of this type of attack. Information gathered from online sources such as social media and company websites are used to tailor the spear phishing attack to the particular target making them highly personalised and effective if awareness training has not been provided. Additionally, forged documents such as emails, PDFs and spreadsheets with official company logos can be used to achieve a sense of legitimacy when encouraging the targets of an attack to click a malicious link or

download a RAT program.

- **Exploiting Web Infrastructures**. As an example of an untargeted attack different from spear phishing, exploits targeting websites with the goal of infecting as many users as possible are used to initiate attacks. The most common examples are XSS (Cross Site Scripting) and SQL (Structured Query Language) injection that use vulnerabilities in web server scripts and database interaction languages. Specifically, XSS is used to upload foreign code to websites, often through the use of iFrames (embedded JavaScript code in a webpage) that can redirect the target users to an external site or initiate the download of malicious files to the user's device. SQL injection can be used to reveal private information from the targets web server, for example, by exposing system passwords, non-public files or by allowing the remote upload of malware directly to the server.

- **Communication Protocol Exploits**. With the goal of remaining undetected during the attack, network protocols can be used to circumvent the normal flow of operations. Using common network protocols to mask lateral movement (i.e., moving between devices compromising them) within the network or retrieve RAT software from a repository online are common uses for exploited protocols. Simple Mail Transfer Protocol servers, primarily used for the processing and delivery of emails can be used to relay malicious software between networks. HyperText Transfer Protocol and File Transfer Protocol servers and their associated protocols can also be used for delivering malicious software across the internet and local networks, since these are examples of common protocols that are found during everyday use, their presence does not raise suspicion making the malicious contents harder to monitor for. Furthermore, low-level communication protocols such as DNS and Address Resolution Protocol can be used to redirect users or create a man-in-the-middle attack to make a copy of a users network traffic for further information gathering.

- **Mobile Exploits**. With the expansion of modern networks to include mobile devices, the traditional network perimeter that was protected by IDSs and firewalls has expanded to include a variety of networked devices using a variety of operating systems of software. With mobile devices not being under the control of the system administrators, a variety of software can be brought into the network which could be used as an entry point into the protected subnets. Mobile protocols such as WiFi and Bluetooth can be used to subvert security and gather information about devices within the network as well as the mobile device itself.

- **Physical Attacks**. When remote exploitation is not possible, physical attacks can be

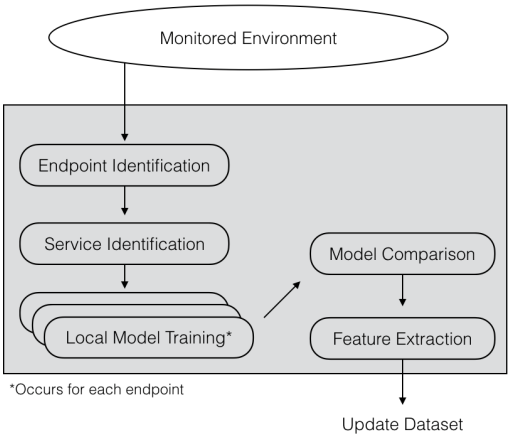

Figure 2.3
Proposed model for feature extraction.

Table 2.1
Examples of Advanced Persistent Threats (APTs)

APT	**Category**	**Target**
Stuxnet	Nuclear control system	PLC software
Duqu	Targeted information gathering	Individual computers
Flame	Windows-based malware exploitation	Windows computers
Red October	Diplomatic, scientific and governmental-targeted malware attacks	Individual computers & mobile devices
RSA Attack	Industry-targeted spear phishing attack	Email accounts
Operation SnowMan	XSS web based	Public website

used to gain entry into the network. The most common example is to install malware onto a USB device and leave them in public places to be found. When brought into the protected network, the malware rapidly spreads throughout the network and "phones home" to download additional malware for use in compromising the wider network.

Technical Analysis

In recent years, several high profile APTs have been discovered and analysed. Virvilis et al. [161] provide a high-level technical analysis of the Stuxnet, Duqu, Flame and Red October. In this section, an overview of each is provided.

Stuxnet First detected in June 2010 after development began 5 years earlier, the APT specifically targeted Iranian nuclear Supervisory Control and Data Acquisition systems. The malware was spread throughout the internet but would remain dormant in the absence of Siemens Step7 software or PLC. Upon detecting either the software of PLC, the malware would reprogram the PLC to cause physical damage to the nuclear centrifuges. The malware was completely autonomous; after release and made use of four zero-day exploits to breach the target's network. Based on the expertise required to build this malware, many researchers have stated it would require state sponsorship to develop.

Duqu The Duqu malware was detected in September 2011 after having been active for over a year unnoticed. The malware was more selective in its targeting by only infecting about 50 devices worldwide and included forensic countermeasures that would cause the malware to self-destruct after 30 days. The primary focus of the malware was to perform information gathering through keylogging and uploading the information back to a CnC (Command and Control) server using a customised encryption protocol over port 443/TCP. Furthermore, the malware would scan systems for a list of security products and use a rootkit to hide its files according to the anti-virus software detected.

Flame Detected in May 2012 after being active for 5-8 years prior, the purpose of this malware was for comprehensive information collection from Windows systems, mainly targeting Middle Eastern countries. The malware impersonated the Windows update server through a complex cryptanalytic attack to spoof the malware's digital signature. More than 80 CnC servers were used for the collection of gathered information reflecting the scale of the APT. Similar to Duqu, the malware scanned for more than 100 security products and adapted the rootkit to evade detection.

Red October Detected in 2012 after being active for over 5 years, the malware had been targeting diplomatic, scientific and governmental institutions. The original malware had a very small footprint and could download many extra modules from a CnC server tailored to the network environment it found itself in. In particular, the malware targeted individual devices and mobile phones and brute forced the Simple Network Management Protocol.

Operation SnowMan Detected in February 2014 made use of a zero-day exploit (CVE-2014-0322) served from the U.S. Veterans of Foreign Wars' website targeting military personnel. Visitors to the website were redirected through a XSS attack to an alternate site where a user-

after-free attack was performed to run arbitrary code on the victim's machine.

RSA Attack Detected in 2011, employees at the RSA company received an excel spreadsheet through email with a zero-day vulnerability attached. The zero-day vulnerability installed a backdoor virus exploiting a vulnerability in the Flash technology. The attackers used the backdoor to harvest credentials from RSA employees to target high-value targets within the network. The method of delivery (i.e., email) is an example of a targeted spear phishing attack that tricked the user removing the email from the spam folder and opening its contents

From the four APT examples, a clear distinction from traditional attacks can be made. APTs are typically well planned out, well resourced and have a particular purpose to achieve some goal. This is different from traditional attacks that are usually less focused and aim to infect as many devices as possible.

An APT is differentiated from normal cyber attacks by the adversary who possesses sophisticated levels of expertise and significant resources to achieve their objectives through multiple attack vectors [48]. Chen et al. [37] describes how APTs have evolved and are no longer limited to the military domain and distinguishes them from regular attacks by the following four features: (1) has specific targets and objectives; (2) are highly organised and well resourced; (3) embarks on a long-term campaign with multiple attempts; (4) utilises stealthy and evasive techniques. APTs are usually performed by groups of skilled adversaries and are meticulously planned, involving multiple steps to achieve the goal. Typically, an APT will involve the following six stages to achieve the goal: (1) reconnaissance and weaponization; (2) delivery; (3) initial intrusion; (4) command and control; (5) lateral movement; (6) data exfiltration. As the connectivity of systems grows through concepts such as IoT and Industry 4.0 [106] the threat of APTs continue to grow as large connected architectures enable the attackers to spread through the network of systems causing increased amounts of damage.

Bhatt et al. [24] use a semi-automatic event correlation architecture to detect multi-stage APTs following the CIKC (Cyber Intrusion Kill Chain) model of system penetration (i.e., information gathering, weaponisation, delivery, exploitation, installation, command and control, further actions) [73]. The event correlation architecture uses algorithms to track attacks between the layers of the CIKC model using data obtained from system and network logs; however, Bhatt notes that the automatic analysis of the events is an ongoing research issue and so requires the input of an experienced analyst to make the final classification. The architecture is composed of an initial logging module that aggregates the log files of many sources to feed into a hadoop big data cluster. Data is then read into an intelligence module with patterns from the kill chain model for malware analysis. It was noted that using the CIKC model as a predictable architec-

ture for intrusion allowed for better tuning of the system by focusing the attention of the IDS on the stages that analysts would commonly expect an intrusion to target.

APTs [61] are differentiated from traditional methods and compromise and malware as they are deliberately slow-moving cyber attacks with the purpose of quietly compromising information systems without revealing itself. This is different from the more general attacks that are more commonly seen that might have the goal of disruption through a denial of service attack or to make a host part of a botnet. The APT is a more targeted, stealthy and well thought out attack that is often aimed at corporate organisations to steal sensitive data. Organisations often fall prey to this type of attack due to their reliance on traditional perimeter model [74] which is often very secure on the outer layers (using IDS and Firewalls) but "soft" on the inside (using limited security technologies) resulting in pivoting attacks being very effective.

During a review of security processes, Singh et al. [145] criticised the reactive way in which security is currently performed throughout the industry. Specifically, attention is drawn to the vulnerability management cycle of discovery and patch which treats each event as independent and results in the security of the system always behind a step behind new attacks. Singh calls for a more holistic approach by stating that emphasis should be placed on understanding human behaviour to detect threat prone behaviours which are regularly a target for adversaries that will follow the path of least resistance which can be used as a methodology to detect attacks. The cybersecurity ecosystem is described as a balance between two actors: the stakeholders and the adversaries, and three layers that could be exploited: the social architecture, the users and the technical architecture. It is noted that both the social and technical architectures rely on each other to not be compromised to function securely, for example, the user may subvert some technical control, such as the classic example of reusing or incrementing passwords.

Geib et al. [63] building on the work of Kautz et al. [83] argued that having a passive IDS that cannot predict attacks or be used as an early warning system is not sufficient for modern network security. Geib proposed a system for attack prediction that using inferred goals to determine the possible future action that the attacker may take. Through a hierarchical tree structure representing possible goals that the attacker may have, with broad goals such as vandalism or theft at the root and simple actions to achieve these goals as the leaf nodes, it is possible to infer previous actions taken by the attacker as a method to trace attack propagation through the system. More complex attacks were modelled through the use of Markov Chains allowing multi-goal attacks to be modelled on a probabilistic scale. While the system provided an analytical tool for understanding attacks, it did not adequately consider that attackers are uncooperative actors within the system and will attempt to remain undetected and subvert

detection mechanisms. The model did not account for attackers that are aware of the system and able to subvert detection by either planting evidence or misleading the sensors into concluding the incorrect goal.

Mees [112] designed a MAS to detect APTs by using the DNS protocol to look up the origins of suspect connections. Within the architecture three agents are described: the first consult external DNS servers to evaluate the location of the host IP (Internet Protocol) address, the second compares the connection characteristics to previously seen connections of the same protocol and the third distinguished between robot-like connections and human-like connections by consulting a database of connections and performing anomaly analysis. Using agents in this way to perform active information reconnaissance on suspect entities would be an advantageous feature to have as part of a wider architecture.

In a separate paper by Mees [113] et al., a more extensive architecture for detecting APTs was designed using multiple agents with limited functions such as log analysis and DNS lookup capabilities to work together to gather a more significant amount of information about a possible APT attack. In particular, the architecture consisted of data being pulled from networked locations and being preprocessed, feature selection performed and then delivered to a network of agents before aggregation and classification. While this system consists of many collection agents, there is less emphasis on intelligently optimising the collection process by analysing the specific type of attack and forming a defence/information gathering plan accordingly. Having agents capable of speculating on the most effective course of action (possibly based on past successes and machine learning) would result in a more efficient and targeted defence system.

Friedberg et al. [61] develop a pattern matching system for use within IDSs. The system attempts to find patterns that can be categorised into (i) periodically occurring patterns (e.g.: timestamps), (ii) rare patterns (e.g.: session ids, or measurement values) and (iii) reoccurring patterns (e.g.: enumeration values or usernames). A system of "ageing", which is functionally similar to genetic algorithm mutation, is used to remove any patterns that are no longer relevant to the search process. Patterns are generated from IP-layer log files, i.e., to recognise the prevalence of GET-requests following a HTTP-packet using semantic processing without knowledge of the meaning (i.e., the system is aware that a GET request occurred but does not understand the implications or purpose of a GET request). While the system provided a true positive rate of 80%, the context-less approach was not able to consider the context of the attack and only capable of highlighting monitored patterns in user behaviour. Slow moving cyber attacks are characteristically deceptive and capable of masquerading as legitimate traffic to remain undetected, furthermore, if an attacker is able to penetrate the network, the possibility of the logging

systems being disabled would render the proposed system inoperable.

2.1.3 Datasets

Obtaining reliable security datasets is not always possible due to the culture of secrecy that often sees security breaches go undisclosed and information not shared. Due to the reluctance to share private network traffic that may contain personal or business-critical information, hybrid datasets containing known secure data and known malicious data are often combined for research and testing. In recent years, synthetic hybrid datasets have been created that combine live network activity with separately classified examples of malicious activity; by using this approach balanced datasets more suitable for machine learning and correctly classified can be developed for intrusion detection research. The main issue facing dataset generation is the ability to label the examples which are increasingly costly to do manually and fraught with inaccuracies when done automatically.

Khalastchi et al. [91] criticised the use of the artificially constructed hybrid datasets for off-line learning stating that the security environment it typically non-deterministic and so could not be properly used to detect future unknown occurrences. Typically, well-known attacks and exploits are detected using signatures to a high degree of accuracy; the ability to detect and prevent new zero-day exploits cannot be learnt from examples of existing attacks. Agent-based systems are commonly designed with an intelligent component with the goal of detecting novel attacks and so require a different methodology for learning the differences between malicious and innocuous.

Lodi et al. [105] highlight the issue of businesses not sharing security information due to factors such as the stigma against security breaches and fear of competitors taking advantage. Lodi proposed a data sharing system named Semantic Room that enables businesses within the same industry to anonymously share security breach information to better strengthen the collective defences. The system is an information gathering an engine for information correlation which allows the anonymous and secure sharing and dissemination of important conclusion-based information derived from the individual networks that it operates in.

2.2 Machine Learning for Intrusion Detection

The intelligent component of a MAS is often derived from ML (Machine Learning) methods of processing data. Many MAS from the literature combine the concepts of ABS (Agent-Based System) and ML in a distributed manner to solve complex problems that have traditionally been difficult to make progress on from a central system. Due to the decentralised nature of the

MAS, observability of the environment is often incomplete, and strategies to learn more about the environment are needed. Typically, it is unrealistic to assume that the environment will be fully known to the agent as the cost associated with maintaining full visibility is high. A common solution is for agents to know only their local environment and share the incomplete views. Using the algorithms outlined in this section, agents can, more meaningfully, interpret the substantial and highly dimensional security datasets that are found on modern networks.

Wu et al. [170] identities that intrusion detection faces problems such as having to analyse a huge volume of variable traffic, which is often of different formats and difficult to uniformly quantify. Wu provided a comprehensive review of various Artificial Intelligence algorithms as they apply to intrusion detection. Since there are many algorithms and many combinations of algorithms that can be used, the ensemble method (intelligently matching the best algorithm to a dataset) was used to help choose the best type of algorithm for a specific input and goal.

2.2.1 Fuzzy logic

Fuzzy logic [170] has been used in intrusion detection as a way to classify vague and imprecise data that is often countered within security systems. One current application of fuzzy logic is in the anomaly detection domain where profiles of normal behaviour are built to compare with malicious behaviour. While monitoring user behaviour, outlier events will inevitably be identified, but are hard to categorise due to the fact the user's behaviour will likely not fall into the "absolutely safe" or "absolutely malicious" category. As a way to reduce false classifications, fuzzy logic can be used to label risk events as increasingly risky or decreasingly risky based on information collected. Instead of merely categorising some risky behaviour either safe or not safe, it would benefit both the system and the user if a more precise measure was used. Fuzzy logic systems are uniquely appropriate for MAS which rely on potentially unreliable observations of the environment that must be quantified.

The main use of fuzzy logic in IDSs has been to aid in the understanding of imprecise and noisy data [10] and the development of intrusion signatures. El-Hajj et al. [50] illustrated this point concerning port scanning, which is one of the main activities intrusion detection systems often monitor. El-Hajj noted that *"no clear boundaries exist between normal and abnormal events"*, which results in non-fuzzy logic returning a higher number of incorrect classifications.

Dickerson et al. [46] developed the FIRE which uses fuzzy logic as the basis for classifying attacks. Continuing with the example of port scanning, rather than looking at whether a port scanning event had occurred and then blocking the connection, the source and destination port along with the service was monitored and based on these three variables the event was modelled

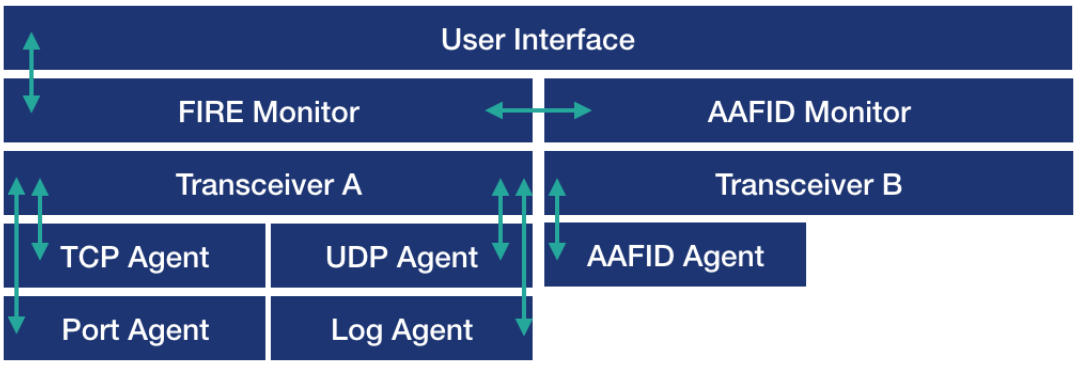

Figure 2.4
FIRE architecture proposed in [46].

using fuzzy logic such that if all three were deemed to be suspicious, then the event would be classified as such. If two of the three variables were classified as innocuous while one was not, then the whole event would likely be classed as less suspicious. This allows for a more fine-tuned classification system which reduces false negatives and positives as events are not prematurely deemed malicious or not based on incomplete information. One beneficial property of fuzzy systems is the combination of multiple sources of information; this lends itself to be useful in the area of MAS, such as seen in the FIRE system whereby specialised agents were used to collect specific pieces of information which are then combined and used as the basis for detection classification. The FIRE architecture (see Figure 2.4) makes use of both the proposed FIRE agents (i.e., agents under the control of Transceiver A) to perform fuzzy intrusion monitoring and also autonomous agents from the AAFID architecture for autonomous monitoring (i.e., agents under the control of Transceiver B) [19]. Agents within the FIRE architecture make use of a common network log source but may analyse different aspects of the file and it is assumed that the system administrator is cable of assisting in the creating of fuzzy rules for use within the architecture.

2.2.2 Neural Networks

Vinayakumar et al. [160] propose the use of a CNN (Convolutional Neural Network) to detect network attacks on the KDD CUP 1999 dataset [84]. CNNs, a technique most commonly used in computer vision, is noted as having the ability to extract higher level feature representations from low level feature sets. Using the one-dimensional KDD network dataset, network traffic time series events are processed to produce a set of features for use in the anomaly classification stage. The authors state that the detection rate of R2L (Remote-to-Local) and U2R (User-to-Root) attacks are lower until 400 epochs have been performed during testing, which highlights

the performance issue of supervised learning for real-time detection in network systems. Furthermore, while the accuracy of 0.684 to 0.885% is achieved, the authors note that attacks within networks change year-to-year which further decreases the scalability of a system that requires constant retraining.

Lee et al. [99] use the LSTM (Long Short-Term Memory) variant of the Recurrent Neural Network, to perform anomaly detection on time-series datasets. The adapted approach is noted as being applicable in time-based systems where anomalies often manifest themselves over a period of time rather than as a single time-point. During the training phase, the training dataset is divided up into three components: (i) the LSTM prediction model (M), (ii) the error vector distribution (N), and (iii) the M-distance distribution which feed into the anomaly detection inference phase. Inference is made by applying the model (M) to a new dataset to make predictions, computing the error vectors of the new data, computing the M-distance between the error vectors and (N), then finally labelling the new data. The approach reports a precision of 0.49 and recall of 0.06.

ART [66, 170] has proven to be an effective method for intrusion detection which implements a series of ANN (Artificial Neural Network)s to cluster the data on the similarity of the input data to map clusters. The current generation of ART systems work much like the ANN and has been used to perform clustering on network data to identify malicious traffic. Xiao et al. [172] used ART in conjunction with PCA (Principal Component Analysis) to perform dimension reduction and classifications of security events. In particular, the PCA component reduced the complexity of high-volume data and the ART component performed sub-classification of security events as a whole, with continuous updates made to the ART model. ART like other ANNs are self-adjusting allowing the nodes to change to reflect the input data.

Aung et al. [14] propose the use of a collaborative IDS based on K-means and ART. The proposed PART algorithm creates rules by making partial decisions trees on the currently known instances (data) and selecting rules that best fit the available data (see Figure 2.5). Aung notes that K-means is one of the most popular algorithms for cluster analysis, in particular, the KDD CUP 1999 dataset used during the evaluation is composed of network flows which could benefit from the clustering approach. While the approach reports upwards of 98% accuracy in correctly identifying attacks in same dataset tests (i.e., training and testing on different portions of the same data), the time to build the model can take upwards of 133 minutes making the approach less suitable for live anomaly detection in network environments where attackers can be particularly adaptive requiring constant rebuilding of the model.

Bukhanov et al. [43] develop an approach to intrusion detection using ART comprised of

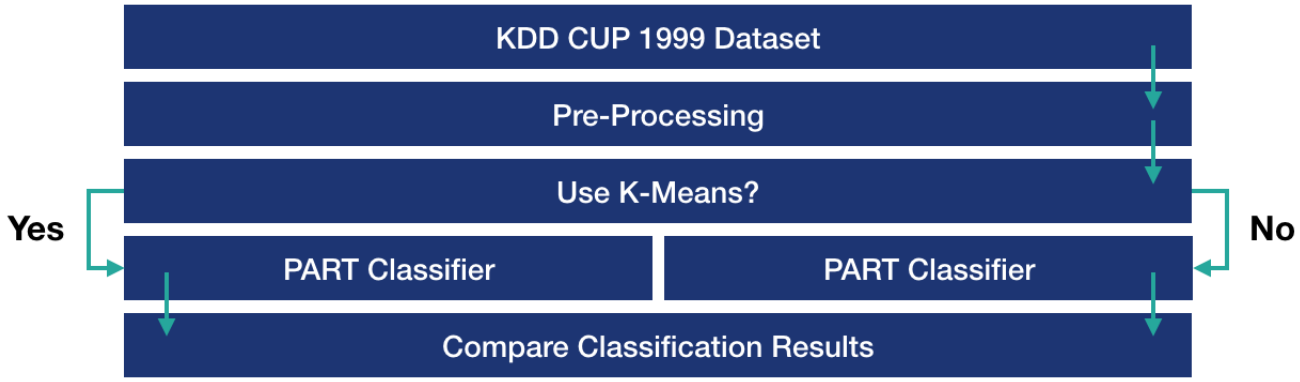

Figure 2.5
ART IDS proposed in [14].

two modules: (i) data acquisition and connection management, and (ii) the network analysis module. The data acquisition module receives network connections and extracts 41 standard features about the connections made. The traffic analysis module uses a neural network based on ART-2 to classify the network connections into clusters. The architecture is proposed as a notification system that will inform the compromised host after analysis has taken place. The detection rate varied from attack-to-attack, however, the majority of attacks were detected at a detection rate of 0.74-0.96 while attacks that did not exhibit a large footprint (i.e., stealthy attacks attempting to remain hidden) only produced a detection rate of 0.03-0.48. While the system has shown to be useful in detecting a broad number of network threats, any attacks that are artificially slowed down to avoid detection are able to evade classification making the system unsuitable for APT detection.

Ashfaq et al. [13] propose the use of a semi-supervised learning model using a single layer feed-forward neural network to classify both labelled and unlabelled network security data. Ashfaq notes that obtaining labelled network intrusion data is a cumbersome process requiring a high degree of manual effort to label the dataset. The proposed semi-supervised model attempts to classify new data from models built using both labelled and unlabelled data using a fuzzy approach. The algorithm is tested on the NSL-KDD test dataset (specifically KDDTest+ and KDDTest-21), producing an accuracy of 84.12% and 68.82% on each test set.

2.2.3 Evolutionary Algorithms

EC (Evolutionary Computation) [170] attempts to mimic the evolutionary process through the use of adaptive pattern recognition. EC algorithms can be optimised to a local minimum/maximum through the process of crossover, merging one parent process with another to produce a

direct child process with attributes of both parents, while also converging to a global minimum or maximum through the act of periodic mutation, the introduction of periodic randomness into the agent evolution. Associated with each algorithm is a user-defined fitness function which acts as the goal the algorithm works towards. Throughout the literature, it has been noted several times that human operators are often too slow in responding and adapting to cyber attacks. Combining evolutionary algorithms with security systems have arisen as a good way to overcome the issues of non-adaptive security. Chavan et al. [36] developed the Evolving Fuzzy Neural Network combining fuzzy techniques with an ANN which sought to classify security events into their corresponding attack type (e.g., normal, probe, DoS, U2R, R2L). Combined with protocol analysis to differentiate between the events the system was able to classify the events to a high degree of accuracy while remaining adaptive to changes in the future.

Balajinath et al. [18] used EC algorithms to learn individual user behaviours through frequently used system commands. It was shown that users exhibit regularities in the use of systems which can be used to detect the presence of non-legitimate users. As in other papers on intrusion detection [18, 22, 98], a comparison between anomaly detection and misuse detection was made highlighting that anomaly detection often incurs a higher number of false positives and is an issue that must be addressed. The proposed model operates on the premise that a sudden change in user commands could indicate a potential intrusion. User behaviour is continually changing; the proposed method takes this into account by implementing an entropy index which shows changes in behaviour. A higher entropy value would make it harder to predict the next user command. To limit the effects of entropy and to ensure the system does not return too many false positives a 'switch' is used to cap the number of new behaviours learnt at any given time. While the system reported an accuracy rate of 96.8% the data used to test the system did not take into account shared devices where multiple users may interact with the operating system. While this may be a viable way to detect intrusion on systems with only one user, if it was implemented in any standard workplace environment it is unlikely that one person would solely use a device. As with the nature of genetic algorithms, they evolve, it is not infeasible to imagine a scenario where the attacker could reprogram the system by introducing his/her usage patterns over time.

Toosi et al. [153] used EC algorithms in a hybrid neuro-fuzzy system to optimise the fuzzy decision-making engine. Based on a number of factors, the rules generated would either remain within the system or be phased out if they were no longer useful. This is a very important property in a dynamic environment where the structure of the network can change quite rapidly so having a dynamic algorithm capable of changing the thresholds will result in a more timeless

system. Abadeh et al. [1] implemented a similar system whereby the fuzzy rules for an IDS were initially generated, then after each cycle are evolved using the Genetic Algorithm and the least effective rules from each cycle were dropped in place of the newer generation of rules.

Rastegari et al. [134] developed an evolutionary approach to building an evolving ruleset for network intrusion detection. The proposed algorithm builds models on a variable number of features following a feature preparation, normalisation, feature selection, validation and seed selection stage. Correlation-based Feature Selection was noted as being a more appropriate algorithm for feature selecting given the higher accuracy performance that can typically be expected; eight features were selected from the NSL-KDD dataset including: src bytes, dst bytes, num root, same srv rate, diff srv rate, srv serror rate, dst host diff host rate and dst host srv serror rate. The system was evaluated using different fitness functions, the highest performing showed a detection rate of 63.3% and a true negative rate of 97.2% with varying results for the number of features and internal algorithm used.

Papamartzivanos et al. [128] propose the use of genetic algorithms for a network intrusion detection system called Dendron. The genetic approach generates new detection rules based on a process of evolution by introducing randomness into the rule-mutation and population-replacement stages. Several classifiers are proposed for use within the fitness function with decision trees (C.45) being selected as the highest performing and is noted for its advantages of producing straightforward rules that can be easily transformed into signature-like rules for use within intrusion detection. Feature selection takes place using the Information Gain Ratio algorithm and is noted for performing particularly well on network intrusion datasets. The algorithm is tested on both the NSL-KDD and UNB-ISCX datasets producing a detection rate of 98.24% and false alarm rate of 0.75% on the NSL-KDD data and 63.76% and 2.61% on the UNB-ISCX data. Papamartzivanos notes that the UNB-ISCX data is different from prior datasets since it is a more recent creation, reflects contemporary network structures and represents a more complex threat environment which poses challenges to machine learning approaches.

2.2.4 Artificial Immune Systems

The AIS (Artificial Immune System) has been used to perform anomaly detection but tackles the problem from the opposite side of the spectrum. Instead of modelling patterns of normal usage, they model patterns of anomalous data and monitor for any matching patterns. The self-non-self principle made of use in AIS, and inspired by immunology, theorises that the body can detect any cell that does not belong to itself.

Saurabh et al. [138] develop the Efficient Proactive Artificial Immune System based Anomaly

Detection and Prevention System to detect novel unseen attacks. The architecture is composed of three modules: the Repertoire Training Module generates and selects efficient detectors, the Vulnerability Assessment Module creates Detector Agents to evaluate test set instances, and the Response Module which takes action in the cases where Vulnerability Assessment Module find an anomaly to overcome the threat perception. Additionally, the model integrates self-tuning to select the best detectors for anomaly detection. The system was trained and tested on the KDD CUP 1999 dataset [84] and reported a detection rate of up to 100%, however, also a false alarm rate of above 62%.

Wu et al. [169] apply an ANN optimised by an AIS to the domain of intrusion detection using an artificial immune system to optimise the system. IDSs process a large amount of variable-typed data which makes traditional ANNs less suitable since they perform better on similarly formatted numeric data to reduce training times. Wu uses a General Regression Neural Network [148] to overcome some of the problems associated with Multilayer Perceptron and Radial Basis Function networks such as the slow training speeds [22, 169] and a tenancy to fall into a local optimal search. One of the main issues with ANNs for intrusion detection is the required training time. Firstly, good datasets for the ANN to learn from are required, without which building an accurate model of the threat environment cannot take place. Secondly, the fact that off-line training is required hinders the system in online environments where the standard for what is considered malicious or not can change depending on the network requirements and evolution of attackers over time.

Cooper et al. [40] propose the use of an AIS for the detection of network attacks using the KDD CUP 1999 dataset. In particular, the approach recognises the importance of feature separation in intrusion detection and aims to analyse each of the network services separately. It is noted that 64% of services only contain abnormal events, 31% of services contain both normal and abnormal events and 5% of services contain only normal events. During the first experiment, the network services were analysed while mixed, resulting in a maximum detection accuracy of 52.70%, however, when analysed separately, the detection rate rose to as high as 99% for certain services. While the results are encouraging, the construction of the KDD CUP 1999 dataset has been criticised in the literature for its unrealistic data, in particular, the fact that the attacker makes use of so many services (64% of which are used only by the attacker and not by normal users). The footprint left in the KDD dataset is uncharacteristically 'noisy' considering the goal of many attackers, in particular, APT threats, is to remain hidden and leave a small undetectable footprint.

2.2.5 Swarm Intelligence

SI (Swarm Intelligence) [93] studies collective behaviour and optimisation in decentralised systems. There are two main types of SI systems; the first is ant-based technologies that attempt to find an optimal path through a search space by using collaborative exploration, the second is Particle Swarm Optimisation which is inspired by collaborative agents (called particles) that move towards the best global solution in a given search space through the use of local search. SI has applications in classification rules discovery, misuse detection and could be applied to keep traffic of intruder paths through a network. The exploratory nature of SI may also have applications in testing defensive systems by repurposing the agents to penetrate the network rather than defend it.

Swarming agent-based systems are particularly good at operating within networked systems. Haack et al. [67] proposed an ant-based architecture and Cooperative Infrastructure Defence model to reduce the human bottleneck problem while allowing administrators to set an overall policy and keep control of the system. The proposed system uses a hierarchical structure (see Figure 2.6 where humans function as supervisors at the top of the tree, below them are the 'Sergeants' which are agents in charge of a particular sub-domain and have the task of aggregating data for the administrators. Below them are 'sentinels' agents that operate on a single host, and are in charge of the sensors (ants) below them. The sensors are light-weight mobile pieces of software that collects evidence of potential problems or intrusion and feed this data back up the chain. The sensors can communicate with each other via the use of digital pheromones which can attract or repel other sensors. Delays are a common part of any networked system, as shown by Flink et al. [58] this can be detrimental to the effectiveness of the system. The effect of the network delays on agents are twofold, firstly the agent may get trapped within an area of the network suffering from high latency and would be unable to reach its goal. Secondly, this could cause a cascading failure with the other agents that rely on the information processing capabilities of the agent in question. In relation to ant-based systems pheromones are used to direct other agents, if there is a sufficiently longer network delay, the pheromones may 'dissipate' before being received by the intended agents meaning instruction data is lost.

Fenet et al. [56] developed an SI IDS to track attackers back through a network through the use of specialised interrogatory agents. The capability to track attackers is an important goal but can be challenging when anonymising technologies are used, or the attacker is outside of the local authorities jurisdiction. The system proposed uses swarm agents that are activated when an attacker is detected within the network, the agent's path traversing capabilities are used to follow the attacker back through a network using DNS interrogation and other informational

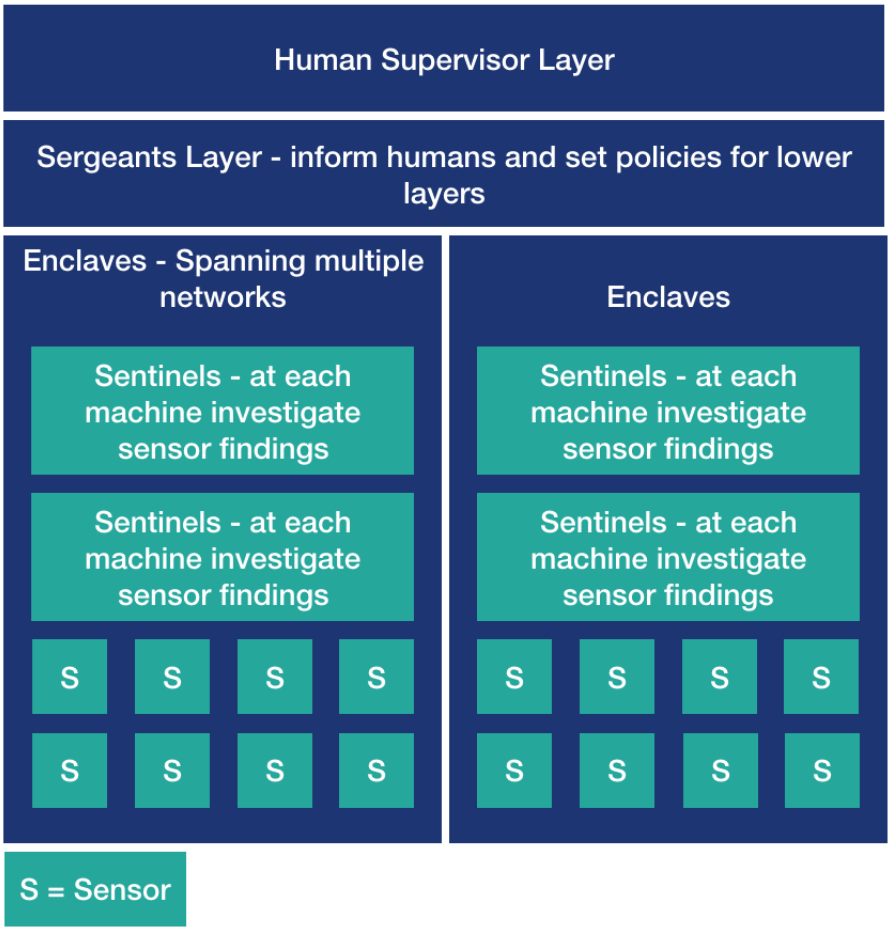

Figure 2.6
Swarm-based architecture proposed in [67].

nodes. The concept of having sets of very specific agents at the disposal of an IDS has shown to be an effective way to perform attack attribution. Designing individual agents in such a way that they become activated when a certain piece of information about an attacker is learned then reporting back to the IDS for further agent activation could provide an efficient way to perform information gathering. This model is good in situations where only a small amount of information about the attacker is known (e.g., a Media Access Control address) as information could be built by performing reconnaissance.

2.2.6 Naive Bayes

Pajouh et al. [126] propose the use of a Naive Bayes classification system for detecting network attacks. By combining the two approaches K-Nearest Neighbour and Linear Discriminant Analysis for dimension reduction, the two-tiered system provides a low-computation training time and good detection rate when monitoring U2R and R2L attacks. Naive Bayes was noted for its good performance when building classification models on low volumes of training data

making the system more appropriate for real-time detection. Pajouh also noted that there was an overlap between the normal and attack classes following the Bayes classification which led to the use of the K-Nearest Neighbour algorithm to further group the instances. While the system could classify certain approaches at a high detection rate (e.g., DoS at 84.68%), complex attacks such as U2R and R2L were detected at 67.16% and 34.81% respectively.

Mehmood et al. [115] develop a multi-agent approach to intrusion detection using naive bayes classification. The architecture uses naive bayes to detect attacks with built-in patterns and makes use of four agent types: the collector, system monitor, actuator and communicator agents. The reported detection probability for the architecture is 70-80% tested on the NSL-KDD intrusion detection dataset. The model is structurally similar to many of the existing hierarchical agent architectures that divide the roles of collection and analysis between different agents and was specifically proposed for the detection of DoS attacks in IoT networks.

2.2.7 Support Vector Machine

Sung et al. [149] used a SVM (Support Vector Machine) in conjunction with a neural network to rank the importance of data points in IDSs. Feature elimination was noted to be a fundamental issue to speed up the processing, enabling the IDS to act more responsively to attacks. Several SVMs were used in conjunction to classify the data, then the features that had the least impact on the results were removed to speed up the detection process. SVMs were used to perform the feature analysis due to their fast speed and capability to handle multi-dimensional data very well. SVMs are more appropriate for basic classification where only a distinction between malicious or not is needed but less appropriate in predictive environments where an increase in risk is analysed. Sung addressed this critique by linking together multiple SVMs to allow for a more precise classification to take place.

Mukkamala et al. [118] performed intrusion detection using an SVM which was capable of identifying a data point as either normal or attack, the problem with this binary classification is the lack of middle ground in the analysis. Mukkamala also performed a comparison to an ANN showing that it was drastically quicker to train a SVM making it more appropriate in a live networking environment and notes that statistical learning of this nature is being deployed more in due to its generalisability.

2.2.8 Other Approaches

Lorliam et al. [107] propose an unsupervised algorithm to apply Benford's distribution law[2] [47] to flow detection using the observation that normal connections tend to follow the law closely while malicious connections do not. Metrics such as flow size and window size are used to distinguish between normal and malicious activities. The flow-based approach to attacker detection is noted to be more computationally efficient than lower level approaches that examine the data in more detail (e.g., packet or host-based IDSs) but can miss application layer information that could be of use to the detection process. Furthermore, Lorliam et al. note that it would be possible for an attacker to change the attack speed to stay below the detection threshold which highlights the underlying problem of relying on features that are under the attacker's control for intruder detection. While the coverage (concerning how much of the network the solution can monitor) and deployability of the solution is high, the manipulability is also unacceptably high and relies on the attacker remaining unaware of detection solution (i.e., security through obscurity). A more robust approach would be to focus on those features that cannot be easily controlled by the attacker and are not affected by network circumstances, for example, network congestion is controlled by network infrastructure and operating systems that send data differently.

Leu et al. [100] propose a profile based behavioural IDS to monitor user system calls and detect anomalous behaviours. The system monitors users from the time they log in to compare the profiles of multiple users to define normal activity. The system reports a recognition accuracy of 98.99% but does not consider the wider security landscape and so can only be used to detect a very limited number of attacks. The IDS, implemented at the host level, assumes a high degree of visibility so that it can monitor both the legitimate and malicious users system calls and compare the two profiles. The approach does not consider the possibility of more advanced stealthy attackers that may launch attacks from auxiliary devices where the IDS is not installed (e.g., printers and mobile devices). Furthermore, the IDS is unable to detect a broader range of attacks such as port scans which do not rely on uncommon system calls such as *format*, *del* or *sudo*[3], and instead, use commonly white-listed network commands to perform the attack.

Cao et al. [34] propose the use of topic models to learn features of normal traffic to perform anomaly detection. Text classification algorithm LDA (Latent Dirichlet Allocation) [28] is applied as the model to learn the structure of the network traffic to distinguish normal from abnormal. LDA is applied by collecting network features in 5-minute intervals and joining the re-

[2]Benford's distribution law states that there are measurable differences between naturally occurring and tampered datasets with which can be used to detect fraud and anomalies.

[3]Common commands that are the focus of system call anomaly detection.

sults to the existing model. This is done locally for each host to reflect the differences in network traffic that will be experienced by devices of different types, for example, separate topic models are created for the mail server and internet proxy endpoints. Validated on the DARPA (Defense Advanced Research Projects Agency) 1999 dataset [116] containing multiple attack types, the system can detect 45 of the 78 attacks from the dataset, but with a detection accuracy of 58% is still unsuitable for broad deployment in live environments.

Rudrapal et al. [137] developed a keystroke monitoring system to distinguish between legitimate and malicious users based on biometric features. The latency between keystrokes, duration of key-press and finger placement were among the features used to learn the biometric behaviours of the user. When tested, the system showed 86% accuracy in identifying different users; however, the system remains susceptible to factors affecting the user such as fatigue, the effects of changing hardware and the possibility that the attacker may capture the user's biometric footprint to bypass the system. Furthermore, host-based solutions such as this are only effective within systems that they are installed. If the attacker were to enter the network through an auxiliary point, such as printer, mobile or legacy device, the detection benefits would not extend to cover these entry points.

2.2.9 Comparison of Machine Learning Approaches for Security

In this section, several key works relating to anomaly detection for cybersecurity are discussed. Table 2.2 summarises related works regarding the techniques that they apply. In particular, each method is summarised in terms of the following attributes: (i) *Type of data* describes the format of the dataset used by each proposal, (ii) *Public dataset* notes whether the data is freely available for comparison, (iii) *User modelling* notes whether the constructed model is user-based (i.e., modelling events as communications between two endpoints) or feature-based (e.g., using packet-level features such as port numbers, payloads, etc.), (iv) *Coverage* describes the scope of the dataset in terms of how much of the network environment is monitored, (v) *Manipulability* is a measure of how protected a dataset is against attacks on its integrity (e.g., the extent to which an attacker can manipulate a feature value and the level of network penetration required to deceive the detection system), finally (vi) *Deployability* measures the feasibility of deploying such technologies in a production scale environment. Additionally, Tables 2.3 and 2.4 provide a comparison of machine learning approaches.

Table 2.2
Comparison of approaches from the literature.

Author	**Data Type**	**Public Dataset**	**User Modelling**	**Coverage**	**Manipulability**	**Deployability**
Lorliam et al. [107]	Flow data statistics	✓	✓	High	High	High
Leu et al. [100]	Log files	✗	✓	Low	Low	Low
Cao et al. [34]	Raw packet flows	✓	✗	High	High	High
Rudrapal et al. [137]	Keystroke data	✗	✓	Low	Low	Low
Heidarian et al. [69]	Raw packet flows	✓	✗	Low	High	High
Lippmann et al. [103]	Keyword data	✓	✗	Low	High	High
Bivens et al. [26]	Raw packet flows	✓	✓	Low	Low	High
Al-Jarrah et al. [6]	Raw packet flows	✓	✗	High	High	High

Table 2.3
Comparison of results from the literature. (Part 1)

Author	**Data**	**Approach**	**Results (DR/FPR)**	**Notes**
Heidarian et al. [69]	UNB ISCX	SVM, Behavioural, HTTP specification-based	89.25% / 8.60%	Compared against only one dataset.
Wang et al. [162]	UNB ISCX	Bayesian statistical modelling (local & global), geographical anomaly detection	74.02% - 89.30%/10.70% - 25.98%	
Pajouh et al. [126]	NSL-KDD	SMOTE balanced, Naive Bayes network anomaly detection	82.0% - 83.24% / 5.43% - 4.83%	Compared against only one dataset.
Pervez et al. [131]	NSL-KDD	Feature selection, SVM based intrusion detection	82% / 15%	Compared against only one dataset.
Tama et al. [150]	NSL-KDD	Gradient Boosted Machine (GBM) based anomaly detection	91.82% - 99.85% / 4.19% - 0.27%	Compared against only one dataset.
Abdalla et al. [2]	Private university network	Ensemble anomaly detection using multiple algorithms	74.0% / 40%	
Anita et al. [10]	Private	Fuzzy intrusion detection for mobile ad hoc networks	59.00% - 99.57%	
Vinayakumar et al. [160]	NSL-KDD	Network anomaly detection using convolutional neural network	68.4% - 88.5%	400 epochs required during training making the approach less suitable for real-time detection
Lee et al. [99]	Social media time series	Deep learning, recurrent neural network, Long Short-Term Memory	49% / 6%	

Table 2.4
Comparison of results from the literature. (Part 2)

Author	**Data**	**Approach**	**Results (DR/FPR)**	**Notes**
Xiao et al. [172]	KDD CUP 1999	ART and PCA based intrusion detection	95.73%	Compared against only one data.
Bukhanov et al. [43]	NSL-KDD	ART based network traffic pattern anomaly detection	48% - 96%	Poor detection performance on stealthy attack with smaller footprints
Ashfaq et al. [13]	NSL-KDD	Semi-supervised learning, feed forward neural network	84.12% - 68.82%	
Rastegari et al. [134]	NSL-KDD	Evolutionary algorithm for intrusion detection	63.3%	
Cooper et al. [40]	KDD CUP 1999	Artificial immune system based intrusion detection	52.70%	Detection accuracy varied depending on the type of attack
Haack et al. [67]	Simulated network	Hierarchical ant-based intrusion monitoring architecture using swarming agents.		
Papamartzivanos et al. [128]	NSL-KDD and UNB-ISCX	Genetic algorithm using decision tree	KDD: 98.24% / 0.75% ISCX: 63.76% / 2.61%	Algorithm performance decreases on modern UNB-ISCX dataset

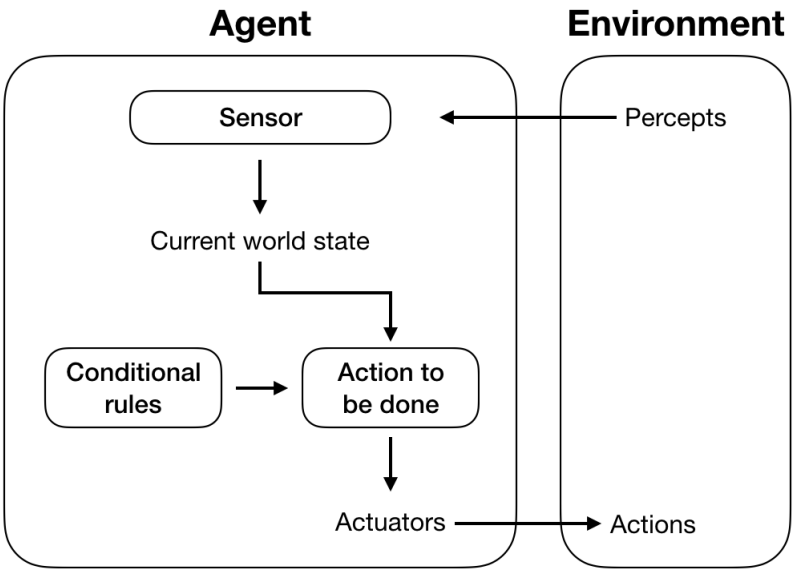

Figure 2.7
A basic reflex agent architecture.

2.3 Agent Based Systems

The term agent is used in many contexts throughout recent works. Rocha et al. [135] describe several examples including economic agents, helper agents for web retrieval, and robotic agents for venturing into inhospitable environments. A common characteristic of the agent is the possession of some degree of autonomy and a sense of agency that goes beyond the typical preprogrammed examples of systems commonly seen. Figure 2.7 shows a basic agent architecture consisting of sensors and actuators for both perceiving and acting within the environment. Agents typically make use of condition-based if-then rules for deciding what action should take place given the monitored state.

Jahanbin et al. [78] build on this definition by stating that agents should adhere to the following five properties:

- The agent should be autonomous having the ability to act independently from the user.
- It should be both reactive and proactive within its environment.
- The agent should be adaptable thereby having the ability to learn according to the experiences encountered.
- It should have the social ability to engage with other agents.
- Agents should be capable of cooperation supporting multiple flows of information to and from other agents.

Using these definitions, a clear distinction between traditional systems and ABS is made by focusing on the intelligent reactive and proactive components that govern the agent's actions.

In an attempt to standardise the definition of a ABS, Poslad et al. [133] published a working definition with the Foundation for Intelligent Physical Agents [76], the standards body setup by the Institute of Electrical and Electronics Engineers [75] to govern technical specifications. Four key characteristics were defined which separates the ABS from traditional software:

- Performance measures (e.g., reactive, model-based, goal-based, and utility-based).
- Their adaptability.
- How they characterise their environment.
- Their degree of autonomy.

Bratman et al. [30] developed a theory of reasoning, known as BDI (Belief, Desire, Intention) as an early example of an ABS. The system is based on agent planning and breaks down all planning activities into the three steps: beliefs are internal statements that the agent believes to be true, a desire is some world state that the agent wishes to produce and intentions are the actions that the agent chooses to perform in order to achieve the goal. This architecture [52] is normally represented as a tree-like structure with nodes representing non-deterministic world states and arcs representing possibly agent intentions. The four standard temporal operators: *next*, *eventually*, *always* and *until* are used within the architecture and extended by Rao et al. [12] to include the temporal operators: *optional* and *inevitable*. With these operators, it becomes possible to represent actions and goals to be achieved in the future.

2.3.1 Multi-Agent Systems

Agent-based system technology has arisen as a valuable paradigm for building scalable and intelligent software and encapsulating complex routines into a distributed package. The capacity of an agent is limited by the available knowledge and the view it has of the environment. Using a multitude of agents with tailored approaches allows problems to be solved using the most appropriate solution or agent. Where a problem domain is overly large or complex, MAS are often the most appropriate solution for performing distributed processing using modular components located close to the sources of information. Wooldridge et al. [165] describe agents as "an encapsulated computational system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives". As described by Ye et al. [173], the definition implies the presence of autonomy that provides

the agent with control over its own internal state and behaviour, reactivity to sense respond to changes in the environment, pro-activeness to perform goal-directed behaviour and cooperation to work with other agents.

Many control systems may be viewed as a MAS providing they contain a sensor and control component. Ye et al. [173] provide the example of a thermostat system containing the temperature sensor component and the decision making components that control whether to turn the heat on or off. MASs have been used in a variety of industrial systems covering a variety of domains and applications.

Self-organisation is the principle of formed cooperative behaviour involving multiple agents without an external control mechanism. Self-organising systems arise in biology, for example, Haack et al. [67] describe a system based on ant colonies that use pheromones to guide groups of agents to a particular goal (i.e., food, water). Within the ant-inspired MAS, individual agents may leave the group to explore new areas of the environment and leave evaporating pheromones to guide the remaining agents when the goal location is found; the pheromones evaporate to prevent agents from acting on old information. The notion of strong self-organising systems that exhibit no central control and weak self-organising systems that exhibit some central control can be used to distinguish between the many MAS available. Many MAS are implemented as a weak self-organising system using layers of agents: at the lowest layer, sensor agents collect information about the environment and deliver the observations to a higher layer that makes decisions about future action based on an aggregation of individual views. While this architecture is easier to control and update due to its similarity with current central systems, it does not fully take advantage of the benefits strong self-organising MAS have to offer.

2.3.2 Multi-Agent Design and Development

Several design concepts encapsulate to describe the mechanics of a MAS. In particular, an agent can be distinguished from traditional monolithic systems by how it interacts with the environment, performs operations and represents knowledge.

Environment The environment may contain challenges for the agent to navigate and interact with. In the context of security systems, the environment is often a network of connected devices with multiple users performing a variety of concurrent actions. Environments may either be described as discrete or continuous depending on whether the environment is fixed in size or continually increasing in size. Observability describes whether the full environmental state can be known at all times or whether in the case of partially observable environments, only a limited amount of information can be known. Many real-world uses of MAS take place in partially ob-

servable environments where the state frequently changes and continuous information retrieval is required. Furthermore, an environment may either be deterministic or non-deterministic which describes whether the next environment state is completely determined by the current environment state or whether additional factors affect future states. The goal of the security domain is to detect malicious activity while remaining efficient as possible in regards to minimising the communications overhead and reducing the number of false positive and negatives, within this, there are multiple environments in which a MAS can operate to collect information.

Sensors & Actuators The mechanism by which agents interact with the environment is through the use of sensors to collect information about the environment and actuators to make changes to the environment. In the context of the security environment, the sensor is the most important component, requiring knowledge of where sources of information can be located.

Knowledge Representation Agents in a MAS are social constructs and require a mechanism to describe and exchange information between one another. The process of building a system to describe information found within an environment is known as ontological engineering and often involves the use of categorisation. While interaction takes place on individual objects, high-level reasoning often takes place on categories of objects, e.g., the goal of security is to find any attackers (the category) rather than a particular attacker (the object). Information is often described as a predicate (e.g., attackers) or the category can be reified as an object (e.g., *attacker(a)*). Therefore, the notation $a \in attackers$ can be used to describe that a is a member of the category *attackers*. The relationship between categories of information is of particular interest to this thesis and is used to allow agents to communicate knowledge about suspicious activity and continue the investigation based on that evidence.

Problem Solving Agents may be viewed as problem-solving systems. In classical environments, this often involves navigation and the avoidance of obstacles to find a goal location. At the implementation level, agents may be regarded as procedural or declarative; procedural agents have the desired behaviour directly embedded in code whereas the declarative approach uses a more flexible system of information sharing, often through requests and responses for information. In classical environments, the procedural approach is often more efficient when uncertainty is at a minimum and the environment remains unchanged over time, however, problems arise when changes in the environment are not reflected in the embedded code. The declarative approach is better suited to handling change and uncertainty but at the cost of an information-sharing overhead through an agent communications protocol, availability issues and

the acquisition of imperfect information. In the security domain, with the recent exponential increase of new technologies and services available for use, the number of attack vectors that require protection has likewise increased requiring more systems to detect the potential attacks. The expansion of technologies has placed pressure on the traditional procedural way of doing security, through embedded signatures used within an IDS.

Local & Global Knowledge The benefit a MAS arises from the disjunction between local and global states. While current examples of security technologies, such as IDS are monolithic and have only a global memory for all types of knowledge, a MAS can be broken up into individual agents each containing a local memory of specialised information, together combining to describe the global knowledge. As discussed in the section on knowledge representation, information is often described in relation to its categories and specifically, how those categories relate to each other. With the architecture of a MAS distributed in nature, it lends itself to maintaining the separation of categories which brings many benefits to the system as a whole. When specialisation can take place, e.g., a group of agents that perform well in relation to the collection and analysis of some category of information, the system as a whole can benefit from performance-based mechanisms to encourage the use of reliable agents. Additionally, in partially observable non-deterministic environments such as the environment of network security, maintaining knowledge of the global state can be increasingly costly and produce inaccurate results which further favours the individual local knowledge-based approach.

There is a clear distinction between a single ABS [80, 166, 129] and a MAS. The agent in a single ABS often resembles a centralised system with intelligent capabilities while the MAS is distributed in design and will have multiple agents that exhibit intelligent and cooperative behaviours. Within the single ABS communication or cooperative behaviours are not required, however, they are core concepts for use within a MAS. The benefits of MAS over a single ABS are that multiple agents can perform more complex tasks such as distributed actions impossible to complete with a single agent. Through the use of concurrent actions, distributed processing and by monitoring the wider environment, MAS have proven effective solutions to large-scale problems. Throughout the literature, there have been many mechanisms proposed to aid in the guidance of the MAS.

Zivan et al. [174] use a distributed team of agents that use local search to solve navigational problems within the environment. The fundamental problem addressed in this work is the positioning of agents to monitor points of interest, often requiring multiple cooperative agents to triangulate the position of a target. Zivan notes that optimally selecting the position of agents is an NP-hard optimisation problem. The particular approach makes use of two agent

sub-teams: a surveillance team and a search-and-detection team that use different algorithms for achieving their respective goals. The surveillance team's primary responsibility is to monitor a target once found while the search-and-detection team uses algorithmic-search to explore the environment to find new targets and communicates the locations to the other team. Agents use past actions to tune the expectation for future exploration and optimise the use of high performing agents. Critically, this MAS did not take advantage of domain knowledge that might be used to increase the performance of the search using knowledge of the target or the environment, for example, facts about the likely locations or historical data could be incorporated to increase the performance.

As an example of a MAS used in a live environment, Alkhateeb et al. [8] designed a wireless system to monitor university campuses through the use of multiple interacting agents collecting a wide range of data from abnormal sound to pollutants in the air. The system used multiple agents to monitor several systems around the campus to provide a context-aware system to reduce the requirements for human security guards to patrol the campus buildings. Sensors were deployed in critical rooms around the university and could respond to perceived incidents by alerting the relevant member of staff, for example sending a text message to a security guard with a map of the building and instructions to investigate when a threat was registered. The system is composed of two categories of agents: system and ACCESS (Agents Channeling (or Conveying) ContExt Sensitive Services) agents. The system agents are responsible for managing the platform by creating and deleting agents as they are needed while the ACCESS agents have generic responsibilities for monitoring services that are used within the architecture (e.g., location services).

Baig et al. [17] performed a survey of the current application of MAS in many critical infrastructure areas including intrusion detection. Emphasis was placed on system resilience so that if the system (or agent) is attacked, the rest of the system should not go off-line. Network topologies were considered and noted to have many entry points (e.g., Ethernet, wireless, Bluetooth), and so the need for technology-specific agents at different points is needed. In traditional MAS domains the environment is often static and does not experience much change. However, in the case of computer networks, there will be a vast change in environment, possibly on a daily basis, as devices are switched on and off or when new devices are installed for the first time. It may, therefore, be beneficial to consider a more adaptive MAS that takes environmental change into consideration and even use it as a way to monitor the system. Agents specifically designed to consider the environmental changes could be used as a way to defend against rogue hardware within a network by investigating new hosts and analysing (through the use of machine learning)

how a network changes shape over time so that it can learn what to expect as change occurs.

2.3.3 Agent Norms

Norms are a popular mechanism for guiding MAS in their operations. Elster et al. [51] define several uses for norms, in particular, the behavioural serves to compel an entity into acting in a particular way. Uszok et al. [155] discuss the use of norms to constrain agents so that they continue to operate within the established bounds. Alechina et al. [7] consider the use of sanctions in norm-oriented systems, in particular, regimentation sanctions to remove the use of past actions from future use and resource sanctions which impose a tax on agents that violate norms. In norm-constrained environments, agents are given the choice to conform to the norm or violate it which is appropriate in some but not all circumstances.

Goldman et al. [65] considered a variation of agent cooperation that, instead of directly working with each other, would make positive changes to the environment to aid the other agents in their goals. In the proposed system, agents are encouraged to use their own resources (processing time) to make the environment better for all of the agents, in the example "Tile-world"[4], where the goal of the agents is to move around the space pushing blocks into holes, the agents can move tiles towards the holes even if it is not the goal of the agent to push it in, this type of cooperation results in other agents having to do less work to complete their goals.

2.3.4 Agent Communication & Efficiency

One way of enabling agents to work together is through the use of coalitions [166] that bring mutually useful agents together with the intention to complete some plan. This is usually achieved through the process of negotiation [54] and auctioning [166] to form the optimal team. There are many types of coalition structures available for use with varying degrees of democracy and authoritarianism. A purely democratic coalition will have each agent considered equal and allow agents to submit participation requests [166] and eventually a coalition will be formed, however, this can prove to be inefficient in a temporally-bound environment where coalitions and decisions must be made promptly. By implementing a more authoritarian model that sees one agent given more importance than others, has the power to veto decisions and choose the team that it wants, decisions can be made quickly but often less optimally.

For any agent communication, a model that defines how messages should be exchanged is required. Gmytrasiewicz et al. [64] considers communication between agents that do not share a common language. The current standard is for language protocols to be hard coded by

[4] A 2D environment that agents must navigate to avoid entities and reach the goal tile.

the agent designer and to have a fixed syntax for communication [163]. However, this limits the autonomy and capabilities of the agents in future unforeseen situations that may require an expanded syntax. As a strategy, Gmytrasiewicz proposes a system of inheritance whereby agents could share their local syntax with the group to form a globally shared syntax for multi-agent communication. Wang et al. [163] further explained the need for such a dynamic language that does not limit agents so that should the agent come across a piece of information or concept that it was not pre-programmed to understand, communication could take place without extensive human interaction.

Negotiation [166] is the process by which agents may agree on some issue. When agents work together a common way to discuss the terms of the mutual deal is by suggesting improvements to the deal. This process is often used to work out logical problems with a deal, for example, [166], if an agent needs assistance in performing some task, it may request the help of another agent. However, the agent may not be able to stop what it is doing for a given period. Given this fact, the initial agent may cancel the assistance request as it needs immediate help with some task. Used in this way, agents can negotiate complex deals and efficiently coordinate their efforts, and avoid concurrency issues such as race conditions or deadlock.

Jennings et al. [79] describes the factors required for negotiation to take place, including, a set of rules that govern the agent's interaction, a protocol for defining how the negotiation should take place an object to be negotiated over. Jennings describes negotiation as a "search through a space of potential agreements" where overlapping spaces represent mutually beneficial agreement that can take place. As an improvement over the traditional proposal-acceptance/rejection model, an improved counter-proposal model was advocated whereby agents could submit critiques of proposals to improve it iteratively. In non-competitive environments where agents work together to achieve a goal, this model allows for a compromise between local agent goals and the global aim of the system (e.g., to protect a systems security).

Auctioning is the act of agents bidding for a service or place in some coalition. Agents in a utility-based model will use the desire for a positive utility score to bid for services when working with other agents [166]. In a scenario where two agents want to join a coalition, and they are both equally suited to the aid in the goals of the coalition, the utility will be used to decide which agent gets the place. There are numerous auction models [166] that can be used to structure the bidding process, each with their distinct advantages and disadvantages. Weerdt et al. [45] use auctioning in a contractor-based model where jobs are outsourced to other agents after a bidding process where jobs were taken based on the efficiency at which they could be completed. The model provided an effective way to find the most efficient agent for the job,

however, under conditions where the most efficient agent was busy, a less effective solution would be found. Furthermore, with auctioning systems, communication overhead increases the overall cost.

Fatima et al. [55] considered how agents should form coalitions [11] with each other and how the negotiation should take place. Two main power structures between the agents were considered: a democracy that gave the agents more weight to vote with and an autocracy that gave most of the voting authority to one particular agent. The conclusion of this theoretical study was that having a pure democracy results in an inefficient welfare distribution and that some level of authority benefits the creation of coalitions. As observed within real-world power structures, in any situation where the environment is time constrained, and a system is under attack (i.e., the military), a top-down command structure is used to improve the efficiency and response time of the organisation. It is commonly accepted to be more efficient to have a power structure with some authority that can veto actions or set plans in motion. Given a situation where a system is under attack, it could be necessary for one agent to force other agents to cooperate in some distributed goal to ensure the security of the system, even if there is no immediate benefit to the agent that is drafted.

Bahrani et al. [16] developed a coalition planning architecture for environments where the environment and information are continually changing. The architecture uses both information and collaborative technologies to verify pieces of data before acting upon them. The architecture is designed for use within military organisations with personnel carrying out some actions; it was found that it is more advantageous to have teams develop plans independent of each other and then bring the plan together at the end when sensitive information was being used, this helps control the flow of information through an organisation. Another emerging requirement was that plans should be able to be constructed using both partial information and full information with a view to enabling plan development in both information-rich and abstract scenarios. Feedback from the use of this system found that future systems should support existing processes and procedures rather than forcing new processes or change existing processes. Also included within the architecture was a meta-level plan analysis used to scrutinise aspects of the plan continuously, used when plans are merged.

2.3.5 Agent Planning

Agent planning is the concept of agents sensing the environment and selecting a series of actions to perform to achieve some goal. Plan making can be performed by a single agent using its internal knowledge to make judgements or by a group of MAS agents that are capable of

communicating with each other, considering multiple plans and sharing their internal states to develop a better plan.

The activity of agent planning is deeply rooted in philosophy and can be expressed using logical notation to formalise the concepts for use in digital systems.

Planning makes use of agent autonomy to sense the environment and select a series of actions to achieve a goal [31]. Planning may be performed by a single agent based on internal knowledge of by a MAS capable of communicating with each other to form more complex plans. Agre et al. [4] consider the computational cost of planning which increases exponentially with the number of possible outcomes. Extensive planning requires an extended amount of time to consider all options and in some dynamic situations can prove untenable where the utility of plans may expire before optimal solutions can be reached. Many planning mechanisms tend to function linearly with information being used iteratively rather than dynamically with little recalculation being done on past beliefs in the light of new evidence [3, 8, 70, 122]. This design choice constrains the MAS making it functionally less responsive to new data as it becomes available but reduces the amount of reprocessing that must be done in the future.

It is argued [30, 38, 12] that it is only logical for an agent to adopt a plan if the agent believes that the plan is achievable. In the security, context plans can be divided into two categories: plans to enforce a policy (i.e., blocking an IP address) and plans to gather additional information about an adversary (i.e., performing a DNS lookup). The concept of believing that a plan is achievable can also be broken down into two categories: belief that the plan can be attempted (i.e., that the required internal/external infrastructure exists) and that the plan will produce the desired result (i.e., that the result of some action will produce some useful information). When performing a policy enforcement plan, it is necessary that the infrastructure exists to allow this to happen, and that the outcome of the plan to result in the intended action. The belief that the plan can be achieved and the belief in the outcome of the result is absolutely known in both cases. However, in the second case where the agent is attempting to learn more information about the adversary, it is still necessary that the required infrastructure should be readily available, but it is not necessary that the outcome of the plan be known ahead of time. In a scenario where there is no clear plan available for learning about an adversary, it becomes necessary for agents to adopt exploratory plans for which there is no clear postcondition. The world in this scenario can be viewed as a disconnected tree-structure (figure 2.8) with groupings of nodes representing sources of information (services) to be collected about an adversary. There may be several disconnected structures (gaps within the knowledge space where information has been deleted by the adversary) within the powerset of all worlds due to the malicious actors

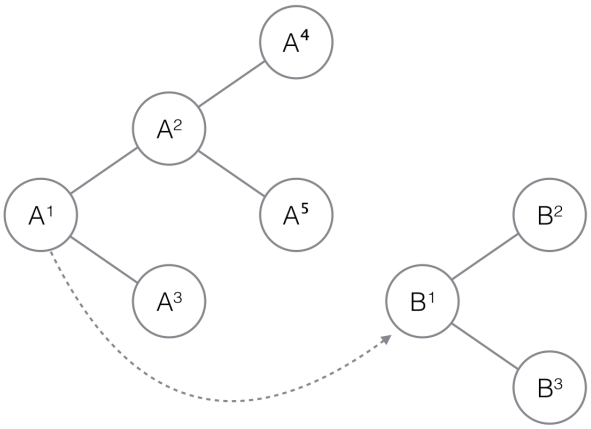

Figure 2.8
A disconnected world structure requiring exploratory plans to move from one tree to another.

covering their tracks by wiping logs or using evasive technologies resulting in knowledge gaps.

For exploratory goals to work the infrastructure for the goal must exist and the agent must be able to interact with it. The only precondition for exploratory planning is that the agent can interact with the systems in the plan, it would, however, be beneficial to order the plans using an estimated utility reward based on past successful plans.

Padgham et al. [124] introduced the notion of *capability* into the BDI model [30] allowing agents to rationally think about planning. Many systems are implemented by using a plan library that consists of actions that can be used within the plans as well as a list of triggering events for each associated plan and any pre-conditions [124]. The notion of capability means that an agent has at least one plan and trigger for some goal. In Multi-Agent environments where agents can work together the list of capabilities available to an agent can extend to capabilities offered by other agents. The current way of doing this is to have agents communicate to tell each other what capabilities they have to offer. Instead of having one agent propose the entire plan, having one agent announce a general goal and then have other agents work towards that goal without unnecessary communication could prove more efficient and more in-line with the distributed nature of MASs.

Agre et al. [4] considered how agents could more efficiently plan [31] their actions, it is noted that planning is often a computationally expensive activity due to the often exponential number of outcomes that must be considered for an optimal plan to be devised. To perform an extensive plan would require an extended period to consider all of the options. In highly dynamic situations such as that of an IDS, plans are needed rapidly to respond to threats so cannot be deliberated on for too long. Agre's agents use indexical functions to describe actions to be taken and possible consequences of those actions. The benefit of describing actions in this

was is that only a relatively small part of the environment needs to be considered, instead of making a plan for the whole environment, the agent just considers the immediate environment that it is in and the affected environment where the result of the action occurs. This highlights the requirement for an agent that can perform actions and consider the results of the actions while doing it efficiently without considering the state of the entire environment.

For an agent to be able to plan, they must be able to choose what actions to perform based on what they believe is happening in their environment. Vargas-Vera et al. [156], while considering the problem of semantic web ontology mapping [164], address the issue of conflicting beliefs. Independent agents might look at two related pieces of evidence and come to very different conclusions from the evidence. Within the application of security, where adversaries[5] can manipulate data sources, it is possible that the adversary could tamper with data leading the agents to come to different beliefs about him. Vargas-Vera uses a fuzzy logic system to assign either a low, medium or high trust level to any given belief. This would then be factored into any situation where a conflict arose.

Many current planning mechanisms within the literature are linear in design [3, 8, 70, 122] with the flow of information only going one way and not being used iteratively to plan more dynamically. Generally, an agent has some intention, which is broken down into several data collection tasks that the sensors perform, and then the data is analysed, and the cycle begins again. It would be more beneficial, especially in the cybersecurity context, if the agent were able to recalculate their intentions as more information is collected. Security data is often located over a wide geographical area (the local network and the Internet), so when plans are formed, they may be computationally intensive. If plans are viewed as tree structures of subtasks to be completed, given the results of some collecting task, a path on that tree may become unnecessary or redundant. Having a mechanism to reconfigure the collection tasks based on the collected information could improve the speed at which security goals are achieved.

Korf et al. [95] develop a planning algorithm to solve the Rubik's Cube problem[6] using an IDA* (Iterative Deepening A* Search) using a lower-bound heuristic built from a lookup table containing the number of moves it would take to reach various sub-goals. IDA* uses depth-first search to find increasingly longer solutions through the search space while using the lower-bound heuristic to reduce the search space for any paths that are less optimal than the currently held optimal solutions. To analyse the algorithm performance, random sampling was used, and then the search heuristic for each stage was computed, the results were then compared to find the

[5] An adversary is taken to mean a malicious entity.

[6] The classic Rubik's Cube game with coloured squares that must be reordered so that all cubes of the same colour are moved to one side.

error between the two calculations to determine the effectiveness. While some similarities can be drawn to the problem faced in cyber attribution problem, there are some subtle differences. If the attribution search space can be abstracted to a weighted graph search problem with nodes representing sources of information about an adversary, before the search starts, the weights (i.e., rewards) of the nodes are unknown, and to find out what the rewards are, an agent must interact with the node, which incurs a computational and temporal cost. Given the example of a node containing a DNS server that could be interrogated, if the agent does not interrogate the node, it may miss out on some information about the adversary, or if the agents do interrogate the node they may or may not learn something about the attacker with a computational and temporal cost associated with either result. The problem for the agent then becomes to intelligently decide whether it is optimal to search that node or not, as is the case in Korf's algorithm, a table of sub-goals (i.e., a series of lucrative nodes to visit in the cyber attribution case) that the agent should aspire to achieve could be kept as a way to guide agent search.

The PDDL (Planning Domain Definition Language) [110] can be used to express planning problems by defining the problem in two components: the domain[7] (predicates, types and actions) moreover, the problem[8] (objects, goals and states). It was proposed as a standardised language for planning so that programs for different environments could be created similarly without the complete rewriting of a new planning language or system. The language supports the following features: (1) Basic STanford Research Institute Problem Solver style actions, (2) Conditional effects, (3) Universal quantification over dynamic universes, (4) Domain axioms over stratified theories, (5) Specification of safety constraints, (6) Specification of hierarchical actions composed of subactions and subgoals, (7) Management of multiple problems in multiple domains.

2.3.6 Control Knowledge

Perez et al. [130] used control knowledge (i.e., information about the environment or specific problem) to improve the quality of plans. Focusing specifically on the automatic acquiring of knowledge, the algorithm compares the currently best held plan (based on the current control knowledge) with future plans (based on future control knowledge), when the algorithm is searching for new solutions it is allowed to search further for better plans, for example, if the currently held best plan was located at depth 8, the algorithm might search for better solutions up to depth 14. Looking for the goal state in this way is not guaranteed to find the optimal solution as it is similar to performing a DFS (Depth-First Search) with an arbitrary cut-off depth, but

[7]The domain does not change with different scenarios.
[8]The problem can change PDDL is designed to be a domain-independent language.

it will find increasingly optimal solutions as the algorithm is allowed to search further into the space based on the control information.

2.3.7 Distributed Planning

Ephrati et al. [53] proposed a Multi-Agent planning system whereby goals are broken down into sub-plans and executed locally by the individual agents. It was noted that the complexity of planning can be measured by the time and space consumed resulting in an $O(b^d)$ where b is the branching factor (i.e., average number of newly generated states at each stage) and d is the depth of the problem (i.e., the path from initial state to the goal). This was shown to be significantly reduced through the use of Multi-Agent planning techniques that divide the goals up into sub-plans. In the proposed system the global plan was constructed out of local plans that were completed by the individual agents based on their local knowledge of the environment. While the use of sub-plans can result in a far more efficient algorithm, this is rarely the case as the sub-plans are rarely independent of each other and can result in the duplication of actions or agents competing for resources (i.e., race conditions and deadlock). In the security environment, the fact that agent actions are not independent could serve to benefit the system as when multiple lines of investigation converge on one goal state it makes it probable that the information is trustworthy. However, for some resource intensive collection tasks, it would be inefficient to duplicate collection efforts when it is likely that the same information will be returned in both instances. The issue of multiple agents attempting to perform the same task could be solved through the implementation of very specific agents that can perform only one collection task each. In this way it forces the agents to cooperate to perform any extended plan and where one agent is responsible for the collection of one piece of data, that agent can keep track of how many other agents have requested the same piece of information and simply hand out the result instead of redoing the collection task each time[9]. It must also be considered that finding a goal state for some problem may not be the optimal goal state and so decide how long the agents should continue to work at the problem even after a goal has been reached must be considered.

Megherbi et al. [114] used reinforcement learning to perform autonomous path planning. The ability to monitor an extended environment is a task particularly suited to distributed MAS. Within the architecture, Megherbi described a method to enable agents to remember the utility of traversing a given path over a number of iterations by remembering which node that they had traversed in the past and associating a utility number with it. This way if the agent, during a

[9]The agent should consider the optimal refresh rate for the resource that it is responsible for collecting and then redo the collection task when it is optimal to do so. For example, performing an Internet Control Message Protocol ping against a target should have a shorter cool-down period between refresh in comparison to a DNS lookup

random walk, found itself traversing a previously taken path they would have some knowledge about the utility of following the path. The problem with applying this concept to the security environment is that it requires many iterations to find an optimal path when a security event happens, it will likely be a one-time event. Assuming that the agents perform one task and are responsible for all similar tasks, over time, if they store the outcomes of various actions, they would accumulate enough data to be able to judge the utility of one result against all previously seen results. Giving the agents the ability to assign utility based on the outcome of previous actions solves the problem of having to hard-code the utility values and allows the agents to use their local knowledge to direct future plan-making. Within Megherbi's proposed system the agents keep a log of all the visited states during a run but in the security context, the time frame for a "run" is open-ended. The agents responsible for making the global and sub-plans should keep a record of how well the plans performed but not necessarily store the data associated with the plan. If each of the agents responsible for performing information gathering stores the data permanently, the agents that make the plan should only need to store the plan that was used as well as the utility reward associated with the plan. Over time as new plans are attempted the agents could then choose from the plans with the highest utility reward rather than having to perform random walks.

Ferrando et al. [125] designed a context-aware intelligent MAS for the creation of plans [44] within the healthcare industry. Argumentation was chosen as the tool to be used for reasoning about events because of its perceived benefits [125] when working with incomplete and inconsistent contextual information. Three approaches to plan making were highlighted: plan selection is where agents construct independent plans and a centralised algorithm selects the best, plan merging where agents construct independent plans for different sub-goals and a centralised algorithm merges the plans and plan construction is where agents iteratively refine a base plan until a joint plan that solves the problem is developed. The proposed solution takes in the input suggestions of multiple agents and attempts to plan and evaluate the effectiveness of the solution. However, this solution does not take into account environments that are mismatched with the internal state of the agent. In effect, the plan is first developed inside of the agent's internal space and then acted out in the environment. This model of plan-then-perform is useful for very predictable environments but may be less effective within the security environment where plans are more likely to fail. An iterative approach is required where small portions of the plan are created and then performed instead of developing an elaborate plan where the failure of one component may cause the failure of many components.

2.3.8 Uncertain Environments

Uncertain environments is the term used to describe those environments that cannot be fully monitored by the agent, are prone to sporadic state change and as a result require dynamic approaches to adapt to environmental changes.

Wu et al. [167] proposed an algorithm for Multi-Agent planning in uncertain environments using online algorithms that only plan one step in advance given all of the currently available information. The proposed algorithm attempts to overcome the challenges of online planning (i.e., no training data) by using coordinated communication between the agents. A Markov Decision Process along with Bayesian probability is used to calculate the likelihood of events occurring given the information discovered at each step of the online discovery process. The concept of agent histories is also taken into consideration during the planning process to speculate how agents will act in a situation given their previous actions. The communication for the proposed solution ensures that each agent has a synchronised copy of the histories for each agent so that coordination can take place even when the communication channels are not completely reliable. While the concept of a history, especially attempting to use the historic actions of an attacker, is useful in some situations, it may lead to agents not collecting information when an attacker makes a mistake. Given the example where 90% of attackers use a proxy to mask their source IP address, if the agents rely on the histories of the attackers, they may choose not to attempt a traceroute as such actions have proven unproductive in the past, if this happened for 10% of the attackers who did not use a proxy, the information would be not be gathered thereby putting the agent at a disadvantage. The concept that one action may not result in a consistent result must be taken into account when operating in such an unreliable environment. While past successful actions should be taken into account, the agents should remain open to the possibility that previously unsuccessful actions could prove worthwhile performing; this should perhaps be implemented as a 'last resort' series of actions in the scenario where all normally-reliable actions fail.

Chapman et al. [35] abstracted the problem of attacker attribution down to a search game of hide-and-seek, considering the various proxy servers and anonymising technologies as nodes in a graph to be visited by an attacker. While attackers will usually choose their positions strategically to remain hidden, it is noted that often attackers are not completely rational and may exhibit behavioural preferences in their chosen technologies. The overall goal is to understand how to best discern a path back to the attacker. One of the most important factors in being able to follow the attacker through the search space is having enough information to be able to draw accurate conclusions. It, therefore, makes sense to enlist the use of MASs to autonomously

collect this data from numerous sources around the web, a task which would take a human operator a long time to complete.

2.3.9 Plan Complexity

Cox et al. [41] discussed different types of Multi-Agent planning systems, it was noted that in some works the planning process is the Multi-Agent component with several specialised planning agents that work together to generate a plan that they could not develop alone, and in other works the product of the planning is Multi-Agent in the sense that it gives activities to multiple actors to perform. Cox focuses on a specialised class of planning problems, called MPCP (Multi-Agent Plan Coordination Problem), in which multiple agents within a system plan their activities but can benefit from mutual coordination and cooperation in some tasks. One such advantage of MPCPs are that agents can communicate to avoid the duplication of work while avoiding extensive computationally expensive planning. The notion of agent coupling is used to describe different degrees of cooperation. Loosely-coupled agents are mostly independent of each other with some interaction, while highly-coupled agents work together and communicate regularly. The algorithm developed by Cox looks for inconsistencies in plans and resolving them to find the optimal solutions. The search uses a branch-and-bound algorithm that checks the search space for the most optimal solution at any given point, as the first solution found will be classed as the optimal one, the search must continue for some specified amount of time as the agents could find a better solution. Pruning is used to discard any solutions that are worse than the currently held "optimal solution" to reduce the size of the search space.

Korf [94] viewed planning as a problem-solving search using sub-goals, macro-operators (a sequence of primitive actions to be taken to satisfy some sub-goal) and abstraction to execute more efficient searches for the goal state. The problem space was modelled as a set of states and a set of operators (i.e., actions) that map between multiple states. The case where no knowledge is available was first considered, this will take the form of a brute-force search which can be inefficient, BFS (Breadth-First Search) and DFS are the most common examples of a brute-force search, BFS uses $\mathcal{O}(b^{\mathrm{d}})$ time and space while DFS uses linear time and requires a cut-off depth to stop the program from running indeterminately. An improvement on the traditional DFS is the Depth-First Iterative Deepening search that starts with a cut-off of 1 and increases it until the goal state is found, this works in $\mathcal{O}(b^{\mathrm{d}})$ time and $\mathcal{O}(d)$ space, eventually all brute for algorithms will find the optimal solution. Sub-goals are noted to be an efficient way to split exponential problems up as decomposing an exponential problem in multiple simple problems divides the exponent, significantly reducing the overall amount of work. Using sub-goals can be

used in planning/search problems where there is something to be learnt about the environment, it can be used to guide the search and prune branches of the planning tree. The final goal state can be reached by concatenating together optimal solutions to individual sub-goals. Korf notes that agents must go further than simple search to solve planning problems efficiently, they must include learning behaviours such that they can learn a general strategy for solving a problem over time, similar to how humans solve problems. Finally, Korf states that abstraction should be used to, at the start, ignore low-level details and focus on a general plan for solving the problem and then fill in the details as the planner progresses.

Cox et al. [42] discuss how, in a Multi-Agent environment, efficiency gains can be made by giving agents the opportunity to find overlapping and duplicate tasks so that tasks can be merged thereby reducing the overall computational cost. It is noted that plan merging can incur unwanted costs, for example, it decreases agent autonomy, making them depending on a network of agents or agents may suffer delays if the task could have been performed quicker locally. Cox implemented the plan merging system as an independent agent capable of finding and implementing plans after an agent has requested a merger. While this solution has proved to be useful, a more carefully decomposed system could avoid the problem of plan merging altogether. In a system with sufficiently low-level agents, each having a particular task that they can perform, when an agent needs to task performing, rather than performing it locally, they would instead request that the specialised agent perform it. Now, when the opportunity for a plan to merge occurs, the specialised agent responsible for the given task which has been duplicated by two agents can internally merge the two tasks into one without any communication or negotiation. While this will have the effect of significantly increasing agent autonomy, it will have the effect of more efficiently dealing with tasks and keep all of the data about one particular task in one place for analysis.

2.3.10 Non-Classical Planning Environments

One method of representing agent plans is through the use of DEC-POMDPs (Decentralised Partially Observable Markov Decision Processes) [23], the goal of which is to find an optimal node to travel in an attempt to maximise the future rewards. MDP (Markov Decision Process) are used most often in planning to represent planning systems with multiple changing states. Due to the properties of non-classical planning environments being inherently non-deterministic and to an extent uncertain, MDP's can be used to model the state changes by updating the values associated with the MDP nodes as changes occur. The DEC-POMDPs model uses multiple distributed agents each with their own observations of the local environment and is defined as a

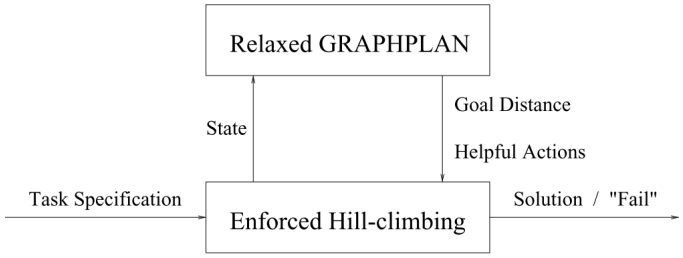

Figure 2.9
The Fast Fordward (FF) planning algorithm architecture.

9-item tuple [168]: $\langle$I, S, $\{A_i\}$, $\{\Omega_i\}$, P, O, R, b^0, $\gamma\rangle$ where I is a collection of n agents ($i \in I$), S is a finite set of states ($s \in S$), A_i is a set of actions for $agent_i$, γ_i is a set of observations, P is the transition function (P: S $\times A\times$ S $\rightarrow$ [0,1]), O is the observation function (O: S $\times$ A $\times$ S $\rightarrow$ [0,1]), R is the reward function (R: S $\times$ A $\rightarrow \Re$), b^0 is the initial state distribution ($b^0 \in \Delta(S)$) and finally γ is the discount factor. Seuken et al. [139] proposed a memory-bounded solution for solving DEC-POMDPs efficiently using heuristics in problems with a finite number of horizons. With every iteration, unnecessary strategies were eliminated to reduce the number of policies needed to be kept in memory. The algorithms used both top-down and bottom-up search and belief points[10] to converge on the optimal policy from two fronts. The algorithm used heuristics, including a random selector, to avoid getting stuck in an optimal local solution. In the security search space, this can be applied but with even more accuracy. When searching for information (represented as nodes) at some nodes, some information will be returned; this can then be used to increase the belief probability of the current plan of investigation and streamline the search for more information. In effect, small pieces of information hidden around the search space can be seen as approximations guiding the agents towards the goal (e.g., attacker attribution). Using these belief points the number of possible worlds that the agents would have to visit can be reduced, making the search more efficient, for example, once an agent finds sufficient evidence that the attacker is using a proxy, the search can be focused around collecting proxy service information.

FFP (Fast Forward Planning) [72, 82] (Figure 2.9) uses heuristic guided algorithms [62, 29] that uses BFS starting from the global optima to increase the search performance. FFP is similar to Heuristic Search Planning [29, 27] but through the use of three properties was able to perform better in the Artificial Intelligence Planning and Scheduling System competition [15]. Firstly relaxed plan extraction was used to ignore any literals that do not lead to the goal

[10] A probability distribution over possible future world states is made whenever the world cannot be observed due to lack of information.

state, this can only be done when the final goal state is known, and the search can take place backwards. An Enhanced Hill-Climbing algorithm [39] was used which works backwards from the global optimum to find a better successor using BFS. The third improvement is the selection of helpful actions during the search. Instead of just considering the weights of traversing edges, the algorithm looks for traversals that will add at least one goal to the plan list. Actions that lead to dead ends can then be avoided as they will not add a new goal to the plan list. It is noted that in some cases this is too restrictive, so when no goals are available, the algorithm reverts to an A* search using the edge weights.

SHOP (Simple Hierarchical Ordered Planner) [119] is a HTN (Hierarchical Task Network) planner that performs problem reduction by decomposing complex plans into simpler *primitive plans* that can be completed by an agent. HTN plans goals in the same order that they will be executed which has the effect of reducing problems at execution time making the effects on the environment more predictable. The issue with the SHOP algorithm is the requirement that it must know the complete world state at every time step for it to function optimally, this condition is often unrealistic when applied to live environments with changing landscapes and uncertainty.

2.4 Summary

This chapter has provided a summary of related work and existing approaches for the areas of intrusion detection, machine learning and MASs. It has reviewed the many approaches to detecting malicious behaviours and discussed some of the drawbacks with the existing systems. In particular, the related work focused on the detection of threats at the network connection and network flow layer where many different types of attacks leave detectable footprints.

During the review of intrusion detection techniques, APTs have been identified as an important issues that require more adaptive solutions to address. Approaches that make use of machine learning techniques are described, however, due to the limitations on the scalability and quality of the data available for training are shown to be inadequate for detected advanced stealthy attacks. Finally, MAS are reviewed as a possible solution to addressing the identified vulnerabilities.

Chapter 3

Decentralised Multi-Agent Security System (DMASS)

In this chapter, an overview of the proposed MAS for performing network intrusion detection and modelling cyber events, first published in [86], is presented. In particular, the proposed model is formalised, a case study is provided and algorithms for use within the system are detailed. This chapter describes a novel approach to detecting stealthy attacks from within the local network. The proposed system, Decentralised Multi-Agent Security System (DMASS), performs agent-based information gathering and analysis for real-time usage without the need for any central control system. The approach looks to proactively collect information and search further locations that have, in the past, been found to be a reliable source of information, given the nature of the event detected.

So far, this thesis has discussed in detail the current application of MAS and machine learning for intrusion detection. It is evident that from Chapter 2 that there is a need for a more autonomous approach to cybersecurity that goes beyond the paradigm of recording intrusion signatures to detect future instances of the same attack. With the widespread use of APTs, an approach less dependant on current knowledge of attacks, that instead focuses on real-time investigation and decision making is required to address current cyber threats. The literature review of current works in this area have shown that no complete system currently exists to address the APT problem. Several MAS were discussed in Chapter 2; however, many of the solutions are structurally similar to central systems containing two components: an evidence collection component and a central system for aggregating and analysing the data.

Traditional approaches have taken to performing a bulk analysis of networks and information which often leads to an increased number of false positives. The fundamental problem with continuous detection systems at a high operating layer (i.e., on network flows) is the amount

of oversampling, resulting in network traffic being checked against a significant amount of signatures. When many signature checks are made, the likelihood of a problem being found is increased, whether an attack is present or not. Network problems, misconfiguration and changes in protocols can all contribute to the incorrect detection of an attack when vast amounts of traffic are continually monitored. The DMASS addresses the increased number of false positives by attempting to perform a targeted search of the environment, only evoking the use of evidence collection agents when given cause to suspect evidence may be found at a particular location.

The main contributions of this approach compared to traditional IDSs are:

- The system selectively makes use of signatures and current detection algorithms while avoiding bulk processing.
- A new layer of agent-based performance measuring is created in addition to the detection approach. Rather than just relying on the performance of the signature, these extra measures allow agents to consider the context in which information is gathered.
- A knowledge representation framework for considering the likelihood of a detected attack being a false result.
- Algorithms for combining the views of multiple agents to classify the event as a whole using several data points (i.e., multiple pieces of evidence).

3.1 Overview of Decentralised Multi-Agent Security System

The DMASS 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 the manual forensic process, currently conducted by trained practitioners, to examining the security environment pragmatically. Manually performed forensics is an involved process of information gathering and analysis to inform the next actions taken by the practitioner. In comparison to the IDS which makes immediate decisions based on a comparison to a signature or deviation from a baseline, the forensic process would analyse information and then look for more evidence in a location determined by the previously collected information. 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 this 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 [175]. 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. In particular, the approach can be described according to the attributes included in Figure 2.1:

- **System Deployment**: Network-based, wired, distributed.
- **Data Source**: Network-packet based, distributed, agent-based.
- **Timeliness**: Proactive-response, continuously monitored, on-line detection.
- **Detection Methodology**: Signature-based, distributed.

The system functions as a set of structurally similar agents that work together to analyse a security event. Each agent is specialised towards the collection of data and analysis for one particular attack vector, for example, an agent may collect and analyse data about DNS logs or port scan attempts. During a detected security event, multiple agents may perform their collection and analysis tasks together to produce a complete analysis of the event sharing conclusions of their findings when necessary. Figure 3.1 shows a flow diagram representing the agent's architecture with seven main processes detailed below. The system as a whole is comprised of multiple agents, each performing these tasks and sharing data to analyse the event.

Initial General Anomaly Detection To begin an agent-based investigation into a suspected security event, an initial trigger must occur to begin the process on-demand. While performing continuous security monitoring is necessary in modern networks, the DMASS approach is instead intended to run only when given cause to do so. The term "extended data collection task" is used to describe the overall process of using the DMASS to gather and analyse security information on-demand and is the primary purpose of the system. However, the individual agents may also be used in a similar way to current IDSs to continually monitor for malicious activity as defined by signatures or anomalous behaviours. The extended data collection task may be triggered by the detection of a malicious or suspicious event detected by an agent during the course of routine monitoring, or by a general anomaly detection algorithm described in Chapter 4.

Agent Selection If an agent detects a malicious event, it will become the initiating agents for the extended data collection task, however, if an anomaly is detected by an external system, such as a general network monitoring algorithm, then an agent will be selected that most suitable for analysing the particular threat. Once an extended data collection task has begun, agent selection

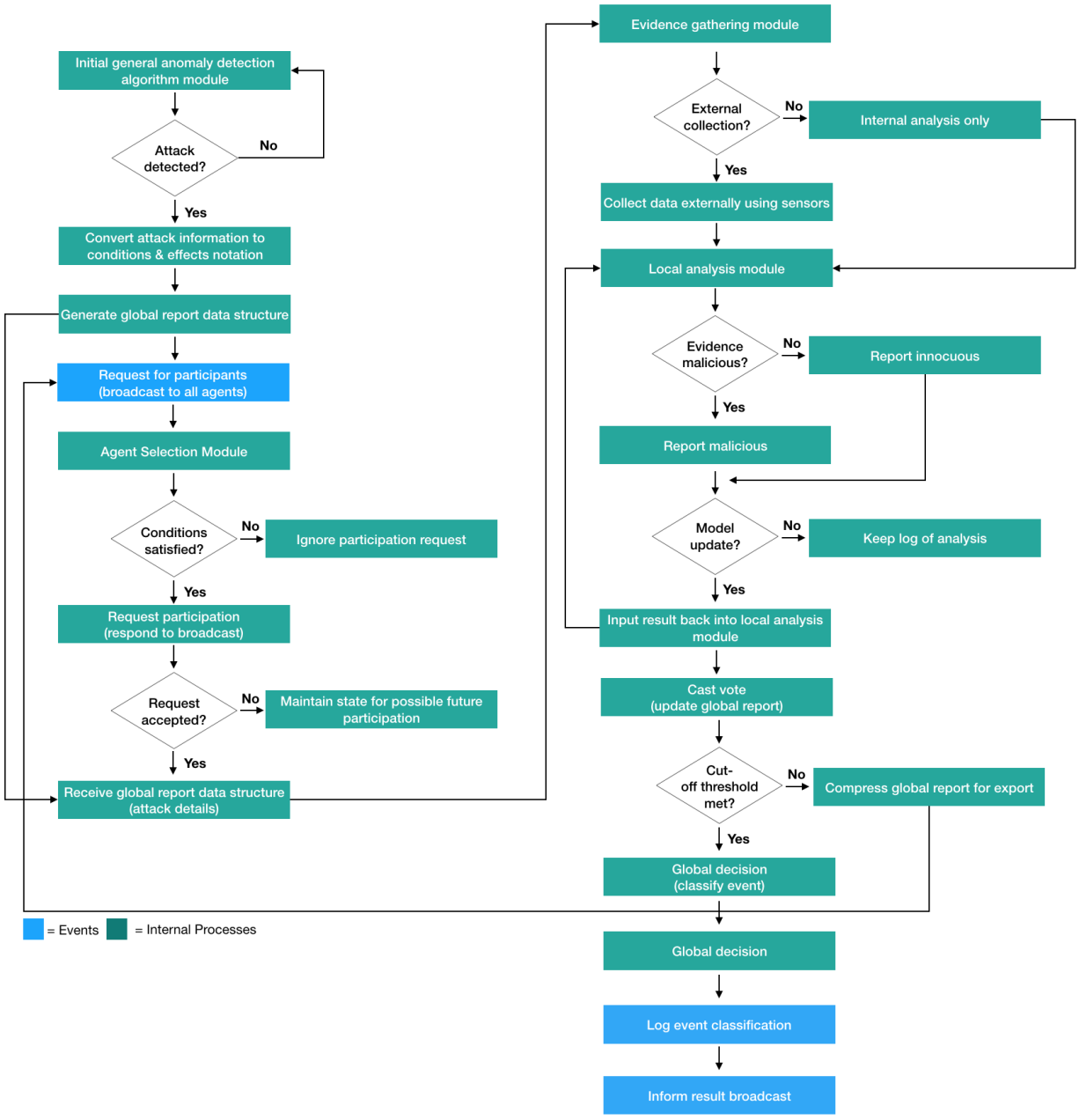

Figure 3.1
DMASS flow diagram.

is handled by the current agent who selects the agent that is most suitable to contribute to the extended data collection task based on the information that has already been gathered. Other factors, including a fitness function, that rank agents based on their prior participation and performance within the extended data collection task are used to determine which agent should participate next.

Evidence Collection Each agent has the ability to collect a particular piece of evidence from the environment, including but not limited to: system logs, security logs, DNS logs, network port information and any data deemed relevant to analysing security events. Agents are limited to one particular evidence type, with a view to creating groups of specialised agents that are experts in analysing one aspect of a potential security event. Bulk collection and analysis of security data is often done by default, however, this leaves a large network footprint with expensive computational costs and does not always produce the intended result of a more secure system. With the complexity of modern networks increasing, the likelihood of detecting a false positive is more likely when performing untargeted scans of the environment, the approach taken by this work avoids these potential issues by waiting until relevant information suggesting the presence of an attacker is found.

Local Analysis Central to the concept of an ABS is the agent's ability to analyse and learn about the environment. Each agent privately and internally maintains information necessary to analyse the evidence gathered in the previous step; this may take the form of signatures, anomaly detection or machine learning techniques. In the case of anomaly detection or machine learning techniques, a local state update may occur following the analysis to update definitions of malicious activity in real time.

Local Decision & Vote Casting The result of the Local Analysis stage is to produce a decision whether the agent believes the evidence gathered is indicative of malicious activity or an innocuous event. Since the whole system is comprised several agents, each with a local analysis of individual aspects relating to the overall security event, the local decisions can be used to judge the performance of the agent compared to all other participating agents, for example, if there is a clear consensus between several agents that an event is malicious but one agent finds it innocuous, the performance rank for the one agent may be downgraded by the others as it fails to come to the group consensus. As previously stated, the local analysis is internal and private to the agent, so only the decision (or vote) and any output information that needs to be analysed by other agents is passed to the next agent. The decisions and output information is

encapsulated within the "Global Report", a data structure containing relevant information about a particular security event that is created by the first agent and transferred between subsequent agents that append information to it.

Global Decision The global decision stage is an optional process that may be performed at the end of the extended data collection task. The exact time at which this process is performed is dependant on the decision algorithm selected. The global decision marks the end of the agent investigation where a final classification for the event as a whole is made; this is done by combining the agent votes using the decision algorithm.

Communication Broadcast To communicate with other agents, a broadcast system is used to find other agents that can contribute to the extended data collection task. Each agent broadcasts any evidence gathered during the evidence collection stage that it cannot analyse locally and takes requests for participation from other agents. Where agents can work with the data type, responses are made to the request for the full global report, at this point agent selection is made based on the past performance of agents as well as the selection algorithm (discussed further in Chapter 3.6). At this point the global report is delivered to the chosen agent the process begins again.

3.2 Formal System Modelling

Formally, the system is composed of several agents[1] $G = \{g_1, ..., g_i\}$, each capable of performing one data collection and analysis task for a particular software service[2]. A set of features F formalises information collected from services, representing information about an activity, e.g., the IP address of a connection, VPN (Virtual Private Network) 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

[1] First detailed in [86].

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

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 the 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.

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 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, the data analysis action is defined 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* $\langle C, e \rangle$*, 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 3.2 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 3.3 shows a similar extended data collection process occurring within the developed 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

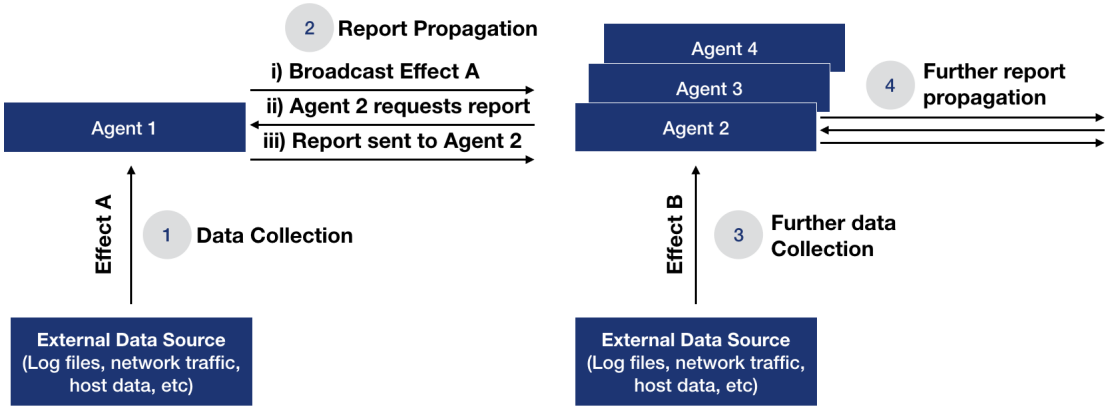

Figure 3.2
Flow diagram for the extended data collection task using agents and data sources.

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;*
- (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

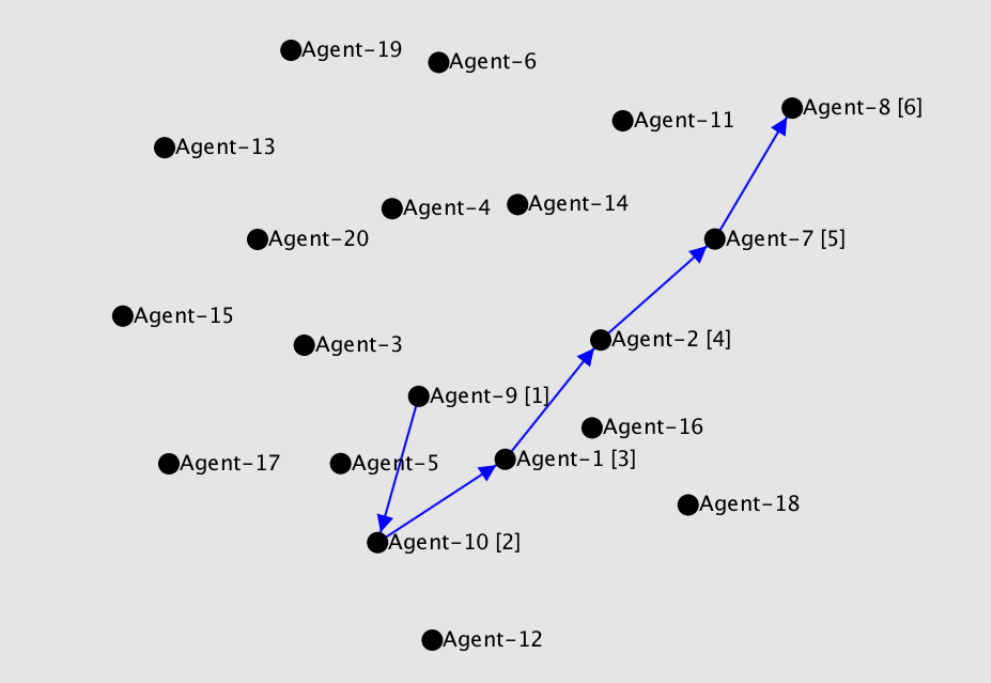

Figure 3.3
A smulated data collection task showing the propagation of information between agents. Agents are selected based on the currently known information about the attack and condition information.

them using one of the voting algorithms discussed in Chapter 3.6. 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 extended 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.*

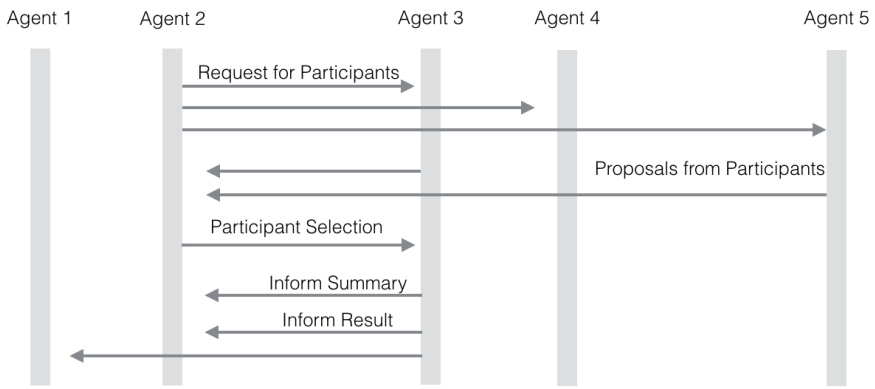

Figure 3.4
An example information flow between 5 agents using the decentralised communications protocol.

3.3 Communications Model

To allow agents participating in an extended data collection to coordinate, this model includes an interaction protocol (depicted in Figure 3.4) [3]. The protocol is formed by five main phases: (i) request for participants; (ii) proposals from available participants; (iii) participant selection; (iv) inform summary; (v) inform result, described as follows.

Request for Participants Once an agent has performed its data collection and analysis tasks, the agent must add its local report to the global report, and then send it to the next agent for further information collection. The communication module is used to facilitate this.

The first step of the interaction protocol is to request help from other agents that can participate in the data collection process. In particular, the set of feature-value pairs are extracted from the global report and then broadcast (by the initiating agent) to the other agents. Given a global report $\{\langle g_1, (f_1, v_1), p_1\rangle, ..., \langle g_n, (f_n, v_n), p_n\rangle\}$ a request for participation is formalised as a set $\{(f_1, v_1), ..., (f_n, v_n)\}$ containing the available information about currently known broadcast out to all agents.

Proposals from Participants Any agents whose data collection action is satisfied by the information contained can respond by indicating their availability to participate in the extended data collection task. Figure 3.4 illustrates this with Agent 2 broadcasting to other agents on the network.

Participant Selection It is possible that several agents respond to the initial request indicating that they can work with the available data. The initiator must decide which agent will

[3] First published in [89]

be selected to continue with the data collection process. In particular, the initiator will send the whole global report to this selected agent.

Unlike in other MAS solutions, the proposed model does not include a central repository of agents which can be queried to find the most suitable agent for a given task. This improves scalability but requires a system to allow agents to find each other. The agents will maintain a local database of agents that they have previously worked with. Deciding which agent should be selected as the preferred agent will affect the performance of the system as a whole. If the most optimal agent is selected for the task most of the time, the search process will improve as less time is spent performing data collection by unreliable agents. There are several ways in which the preferred agent can be identified based on what is important in a given situation. If accuracy is important for the current event the agents may select the agent that most often votes correctly, this will result in a more accurate search. If time is a major factor during an event the agent may choose the fastest performing agent to collect information quickly, this will produce a result faster than the previous but could potentially lead to a less certain decision. While an analysis of factors such as these could be done to determine the optimal preferred agent selection algorithm, events within the security environment can often be unpredictable and allowing the agents to choose the preferred agent at runtime could produce a more adaptable solution.

Inform Summary The agent selected from the previous stage (termed the child agent) will send back a summary to the previous agent (termed the parent agent) containing information about the decision it made during its own data collection task, this is done so the parent agent may evaluate the performance of the child agent by comparing its decision to the groups. This process can be viewed in Figure 3.4 with Agent 3 (the child) that sends back a summary to Agent 2 (the parent). The parent agent will log this summary for use in selecting the child agent in future extended data collection tasks. The parameters sent in the summary will include the agent's local decision about the maliciousness of the event in addition to performance variables such as the time taken to perform the collection task, the importance of the data collected and the computational cost of performing the collection.

Inform Result Once a final decision has been reached, the final decision will be sent to all of the participating agents, this can then be used by the agents to review its method for selecting the preferred agent (e.g., the preference can be increased for those agents with local decisions in-line with the final decision).

Table 3.1
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)

3.3.1 Case Study

To illustrate the advantages of using the proposed model, several agents were deployed in a live networking environment to detect and respond to cyber reconnaissance activities (refer to Table 3.1). Typically there are several stages to a network breach, of which the first is information gathering and reconnaissance [71]. 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 [92]. 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 [136]. The methodology of monitoring a network and making judgements about the 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 3.8). 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 3.1). Using the system of conditions and effects, 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 3.5 shows the communications stack after Agent-1 has broadcast its effect to the network[4].

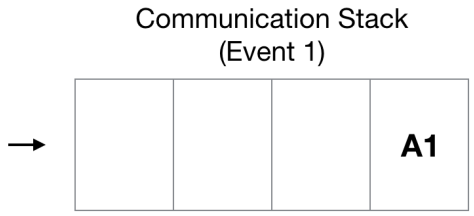

Figure 3.5
The communications stack after event 1 showing Agent-1 (A1) is the only agent to have communicated.

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 over Agent-2 was made based on the reputation of both agents (See Figure 3.6). Agent-4's functions are broader 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.

Event 3 (event classification) In this example with few agents, there are no more responses 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

[4]The agent communications model described here was first introduced in [88].

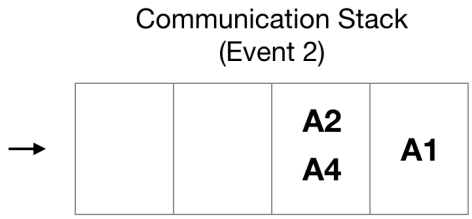

Figure 3.6
The communications stack after event 2 showing Agents 2 and 4 have responded to Agent-1's broadcast.

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.

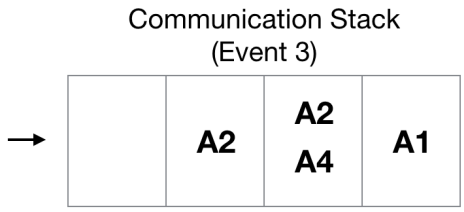

Figure 3.7
The communications stack after event 3 showing Agents 2 being selected to participate in the extended data collection task from its previous request made in event 2.

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, Network Mapper, 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 the DMASS. The strength and novelty of the DMASS system is that 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.

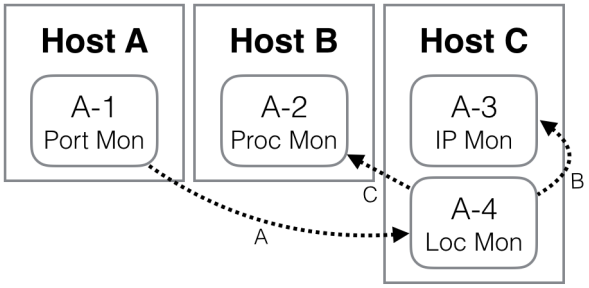

Figure 3.8
Example using agents (A-1,2,3,4) distributed across three devices (Hosts-A,B,C) to detct and investigate the cause of a port scan attempt.

3.4 Environment Model

The concept of domains is introduced to describe the complexities found within the cybersecurity 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 [121]. Figure 3.9 shows this graph with nodes representing agents, the colouring representing groups of agents belonging to the same service, and edges representing

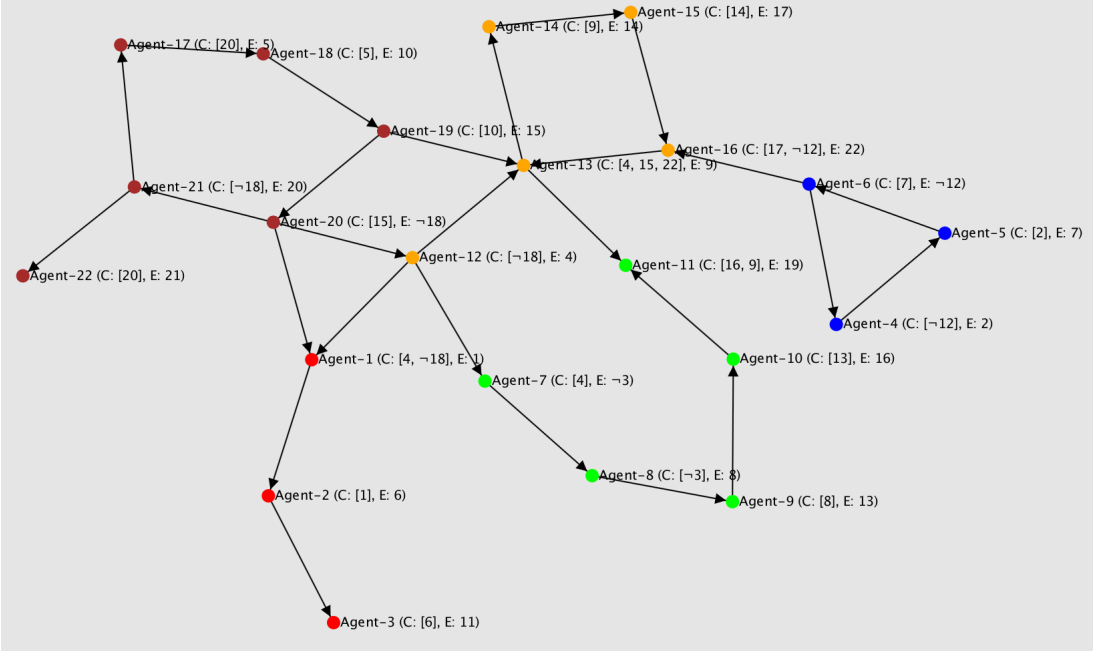

Figure 3.9
The simulated network environment containing several technological domains (represented by colours). Conditions (C) and Effects (E) are numerically represented to abstract the data they represent.

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 [57], 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, the attack is expected 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 [49]. Agents use these predictable and well-defined movement patterns to determine the plausibility of monitored events.

Given the example of a supply chain network, used for facilitating communication between multiple businesses [146], the typical IDS solution would offer limited protection against more complex multi-stage cyber attacks [152]. By participating in the supply chain network, the larger corporation, which may have a more robust cybersecurity solution, puts itself at risk since the attacker is more likely to penetrate through less protected supply networks and pivot to the real target of the attack. The concept of domains can accurately model this threat with the DMASS approach 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 APT which is more likely to see the use of novel zero-day exploits that currently have no matching signature for detection. The 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.

3.5 Simulator Description

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 3.2): (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. The main variables from each category is described 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 3.9). Over time these variables may 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, 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 cybersecurity attacks, for example, a DDoS (Distributed Denial of Service) attack would rank low on the attack penetration but high on attack detectability.

Table 3.2 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.

3.6 Domain Evaluation Algorithms

Using the concept of domains, several algorithms for the analysis of the network environment were developed. 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 domain exploration algorithms is provided in Chapter 5.2.

Table 3.2
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.

Baseline (Refer to Algorithm 3.1) 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 3.2) The first 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

Algorithm 3.1 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.
eid is a unique event identifier.
ts is the events timestamp.
$g \in G$ is the agent's identity.
(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.

Define:

$T_{malicious} \leftarrow 0$ a tally of malicious votes.
$T_{innocuous} \leftarrow 0$ a tally of innocuous votes.

1: **for** $\langle eid, ts, g, (f, v), p\rangle \in R_{Global}$ **do** ▷ Iterate and tally local decisions
2: **if** $p \geq \alpha$ **then**
3: $T_{malicious} \leftarrow (T_{malicious} + 1)$
4: **else**
5: $T_{innocuous} \leftarrow (T_{innocuous} + 1)$
6: **end if**
7: **end for**
8: **if** $T_{malicious} \geq T_{innocuous}$ **then** ▷ Return global decision
9: **return** $\langle eid, decision : malicious\rangle$
10: **else**
11: **return** $\langle eid, decision : innocuous\rangle$
12: **end if**

Algorithm 3.2 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.

1: **for** $\langle eid, ts, \{g, (f, v), p\}\rangle \in R_{Global}$ **do** ▷ Iterate and tally local decisions
2: **if** $p \geq \alpha$ **then**
3: $T_{malicious} \leftarrow \gamma$
4: **if** $\beta = decision : malicious$ **then**
5: **if** $\gamma < 5$ **then**
6: $\gamma \leftarrow (\gamma + 1)$ ▷ Increase series weighting bonus when additional votes are cast
7: **end if**
8: **end if**
9: $\beta \leftarrow decision : malicious$ ▷ Store the last counted decision in β
10: **else**
11: $\gamma \leftarrow 1$ ▷ Reset the counter when the series is broken
12: $T_{innocuous} \leftarrow (T_{innocuous} + \gamma)$
13: $\beta \leftarrow decision : innocuous$
14: **end if**
15: **end for**
16: **if** $T_{malicious} \geq T_{innocuous}$ **then** ▷ Return global decision
17: **return** $\langle eid, decision : malicious\rangle$
18: **else**
19: **return** $\langle eid, decision : innocuous\rangle$
20: **end if**

final event classification. The algorithm uses the corrective measure of giving extra weighting to a 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 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 3.3) 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 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.

Series Weighting with Self-Selected Groups (Refer to Algorithm 3.4) To increase the adaptability of the system, agent preference is introduced in the form of self-selected groups to

[5]Currently a value δ is used as a static value for the number of local reports to be processed. In future work, an adaptive threshold will be developed for this value based on the network size.

Algorithm 3.3 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.

1: **while** $increment(R_{Global}) \leq (\delta * |R_{Global}|)$ **do** ▷ Iterate over the global reports
2: **for** $\langle eid, ts, \{g, (f, v), p\} \rangle \in R_{Global}$ **do** ▷ Iterate and tally local decisions
3: **if** $p \geq \alpha$ **then**
4: $T_{malicious} \leftarrow \gamma$
5: **if** $\beta = decision : malicious$ **then**
6: **if** $\gamma < 5$ **then**
7: $\gamma \leftarrow (\gamma + 1)$ ▷ Increase series weighting bonus when additional votes are cast
8: **end if**
9: **end if**
10: $\beta \leftarrow decision : malicious$
11: **else**
12: $\gamma \leftarrow 1$ ▷ Reset the counter when the series is broken
13: $T_{innocuous} \leftarrow (T_{innocuous} + \gamma)$
14: $\beta \leftarrow decision : innocuous$
15: **end if**
16: **end for**
17: **end while**
18: **if** $T_{malicious} \geq T_{innocuous}$ **then** ▷ Return global decision
19: **return** $\langle eid, decision : malicious \rangle$
20: **else**
21: **return** $\langle eid, decision : innocuous \rangle$
22: **end if**

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 used 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 5.1 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.

The algorithms provided here are are designed to provide an extra layer to the common process of monitoring data and then classifying it based on signatures. By introducing the series weighting algorithm, more data points are gathered before a classification about an event in made, producing a more reliable classification.

3.7 Summary

In this chapter, the design of the DMASS architecture is presented; an autonomous agent-based architecture for collecting and analysing information found within the local network. The DMASS features a collection of low-level agents responsible for independently collecting and locally analysing small pieces of information before combining their findings with the groups to produce a final classification for the event. The architecture design makes use of the knowledge representation framework called conditions and effects to describe evidence found within the network and facilitate the propagation of information between agents. A system of domains is described to allow agents to analyse the plausibility of information found within context based on expected attack patterns. In particular, domains allows the agents to evaluate whether a series of

Algorithm 3.4 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.

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

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

local decisions made by the agents is indicative of an actual attack or a false positive based on the attack pattern displayed by the evidence and the agent's confidence in the quality of other agent's abilities. Finally, agent-based algorithms for evaluating information were proposed including three novel features: (i) agent preference through self-selected groups, (ii) series weighting, (iii) and decision threshold cut-off.

Chapter 4

Algorithms for Distributed Analysis

In this chapter, an overview and formalism of a Generalised Anomaly Detection Algorithm is provided for use within networked systems. The motivation for a generalised algorithm arises from the current literature that often proposes algorithms for targeting only a single or small number of network-based attacks. While individual algorithms provide an important mechanism for detecting attacks, given the rise of the APT as a new and emerging threat, a more adaptable approach to detect attacks that have been purposefully obfuscated is required. The generalised algorithm presented in this chapter addresses these concerns by detecting a wide variety of network attacks and aims to still detect attacks given two assumptions about the attacker: (i) the attacker knows the algorithm is in use and (ii) the attacker can evade current signature-based security mechanisms.

The DMASS detailed in Section 3 provides a comprehensive MAS solution for the collection and analysis of evidence found within network systems. While automated systems provide many benefits over traditional centralised and brute-force approaches than analyse everything, they require an initial starting trigger to begin the search for additional evidence. While signatures could be used for this purpose, it is assumed that an APT attacker is capable of evading most signatures by either operating below the threshold of the signature or using a zero-day exploit to bypass the security system altogether. While signatures could be applied with increased sensitivity (i.e., decreased thresholds), they cannot be applied system-wide with increased sensitivity due to the unacceptably high number of false alarms that would result. Instead, they should only be used within the context of the DMASS where prior evidence has given agents reason to suspect an attacker may be present. Therefore, to begin the extended data collection task performed by the DMASS or other automated security systems, a broad anomaly detection algorithm is required to detect the presence of an attacker who can then be further investigated. This chapter presents an anomaly detection algorithm for use within local networks to detect

a wide range of attacks at an average detection rate of 85%, which is a marked improvement over current anomaly detection algorithms that are often specialised towards detecting a single type of attack. Multi-step user traffic patterns are used as the basis for distinguishing between malicious and innocuous traffic activity at a high level that cannot be easily tampered with by the attacker.

Many algorithms for detecting specific types of attacks using anomaly detection already exist [158]. For example, signature-based detection algorithms use a database of predefined examples of malicious activity created by domains experts to identify attacks which is unsuitable for scalable deployment to detect a wide range of attacks. Similarly, anomaly-based detection approaches [5] make use of supervised and unsupervised machine learning to detect normal and abnormal patterns in features obtained from network data. This also presents some limitations as network architectures vary greatly resulting in different *normal* patterns for each operating environment. Furthermore, the attacker can control and manipulate a variety of features (e.g., port number, the length of connection and the overall footprint of the attack) which complicates the process of identifying reliable features that do not change between attackers or instances of an attack. As a result, both anomaly and signature-based detection algorithms suffer from high false positive and false negative rates when deployed to live networks.

The algorithm described here, fits into this existing ecosystem of security technologies as a heuristic-style[1] algorithm for identifying endpoints that do not appear to act like any others. This behavioural difference is not based on specific features that can be easily manipulated by attackers, but on behavioural patterns derived from the interaction with networked services. A broad algorithm such as this could be used to tune an IDSs sensitivity to focus the detection efforts on those endpoints that appear to be behaving anomalously so they can be analysed with increased suspicion.

The development of network anomaly detection algorithms entails (i) the selection of features, of which there are many to choose from, and (ii) the development of the model, which determines how the features are used during the classification process. There are many different ways to model normal and abnormal traffic evidenced by the variety of different datasets and existing solutions. Design choices regarding the scalability, efficiency, deployability and accuracy influence the selection of features and ultimately the utility of the system in live environments. Palmieri et al. [127] argue that systems should be designed to be protocol and service independent, so as to work on the broadest possible range of applications. The approach advocated in this thesis extends these design principles to select features that cannot be easily influenced by

[1]The term "heuristic" is used to reference the algorithms intended use as a generalised model for detecting attackers rather than specific attacks.

the attacker, resulting in a far more reliable solution that can be applied in many environments.

The generalised algorithm, focusing on the early detection of attackers within the local network, relies on features that are not directly recorded in traditional network datasets[2] but are instead inferred from the behaviours inherent in the data as a whole. The algorithm runs in two phases that are continually performed during online detection. During the first phase, the system monitors all network traffic on the internal network and groups the traffic into endpoint-to-endpoint pairs to learn behavioural norms for individual devices. In the second phase, an iterative analysis of the model is performed to find groups of endpoints that act similarly, which is used as the basis for distinguishing between normal and anomalous traffic. The underlying concept of anomaly detection is the existence of measurable differences between normal and abnormal data, without this, classification could not take place. The algorithm uses features found within the internal L2L (Local-to-Local) traffic as the basis for what is considered normal as it is more protected than traffic that originates from outside of the local network. Protected networks have a strict boundary separating the local network and internet by typically using firewalls and IDSs to limit the amount of incoming traffic and are generally used for a specific purpose (i.e., business operations). User activities within the protected network can be classified as either legitimate or adversarial as defined by the organisation's security policy. Using these behavioural norms to distinguish between the two classes is more manageable on a per-network basis owing to the size of the problem. Furthermore, behavioural norms differ between networks, and as such, what is considered normal for one network will not extend to other networks. Therefore, the algorithm is implemented as an unsupervised learning system that uses online training to learn the norms of the specific network. The detection of external attacks outside of the local network is out of the scope of this thesis and has been extensively studied [5].

4.1 System Modelling

To identify anomalous events, a clear baseline for what is considered normal is required. This subjective metric is often based on the network policy that defines which users are allowed to access network resources. At the application layer, this policy is often implemented as access controls (i.e., user identification and password authentication) to prevent unauthorised users from interacting with resources without permission. From the perspective of a network monitor, endpoints that have permission to interact with a service will generate increased amounts of predictable traffic during its interaction. Unauthorised users will not be able to interact in the same way because access controls hinder them. Detecting the differences in these two network

[2]Recall these features may be tied to a particular attack and easily manipulable by an attacker.

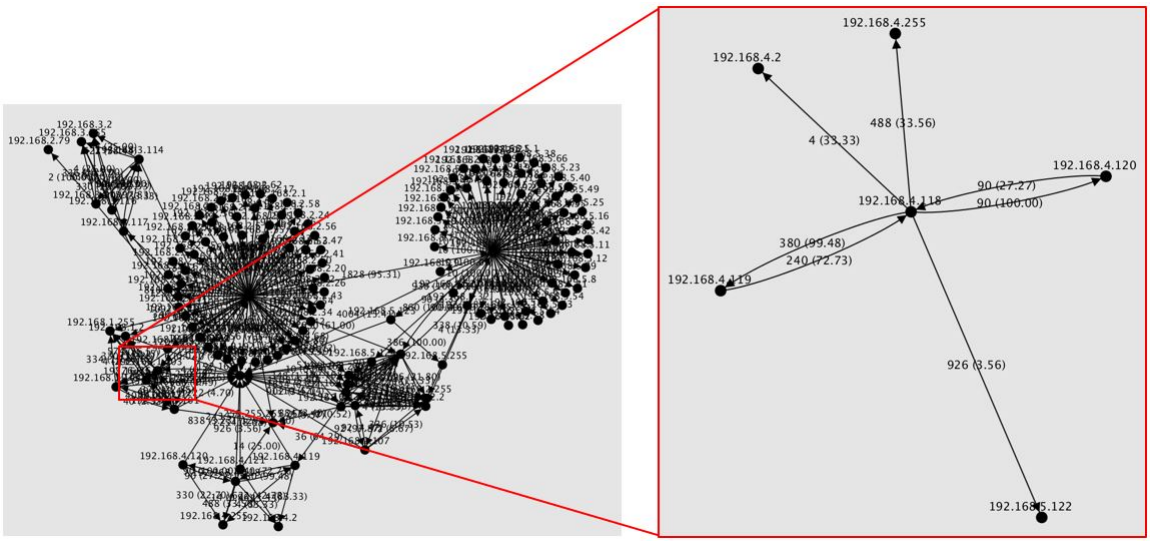

Figure 4.1
Graphical representation of the UNB ISCX dataset highlighting neighbouring endpoints.

patterns forms the basis of the generalised anomaly algorithm. A traffic model is constructed by building a connection graph to represent the connections between pairs of endpoints. Figure 4.1 shows the visualised communications from the UNB ISCX Intrusion Detection Evaluation Dataset [144], containing both normal data and a variety of networked attacks. Within Figure 4.1, nodes represent the individual endpoints labelled by their IP addresses, and the directed edges represent the total amount of packets sent to another host (i.e., all connections made to a host are grouped visually). Additional information attached to the edges is discussed further in Section 4.4. For each node, a local set of communications between its direct neighbours can be isolated to find similar communication patterns between the node's direct neighbours. The decision to model endpoint connections as a whole within the model was made to reduce the attack surface of the system. Flow-based anomaly detection often uses aggregated statistics about the shape and size of individual flows without considering the control an attacker has over them. An attacker may, for example, fragment a single attack over many network flows to obfuscate it from detection. By implementing the algorithm in a more generalisable way, the aim is to detect broad behavioural changes that may be indicators of an attack and avoid the associated problems of attack obfuscation.

The model is used to distinguish between normal and anomalous connections by finding other endpoints that have a similar network footprint. Using the hypothesis that legitimate users will exhibit broadly similar connection patterns when interacting with a networked service, the graph in Figure 4.1 is iteratively processed to find endpoints that communicate with other endpoints similarly. The similarity is defined by the number and ratio of connections shared between two

endpoints as compared to other endpoints. For example, a group of endpoints are considered similar if they send and receive a similar number of flows to a common endpoint, this is used as the basis for defining a network footprint for groups of endpoints. If a particular endpoint makes many connections that fit into the network patterns of many groups, it can be said with confidence that the connections are legitimate because the connections can be explained by endpoints interacting with networked services protected by authentication systems. Typically, normal traffic is more abundant than attack traffic, therefore, basing the network norms on the majority interaction can provide an accurate way to model normal activity. Note, users must have the required application-layer authentication to interact with many networked services. On the contrary, attackers attempting to penetrate the internal network must go through a period of acquiring these privileges, during which time they will generate a unique network footprint as they attempt reconnaissance and privilege escalation which will cause unusual traffic to be generated. It is at this point of the attack lifecycle that the algorithm attempts to detect the intrusion.

4.2 Dataset Analysis

For the goal of detecting infiltrations from the inside (i.e., post-compromise detection), consideration is given the available datasets and the effects of data granularity against the likely outcome of anomaly detection algorithms.

The granularity of the dataset is an important consideration as it defines the available detection scope, for example, a highly granular dataset of features for biometric keyboard analysis (e.g., typing speed, key-press pressure, average words per minute, etc.) will limit the detection of attackers to a particular device and set of attacks. Within the literature, authors who select granular datasets often report more accurate results, for example, Rudrapal et al. [137] reported 86% accuracy in distinguishing between users by employing the aforementioned biometric features to detect attackers. However, when datasets with a low granularity are used, for example, network-layer datasets consisting of raw packets or flow statistics, the detection accuracy can often decrease significantly (e.g., Cao et al. [34] reported a detection accuracy of 58%). The poor detection accuracy of low granularity datasets can be attributed to the datasets wide scope containing a variable number of attacks, each of which manifests in a unique network footprint. The difficulty of distinguishing between normal and abnormal using machine learning or anomaly detection increases as the definition of abnormal expands to include the variety of possible attacks. When the definition of abnormal is constrained to mean any value outside of a given range for a particular feature (e.g., using the words per minute feature), the only vari-

ability is how well the selected feature can distinguish between the legitimate and illegitimate user in that use-case. However, in highly variable environments with many different types of attacks, some of which remain unknown until they occur for the first time (e.g., zero-day exploits), the suitability of features to each attack may be different and cannot be easily modelled in the same way. Furthermore, at the network-layer multiple users interact which transforms the problem from a single user-to-attacker relationship to a problem of defining multi-user norms to encapsulate all of the legitimate user's behaviours. With an increased number of possible attacks and the inclusion of multiple users, the variability of defining what is normal increases exponentially. However, the benefit of performing detection at the network layer is that more of the information about the attacker is available, this manifests as network footprints left by the attacker that can be modelled for detection.

4.2.1 Selecting Features for Anomaly Detection

Following the decision to use flow-level data to increase deployability, consideration for feature selection is needed. Within cyber security environments, the problem of overfitting should be given particular attention because of the dynamic nature of network environments that can also vary between organisations, in addition to the adversary, whose goal it is to remain hidden, and can control many of the features recorded in the datasets. Many flow-level datasets follow a similar structure to the UNB ISCX Intrusion Detection Evaluation dataset [144] which uses nineteen packet features including the application protocol name, payload sizes, payload contents, Transmission Control Protocol flags, IP addresses, port numbers and timestamps (refer to Table 4.1). Given that an attacker can directly influence each feature in flow-level datasets, either by targeting a different service, IP address or directly manipulating the values (e.g., port numbers, flags and payloads), relying on any subset of features may result in a model overfitted to the particular implementation of the attack. Implementational overfitting occurs because there is no standard footprint for an attack in relation to the feature-values, and as such, machine learning algorithms will often model the specific implementation of an attack, unique to an attacker, rather than correctly model a reproducible attack footprint that can be used across many occurrences. Note that features in Table 4.1 marked with an asterisk were manually removed from the dataset for testing to increase the performance of the machine learning algorithms. Throughout the literature, these features are often removed as they are a poor indicator of an attackers general pattern (e.g., the attackers IP address will change between datasets and attacks). Furthermore, some features like the source/destinationPayloadAsUTF and source/destinationPayloadAsBase64 were removed as they are text-based features which

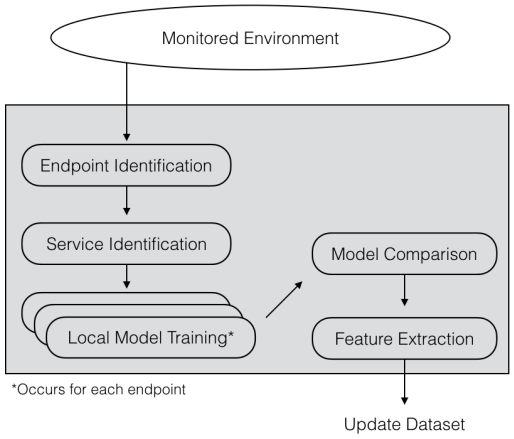

Figure 4.2
Proposed model for feature extraction.

are not well supported by the chosen machine learning algorithms and are better suited for signature-based detection. Finally, the startDateTime and stopDateTime were removed as they are not a good indicator of stealthy attacks that may be purposefully slowed down to avoid detection and do not generalise well across multiple datasets or attack types.

Within the UNB ISCX dataset, several malicious activities are performed:

- Information gathering about the target including network IP ranges, nameservers, mail servers and user email accounts.
- Mail server enumeration
- Buffer overflow attack against a vulnerability in Adobe Reader
- Opening reverse connections to the internet
- Uploading post-compromise tools
- Internally scanning the network
- SQL Injection
- Backdoor creation

The attacks performed throughout this scenario are commonly used during network intrusions and separate solutions focusing on each task have been individually proposed. However, detecting the presence of the attacks at the network layer has proven increasingly difficult as the number of possible attacks and the sophistication of advanced evasion techniques grows. Tavallaee et al. [151] in a criticism of the KDD CUP 1999 dataset [84] comment on results

Table 4.1
UNB ISCX Intrusion Detection Evaluation Dataset Features.

Feature	**Type**	**Rank**†
appName	Nominal	5
totalSourceBytes	Numeric	9
totalDestinationBytes	Numeric	8
totalDestinationPackets	Numeric	12
totalSourcePackets	Numeric	14
sourcePayloadAsBase64*	Nominal	15
sourcePayloadAsUTF*	Nominal	18
destinationPayloadAsBase64*	Nominal	16
destinationPayloadAsUTF*	Nominal	17
direction*	Nominal	19
sourceTCPFlagsDescription	Nominal	10
destinationTCPFlagsDescription	Nominal	11
source (IP)*	Nominal	7
protocolName	Nominal	13
sourcePort	Nominal	3
destination (IP)*	Nominal	6
destinationPort	Nominal	4
startDateTime*	Nominal	2
stopDateTime*	Nominal	1
tag	Normal, Attack	N/A

* Features manually removed to prevent overfitting and to improve machine learning algorithm performance.
† Ranked through information gain attribute evaluation.

from the machine learning literature that often report a very high detection rate (upwards of 98%). The results are compared against the low adoption rates of commercial IDS solutions that overwhelmingly use signature-based detection rather than the proposed models.

4.3 Critical Analysis of Machine Learning Approaches

In this section, several aspects of the network security environment that contribute to the low adoption of machine learning compared to signature-based intrusion-detection are discussed. The contents of this section were first published in [90].

Attack Implementation Differences The general footprint for an attack is often used to detect similar attacks in the future. Features such as which device was targeted, the duration of the attack, payload and protocols utilised can be used to model the event. However, using the model to detect future instances of the same attack may result in increased false negatives as different attackers perform the same type of attack but configured with different parameters. The issue of implementation differences causing the same attack to look different arises at the network layer as attacks are often defined by multivariate collections of features, often under the control of the attacker, rather than a single feature with an acceptable range. Building a model to detect a specific attack using a subset of features from Table 4.1 may not be fully generalisable to future events if there are implementation differences in the way that the attack is performed.

Preventing Overfitting Overfitting is a particular problem in network anomaly detection where the many nominal features, with a variety of possible values, increases the complexity of the detection models. Within the literature, several features are commonly removed from datasets to decrease the probability of overfitting occurring (refer to features marked by * in Table 4.1), however, overfitting can still occur in the cleaned datasets because of the aforementioned implementation differences. With most features under the control of the attacker, any over-reliance on a particular feature-value will result in reduced accuracy if altered by the attacker after the model is built.

Detecting Stealthy Attacks In the case of advanced stealthy attacks, many of the numeric features measuring information about the volume of traffic (e.g., totalSourceBytes, totalDestinationBytes, totalSourcePackets and totalDestinationPackets) become less reliable as they do not accurately represent the actual attack. Many attacks can be slowed to make the packets appear as if they are apart of separate network flows rather than belonging to a single network

event. The effectiveness of statistically analysing the volume of traffic flows also decreases as the number of users and networked services grows because of the increased variability experienced.

Generalised Models Many types of attacks will leave detectable footprints at the network layer if they move between host. As such, performing anomaly detection at this layer necessarily involves the detection of many attacks, most of which are unknown to both the network administrator and the security community. Given many possible attacks with a propensity to manifest in different network footprints, any generalised model built to detect multiple attacks will likely detect some but miss others.

4.3.1 Proposed Feature Selection Model

To overcome the discussed problems, the use of two new features generated from the traffic flows to improve the classification accuracy are considered. While relying on any subset of feature values may result in a poor performance due to the variable nature of the attacker and attack implementation, using the broad communication patterns monitored from both a local and global perspective to classify connection flows increases scalability and performance. The proposed architecture for feature extraction is displayed in Figure 4.3 and is composed of the following modules:

Monitored Environment The architecture takes as input network flows or raw packet connections from either a centrally collected network monitor or collection of distributed network sensors. For this prototype, data is taken directly from the labelled UNB ISCX dataset containing network flows for an internal intrusion.

Endpoint Identification The proposed model assumes a set of endpoints $e \in E$ that exist within the local network. Each endpoint is modelled, and the connections or flows associated with it are locally stored for use within the model.

Service Identification The model assumes the existence of networked services $s_e \in S$ where each service belongs to some endpoint e. Services are identified within the model by the volume of incoming traffic to be expected from multiple endpoints such that services are expected to have a one-to-many relationship between itself and other endpoints. Normal traffic patterns are defined under the proposed model by comparing the traffic models of endpoints interacting with networked services to find the normal interaction pattern and then grouping agents by how they interact over multiple services.

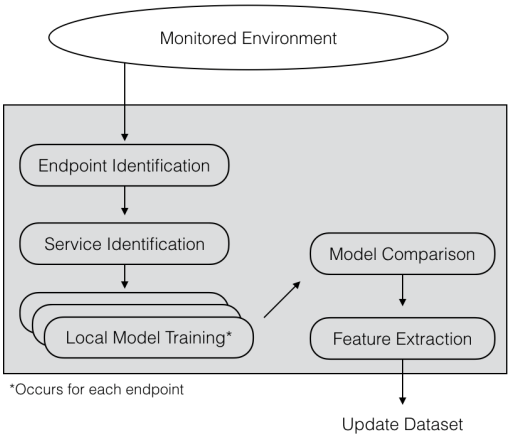

Figure 4.3
Proposed model for feature extraction.

Local Model Training For each endpoint if a network service can be identified, a local model is built to classify the one-to-many relationship between itself and other endpoints using the volume and directionality of the traffic. For each endpoint, there exists a set of connections or flows C_e such that some are incoming $c_{in} \subseteq C_e$ and others outgoing $c_{out} \subseteq C_e$. The model locally identifies groups of endpoints that exhibit similar connection patterns by measuring the volume difference between sizes c_{in} and c_{out}. The output from this stage is to add a new feature to the dataset named *localGroupSize* (L_e) describing the number of endpoints that could be locally grouped together as exhibiting similar traffic patterns.

Model Comparison To validate the results of the previous step, the local groups are compared to find crossover between locally grouped endpoints and multiple services. A new feature *globalGroupSize* (G) is formally defined as $\{G : |L_e \subseteq \{L_1, ..., L_n\}| >= \alpha\}$ where the size of any two local set comparisons is greater than the threshold α (presently set as 2). If a set of locally grouped endpoints is found to share the connection profile of many endpoints, more confidence can be placed in a classification that the traffic is legitimate rather than abnormal.

Feature Selection & Update Dataset For this prototype algorithm, the features described above are extracted and written to the initial input dataset for manual off-line analysis.

4.3.2 Evaluation of Novel Features

Table 4.2 lists the results of a feature analysis of the UNB ISCX dataset with the additional *LocalGroupSize* and *GlobalGroupSize* features included in the dataset. Several feature-analysis algorithms and search methods were used during the evaluation to calculate the most predictive

Table 4.2
Results of Feature Analysis

Attribute Evaluation Method	**Search Method**	**Most Predictive Features**
Correlation-Based Subset Selection [68]	Best First (Greedy Search)	Source Port, Protocol, Global Group Size
Classifier Subset Evaluation using C.45 decision tree without error pruning *	Best First	App Name, Total Source Packets, Source Port, localGroupSize
Classifier Subset Evaluation using C.45 decision tree with error pruning *	Best First	Total Source Bytes, Source Port, Global Group Size, Local Group Size
Classifier Subset Evaluation using Naive Bayes *	Best First	Source Port, Local Group Size
Classifier Subset Evaluation using RBF Neural Network *	Best First	Source Port, Global Group Size, Local Group Size
Consistency Subset Evaluation [104]	Best First	App Name, Total Source Bytes, Local Group Size
Grain Ratio Attribute Evaluation	Ranked Search	Global Group Size, Source Port, Protocol

* Made use of an additional testset: UNSW-NB15 dataset [117].

features. The Classifier Subset Evaluation made use of an additional test set name UNSW-NB15 [117] containing similarly structured network flows. The algorithm evaluates the predictiveness of the features over the training and test sets using a chosen algorithm and ranks the most predictive features using a best first greedy search. The C.45 decision tree, Naive Bayes and RBF Neural Network classifiers were chosen for analysis due to their ability to perform well on noisy data. From the results, the two proposed features were continually ranked high by the respective algorithms in their predictive ability when tested on both a single and multiple datasets.

The proposed features overcome the previously identified problems of scalability and implementation differences by combining the local view of the network with the global view. Advanced evasion techniques will typically be able to evade detection at the local level but not at the global level where traffic patterns are compared across the entire network. Locally, it is possible to tailor attack activities to remain undetected by performing such activities below the detection thresholds, however, if other users in the network do not also appear to interact in a similar way, increased suspicion would be placed on the activity. Under such circumstances, the connections belonging to the attacker may not fit any global group and thus have the global group size

variable set to zero.

While the results from the feature analysis are encouraging, the proposed model would be better suited as a separate algorithm to fully take advantage of the proposed model. While grouping endpoints into local and global sets have proven to be a good indicator of the class, other features with problems discussed in Section 4.3.1 were also selected and would limit the deployability of any machine learning model used as a result. As such, future work will focus on the continued development and separation of the proposed model from the problematic features. The impact of this result is to demonstrate the use of behavioural-based features that go some way to addressing the problem of overfitting in network-layer anomaly detection.

4.4 Generalised Anomaly Detection for DMASS

In this section, the generalised anomaly detection algorithm is detailed. The algorithm consists of four ordered steps that are continuously performed to monitor and log network connections, perform local model traversal, perform remote model traversal and finally to remove explained connections.

Local Connection Analysis (Refer to Algorithm 4.1) During the first stage, the connections received by each endpoint in the local network are analysed. This requires connection monitoring, which can be done centrally (e.g., by an IDS) or distributively (e.g., by sensors located on the network). Given that only the flow-layer network connections are required (e.g., TCP/IP are the most common protocols) rather than host-based data, it is assumed that 100% visibility is achievable.

More formally, the network is composed of a set of endpoints (this set is denoted by E) identified by their IP address. The set E is formed by local (E_l) and remote (E_r) endpoints ($E = E_l \cup E_r$). The set of monitored connections (denoted by C) is formed by tuples $\langle s, r \rangle$ where $s \in E$ is the endpoint sending the data through the connection and $r \in E$ is the endpoint receiving the data. The connections monitored are organised into logical data structures within the model to preserve the higher-level application layer features that will be the focus of the classification stage. Specifically, for each local endpoint $e \in E_l$ the algorithm maintains the set of "neighbours" (denoted by N_e) containing information about other local endpoints that are directly communicated by e. Within the logical structure, value d is defined as the percentage difference between the number of incoming and outgoing flows.[3]. The volume-based metric d is

[3]Note the particular data exchanged between two nodes is not used by the algorithm, and hence it is is not captured by the representation.

Algorithm 4.1 Local Connection Analysis Function

Require:
$e \in E_l$ the endpoint whose connections are being analysed
N_e the information maintained about of neighbours of e
$C = \{c_1, ..., c_2\}$ a set of connection flows starting and ending at the endpoint $(\forall \langle s, r\rangle \in C : s = e \text{ or } r = e)$

1: **for** $\langle s, r\rangle \in C$ **do**
2: **if** $\{s, r\} \cap E_r = \emptyset$ **then** ▷ The connection is local
3: **if** $s = e$ **then** ▷ The connection is sent by node e
4: **if** $\exists \langle r, sC, rC, d, t\rangle \in N_e$ **then** ▷ The receiver endpoint is in the list of neighbours
5: $d \leftarrow |(sC - (rC + 1))/((sC + (rC + 1))/2.0) * 100|$
6: $N_e \leftarrow N_e \setminus \{\langle r, sC, rC, d, t\rangle\} \cup \{\langle r, sC, rC + 1, d, t\rangle\}$ ▷ Update neighbour information
7: **else** ▷ The receiver endpoint is not in the list of neighbours
8: $N_e \leftarrow N_e \cup \{\langle r, 0, 1, 50, 0\rangle\}$ ▷ Add the receiver endpoint to the list of the neighbours
9: **end if**
10: **else** ▷ The connection is received by node e
11: **if** $\exists \langle s, sC, rC, d, t\rangle \in N_e$ **then** ▷ The sender endpoint is in the list of neighbours
12: $d \leftarrow |((sC + 1) - rC)/(((sC + 1) + rC)/2.0) * 100|$
13: $t \leftarrow (sC/rC) * 100$ ▷ Update the percentage of total flows received
14: $N_e \leftarrow N_e \setminus \{\langle s, sC, rC, d, t\rangle\} \cup \{\langle s, sC + 1, rC, d, t\rangle\}$ ▷ Update neighbour information
15: **else** ▷ The receiver endpoint is not in the list of neighbours
16: $N_e \leftarrow N_e \cup \{\langle s, 1, 0, 50, 0\rangle\}$ ▷ Add the receiver endpoint to the list of the neighbours
17: **end if**
18: **end if**
19: **end if**
20: **end for**

Algorithm 4.2 Local Model Update Function

Require:
$e \in E_l$ the endpoint whose connections are being analysed
N_e the information maintained about of neighbours of e
$C = \{c_1, ..., c_2\}$ a set of connection flows starting and ending at the endpoint $(\forall \langle s, r\rangle \in C : s = e \text{ or } r = e)$
$S_e = \{..., \langle d, s\rangle, ...\}$ is a set of distinct locally similar pair of endpoints (s and s') of N_e indexed by the percentage difference value d
$\epsilon \in \mathbb{R}$ an error value used in the calculation of S_e

1: **for** $\langle s, sC, rC, d, t\rangle \in N_e$ **do** ▷ Begin loop to compare all neighbouring endpoints
2: **for** $\langle s', sC', rC', d', t'\rangle \in N_e$ **do**
3: **if** $s <> s'$ and $d >= (d' - \epsilon) \wedge d <= (d' + \epsilon)$ **then** ▷ Group endpoints by d if they are within the margin of error ϵ
4: $S_e \leftarrow S_e \cup \{s, s'\}$
5: **end if**
6: **end for**
7: **end for**

Algorithm 4.3 Remote Model Update Function

Require:

$e \in E_l$ the endpoint whose connections are being analysed
$S_e \subseteq E_l$ the set of distinct locally similar endpoints
$G_e \subset E_l$ the set of remotely similar endpoints
$\eta \leftarrow \mathbb{N}$ the minimum amount of endpoints required for classification as a remotely similar set.

1: $NS \leftarrow []$ ▷ is the list of similar endpoints for all the neighbours and neighbours of neighbours
2: **for** $s \in S_e$ **do**
3: $NS = NS \oplus f(s)$ ▷ Function initiated by e to remotely retrieve from $s \in N_e$ its the similar set S_s
4: **end for**
5: **for** $\langle s, sC, rC, d, t\rangle \in N_e$ **do**
6: $NS = NS \oplus f'(s)$ ▷ Function initiated by e to remotely retrieve from $s \in N_e$ the similar set they have obtained from its neighbours
7: **end for**
8: **for** $\langle s, sC, rC, d, t\rangle \in N_e$ **do**
9: **if** $count(s, NS) >= \eta$ **then** ▷ If s appears in the similar sets of η neighbour endpoints
10: $G_e \leftarrow G_e \cup \{s\}$ ▷ Add similar endpoint to G_e
11: **end if**
12: **end for**

Algorithm 4.4 Model Analysis Function

Require:

$e \in E_l$ the endpoint whose connections are being analysed
$\alpha \in \mathbb{R}$ the maximum allowed value for the percentage of total received metric t ▷ Prevents endpoints with few connections from being prematurely classified as non-malicious
N_e the information maintained about of neighbours of e
$G_e \subset E_l$ the set of remotely similar endpoints belonging to e

1: **for** $\langle s, sC, rC, d, t\rangle \in N_e$ **do** ▷ Iterate the neighbours set
2: **if** $s \in G_e$ **then**
3: **if** $t <= \alpha$ **then** ▷ Compare the percentage of total received value t against the α threshold
4: $\gamma \leftarrow s$ ▷ Set of innocuous endpoint identities
5: **else**
6: $\delta \leftarrow s$ ▷ Set of malicious endpoint identities
7: **end if**
8: **else**
9: $\delta \leftarrow s$ ▷ Set of malicious endpoint identities
10: **end if**
11: **end for**

used to distinguish between legitimate traffic which typically exhibits longer and more complex exchanges of traffic and malicious traffic which manifests in shorter or prematurely terminated connection flows due to preventative access controls. The metric t measures the amount of traffic exchanged between two particular endpoints in relation to the total amount of traffic exchanged by that endpoint. This metric is used in the later stages of the algorithm to prevent the premature classification of low volume endpoints.

Definition 7 (Neighbour Set) *Given a local endpoint $e \in E_l$, its neighbours are defined as a set formed by tuples $\langle s, sC, rC, d, t \rangle$ where*

- *$s \in E_l$ is an endpoint that has communicated with e, i.e., $\exists \langle e, n, d \rangle \in C$ or $\exists \langle n, e, d \rangle \in C$;*
- *$sC \in \mathbb{N}$ is the number of flows sent by s, i.e., $sC = |\{(e, n, d') : (e, n, d') \in C\}|$;*
- *$rC \in \mathbb{N}$ is the number of flows received by s, i.e., $rC = |\{(n, e, d') : (n, e, d') \in C\}|$;*
- *$t \in \mathbb{R}$ is the percentage of total flows received from a particular endpoint, defined as: $(sC/rC) * 100$;*
- *$d \in \mathbb{R}$ is the percentage difference between the amount of sent and received flows defined as:*

$$\frac{|sC - rC|}{\frac{sC+rC}{2.0}} \times 100$$

For each endpoint within the local network, a neighbour set is created to store relevant information and to facilitate data exchange between the entities within the model. As new connections are captured for an endpoint, the algorithm can update the relevant entities neighbour rather than recalculate all neighbour sets.

Local Model Update (Refer to Algorithm 4.2) During the second stage, the neighbour sets for each local endpoint are traversed to discover similar interaction patterns that will define the normal activity for each endpoint. The volume difference of traffic exchanged between two endpoints is used as a metric to group endpoints that act similarly. Hence the normal traffic patterns are inferred from how the majority of endpoints interact with a given endpoint.

In particular, this algorithm finds for each local endpoint $e \in E_l$ a set of endpoints that interact with e similarly to other neighbour endpoints in (this set is denoted by the set $S_e \subseteq E_l$). Pairs in S_e interact similarly with e. Any pair of endpoints in N_e whose percentage difference value (d) are similar within a margin of error (ϵ) are added to S_e as similar endpoints that

are assumed to be communicating with a service in a common way (i.e., producing a similar interaction pattern). The percentage difference value of connection flows is used as a tamper-resistant feature when compared to how other endpoints also interact with a service. This is owing to the authentication mechanisms commonly used to protect local network services that will prevent unauthorised attackers from interacting with the service in the same way, preventing the generation of similar flow-layer interaction patterns. Attackers are more likely to evoke the ending of a flow by attempting to interact with a service while unauthenticated causing the service to close the connection (i.e., by sending a TCP RST or FIN packet[4]). In future iterations, if new endpoints are found to act in a similar way to previously defined patterns, they are added to the model for analysis in the next steps.

Definition 8 (Similar Endpoints) *Given a local endpoint $e \in E_l$, the similar endpoints for e (denoted by $S_e \subseteq E_l$) are defined as a set formed by local endpoints where for all $s \in S_e$:*

- *There is a neighbour element corresponding to endpoint s: $\langle s, sC, rC, d, t\rangle \in N_e$;*
- *There is another neighbour element such that $\exists s' : \langle s', sC', rC', d', t'\rangle \in N_e$;*
- $d \in [d' - \epsilon, d' + \epsilon]$

Remote Model Update (Refer to Algorithm 4.3) The previous step can be viewed as each endpoint locally finding its normal interaction footprint defined by how other endpoints interact with it. Following this definition, a similar process is undertaken to validate the local norms by finding groups of endpoints that act similarly when communicating with other service-endpoints. In modern networks containing a variety of networked services that can be accessed by the endpoints, this process reveals multi-step traffic patterns that legitimate endpoints perform forming the basis of how the system distinguishes between innocuous and malicious users. Formally, this function retrieves the set of locally similar endpoints from its neighbours and compares it with its local similar-set to find similar overlapping interactions. A communication action $f(s)$ is defined to allow the remote retrieval of information from s about s's similar set S_s (the similar sets it has received from its neighbouring endpoints). The sets retrieved are stored in a list (denoted by NL) formed by the similar sets obtained from the neighbours and neighbours of neighbours. In the case where the system is deployed as distributed sensors around a network, these actions will allow sensors to exchange the information required during this stage. The local set S_e is compared to the remote list NS by counting how many times a given endpoint in S_e appears in the list NL (this is calculated by the function *count*). The purpose of

[4]The TCP protocol uses RST and FIN packets to end attempted communication flows.

this is to find multiple common communications vectors that an endpoint conforms with. If the count is greater than a threshold η the endpoint is added to the set G_e. Endpoints placed into G_e have satisfied the condition that they (i) interact predictably as defined by the local norm (i.e., Algorithm 4.2), and (ii) interact with the endpoints surrounding neighbours in a similar way (i.e., Algorithm 4.3). Endpoints which act similarly when interacting with a small subset of services are inferred to be apart of the same clique (i.e., endpoints belonging to a particular department), and deviation from group norm can be used as an indicator of compromise.

Definition 9 (Remotely Similar Endpoints) *Given a set of locally similar endpoints S_e, and the list of similar sets NS, remotely similar endpoints for e (denoted by G_e) are defined as a set of endpoint identities where for all $s \in G_e$:*

- *$s \in S_e$;*
- *s appears at least η times in the list NS;*
- *where $\eta \in \mathbb{N}$ is a threshold value currently set to 2.*

Model Analysis (Refer to Algorithm 4.4) The final function iteratively processes the model build during the previous stages to classify endpoints as either malicious or innocuous. The set of neighbours is iteratively processed to find to find remotely similar endpoints (i.e., those in G_e) for which the percentage of total received value t is greater than a threshold α; to prevent endpoints with only a few connections from being considered normal without due consideration. Note that t is recalculated during the flow processing stage in Algorithm 4.1. A minimum threshold for exhibited traffic is desirable to prevent unrepresentative samples of network traffic to define system norms. By default, all endpoints are considered to be malicious and innocuous endpoints are iteratively marked as such during the classification stage.

Definition 10 (Model Analysis) *Given a set of neighbours N_e and the set of remotely similar endpoints G_e, endpoints $\langle s, sC, rC, d, t \rangle \in N_e$ are classified into innocuous endpoints if:*

- *innocuous iff $s \in G_e$ and t is greater or equal than a threshold alpha that must be reached before a classification of innocuous can be made;*
- *malicious otherwise.*

The purpose of the algorithm is to provide a method of detection for use within the network layer to detect a wide range of attacks. While the algorithm achieves this goal of being able to detect a wide variety of attacks at an acceptable detection accuracy (see Section 5.3 for results)

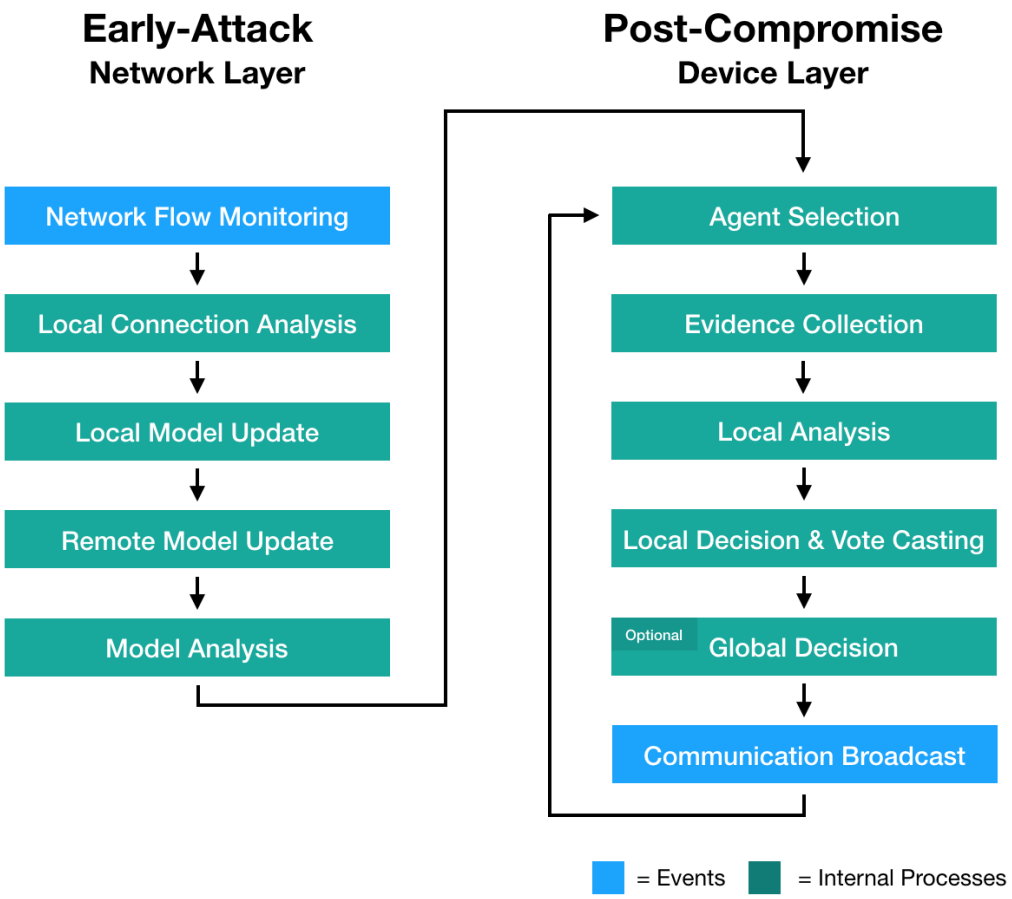

Figure 4.4
Usage of the Generalised Anomaly Detection Algorithm and the DMASS together.

the vulnerability of this system is its inability to identify the specifics of the attack or perform an in-depth investigation of further compromise within the system. As such, the algorithm is designed to be used specifically within the DMASS to guide the agents to an initial start -point and allow further investigation to take place. Figure 4.4 shows the process flow for the algorithm and DMASS when working together with the output of the model analysis stage resulting in the identification of a device that should be the target of the DMASS-based agent investigation.

Together, each system address the weakness of the other system to provide a comprehensive solution to detect and respond to cyber attacks. The advantages of the generalised anomaly detection algorithm is its ability to (1) detect a wide range of attacks, and (2) operate on the network layer, while its weaknesses are (1) the output of the algorithm is a device address that appears to be acting anomalously, not a comprehensive analysis of the event. The advantages of the DMASS are (1) its ability to perform in-depth investigations into events, (2) its ability to proactively gather relevant information, and (3) its ability to work on-demand rather than as a brute-force detection system, while its weaknesses are (1) the on-demand approach requires an

external indication to begin the investigation. It is clear that the two systems are mutually beneficial and addresses the weaknesses inherent within each system with the generalised anomaly detection algorithm providing the starting point for the DMASS investigation.

4.5 Summary

In this chapter, the generalised anomaly detection algorithm was presented as a high layer algorithm for detecting a wide variety of attacks. The literature review identified the APT as a particularly stealthy and evasive threat to modern networks requiring adaptable solutions that do not rely on any one feature of the attack. The proposed generalised algorithm addresses this threat model by relying on user interaction patterns rather than network layer features such as service type or port numbers. The algorithm instead models attackers in a tamper-resistant way that cannot be manipulated by slowing down an attack or performing lateral movement using only a small amount of connections. To show the predictiveness of the algorithm, a feature analysis was performed on the two main tamper-resistant features extracted from the data: (i) local group size, and (ii) global group size. Finally, a formal description of the algorithm is provided with details of how the algorithm can be used together with the DMASS provided in Chapter 3.

Chapter 5

Simulation Results & Discussion

In this chapter, the results of the algorithms from both the DMASS and Generalised Anomaly Detection Algorithm are provided. Evaluation of the algorithms took place within simulation software specifically designed to allow MAS to work with security datasets and is detailed in Chapter 5.1. The results from the DMASS show a detection rate increase of up to 20% in high false alarm environments and an efficiency increase of up to 50% made over traditional monolithic intrusion detection systems. The results of the Generalised Anomaly Detection algorithm show its capability to detect a wide variety of network-based attacks at an average detection rate of 85% providing an accurate and scalable way to detect the initial traces of compromise.

5.1 Simulator Design

To facilitate the development of the DMASS and generalised anomaly detection algorithm, a simulator was developed to work with the appropriate data. Figure 5.1 shows the core packages within the software (written in Java) structured as follows (i) development of the DMASS is largely contained within the *Agent*, *DataStructures* and *Profiles* packages, with code for the various algorithms described in Chapter 3.6, (ii) the generalised anomaly detection algorithm is contained within the *NetworkFlow* and *DataStructures* packages, and (iii) tertiary work done on the application of MAS to cloud web service composition is contained within the *Cloud* package.

Utilities. The simulator is designed to test agent-based simulations in many environments. Figure 5.2 shows the utility window (left) with details of the current test (e.g., the number of simulations, attacks and repetitions). Additional information about the performance statistics of the simulator (e..g,., used RAM, Maximum RAM and available RAM) are continually updated as the simulator runs. During non-demonstration tests, the simulator is performed without a graphical interface to increase the speed at which tests are performed, allowing for many thou-

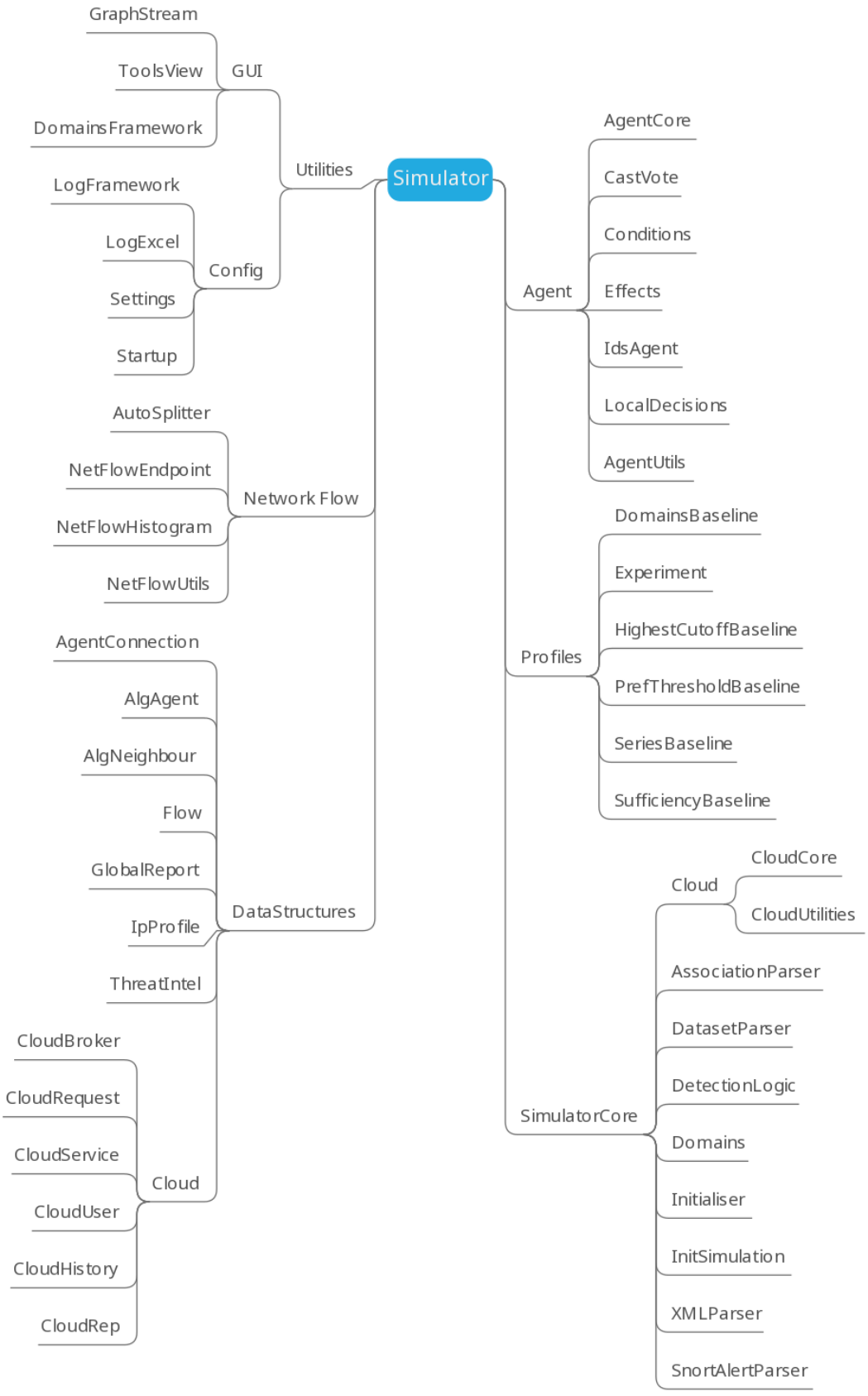

Figure 5.1
The core packagages of the simulator used during the development of the DMASS and generalised anomaly detection algorith.

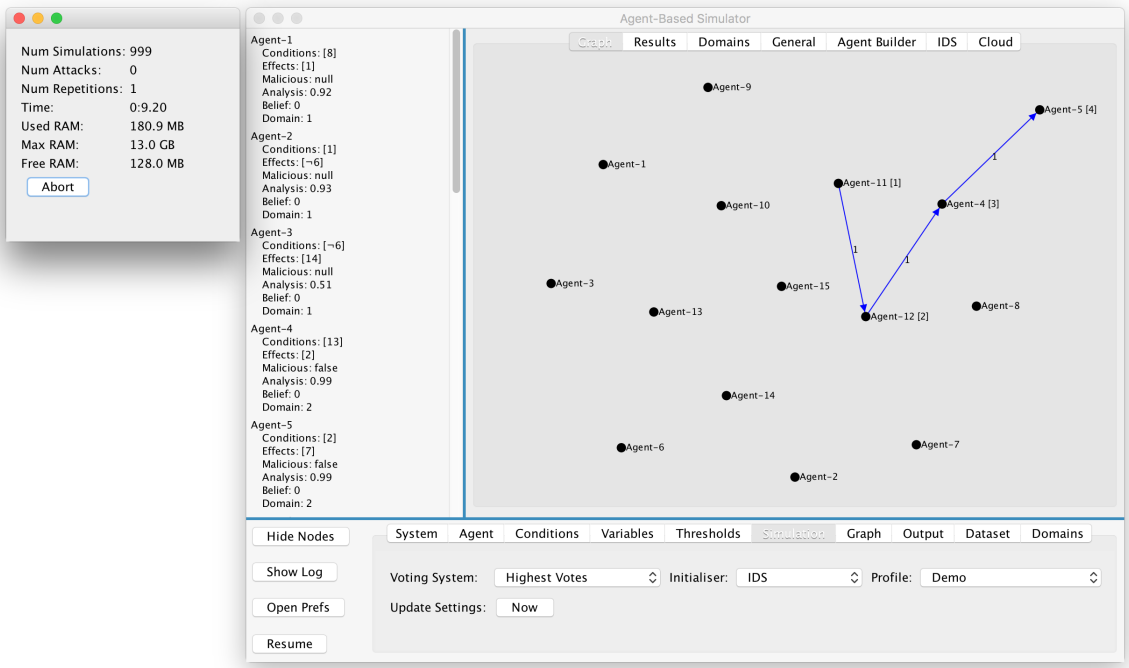

Figure 5.2
Agent view of the simulator.

sands of simulations to be performed in a short amount of time. Information about initialised agents is shown in the left-hand portion of the main window with details of each agent's internal state (e.g., conditions, effects, performance analysis and information about the corresponding domain). The simulator uses 62 configurable settings stored within an external configuration file to control many aspects of the simulator, GUI and data output. A subset of the settings controlling the agents are shown below:

```
09) Prioritise number input types = false
10) Show edge action labels = false
11) Use string agents = false
12) 1 condition = 80
13) 2 conditions = 15
19) Global malicious threshold = 1.0
20) Local report threshold = 0.0
21) Event flagging threshold = 0.0
22) Preferred agent threshold = 0.7
23) Do column resize = false
24) Analysis max = 1.0
26) Analysis min = 0.5
```

AgentCore. Each agent is programmatically similar to each other with similar functions, however, each is initialised with unique variables such as conditions and effects to allow the agents to interact with each other. Two conditions and effects classes exist to distribute the information among the agents, within the simulator a maximum of 6 conditions is allowed per agent. Agent decisions are a combination of local decisions and the global report. The LocalDecision class is internal to each agent and is placed inside the GlobalReport class stored within the DataStructures package. Each agent has the ability to target another agent and deliver the global report if they currently have it.

Profiles. Within the simulator an experiment can be created through the addition of a new profile. A profile is a class composed of an *initialSettings* method that overrides the default settings within the simulator and then 1 to 11 optional methods titled *round 1* to *round 11* which may contain additional settings overrides unique to each round. The purpose of the profile class is to facilitate testing groups of agents under different environment settings. This allows the same group of agents, under the same environmental conditions, to be simulated in different ways to enable a fair comparison to take place. For the purposes of this thesis, profiles were used to test the performance of different algorithms and settings on the same group of agents, for example, to compare the baseline and the series weighting algorithm and to test the effectiveness of the preferred agents feature on those algorithms. Below lists three rounds used during the series baseline tests, round 1 uses the baseline algorithm with the "usePreferredAgent" setting disabled, round 2 uses the series weighting algorithm with the same setting disabled, and round 3 uses the same algorithm with the setting enabled.

```
@Override
public void round_1() // Baseline
{
    Settings.selectedVotingSystem = "Highest Votes";
    Settings.usePreferredAgent = false;
    Settings.ignoreBadAgents = false;
    Settings.letAllAgentsRun = true;
    Settings.selectAgent = "None";
}

@Override
public void round_2() // Series weight baseline
{
    Settings.selectedVotingSystem = "Series Weighting";
```

	A	B	C	D	E	F	G	H	I	J	K	L
1	Run	Iteratio	Agent	Domain	Conditions	Effect	Current Knowledge	Agents LD	Analysis	Is Corr	Requestees Queue	Worker History
2	1	1	Agent-40	10	[25]	[40, 25]	[40, 25]	false	0.96	true	{Agent-41=1}	
3	1	1	Agent-41	11	[25]	[11]	[40, 25, 11]	true	0.589	false	{Agent-42=1}	
4	1	1	Agent-42	11	[11]	[26]	[40, 25, 11, 26]	false	0.934	true	{Agent-43=1}	
5	1	1	Agent-43	11	[26]	[41]	[40, 25, 11, 26, 41]	false	0.799	true	{Agent-44=1}	
6	1	1	Agent-44	11	[41]	[43]	[40, 25, 11, 26, 41, 43]	true	0.051	false	{Agent-45=1}	
7	1	1	Agent-45	11	[43]	[51]	[40, 25, 11, 26, 41, 43, 51]	false	0.947	true	{}	
8	1	2	Agent-15	4	[¬14]	[4, ¬14]	[4, ¬14]	true	0.459	false	{Agent-16=1, Agent-53=	
9	1	2	Agent-16	4	[4]	[19]	[4, ¬14, 19]	false	0.598	true	{Agent-17=1, Agent-53=	
10	1	2	Agent-53	14	[¬14]	[29]	[4, ¬14, 19, 29]	false	0.595	true	{Agent-17=2, Agent-54=	
11	1	2	Agent-17	4	[19]	[34]	[4, ¬14, 19, 29, 34]	true	0.338	false	{Agent-54=2}	
12	1	2	Agent-54	14	[29]	[45]	[4, ¬14, 19, 29, 34, 45]	false	0.609	true	{Agent-52=1}	
13	1	2	Agent-52	14	[45]	[¬14]	[4, ¬14, 19, 29, 34, 45, ¬14]	false	0.666	true	{}	
14	1	3	Agent-42	11	[11]	[26, 11]	[26, 11]	true	0.934	true	{Agent-43=1}	Agent-43=(1/1)
15	1	3	Agent-43	11	[26]	[41]	[26, 11, 41]	true	0.799	true	{Agent-44=1}	Agent-44=(1/-1
16	1	4	Agent-37	9	[39, 12]	[55, 39, 12	[55, 39, 12]	true	0.965	true	{Agent-47=1}	

Figure 5.3
Results document showing each agent interaction during the simulation.

```
        Settings.usePreferredAgent = false;
        Settings.ignoreBadAgents = false;
        Settings.letAllAgentsRun = true;
        Settings.selectAgent = "None";
    }

    @Override
    public void round_3() // Series weight with pref agent baseline
    {
        Settings.selectedVotingSystem = "Series Weighting";
        Settings.usePreferredAgent = true;
        Settings.ignoreBadAgents = false;
        Settings.letAllAgentsRun = true;
        Settings.selectAgent = "None";
    }
```

As output, the simulator produces an Excel spreadsheet for each round containing the details of every agent action as well as their internal states (See Figure 5.3), the details of each iteration (i.e., for each attack scenario, see Figure 5.4), and for each run (i.e., the runtime history of all agents within the current simulation scenario). Additional information about the statistics of the whole simulation including a comparison table for each of the rounds is included as well as the current configuration of all system settings and the domains environment layout. Using the results document, a step-by-step history and performance report for each agent is made available to verify the validity of the tests and conclusions.

NetworkFlow. For the purposes of testing the Generalised Anomaly Detection Algorithm, the simulator was extended to work with the appropriate flow-layer datasets in CSV and XML format. The code format is structurally similar to that described in Chapter 4.4 with the

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Init Agent	Run	Iteration	Report Length	Group GD	Is Correct GD	CM	Is Correct GD Bin	Accuracy	Avg Analysis	True	False	True	False	GD	Sens
2	Agent-40	1	1	6	false	true	TN	1	0.667	0.713	0	2	4	0	1.0	0
3	Agent-15	1	2	6	false	true	TN	1	0.667	0.544	0	2	4	0	1.0	0
4	Agent-42	1	3	2	true	true	TP	1	1	0.866	2	0	0	0	1.0	1
5	Agent-37	1	4	5	false	false	FN	0	0.8	0.592	4	0	0	1	1.0	0.8
6	Agent-38	1	5	8	false	true	TN	1	0.625	0.75	0	3	5	0	1.0	0
7	Agent-35	1	6	2	false	true	TN	1	1	0.563	0	0	2	0	1.0	0
8	Agent-56	1	7	5	true	true	TP	1	0.8	0.537	4	0	0	1	1.0	0.8
9	Agent-27	1	8	12	true	false	FP	0	0.5	0.574	0	6	6	0	1.0	0
10	Agent-46	1	9	4	true	true	TP	1	0.75	0.499	3	0	0	1	1.0	0.75

Figure 5.4
Results document showing logging at the interaction level.

whole system composed of four methods and several utility functions called on as needed. The connection diagrams produced for Figure 4.1 produced following the full processing of the dataset and subsequently updated during the model processing phase where the connection information is used to discern the malicious nodes. The connections graph is interactive providing detailed information about the internal state of the node during the model processing, an example output for a particular node is as follows:

```
Name: 192.168.1.101
Neighbours: (9) [192.168.1.1, 192.168.1.2, 192.168.1.103, 255.255.255.255, 192.168.1.104,
    192.168.1.105, 192.168.1.255, 192.168.1.102, 192.168.5.122]
Total Sent: 2348 Total Received: 1722
Total Malicious: 2 Total Safe: 2346

Neighbour 192.168.1.1: CountReceived: 2, CountSent: 0, %Diff: 200.0, %Total Recieved: 0.0
Neighbour 192.168.1.2: CountReceived: 4, CountSent: 0, %Diff: 200.0, %Total Recieved: 0.0
Neighbour 192.168.1.103: CountReceived: 372, CountSent: 404, %Diff: 8.24742268041237,
    %Total Recieved: 0.0
Neighbour 255.255.255.255: CountReceived: 2, CountSent: 0, %Diff: 200.0, %Total Recieved:
    0.0
Neighbour 192.168.1.104: CountReceived: 40, CountSent: 40, %Diff: 0.0, %Total Recieved:
    0.0
Neighbour 192.168.1.105: CountReceived: 40, CountSent: 1188, %Diff: 186.97068403908793,
    %Total Recieved: 0.0
Neighbour 192.168.1.255: CountReceived: 576, CountSent: 0, %Diff: 200.0, %Total Recieved:
    0.0
Neighbour 192.168.1.102: CountReceived: 90, CountSent: 90, %Diff: 0.0, %Total Recieved:
    0.0
Neighbour 192.168.5.122: CountReceived: 1222, CountSent: 0, %Diff: 200.0, %Total
    Recieved: 0.0
Similar Neighbours: []
Removed Edge Count: 0
```

```
[**] [129:12:1] Consecutive TCP small segments exceeding threshold [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
06/13-04:03:22.760230 192.168.5.122:143 -> 192.168.2.111:4480
TCP TTL:63 TOS:0x0 ID:2788 IpLen:20 DgmLen:86 DF
***AP**F Seq: 0x94E12B3C  Ack: 0xD6C6FB79  Win: 0x1920  TcpLen: 20

[**] [129:12:1] Consecutive TCP small segments exceeding threshold [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
06/13-04:04:26.262649 192.168.5.122:110 -> 192.168.4.120:3979
TCP TTL:63 TOS:0x0 ID:32542 IpLen:20 DgmLen:60 DF
***AP*** Seq: 0xCDFA3E00  Ack: 0x38D8FCEE  Win: 0x16D0  TcpLen: 20

[**] [129:12:1] Consecutive TCP small segments exceeding threshold [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
06/13-04:11:35.733579 192.168.5.122:21 -> 192.168.2.107:1178
TCP TTL:63 TOS:0x0 ID:32863 IpLen:20 DgmLen:60 DF
***AP*** Seq: 0x607A68B7  Ack: 0x9C2F1C3B  Win: 0x16D0  TcpLen: 20
```

Figure 5.5
Snort log output file containing the classification of the UNB ISCX dataset.

```
Normal Connections: 2346, Malicious: 2, No. Nodes: 1
```

The log contains the local statistical information about each node used during the processing of the dataset. The information such as *CountSend* and *%TotalReceived* is directly used by the proposed algorithm to calculate the likelihood of a node being used for malicious activity.

SnortAlertParser. The simulator is also designed to work with both Bro IDS and Snort IDS datasets. Chapter 5.2.1 contains a case study using a dataset first analysed Snort and then within the agent-based simulator to find additional nodes that were not detected by the IDS. The simulator can take in as input both a labelled dataset and the output of a Snort IDS classification in the format shown in Figure 5.5.

5.2 DMASS Simulation Evaluation

To evaluate the DMASS 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 3.9) is generated, and 100 simulated security events consisting of both attacks and false alarms are initiated to assess the agent's performance. The purpose of these tests is to show that MAS can provide a useful mechanism for detecting attacks and to minimise the overall amount of work done in terms of data collection to reduce the problems associated with bulk analysis.

Table 5.1 shows a comparison of the DR (Detection Rate) and FAR (False Alarm Rate) of the four algorithms during the individual evidence evaluation stage with a 20% DR improvement

Table 5.1
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

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 can be judged in terms of both the individual performance of the agent (Table 5.1) and the final classification made by the group (Figure 5.7).

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

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

This DR improvement can be attributed to the corrective measures introduced by the 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, this is by an acceptable amount given the current industry standards [9] 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 5.6 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

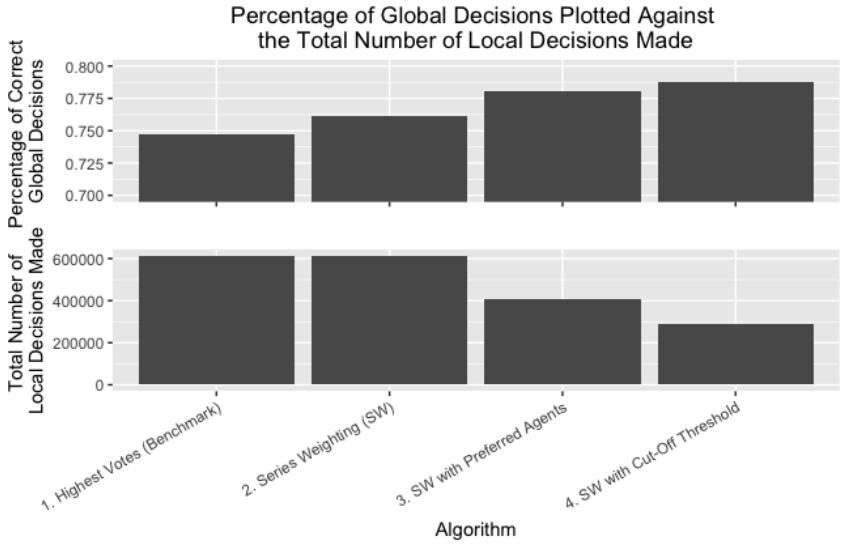

Figure 5.6
The total number of local decisions (i.e., cost associated actions) using each of the four algorithms described in Table 5.1 plotted against the percentage of correct global decisions.

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 5.7 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 FAR. 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.

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 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

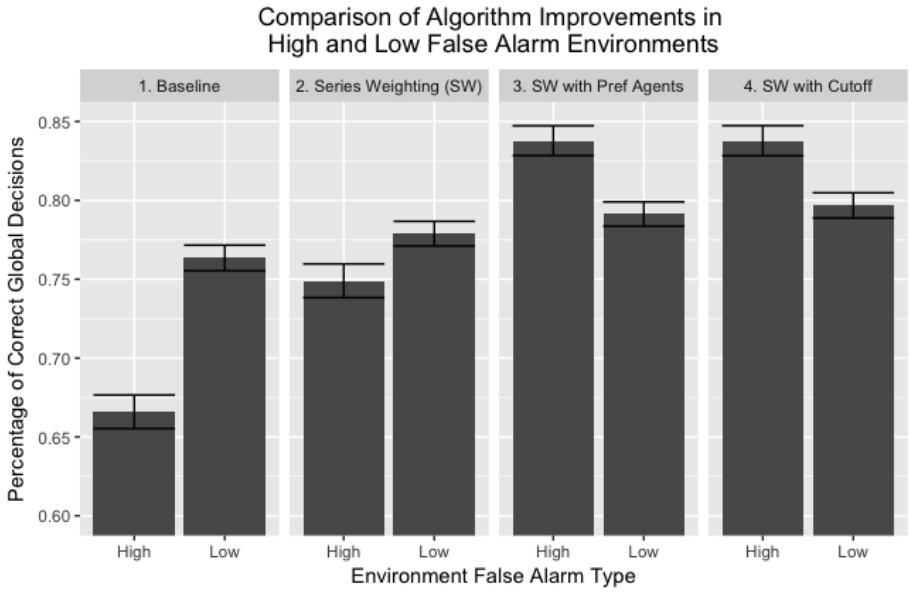

Figure 5.7
A comparison of the 4 algorithms featured in Table 5.1 in both a low and high false alarm environment (95% confidence intervals).

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 5.8 and 5.9 show the change in the 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 the number of agents) is available, the more reliable and highly performing agents are selected for participation. The performance gain of adding 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 of adding more agents to the network, the number of messages sent between agents for participation increases with each additional agent. The attack detectability measure which represents the stealthiness of the

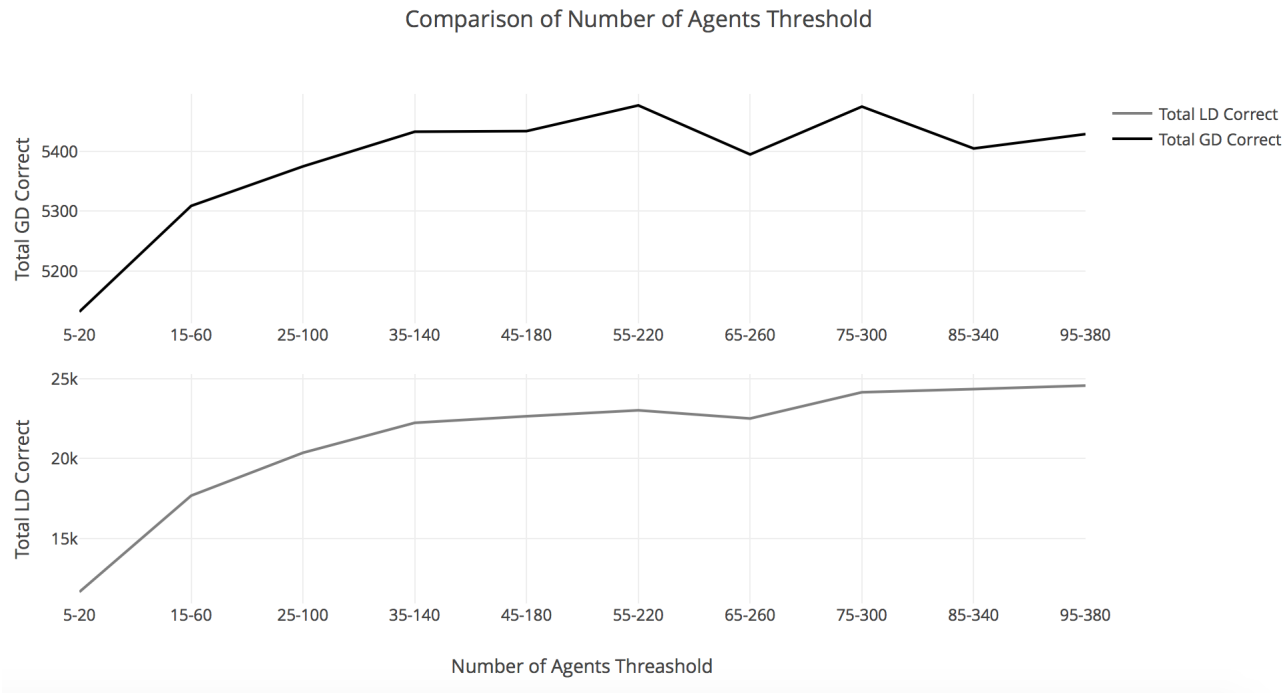

Figure 5.8
Comparison of Number of Agents (No. Agents) Threshold.

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.

5.2.1 Snort IDS Evaluation

To further show the need for the DMASS and demonstrate its performance benefits, an evaluation using the Snort IDS [136] was performed on the UNB ISCX Intrusion Detection Evaluation Dataset [144] 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 signatures and then processed by the agents to find the remaining connection flows that were not detected. Figure 5.10 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 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

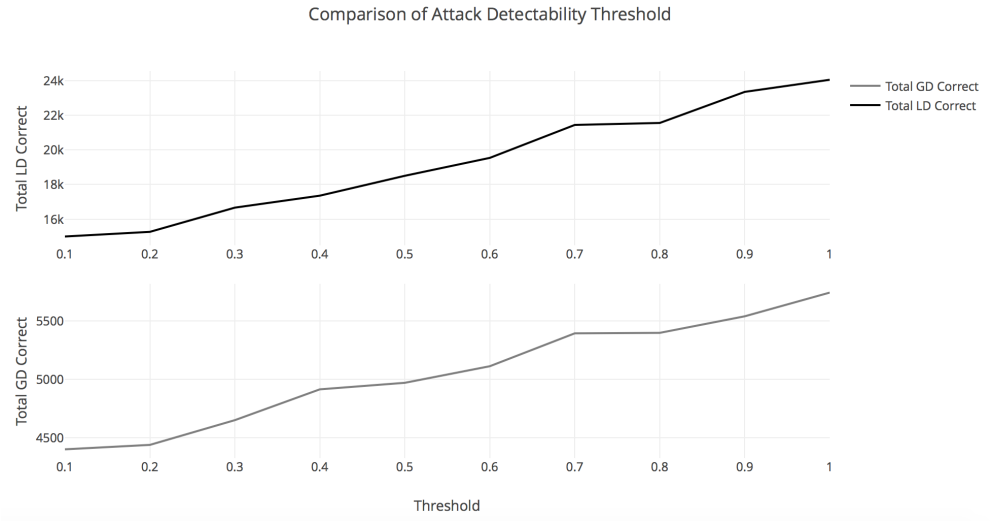

Figure 5.9
Comparison of Attack Detectability ($Av_{detectability}$ threshold).

the role as the data gathering/analysis module by using its signatures to classify the 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 attempting to scan the network, in particular where the attacker scanned an IP address that was not in use. Typically, this behaviour is considered non-malicious by most IDSs as its cause often occurs normally within local networks as a result of typical probing and non-malicious networking problems. Figure 5.11 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 an 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 3.2) 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

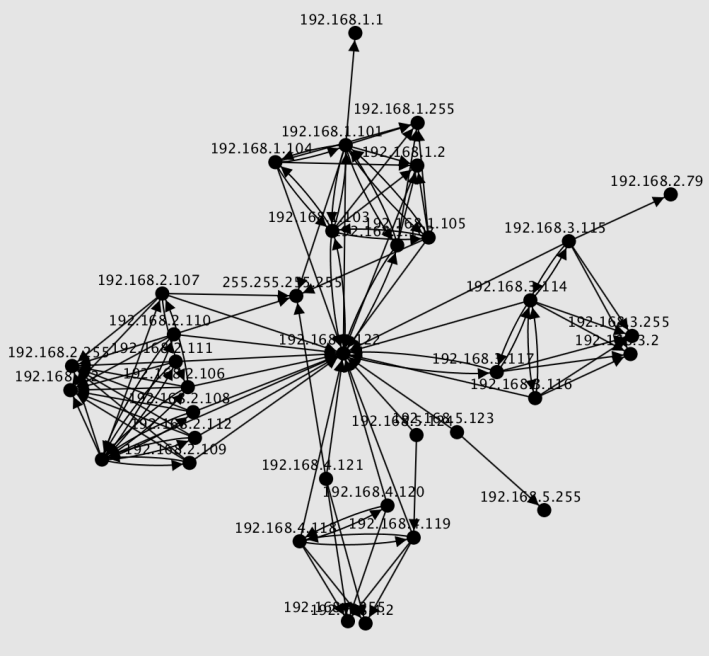

Figure 5.10
An example of the USB ISCX dataset network.

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 the 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.

5.2.2 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 being processed. This is functionally similar to the system benchmark shown in Figure 5.6 that has a large number of local decisions must be processed to analyse the global decision. The algorithms improve on this by requiring less local decisions (i.e., pieces of information) to be analysed for a similar overall result. In particular, the Series Weighting with Cut-Off Threshold algorithm 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 while communication is kept as lightweight as possible by separating the system functions. The function

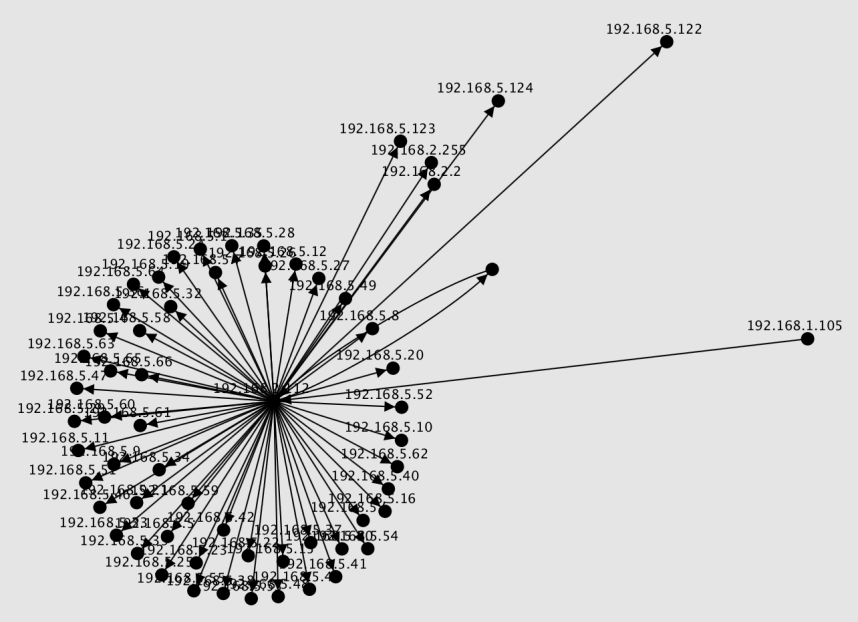

Figure 5.11
An example of probe activity not detected by snort IDS.

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 5.6 that show the number of local decisions required is substantially reduced is in proportion with the number of network messages that are sent as every unique local decision requires the global report to be sent.

5.2.3 Existing Approaches

Given that the DMASS 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 domain exploration 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. The results in Table 5.1 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. Traditional systems that attempt to match as many signatures as possible result in worse performance than the extended data collection approach because irrelevant classifications can dilute the final event analysis.

5.2.4 Model Vulnerabilities

The model provides a robust solution to encompassing many detection technologies for use by an automated agent-based forensic architecture. 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 MASs, the communications overhead is often computationally expensive due to the decentralised nature of the system. Figure 5.6 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 5.7 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.

5.3 Generalised Anomaly Detection Evaluation

In this section, an evaluation of the generalised anomaly detection algorithm is given in comparison to existing techniques from the literature and commonly applied machine learning algorithms. When comparing the algorithm against both supervised and unsupervised approaches to show the effect of overfitting in both categories. Three types of evaluations are performed: (i) *multi-dataset supervised evaluation* tests the performance of supervised algorithms in training

on one dataset (using 10-fold cross-validation) and testing on another dataset, (ii) *single-dataset unsupervised evaluation* tests the performance of unsupervised clustering algorithms using a single dataset since the strength of the unsupervised approach is that no offline training is required, and (iii) a comparison is made to algorithms from the literature that use the same dataset. The algorithm is evaluated using two separate datasets to show that it can be applied to different networks without suffering from implementation overfitting.

5.3.1 Datasets for Generalised Anomaly Detection

There has been a historical problem of obtaining suitable datasets for cybersecurity with most organisations preferring to keep their data private due to fears of unintended information leakage. With the general lack of good data available, research is often validated against outdated datasets such as the DARPA 1999 dataset [116] which contain known structural problems and may not be representative of the modern network due to its age. Furthermore, the type of data used as the basis for detection will govern the systems real-time deployability and scope of detection. As the amount of preprocessing required increases, the suitability for real-time detection decreases, and as the scope of the dataset narrows, so does the ability to detect a broad range of attacks. To fully capture a broad range of attacks, the data must be wide in scope justifying the choice to use flow-level data collected directly from the wire. Table 5.2 lists several commonly used datasets with a discussion on the type of data as well as common criticisms.

5.3.2 Performance Evaluation

To evaluate the algorithm's performance, the UNB ISCX dataset is analysed by the generalised algorithm to build a network model and compared against several alternative techniques. As previously stated, the algorithm requires that only L2L data be used within the model, as such, non-local data was disregarded from the evaluation. Direct comparisons between this algorithm and others that analyse the whole dataset are made, however, the comparison is fair since 19,610 of the 20,358 flows labelled as "attack" present in the UNB ISCX dataset exist within the L2L portion of the data[1]. The metrics used to compare the generalised algorithm with existing systems and machine learning algorithms are listed below with the detection rate and false alarm rate being the most important measures from a security point of view.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

[1]For the day June 13th, 2010 (day 3) containing a high number of internal network attacks.

Table 5.2
Comparison of commonly used datasets.

Dataset	Type of Data	Discussion
DARPA 1999 [116]	Simulated Tcpdump data from inside and outside of a medium-sized U.S. air force base.	There are many criticisms [144, 111, 33] concerning the reliability of the dataset including the TTL† values for the packets and the baseline performance when tested using a signature-based IDS and age of the dataset.
KDD CUP 1999 [151]	An improved version of the DARPA 1999 dataset.	Many problems with the DARPA 1999 dataset persist, however, the dataset was cleaned to make both the training and testing sets more realistic.
UNB ISCX Intrusion Detection Evaluation Dataset 2012 [144]	Simulated labelled Tcpdump data.	Simulated data extracted from real user logs. Attacks include a diverse range of multi-stage attacks originating from both inside and outside of the local network‡.
UNSW-NB15 Network Data Set 2015 [20, 117]	Simulated labelled Tcpdump data.	A large synthetic-hybrid dataset collected by the Cyber Range Lab of the ACCS containing nine types of attacks: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms.

† IP Time To Live field.
‡ 8 categories of attacks including: fuzzers, reconnaissance, shellcode, analysis, Denial of Service, backdoors, exploits and worms.

$$Detection\ Rate = \frac{TP}{TP + FN}$$

$$False\ Alarm\ Rate = \frac{FP}{FP + TN}$$

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

$$Matthews\ Correlation\ Coefficient =$$

$$\frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}}$$

Table 5.3 lists the performance evaluation for the generalised algorithm when tested with the UNB ISCX dataset (day 3). Given the high-level nature of the dataset containing many types of attacks, the system is effective in classifying attacks to an acceptable degree. The high specificity of the system (90.11%) shows the accuracy of the model in iteratively grouping together connection flows to explain their presence within the network. Furthermore, since the algorithm is presented as a classifier for use within a wider security context (e.g., to tune an

Table 5.3
The detection performance metrics for the generalised anomaly detection algorithm using the UNB ISCX dataset (day 3).

Metric	**Value**
Accuracy	87.06%
Detection Rate	81.03%
False Alarm Rate	9.89%
Specificity	90.11%
MCC	71.05%

Table 5.4
The detection performance metrics for the generalised anomaly detection algorithm using the UNSW-NB15 dataset.

Metric	**Value**
Accuracy	94.59%
Detection Rate	88.80%
False Alarm Rate	2.51%
Specificity	97.65%
MCC	87.74%

IDS), it is preferable to have an increased number of false positives which can later be further investigated and verified, rather than a high number of false negatives which would not be further investigated.

To validate the generalisability of the algorithm it was also tested on the UNSW-NB15 [20, 117] dataset containing a variety of attacks captured at the network flow layer. By testing the model on a different dataset containing different attacks, the algorithm is shown to be scalable in different networks while not producing over-fit results for a particular attack implementation or dataset. Table 5.4 shows a similar results distribution with a low false alarm rate and high system specificity. Overall, the algorithm performed similarly on both datasets providing an effective way to detect attacks across a variety of network topologies.

5.3.3 Machine Learning Comparison

Comparing the results with alternative techniques is challenging in the absence of a general framework in addition to the variety of datasets (refer to Table 5.2) used throughout the literature, some of which are recognised as being unsuitable for making general classifications and

Table 5.5
Performance comparison of related works and machine learning algorithms.

Author	**Dataset**	**Detected Attacks**	**Detection Rate**	**False Alarm Rate**
Generalised algorithm[a]	UNB ISCX	Local Network Infiltration	81.03%	9.89%
Generalised algorithm[b]	UNSW-NB15	DoS, Fuzzers, Exploits, Recon, Shellcode, Worms	88.80%	2.51%
Heidarian et al. [69]	UNB ISCX	DDoS Only[c]	89.25%	8.60%
Wang et al. [162]	UNB ISCX	DDoS Only	74.02% - 89.30%[d]	10.70% - 25.98%[d]
Pajouh et al. [126]	NSL-KDD	Probe, DoS, U2R, R2L	82.0% - 83.24%	5.43% - 4.83%
Pervez et al. [131]	NSL-KDD	Probe, DoS, U2R, R2L	82%	15%
Tama et al. [150]	NSL-KDD	Probe DoS, U2R, R2L	91.82% - 99.85%	4.19% - 0.27%
Tama et al. [150]	UNSW-NB15	DoS, Fuzzers, Exploits, Recon, Shellcode, Worms	91.31%	8.60%
Abdalla et al. [2]	Private	Local Network Infiltration	74.0%	40%
Jiang et al. [81]	Abilene network traffic	DDoS & Mixed Attacks	87%	40%
Lakhina et al. [97]	Abilene network traffic	Worms	80%	10%
Papamartzivanos et al. [128]	UNB ISCX	Local Network Infiltration	63.76%	2.61%
C.45 Decision Tree[e]	UNB ISCX	Local Network Infiltration	43%	51%
RBF Neural Network[f]	UNB ISCX	Local Network Infiltration	52%	76%
Simple K-Means[g]	UNB ISCX	Local Network Infiltration	40%	28%
Expectation Maximisation[h]	UNB ISCX	Local Network Infiltration	50%	70%

a Accuracy: 87.06%, Specificity: 90.11%, MCC: 71.05%, dataset day 3.
b Accuracy: 94.59%, Specificity: 97.65%, MCC: 87.74%.
c Evaluated using the dataset for June 13th, 2010 (day 3).
d Results depend on the model configuration.
e Confidence factor: 0.25, Number of folds: 3.
f Minimum standard deviation: 0.1, Number of clusters: 2.
g Distance function: Euclidean, Maximum iterations: 500, Number of clusters: 2.
h Maximum iterations: 100, Minimum standard deviation: 1.0E6.

are considered to be error-prone. The commonly used DARPA 1999 dataset has been found by many authors to be inaccurate in representing the typical network environment and suffering from statistical problems arising from the synthetic construction of the data [102, 25, 33]. For this reason, the UNB ISCX and UNSW-NB15 datasets were used as alternatives.

To highlight the aforementioned problems of implementation overfitting, several machine learning algorithms were trained and tested using two datasets similarly to the previous experiment. To benchmark the generalised anomaly detection algorithm, four well-known and widely deployed machine learning algorithms were trained on the UNB ISCX data and tested on the UNSW-NB15 dataset to show the performance reduction that occurs when attempting to generalise between networks. Both datasets were cleaned and normalised to produce nine matching features commonly used to evaluate machine learning algorithms: Application Name (nominal),

total source bytes (numeric), total destination bytes (numeric), total source packets (numeric), total destination packets (numeric), source port (nominal), destination port (nominal), protocol name (nominal) and the class value (attack/normal). The algorithms were trained using 10-fold cross-validation and compared against the known class values for each flow. Specifically, the four algorithms were chosen for their widespread use in the literature and to test the dataset using different approaches (i.e., decision tree, neural network and unsupervised clustering). Table 5.5 lists the detection rate and false alarm rate for both the supervised and unsupervised learning algorithms tested. While some approaches detect multiple attacks, others that only detect one type will often perform better in terms of detecting the attack, however, for the purposes of detecting a variety of attacks, the proposed algorithm performs better overall.

The original dataset of 19 features (See Table 4.1) and the class was reduced to 11 through feature selection based on a literature review and testing of several features. The removed features fall into three categories: (i) string-based features, (ii) device IDs, and (iii) temporal features. The string-based feature (i.e., the payload) cannot be easily modelled through machine learning due to its complex format and the prominence of encryption resulting in unique values for identical payloads. Device IDs (i.e., IP addresses) are unsuitable for local network anomaly detection as the IP addresses used within these features are re-used between networks (and datasets) since they are within the local-network IP address range, as such any model relying on a particular IP address for classification will classify future instances incorrectly. Finally, temporal features (i.e., startDateTime and stopDateTime) are dependant on the time the attack was initiated, the length of the attack and any evasive techniques used (e.g., slowing a port scan down to avoid detection). Table 5.6 lists the removed features with a comparison of the gain ratio rank scores showing the predictability of each feature and discussion on why it was removed from the data; these features may be useful for other approaches to detecting attacks (e.g., the start and stop time may be useful for temporal-modelling), however, they are deemed inappropriate for broad-based APTdetection for the reasons noted in the table.

Further tests were performed to compare the performance comparison of the removed features, however, through feature selection performed within the selected algorithms (e.g., C4.5 leaf pruning), the identified features were typically removed early. During a comparative evaluation of the datasets with and without the source and destination IP addresses using a C.45 decision tree, the IP address features did not appear in either tree having been pruned by the algorithm. Both algorithms performed equally within a 1% margin of error due to the IP address features being removed. The results of various algorithm comparisons are included below.

Table 5.6
Comparison of Removed Features.

Feature	Grain Ratio Rank Score	Reason for Removal
Destination IP	0.175	While the target of an attack is an important feature, within the local network where IP addresses are allocated within the private IP range (i.e. 192.168.0.0 to 192.168.255.255) any model built using this feature could not be applied for future attacks.
Source IP	0.130	Like destination IPs, APTs launched from the local network use IP addresses within the private range.
StartDateTime	0.062	Poor performance and APTs are not modelled well temporally.
StopDateTime	0.062	Poor performance and APTs are not modelled well temporally.
SourcePayload-AsBase64	0.0004	Poor performance and prevalence of payload encryption.
DestinationPayload-AsBase64	0.0003	Poor performance and prevalence of payload encryption.
DestinationPayload-AsBaseUTF	0.0002	Poor performance, prevalence of payload encryption and similar to Base64 representation.
SourcePayload-AsBaseUTF	0.0001	Poor performance, prevalence of payload encryption and similar to Base64 representation.

C4.5 Decision Tree The C4.5 algorithm [171] is a statistic classifier for building decision trees. When trained on one dataset and tested on another, the algorithm correctly identified 49% of instances with a FAR of 0.51, a DR of 0.43 and a ROC Area of 0.459.

Radial Basis Function Network Radial Basis Function (RBF) [32] is a neural network chosen for its resistance to high-dimensional noisy data and efficiency during the training phase[2]. When trained on one dataset and tested on another, the algorithm correctly identified 52% of instances with a FAR of 0.76, a DR of 0.52 and a ROC Area of 0.355.

Simple K-Means Clustering Simple K-Means is a simple clustering algorithm that performs well on noisy high-dimensional data. When tested on the UNB ISCX dataset using a classes-to-

[2]Note that 10-fold cross validation could not be used during the training phase due to the exponential increase in the time required caused by the amount of nominal variables.

clusters evaluation[3] the algorithm correctly identified 73% of the instances with a FAR of 0.28 and a DR of 0.40.

Expectation Maximisation Clustering Expectation Maximisation clustering performs cross-validation to determine the number of classes. When tested on the UNB ISCX dataset using a classes-to-clusters evaluation the algorithm correctly identified 60% of the features with a FAR of 0.70 and a DR of 0.50.

From these results, the problem of overfitting is highlighted by showing the performance reduction when evaluating the supervised algorithms on different training and test datasets. Often within the literature, an algorithm is trained and tested on different portions of the same dataset which results in an artificially high classification result (upwards of 98%) because the attack footprint will be static for the particular attack. When tested on two datasets, the performance reduction is explained by the variety of attacks, the dynamic nature of the attack footprint, and the adversarial control over the flow features, resulting in a model that is not truly representative of the wider attack but are overfitted to the specific implementation of the attack. The results show that unsupervised clustering algorithms are more suited to this environment as there is no requirement to perform prior offline training allowing the algorithms to remain sensitive to the network's normal traffic patterns. These results demonstrate the advantages of focusing on detecting the abstract behavioural differences between the connections and endpoints rather than focusing on a particular feature that would identify general malicious behaviour.

5.3.4 Dataset Comparison & Challenges

Within the literature authors often use datasets to detect specific types of attacks rather than multiple attacks. Heidarian et al. [69] use Support Vector Machines to achieve 89.25% accuracy in detecting denial of service attacks from the UNB ISCX dataset. Algorithms with a narrow scope often outperform broader models that attempt to detect a variety of attacks because the variability of the data is lower. However, many authors do not perform an evaluation to test for attack overfitting as described in Section 5.3.3 by testing the model on multiple datasets. Table 5.5 shows the performance of current models from the literature that attempt to detect attacks in the UNB ISCX dataset. The Generalised Anomaly Detection Algorithms achieves a similar detection and false alarm rate in classifying a wider variety of attacks (e.g., DoS, fuzzers, system exploits, reconnaissance, shellcode and worms) than the listed approaches do in detecting a single attack. Furthermore, the results are validated on a secondary dataset (UNSW-NB15) to

[3]The model is first built using the available features and then evaluated by comparing the distribution of the two classes over the clusters.

show that the model is not overfitted to any particular dataset but can be applied to networks of different architectures and containing different attacks.

Data balancing is an important consideration for the training and testing of ML algorithms requiring a dataset containing a similar number of both malicious and innocuous samples. While in a live environment, the ratio of malicious-to-innocuous will often be unbalanced with the volume of innocuous being far greater, especially in the case of stealthy attacks where the objective is to remain hidden and undetected. While the datasets used are more balanced than would be typically found in live environments to improve the quality of the ML model, supervised learning techniques require re-training in live environments which can be challenging given the lack of labelled network security datasets that are properly balanced and representative. Furthermore, the approach to algorithm evaluation proposed in this thesis is to make use of multiple datasets for the training and testing stages. Given the reality that the footprint of a given attack can change between attacks and that there are multiple ways to achieve any particular goal, building a model based on the data from one source often leads of incorrect classification of future instances.

5.3.5 Generalised Anomaly Detection for DMASS

The generalised anomaly detection algorithm is designed to be of particular use to the DMASS. The two systems can be viewed as addressing different aspects of the cybersecurity problem, (i) the generalised anomaly detection algorithm works towards detecting the initial traces of an attack, but it cannot detect the specifics of the attack, whereas (ii) the DMASS system can perform in-depth investigations into a network, but it relies on signatures of existing approaches to begin the extended data collection process. Both systems are mutually beneficial to each other with the generalised algorithm providing the trigger for the DMASS to perform the in-depth investigation.

Together the systems provide an automated approach to detecting and investigating network threats with a focus on detecting advanced stealthy attacks that employ evasive techniques to bypass IDS triggers.

5.4 Summary

In this chapter, the results of the DMASS and generalised anomaly detection algorithm were provided as well as a description of the simulator in which they are developed. The simulator contains features to enable the fair comparison of different algorithms under the same environmental conditions. Through the use of system profiles, the same groups of agents can be

compared using various algorithms to produce a more accurate comparison of the detailed agent-based features. Additionally, in-depth logging of agent actions are produced for interpreting the results produced by the simulator. The DMASS produced a detection rate of 80-85% under high false alarm environments in comparison to the baseline approach that is more closely tied to the performance of signatures. In addition to the detection rate improvement, the DMASS increased efficiency by 50% by introducing agent preference and threshold cut-off to avoid brute-force analysis. The generalised anomaly detection algorithm produced an average detection rate of 85% while detecting a wide variety of network attacks. The algorithm is unsupervised and so can be applied directly to the network without a prolonged training period making it more suitable for real-time detection. Furthermore, by focusing on the anomalous lateral movements made by the attacker, the algorithm is more adaptive to future variations of attacks that may be implemented with different feature-values.

Chapter 6

Conclusion & Future Work

When cyber attacks are successful, they can result in widespread disruption of services, monetary costs and the compromise of user privacy. Generally, the longer a cyber attack persists, the most costly it becomes; as a result, security systems must prioritise the timely detection and removal of threats that penetrate the perimeter of the network.

Persistent threats that can remain undetected and have long-lasting ramifications have arisen as a challenging issue for modern network security. With current detection technologies mostly relying on manually defined signatures to describe known malicious behaviours, unseen and stealthy attacks without a corresponding signature can effectively bypass security undetected. In recent years, examples of high-profile advanced persistent threats such as Stuxnet, Duqu, Flame and Red October have been studied as examples of persistent threats that are well sponsored, researched and targeted. While the static signature-based solution has proven unable to adapt to these threats, new approaches using machine learning and adaptive multi-agent systems have shown promising results in addressing the new generation of attacks.

The evidence from the research suggests that the current generation of monolithic intrusion detection systems that effectively brute-force monitor the network environment for rule violations cannot adequately protect systems from persistent threats that make use of evasion technologies. The first significant contribution was the application of multi-agent systems to the problem of advanced persistent threat detection, which was shown to be a viable and adaptive solution for dynamically collecting evidence, knowledge representation and reducing the number of false results. The methodology presented for use within the Decentralised Multi-Agent Security System allows for the automatic operation of agents to collect and analyse evidence from around the network. Mechanisms for guiding the agents to reliable sources of information and avoiding the inclusion of poorly performing agents were developed to avoid the disadvantages of the current generation of static intrusion detection technologies. Through the development

of distributed algorithms for use within the proposed multi-agent systems, the results show that the amount of work done (regarding processing signatures) can be reduced by using the multi-agent approach. Given the complexity of the network environment with great variety in terms of technologies and protocols, the targeted multi-agent approach resulted in an increase in accuracy while significantly reducing the amount of processing required. This result is credited to the agent-based mechanisms such as agent-preference and the ability of agents to analyse the current effectiveness of the evidence gathering investigation.

Machine learning has arisen as a promising approach to detecting malicious behaviours within cybersecurity. Anomaly detection is based on the idea that there exists distinguishable behaviour between malicious and legitimate users, and through modelling user behaviour, these differences can be detected. One of the main issues in the application of machine learning technologies to the live environment is overfitting. When a model learns the training data too well, it can negatively impact the performance of the algorithm resulting in solutions that are tied to a particular implantation of an attack, rather than to the category of attacks it is trained to attack. The second major contribution of this thesis is the development of a broad-based algorithm for use at the network-flow layer to detect the initial traces of persistent threats. While the goal of the advanced persistent threat is to remain hidden from detection, certain actions are necessary for the development of the attack. In particular, lateral movement throughout the network cannot be avoided and cannot be easily hidden from detection. While separating the malicious and legitimate movements at this layer requires an additional system, such as the proposed Decentralised Multi-Agent Security Systems, the information gathered at this layer can be used to indicate the broad location of malicious users. With the goal of avoiding implementation overfitting, the proposed algorithm aims to detect a variety of attacks to identify hosts for further investigation.

6.1 Summary of Novel Contributions

The research in this thesis presents the application of multi-agent systems to the problem of automated evidence collection and processing as well as the detection of stealthy persistent threats that cannot be easily detected by rule-based systems.

The main novel contributions of this thesis are as follows:

- **Decentralised Multi-Agent Security System**. The development of the decentralised multi-agent security system aims to improve the way in which detection is done. The monolithic approach of monitoring all network communications and making a final deci-

sion based on a comparison against a signature is static and does not take into account situational information about the order in which evidence is found or how it relates to other pieces of evidence. The developed architecture separates individual agent evidence analysis and the global classification stage to allow for the inclusion of additional mechanisms such as agent performance a dynamic system for deciding when the evidence collection stage should be ended and the classification made.

- **Development of the Domains Model**. The system of domains was developed to work with the concept of conditions & effects, core to the operation of the decentralised multi-agent security system. The model recognises that technologies found within local networks relate to each other in different ways and this is used by the attackers to perform lateral movement around the network. The series weighting algorithms were developed to make use of this environmental fact to judge the quality of the information found by the agents. For example, where collected evidence originated from around the network in a sporadic pattern, the likelihood of it being a false positive result can be concluded, based on the system of domains.

- **Analysis of Overfitting in Cyber Security**. In security research, the topic of overfitting often goes undiscussed despite the environment and datasets being extremely prone to its associated problems of producing inaccurate models that fail to adapt to new environments. Unlike other fields of study, attackers, particularly advanced persistent threats have the goal to remain hidden and deceive those attempting to detect them. Additionally, attackers can directly affect the features found within the datasets used for training and testing machine learning algorithms since the data features (such as packet sizes, destination and payload) all originate from either a user or attacker. The comparison shows that machine learning algorithms often heavily bias towards the particular dataset and the shape that the attack takes rather than the general pattern of the attack.

- **Generalised Anomaly Detection**. As an improvement over the current machine learning approaches that often overfit to the attacker, the generalised anomaly detection algorithm was presented to detect a wide variety of attacks by studying the lateral movement of attackers through the network. The algorithm specifically avoids using features of the dataset and instead remodels the iterations between groups of users to find outliers for further investigation. The results show that the algorithm can perform well in different environments by testing it against different data sets containing a variety of attacks. The disadvantage of the developed algorithm is that it cannot identify the type of attack, but

only the location of the attack for further in-depth investigation. As such, the generalised anomaly detection algorithm and the decentralised multi-agent security system work well together to detect the initial traces of an attack and then investigate it further.

Together, the DMASS and generalised anomaly detection algorithm address the weakness inherent in the other system to provide a comprehensive solution to detect and respond to cyber attacks. The advantages of the generalised anomaly detection algorithm is its ability to (1) detect a wide range of attacks, and (2) operate on the network layer, while its weaknesses are (1) the output of the algorithm is a device address that appears to be acting anomalously, not a comprehensive analysis of the event. The advantages of the DMASS are (1) its ability to perform in-depth investigations into events, (2) its ability to proactively gather relevant information, and (3) its ability to work on-demand rather than as a brute-force detection system, while its weaknesses are (1) the on-demand approach requires an external indication to begin the investigation. When used together, the system addresses both the problem of detecting events from a high-layer (i.e., from network flows) and investigating the event further through the agent-based DMASS architecture.

6.2 Future Work

Future work would focus on the increased integration of the proposed systems with current network detection technologies, the continued application of the generalised algorithms to new environments and the use of Multi-Agent Systems to autonomously address cyber security threats.

- In particular, moving the DMASS from the developed simulator into a live environment would require the re-working of current IDS technologies to operate as a MAS, several steps have been taken to adapt the Bro IDS for this purpose. Due to the complex nature of IDSs with many variable parts, the full development of a decentralised IDS was placed outside of the scope of this research and testing to prove the individual algorithms was performed instead. In addition, future work researching the automatic tailoring of IDS signature sensitivity (i.e., the degree to which a piece of evidence is deemed to match the signature) could be undertaken to increase automation of the system as a whole. The exact degree to which each agent can increase signature sensitivity is an open research question and will be studied as development into the MAS and IDS fields continue; a lack of datasets to test this has lead to its inclusion as future work.

- The series weighting algorithms presented as part of the DMASS have provided interesting results and shown the benefit of on-demand decentralised processing of information com-

pared to the bulk processing of all available data. By introducing additional mechanisms to consider the accuracy of the analysis (i.e., the domains and preferred agent model), false alarms and the overall amount of processing can be reduced. Algorithms that make use of these variables and include problem-specific information to evaluate the information collection process have been shown to be more reliable than the brute-force approach and may have applications in other areas were correct classifications are difficult to make without objective standards for making decisions. The context-based approach to problem-solving could be applied in other fields where the data is complex and variable, requiring more in-depth solutions. While classifying events on a single data point has proven to be the most popular method of analysis, considering several data points and the context in which they were collected could be applied to many fields to solve complex problems.

- An adapted multi-agent model developed within the simulator has already been applied to the problem of web service composition in cloud computing environments. Web service composition shares many environmental factors with that of network security; in particular, additional searching can result in higher costs and slower composition fulfilments. The multi-agent approach of using meta-information to reduce searching by relying on past composition histories, information about the distribution of web services and the requirements of the users has shown to be a more efficient way of addressing the issues within this area.

References

[1] Abadeh, M. S., Habibi, J., and Lucas, C. Intrusion detection using a fuzzy genetics-based learning algorithm. *Journal of Network and Computer Applications 30*, 1 (2007), 414–428.

[2] Abdalla, A., Jamil, H. A., Hamza Ibrahim, H. A., and Nor, S. M. Malware dection using IP flow level attributes. *Journal of Theoretical and Applied Information Technology 57*, 3 (2013), 530–539.

[3] Afzali Seresht, N., and Azmi, R. MAIS-IDS: A distributed intrusion detection system using multi-agent AIS approach. *Engineering Applications of Artificial Intelligence 35* (2014), 286–298.

[4] Agre, P. E., and Chapman, D. Pengi: An Implementation of a Theory of Activity. *Proceedings of the Sixth National Conference on Artificial Intelligence 278* (1987), 268–272.

[5] Ahmed, M., Naser Mahmood, A., and Hu, J. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications 60* (2016), 19–31.

[6] Al-Jarrah, O. Y., Alhussein, O., Yoo, P. D., Muhaidat, S., Taha, K., and Kim, K. Data Randomization and Cluster-Based Partitioning for Botnet Intrusion Detection. *IEEE Transactions on Cybernetics 46*, 8 (2016), 1796–1806.

[7] Alechina, N., Dastani, M., and Logan, B. Reasoning about normative update. *IJCAI International Joint Conference on Artificial Intelligence 1* (2013), 20–26.

[8] Alkhateeb, F., Al-fakhry, Z. A.-a., Maghayreh, E. A., Aljawarneh, S., and Al-taani, A. T. A multi agent-based system for securing university campus: Design and architecture. *International Conference on Intelligent Systems, Modelling and Simulation, IEEE 2*, March (2010), 75–79.

[9] Alsubhi, K., Bouabdallah, N., and Boutaba, R. Performance analysis in Intrusion Detection and Prevention Systems. *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops* (2011), 369–376.

[10] Anita, R., and Murugan, K. Improving The Accuracy of Intrusion Detection Systems In Mobile Adhoc Network Using Fuzzy Logic Method. *International Journal of Applied Engineering Research 10*, 7 (2015), 17933–17950.

[11] Arib, S., and Aknine, S. A plan based coalition formation model for multi-agent systems. *Proceedings - 2011 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2011 2* (2011), 365–368.

[12] a.S. Rao, and Georgeff, M. P. Modeling rational agents within a BDI-architecture. *Readings in agents* (1997), 317–328.

[13] Ashfaq, R. A. R., Wang, X. Z., Huang, J. Z., Abbas, H., and He, Y. L. Fuzziness based semi-supervised learning approach for intrusion detection system. *Information Sciences 378* (2017), 484–497.

[14] Aung, Y. Y. A Collaborative Intrusion Detection Based on K-means and Projective Adaptive Resonance Theory. *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)* (2017), 1575–1579.

[15] Bacchus, F., and Kabanza, F. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence 116*, 1-2 (2000), 123–191.

[16] Bahrani, Ali, Jun Yuan, Chukwuemeka David Emele, Daniele Masato, Timothy J. Norman, D. M. Collaborative and context-aware planning. In *Military Communications Conference* (2008), pp. 1–7.

[17] Baig, Z. a. Multi-agent systems for protecting critical infrastructures: A survey. *Journal of Network and Computer Applications 35*, 3 (2012), 1151–1161.

[18] Balajinath, B., and Raghavan, S. Intrusion detection through learning behavior model. *Computer Communications 24*, 12 (2001), 1202–1212.

[19] Balasubramaniyan, J. S., Garcia-Fernandez, J. O., Isacoff, D., Spafford, E., and Zamboni, D. An architecture for intrusion detection using autonomous agents. *Proceedings 14th Annual Computer Security Applications Conference Cat No98EX217 34*, 4 (1998), 13–24.

[20] Barron, P., Horsky, M., and Persson, J. Intrusion Detection Systems. *Report* (2005), 1–7.

[21] Bass, T. Intrusion Detection Systems & Multisensor Data Fusion : Creating Cyberspace Situational Awareness. *Communications of the ACM 43*, 4 (2000), 100–105.

[22] Beghdad, R. Critical study of neural networks in detecting intrusions. *Computers & Security 27*, 5-6 (2008), 168–175.

[23] Bernstein, D., Givan, R., Immerman, N., and Zilberstein, S. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research 27*, 4 (2002), 819–840.

[24] Bhatt, P., Yano, E. T., and Gustavsson, P. Towards a framework to detect multi-stage advanced persistent threats attacks. *Proceedings - IEEE 8th International Symposium on Service Oriented System Engineering, SOSE 2014* (2014), 390–395.

[25] Bhuyan, M. H., Bhattacharyya, D. K., and Kalita, J. K. Network Anomaly Detection: Methods, Systems and Tools. *Communications Surveys & Tutorials, IEEE 16*, 1 (2014), 303–336.

[26] Bivens, A., Palagiri, C., Smith, R., and Szymanski, B. Network-based intrusion detection using neural networks. *Intelligent Engineering Systems through Artificial Neural Networks 12* (2002), 579–584.

[27] Blai, B., and Geffner, H. Heuristic Search Planner 2.0. *AI Magazine 22*, 3 (2001), 77–80.

[28] Blei, D. M., Edu, B. B., Ng, A. Y., Edu, A. S., Jordan, M. I., and Edu, J. B. Latent Dirichlet Allocation. *Journal of Machine Learning Research 3* (2003), 993–1022.

[29] Bonet, B., and Geffner, H. Planning as heuristic search. *Artificial Intelligence 129*, 1-2 (2001), 5–33.

[30] Bratman, M. *Intention, Plans, and Practical Reason (David Hume Series)*. Center for the Study of Language and Information (March 1, 1999), 1987.

[31] Bratman, M. E. *Intention, Plans, and Practical Reason*, new editio ed. The Center for the Study of Language and Information Publications, 1999.

[32] Broomhead, D. S., and Lowe, D. Radial basis functions, multi-variable functional interpolation and adaptive networks. *Royal Signals and Radar Establishment Malvern* (1988), 1–.

[33] Brugger, S., and Chow, J. An assessment of the DARPA IDS Evaluation Dataset using Snort. *UCDAVIS department of Computer Science* (2007), 1–19.

[34] Cao, X., Chen, B., and Li, H. Packet Header Anomaly Detection Using Bayesian Topic Models. *IACR Cryptology ePrint Archive* (2016), 1–12.

[35] Chapman, M., Tyson, G., Parsons, S., Mcburney, P., and Luck, M. Playing Hide-and-Seek : An Abstract Game for Cyber Security. In *Proceedings of the 1st International Workshop on Agents and CyberSecurity Article No. 3* (2014).

[36] Chavan, S., Shah, K., Dave, N., Mukherjee, S., Abraham, A., and Sanyal, S. Adaptive neuro-fuzzy intrusion detection systems. *Information Technology: Coding and Computing 1* (2004), 70–74.

[37] Chen, P., Desmet, L., and Huygens, C. A study on advanced persistent threats. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2014), vol. 8735 LNCS, pp. 63–72.

[38] Cohen, P. R., and Levesque, H. J. Persistence, intention, and commitment. *Proceedings of the 1986 workshop on Reasoning about Actions and Plans* (1990), 297.

[39] Coles, A. Enforced Hill Climbing Search, 2007.

[40] Cooper, A. Experiments with Applying Artificial Immune System in Network Attack Detection. *2017 KSU Conference on Cybersecurity Education, Research and Practice, KSU Proceedings on Cybersecurity Education, Research, Research and Practice.* (2017).

[41] Cox, J., and Durfee, E. An efficient algorithm for multiagent plan coordination. *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems* (2005), 828–835.

[42] Cox, J., and Durfee, E. H. Discovering and exploiting synergy between hierarchical planning agents. *Proceedings of the second international joint conference on Autonomous agents and multiagent systems* (2003), 281–288.

[43] D G Bukhanov, V. M. P., and Belgorod. Detection of network attacks based on adaptive resonance theory. *International Journal of Computer Applications 166.4* (2017), 13–17.

[44] DARPA. Innovative Approaches to Planning, Scheduling and Control 1990: Proceedings of the 1990 DARPA Workshop. In *Proceedings of the 1990 DARPA Workshop* (1990), pp. 1–507.

[45] de Weerdt, M., Zhang, Y., and Klos, T. Distributed task allocation in social networks. *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems (AAMAS '07) 5* (2007), 1.

[46] Dickerson, J., Juslin, J., Koukousoula, O., and Dickerson, J. Fuzzy intrusion detection. *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference (Cat. No. 01TH8569) 3*, C (2001), 1506–1510.

[47] Digits, D., Author, N. N., and Source, S. N. Note on the , Frequency of Use of the Different Natural Digits in. *American Journal of Mathematics 4*, 1 (2015), 39–40.

[48] Division, C. S. Integrated Enterprise-Wide Risk Management: Organization, Mission and Information System View. Tech. Rep. March, National Institute of Standards and Technology, 2011.

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

[50] El-Hajj, W., Aloul, F., Trabelsi, Z., and Zaki, N. On Detecting Port Scanning using Fuzzy Based Intrusion Detection System. *2008 International Wireless Communications and Mobile Computing Conference* (2008).

[51] Elster, J., Journal, T., and Autumn, N. Social Norms and Economic Theory. *The Journal of Economic Perspectives 3*, 4 (2008), 99–117.

[52] Emerson, E. A., and Halpern, J. Y. "Sometimes" and "not never" revisited: on branching versus linear time temporal logic. *Journal of the ACM 33*, 1 (1986), 151–178.

[53] Ephrati, E., and Rosenschein, J. S. Divide and Conquer in Multi-agent Planning. *AAAI Conference on Artificial Intelligence* (1994), 375–380.

[54] Espinasse, B., Picolet, G., and Chouraqui, E. Negotiation support systems: A multi-criteria and multi-agent approach. *European Journal of Operational Research 103*, 2 (1997), 389–409.

[55] Fatima, S., Michalak, T., and Wooldridge, M. Bargaining for Coalition Structure Formation. In *ECAI 2014: 21st European Conference on Artificial Intelligence* (2014).

[56] Fenet, S., and Hassas, S. A distributed intrusion detection and response system based on mobile autonomous agents using social insects communication paradigm. *Electronic Notes in Theoretical Computer Science 63*, 4 (2002), 43–60.

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

[58] Fink, G. a., and McKinnon, a. D. Effects of network delays on swarming in a multi-agent security system. *Proceedings of the 1st International Workshop on Agents and CyberSecurity - ACySE '14* (2014), 1–8.

[59] FireEye. Threat Report: A view from the front lines. Tech. Rep. 1, https://www2.fireeye.com/WEB-2015-MNDT-RPT-M-Trends-2015_LP.html, 2015.

[60] Frank, J. Artificial intelligence and intrusion detection: Current and future directions. In *Artificial intelligence and intrusion detection: Current and future directions* (1994), pp. 1–12.

[61] Friedberg, I., Skopik, F., Settanni, G., and Fiedler, R. Combating advanced persistent threats: From network event correlation to incident detection. *Computers and Security 48* (2015), 35–57.

[62] Geffner, B. B., and Hector. Planning as Heuristic Search: New Results. *Recent Advances in AI Planning* (2000), 360–372.

[63] Geib, C., and Goldman, R. Plan recognition in intrusion detection systems. *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01 1* (2001), 46–55.

[64] Gmytrasiewicz, P., Summers, M., and Gopal, D. Toward automated evolution of agent communication languages. *Proceedings of the 35th Annual Hawaii International Conference on System Sciences 00*, Austin 1962 (2002), 1–10.

[65] Goldman, C. V., and Zilberstein, S. Optimizing information exchange in cooperative multi-agent systems. *Proceedings of the second international joint conference on Autonomous agents and multiagent systems - AAMAS '03* (2003), 137.

[66] Grossberg, S. Adaptive Resonance Theory: How a brain learns to consciously attend, learn, and recognize a changing world. *Neural Networks 37* (2013), 1–47.

[67] Haack, J. N., Fink, G. a., Maiden, W. M., McKinnon, a. D., Templeton, S. J., and Fulp, E. W. Ant Based Cyber Security. *Proceedings - 2011 8th International Conference on Information Technology: New Generations, ITNG 2011* (2010), 918–926.

[68] Hall, M. Correlation-based Feature Selection for Machine Learning. *Methodology 21i195-i20*, April (1999), 1–5.

[69] Heidarian, Z., Movahedinia, N., Moghim, N., and Mahdinia, P. Intrusion Detection Based on Normal Traffic Specifications. *International Journal of Computer Network and Information Security(IJCNIS) 7*, 9 (2015), 32.

[70] Helmer, G., Wong, J. S. K., Honavar, V., Miller, L., and Wang, Y. Lightweight agents for intrusion detection. *Journal of Systems and Software 67*, 2 (2003), 109–122.

[71] Herzog, P. OSSTMM 3.0 - The Open Source Security Testing Methodology Manual. Tech. Rep. 3.0, ISECOM, 2010.

[72] Hoffmann, J. FF: The fast-forward planning system. *AI magazine 22* (2001), 57–62.

[73] Hutchins, E. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Issues in Information Warfare & Security Research*, July 2005 (2011), 1–14.

[74] IBM. Perimeter Model Security, 2008.

[75] IEEE. IEEE - The world's largest professional association for the advancement of technology, 2015.

[76] IEEE. Welcome to the Foundation for Intelligent Physical Agents, 2015.

[77] ISACA. State of Cybersecurity : Implications for 2015. *CyberSecurity Nexus* (2015), 22.

[78] Jahanbin, A., Ghafarian, A., Seno, S. A. H., and Nikookar, S. A Computer Forensics Approach Based on Autonomous Intelligent Multi-Agent System. *International Journal of Database Theory and Application 6*, 5 (oct 2013), 1–12.

[79] Jennings, N. R. Automated Negotiation : Prospects , Methods and Challenges. *Group Decision and Negotiation*, Wooldridge 1997 (2001), 199–215.

[80] Jennings, N. R., Sycara, K., and Wooldridge, M. A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems 38* (1998), 7–38.

[81] Jiang, D., Xu, Z., Zhang, P., and Zhu, T. A transform domain-based anomaly detection approach to network-wide traffic. *Journal of Network and Computer Applications 40*, 1 (2014), 292–306.

[82] Jorg Hoffmann, B. N. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Artificial Intelligence 2*, 27 (2001).

[83] Kautz, H. a., and Allen, J. Generalized Plan Recognition. *AAAI 86* (1986), 32–37.

[84] KDD. KDD Cup 1999 Data (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html).

[85] Kendrick, P., Baker, T., Maamar, Z., Hussain, A., and Buyya, R. An Efficient Multi-Cloud Service Composition Via A Distributed Multiagent-based , Memory-driven Approach. *IEEE Transactions on Sustainable Computing* (2018).

[86] Kendrick, P., Criado, N., Hussain, A., and Randles, M. A self-organising multi-agent system for decentralised forensic investigations. *Expert Systems with Applications 102* (2018), 12–26.

[87] Kendrick, P., Criado, N., Hussain, A., and Randles, M. An Unsupervised Approach to Detecting Local Network Infiltrations. *IEEE Transactions on Sustainable Computing, Special Issue on Sustainable Cyber Forensics and Threat Intelligence* (2018), 1–12.

[88] Kendrick, P., Hussain, A., and Criado, N. Multi-Agent Systems for Dynamic Forensic Investigation. In *International Conference on Intelligent Computation (ICIC)* (Lanzhou, 2016), Springer, pp. 796—-807.

[89] Kendrick, P., Hussain, A., Criado, N., and Randles, M. Multi-agent systems for scalable internet of things security. In *ICC '17 Proceedings of the Second International Conference on Internet of things and Cloud Computing* (2017).

[90] Kendrick, P., Hussain, A., Criado, N., and Randles, M. Selecting Scalable Network Features for Infiltration Detection. In *2017 10th International Conference on Developments in eSystems Engineering (DeSE)* (2017), pp. 88–93.

[91] Khalastchi, E., Kalech, M., and Rokach, L. A Hybrid Approach for Fault Detection in Autonomous Physical Agents. In *International Conference on Autonomous Agents & Multi-Agent Systems* (2014), pp. 941–948.

[92] Kikuchi, H., Kobori, T., and Terada, M. Orthogonal expansion of port-scanning packets. In *NBiS 2009 - 12th International Conference on Network-Based Information Systems* (aug 2009), Ieee, pp. 321–326.

[93] Kolias, C., Kambourakis, G., and Maragoudakis, M. Swarm intelligence in intrusion detection: A survey. *Computers and Security 30*, 8 (2011), 625–642.

[94] Korf, R. Planning as search: A quantitative approach. *Artificial Intelligence 33*, 1 (1987), 65–88.

[95] Korf, R. E. Finding Optimal Solutions to Rubik's Cube Using Pattern Databases. *American Association for Artificial Intelligence (AAAI)* (1997), 700—-705.

[96] Kothari, V., Blythe, J., Smith, S., and Koppel, R. Agent-based modeling of user circumvention of security. *Proceedings of the 1st International Workshop on Agents and CyberSecurity - ACySE '14* (2014), 1–4.

[97] Lakhina, A., Crovella, M., and Diot, C. Mining anomalies using traffic feature distributions. *ACM SIGCOMM Computer Communication Review 35*, 4 (2005), 217.

[98] Langin, C., and Rahimi, S. Soft computing in intrusion detection: The state of the art. *Journal of Ambient Intelligence and Humanized Computing 1*, 2 (2010), 133–145.

[99] Lee, B., Amaresh, S., Green, C., and Engels, D. Comparative Study of Deep Learning Models for Network Intrusion Detection. *SMU Data Science Review 1*, 1 (2018), 8.

[100] Leu, F.-Y., and Hu, K.-W. A real-time intrusion detection system using data mining technique. *Int Conf on Cybernetics and Information Technologies, Systems and Applications/Int Conf on Computing, Communications and Control Technologies, Vol Ii* (2007), 148–153.

[101] Li, X. A Multi-Agent Based Legacy Information System Integration Strategy A . the integration framework based on multi-Agent. *Networking and Digital Society (ICNDS)* (2010), 72–75.

[102] Lippmann, R., and Haines, J. Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation. *Recent Advances in Intrusion Detection, Third International Workshop* (2000), 16–182.

[103] Lippmann, R. P., and Cunningham, R. K. Improving intrusion detection performance using keyword selection and neural networks. *Computer Networks 34*, 4 (2000), 597–603.

[104] Liu, H., and Setiono, R. A probabilistic approach to feature selection - a filter solution. *Proc 13th International Conference on Machine Learning 96* (1996), 319–327.

[105] Lodi, G., Aniello, L., Luna, G. D., and Baldoni, R. An event-based platform for collaborative threats detection and monitoring. *Information Systems 39*, 1 (jan 2014), 175–195.

[106] Lopez, J., Alcaraz, C., Rodriguez, J., Roman, R., and Rubio, J. E. Protecting Industry 4 . 0 against Advanced Persistent Threats The threat of APTs. *Euro CIIP Newslett 11* (2017), 27–29.

[107] Lorliam, A., Tirunagari, S., Ho, A., Li, S., Waller, A., and Poh, N. Flow Size Difference Can Make a Difference: Detecting Malicious TCP Network Flows Based on Benford's Law. In *TBA* (2016).

[108] Lyon, G. F. Nmap: the Network Mapper - Security Scanner, 2008.

[109] Mandiant. M-Trends 2018 Report. Tech. rep., FireEye, 2018.

[110] McDermott, D., Ghallab, M., Howe, a., and C. PDDL-the planning domain definition language. *The AIPS-98 Planning* (1998).

[111] McHugh, J. Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and System Security 3*, 4 (2000), 262–294.

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

[113] Mees, W., and Debatty, T. Multi-agent System for APT Detection. *2014 IEEE International Symposium on Software Reliability Engineering Workshops* (2014), 401–406.

[114] Megherbi, D. B. A Collaborative Distributed Multi-Agent Reinforcement Learning Technique for Dynamic Agent Shortest Path Planning Via Selected Sub-goals in Complex Cluttered Environments. In *IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)* (2015), pp. 118–124.

[115] Mehmood, A., Mukherjee, M., Ahmed, S. H., Song, H., and Malik, K. M. NBC-MAIDS: Naïve Bayesian classification technique in multi-agent system-enriched IDS for securing IoT against DDoS attacks. *Journal of Supercomputing 74*, 10 (2018), 1–15.

[116] (MIT), M. I. o. T. MIT Lincoln Laboratory: DARPA Intrusion Detection Evaluation, 1998.

[117] Moustafa, N., and Slay, J. The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective 3555*, January (2016), 1–14.

[118] Mukkamala, S., Janoski, G., and Sung, a. Intrusion detection using neural networks and support vector machines. *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02* (2002), 1702–1707.

[119] Nau, D., Cao, Y., Lotem, A., and Muftoz-Avila, H. SHOP: Simple hierarchical ordered planner. *IJCAI International Joint Conference on Artificial Intelligence 2* (1999), 968–973.

[120] NIST. Framework for Improving Critical Infrastructure Cybersecurity. *National Institute of Standards and Technology* (2014), 1–41.

[121] Obes, J. L., Sarraute, C., and Richarte, G. Attack Planning in the Real World. In *AAAI Conference on Artificial Intelligence* (Atlanta, 2010), pp. 10–17.

[122] Orfila, A., Carbo, J., and Ribagorda, A. Intrusion Detection Effectiveness Improvement by a Multi-agent System. *International Journal of Computer Science Applications 2*, 1 (2005), 1–6.

[123] Oriwoh, E., Jazani, D., Epiphaniou, G., and Sant, P. Internet of Things Forensics : Challenges and Approaches. *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 9th (2013).

[124] Padgham, L., and Lambrix, P. Formalisations of capabilities for BDI-agents. *Autonomous Agents and Multi-Agent Systems 10*, 3 (2005), 249–271.

[125] Pajares Ferrando, S., and Onaindia, E. Context-Aware Multi-Agent Planning in intelligent environments. *Information Sciences 227* (2013), 22–42.

[126] Pajouh, H. H., Dastghaibyfard, G., and Hashemi, S. Two-tier network anomaly detection model: a machine learning approach. *Journal of Intelligent Information Systems 48*, 1 (2017), 61–74.

[127] Palmieri, F., and Fiore, U. Network anomaly detection through nonlinear analysis. *Computers & Security 29*, 7 (2010), 737–755.

[128] Papamartzivanos, D., Gómez Mármol, F., and Kambourakis, G. Dendron: Genetic trees driven rule induction for network intrusion detection systems. *Future Generation Computer Systems 79* (2018), 558–574.

[129] Parunak, H. V. D., Baker, a. D., and Clark, S. J. The AARIA agent architecture: an example of requirements-driven agent-based system design. *Proceedings of the first international conference on Autonomous agents* (1997), 482–483.

[130] Perez, M. A., and Carbonell, J. Control Knowledge to Improve Plan Quality. *Proc. Second International Conference on Artificial Intelligence Planning Systems* (1994), 323–328.

[131] Pervez, M. S., and Farid, D. M. Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs. *SKIMA 2014 - 8th International Conference on Software, Knowledge, Information Management and Applications* (2014).

[132] Ponemon Institute LLC. 2017 Cost of Data Breach Study. Tech. Rep. March, Ponemon Institute, 2017.

[133] Poslad, S. Specifying protocols for multi-agent systems interaction. *ACM Transactions on Autonomous and Adaptive Systems 2*, 4 (2007), 15–es.

[134] Rastegari, S., Hingston, P., and Lam, C. P. Evolving statistical rulesets for network intrusion detection. *Applied Soft Computing Journal 33* (2015), 348–359.

[135] Rocha, L. From Artificial Life to Semiotic Agent Models. *Evolution 836* (1999), 29.

[136] Roesch, M., and Green, C. Snort User Manual. Tech. rep., Sourcefire Inc, 2016.

[137] Rudrapal, D., Das, S., Debbarma, N., and Debbarma, S. Internal attacker detection by analyzing user keystroke credential. *Lecture Notes on Software Engineering 1*, 1 (2013), 49.

[138] Saurabh, P., and Verma, B. An efficient proactive artificial immune system based anomaly detection and prevention system. *Expert Systems with Applications 60* (2016), 311–320.

[139] Seuken, S., and Zilberstein, S. Memory-bounded dynamic programming for DEC-POMDPs. *IJCAI International Joint Conference on Artificial Intelligence* (2007), 2009–2015.

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

[141] Shakarian, P., Simari, G. I., Moores, G., Parsons, S., Falappa, M. A., Engineering, E., Science, C., Academy, U. S. M., and Point, W. An Argumentation-Based Framework to Address the Attribution Problem in Cyber-Warfare. *CoRR abs/1404.6* (2014).

[142] Shakarian, Paulo, Gerardo I. Simari, M. A. F. Belief revision in structured argumentation. *Foundations of Information and Knowledge Systems* (2014), 324–343.

[143] Shanmugasundaram, K., Memon, N., Savant, A., and Bronnimann, H. ForNet : A Distributed Forensics Network. *Computer Network Security. Springer* (2003), 1–16.

[144] Shiravi, A., Shiravi, H., Tavallaee, M., and Ghorbani, A. A. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers and Security 31*, 3 (2012), 357–374.

[145] Singh, M. P. Cybersecurity as an Application Domain for Multiagent Systems. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems* (2015), pp. 1207–1212.

[146] Smith, G. E., Watson, K. J., Baker, W. H., and Pokorski II, J. a. A critical balance: collaboration and security in the IT-enabled supply chain. *International Journal of Production Research 45*, 11 (2007), 2595–2613.

[147] Spafford, E. H., and Zamboni, D. Intrusion detection using autonomous agents. *Computer Networks 34*, 4 (2000), 547–570.

[148] Specht, D. F. A general regression neural network. *Neural Networks, IEEE Transactions on 2*, 6 (1991), 568–576.

[149] Sung, A. H., and Mukkamala, S. Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks. *Proceedings of the Symposium on Applications and the Internet (SAINT'03) 1*, 1 (2003), 209–216.

[150] Tama, B. A., and Rhee, K. H. An in-depth experimental study of anomaly detection using gradient boosted machine. *Neural Computing and Applications* (2017), 1–11.

[151] Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. A Detailed Analysis of the KDD CUP 99 Data Set. *Computational Intelligence for Security and Defense Applications* (2009).

[152] Tobergte, D. R., and Curtis, S. Modeling Multistep Cyber Attacks for Scenario Recognition. *Journal of Chemical Information and Modeling 53*, 9 (2013), 1689–1699.

[153] Toosi, A. N., and Kahani, M. A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers. *Computer Communications 30*, 10 (2007), 2201–2212.

[154] Uppuluri, P., and Sekar, R. Experiences with Specification-Based Intrusion Detection. In *International Workshop on Recent Advances in Intrusion Detection* (2001), pp. 172–189.

[155] Uszok, a., Bradshaw, J., Jeffers, R., Suri, N., Hayes, P., Breedy, M., Bunch, L., Johnson, M., Kulkarni, S., and Lott, J. KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. *Proceedings - POLICY 2003: IEEE 4th International Workshop on Policies for Distributed Systems and Networks* (2003), 93–96.

[156] Vargas-Vera, M., and Nagy, M. Establishing agent trust for contradictory evidence by means of fuzzy voting model: An ontology mapping case study. *Computers in Human Behavior 30* (2014), 745–752.

[157] Vecchiato, D., Maciel, C., and El, A. Using agents towards providing security on a context-aware architecture (Position Paper). In *Proceedings of the 1st International Workshop on Agents and CyberSecurity* (2014), pp. 1–4.

[158] Verwoerd, T., and Hunt, R. Intrusion detection techniques and approaches. *Computer Communications 25*, 15 (2002), 1356–1365.

[159] Villata, S., Boella, G., Gabbay, D. M., and van der Torre, L. Arguing about Trust in Multiagent Systems. *11th Symposium on Artificial Intelligence of the Italian Association for Artificial Intelligence (AI*IA 2010)*, arg c (2010), 1–8.

[160] Vinayakumar, R., Soman, K. P., and Poornachandran, P. Applying convolutional neural network for network intrusion detection. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (2017), 1222–1228.

[161] Virvilis, N., and Gritzalis, D. The big four - What we did wrong in advanced persistent threat detection? *Proceedings - 2013 International Conference on Availability, Reliability and Security, ARES 2013* (2013), 248–254.

[162] Wang, B., Zheng, Y., Lou, W., and Hou, Y. T. DDoS attack protection in the era of cloud computing and Software-Defined Networking. *Computer Networks 81* (2015), 308–319.

[163] Wang, J., and Gasser, L. Mutual online concept learning for multiple agents. *Proceedings of the first international joint conference on Autonomous agents and multiagent systems part 1 - AAMAS '02* (2002), 362.

[164] Wang, Y., and Luo, J. Ontology learning and mapping in semantic web based on formal concept analysis technology. *Journal of Convergence Information Technology 7*, 10 (2012), 381–388.

[165] Wooldridge, M., Street, C., Manchester, M., and Jennings, N. R. Intelligent Agents : Theory and Practice. *The Knowledge Engineering Revie 10*, October 1994 (1995), 1–62.

[166] Woolridge, M. *An Introduction To MultiAgent Systems*, 2 ed. Wiley, 2011.

[167] Wu, F., Zilberstein, S., and Chen, X. Online planning for multi-agent systems with bounded communication. *Artificial Intelligence 175*, 2 (2011), 487–511.

[168] Wu, F., Zilberstein, S., and Jennings, N. R. Monte-Carlo expectation maximization for decentralized POMDPs. *IJCAI International Joint Conference on Artificial Intelligence* (2013), 397–403.

[169] Wu, J., Peng, D., Li, Z., Zhao, L., and Ling, H. Network Intrusion Detection Based on a General Regression Neural Network Optimized by an Improved Artificial Immune Algorithm. *Plos One 10*, 3 (2015), e0120976.

[170] Wu, S. X., and Banzhaf, W. The use of computational intelligence in intrusion detection systems: A review. *Applied Soft Computing 10*, 1 (2010), 1–35.

[171] Wu, X., Kumar, V., Ross, Q. J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z. H., Steinbach, M., Hand, D. J., and Steinberg, D. Top 10 algorithms in data mining. *Knowledge and information systems 14* (2008), 1–37.

[172] Xiao, J. X. J., and Song, H. S. H. A Novel Intrusion Detection Method Based on Adaptive Resonance Theory and Principal Component Analysis. *2009 WRI International Conference on Communications and Mobile Computing 3* (2009).

[173] Ye, D., Zhang, M., and Vasilakos, A. V. A Survey of Self-organisation Mechanisms in Multi-Agent Systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems 47*, 3 (2016), 1–21.

[174] Zivan, R., Yedidsion, H., Okamoto, S., Glinton, R., and Sycara, K. Distributed constraint optimization for teams of mobile sensing agents. *Autonomous Agents and Multi-Agent Systems* (2014), 495–536.

[175] Zuech, R., Khoshgoftaar, T. M., and Wald, R. Intrusion detection and Big Heterogeneous Data: a Survey. *Journal of Big Data 2*, 1 (2015), 1–41.