Mining Intrusion Detection Alert Logs to Minimise False Positives & Gain Attack Insight

Riyanat O. Shittu

A dissertation submitted in partial fulfilment

of the requirements for the degree of

Doctor of Philosophy

of the

City University London.

CITY UNIVERSITY
LONDON

School of Engineering, Mathematics and Computer Sciences

2016

Declaration

No portion of the work contained in this document has been submitted in support of an application for a degree or qualification of this or any other university or other institution of learning. All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged.

Signed:

Date:

Abstract

Utilising Intrusion Detection System (IDS) logs in security event analysis is crucial in the process of assessing, measuring and understanding the security state of a computer network, often defined by its current exposure and resilience to network attacks. Thus, the study of understanding network attacks through event analysis is a fast growing emerging area. In comparison to its first appearance a decade ago, the complexities involved in achieving effective security event analysis have significantly increased. With such increased complexities, advances in security event analytical techniques are required in order to maintain timely mitigation and prediction of network attacks.

This thesis focusses on improving the quality of analysing network event logs, particularly intrusion detection logs by exploring alternative analytical methods which overcome some of the complexities involved in security event analysis.

This thesis provides four key contributions. Firstly, we explore how the quality of intrusion alert logs can be improved by eliminating the large volume of false positive alerts contained in intrusion detection logs. We investigate probabilistic alert correlation, an alternative to traditional rule based correlation approaches. We hypothesise that probabilistic alert correlation aids in discovering and learning the evolving dependencies between alerts, further revealing attack structures and information which can be vital in eliminating false positives. Our findings showed that the results support our defined hypothesis, aligning consistently with existing literature. In addition, evaluating the model using recent attack datasets (in comparison to outdated datasets used in many research studies) allowed the discovery of a new set of issues relevant to modern security event log analysis which have only been introduced and addressed in few research studies.

Secondly, we propose a set of novel prioritisation metrics for the filtering of false positive intrusion alerts using knowledge gained during alert correlation. A combination of heuristic, temporal and anomaly detection measures are used to define metrics which capture characteristics identifiable in common attacks including denial-of-service attacks and worm propagations. The most relevant of the novel metrics, *Outmet* is based on the well known Local Outlier Factor algorithm. Our findings showed that with a slight trade-off of sensitivity (i.e. true positives performance), outmet reduces false positives significantly. In comparison to prior state-of-the-art, our findings show that it performs more efficiently given a variation of attack scenarios.

Thirdly, we extend a well known real-time clustering algorithm, *CluStream* in order to support the categorisation of attack patterns represented as graph like structures. Our motive behind attack pattern categorisation is to provide automated methods for capturing consistent behavioural patterns across a given class of attacks. To our knowledge, this is a novel approach to intrusion alert analysis. The extension of CluStream resulted is a novel light weight real-time clustering algorithm for graph structures. Our findings are new and complement existing literature. We discovered that in certain case studies, repetitive attack behaviour could be mined. Such a discovery could facilitate the prediction of future attacks.

Finally, we acknowledge that due to the intelligence and stealth involved in modern network attacks, automated analytical approaches alone may not suffice in making sense of intrusion detection logs. Thus, we explore visualisation and interactive methods for effective visual analysis which if combined with the automated approaches proposed, would improve the overall results of the analysis. The result of this is a visual analytic framework, integrated and tested in a commercial Cyber Security Event Analysis Software System distributed by British Telecom.

Acknowledgements

The last four years have been an intellectually stimulating, challenging, yet exciting journey. I can genuinely say *"I am who I am because of everyone who has supported me"*. Through out the course of this research degree, I have received continuous support, advice and guidance by those who surround me. I would like to use this opportunity to express my gratitude to those people.

First and foremost, I would like to thank my Supervisor, Professor Rajarajan Muttukrishnan. I am grateful for your academic and industrial expertise, research guidance, constructive criticism, mentorship, and continuous support. Your invaluable contributions made it possible to establish a strong research collaboration with the Security Futures Practise research team at British Telecom and it is through this collaboration that this research direction evolved. Without all your support, the contributions and results from this research degree would not have been possible.

I would like to thank EPSRC and City University London for sponsoring and supporting this research degree. I would also like to express special gratitude to Robert Ghanea-Hercock, Alex Healing and the Security Futures Practice research team at British Telecom, Adastral Park for their co-sponsorship, mentorship and for providing a great insight into the domain of cyber security. This really helped me gain an understanding of the domain problems and how to define research objectives.

To Dr Yogachandran Rahulamathavan and Dr Suresh Veluru, your positivity, optimism, calmness and highly intellectual approaches to using mathematical techniques is very admirable. Thank you for your support and for the numerous discussions on mathematical methodologies.

I am extremely grateful to my mum and dad, for supporting me through the last four years of this research degree. Their patience, advice, proof-reading skills, emotional and spiritually support has been invaluable. I couldn't ask for better parents. You are the best.

To my best friend Vivienne, thank you for being a great motivator and listener. I am extremely grateful.

While it is rather unusual, I would also like to thank myself. More specifically, *my former self - the yesterday me*. Thank you for the little progress everyday, for not giving up, for the self motivation. I have learnt that sometimes, it is these little things that make a big difference.

Lastly but certainly most importantly, I would like to thank God. For through God I have learnt one of the simplest lessons of faith, which, indeed made this journey all the more invaluable. Through many experiences in these four years, I have learnt, that *"Faith is not belief without proof, but trust without reservation"*. This, I am extremely grateful for.

Dedication

To the amazing strong women in my family - My grandmother, my mum, my aunty and my best friend. Thank you for being the women that you are. Thank you for your patience, encouragement and unending support.

Publications

Shittu, R., Healing, A., Bloomfield, R., and Muttukrishnan, R. (2012). Visual analytic agent-based framework for intrusion alert analysis. *Proceedings of the 2012 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2012*, pages 201–207

Rowlingson, R., Healing, A., Shittu, R., Matthews, S. G., and Ghanea-Hercock, R. (2013). Visual Analytics in the Cyber Security Operations Centre. *Proceedings of The Information Systems Technology Panel Symposium on Visual Analytics*

Shittu, R., Healing, A., Ghanea-hercock, R., and Bloomfield, R. (2014). OutMet : A New Metric for Prioritising Intrusion Alerts using Correlation and Outlier Analysis. *19th IEEE Conference on Local Computer Networks*

Shittu, R., Healing, A., Ghanea-Hercock, R., Bloomfield, R., and Rajarajan, M. (2015b). Intrusion alert prioritisation and attack detection using post-correlation analysis. *Computers & Security*, 50:1–15

Shittu, R., Healing, A., Ghanea-hercock, R., Bloomfield, R., and Muttukrishnan, R. (2015a). Real-time Graph Clustering for automatically discovering novel attack patterns. *Network and Computer Applications (Under Review)*, pages 1–25

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Network attacks have rapidly increased over the years. As of 2013, 200 new threats to computer networks were detected every minute [104].

To minimise the number of successful attacks targeted at computer networks, defence security systems such as firewalls, intrusion detections systems, intrusion prevention systems, anti-virus systems and many more are placed at various layers of the computer network. Many of these devices prevent malicious network traffic from flowing in and out of the network by blocking malicious traffic or triggering alerts which can be acted upon.

These devices, particularly those placed at the perimeter of the network, play an important role in the detection and prevention of attacks. The alerts triggered by these devices are rich in attack information and *if utilized effectively*, can be used to gain insight into the context and details of a past or ongoing attack against the network. In well known attacks and case studies such as [95, 65, 57, 73], investigating network and security event logs played a major role in tracing the sources of the attacks as well as creating new counter measures for preventing future re-occurrences of the attacks.

Unfortunately, security devices have limitations which subsequently make it difficult to manually validate and utilize the event logs they generate [21]. This thesis describes a research study focused on improving attack detection by proposing analytical methods for investigating the logged events generated by such security devices. Through out this research, an emphasis is placed on the Intrusion Detection System (IDS). In this research,

a set of statistical and data mining techniques are proposed for the effective analysis of IDS events. These techniques are applied to provide an interpretation to security data and to furthermore discover behavioural patterns unique to remote attacks carried out over a period of time.

The rest of this chapter is outlined as follows: the motivation behind this research study is discussed in Section 1.1, the goals of this thesis are outlined in Section 1.2, the research contributions are discussed in Section 1.3, the publications produced during this research study are detailed in Section 1.4, and the outline of the rest of this research study is described in Section 1.5.

1.1 Motivation

Intrusion Detection Systems are network security devices which monitor network traffic for malicious activity. When malicious network traffic is detected, one or more alerts are triggered. Intrusion Detection Systems have limitations [77, 22, 23]. The consequence of their limitations is that the security level of the computer network which they protect becomes compromised and the network's resilience to attacks weakens. In most computer networks, this issue is addressed by deploying multiple layers of varying security devices at different perimeters of the network [21]. Thus, no single device is responsible for preventing all levels of an attack.

This integrated solution may improve the network's attack resilience and allow devices to focus on specialized areas of an attack. However, this does not address each devices performance issues. In prior research studies, [11], two major performance issues of an IDS have been identified:

1. *False negative rate:* The false negative rate of an IDS is the number of attacks that go undetected on the network. Typically, this can be a low value. However, the cost of false negatives, even if low, could be highly expensive to the network as it could result in asset, financial, or service loss. Today's attacks are stealth, sophisticated and difficult to detect. Many major attacks involve "zero day exploits" which are exploits or vulnerabilities that are only known to the defence community after the

attack has occurred.

2. *False positive rate:* The major performance issue that makes using an IDS difficult for attack detection is its high false positive rate [22, 23, 77]. Irrespective of the algorithmic approach used by an IDS to detect attacks, IDSs are likely to have a high false positive rate [47]. This means that a large volume of alerts are triggered whereby a large percentage of these alerts are not relevant to a real threat or attack. False positives make it extremely difficult for security analysts to utilize and prioritise the alerts generated by an IDSs. In many scenarios, false positives are a result of many factors such as IDS mis-configurations, traffic overload, and inconsistent detection methods [12].

Cuppens et al. [47] identified that a promising approach to reducing false positives and increasing the detection rate of attacks was to develop a *"cooperation module between several IDS to analyse alerts and generate more global and synthetic alerts"*. This idea is based on the notion that intrusion alerts should not be evaluated as singular events in isolation, but rather as related events which may correspond to different phases of the same attack. Thus, understanding the relationship between multiple alerts will add clarity and context to the higher level attack they represent. This notion is also supported by key pioneering research studies [116, 47, 49, 72]. In this research, we focus on the generation of these high-level global and synthetic alerts.

Motivated by this, this research study explores an alternative approach to intrusion alert analysis. It explores how to eliminate false positive alerts and increase attack insight by using knowledge gained from alert correlation - the method for identifying and aggregating alerts related to the same attack.

Therefore, our research question is as follows:

How can the information learnt from alert correlation be used to reduce false positive alerts? Furthermore, how can this knowledge be transformed into attack insights in order to increase the attack detection rate of a network?

1.2 Goals

The key goal of this research study was to improve attack detection by focussing on reducing the false positive rate of Intrusion Detection Systems. Successfully achieving this would, additionally, lead to achieving the following:

1. *Increased Attack Insight:* IDSs trigger a large amount of alerts which if effectively utilized can provide a detailed overview of network attacks and their impact on the security state of the network. Although IDSs have a high false positive rate, they also have a high true positive rate. According to Verizon [168], up to 84% of remote attacks can be discovered by analysing network events. The presence of false positives, however, make it difficult to distinguish the false positives from the true positives thus reducing the ability of any analytical process to detect attacks. Reducing the false positives allows the true positives to be effectively utilized to understanding attacks.

2. *Optimised Attack Analysis:* Due to the sheer volume of alert traffic generated by the IDSs in the network, comprehending the scope of a network attack by an analyst would require hours of manual labour even if all false positive alerts were eliminated. It is also infeasible to fully automate the process of attack analysis using computational intelligence. Achieving a balance between human and computational intelligence in the process of attack analysis is an active open ended issue in information security. Many previously proposed methods such as [79, 75, 13, 29] include using data mining and machine learning approaches which are supervised by user input and driven by statistical and empirical based learning models.

3. *Research Gap:* An observation made during the preliminary stage of this research study was that there are two key approaches to achieving improved intrusion detection. While pioneering researchers such as [118, 166] focused on post-intrusion analysis which is the analysis of *alert data* in order to reduce false positives, more recent effort has focussed on pre-intrusion analysis. Pre-intrusion analysis is the

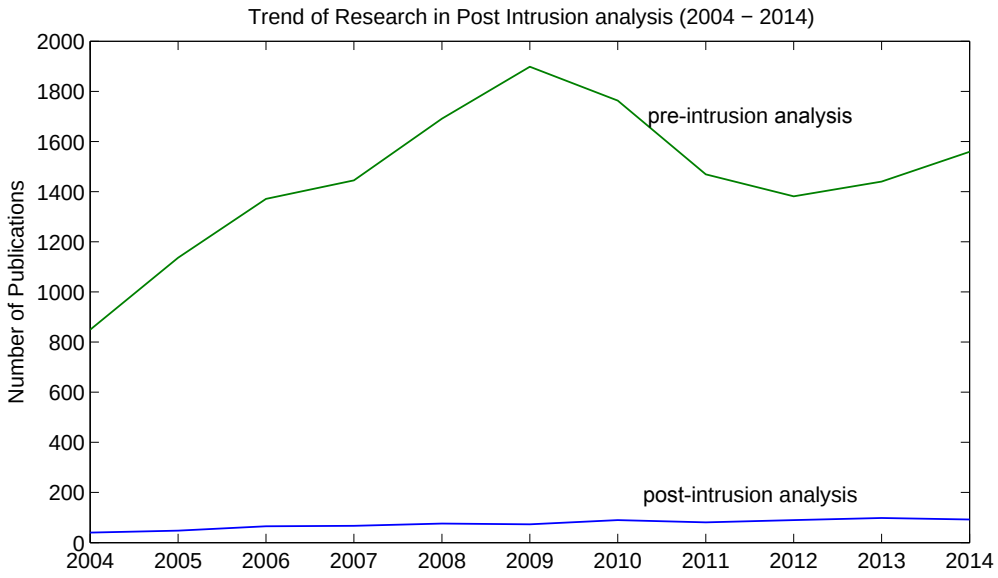

Figure 1.1: Trend of Research in Intrusion Analysis

analysis of *network data* to reduce false positives. Despite the increase in the research of post intrusion analysis as shown in Figure 1.1, the trends show that more research effort has consistently focussed on the research of pre-intrusion analysis[1]. In this research study, a key objective was to explore the domain of post-intrusion analysis.

1.3 Contributions

Improved Semi-automated Alert Correlation Models: Given a log or stream of intrusion alerts, the motive behind alert correlation is to discover temporal sequences of alerts which are part of the same attack [29]. This subsequently improves attack insight. While pioneering research approaches successfully used rule based techniques for alert correlation, such methods are unsuitable for the scope and complexity of modern attacks and networks. In this research, a set of non-rule based methods are explored and two models for correlation were proposed. Both models are based on statistical correlation using Bayesian interference which provides high quality correlations with little user configurations.

[1]Trend is based on results collected from IEEEXplore (ieee.org/ieeexplore), ACM (dl.acm.org), LNCS(link.springer.com) and ScienceDirect (sciencedirect.com) using search term "alert correlation" for post intrusion analysis and "intrusion detection" for pre-intrusion analysis

A Set of Novel Prioritisation Metrics: A key observation following the application of alert correlation was that in real computer networks, many attacks are likely to occur simultaneously, thus applying alert correlation on a large database, log or flow of security alerts will likely result in the discovery of multiple attacks occurring simultaneously. Given this observation, a set of metrics based on both domain knowledge and pattern analysis are proposed. These metrics facilitate the prioritising of non-trivial alert sequences and the elimination of sequences with little or no importance.

Automated Discovery of Attack Patterns using Graph Clustering An attack graph is a succinct representation of all paths through a system that end in a state where an intruder has successfully achieved his goal [74]. A novel approach for defining components of an attack graph is proposed. In this research, a set of attributes consistently observed in a type of attack such as a *denial-of-service* or a *worm outbreak* is referred to as an *attack pattern* where one or more attack patterns may be components of an attack graph. This contribution focusses on an automated approach to extracting attack patterns from a set of alert sequences. Furthermore, given a set of defined attack patterns, we present a real-time clustering system which categorises new alert sequences in attack clusters. This contribution facilitates the possibility of early threat detection allowing security analysts to mitigate threats earlier. To the knowledge of the author, this is a novel area of research given that it implores a bottom up approach to constructing attack graph components.

1.4 Publications

- Shittu, R., Healing, A., Bloomfield, R., and Muttukrishnan, R. (2012). Visual analytic agent-based framework for intrusion alert analysis. *Proceedings of the 2012 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2012*, pages 201–207

- Rowlingson, R., Healing, A., Shittu, R., Matthews, S. G., and Ghanea-Hercock, R. (2013). Visual Analytics in the Cyber Security Operations Centre. *Proceedings of The Information Systems Technology Panel Symposium on Visual Analytics*

- Shittu, R., Healing, A., Ghanea-hercock, R., and Bloomfield, R. (2014). OutMet : A New Metric for Prioritising Intrusion Alerts using Correlation and Outlier Analysis. *19th IEEE Conference on Local Computer Networks*

- Shittu, R., Healing, A., Ghanea-Hercock, R., Bloomfield, R., and Rajarajan, M. (2015b). Intrusion alert prioritisation and attack detection using post-correlation analysis. *Computers & Security*, 50:1–15

- Shittu, R., Healing, A., Ghanea-hercock, R., Bloomfield, R., and Muttukrishnan, R. (2015a). Real-time Graph Clustering for automatically discovering novel attack patterns. *Network and Computer Applications (Under Review)*, pages 1–25

1.5 Outline

The reminder of this thesis is organised as follows:

Chapter 2 provides a background study on intrusion detection taxonomies, the characteristics of the types of attacks detectable by IDSs, and a detailed survey on pioneering and state of the art techniques which are applied in the domain of intrusion alert analysis focussing particularly in the area of alert correlation and prioritisation.

Chapter 3 presents the two statistical correlation approaches proposed in this research. Both methods explore how alerts can be correlated in real-time by learning correlation likelihoods from historic alerts through the application of Bayesian inference. In both models, the application of Bayesian inference learns and updates the likelihoods of alert types co-occurring. To improve the dependencies discovered by Bayesian inference, we explore how weighted scoring and rule matching can be used to set thresholds to ensure only valid information is considered in the learning of likely correlations.

Chapter 4 presents four novel prioritisation metrics based on temporal and attribute characteristics that define the importance of an attack. In this chapter, each metric is evaluated

by measuring its effectiveness in correctly prioritising high priority alerts in denial-of-service attack scenarios and worm propagation attack scenarios. Finally a method for combining the results from each metric into a single metric was explored.

Chapter 5 presents two research studies. Firstly, it explores the clustering of similar attack behaviour discovered during alert correlation into clusters. In this study, a real-time clustering algorithm for categorising multi-attributed Directed Acyclic Graphs (DAG) whereby a typical alert correlation model outputs multi-attributed DAGs is proposed. Secondly, it explores improved visual representations of attack behaviour. In this second study, a new approach for refining and abstracting alert correlation DAGs into human readable attack pattern graphs is explored.

Chapter 6 presents a set of visualisations and interactive methods for effective visual analysis which can be combined with the correlation, prioritisation and attack pattern discovery techniques to allow input of a security analyst to explore the results, give valuable input during stages of the analysis and draw actionable conclusions regarding the network state. The result of this Chapter is the development of a visual analysis framework integrated into and tested in a commercial Cyber Security Event Analysis Software System distributed by British Telecom.

Chapter 7 concludes this thesis with a summary of its key achievements, challenges and open ended research questions which may be relevant to future research studies.

Chapter 2

A Survey on Intrusion Detection Systems

2.1 Overview

According to Heady et al. [69], an Intrusion is *"any set of actions that attempt to compromise the confidentiality, integrity or availability of a computer network"*. While an intrusion could be either physical or remote, this research focusses on remote intrusions. A remote intrusion is an attack or violation on a computer network from one or more machines connected to the computer network. Through carefully inspecting network traffic, such intrusions can be detected.

The main challenge in the detection of intrusions is that in recent years, remote attacks have advanced in stealth and sophistication by becoming more coordinated, distributed and persistent thus making them more difficult and complex to detect and distinguish from normal network traffic. As described in Section 1.1, IDSs, which are responsible for detecting intrusions, have significantly high false positive rates.

The objective of this chapter is to survey the state-of-the-art of intrusion detection in order to identify specific factors affecting the performance of the IDS, identify the pioneering and state of the art analytical techniques used to reduce the false positive rate of an IDS through analysing security alerts and further investigate how these approaches can be improved on. The rest of this chapter is organised as follows: In Section 2.2,

the different categories of intrusions are identified with the intention of categorising their characteristics into attack classes. Section 2.3 and 2.4 introduce the classifications and components of IDS and identifies their strengths and limitations as well as most recent potential solutions to these limitations. Finally, Section 2.5 to 2.7 focus specifically on Signature-based Network IDSs, and details the analytical techniques used in post intrusion alert analysis of Signature-based Network IDS alerts.

2.2 Intrusion Classifications

As defined in Section 2.1, an intrusion is any set of actions which compromise a computer network. This definition covers a wide scale of actions. Therefore, prior to reviewing the state of the art in intrusion detection, it is important to clearly comprehend the scope and scale of intrusions. During this research study, ten key classification models for categorising intrusions were investigated. These are detailed in Table 2.1.

The classification model described by Baumerucker et al. [27] provided three mutually exclusive attack classes. Due to its simplicity, this classification model is used to describe the intrusions covered by this study.

Table 2.1: Pre-existing Classification Systems for Intrusions

Reference	Year	Classes of Intrusions
Zhou et al. [182]	2010	Stealth scans, Worm outbreaks, Denial of Service.
Baumerucker et al. [27]	2003	Reconnaissance, Access, Denial of Service.
Cheswick et al. [44]	2003	Information leakage, Stealing passwords, Social engineering, Bugs and Backdoors, Authentication failures, Protocol failures, Denial of Service.
Hansman and Hunt [68]	2005	Information Gathering, Physical Attacks, Network Attacks, Trojans, Buffer Overflows, Worms, Viruses, Password Attacks, Denial of Service
Weber [172]	1998	Probing, Interception, Modification, Illegal System Access, Denial of Service.
Kruegel et al. [85]	2005	Surveillance(reconnaissance), Exploitation, Masquerading.
Stallings [160]	2007	Interception, Interruption, Modification, Fabrication.
Neumann and Parker [113]	1989	External misuse, Hardware misuse, Masquerading, Setting up subsequent misuse, Bypassing intended controls, Active misuse of resources, Passive misuse of resources, Misuse resulting from inaction, Use as an indirect aid in committing other misuse.
Denning [52]	1987	Attempted break-in, Masquerading or successful break-in, Penetration by legitimate user, Leakage by legitimate user, Inference by legitimate user, Trojan Horse, Virus, Denial of Service.

2.2.1 Reconnaissance Intrusions

Hutchins [72] and Baumerucker et al. [27] refer to a reconnaissance intrusion as an act that is carried out to learn about the computer network illegitimately. In work by Cheswick et al. [44] and Hansman and Hunt [68], such intrusions are classified as *information gathering* and *leakage* intrusions. These intrusions mainly involve ping sweeps, scans and probs which guess and scan entire ranges of potential IP addresses, ports and services in order to establish which ones are running on the network. For example, an attacker may have knowledge that the IP network domain 172.16.48.0/20 belongs to an organisation. In order to discover which hosts are running, the attacker may scan the entire range of IP addresses 172.16.48.0/20 - 172.16.63.255/20. When one or more hosts are identified, the attacker may ping the hosts on common ports such as 21, 23, 25, 53, 80. A response from the host on one of these ports indicate that the host uses FTP, Telnet, SMTP, DNS, HTTP protocols (respectively). With knowledge of these protocols, one can determine which applications may potentially be running on the host and which may be vulnerable to attacks.

Ping Sweeps can easily be automated using tools such as Nmap [1], Metasploit[2], and various sub-tools in pen-testing environments such as Backtrack [3]. While this makes it easy to detect Scans, internal network assets such as servers and workstations may legitimately scan the network for running services, therefore careful observations are required in order to distinguish a legitimate scan from an illegitimate. We note that physical reconnaissance activity also exist. However, since these types of activities cannot be detected by an IDS, they are outside the scope of this research.

In many scenarios, reconnaissance intrusions are early steps of a coordinated progressive attack to identify the computer network's weaknesses before later attempting to exploit these weaknesses.

[1] Nmap - Free Security Scanner For Network Exploration - nmap.org
[2] Metasploit - Penetration Testing Software - www.metasploit.com
[3] BackTrack Linux - Penetration Testing Distribution - www.backtrack-linux.org

2.2.2 Access Intrusions

These intrusions and attacks involve the compromise of the networks assets such that access is gained to the asset by an attacker remotely. A close review of the classification models described in Table 2.1 shows that most intrusions fall under *access intrusions*. For example, in the classification system presented by Cheswick et al. [44], password attacks, buffer overflows, worms, trojans, viruses, and authentication/protocol failures all fall under access intrusions. Access intrusions exploit vulnerable services or applications running on hosts or servers. In some cases, such as with worms, trojans and viruses, an attacker uses the identified vulnerability in the service to manipulate the service into executing malicious code. Once such code is executed, this could provide an attacker control over the host. Unlike reconnaissance intrusions, due to the stealth involved in delivering access intrusions and the large number of variations of many known access intrusions, these intrusions can be difficult to detect. Section 2.3.2 describes how different intrusion detection systems discover access intrusions.

According to CERT-UK mid-year 2014 report, more than 76% of 2014's attacks were made up of worms (malware), web vulnerabilities, and infrastructure compromise [41].

The most common types of web vulnerabilities are found in browsers such as Apple Safari, Google Chrome, Mozilla Firefox, and Microsoft Internet Explorer [19]. Some known services and applications which also contain well-known vulnerabilities commonly exploited include: Microsoft Word [57], SQL, Adobe Flash Player, Oracle Sun Java, Adobe Acrobat Reader, and Apple QuickTime [19].

2.2.3 Distributed Denial Of Service Intrusions

Denial of service intrusions and attacks cause a disruption in the operations of the targeted network. This normally involves overloading one or more hosts on the targeted network such that their resources are consumed and the target host can no longer operate efficiently. Modern DoS attacks are distributed denial of service (DDoS) attacks whereby a number of hosts are controlled remotely by a single attacker to take a victim host off-line.

While DDoS attacks are critical, the CERT-UK report for mid-year 2014 showed that the number of DDoS attacks are declining [41]. This is most likely because the objective

of attackers is more concentrated on stealing valuable assets from the network rather than sabotaging the network's operations.

A common DDoS is the SYN attack where the TCP handshake is exploited. In this type of attack, the attacker uses multiple clients to request a service from the targeted host, this establishes the TCP handshake. None of the clients would complete the TCP handshake with the target, therefore leaving many open connections on the target which could exhaust the target's resources and lead to a DDoS on the target. Other types of DDoS attacks include High Ping Volumes and Smurf Attacks[4].

2.3 IDS Taxonomies

Intrusion Detection Systems (IDS) detect intrusions by monitoring incoming and outgoing traffic for internal and external attacks. Figure 2.1 shows a sample network and examples of where an IDS may be physically situated on the network.

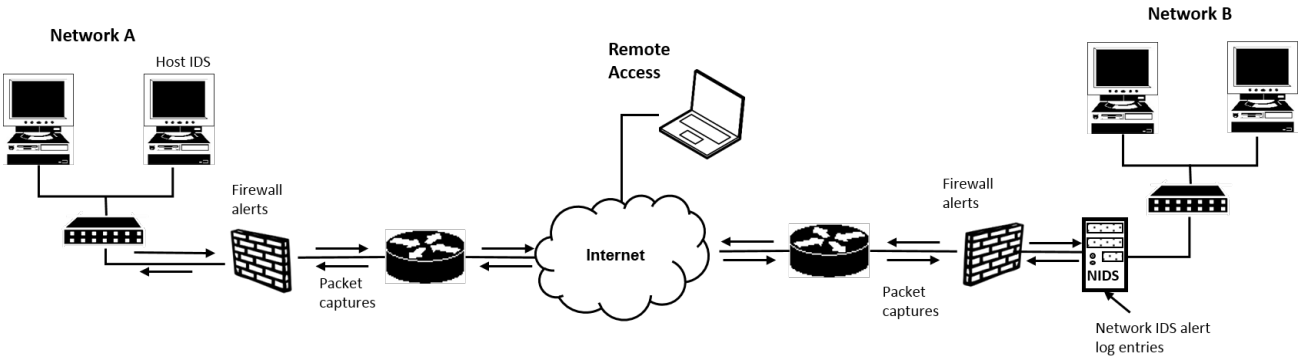

Figure 2.1: A Sample Computer Network

When an IDS detects a network packet which is potentially an intrusion activity, the IDS logs one or more alerts. While an IDS may log an alert which holds a reference to the network packet that triggered the intrusion, the information contained at the alert level differs to the information contained at the network packet level. Table 2.2 shows a comparison between the data logged by an IDS in comparison to packet flows.

To add additional context to the definition of an intrusion, in this study, we identify two levels of an intrusions. The primary intrusion alerts triggered by an IDS are referred to as low level intrusions. Low-level intrusions are detected by analysing small numbers

[4]Datasource: IBM ISS Site : http://www.iss.net/security_center/reference/vuln/Smurf.htm

Table 2.2: Comparison of Relevant Data Fields in an IDS Alert versus Network Packet

	Timestamp	Header Version	ToS	Identification	Flag	Offset	TTL	Protocol	Checksum	Source IP	Destination IP	Intrusion Type	Intrusion Category	Option	Datagram	Datagram Length	IP Length	Priority
Intrusion Alert	*		*	*			*	*		*	*	*	*			*	*	*
Network Packet	*	*	*	*	*	*	*	*	*	*	*			*	*			

of network packets. The second level of alerts are the *"global and synthethis alerts"* which are high level alerts as described by Cuppens and Miège [46]. In this research, we focus on the generation of these high-level global and synthetic alerts but first, we investigate low level intrusions. In most IDS models such as Snort [5], an IDS consists of a single key component which is a detection unit consisting of one or more sensors that monitors the network traffic and triggers low-level intrusion alerts. IDSs can be classified using characteristics of their detection units. Two classification taxonomies are described.

2.3.1 A Location based Taxonomy

Network Intrusion Detection Systems (NIDSs)

This taxonomy classifies the type of IDS based on the physical location of the IDS on a computer network. Figure 2.1 shows two types of IDSs - Host IDS (HIDS) and Network IDS(NIDS). Network A deploys a Host IDS while Network B deploys a Network IDS. Network Intrusion Detection Systems (NIDSs) are situated at the perimeter of a computer network, typically, behind a firewall (shown in Figure 2.1). NIDSs sniff packet captures at the network layer and use either the anomaly detection or signature based approach. When one or more related packet captures or traffic flows are identified as attacks or network violations, an alert is raised and logged in an alert log.

Since an NIDS is situated at the perimeter of the network, it provides a detailed log of the detected malicious activity on the network area it covers. NIDS are very critical to the security of a computer network and most computer networks have one or more IDSs

[5]Snort - https://www.snort.org/

deployed. A range of commercial IDSs include Snort[5], Suricata IDS [6], Juniper IDP [7], and Cisco IDS Sensors[8].

A limitation of an NIDS is that due to its location on the network, it monitors a large amount of network traffic. According to [12, 78, 182] studies on NIDS performances, the performance of an IDS deteriorates under intense traffic, particularly those that are *centralized*. A distributed architecture is less affected by this since the workload is shared by two or more NIDSs. This performance deterioration leads to increase in false positives (up to 90%) and an increase in true negatives [22]. An additional key limitation is that NIDSs do not provide application information such as the applications running on the target host [27].

Host Intrusion Detection Systems

HIDS reside on network servers or hosts as software applications. HIDSs also inspect packet captures at the network layer but only packets on its network interfaces therefore only packets related to the HIDS's host. This also implies the HIDS host is the target of the intrusion. HIDSs are suitable for detecting access intrusions targeted on the operating system and application vulnerabilities of the servers. HIDSs also inspect OS binaries, system logs and capture failed login attempts [27].

2.3.2 A Technique-based Taxonomy

Signature-based IDS

This taxonomy classifies IDSs based on the detection technique used to discover intrusions. Signature-based IDSs (also known as Misuse-based IDSs), use a database of rules which describe various attacks. Using the database, a signature-based IDS labels one or more network packets as an intrusion if the network packet(s) match one of the rules in its database. Signatures are also assigned homogeneous priorities based on the classification of the attack or violation. In most cases, signatures that identify reconnaissance intrusions are assigned lower priorities than those which identify access and DDoS intrusions.

[6]Suricata - suricata-ids.org/
[7]Juniper IDP - https://www.juniper.net
[8]https://www.cisco.com

A Signature-based IDS can be an NIDS or HIDS.

Reconnaissance intrusions can easily be detected by a Signature-based NIDS without inspecting the actual payload of a network packet. For example, when using NMAP to scan a network, NMAP will typically send an Echo message to a host to test if it is alive. The packet payload is typically empty and the ICMP type value will be 8 (code for Echo messages) [9]. A signature that tests the packet size and ICMP type value as well as some additional attributes will detect reconnaissance. The primary challenge is that some legitimate scans may also be categorised as malicious.

Access intrusions and DDoS intrusions can also be detected by Signature-based NIDSs. Access intrusions can be detected by inspecting the headers for invalid content or the payload of the packet in search for malicious content such as urls, executable code etc. DDoS Intrusions can also be detected by evaluating the packet's flag attributes, content, ICMP type, ICMP sequence and more.

Anomaly-based IDS

An Anomaly-based IDS can also be NIDS or HIDS. Anomaly-based IDSs learn the normal state of the network by using statistical analysis to model normal network traffic behaviour and protocols. When network traffic deviates from what it models as 'normal', alerts are triggered. Attacks come in large variations and *"normal behaviour"* can be difficult to model by a statistical or computational model. As a result Anomaly-based IDSs trigger a large volume of false positives. Due to the high magnitude of false positives, signature-based IDSs are more widely accepted and deployed in real networks. This was an issue identified over a decade ago by Axelsson [22] and to date, remains a challenge.

2.4 IDS Correlation Unit

In Section 2.3 it was mentioned that most IDSs consist of a single key component which is the detection unit. Many IDSs contain additional components for alert management,

[9] Attack description is based on Snort rule Sid-1-469 : https://www.snort.org/rule_docs/1-469

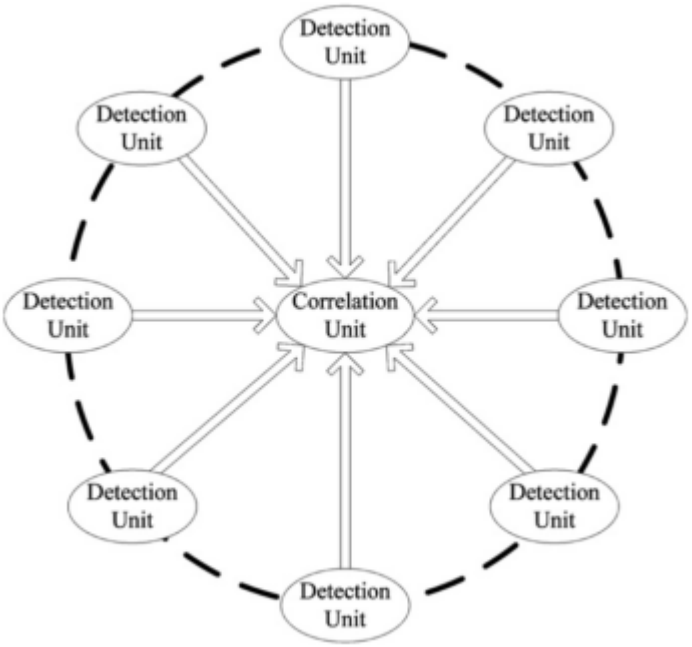

Figure 2.2: A Centralised IDS framework with multiple detection units and a single correlation unit Zhou et al. [182]

reporting and situational awareness. These components allow the detection of the second level of intrusions (also referred to as high level intrusions) which are the result of modern network attacks that evolved from persistent, progressive and collaborated low-level intrusions. More recently, in order to address this issue, various work such as [66, 49, 166, 182, 56, 146] proposed an additional component referred to as a correlation unit which is either integrated into the IDS or tightly coupled with the IDS. Figure 2.2 illustrates a framework for an IDS with an integrated correlation unit proposed by Zhou et al. [182].

The correlation unit transforms the low-level intrusion alerts into high level intrusion information which allows network security experts to determine the security state of the network. Alert correlation is based on the notion that *"most (high-level) intrusions are not isolated, but related as different stages of attacks, with the early stages preparing for the later ones"*. This theory is supported in work by [119, 72]. In this section, the processes involved in the correlation unit are detailed.

A taxonomy proposed by Salah et al. [146] (shown in Figure 2.3 and 2.4) classifies the sub-components of the correlation unit into four key sub-components.

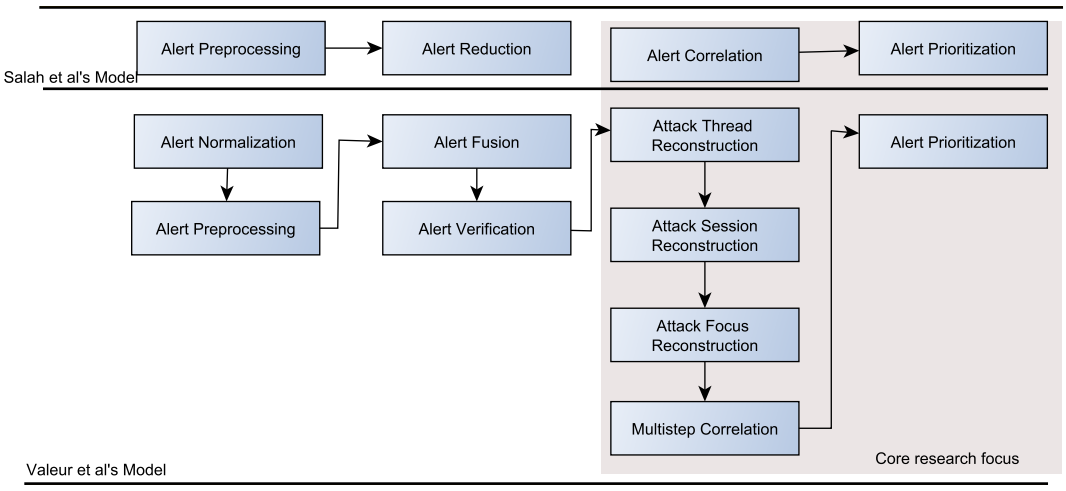

Figure 2.3: A Comparison of the Correlation Modules proposed by Salah et al. [146] and Valeur et al. [167]

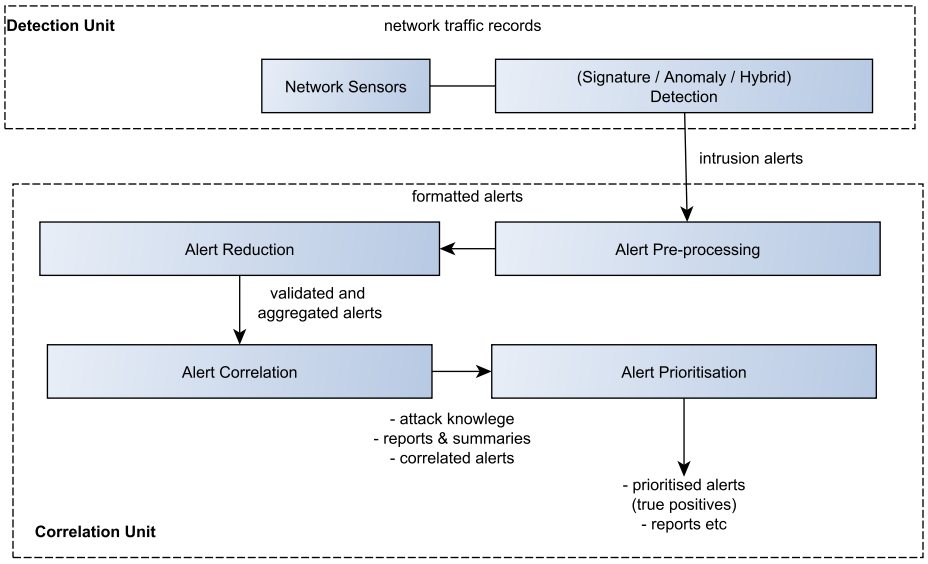

Figure 2.4: The components of a Correlation Unit as proposed by [146]

2.4.1 Intrusion Alert Pre-processing

This is the first component of a correlation unit. It receives alerts triggered by the Detection unit of the IDS and is responsible for converting the alerts into a single uniformed data structure in preparation for analysis. Pre-processing is vital in computer networks were multiple IDSs of different platforms are deployed across the network. Often, these alerts are all collected and analysed in a single environment. The CBE (Common Base

Event) is the standard introduced by IBM[10] to use as a format for logging multiple events from heterogeneous devices in the same log format [127]. Also, MITRE[11] proposed CEE for the same purpose [107]. IDMEF is a common format for unifying alerts from different IDS models which though yet to be formalised is till vastly used [12].

2.4.2 Intrusion Alert Reduction

This receives alerts formatted by the pre-processing component and filters *duplicate* and *irrelevant* alerts. Debar and Wespi [50] describe duplicate alerts as alerts triggered when more than one IDS raises an identical alert over the same network packet(s). Irrelevant alerts are typically redundant and otherwise referred to as false positives because they refer to either an alert raised on normal network events/packets, an alert caused by the underlying configurations of the network or simply an uninteresting event. As previously emphasised in Section 1, IDSs are well known for raising a high volume of irrelevant alerts [22]. Uninteresting alerts are defined according to the security interest of the network e.g. thus it may not be redundant, but simply uninteresting to the analyst. In many real systems, the rate of redundant and uninteresting alerts can be as high as 99% hence alert reduction is a key area of research [22].

2.4.3 Intrusion Alert Correlation

This receives alerts from the reduction component and groups alerts related to the same network attack into a single higher level event. A wide range of network data sources provide information for correlating alerts. In addition, a wide range of analytical methods can be applied for correlating alerts. These are discussed in depth in Section 2.5. By correlating alerts, insight is gained into the types of attacks being carried out across the network - whether they are benign or highly severe.

2.4.4 Intrusion Alert Prioritisation

This receives alerts from the correlation component and uses the knowledge gained by the correlation component to prioritise the alerts such that alerts with higher priorities

[10]IBM - www.ibm.com
[11]MITRE - www.mitre.org
[12]The Intrusion Detection Message Exchange Format (IDMEF): http://www.ietf.org/rfc/rfc4765.txt

are attended to first by a security analyst. Both the correlation and prioritisation sub-components of the correlation unit include a broad range of analytical techniques and are much more complex to perform. Both these sub-components are discussed in more depth next.

2.5 A Taxonomy for Alert Correlation Types

2.5.1 Introduction

In this section, a new taxonomy for classifying alert correlation units based on the type of data used is discussed. Based on observation from prior art such as [167], four types of data sources are commonly applied during the correlation of security alerts.

2.5.2 Alert to Vulnerability Knowledge Correlation

One of the earliest descriptions of this correlation type was provided by Gula [67]. When an IDS triggers an alert, in particular, an *access intrusion* alert involving an exploit, it is impossible to know at the NIDS level if the access intrusion was successful or not. Correlating alerts to the vulnerability information of the targeted host can provide this information. Figure 2.5 shows a sample intrusion alert. Lines 7 - 9 of this figure represent the CVE [13] identifiers, associated with this intrusion that was raised on the targeted host 154.241.88.201. The CVE identifier is a global standardised identifier which represents the type of exploit. Vulnerabilty scanners provide capabilities for logging all the vulnerabilities of a host including their CVE code. If the vulnerability scanner logs contains any vulnerabilities with the same CVE code then it is highly likely that the exploit on host 154.241.88.201 was successful.

```
1 [**] [1:2003099:4] ET WEB-MISC Poison Null Byte [**] [Priority: 2]
2 [Classification: access to a potentially vulnerable web application]
3 11/10-09:57:34.328310 10.2.197.245:33914 -> 154.241.88.201:80
4 TCP TTL:61 TOS:0x0 ID:26947 IpLen:20 DgmLen:346 DF
5 ***AP*** Seq: 0xCE16B56C  Ack: 0xB3CC103C  Win: 0xB7  TcpLen: 32
6 TCP Options (3) => NOP NOP TS: 1498161 138777430
7 [Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2006-3602]
8 [Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2006-4458]
9 [Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2006-4542]
```

Figure 2.5: Sample Intrusion Alert from CDX 2009 [148]

[13] Common Vulnerabilities and Exposures (CVE) [131]

Roschke et al. [140] proposed a method to improve alert correlation by correlating intrusion alerts with vulnerability based knowledge. In their work, a comprehensive list of known vulnerabilities are collected from publicly available vulnerability databases such as the OSV Database [14]. In addition, they extract system properties from hosts on the network. These system properties include the type and version of OS installed on the host as well as the type and versions of all applications running on the host. System properties, intrusion alerts and vulnerabilities in the database are correlated. For example, they explain that an intrusion alert is less critical if the intrusion the alert represents was a Linux OS Exploit targeted at a Windows OS host machine. Contrarily, if the intrusion was a Windows OS exploit it would be more critical.

2.5.3 Alert to Network Knowledge Correlation

This correlation type is used by [132] who proposed a correlation model called *M-Correlator*. Typically, this type of correlation adds meta-data to the alerts. For example, to provide more information about a host under attack, the IP address of the host can be mapped to more meaningful information about the host such as the type of services running on host, the value of the host, the geo-graphical location of the host, and the network user's who have access to the host.

2.5.4 Alert to Alert Correlation

This section is the broadest, most common and highly useful type of correlation. One of the most comprehensive standard definitions of alert-to-alert correlation is provided by NIST (National Institute of Standards and Technology)[81]. In this article, alert correlation is defined as:

> *"...finding relationships between two or more log entries. The most common form of event correlation is rule-based correlation, which matches multiple log entries from a single source or multiple sources based on logged values, such as timestamps, IP addresses, and event types. Event correlation can also be performed in other ways, such as using statistical methods or*

[14] Open Source Vulnerability Database - http://osvdb.org/

visualization tools. If correlation is performed through automated methods, generally the result of successful correlation is a new log entry that brings together the pieces of information into a single place. Depending on the nature of that information, the infrastructure might also generate an alert to indicate that the identified event needs further investigation" [81].

In the domain of alert correlation an alert generated by an alert correlation system is a high-level alert. In various research studies, such an alert has also been referred to as a meta-alert [167], hyper-alert [117], or an alert correlation graph [162, 133].

2.5.5 Alert to Attack Knowledge Correlation

Salah et al. [146] classified this correlation type into three categories namely - Ontology based correlation, case-based correlation and knowledge based representation. This correlation type is highly related to alert-to-alert correlation and is therefore described further in Section 2.6.3.

2.6 A Taxonomy for Alert Correlation Techniques

2.6.1 Introduction

A correlation technique provides a definition or function such as $f(a_1, e)$ which defines the correlation between an intrusion alert, a_1 and a network event, e such as a vulnerability alert, network information, or another subsequent intrusion alert. The output of this function or definition indicates whether the two entity arguments are correlated or not.

Taxonomies for classifying the methods in alert correlation have been proposed by Reza and Ghorbani [137], Al-Mamory and Zhang [10], Salah et al. [146], Mirheidari et al. [106] and many others. Figure 2.6 attempts to incorporate the aspects of each of their taxonomies. In addition, during this research study, it was discovered that while visual based correlation is highly active in the alert correlation and security event analysis domain, it is yet to be included in any of the mentioned taxonomies. Therefore, this research study enhances the prior taxonomies by including this method. We consider the pioneering and state of the art alert correlation techniques using the following method - an

alert correlation technique can be classified as either: *similarity based, knowledge based, sequential based or visual based*.

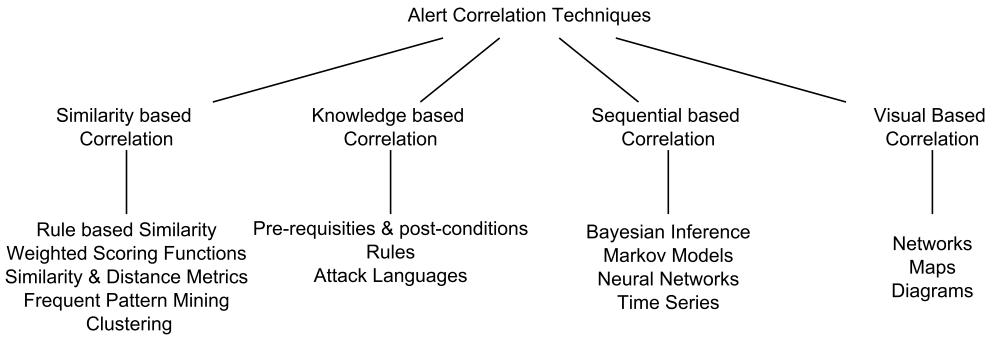

Figure 2.6: Classification of Alert Correlation Techniques based on taxonomies proposed by Salah et al. [146], Reza and Ghorbani [137] and Al-Mamory and Zhang [10]

2.6.2 Similarity Correlation

A similarity based function is used to correlate two entities based on how much their features are similar. Features may be spatial, categorical, temporal or numerical. Similarity correlations are calculated using rule-sets, weighted scorings, clustering algorithms, the complement of distance functions such as euclidean cosine, jaccard, mahalonibis distances and other similarity related metrics. Let us consider two alerts a_1 and a_2, typically, these two alerts are correlated if $f(a_1, a_2) = y$ where $0 \leq y \leq 1$ and $Y > \theta : 0 < \theta \leq 1$. θ is a threshold parameter configured according to the requirements of the correlation. If strict correlation is required, a higher threshold is applied. Figure 2.7 shows an example log of four alerts with six attributes - *intrusion type, source ip, destination ip, source port, destination port and timestamp*. We review the most applied similarity techniques for correlating such alerts.

```
1 a1: (03/07-16:28, 1.1.1.40,26582, 5.5.5.3, 25, Policy attempted download of a pdf)
2 a2: (03/07-16:28, 5.5.5.3, 3727, 10.0.0.3, 25, Policy attempted download of a pdf)
3 a3: (03/07-17:30, 1.1.1.43,48097, 5.5.5.3, 25, Policy attempted download of a pdf)
4 a4: (03/07-17:30, 5.5.5.3, 3730, 10.0.0.3, 25, Policy attempted download of a pdf)
```

Figure 2.7: Sample Alert Log Example

I. Rule based Similarity

Valeur et al. [167], and Debar and Wespi [50] used feature matching rules to determine two alerts being correlated. For example, two alerts are correlated if they share the same source IP etc. Other examples of such rules includes: *one-to-many* where alerts are grouped if they are from the same attacker targeted at multiple victims, *many-to-one* where alerts are grouped if they are from multiple attackers targeted at a single victim, and *one-to-one* where alerts are grouped if they are from a single attacker targeted at a single victim [182].

Others which have used rules to define the similarity between alerts includes work by Wang et al. [171], Sadoddin and Ghorbani [145] and Li et al. [90].

II. Weighted Scoring Coefficients

Valdes and Skinner [166] as well as Dain and Cunningham [49] first introduced the similarity weighted coefficient approach for alert to alert correlation. A set of alert attributes, $A = \{\alpha_0, \alpha_1, \alpha_2, ..., \alpha_n\}$ are considered : *intrusion type, source ip, destination ip, source port, destination port and timestamp*. For any two alerts, the similarity between their i^{th} attribute - α_i, was defined heuristically. For example, in the approach by Dain and Cunningham [49], the IP addresses was based on the common prefix bits which is depicted in Table 2.3. The common prefix similarity method gives an estimation of the logical proximity of two hosts.

Table 2.3: IP Common Prefix Length

172.16.113.20	10101100 . 00010000 . 01110001 . 11001111
172.16.115.20	10101100 . 00010000 . 01110011 . 00010100
Common Mask	11111111 . 11111111 . 11111100 . 00000000
	22/32 = 0.68

$$f(a_1, a_2) = SIM(a_1, a_2) = \frac{\sum_{i=1}^{k} W_i \times SIM(a_{1_i}, a_{2_i})}{\sum_{i=1}^{k} W_i}. \quad (2.1)$$

For the time-stamp attribute, Dain and Cunningham [49] used a sigmoid function where the similarity exponentially decreases as the time difference between the alerts increases. The general hypothesis is that the wider the time interval between two alerts, the less likely they are correlated. The time difference between the timestamps of any two alerts, a_1, a_2 is denoted as $\sigma_{a_1,a_2}(\Delta t)$ and the sigmoid function as $\frac{1}{1+e^{\alpha+\beta\Delta t}}$ where Δt is the time difference and α and β define the shape of function. The shape defines how fast or slow the curve falls and rises. In their work, they hypothesise that different intrusion types share different time relationships. For example, intrusion alerts of type *DDoS* will highly likely occur within a short period of time thus the curve should rise and fall quicker. On the other hand, intrusion alerts of type, reconnaissance followed by access alerts may occur within a wider time window, thus the curve should fall slower. A sample sigmoid curve is illustrated in Figure 2.8.

Valdes and Skinner [166] use a much simpler approach called a step function, in this case, the curve abruptly drops when the time difference between two alerts, Δt reaches a threshold value. A sample curve is illustrated in Figure 2.9.

Browne et al. [35] also hypothesised that the time relationship between alerts related to certain exploits can be modelled using a regression function : $1 + S \times \sqrt{M}$ where I and S are regression coefficients that define the shape of the curve and M is the time difference between two alerts. In the last decade, such functions have become popular in measuring the time similarity and correlation between alerts. They have been used in recent work by Marchetti et al. [102] Lee et al. [89] Shiravi et al. [152] Shiaeles et al. [151].

In weighted scoring coefficients, the overall correlation similarity is typically measured using a function similar to Equation 2.1. W_i is the weight of the i^{th} attribute and

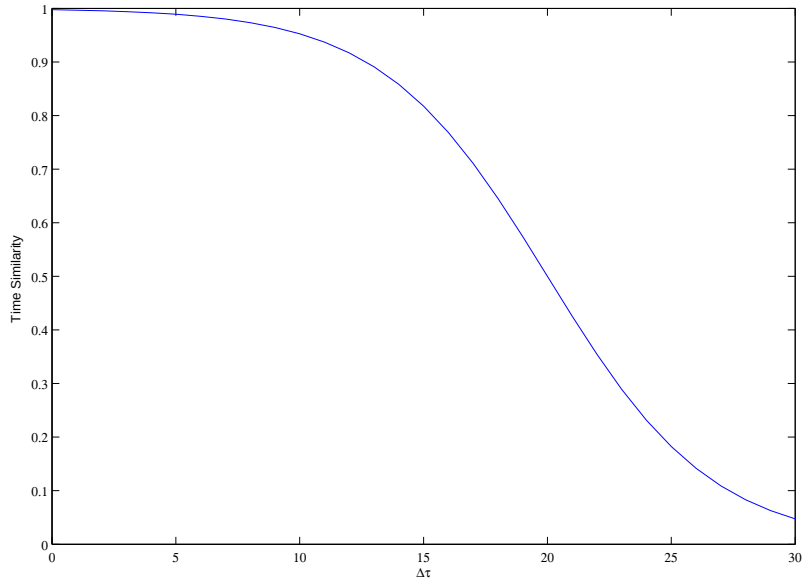

Figure 2.8: Time based similarity using Sigmoid function ($\Delta T = 15$).

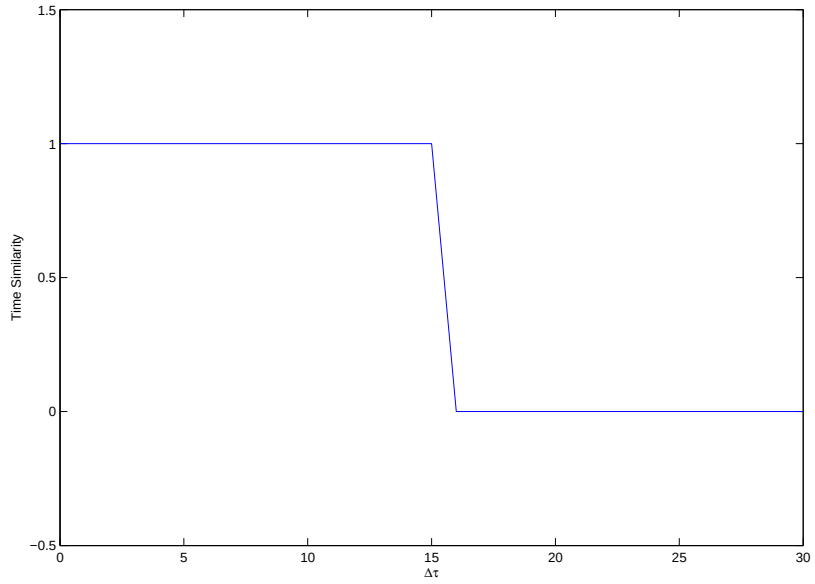

Figure 2.9: Time based similarity using Step Function ($\Delta T = 15$).

$SIM(a_{1_i}, a_{2_i})$ is similarity between the two alerts i^{th} attributes.

Others which have used the weighted similarity coefficient includes Zhuang et al. [184].

III. Distance Metrics and Clustering approaches

Distance metrics are parameters used in conjunction with clustering algorithms for grouping similar objects into clusters. Lee et al. [88] correlates network traffic. In their approach, hierarchical clustering is used to group network traffic records into the same cluster based on their similarities. They use the euclidean distance metric to calculate the distance between any two network traffic records. This is defined as:

$$d(a_1, a_2) = \sqrt{\sum_{i=1}^{n} (a_1(i) - a_2(i))^2} \tag{2.2}$$

where $a_1(i)$ is the i^{th} attribute of a_1. More Recently, Hofmann and Sick [71] and Chen et al. [42] proposed improved similarity techniques using better feature selection and real-time clustering techniques. In the approach by Hofmann and Sick [71], two alert correlation component's are considered. An offline component is used to cluster a set of alerts into M clusters using the Expectation Maximisation Clustering algorithm [51]. The algorithm requires a distribution be defined over the dataset for measuring the variance in the alert attributes, thus defining a joint distribution which is the product of a multinomial distribution for categorical attributes and a Gaussian distribution for the continuous attributes. In the on-line process, a continuous stream of real-time alerts are received and each alert in the stream is added to one of the M clusters to which it is mostly related to in distribution.

A major challenge in applying distance metrics to network traffic and alert analysis is that most of the data attributes such as IP addresses and ports are categorical. Standard distance metrics, however, are most suitable for numerical values. In most cases, heuristics, such as those used in the weighted coefficients sections are applied to deriving distances between categorical attributes.

IV. Frequent Pattern Mining

Frequent pattern mining has been popularly used to discover alert correlation rules and attack patterns. Bai et al. [24] applied FP-Growth, a pattern mining algorithm on an alert database in order to discover frequent attack patterns. Jinghu et al. [75] proposed a more

effective approach to mining attack patterns from correlated alerts. In their work, an algorithm DGSpan based on the standard GSpan [176] was proposed for the mining of frequent subgraph patterns from alerts. Frequent sub-graphs are then referred to as attack patterns which are used as the basis of generating attack graphs. Others who perform sequential mining on alert data using various adaptations of association rule mining and the a-priori algorithm [7] in order to extract attack patterns (i.e. attack scenarios) include [92, 43, 91, 97, 171] and [82].

The challenge with frequent pattern mining is that frequent pattern mining is computationally expensive both in memory and time [174, 1]. As the size of the dataset to be mined for patterns increases, the computational time and memory required exponentially increases [174].

V. Other Mining Techniques

Yang et al. [177] proposed a function for correlating alert features by using a Q-gram algorithm based on String matching. While it is not quite clear, it is assumed that alert log entries are treated as strings and the similarity between two alert strings is derived by extracting all sub-strings of length-3 in each alert string. A value between 0 and 1 is derived from measuring how similar the substring between the two alerts are.

Challenges in Similarity based Correlation

While similarity methods are simpler to implement, such methods do not capture complex nor hidden correlations. This is typically due to the fact that complex attacks are stealth and distributed and the intention of the attacker is to go unnoticed. Thus, various phases of the attack would be carried out across a number of heterogeneous hosts and the attacker would apply techniques such as IP Spoofing, to mask the relation between the hosts. In this case, the alerts triggered as a result of the attack may not have similar features in common.

2.6.3 Case based Correlation

Some of the earliest pioneering research in alert correlation used case based methods. In this method, expert knowledge is used in the process of determining whether alerts are relational or causally correlated by explicitly or implicitly defining scenarios, classes or instances of attack groups.

Scenarios involve a rule language that defines complex events that may occur in the event of a type of attack. Each alert type is associated with one or more attack types or intrusion objectives and alerts are correlated based on how well they fit together into the same scenario.

Steven Eckmann [161] proposed STATL, a state-transition base attack detection language which allows a user to define an attack scenario. A STATL definition represents an attack scenario consisting of a set of states where a state represents a phase or action of the attack, and a transition represents a progression in the attack. Each state definition has an initialisation state and at least one ending state. A set of real-time alerts will be correlated to a STATL attack definition if they each satisfy a set of conditions.

Cédric Michel [40] proposed ADELE, an XML based attack descriptive language for describing the phases of various attacks and correlating alerts to these definitions. Their approach used a set of pre-conditions that must be met for an alert to be correlated with an ADELE attack description, it also includes a set of post-conditions describing what has been achieved by the attacker. Other similar correlation methods include LAMBDA and CAML, both proposed in [48] and [45] respectively. Figure 2.10 shows a snippet from an ADELE definition of an "NFS MOUNT" attack and Figure 2.11 shows a definition of a LAMBDA "NFS ABUSE" attack.

The Snippet in Figure 2.10, shows that the ADELE language describes that for the high level intrusion called "NFS MOUNT" to occur, a *precondition* is to be satisfied. This precondition states that the intruder must have remote access on the target. The *attack*, defined using the <*attack*> tag is then described from line 6. In particular, the attack description states that for this type of attack, the attacker is expected to execute shell commands using the listed variables.

```
1   Alert NFS_Mount (IN IPaddr targetip, OUT String account, OUT Connection cnx) {
2   <EXPLOIT>
3     <PRECOND>
4       Accesslevel == "REMOTE" #initial access level required
5     </PRECOND>

6     <ATTACK> <LANG> "EDL" </LANG>
7       String output;   #gets everything displayed in the console
8       Integer ret_val;  #exit code for the command
9       String rpc_services;
10      Integer ret_val0;
11      Connection shellhandler;

12      EVENT E0{
          #Exec_shell_cmd(<shell_command>,<console_output>,<return_value>)
13        Exec_shell_cmd("rpcinfo -p "+targetip,rpc_services,ret_val0);
14      }
```

Figure 2.10: Snippet from an ADELE definition of an Attack Scenario [40]

attack $NFS\_abuse(\text{Target-IP})$
pre : $remote\_access(A, H) \wedge ip\_address(H, \text{Target-IP})$
$\wedge\ use\_service(H, portmapper) \wedge use\_service(H, mountd)$
$\wedge\ exported\_partition(H, P) \wedge mounted\_partition(H, P)$
$\wedge\ connected\_user(U, H) \wedge userid(U, H, Userid)$
$\wedge\ use\_service(H, fingerd) \wedge root\_user(A, H_A)$
$\wedge\ connected\_user(A, H_A) \wedge owner(Directory, U)$
post : $can\_access(A, Directory)$
scenario : $((E_1 ; (E_2 \,\&\, E_3)) \,\&\, E_4 \,\&\, E_5) ; E_6$
where $action(E_1) = \texttt{rpcinfo -p}\ \text{Target-IP}$
$\wedge\ action(E_2) = \texttt{showmount -e}\ \text{Target-IP}$
$\wedge\ action(E_3) = \texttt{showmount -a}\ \text{Target-IP}$
$\wedge\ action(E_4) = \texttt{finger @}\text{Target-IP}$
$\wedge\ action(E_5) = \texttt{adduser --uid}\ Userid\ U$
$\wedge\ action(E_6) = \texttt{mount -t nfs}\ P\ \texttt{\textbackslash mnt}$
$\wedge\ actor(E_1) = A \wedge actor(E_2) = A$
$\wedge\ actor(E_3) = A \wedge actor(E_4) = A$
$\wedge\ actor(E_5) = A \wedge actor(E_6) = A$
detection : $((F_1 ; (F_2 \,\&\, F_3)) \,\&\, F_4) ; F_5$
where $action(F_1) = detect(E_1)$
$\wedge\ action(F_2) = detect(E_2)$
$\wedge\ action(F_3) = detect(E_3)$
$\wedge\ action(F_4) = detect(E_4)$
$\wedge\ action(F_5) = detect(E_6) \wedge date(F_5) = t$
verification : W_1
where $action(W_1) = foreign\_mount() \wedge date(W_1) = t'$
$\wedge\ t \leq t'$

Figure 2.11: Snippet from an LAMBDA definition of an Attack Scenario [48]

Similarly to the previous attack description, the attack described in the LAMBDA snippet (Figure 2.11), states that for the NFS Abuse attack to occur, a precondition of gaining remote access by the attacker is expected to be satisfied. Thus, the attacker must first gain remote access before he/she can successfully carry out an NFS Abuse. The attack itself is then described using a set of events $E_1 - E_6$. In summary, these events describe that the attack must first initiate a Remote Procedure Call (RPC) to the target, add a new user to the target machine, and finally mount the nfs attack. This attack allows the attacker to read and modify files on the target host.

In both examples it is observed that case based approaches also use rules to infer the relationship between events/alerts. For example, the pre-conditions will be represented by alerts which must correlate with the attack description. These are events that must have happened in the past before the attack is considered ”successful". Consequences (post conditions) are events that could occur if the current intrusion being evaluated was “successful". Both event types are pre-defined with expert knowledge. Each alert is mapped to a set of pre-requisites and consequences hence with this information, new incoming alerts instance can be correlated with past alerts if at least one of the pre-requisites of a new alert matches one of the consequences of a past alert.

In a correlation model by Ning et al. [116], expert knowledge is used to associate each alert with a set of pre-requisite and consequence events. Alerts are then correlated if consequences of an earlier alert implies the pre-requisite of a later alert. The model by Debar and Wespi [50] also consisted of a component which used consequences for finding causal alerts. A more subtle method was illustrated in the multi-correlation framework proposed by Qin [133] which used highly abstract pre-requisites and consequences. Sundaramurthy et al. [162] proposed a correlation technique similar to pre/post conditions. In their approach, domain knowledge is used to map alerts and other security events of interest (referred to as observations) to entities called internal conditions. This is done in an off-line or training phase and “internal conditions” indicate what the alert could mean at a high level. Furthermore, each internal conditions was associated a qualitative measure of confidence i.e. how likely it is that the observation is true. In analysing a stream of intrusion alerts, alerts are mapped to their respective internal conditions. A set of alerts are correlated if they share or connect to the same “internal conditions”. This is also similar to the model by [116]. Tedesco and Aickelin [164] proposed an extension of the model proposed by [116]. Lin et al. [94] also proposed an extension of the model by [116]. Other case based methods that involve constructing attack scenarios from known knowledge include Yu et al. [179], Roschke et al. [141], Fredj [61], Sheyner [150], Wang [170].

Challenges in Case based Correlation

Case based approaches were initially highly successful due to the expressiveness of the attack definitions [161] however today, the volume and variations of attacks would require a vast number of attack definitions which would be tedious to develop and infeasible to maintain. Furthermore, developing these definitions for the wide range of IDS platforms available would be trivial. Since the rapid rate at which new attack scenarios are introduced and older ones deprecated is very high, constant manual effort is required to keep the knowledge database up to date. For example, if an attempt was made to map the alert system of a single IDS type e.g. Snort to pre-requisites and consequences, a quick survey reveals that the Snort community version contains over 2,000 alert types and 35 classifications by default. Thus, expertise would be required to define at the very least 2,000 pre-requisite and consequence definitions. Many large scale networks however would use multiple IDS types, custom signatures and more premium editions which include many more signatures. Hence, the number of alert types increases. With defining “pre-defined scenarios", the same problem applies. In the model by [161], an average of 15 definitions were developed for three IDS platforms which by now, would most certainly be outdated.

2.6.4 Sequential based Correlation

Sequential correlation methods discover correlated alerts by using temporal based statistical methods to discover whether an earlier alert, a_i triggered a later alert a_j. This inference is modelled by learning from Bayesian inference, Markov models, and various other statistical approaches.

Bayesian Inference

Bayesian Inference is used to calculate the posterior probability of a Hypothesis in light of some evidence using Bayes Rule. Given a *hypothesis* H, and some *evidence* E ,the posterior probability i.e. probability of H, given the Evidence, $P(H|E)$ is derived using Bayes Rule as:

$$P(H|E) = \frac{P(E|H) \times P(H)}{P(E)} \tag{2.3}$$

This inference assumes that the likelihood function $P(E|H)$ is derivable from a set of observations. A key benefit of applying Bayesian Inference is that it provides a probabilistic approach to combining new evidence with prior beliefs. Bayesian Inference is suitable in alert correlation and can be applied in numerous ways. Two common approaches have been observed in prior literature.

One approach is to model the probability that one intrusion alert type will occur given a previous type already occurred. Consider two alert types e.g. *Port Sweep*, *Buffer Overflow*. Expert knowledge Ning et al. [115] indicates that alerts of type *Port Sweep* may follow *Buffer Overflow*, therefore, two alerts a_i, a_j occurring in succession where a_i has type *Port Sweep* and a_j has type *Buffer Overflow* may likely be correlated without even considering whether they have similar features or not. This knowledge can be learnt from a set of historical alerts and can be updated using Bayesian inference. In this case the hypothesis is $T(a_j)$ and the evidence is $T(a_i)$. Therefore, given M intrusion types and a set of observations N, Bayesian inference can be used to model at most $M \times M$ dependencies between all M intrusion alert types. This can be represented as a Bayesian network which consists of a graphical model and conditional probabilities.

This approach is used by Marchetti et al. [102]. In their work, a Pseudo-Bayesian correlation model which consisted of two components was proposed. The first component performed historical analysis on a set of historical alerts in order to derive the conditional probabilities of all the intrusion types. For a historical dataset with M intrusion types, the correlation model builds a Bayesian network with $M * M$ nodes showing the correlation between each intrusion type (represented as nodes). Consider two intrusion types A and B. According to [102], the correlation likelihood of an alert of type B occurring given an earlier alert of type A is calculated as:

$$P(Corr(A,B) = \frac{1}{|corr(A,B)|} \times \sum (a,b) \in Corr(A,B) e^{\frac{(t_a - t_b)^2}{K}} \tag{2.4}$$

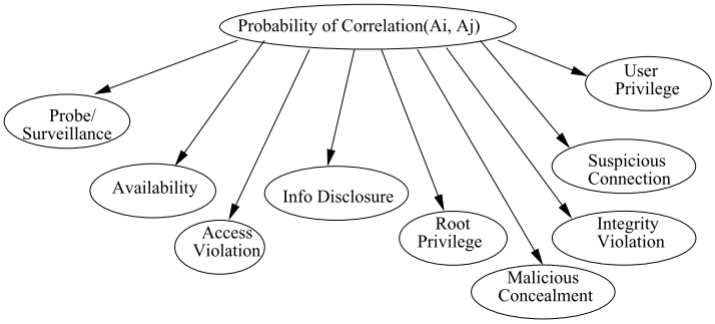

Figure 2.12: Qin's Bayesian-based correlation model [133]

Note that only one alert attribute is considered - *time* where t_a and t_b are the timestamps of alerts a and b, $Corr(A,B)$ is the set of all pairs of alerts of type A and B that occurred within a given time threshold of each other, K is a parameter in a Gaussian decay function such that the time correlation value decays exponentially as the difference between the timestamps increases. Ren et al. [135] also proposed a Bayesian correlation model that discovered causal relations and dependencies between intrusion alert types. In contrast to the model proposed by Marchetti et al. [102], more than a single attribute was used. Kavousi and Akbari's correlation model extended the correlation model by Ren et al. [135] in order to optimise the discovery of causal relations process by adding extra techniques to refine the learnt Bayesian network.

Rather than modelling posterior probability of intrusion types co-occurring, another approach is to model the posterior probability that two alerts would be correlated given some evidence is taken into account. This approach is used by Qin [133]. Alongside using pre-requisites and consequences to correlate alerts, Qin [133] uses conditional probability which measure the likelihood of various attack objectives given two events are correlated. In the inference model, the root node represents the hypothesis that two alerts are correlated and each child node represents a type of evidence. This is illustrated in Figure 2.12.

The benefit of the first application of Bayesian Inference in alert correlation is that it allows capturing causally and relational correlated alert pairs. For example, if $P(a_j|a_i) > P(a_j)$, then a_i has a positive influence on a_j occurring. Likewise if $P(a_j|a_i) < P(a_j)$, then

a_i has a negative influence on a_j occurring [29]. On the other hand, a disadvantage in the application of Bayesian inference is that if combined evidence is dependent, the overall results would be biased.

Other Statistics approaches

Qin [133] also proposed a statistical based correlation engine based on Granger Causality Test (GCT), for discovering alerts which share a causal relationship. The intuition is that if an alert a_i is the cause of another alert a_j, then a_i should precede a_j. To apply GCT, they split a stream of alerts into sliding windows and represent each sliding window in a time series where a time series is an ordered set of time intervals t and the i^{th} entry in a time series represents the number of alerts that occurred during a given time. Their approach used two variables X and Y as attributes of the time series representations. Furthermore, GCT uses statistical functions to test if lagged information on a time-series of variable X provides any statistically significant information about another time-series variable Y. The output of the test would indicate if two alerts from either the same time series or multiple time series are correlated. In their work, they claim that the GCT approach is more effective than typical correlation coefficients for example Pearsons coeffient. More recently, the GCT approach is used for alert correlation in work by Maggi and Zanero [100].

2.6.5 Visualisation based correlation

Visual Analytics is *the science of analytical reasoning facilitated by interactive visual interfaces* [165]. Bertin, a theorist in information visualisation defined the theoretical categories to grouping visualisations namely:- *Diagrams*, *Networks* and *Maps* [30]. More specifically related to security analysis, [154] identified five core applications for security visualisations namely:- *host/server monitoring, internal/external monitoring, port monitoring, router behaviour and attack patterns*. The visual correlation methods in alert correlation are described with respect to both models.

Network Visualisation

Networks are popular in alert correlation. Visualisations that fall under this category are

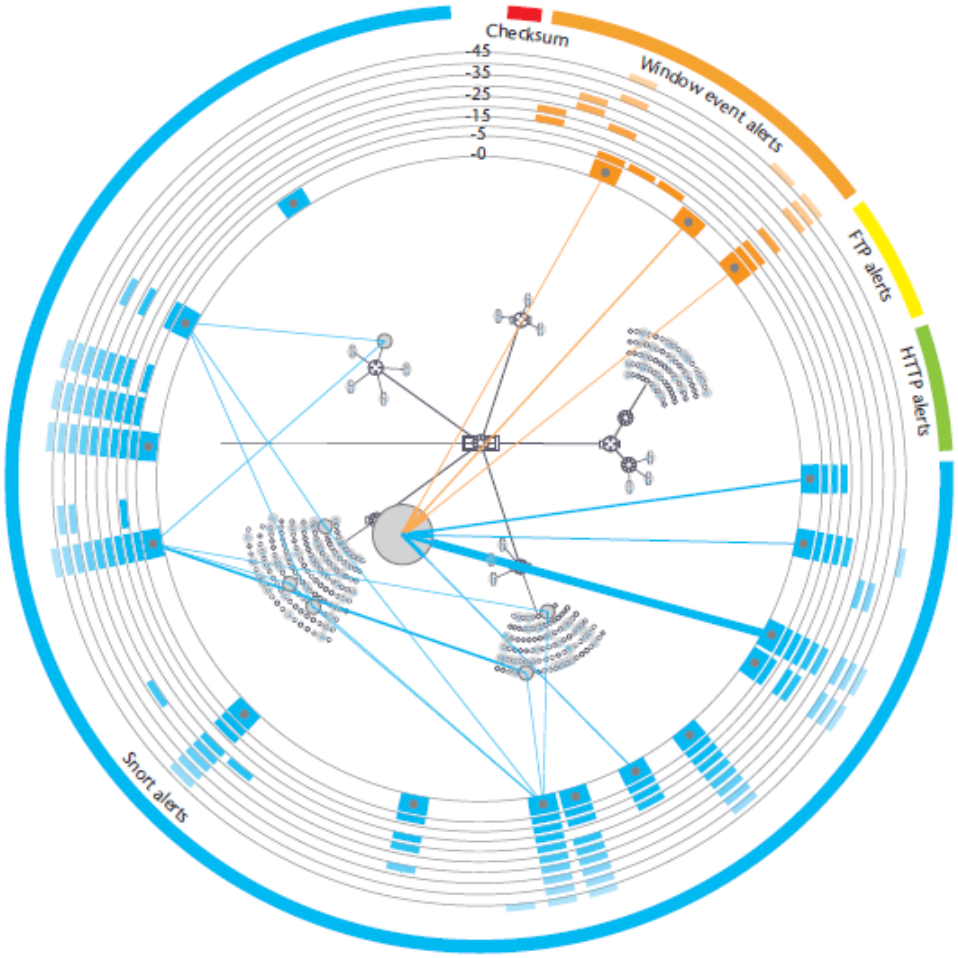

Figure 2.13: VisAlert - A Radial Network Visualisation proposed by Livnat et al. [98]

those where line visual elements can be used to connect a set of other visual elements together to show a relationship[30].

Livnat et al. [98] proposed VisAlert, a radial network visualisation which allows a security analyst to correlate alerts (visually) based on their relation with network hosts. As shown in Figure 2.13, the visualisation consists of multiple layered circles [Figure 2.13]. The innermost circle contains a set of internal hosts which are being monitored and represented as grey circles in Figure 2.13. The outermost arcs on the other hand, each represent an intrusion alert type. An edge between a grey circle representing an intrusion type and an outer arc representing a host indicates the host has been affected by the intrusion. The visualisation also uses a set of spiral layers to represent temporal information. In Figure 2.13, these are the spirals between the innermost circle and outer arcs.

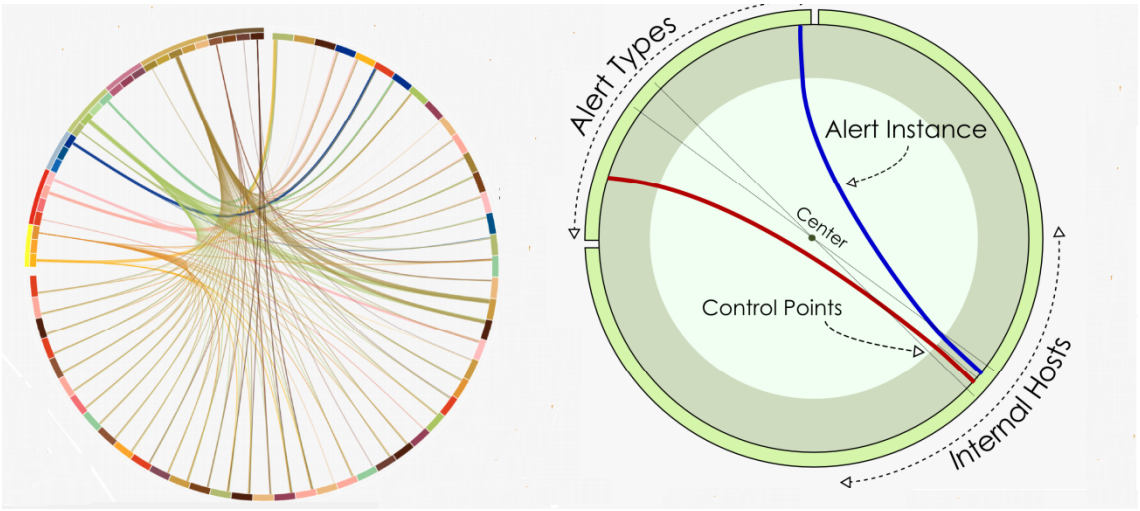

Figure 2.14: Avisa - A radial network visualisation proposed by Shiravi et al. [152]

Shiravi et al. [152] also proposed a radial network visualisation *Avisa* for alert correlation [152]. Unlike VisAlert which was used to represent alerts from various systems, Avisa is specifically for the IDS alert types. In this correlation system, the circumference of the radial is split into 2 arcs. The first arc is used to represent different intrusion types and the second part is used to represent internal hosts in the network. The lines (i.e. edges or links) between each type also show that the connected host has been targeted by an attacker using the connected intrusion type. Similarly to VisAlert grouping techniques are used to categorise alerts by intrusion types and both are suitable for representing a high volume of alerts. Both Shiravi et al and Livnat et al illustrated successfully how the visualisation representations could be used to detect a progressive attacks over a period of time by comparing the visualisation snapshots at different times.

Other alert correlation visualisation techniques that fall under the *network* category include Spiral View [31] and Yang et al's network visualisation in [177]. An effective approach to visualising large volumes of network traffic is taken in ClockMap [83]. ClockMap manages a large space of IP addresses and visualises upto 300 million entries of network traffic collected as NetFlows. The main visual representation is a network view which provides an overview of the traffic flowing in and out of internal hosts. Each host is represented as a node that is split into 24 segments. Each of the segments represents an hour of the day and the colour of each segment represents the intensity of the

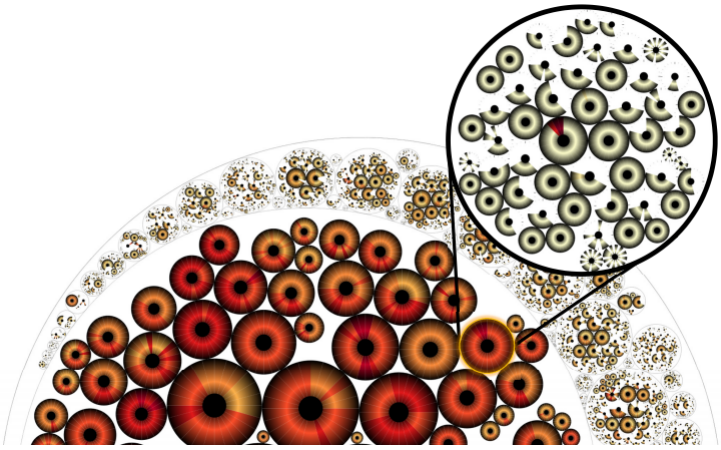

Figure 2.15: ClockMap - A network visualisation [83]

traffic at that hour of the day. Figure 2.15 shows ClockMap.

Diagrams

Bertin defined diagrams as visualisations where *correspondences can be established between all the divisions of one component and all the divisions of another* [30]. In simpler terms, these are visualisations that use two or more axes to represent data relations.

Figure 2.16 shows *Snort View*, a diagram visualisation proposed by Koike and Ohno [84] which shows alert correlations by representing the relations between the source IP addresses of the alerts and the time in which the alerts occurred. It represents this information in a two dimensional graph where the y-axis lists the set of source IP addresses and the x-axis represents an ordered set of time stamps. In alert logs source IPs are usually one of the most variant attributes therefore there are usually a large amount of unique source IPs. This quickly presents a challenge in Snort View because it becomes cluttered as the number of Source IP addresses increases.

A diagram visualisation proposed by Musa and Parish [112] was developed specifically for analysing Snort alerts . The diagram visualisation was a 3D scatterplot used to show the relationship between internal hosts and the magnitude of alerts which affected them. In the scatterplot, the x-axis represents time, the y-axis internal hosts and the z-axis represents frequencies of a set of alerts at a specific time period affecting a specific internal host. Alerts were grouped together based on similar attributes such as those that share

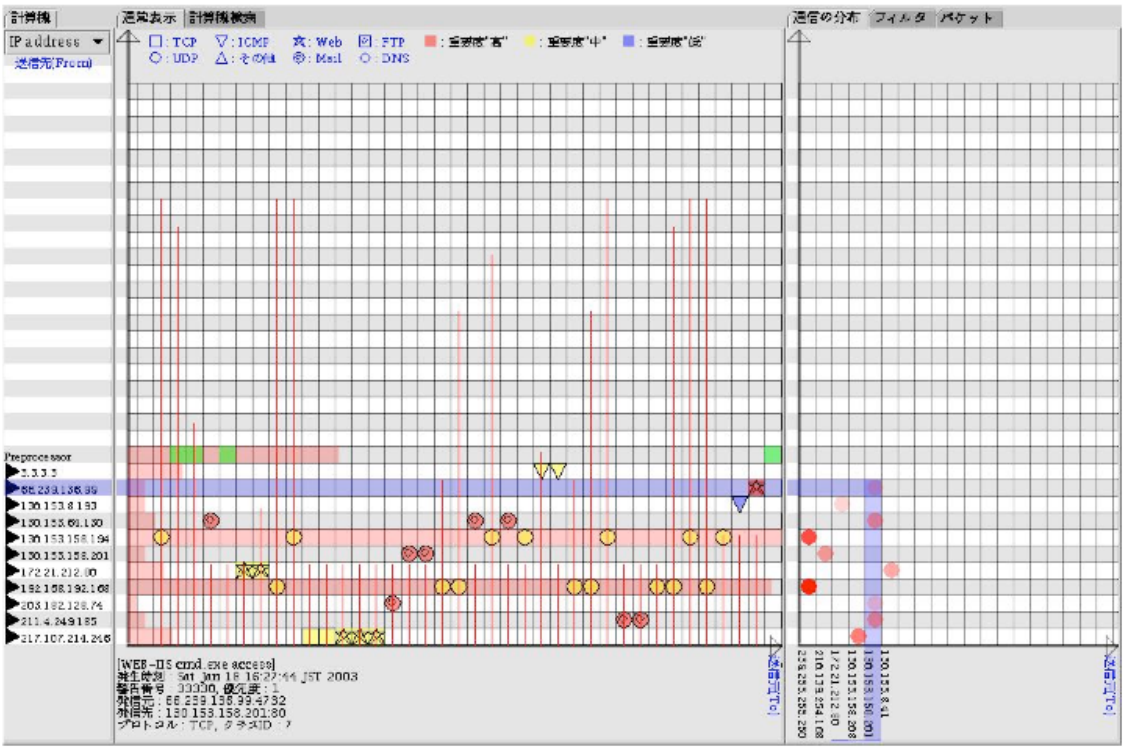

Figure 2.16: Snort view for Alert Correlation developed Koike and Ohno [84]

the same source port and destination IP. Figure 2.17 illustrates Musa and Parish's 3D Scatterplot.

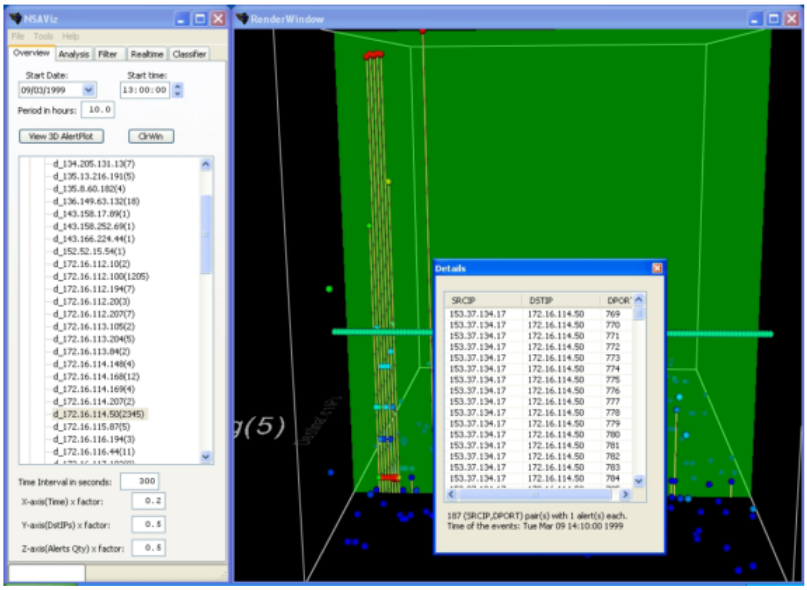

Figure 2.17: A 3D Scatterplot visual representation of alerts by Musa and Parish [112]

Other examples of diagram visualisation which have been used in alert visualisation and correlation include IDSRainstorm [2], IDGraph [136], and NIVA [126].

Maps

Maps are visualisations which represent geographical locations on a map type visualisation. SATURN Rowlingson et al. [143] uses cartography maps for visualisation alert correlations by showing the relationship between the origin and destination of alerts. Figure 2.18 shows an example.

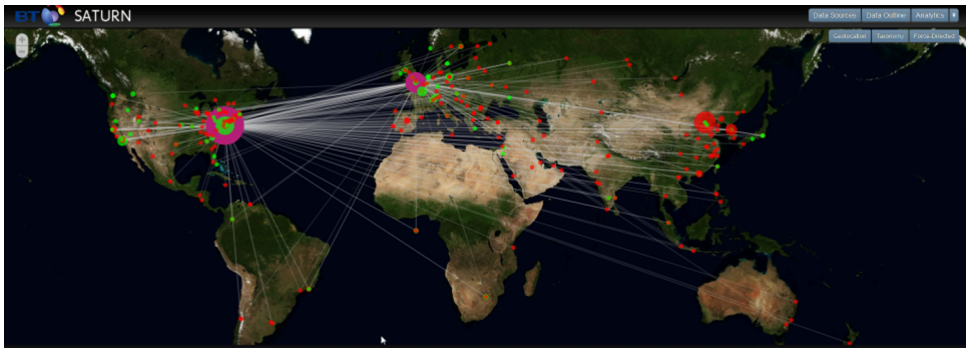

Figure 2.18: Geo-location Map visualisation developed by SATURN [143]

Challenges in using Visual Methods for Alert Correlation

Security devices raise a large amount of alerts and according to [22], a vast amount of these alerts may be irrelevant thus tagged as false positives. When visualising this data, a range of challenges need to be addressed. For example, the *Volume* and the number of *Dimensions* in the alert data may exceed the capacity of a visual representation. Typically alert data may have up to 10 dimensions. Visualisations which represent up to 3 dimensions such as Musa and Parish's scatterplot may be difficult to interpret when used to visualise a large amount of data. On the other hand, more effective visualisations such as Avisa and VisAlert only accommodate two dimensions. Either approach indicates that a lot of data is unaccounted for in these visual representations. This may lead to perceiving misinterpreted correlations. The quality of the alert data may also present a challenge. Due to the large volume of false positives, useful information such as patterns and correlations may be difficult to immediately perceive. As a result it may be more effective to pre-analyse the data before using visual methods.

Table 2.4: Summary of Prior and State of the art Alert Correlation Techniques

Classification	Technique	Alert to Vulnerability	Alert to Network Knowledge	Alert to Attack Knowledge	Alert to Alert
Similarity based	Ruleset	Roschke et al. [140], Morin et al. [109], Cuppens and Miège [46], Yu et al. [179], Morin et al. [110]	Liu et al. [97]	Roschke et al. [141, 142]	Yu et al. [179], Li et al. [90], Wang et al. [171], Morin et al. [110], Ebrahimi et al. [55]
	Weighted metrics & Distances			Noel et al. [125]	Ahmadinejad and Jalili [8], Dain and Cunningham [49]
	Clustering				
	Frequent Pattern mining		Lagzian [87]		Sadoddin and Ghorbani [145], Jinghu et al. [75], Khan et al. [82]
	Others				Taha and Ghaffar [163], Shittu et al. [156], Liu et al. [96]
Case based					Ning et al. [118], Lin et al. [94], Sundaramurthy et al. [162], Ning et al. [116], Ou et al. [129], Zali et al. [180]
Sequential based	Bayesian/Probablistic			Cheng et al. [43]	Benferhat et al. [29], Kavousi and Akbari [79], Marchetti et al. [102], Du and Yang [54], Benferhat et al. [28]
	Artificial Intelligence				Bateni et al. [26]
	Statistical				Ou et al. [130], Bateni and Baraani [25], Li et al. [90]
Visualisation based	Diagrams				Musa and Parish [112], Xiao et al. [175], Abdullah et al. [2], Ren et al. [136]
	Maps				
	Networks				Yang et al. [177], Shiravi et al. [152], Ying et al. [178], Bertini et al. [31], Du and Yang [54]

2.7 A Taxonomy for Intrusion Alert Prioritisation Techniques

2.7.1 Introduction

In this section, the techniques used to achieve alert prioritisation are reviewed. A measurable priority is assigned to alerts which indicates how severe the attack it represents may be. This also determines how immediate the alert should be addressed by a security analyst or response system. While this is often regarded as trivial, in 2013, it was reported in Verizon [168] that 84% of attacked organisations contained evidence of the

attack in their network logs - most of which were not immediately addressed by security analysts or other response systems. Various reasons exist as to why attacks are commonly unaddressed, however, a key reason often identified is that the volume of network logs is so high that it is difficult to identify events related to real threats thus, most evidence of attacks are ignored. Intrusion alerts are types of evidence of a *potential* network attack and according to Axelsson [22] the false positive rate of an IDS (which can be up to 99%), makes it difficult to identify true positive alerts hidden in the mass. Re-prioritising alerts based on validating the alerts with respect to real threats could vastly improve the attack detection rate.

Alert prioritisation is applied in similar domains such as in static code analysis where a vast amount of false positive alerts are also generated [70].

In the intrusion detection domain, only a few research studies such as [185, 17, 124] and [114] have been done in this area as most prioritisation techniques are based on heuristics and methods used by vendors of IDSs and alert correlation products. Furthermore, many aim to tackle the issue of false positives by improving the detection methods of the IDS either by improving the detection approaches used by an anomaly detection IDS or fine-tuning the signatures used by a signature based IDS. Prioritisation of alerts is only one of the many ways to address false positive alerts.

Recently, [16] provided a comprehensive research study on prioritising security alerts. In their work, three categories of prioritisation metrics - *static, vulnerability and post-incident based prioritisation metrics* are presented. The state-of-the-art research in this section is reviewed and discussed according to this taxonomy.

2.7.2 Static based Prioritisation

This prioritisation technique focusses on assigning a priority value to alerts triggered without knowing if the intrusion represented by the alert was successful. The prioritisation is solely based on what is known of the intrusion type and its potential effects.

Based on observation, static based prioritisation metrics are commonly applied to commercial signature based intrusion detection systems such as Snort[139] and is achieved by assigning a prioritisation value to each signature in an IDS's configuration.

The limitations of static priority metrics is that they require extensive expertise such that in order to assign prioritisations to given intrusion signatures, expert knowledge about the effect of the intrusion signatures is required. Furthermore, this approach requires frequent maintainability. As old signatures are deprecated and new ones are recreated, a manual effort is required to assign new prioritisation to newly created signatures. In addition, static alert prioritisation results in homogeneous alert prioritisation. This means that all alerts sharing the same signature will always share the same priority. e.g. in signature based detection, all alerts of type x will always have priority P_x. This however should not be the case, as other factors such as the value of the asset and the validation of the alert should be considered.

Other than expert knowledge which is required, it is unknown if research effort can be applied to improve static prioritisation as there was no prior research identified.

2.7.3 Vulnerability based Prioritisation

As discussed in Section 2.5.2, intrusion alerts are often correlated with vulnerability prioritisation techniques. This focuses on assigning priority values to alerts based on whether their associated vulnerability confirms that the intrusion represented by the alert was successful or not.

In [140], an alert is first correlated to a host's system properties and vulnerabilities. Following this, the alert is assigned the priority of the CVE associated with the alert's respective vulnerability. In the model by Roschke et al. [140], alerts with less than a priority value of 5.0 are filtered. The priority is a quantitative value based on the *Common Vulnerability Scoring System* (CVSS) which is a standard framework for prioritisation vulnerabilities [105]. The CVSS scoring ranges between 0 - 10 and are defined as follows.

- Vulnerabilities are labelled "Low" if they have a CVSS score of 0.0-3.9.
- Vulnerabilities will be labelled "Medium" if they have a CVSS score of 4.0-6.9.
- Vulnerabilities will be labelled "High" if they have a CVSS score of 7.0-10.0.

2.7.4 Post-Incident based Prioritisation

Post-incident based prioritisation focusses on *"investigating and evaluating incidents based on the level of potential risk after incidents occur"* [16] . This technique seeks information about assigning priorities from various sources (including the vulnerability based prioritisation) before finally combining the information into a single priority value. Anuar [16] classifies information for this type of prioritisation into: 1) System related which measures the impact of the intrusion on the targeted network host, system or asset and 2) Attack related which measures the likelihood that the intrusion is a threat.

System Related

System related metrics measure the impact of an alert if it was successful. A key factor which determines the severity of an alert is the value of the target which it affects. In previous work [15, 132], a targeted host's value can be measured by evaluating what type of asset the host is and the type of network access the host would give to an intruder if successfully compromised.

Attack Related

The system related priority metrics indicate how likely an alert is a true positive intrusion. Porras et al. [132] first proposed an alert ranking framework, M-Correlator, with a prioritisation component that consisted of two security metrics: *relevance* and *priority scoring*. Relevance scoring measures the validity of an alert while priority scoring measured the severity of an alert given the targeted asset's value. The *relevance scoring* was computed using validation information derived during alert reduction. Thus, the higher the likelihood that a targeted host is vulnerable to a given intrusion, the higher the relevance scoring.

The *priority scoring* combined an interest score which measured the degree to which an analyst expressed interest in the attack category the alert belonged. Using a Bayesian model they determine the overall priority of an alert based on the acquired evidence. A limitation in their approach is that knowledge from alert correlation is not taken into account during the prioritisation despite their framework consisting of a similarity based correlation component.

A more robust alert prioritisation system is proposed by Alsubhi et al. [15, 14] who define 7 attack-related prioritising metrics. One of the metrics, called *alert relationship metric* measures the degree to which the alert correlates with successive alerts. Using this prioritisation metric a high value could indicate the alert is potentially a causal alert. The reliability of the security device which raised the alert is also taken into account.

Zomlot et al. [185] also proposed a prioritisation model for the alert correlation system they had previously presented [162]. In their work on prioritisation, they use Dempster-Shafer to assign a degree of belief to alerts received from the correlation system which indicated the likelihood of true positivity given the quality of the IDS sensor which raised the alerts.

Noel and Jajodia [124] proposed an alert prioritising framework which used a different metric. The metric calculated the proximity of an alert to a critical asset. Thus, alerts targeted at assets closer to critical assets had a higher priority over those further away.

Additionally, Porras et al. [132] identified that a metric of *Interest* which does not fall under either of these classifications. In their work they describe *Interest* as a subjective attribute which describes the degree to which an analyst may be interested in a type of alert. Intuitively, interest levels positively correlate with severity and relevance. An analyst is likely to be more interested in alerts with high severities and high relevance scores. In the prioritisation metrics proposed by Valeur et al. [167] and Porras et al. [132], interest levels were defined by assigning 'measures of interest' to different classes of alert types such as Denial of Services (DoS), read and write attacks, privacy violations etc. Each class was associated with a manually defined interest value.

2.8 Summary

In this chapter the pioneering as well as state of the art taxonomies for classifying intrusions and intrusion detection systems were reviewed. Two levels of intrusions were identified. Signature and Anomaly detection methods are primarily targeted at discovering low-level intrusions. However, both detection methods have a high false positive rate and neither are suitable for detecting high-level intrusions. Detecting a high level intrusion

from low-level intrusions can increase attack knowledge and can be used to filter false positive low-level intrusions. Salah et al. [146] proposed a framework for detecting high level intrusions. The most significant components in this framework are the correlation and prioritisation components.

Salah et al. [146] and Valeur et al. [167] each provided classification models for grouping techniques used in the correlation component. Salah et al. [146] identified the challenges involved in using similarity metrics and case based methods for discovering high-level intrusions. Similarity metrics are often unsuitable because these metrics only detect correlations when low-level intrusions share similar attributes. Vincent Zhou et al. [169] however identified that modern attacks (high-level intrusion) are collaborative, stealth and span over a long period of time, thus, a set of low-level intrusions part of the same attack may not share similar attributes. Case based methods which are domain knowledge dependent and appear to be more robust require a large amount of expertise and resources which require constant updates in order to remain effective. Salah et al. [146] identified sequential in particular probabilistic approaches as the most popular in correlation research progression.

As described in Section 2.6, probabilistic analysis in alert correlation requires learning the likelihood of two or more alerts correlating given one or more conditions are met. Deriving the complete conditional probabilities for a stream or database of alerts requires considering a large number of variables such as the likelihood of various alert types correlating, the likelihood of correlation they occur over a period of time, as well as the likelihood of correlation given based on their attribute values. As more variables are considered, the complexity of the correlation process increases significantly. Thus there is often a trade-off with performance and accuracy in many correlation models.

In this thesis we investigate how to build a probabilistic correlation technique which is optimal in speed and has a high accuracy rate by using similarity metrics to filter out non-trivial variables during the building of a probabilistic correlation model. This is described in details in Chapter 3.

The second significant component that was identified was the prioritisation component. During the critical review of prior art in prioritisation techniques, it was observed that despite correlation analysis providing a wealth of attack knowledge, very few state-of-the-art models consider knowledge gained during correlation analysis for prioritisation. In this research study, we investigate how to improve prioritisation through using knowledge gained during correlation. This is detailed in Chapter 4.

Chapter 3

Statistical Alert Correlation for Discovering Attack Structures

3.1 Overview

The increased size, structure and value of a modern day network has resulted in an exponential increase in network activity consequently leading to an equivalent increase in network traffic, network attacks and subsequently, an increase in the number of alerts triggered by security devices on the network. Hence, many of the correlation techniques described in Section 2.6, particularly those proposed in earlier years, require improvements in speed and accuracy. Many remote attacks of today are also stealthier and much more sophisticated. When traditional correlation approaches are used, the correlation between key phases of an attack may not be detected.

Despite these limitations, researchers such as Zhou et al. [182] identified that security alert correlation analysis is still a viable solution to gaining attack insight and detecting attacks. They also identified that the issue of how to effectively increase correlation performance and accuracy remains unsolved and is an open ended research problem.

In this Chapter, our object is to generate high-level global and synthetic alerts through applying alert correlation. The focus is to improve the rate of attack detection through improving the accuracy of correlation analysis. Three key factors which affect the accuracy of a correlation model have been identified:

1. *The duration of the attack:* More and more remote attacks known as advanced persistent threats are carefully crafted and well planned [72]. The attackers invest time in infiltrating the victim's network without being detected. In this case an attack could be carried out over several months or years. For example, in 2012 a large and complex threat called Stuxnet was discovered which infected an estimated 100,000 machines across 150 countries. It was targeted at disrupting the operations of critical infrastructures which ran industrial control systems. Prior to detection, the attack was said to have existed for almost a year [57].

   For a correlation model to successfully link each of the phases of such an attack, the correlation model would need to continuously analyse a large dataset captured over months or years. As described in Section 2.3 of Chapter 2.1, the amount of data flowing through an average sized network in a single day could be terabytes in size. Performing such an analysis with a year's data is almost infeasible if not impossible.

2. *The origin of the attack:* Many remote attacks are distributed. In some cases, the origin of the attack is spoofed. Both issues imply that a single attack may be carried out using multiple source identities. Take for instance the attack campaign called GhostNet which successfully compromised over 1,000 computers belonging to a number of non-governmental organisations. These compromised computers were used to infiltrate systems from over 100 countries which included systems from organisations located in the UK. In such cases, a correlation model may not be able to find the link between the different phases unless expert knowledge is incorporated into the correlation model. The link between the attacks initialized by the multiple sources may not be detectable from analysing the data.

3. *The phases of the attack:* Although the final impact of an attack may be remote, each of the phases of the remote attack may not be remote. Some of the phases may be physical attacks and these are not included in network logs. In some cases, the attacks carried out are zero day and therefore may go undetected by a security

device such as an IDS (these are known as false negatives). An example of such an attack is the Hydraq attack(also as Aurora) which was detected to have compromised a major organisation. In this attack, a zero day exploit discovered in Internet Explorer was used to deliver a Trojan called "Hydraq.Trojan" to the targeted networks [59]. In such scenarios, this means if a signature IDS was used as the security measure on the network, it is highly unlikely to detect the attack. While an anomaly IDS may detect the attack, it is also highly unlikely. We are therefore presented with missing alerts i.e. false negatives.

In Section 2.6 of Chapter 2, three categories of correlation methods were identified. Salah et al. [146] identified *Similarity correlation methods* as the simplest yet the least effective approach given they rely heavily on the similarity between alert features in order to discover correlations. However, with respect to the limitations identified above, when the *origin* of an attack is distributed or ambiguous, alerts that represent the same attack may not share the same source or destination addresses therefore similarity correlation methods will miss such correlations. *Case based methods* being more effective with finding correlations, require an extensive amount of expertise knowledge to function. In order to cover the wide range of today's network attacks and their variations, the number of required rules to define each attack case would require an infeasible amount of effort. Therefore, it was concluded during this research study that sequential correlation methods have the most potential for addressing the described limiting factors without compromising performance. In prior art,[146, 28], this technique is described as fast, scalable, optimizable, and are suitable for correlating alerts despite missing attack phases. Both further identified the popularity of sequential correlation models based on Bayesian Inference due to their transparency, ease of use and ability to handle uncertain correlations.

Our objective is to generate accurate high-level global alerts, however, given the limiting factors addressed, the research aim in this chapter is to answer the following question *"Can an optimised Bayesian inference correlation model analyse a large volume of security alert data whilst still providing high quality correlations despite the missing*

data and noise contained in the alert data?". By achieving high quality correlations we are subsequently improving how well IDSs detect attacks and through achieving this, it becomes possible to eliminate false positive alerts. This is in accordance with the primary objectives of this research study as described in Section 1.2.

Whilst Bayesian inference appears to be an ideal approach, its limitations cannot be ignored. In Section 2.6, we described a survey on correlation models which were based on Bayesian inference. From this survey, we identified the following limitations:

1. *The time complexity of computing Bayesian networks in alert correlation* is dependent on at least three factors - the number of alerts used to train the model, the types of alerts it is to be trained to correlate, and the attributes of the alerts considered during correlation. Let each factor be represented as a variable n, m, k respectively. For example, the Bayesian approach proposed by Ren et al. [135](described in Section 2.6) required that for a historical dataset of n security alerts to be used for training where there are k alert attributes their correlation model would use approximately $\sum_{i=1}^{k} n!/k!(n-k!)!$ iterations to discover relevant conditional probabilities before finally computing a Bayesian network of m^2 nodes. Despite, Bayesian correlation models being scalable, the defined factors for alert correlation can easily increase the time complexity thus a longer duration of time would be required to train the model using a dataset with larger values of n, m, and k.

2. *Bayesian inference assumes attributes are independent.* When considering the attributes of alert data, dependencies may exist between features, for example, attribute *http intrusions* will often be associated with attribute *port number 80*. If both attributes are considered during the learning phase, a Bayesian model would (by default) assign equal weights to each attribute, which, if heavy dependencies exist in the dataset would eventually lead to inaccurate correlations. A potential solution is to utilise a single attribute during the model training. The Pseudo-Bayesian correlation model proposed by Marchetti et al's which is inspired by Bayes Theorem utilised only one temporal property - time. In comparison to Ren et al. [135],

Marchetti et al. [102] model reflects a better performance (i.e. lower time complexity). The complexity of their causal relation discovery process can be described as quadratic as it does not explore other properties. The challenge however is that *"Is a single attribute sufficient?"*. Our observation is that too few features result in false correlations.

3. Incomplete or unrepresentative training data will result in an inaccurately trained correlation model. Noise is typically a problem in alert data. Training data with a large noise component would increasingly affect the accuracy of the training process.

In this Chapter we aimed to combine Bayesian inference, a sequential correlation approach with a similarity correlation approach to achieve high quality correlations. This allows to benefit from both the approaches. Two hybrid models are proposed. The rest of this chapter is outlined as follows: Section 3.2 introduces the correlation models which are proposed in this study. Section 3.3 describes the case studies and datasets used to evaluate the proposed models. Section 3.4 presents the experiments' results and evaluation of the correlation models. It evaluates the proposed correlation models against the state-of-the-art presented. Finally, in Section 3.5 the conclusions are discussed including the motivation and framework for Chapter 4.

3.2 Proposed Approach: A Temporal Statistical Correlation Model

Two probabilistic correlation models are proposed. Both systems receive a stream of intrusion detection alerts as input, discover intrusion alerts related to the same attack(s) based on prior probabilities and new evidence and output these discoveries in a comprehensive data structure. Figure 3.1 illustrates this overall concept. For clarification purposes, a set of key definitions are provided.

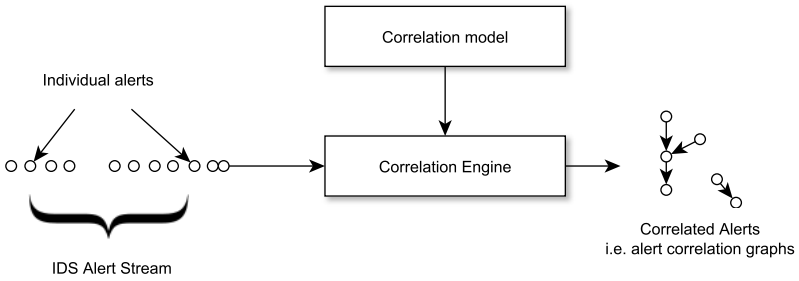

Figure 3.1: Overview of Correlation Models

3.2.1 Definitions

Alert

An alert is represented as a 6-tuple $(\alpha_1, \alpha_2, ..., \alpha_6)$ where the elements of the tuple are the attribute values of the alert's timestamp, source IP, source port, destination IP, destination port, and intrusion type respectively. These attributes are either text, IP address or of numeric data types. An alert of type T_a is an alert instance which has T_a as the value for α_6.

Meta-alert

A meta-alert is a higher-level alert which contains one or more low-level alerts (e.g. Snort alerts) grouped together during correlation. Figure 3.2(a) shows a set of intrusion alert tuples labelled a_1 to a_6. After some form of correlation, a logical relation such as Figure 3.2(b) is established and is referred to as an "Alert Correlation Graph".

In prior research [167, 117, 162], a meta-alert has been referred to as a hyper-alert, hyper-alert graph, alert correlation graph and attack graph. To ensure clarity, only the terms meta-alert and alert correlation graphs are used in this Chapter. In general, the term meta-alert is used when describing correlated alerts. As a more technical definition, particularly when referencing the "graph" data structure of the correlation alerts depicted in Figure 3.2(b), the term "Alert Correlation Graph" is used.

Alert Correlation Graph

An alert correlation graph is a weighted directed acyclic connected graph G = (V, E) where V represents a set of nodes and each node v $\in$ V represents a 6-tuple low-level alert. Each edge, $e_{v_i,v_j} \in$ E is a connection between two nodes v_i, v_j which indicate that 1)

```
1  a1: (03/07-16:28, 1.1.1.40,26582, 5.5.5.3, 25, Policy attempted download of a pdf)
2  a2: (03/07-16:28, 5.5.5.3, 3727, 10.0.0.3, 25, Policy attempted download of a pdf)
3  a3: (03/07-17:30, 1.1.1.43,48097, 5.5.5.3, 25, Policy attempted download of a pdf)
4  a4: (03/07-17:30, 5.5.5.3, 3730, 10.0.0.3, 25, Policy attempted download of a pdf)
5  a5: (03/07-17:35, 1.1.1.48,53514, 5.5.5.3, 25, Policy attempted download of a pdf)
6  a6: (03/07-17:35, 5.5.5.3, 3727, 10.0.0.3, 25, Policy attempted download of a pdf)
```

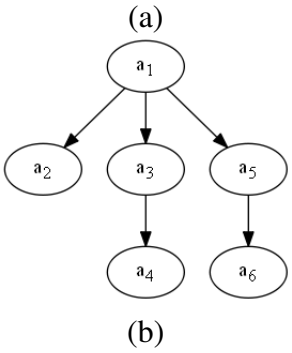

Figure 3.2: (a)A set of alert tuples -(timestamp, source IP, source port, destination IP, destination Port, and intrusion type) before correlation and (b) A meta-alert/alert correlation graph

v_i and v_j are correlated and 2) v_i represents an alert that occurred before v_j. The weight of the edge depicts the correlation strength between both alert nodes.

3.2.2 Approach One: Correlation using Bayesian Inference and A-prior Rule Learning

Figure 3.3 illustrates the architecture of the first proposed correlation model.

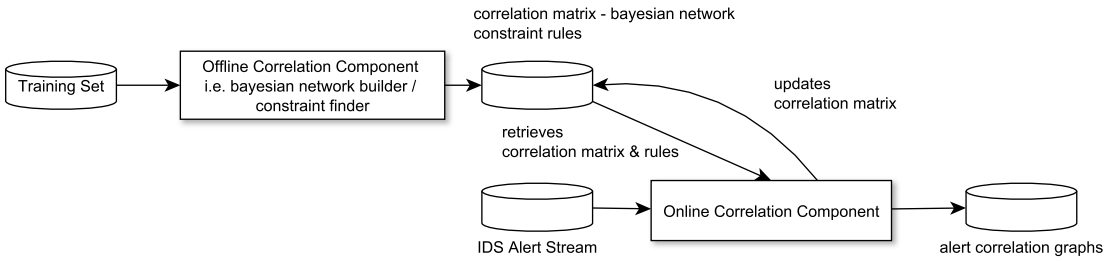

Figure 3.3: Architecture of the Posterior Correlation Model.

A correlation matrix is built by the off-line correlation component and is periodically used and updated by the on-line correlation component. The correlation model consists of two knowledge tables: (i) Correlation Likelihood Table and (ii) Correlation Constraint Table. A correlation likelihood represents the strength between two alert types T_a and T_b. More specifically, it refers to the *likelihood* of an alert of type T_b occurring after an

alert of type T_a. For any two alert types, T_a and T_b, the correlation likelihood $L(T_a, T_b)$ is conditional to a constraint being true. The relationship between correlation likelihood and constraint is represented in Equation 3.1.

$$L(T_a, T_b) = P(T_a \rightarrow T_b | C) \tag{3.1}$$

A constraint, C, is a rule which captures the conditions under which two alerts are correlated. For example a constraint such as $\{\text{time}_{a,b} \leq 20 \text{ secs}\}$ indicates that both alert types T_a and T_b are correlated when they occur within 20 seconds of each other. Another illustration of a constraint is $\{\text{destIP}_{a,b} = 1\}$, this indicates that both alert type T_a and T_b are correlated when they share the same destination IP (In other words, the difference between their destination IP addresses is zero). Table 3.1 illustrates further examples of constraints encountered.

Constraints	Descriptions
$\{\text{DestPort}_{a,b} = 1\}$	The destination port of alert of T_a and an alert of T_b must be identical
$\{\text{DestIP}_{a,b} \geq = 0.5\}$	The destination IP of alert of T_a and an alert of T_b must be common up to at least the 2nd Octet.
$\{\text{DestIP}_{a,b} \geq 0.5$, $\text{SourceIP}_{a,b} \geq 0.2$, $\text{DestPort}_{a,b} = 1\}$	The destination IP of alert of T_a and an alert of T_b must be common up to at least the 2nd Octet, their source IPs must be common up to at least the 1st Octet and their destination port must be identical.

Table 3.1: Examples of Constraints between Alert Types

When two alert types have $n : n > 1$ constraints between them, the correlation likelihood between the two alert types is the minimum likelihood of the n constraints:

$$L(T_a, T_b) = \min\{P(T_a \rightarrow T_b | C_i)\}_{i=1}^{n} \tag{3.2}$$

where the probability of $T_a \rightarrow T_b$ occurring given C_i is defined as:

$$P(T_a \rightarrow T_b | C_i) = \frac{P(T_a \rightarrow T_b) * P(C_i | T_a \rightarrow T_b)}{P(C_i)} \tag{3.3}$$

In Equation 3.3, given a set of historical alerts, $P(T_a \rightarrow T_b)$ refers to the number of times T_b occurs after T_a in the same time window W with respect to the number of times T_a occurs in that same window. $P(C_i)$ refers to the number of times an alert of Type T_b occurs after an alert of T_a where both types satisfy constraint C_i with respect to the number or times T_a occurs in the total historical alert dataset H. Finally, $P(C_i|T_a \rightarrow T_b)$ is the probability of C_i given both Type T_a and T_b occur within the same time window.

Algorithm 1 shows how the off-line correlation component computes all correlation likelihoods and constraints.

Algorithm 1 Offline Correlation Process for Approach One

```
1:  function OFFLINE PROCESS
2:      A = All alert attributes
3:      H = Historic Alerts
4:      T = All alert types in H
5:      T' = All pairs of types in T
6:      for all T_a, T_b ∈ T' do
7:          C(T_a, T_b) =
8:              GETCONSTRAINTS(A, T_a, T_b)
9:          L(T_a, T_b) =
10:             min{
11:              P(T_a → T_b | C(T_a, T_b)_i)
12:             }_{i=1}^{n}
13:     end for
14: end function
```

```
1:  function GETCONSTRAINTS(C, T_a, T_b)
2:      k = 1
3:      C(T_a, T_b) ← ∅
4:      Get first order feature set
5:      for all c_i ∈ C do
6:          if P(T_a → T_b | c_i) > θ then
7:              C(T_a, T_b) ← C(T_a, T_b)_k ∪ c_i
8:          end if
9:      end for
10:     Gets k relevant feature sets
11:     k ← 2
12:     C ← ∅
13:     while (C(T_a, T_b)_{k-1} ≠ ∅) do
14:         C ← All k combinations from
15:             C(T_a, T_b)_{k-1}
16:         for all c_i ∈ C do
17:             if P(T_a → T_b | c_i) > θ then
18:                 C(T_a, T_b)_k ← C(T_a, T_b)_k ∪ c_i
19:             end if
20:         end for
21:         k + 1
22:     end while
23:     return C(T_a, T_b)
24: end function
```

Lines 1 to 5 describe the dataset required to initialise the off-line process. A is the set of alert fields used, H is a set of historical alerts used to train the model which are retrieved from log files or an alert database, T is a finite set of all values possible for field *type* and T' is a set containing all 2-permutations of the set T where T'_i represents the i^{th}

T_a, T_b pair.

For each pair, the function GETCONSTRAINTS generates a set of constraints by computing all possible k-combinations of the attributes in A using a step-wise a-priori approach (this was inspired by Ren et al. [135]). Firstly, we start with the $k-$combination where $k = 1$. This means we generate constraints of length 1, where each constraint, C only contains one attribute $a \in A$. For each constraint, we measure the probability that T_a will occur before T_b given they have the constraint C in common. If the probability does not exceed a given threshold θ, it is pruned and considered as non-relevant to the pair T_a, T_b. At the end of each incremental stage, non-pruned constraints are used to generate $K+1$ combinations : $k <= |A|$. This is how the constraint 3 in Table 3.1 is generated.

Each incoming alert a_j, received in real-time is analysed against a set of alerts $S = \{a_1, a_2, ..., a_n\}$ that had occurred within the last T_θ seconds before alert a_j. To determine if a_j and an alert in S are correlated, their types are extracted and used to find the relevant correlation likelihood and constraints (stored in the knowledge repository by the offline component). Two alerts are correlated if:

1. The correlation likelihood of Type a_i and Type a_j is greater than or equal to a threshold, C_θ

2. At least one of the respective constraints of Type a_i and Type a_j holds true for a_i and a_j.

Each analysed historic alert is stored as a node in the database. If the incoming alert a_j is correlated with an existing alert a_i, a_j is added to the meta-alert which a_i belongs to. Consequently, an edge is added to the meta-alert to depict the correlation.

3.2.3 Approach Two: Correlation using Bayesian Inference and Weighted Scoring

The architecture of the second proposed correlation model is very similar to Figure 3.3. Similarly, in order to correlate alerts in real-time, this approach requires that the relationship between various alert types are learnt in an off-line training model which is periodically updated. The approach also consists of two key components. In the off-line

correlation component, a training set is required. The model learns the correlation likelihoods between all alert types. This is done to model the sequence of events that lead to an attack. For example, the correlation likelihoods model "The likelihood of PortScan intrusion alerts co-occurring" or "The likelihood of buffer-overflow intrusions occurring in sequence with sql-injection intrusions". The training set, D enables the computation of the correlation likelihood matrix. Given the training set, D where 1) each entry $d \in D$ represents an alert tuple, and 2) n is the number of unique alert types observed, the correlation matrix is computed as an n-by-n matrix containing the correlation likelihoods between every pair of n alert types. Each entry $C(T_p, T_q)$ in the correlation matrix contains the correlation likelihood between two alert types. The correlation likelihood between two alert types, T_p, T_q is defined as:

$$C(T_p, T_q) = P(T_p|T_q) = \frac{P(T_q|T_p) * P(T_p)}{P(T_q)} \tag{3.4}$$

To calculate $C(T_p, T_q)$, for every alert type pair T_p, T_q all alerts of type p and type q are extracted. An alert is of type T_p if it has the value T_p for its "type" attribute. In addition D_p is the set of all alerts in the training set of type T_p. We then define the following:

$$P(T_q) = \frac{|D_q|}{|D|} \tag{3.5}$$

$$P(T_p) = \frac{D_p}{|D|} \tag{3.6}$$

$$P(T_q|T_p) = \frac{\text{\# of times successfully correlated} T_p \text{with} T_q}{\text{\# of times attempted to correlate} T_p \text{ with } T_q} \tag{3.7}$$

In equation 3.7, *# of times attempted to correlate T_p with T_q* is the # of times successful and unsuccessful correlations attempted. Two alerts are correlated if their weighted similarity scoring is above a threshold θ. Given two alerts, a_i, a_j, their weighted scoring is computed by combining the following similarities between the alerts:

1. Time Proximity (f_1). This feature represents the time proximity between two alerts.

It is derived as a sigmoid function such that the time proximity between two alerts decreases as the time between them increases. In Equation 3.8, t represents the time difference between two alerts in minutes.

$$f_1 = \frac{1}{1+e^t}; \tag{3.8}$$

2. Common Prefix Similarity(f_2, f_3). This compares the source and the destination IP of a_i to the source and destination IP of a_j respectively. It is a measure that indicates that a_i and a_j are targets to a similar destination node or/and are from a similar host. The higher the value, the more likely this statement holds true.

   The common prefix length measure is applied for all IP address features. It is the common prefix bits between any two IPs as shown in Table 3.2.

Table 3.2: IP Common Prefix Length

172.16.113.20	10101100 . 00010000 . 01110001 . 11001111
172.16.115.20	10101100 . 00010000 . 01110011 . 00010100
Common Mask	11111111 . 11111111 . 11111100 . 00000000
	22/32 = 0.68

   An alternative comparison is also provided. This is the *Crossed Common Prefix*. This similarity metric measures the similarity between the source and the destination IP of a_i to the destination and source IP of a_j respectively. This feature indicates that a_j is a responsive intrusion to a_i. For example, if DestIP$_{a_i}$ == SourceIP$_{a_j}$ it could indicate that a_i was a successful attempt to exploit DestIP$_{a_i}$. After this success,a_j could indicate that this host is now performing intrusive activities. On the other hand, a_j could be an alert indicating echo activity which was a response to a_i. In this case not only is the above condition satisfied but also SourceIP$_{a_i}$ == DestIP$_{a_j}$. Since f_2 and f_3 conflict each other, i.e. the relationship between two intrusions is likely to be one or the other but not both, we select only one of the features based on the feature with the highest similarity.

3. Port Similarity (f_4). This feature indicates 1 if both alerts share the same destination

port and 0 if they don't.

Using the described similarities, the similarity weighted scoring, $WS(a_i, a_j)$ is derived using Equation 3.9 where F is the set of the similarity measures described above and ω_k is the weight assigned to the k^{th} similarity. The correlation matrix is stored in a correlation model repository similar to the one shown in Figure 3.3.

$$WS(a_i, a_j) = \frac{\sum_{k=0}^{|F|} \omega_k \times f_k(a_i, a_j)}{\sum_{k=0}^{|F|} \omega_k} \tag{3.9}$$

Given the knowledge acquired in the off-line process, a stream of alerts received in real-time can now be analysed. This is performed by an on-line model which attempts to correlate each alert in the stream to an alert occurring before it. The overview of the online correlation analysis is illustrated in Figure 3.4.

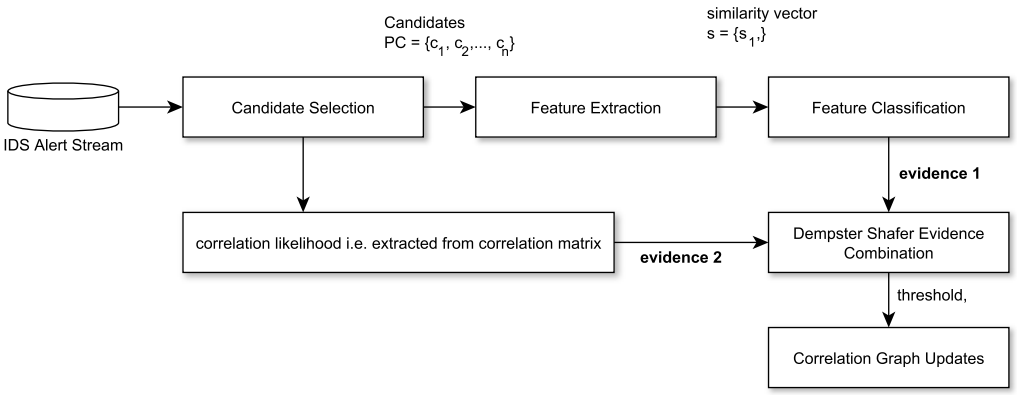

Figure 3.4: Processes involved in the Online Correlation Component of the Dempster Shafer Correlation Model.

For each alert in the stream, a_i, the model finds a set of potential correlation candidate alerts $PC = \{a_1, a_2, a_m\}$ such that $\forall a_j \in PC, a_j \neq a_i$. PC contains zero or more previously analysed alerts which occurred within an acceptable time window of a_i. We use T_θ to denote the acceptable time window. This means that the timestamp of a_i and a_j must be no more than T_θ apart. T_θ is measured in seconds unit.

Given $PC = \emptyset$, this means no potential candidate alerts are found. In this case, a new

meta-alert is created and a_i is added to a new meta-alert. When one or more potential candidates are found, i.e. $PC \neq \emptyset$, the process proceeds with feature extraction. For each potential candidate, $a_j : a_j \neq a_i$, the similarity weighted scoring between each potential candidate and a_i is computed. This is referred to as **evidence 1**.

For each potential candidate, $a_j : a_j \neq a_i$, we also extract their respective correlation likelihood from the correlation matrix. This is referred to as **evidence 2**. Note that this is the prior probability of the alert type of a_j given the alert type of a_i. This probability indicates the probability that a_i and a_j will be correlated without even considering their weighted scoring. Finally, correlation strength between two alerts is the average of both evidences.

A configurable correlation threshold, θ is used to determine strong correlations. a_i is correlated to every alert in PC with an overall correlation likelihood that exceeds θ. Since each candidate alert had been previously analysed, it already belongs to meta-alert. Let M be the set of all meta-alerts that the members of PC belong to. If $|M| > 0$ all the meta-alerts are merged to form a single new meta-alert g_m. Otherwise, the single member of M is denoted as g_m. To add a_j to g_m, a_j is added as a vertex in g_m and for every alert in PC that also belongs to g_m, the value of $C(a_j, g_m)$ is assigned to a new edge created between a_j and $a_m \in g_m$ for a_m.

3.3 Datasets

For experimental purposes, a series of network remote attacks are carried out on two simulated networks. Both experiments and datasets are provided by external institutions. Our objective is to use these datasets to evaluate how well the proposed correlation models generate global high-level alerts given that the models were designed to overcome certain limitation factors in alert correlation. With this, we aim to be able to identify false positive alerts better and furthermore, gain attack insight. The details of the attacks, simulated networks and collected data are described next in Section 3.3.1 and 3.3.2.

3.3.1 Case Study I: DARPA 2000

The DARPA Case Study was a cyber security experiment carried out in March/April 2000 and includes two attack scenarios LLSDOS1 and LLSDOS2. A full evaluation of the experiment is described by Lippmann et al. [95]. It is publicly available and has been used for model evaluation in many research studies including - [171, 145, 121, 101, 135]. Therefore it allows us to compare our model with prior-art. In both scenarios, a DDoS attack is launched against a simulated US government network (referred to as AFB). The DARPA network architecture [95] of the US government network is illustrated in Figure 3.5.

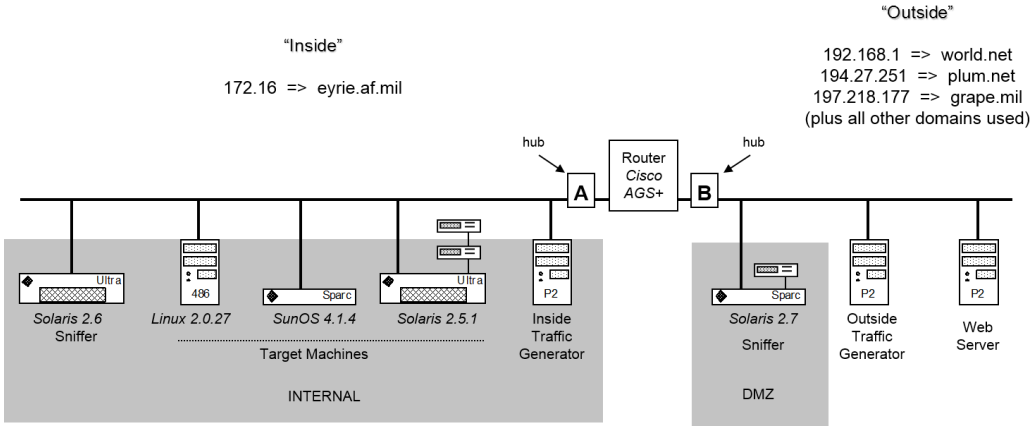

Figure 3.5: Network Architecture of Simulated US Government from DARPA Case Study [95]

The DDoS is targeted on a host situated on the simulated government network. In both attack scenarios, a remote attacker breaks into the network, compromises three network hosts 1) A Linux 2.0.27 machine, a SunOS 4.1.4. machine and a Solaris 2.5.1 machine by exploiting their service vulnerabilities and finally launches a DDoS from the three hosts. Both attacks are carried out over 5 steps and described in Table 3.3.

LLSDOS1

Table 3.3 illustrates this is a naive DDoS attack. Network traffic logs in tcpdump format are collected from the *Inside Sniffer* depicted in Figure 3.5. According to the DARPA documentation [95], the data was captured over a span of approximately 3 hours on Tuesday, 7 March 2000, from 9:25 AM to 12:35 PM, Eastern Standard Time. The attack occurred

Table 3.3: Phases of the LLSDOS1 and LLSDOS2 Darpa 2000 Activity

Steps	LLSDOS1 Intrusion Activity	LLSDOS2 Intrusion Activity
1	IPsweep of the AFB from a remote site	Probe of AFB via the HINFO query
2	Probe of live IP's to look for the sadmind daemon running on Solaris hosts	Breakin-to AFB via the sadmind exploit
3	Breakins via the sadmind vulnerability, both successful and unsuccessful on those hosts	FTP upload of DDoS software and attack script, to break-into more AFB hosts.
4	Installation of the trojan mstream DDoS software on three hosts at the AFB	Initiate attack on other AFB hosts. (repeat of step 1 and 2)
5	Launching the DDoS	Launching the DDoS

within the documented time window. Since no intrusion detection systems were deployed on the network, we configured a *Snort* intrusion detection system in offline mode to parse the sniffer logs. The default snort configurations are used and all snort default rules were switched on. From the internal sniffer network traffic logs, 27,145 snort intrusion alerts were generated. From the DMZ sniffer logs, 2282 intrusion alerts were generated. Since the offline Snort analysis was performed in 2015 and the actual data was collected in 2000, it is assumed that all phases of the attack would be recognised by the snort IDS therefore, there should be 0% false negatives and 100% true positives. However, there maybe a large volume of false positives, the percentage of this is unknown.

LLSDOS2

This is a more stealth distributed denial of service attack. Network traffic logs in tcpdump format are also collected from the *Inside Sniffer* depicted in Figure 3.5. According to the DARPA documentation, the data was captured over a span of approximately 1 hour, 45 minutes on April 16 2000, from 14:45 to 16:28. The attack occurred within the documented time window. Similarly to LLSDOS1, a *Snort* intrusion detection system with default configuration is used to parse the sniffer logs. From the internal sniffer network traffic logs, 13226 snort intrusion alerts were generated. From the DMZ sniffer logs, 580

intrusion alerts were generated. It is also assumed that all phases of the attack would be recognised by the snort IDS.

3.3.2 Case Study II: Northrop Grumman

A cyber range experiment was carried out in 2012 by industrial partners of the British Telecom Security Practice Research Labs [173]. The cyber-range experiment models a simulated computer network which comprises of two sub-networks - a main network with approximately 200 workstation clients and a branch network comprising of 10 workstation clients. To model real network activities, the experiment utilised comprehensive scripts for simulating email sending, server activity and content download activities. Figure 3.6 shows the network architecture.

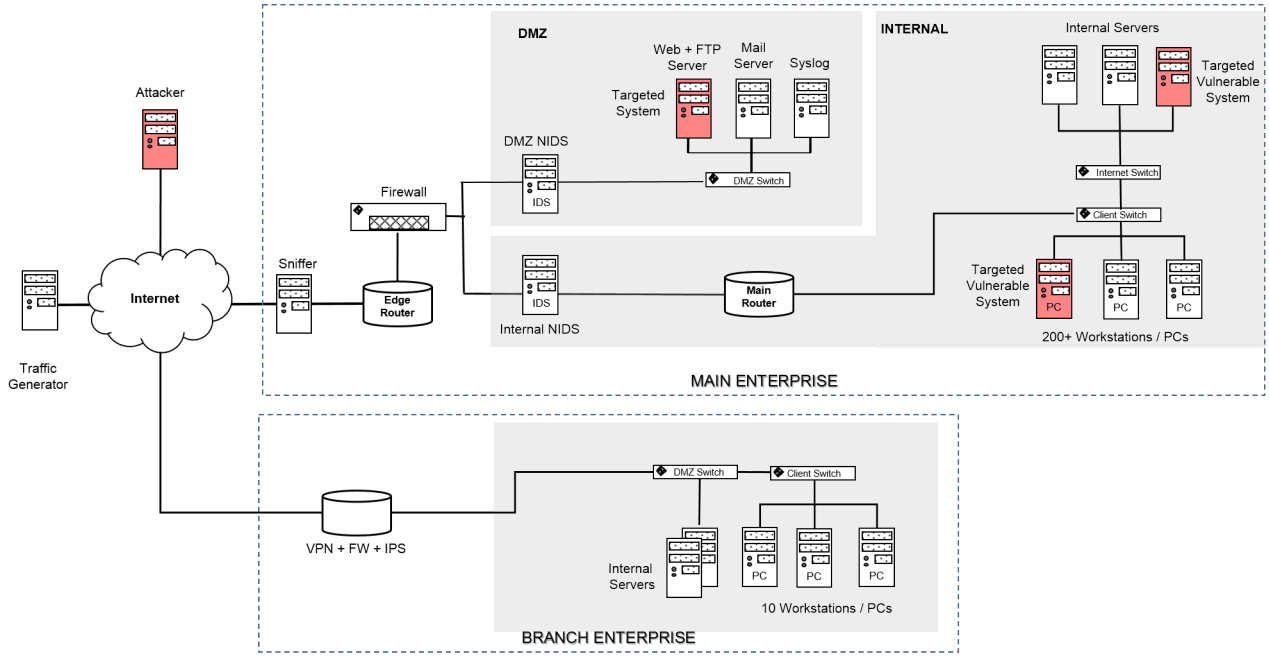

Figure 3.6: Network Architecture of Simulated Network from Northrop Grumman Case Study

In total, the cyber-range experiment includes four different attacks on the simulated network (carried out across four days). Only two of the attack dataset are considered in this study.

NGDMZ1

This is an attack to exploit the DMZ web server shown in Figure 3.6. The attacker aims at compromising the DMZ web server by exploiting its vulnerabilities. The attacker is situated outside the network. The attack comprised of 4 phases - DMZ Scanning (Casual & Intense), vulnerability assessment, exhaustive penetration and brute-force audit from the attacker on the web server. Normal network activity includes server activity such as email routing from a DMZ mail server to an internal mail server and network pinging between the DMZ mail and FTP servers.

Table 3.4: Phases of the NGDMZ1 and NGDMZ2 Attacks during the Northrop Grumman Cyber-range Experiment

Steps	NGDMZ1 Intrusion Activity	NGDMZ2 Intrusion Activity
1	Attack1 starts casual port scanning against Web Server	Attack1 starts casual port scanning against Web Server
2	Attack1 starts intrusive port scanning against Web Server	Attack1 starts intrusive port scanning against Web Server
3	Attack1 performs Nexpose vulnerability assessment on Web Server	Attack1 performs Nexpose Exhaustive Penetration audit on Web Server
4	Attack1 performs Nexpose Exhaustive Penetration audit on Web Server	Attack1 performs Nexpose webscan assessment on Web Server
5	Attack1 performs Nexpose Bruteforce audit on Web Server	Attack1 performs Nexpose Bruteforce audit on Web Server

The data was captured over a span of approximately 24 hours on Tuesday, 2 March 2012, from 00:00AM to 23:59PM, GMT whereby the attack occurred between 11:00AM to 14:00PM. The logs from the DMZ IDS are to be analysed by the proposed models. 3199 intrusion alerts were generated. The exact number of true positives is unknown. From inspecting the data, it is known that the false positives are high.

NGDMZ2

This is a repetition of the NGDMZ1 attack with slight modification. The data was captured over a span of approximately 5 hours on Tuesday, 5 March 2012, from 10:24AM to 15:00PM. The DMZ logs are also collected. 3226 intrusion alerts were generated. The numbers of false positives are high and true positive rate is unknown.

3.4 Evaluation

The key objective of this Chapter identified in Section 3.1 was to investigate if attack detection could be improved by generating accurate global high-level alerts through combining similarity and sequential correlation techniques. The two proposed models are evaluated to determine if there is improved attack detection and correlation accuracy. The following metrics are used:

3.4.1 Metrics

False Positive Correlation Rate : For any given dataset, the false positive correlation rate (FPCR) measures the quality of the correlation model by measuring the number of incorrect correlations that took place. This is defined as:

$$\text{FPR} = \frac{\text{FP}}{\text{N}} = \frac{\text{\# of incorrectly correlated intrusion alert types}}{\text{\# of correlated intrusion alert types}} \tag{3.10}$$

True Positive Correlation Rate : The TPCR evaluates the ability of the system to correctly correlate alerts. It is defined as follows:

$$\text{TPR} = \frac{\text{TP}}{\text{P}} = \frac{\text{\# of correctly correlated intrusion alert types}}{\text{\# of true positive correlated intrusion alert types}} \tag{3.11}$$

Time Complexity : As well as improving the accuracy of correlation models, it was also mentioned in Section 1 that we aimed to achieve this without compromising the performance of the model. For the proposed models, the time taken by the offline and online processes is measured in comparison to existing models.

Observation : Given that an alert correlation graph represents a set of intrusions that constitute part or an attack, one or more alert correlation graphs should also represent a single attack. Similarly, an alert correlation graph should represent no more than a single attack unless two attacks are highly related or similar. Using observation, we investigate how well each alert correlation graph depicts its corresponding attack.

3.4.2 Environment

During the experiments, the correlation models were developed in Java and evaluated on a 64-bit Windows System with an Intel(R)Core i5 CPU processor at 2.40GHz, JVM 1.4.2 and 6GB for maximum heap memory.

3.4.3 Evaluation One - DARPA Case Study

This evaluation is based on the DARPA case study. Both a naive and stealth DDoS attack is described in Section 3.3. To evaluate the proposed models, The correlation matrices are generated via the off-line components using the dataset captured from the naive attack - LLSDOS1, and the stealth attack data - LLSDOS2, is analysed by the on-line components as a stream of real-time events.

Off-line Results

42 intrusion types were discovered in the LLSDOS1 inside and DMZ logs. Therefore both correlation models generated a 42-by-42 correlation matrix. Figure 3.7 shows a list of all the intrusion types.

This evaluation is focussed on model 2. The off-line components of this model was tested using a range of configurations. Table 3.5 shows the TPCR, FPCR and time duration taken to generate the correlation matrices given a range of varying parameters, T_θ and C_θ. The values for both parameters where selected using trial-and-error. In each iteration an increase in the size of the correlation window, T_θ and C_θ is done by at least doubling is current value. A larger value for T_θ enables the detection of attacks carried out over a longer time period and vice versa. Smaller values for C_θ enable the detection of both naive and stealth attacks. In Table 3.5, T_θ is represented in minutes.

Table 3.5 shows that in Test cases 1 - 3 where a small time window threshold ranging between $T_\theta = 1 - 10$ mins, combined with a high correlation thresholds of $C_\theta = 0.8$, yielded a better performance (i.e. a higher TPCR and lower FPCR).

Test cases with larger window thresholds (such as such as Test cases 4 - 6 and 10

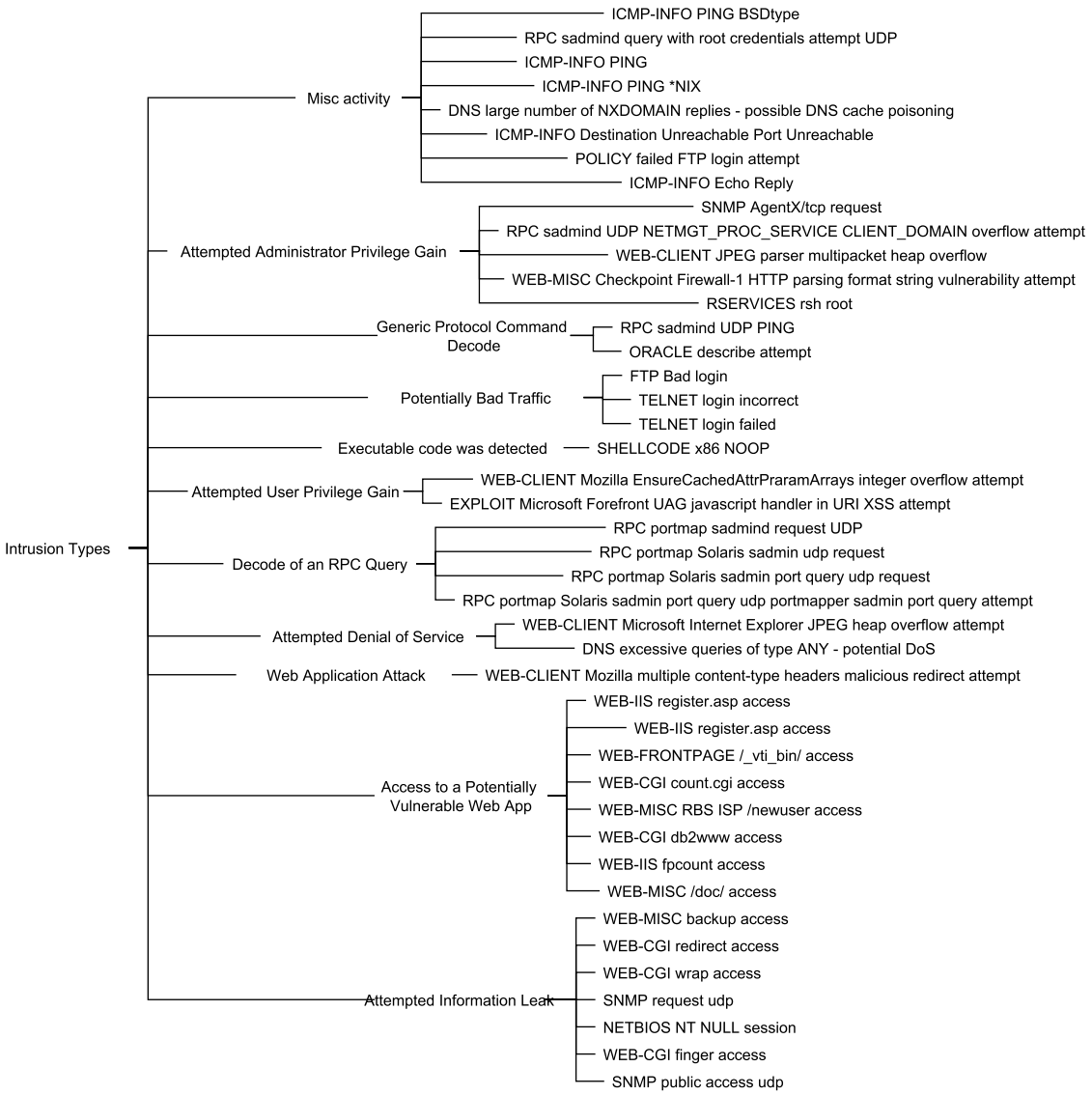

Figure 3.7: Intrusion types and categories triggered by Snort IDS for LLSDOS1

- 12 on Table 3.5 yielded a higher false positive rates (FPCR). This is undesirable as it indicates that particular alert types which should not be correlated were being correlated. It also indicates that alerts part of multiple attacks are being identified as a single attack. A possible explanation to this is that many of the alerts in the DARPA LLSDOS1 dataset share common attributes. To minimise such false positives, the similarity threshold was set to a higher threshold. When C_θ was set to a higher correlation threshold value, the correlation strictness increased and a higher value ensured only close related alerts(i.e. alerts with highly similar features), were correlated.

The time taken to generate the 42×42 matrix given 3199 intrusion alerts and 42

Table 3.5: Evaluation of Offline Analysis using LLSDOS1

			Model 2		
Test case	T_θ	C_θ	**FPCR**	**TPCR**	**Time (seconds)**
1	1	0.8	0.01	0.77	192
2	5	0.8	0.01	0.77	353
3	10	0.8	0.01	0.78	356
4	20	0.8	0.02	0.76	497
5	40	0.8	0.02	0.76	444
6	180	0.8	0.02	0.77	815
7	1	0.4	0.05	0.64	242
8	5	0.4	0.08	0.69	372
9	10	0.4	0.11	0.70	381
10	20	0.4	0.16	0.70	394
11	40	0.4	0.23	0.70	554
12	180	0.4	0.49	0.70	761

intrusion types was $O(42 \times 3199)$ for Model 2. While this indicates a large number of iterations, the analysis took approximately 7 minutes. The top 10 correlations of Model 2 are shown in Table 3.6.

Table 3.6: Top Correlation generated by Model Two Offline Correlation using LLS-DOS1 dataset

Top Correlated Pairs T_a, T_b	**Model 2** $P(T_b\|T_a)$
ICMP-INFO Destination Unreachable Port Unreachable , ICMP-INFO Destination Unreachable Port Unreachable	0.84
ICMP-INFO PING,ICMP-INFO Echo Reply	0.61
ORACLE describe attempt , SHELLCODE x86 NOOP	0.73
DNS excessive queries of type ANY - potential DoS,RSERVICES rsh root	0.70
TELNET login incorrect,WEB-CGI db2www access	0.36
NETBIOS NT NULL session,POLICY failed FTP login attempt	0.31
DNS large number of NXDOMAIN replies - possible DNS cache poisoning,NETBIOS NT NULL session	0.63
RPC portmap Solaris sadmin port query udp request,RPC sadmind UDP PING	0.26
WEB-CGI redirect access,SNMP AgentX/tcp request	0.56

Table 3.6 shows the correlations derived by Model 2 using the configuration $T_\theta = 10mins$ and $C_\theta = 0.8$. The first two pairs show correlations between intrusion types that

capture the reconnaissance activity. These correlations depict that many of the host IP addresses and ports *swept* by the attacker were in-existent. It also depicts that over 50% of the ip addresses pings discovered hosts. Following this, the remaining lines depict correlations between intrusion types that reflect a series of exploits of services on the discovered hosts and multiple failed attempts to logon to the discovered (and potentially exploited) hosts. These correlations are consistent with the attack phases described in Section 3.3. For Model 2, a total of 103 correlations between intrusion types were discovered (including the 10 illustrated in Table 3.6). 80 correlations were true positive correlations while 23 were false positives thus, using the FPCR and TPCR metrics defined in Section 3.4, a 1% FPCR and an 78% TPCR was achieved.

Comparison of proposed approach to prior State of the art

A comparison between the proposed model and an existing model proposed by Ren et al. [135] is performed. While more recent research such as [79] exists, the model proposed by Ren et al. [135] provides a more comparable platform due to the well documented design and their choice of evaluation methods. This existing model was also evaluated using the same LLSDOS dataset. Both Figure 3.8 & 3.9 shows that both the proposed models in this research have lower false positives rates and lower true positive rates respectively. While a lower false positive rate is desirable, the true positive rate indicates a lesser performance. A possible reason is that the truth table used to evaluate the dataset was not appropriately labelled since this was done manually based on the our knowledge. In most cases, majority of the alerts were expected to be correlated. It is likely that this is not the case.

Online Results

In the on-line component, the alerts from scenario LLSDOS2 are correlated using the correlation matrix and constraints generated from the off-line component with the configuration described in Line 3 of Table 3.5. A total of 238 graphs were generated. Figure 3.10 shows an alert correlation graph generated by the on-line component.

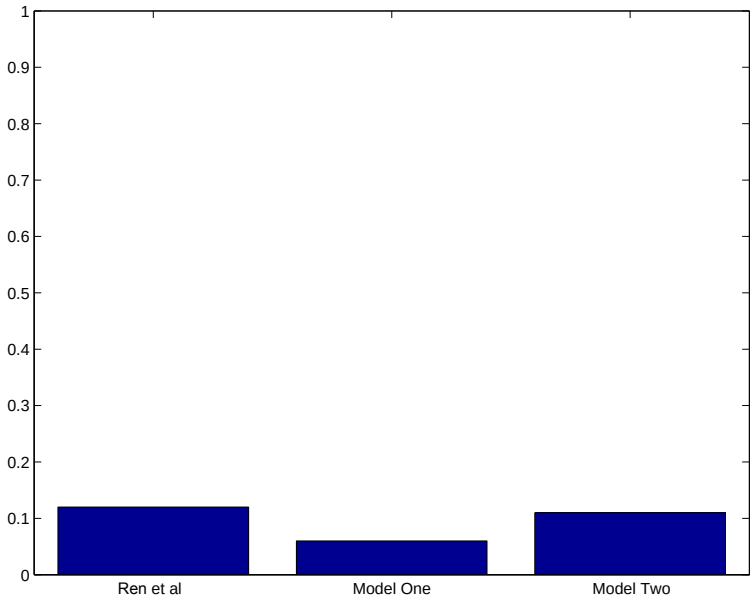

Figure 3.8: FPRC from LLSDOS1 Dataset

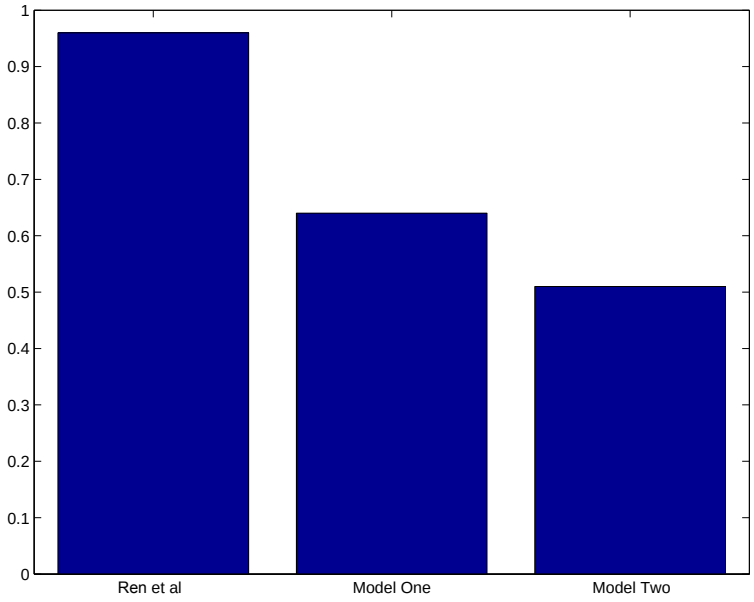

Figure 3.9: TPRC from LLSDOS1 Dataset

3.4.4 Evaluation Two - NG Case Study

Similarly to the DARPA analysis, to evaluate the proposed models, the correlation matrices are generated via the off-line components using the dataset captured from the naive attack - NGDMZ1, and the stealth attack data - NGDMZ2, is analysed by the on-line components as a stream of real-time events.

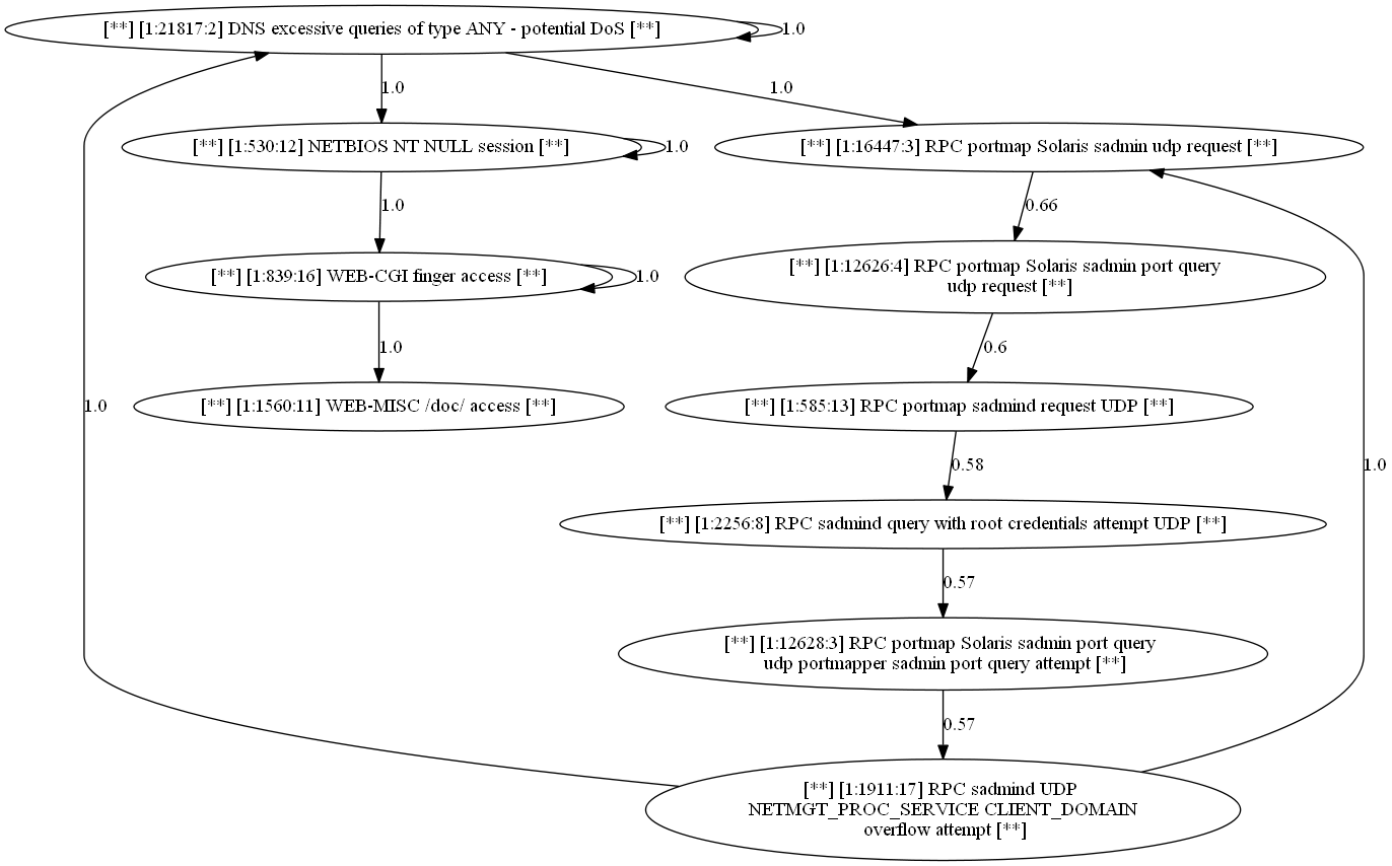

Figure 3.10: Alert Correlation Graphs generated from the DARPA LLSDOS2 Activity

Off-line Results

32 intrusion types were discovered in the NGDMZ1 attack. Therefore, both correlation models generated a 32 by 32 correlation matrix. Figure 3.11 shows a list of all the intrusion types. The off-line components of both models were also tested using a range of configurations for C_θ and T_θ. The results are shown in Table 3.7. T_θ is measured in minutes and the *Time* column represents the duration of the analysis process in seconds.

Table 3.7 shows that high quality correlations were achieved particularly by Model 2. Similarly to the DARPA evaluation, it is observed that for Model 2, test cases 1 - 6, there are better quality correlations i.e. high TPCR and low FPCR when C_θ is set to a high value in comparison to Model 1. It was also observed that the window T_θ did not affect the accuracy of the model nor the time taken to analyse the data.

Given the consistency with the DARPA evaluation, it was concluded that this was as a result of both networks being simulated and of small to medium size, therefore, the similarity between the attributes in the network traffic and security alerts were also

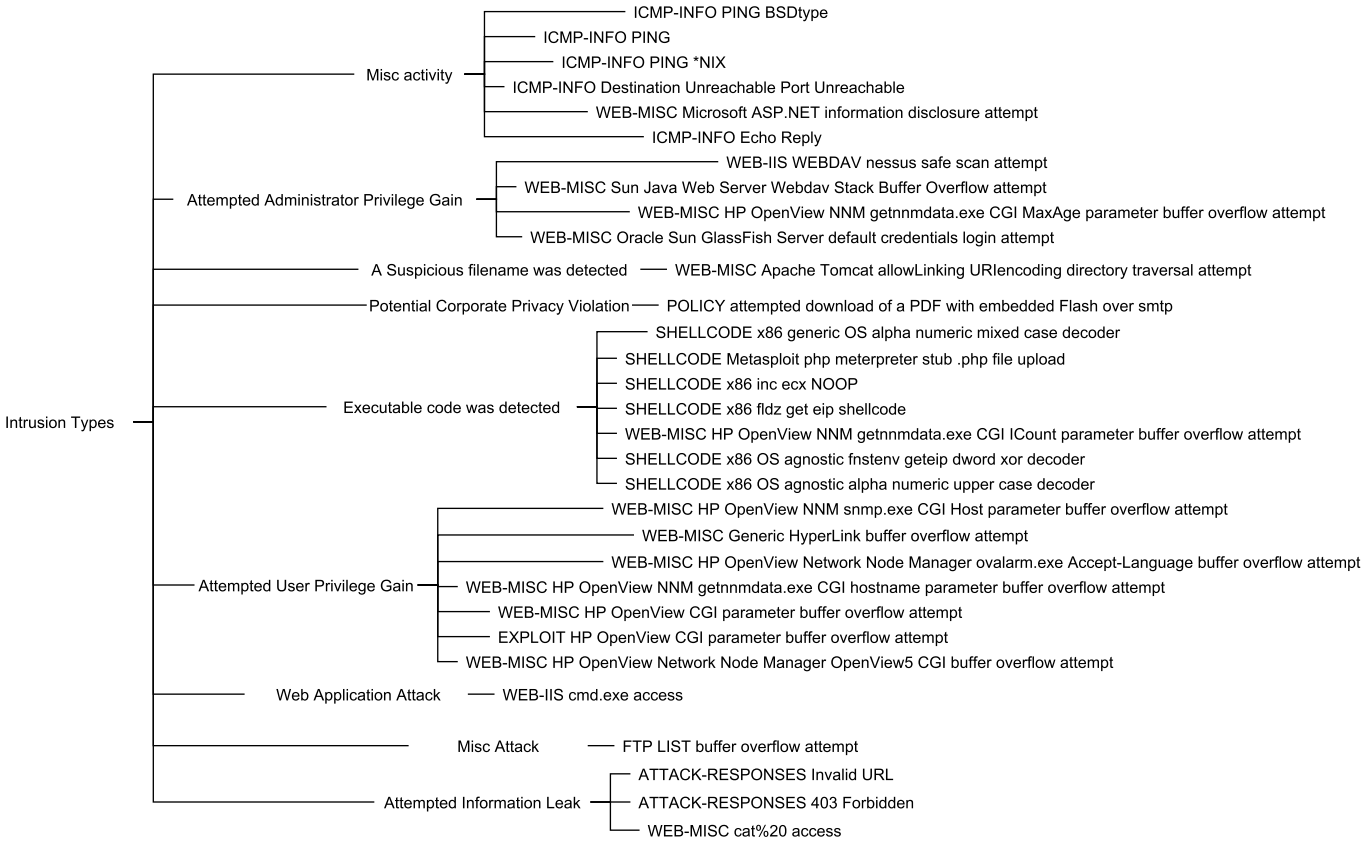

Figure 3.11: Intrusion Types and Categories Triggered by Snort IDS for NGDMZ1

high. Hence, using a high correlation threshold minimised the false positive correlations and correctly identified the true positive correlations. The scope of the attacks were also limited i.e. the same attacker attacked a limited number of hosts.

Table 3.8 shows the top correlations produced by both the proposed models using the configurations - T_θ = 30 mins and C_θ = 0.8. Overall, Model 2 shows a higher accuracy rate with lower false positives and higher true positives, this is shown in Figure 3.12 and Figure 3.13

Online Results

Using the same configuration - T_θ = 30 mins and C_θ = 0.8, 165 alert correlation graphs were generated by Model 2.

Fig.3.14 shows a set of alert correlation graphs generated by Model 2. Due to the frequent communication between servers on the network, many ICMP Ping and Reply alerts

Table 3.7: Evaluation of Offline Analysis using NGDMZ1

			Model One			Model Two		
Test case	T_θ	C_θ	**FPCR**	**TPCR**	**Time**	**FPCR**	**TPCR**	**Time**
1	1	0.8	0.50	0.80	74	0.02	0.88	20
2	5	0.8	0.50	0.80	89	0.02	0.91	52
3	10	0.8	0.50	0.81	121	0.02	0.91	62
4	20	0.8	0.50	0.81	134	0.02	0.91	70
5	40	0.8	0.59	0.81	116	0.02	0.90	77
6	180	0.8	0.61	0.99	142	0.02	0.92	95
7	1	0.4	0.55	0.77	79	0.05	0.85	19
8	5	0.4	0.55	0.81	93	0.07	0.83	50
9	10	0.4	0.58	0.81	98	0.14	0.74	63
10	20	0.4	0.58	0.81	108	0.29	0.61	71
11	40	0.4	0.58	0.81	159	0.45	0.53	79
12	180	0.4	0.74	1.00	181	0.66	0.58	99

Table 3.8: Top Correlation generated by Model One and Model Two Offline Correlation using NGDMZ1 dataset

	Model 1		**Model 2**
Top Correlated Pairs T_a, T_b	Constraints	$P(T_b\|T_a)$	$P(T_b\|T_a)$
ICMP-INFO Destination Unreachable Port Unreachable,ICMP-INFO Destination Unreachable Port Unreachable	SourcePort=1, DestIP=1, SourceIP=1, DestPort=1	1	0.77
SHELLCODE x86 inc ecx NOOP,WEB-MISC HP OpenView NNM getnnmdata.exe CGI hostname parameter buffer overflow attempt	SourcePort=1, DestIP=1, SourceIP=1, DestPort=1	1	0.69
SHELLCODE x86 inc ecx NOOP,WEB-MISC HP OpenView NNM getnnmdata.exe CGI ICount parameter buffer overflow attempt	SourcePort=1, DestIP=1, SourceIP=1, DestPort=1	1	0.60
WEB-MISC Microsoft ASP.NET information disclosure attempt,WEB-MISC Apache Tomcat allowLinking URI-encoding directory traversal attempt	SourcePort=1, destSourceIP=1	1	0.50
POLICY attempted download of a PDF with embedded Flash over smtp,POLICY attempted download of a PDF with embedded Flash over smtp	SourcePort=1, DestIP=1, SourceIP=1, DestPort=1	0.5	0.84
WEB-MISC Microsoft ASP.NET information disclosure attempt,WEB-IIS WEBDAV nessus safe scan attempt	SourcePort=1, destSourceIP=1	1	0.50
ATTACK-RESPONSES 403 Forbidden,ATTACK-RESPONSES Invalid URL	SourcePort=1, DestIP=1, SourceIP=1	1	0.47

were triggered by the IDS. This resulted in many frequent alert correlation graphs such as Figure 3.14(b) which capture the Ping-Reply behaviour between servers. These graphs

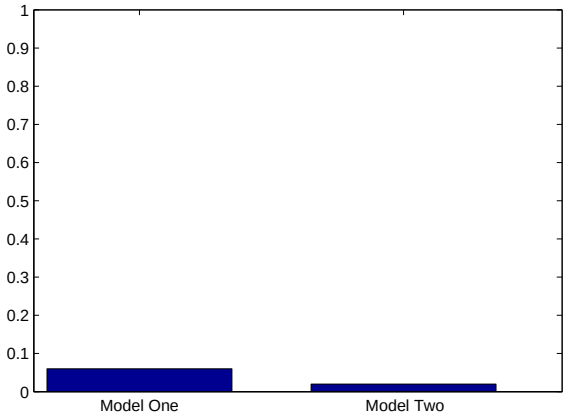

Figure 3.12: FPRC from NGDMZ1 Dataset

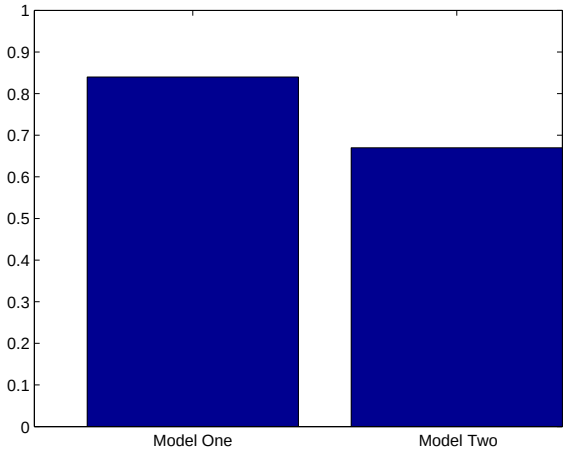

Figure 3.13: TPRC from NGDMZ1 Dataset

typically contained an average of 2 - 4 alerts occurring between 2 - 3 minutes of each other. Fig3.14(a) illustrates an alert correlation graph which captures the behaviour of an outsider sending suspicious email to a client residing on the network. After studying the network topology and configurations it was discovered that packets were routed from the outsider to the DMZ mail server and from the DMZ mail server to the internal mail server where the mail content becomes available to the local client. Many graphs (although with variations of size and noise) captured this network behaviour. Finally, Fig3.14(c) shows the alert correlation graph of the real attack launched by the attacker on a DMZ web server. Most of the attacks in this graph were targeted to exploit web vulnerabilities.

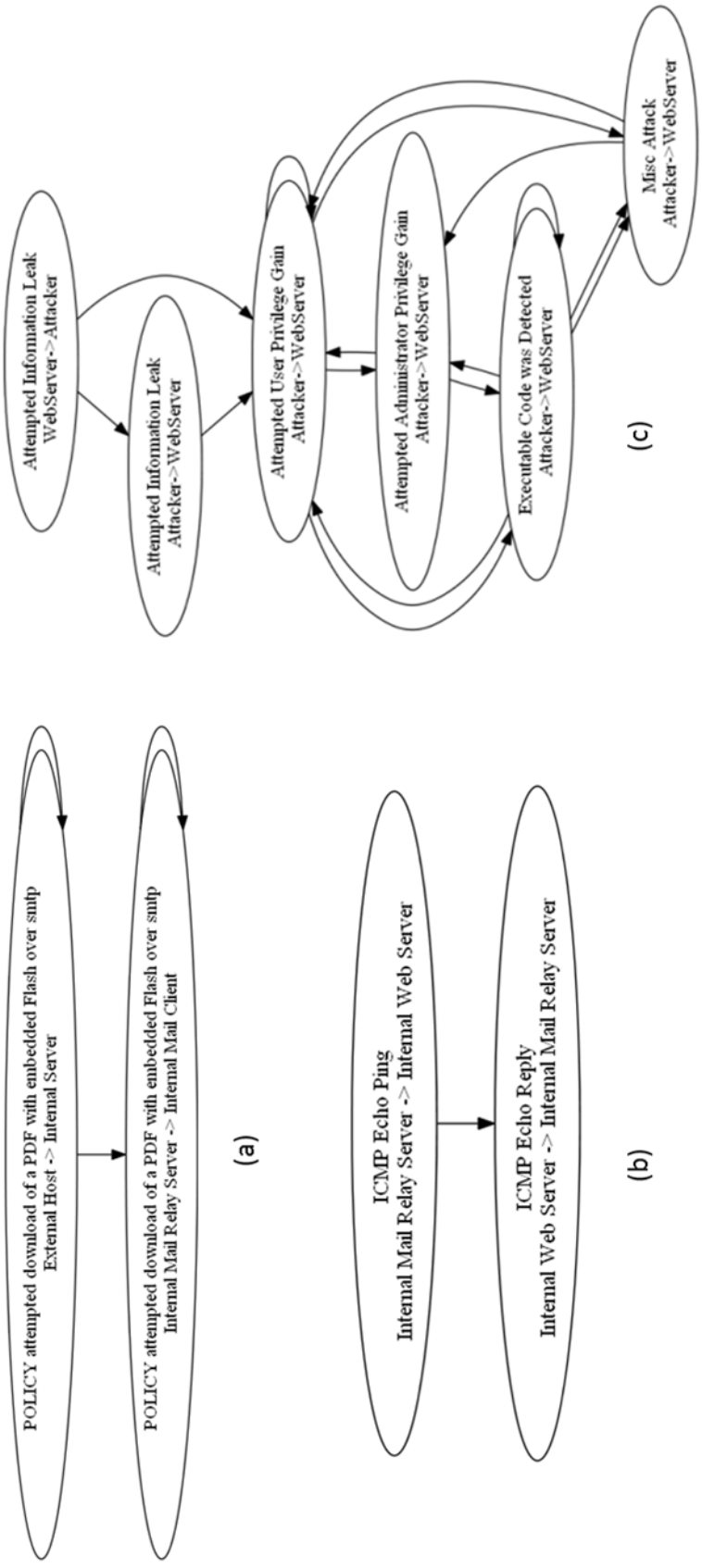

Figure 3.14: Sample Alert Correlation Graphs (Meta-alerts) derived from analysis Model 2 Analysis

3.5 Conclusion

At a high level, this objective of this chapter was to generate accurate global alerts using alert correlation to depict network attacks more efficiently. In order to achieve this, this chapter aimed to address the following question: "Can an optimised Bayesian inference correlation model analyse a large volume of security alert data whilst still providing high quality correlations despite missing and ambiguous alerts?". Two sequential based correlation models were proposed. Both used feature similarity combined with Bayesian inference to detect more high quality correlations i.e. lower false positive correlations and higher true positives. The models were tested on two datasets, both of which contained short duration attacks. Both correlation models consisted of two parameters, T_θ and C_θ which were used to optimise correlation accuracy.

For Model 1, a-priori frequent pattern mining was applied to detect frequent patterns referred to as constraints from the training data. The constraints combined with Bayesian probabilities were used to determine the correlations between intrusion types in the off-line process. The objective behind this approach was to explore automated rule generation and to discover more specific correlations. Model 1 was inspired by a prior existing model, [135]. With respect to the models performance, the results inTable 3.6 and Table 3.8 showed that in the tested environment, the time taken to analyse approximately 2,000 alerts was on average 44 seconds. With respect to the models accuracy, the results showed that there was a high true positive rate, this indicating that despite the volume of the dataset and noise in the dataset, correct correlations were still detected. The results also showed that there was also a high false positive rate during certain configurations of the model. The observation was that this high false positive rate was due to the generation of irrelevant frequent patterns which were later used in determining if two intrusion types were correlated. It is suggested that for this approach to be highly successful, rules could be validated or pruned using expert knowledge or additional constraints. In conclusion, Model 1 provides high quality correlations despite noise or missing data however, it also provides a high false positive rate and this is undesirable.

For Model 2, weighted similarity scoring was integrated into the process of constructing the Bayesian network during the off-line process. The objective behind this was to provide a faster approach than Model 1 with the likelihood of a reduced false positive rate. This was successfully achieved. The results in Table 3.6 and Table 3.8 showed that its speed performance was similar to Model 1 however, with respect to the accuracy, a significantly lower false positive rate was achieved.

In comparison to prior art, we have shown that our models have a better false positive correlation rate but slightly lower true positive correlation rate. The shows that in some cases, a trade-off in performance and accuracy must be made. This partially answers the question we aimed to address. While Bayesian inference based correlation models can provide high quality correlations, a trade-off between accuracy and performance may be required.

The on-line processes of both models was evaluated using observations. Both models showed the ability to detect alerts part of the same attack by grouping them into alert correlation graphs. In the on-line analysis, both models generated over 200 alert correlation graphs. While this shows a data reduction of approximately 85%. The major challenge in the on-line process is that given the large number of alert correlation graphs generated, a security analyst may still need support in identifying which alert correlation graphs are relevant. In this Chapter, a manual approach was used to investigate the alert correlation graphs and to determine which alerts were interesting and non-trivial.

While the initial research question was answered, another issue was raised. This relates to how to successfully determine which alert correlation graphs reflect real attacks. During the experiments detailed in this Chapter (Section 3.4.3 and 3.4.4), relevant graphs were manually selected based on a number of factors such as: the number of alert nodes contained in the graph, the network hosts involved, the duration of the graph, and the interval rate between each alert. Chapter 4 investigates how to convert such factors into a qualitative priority metric for identifying attack related graphs and filtering non-trivial graphs in order to eliminate false positive alerts.

Chapter 4

Temporal and Probablistic Outlier Metrics for IDS Alert Prioritisation

4.1 Overview

As was discovered in Chapter 3, correlation analysis successfully detects events related to the same attack. It was also discovered that given intense network traffic, a large number of alert correlation graphs each representing one or more attack scenarios may be detected. In order to determine which attack scenarios and alert correlation graphs are non-trivial, prioritisation analysis can be performed. Due to the large volume of security alert data to be prioritised, it would be infeasible, error prone and resource consuming to fulfil the prioritisation process manually, therefore automated approaches are used.

Salah et al. [146] described a prioritisation component as one which performs severity and attack analysis (Fig.4.1). Similarly, Anuar [16] described that the priority of a security alert can be determined by factors which measure - (1) the impact of the security alert on the affecting network and (2) the likelihood that the alert is a real threat. While the first factor has little relevance to research studies and is subjective to the network's assets, determining whether one or more security alerts are likely to lead up to an attack remains a challenge in the domain due to ambiguities involving the network, the attacker and the attack itself.

In this Chapter, a set of new metrics for prioritising security alerts are explored,

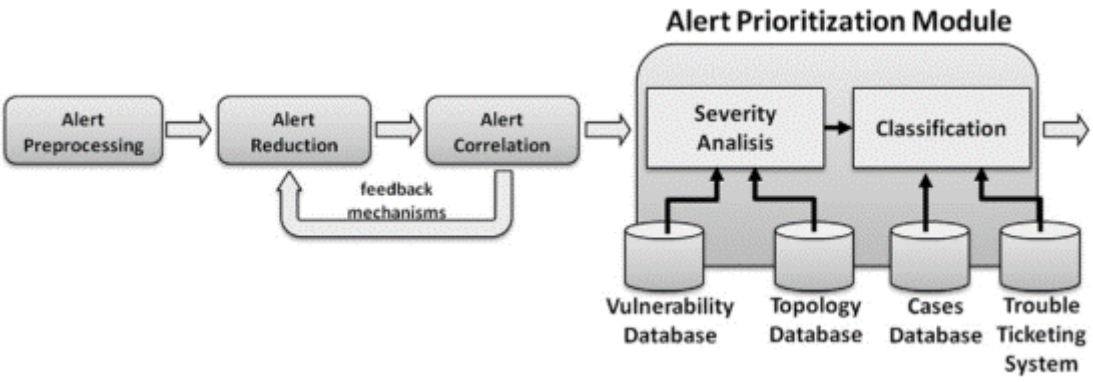

Figure 4.1: Details of a prioritisation component as proposed by [146]

proposed and evaluated. The prioritising metrics are based on the three factors that were considered earlier in Chapter 3. We defined these factors as key factors to measuring the complexity of an attack.

1. *The duration of the attack:* In Chapter 3, alert correlation graphs which were developed over various time durations were discovered. In the evaluation described in Section 3.4.4 of Chapter 3, it was observed that at least 40% of alert correlation graphs represented network *ping-reply graphs* activity. These alert correlation graphs spanned over shorter time periods in comparison to other alert correlation graph types. Following this observation we propose the following question, *"Would a metric which measures the duration of an attack be useful for measuring the complexity of an attack? For example, the longer the duration of the attack, the higher the metric value?"*.

2. *The origin of the attack:* In Chapter 3, we also observed that alert correlation graphs could be used to determine the origin of an attack given that alert correlation graphs contained alerts triggered from internal network hosts and/or external network hosts. While this would require a small amount of network knowledge, by measuring this property, we aim to answer the following: *"Would a metric which measures whether the attack is internal or external be useful for prioritising security alerts?"*.

3. *The anonymity of an attack:* Finally, it was observed that the anonymity of an

attack was difficult to characterise from the alert correlation graphs. This is mainly because no knowledge of the attack an alert correlation graph represents is known. It is unknown until user validation if the alert correlation graph is itself a false positive correlation. In this Chapter, an approach for potentially measuring the anonymity of an attack is explored.

Anuar et al. [18] described five factors for measuring the likelihood of an attack, namely - *exploitability, severity, sensitivity, similarity and frequency*. An observation made during our research study was that the exploitability, severity and sensitivity were commonly used factors for defining prioritisation metrics, for instance, these factors were considered in work by [18, 17, 16, 14] and [15]. These factors are derived from known network knowledge such as knowledge contained in the vulnerability and topology databases (illustrated in Figure 4.1). It was also observed that despite the vast research on alert correlation, knowledge learnt during alert correlation is rarely used to prioritise alerts. During the literature survey carried out, only few researchers such as [15, 14, 166] considered alert correlation knowledge when prioritising IDS alerts.

In [18], the *exploitability, severity*, and *sensitivity* metrics of an alert were measured by using network knowledge while *similarity and frequency* [18, 17, 16] were measured based on statistical analysis. For each alert, a heuristic function for IP address and port similarity was used to derive the similarity measure by computing the degree to which a given alert's IP addresses and ports were similar to another alert's IP addresses and ports. For each alert, the frequency factor was measured as the number of times the alert's type occurred. In their work, they combine the values from each factor using Analytic Hierarchy Process (AHP) to determine an alert's overall priority. Njogu et al. [122] and Nguyen et al. [114] also used the frequency factor alongside a set of other metrics. In the prioritisation approach proposed by Njogu et al. [122], four metrics were combined using fuzzy logic to determine the overall priority. In the approach by Nguyen et al. [114], a decision tree classifier was used to combine the values. When tested on real data, they recorded a 90% accuracy rate. Their model successfully identified and filtered 70% of the false positive security alerts.

A more robust alert prioritisation system is proposed by Alsubhi et al. [15, 14] who defines seven metrics for prioritising alerts. One of the metrics, an *alert relationship metric* is highly similar to the *similarity* metric. The alert relationship metric prioritises causal alerts by measuring the degree to which an alert correlates with successive alerts and similarly to the approach by Njogu et al. [122], they use a fuzzy logic to combine the output of the seven metrics.

While each of these prioritisation approaches combine a set factors, we argue that the attack information gained using the frequency and similarity factors is minimal and in some scenarios misleading. For example, it is misleading to assume alerts with high frequencies should have higher priority. A well known example of an alert type with high frequency but low priority is the *Port Scan*. Such an alert type often occurs in high magnitude without indicating an immediate threat. In addition, both the similarity and frequency metrics can be classified as similarity based correlations. Thus, they are very basic metrics. In Chapter 3 we identified these metrics as not being able to quantify alerts which are part of stealth attacks.

In this Chapter we address how to effectively use the knowledge learnt from correlation analysis to enhance prioritisation analysis. The significant difference between the prioritisation metrics and prior art is that we focus on measuring a new set of properties which can mainly be observed when alerts are correlated into groups (i.e. alert correlation graphs). More specifically, we measure temporal relationships between alerts, IP relationships and anomaly alert patterns. Since these three metrics characterise elements from the duration, origin and type of attack, we hypothesise that these metrics combined will provide better insight to an attack in comparison to the alert similarity, frequency and relationship metrics. Four new metrics are proposed based on features extracted from alert correlation graphs. Another major difference between the methods proposed in this chapter and prior art such as similarity and frequency metrics is that our proposed metrics are measured from the results of temporal correlations. In Chapter 3, it was identified that temporal correlation are more likely to capture the relationship between alerts that reflect a stealth attacks. Therefore, we also hypothesise, that if performed effectively, these new

prioritisation metrics can correctly prioritise alerts from stealth attacks. The results from Chapter 3 showed that correlating alerts into alert correlation graphs provided a better overview of a potential attack. The correlation also showed that alert correlation graph features such as their *size, structure, content and temporal properties* were useful in determining which alert correlation graphs were interesting and which were non-trivial. Each of these proposed metrics are novel.

The rest of this chapter is organised as follows: Section 4.2 describes the implementation of each metric and Section 4.3 describes the datasets and case studies used to evaluate the metrics. In Section 4.4 & 4.5, the metrics are evaluated and their effectiveness is addressed.

4.2 Technical Approach: Novel Alert Prioritisation Metrics

Figure 4.2 illustrates the process of the prioritisation metrics. Given a stream of n alert correlation graphs $G = \{G_1, G_2, \ldots, G_n\}$, four priority metrics are derived for each graph, G_i. Using a user defined threshold, graphs with lesser priorities are filtered out and the highly prioritised graphs are presented for validation. The four new prioritisation metrics are described next.

4.2.1 Attack Duration Metric (ADM)

The ADM is a metric with a value between 0 and 1 that accounts for the duration of an attack. In Section 3.2 of Chapter 3, related alerts are structured into an alert correlation graph. Alert correlation graphs with wider time ranges indicate intrusive activity occurring over a longer period of time. An alert correlation graph with an ADM value closer to 0 indicates it occurred over a short time period while an alert correlation graph with an ADM value closer to 1 indicates it occurred over a longer time period.

A sigmoid function is used to derive the ADM. Let the time-range of an alert correlation graph, G be represented as $R_t(G) = |t_0 - t_n|$ where t_0 is the timestamp of the earliest alert in G, t_n is the timestamp of the last alert added to G, and the unit measure for R_t is

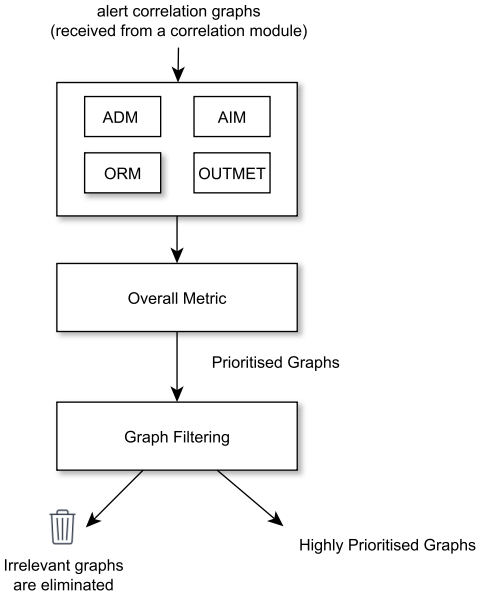

Figure 4.2: The Proposed Prioritisation Process

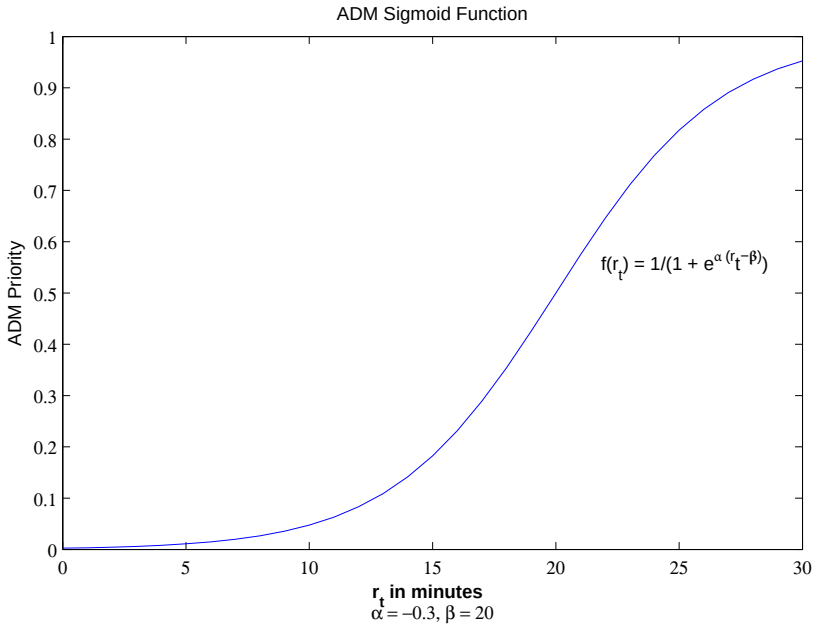

Figure 4.3: Sigmoid Function for Attack Duration Metric and Attack Interval Metric.

minutes. As $R_t(G)$ increases, the ADM value of G increases as a function of the sigmoid curve in Figure 4.3. In the function, α modulates the steepness of the curve. By default, α is selected experimentally and determines how quickly the $y-$axis increases. When $R_t(G))$ increases beyond a point, the ADM value nears its max i.e. 1 and does not exceed 1. β is also selected experimentally and represents the point on the $x-axis$ which meets the middle of the $y-axis$ (i.e. 0.5). For example using the value $\beta = 20$ as shown in

Figure 4.3, the $y-axis$ of the graph centres at 20 minutes thus indicating that two alerts with a time difference of 20 minutes would produce an ADM of 0.5.

4.2.2 Attack Interval Metric (AIM)

This metric accounts for the interval between the steps of the attack. For example, shorter intervals could indicate quick attacks such as DDoS while longer intervals could indicate more stealth attacks. Although longer intervals do not take precedence over shorter interval, a graph with a smaller interval rate is assigned a lower value and vice versa.

The AIM can provide vital information. For example, an alert correlation graph with a high ADM and a low AIM indicates that a large volume of related alerts occurred rapidly between each other over a long period of time. This is a common attribute of a severely extreme DDoS attack.

$$AIM(G) = \frac{\sum_{k=1}^{n} t_i}{n} \tag{4.1}$$

The AIM of an alert correlation graph, G, is the mean time interval between the alerts in an alert correlation graph. For explanatory purposes, let all the alerts in an alert correlation graph be placed as an ordered set of alerts, $A = a_0, a_1, ..., a_n$. $I_t(G)$ will be the set of all the time intervals between the alerts such that $t = t_0, t_1, ..., t_n - 1$ where $t_i = a_{i+1}(t) - a_i(t)$. The interval rate between a set of related events has been previously and successfully used in prior art [32]. The mean interval (measured in minutes) is used to derive a value between 0 - 1 by using the same sigmoid function as above (Figure 4.3).

4.2.3 Outgoing Rate Metric (ORM)

This metric accounts for the origin of the attack. Typically, attacks may be internal or external. Domain knowledge is required to use this metric. For each alert correlation graph, FM is the ratio of intrusions triggered by internal hosts to the total number of intrusions in the alert correlation graph. This value also ranges from 0 - 1. Therefore, lower values, indicate most intrusions are triggered by external attackers, a value of 0.5 indicates there is a consistent communication between internal and external hosts (which

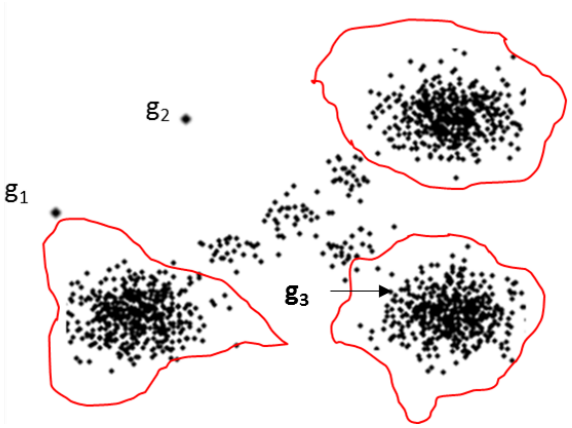

Figure 4.4: Outlier Example

could further indicate command and control activity or ping-reply activity) while values closer to one indicate internal attacks.

Let the set of alerts triggered by internal hosts in a single alert correlation graph, G be denoted as $A'(G) : A' \subseteq A(G)$ i.e. $A(G)$ is a set of all the alerts contained in G. then:

$$FM(ACG) = \frac{|A'(G)|}{|A(G)|} \tag{4.2}$$

4.2.4 Outmet

The Outlier Metric (Outmet), accounts for the unusualness of the attack depicted by the alert correlation graph.

The Outmet of an alert correlation graph is measured using the local outlier factor algorithm proposed by Breunig et al. [34]. Given a set of alert correlation graphs, the more an alert correlation graph differs to the set, the higher its Outmet priority. Figure 4.4 illustrates a set of alert correlation graphs represented in Euclidean space based on their similarities to each other. g_3 represents an alert correlation graph which shares common attributes with many other alert correlation graphs. Therefore, this graph as well as the cluster surrounding it will have lower Outmet priorities. On the other hand, g_1 and g_2 differ to many of the observed correlation graphs, therefore they have higher Outmet priorities. Given an alert correlation graph database, GD, the Outmet for each alert correlation graph G, is computed over 5 steps:

1. $\forall G_i \in GD : G_i \neq G$, derive the structural distance between G and G_i. The structural distance focusses on the differences between two graphs nodes and edges. For simplicity purposes, this ignores the temporal order of the alerts in the graph as well as the structure of the graph. In addition, it is only concerned with the *types* of alerts contained in the graphs and whether they have any in common. Thus, given two alert correlation graphs G_1 and G_2, the structural distance is defined using the Jaccard similarity coefficient [134]:

$$StructDist(g_1, g_2) = 1 - \frac{g_1(V,E) \cap g_2(V,E)}{g_1(V,E) \cup g_2(V,E)} \tag{4.3}$$

   Where V and E are the nodes and edges contained in a graph.

   While other distance metrics for computing graph distances such as Graph Edit Distance [134], Cosine Similarity [134] were considered, for simplicity and efficiency purposes, the Jaccard distance proved effective.

2. $\forall G_i \in GD : G_i \neq G$, derive the k^{th} reachability distance between G and G_i. G_i is an alert correlation graph in the graphset, GD that is not G and $kdist(G)$ is the distance between G and it's k^{th} nearest neighbour.

$$rd_k(G, G_i) = \max\{kdist(G), d(G, G_i)\} \tag{4.4}$$

3. Derive the local density of G: This is the inverse of the average reachability distance of G which is defined in Equation 4.5. $N_k(G)$ is the set of alert correlation graphs that are less than k distance to G.

$$lrd(G) := 1/\frac{\sum_{G_i \in N_k(G)} rd_k(G, G_i)}{|N_k(G)|} \tag{4.5}$$

4. Derive the local outlier factor, $lof(G)$:

$$lof(G) = \frac{\sum_{o \in N_g} \frac{lrd_g}{lrd_{g_j}}}{|N_g|} \tag{4.6}$$

5. Derive the Outmet from the normalised local outlier factor.

$$nlof(g) = \frac{lof(G)}{\max\{lof(G_i)\}_{i=0}^{|GD|}} \tag{4.7}$$

In the next section a set of case studies which provide datasets that require alert prioritisation are described and in Section 4.4 the described metrics above are evaluated by measuring how effectively the datasets are prioritised.

4.3 Datasets

To evaluate the prioritisation metrics, a set of alert correlation graphs which require prioritisation are required. Therefore, we use the output from the correlation mode,l *Model 2*, described in Section 3.2.3. We also use two of the priorly described datasets - DARPA 2000 and Northrop Grumman I described in Section 3.3.1 and 3.3.2 respectively. A summary of how alert correlation graphs are generated from these datasets is given:

Using the DARPA 2000 dataset, the IDS alerts from the LLSDOS1 and LLSDOS2 attack scenario (described in Section 3.3.1) are first analysed using *Model 2* (also described in Section 3.4.3) with configuration parameters: $C_\theta = 0.5$ and $T_\theta = 180mins$. During the on-line analysis, 54 alert correlation graphs are generated. The generated alert correlation graphs are to be analysed and used in the evaluation of the proposed prioritisation metrics.

Using the Northrop Grumman I dataset, the IDS alerts from the NGDMZ1 and NGDMZ2 attack scenario (described in Section 3.3.2) are also analysed using *Model 2* with the same configuration parameters as above. During the on-line analysis, 30 alert correlation graphs were generated. The generated alert correlation graphs are to be analysed and used in the evaluation of the proposed prioritisation metrics.

In addition to the above datasets, we present an additional case study and dataset in

order to improve the validation process. This is described next in details.

4.3.1 Case Study III: Northrop Grumman II

The Northrop Grumman case study introduced in Section 3.3.2 consisted of four attack scenarios where only two were described and used to evaluate the correlation models. This Chapter uses a third attack scenario to evaluate the proposed prioritisation approach. Figure 4.5 illustrates the network with respect to this scenario and Table 4.1 shows the steps of the attack.

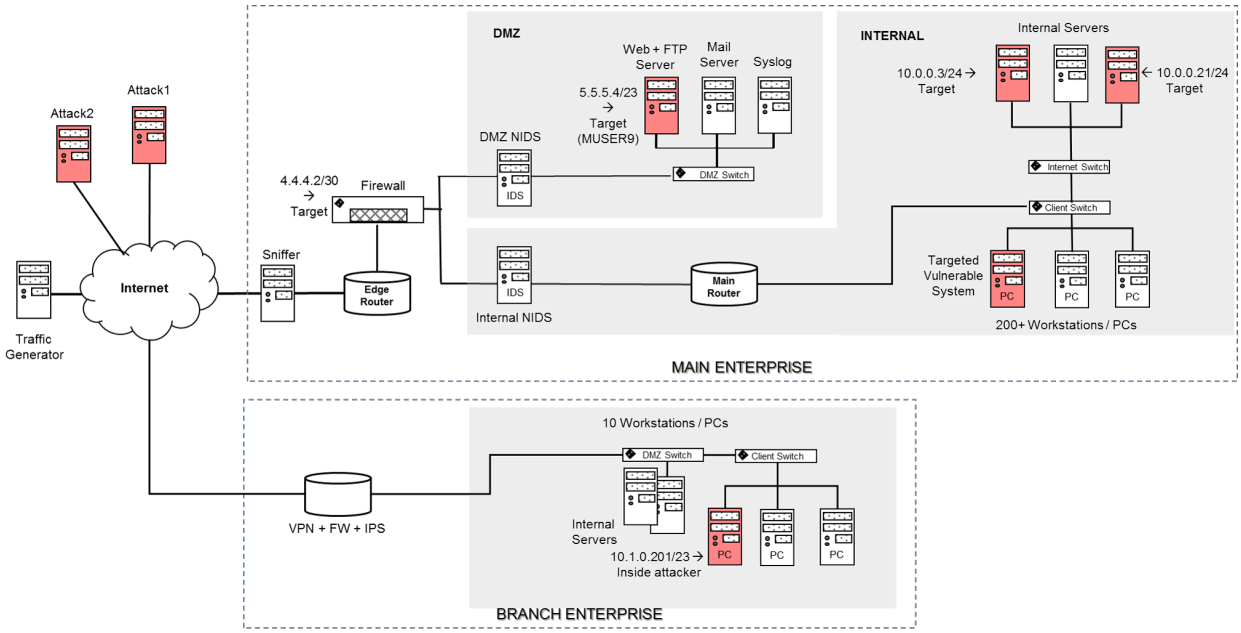

Figure 4.5: Network Architecture of Simulated Network from Northrop Grumman Case Study with respect to scenario NGINT

NGINT

The network was captured over a span of approximately 24 hours on Tuesday, 6 March 2012, from 00:00AM to 23:59PM, GMT whereby the entire attack (i.e. Steps 1 - 17 of Table 4.1) occurred between 11:00AM to 14:00PM. From the knowledge of the attack it is assumed Step 1 - 3 and Step 15 - 17 of Table 4.1 and will not be traceable from the intrusion logs. This is because Steps 1 - 3 depict an internal attack on the branch enterprise which is outside the scope of both IDS. Similarly Steps 15 - 17 are an attack against the firewall outside the scope of the IDS.

The logs from the Internal IDS and DMZ IDS were collected. A total of 1,280 alerts

were triggered by the Inside DMZ and 369,129 by the DMZ. A total of 370,409 intrusion alerts were generated. Aside the main attack other network activities also occurred within the time-frame. Normal network activity also triggered benign alerts.

Table 4.1: Phases of the NGINT Attack During the Northrop Grumman Cyber-range Experiment

Steps	NG Internal Attack Activity (NGINT)
1	Backtrack started on Branch client 10.1.0.201 (insider)
2	Quick scan from 10.1.0.201 of local subnet
3	Intense scan from 10.1.0.201 of local subnet
4	Quick scan from 10.1.0.201 of 10.0.0.0/24
5	Quick scan from 10.1.0.201 of 10.0.2.0/24
6	Unsuccessful attempts to break into 10.0.0.3 and 10.0.0.21 from 10.1.0.201
7	Unsuccessful attempts to get MUSER9 to open malicious email from geek@coldmail.com with reverse backshell payload in attached PDF
8	Spearphishing attempt via email to MUSER9 from geek@coldmail.com containing weblink to malicious website at Attack1 with multiple exploits. The attempt was unsuccessful because the malicious website recognized as "dangerous" by AV (Microsoft Forefront)
9	Successful spearphishing attempt via email to MUSER9 from geek@coldmail.com with link to malicious website at Attack1
10	Malicious site exploits just one vulnerability which was not detected by AV (Microsoft Forefront)
11	Attack1 gained full control over MUSER9's computer (5.5.5.4) and collected: 1) system information and running processes 2) password hashes and SSH keys and 3) screenshot of MUSER9's desktop
12	Download files (boot.ini) from MUSER9 to Attack1
13	Attack1 attacker obtains admin rights on MUSER9
14	Attack1 explores MAIN intranet using MUSER9's computer as pivot and obtains access to MFILE file share
15	Several short DOS attacks from Attack2 against 4.4.4.2.
16	DDOS from traffic generator against 4.4.4.2
17	DDOS from traffic generator against 4.4.4.2

These alerts are analysed by the correlation *Model 2* described in Section 3.2.3 using configuration parameters: $C_\theta = 0.5$ and $T_\theta = 30mins$. 1995 alert correlation graphs were generated. At least one alert correlation graph should contain one or more alerts that depict the main attack. The generated alert correlation graphs are to be analysed and used in the evaluation of the proposed prioritisation metrics.

4.4 Evaluation

4.4.1 Metrics

Our objective is to illustrate the usefulness of the proposed metrics in filtering out false positive alerts. Similarly to the evaluation in Chapter 3, the false positive and true positive evaluation metrics are used. We define these metrics with respect to prioritisation.

True Positive Rate (TPR)

A good performance will indicate a higher TPR. This is measured as:

$$\text{TPR} = \frac{\text{TP}}{\text{P}} = \frac{\text{\# of correctly prioritised alerts}}{\text{\# of true positive alerts}} \tag{4.8}$$

False Positive Rate (FPR)

This compares the false positive rate in the alert dataset before and after applying the proposed prioritisation metric. A good performance will indicate a lower FPR after the metrics have been applied. The false positive rate is measured below:

$$\text{FPR} = \frac{\text{FP}}{\text{N}} = \frac{\text{\# of incorrectly prioritised alerts}}{\text{\# of prioritised alerts}} \tag{4.9}$$

Environment

During the experiments, the prioritisation models were developed in Java and evaluated on a 64-bit Windows System with an Intel(R)Core i5 CPU processor at 2.40GHz, JVM 1.4.2 and 6GB for maximum heap memory.

4.4.2 Evaluation One: Darpa 2000 LLDOS2

13,806 alerts were generated. A large volume of alerts with type *"DNS response for rfc"* were generated. We noted that this alert type was due to snort mis-configurations therefore a manual process was used to eliminate these alert types. this reduced the alert data to 856 alerts. In addition, 54 alert correlation graphs were generated from 856 alerts. In order to provide a detailed analysis of each metrics performance, each proposed metric is first used to prioritise the alert correlation graphs prior to using Outmet to combine the priority values into an overall priority.

Figures 4.6 - 4.9 show timeseries graphs of the alerts expected to be highly prioritised over time against the alerts which were actually prioritised by each metrics. A highly prioritised alert is an alert with a prioritisation metric greater than P_θ. Let $P(a_i)$ be the priority of an alert a_i then:

$$P(a_i) = \begin{cases} \text{High Priority} & f(a_i) \geq P_\theta \\ \text{Low Priority} & \text{Otherwise} \end{cases} \tag{4.10}$$

The function $f(a_i)$ represents the priority value achieved from prioritising an alert (a_i) with either the ADM, AIM, FM, Outmet metrics. In this evaluation, P_θ is set to 0. In each graph, the 3 hour attack is split across 21 time-windows of length 5 minutes where the denial-of-service attack occurs at Time T_{20}.

Results and Discussion

The DARPA attack occurred over a short time period, consisting a series of persistent exploits and finally a denial of service attack both carried out by a set of compromised hosts on the victim's network. The outgoing rate metric is expected to highly prioritise alerts which represent attacks that are carried out by internal network hosts. In Figure 4.6, it is shown that the outgoing rate metric's actual prioritised alerts are consistent with the expected prioritised alerts with a True Positive Rate of 92%. For a comparative study, the frequency metric (described in Section 4.1) is also used to prioritise the same alerts. The results from the frequency priority metric is shown in Figure 4.7. This metric focusses on highly prioritising intrusion types that occur in large volumes therefore, only the alerts which represented the denial of service attack were highly prioritised. It is shown the actual and expected priority of the alerts is consistent with a True Positive Rate of 70%.

The ADM and AIM metrics had a lower performance in the DARPA attack scenario analysis. Both the ADM and AIM focus on prioritising alerts which represent intrusions carried over longer periods of time with wider time intervals. Since the DARPA attack occurred over a short period of time i.e. approximately 3 hours, the attack durations and interval rates observed in the alert data were equivalently low (as expected). Figure 4.8,

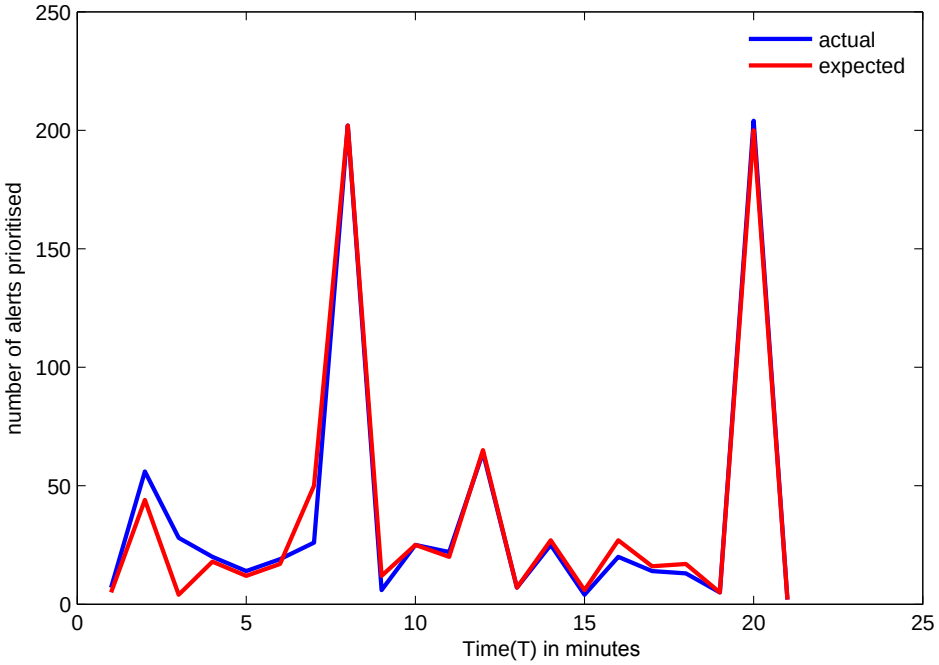

Figure 4.6: Performance of Outgoing Rate Metric on DARPA Dataset showing Actual vs Expected Priority Distribution

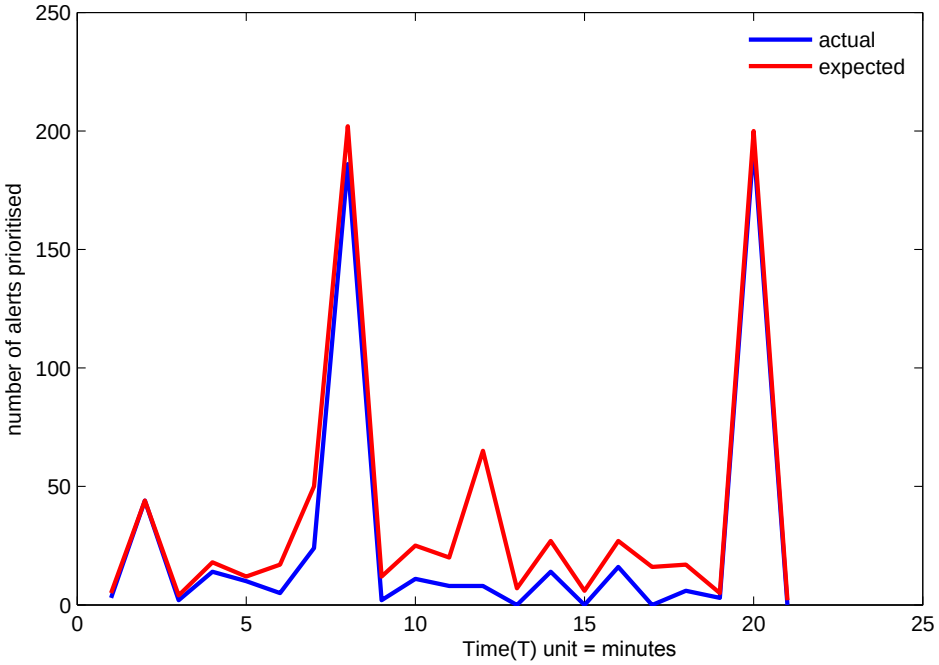

Figure 4.7: Performance of Attack Frequency Metric on DARPA Dataset showing Actual vs Expected Priority Distribution

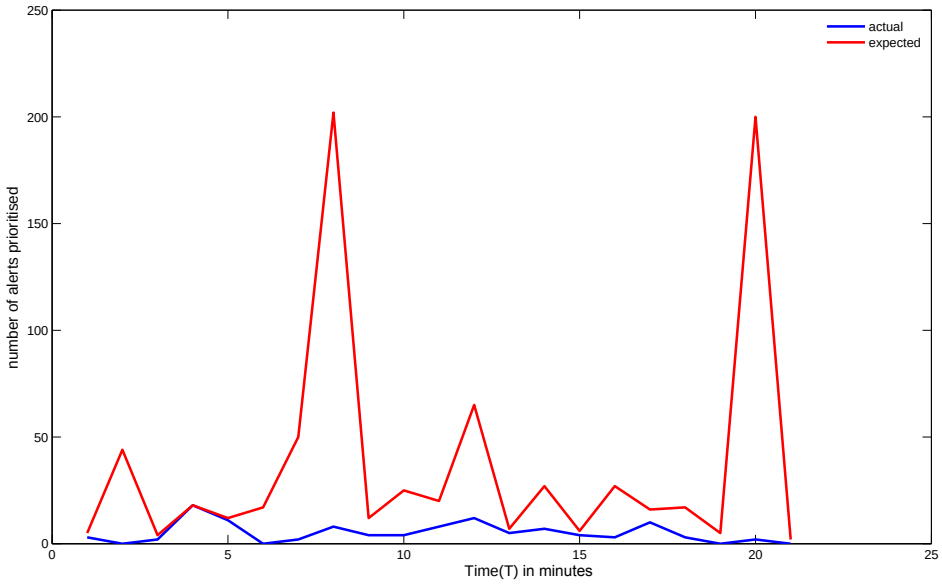

Figure 4.8: Performance of Attack Duration Rate Metric on DARPA Dataset showing Actual vs Expected Priority Distribution

shows the results from prioritising the alerts with ADM. The AIM failed to prioritise any alerts as high priorities.

The priorities from ADM, AIM and FM are combined using Outmet. Figure 4.9 shows the overall priorities. Outmet successfully prioritised all 781 alerts correctly. It was observed that Outmet produced a higher performance because it identified all alerts as true positives. While it produced a high true positive rate of 100%, a false positive rate of 8% was also produced.

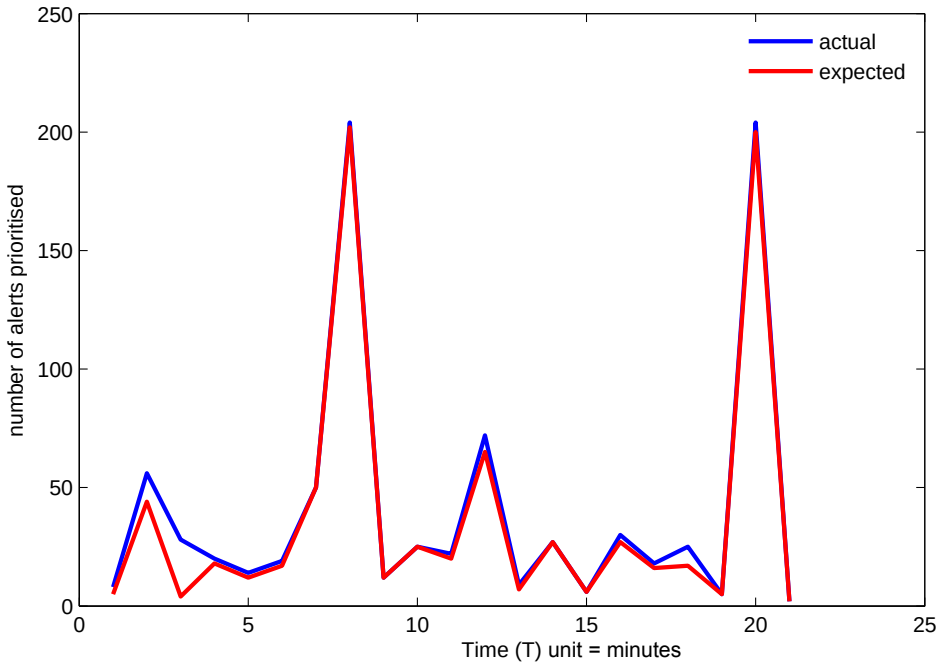

Figure 4.9: Performance of Outmet on DARPA Dataset showing Actual vs Expected Priority Distribution

4.4.3 Evaluation Two: Northrop Grumman 2012 DMZ Attack

Figure 4.10 shows a timeseries graphs of the alerts expected to be highly prioritised. Similarly to the evaluation of the DARPA dataset in the previous Section (Section 4.4.2), an alert is highly prioritised if the value assigned to it is greater than 0.5. These alerts represent the attackers intrusion attempts to exploit the Web Server by persistently attempting a range of web exploits on the targeted network host. In each figure, the 24 hour attack is split into time-windows of length *5 minutes* where the web exploit activity occurs between $T_{100} - T_{150}$. In order to provide a detailed analysis of each metrics performance, each proposed metric is first used to prioritise the alert correlation graphs prior to using Outmet to combine the priority values into an overall priority.

Results and Discussion

In this scenario, the attack duration, attack interval, outgoing rate and frequency metrics failed to prioritise the appropriate alerts. This is likely as a result of this attack being more stealth than the attack in the evaluation described in Section 4.4.2. In this scenario, the attacker persistently attempts a range of buffer overflow and executable code techniques

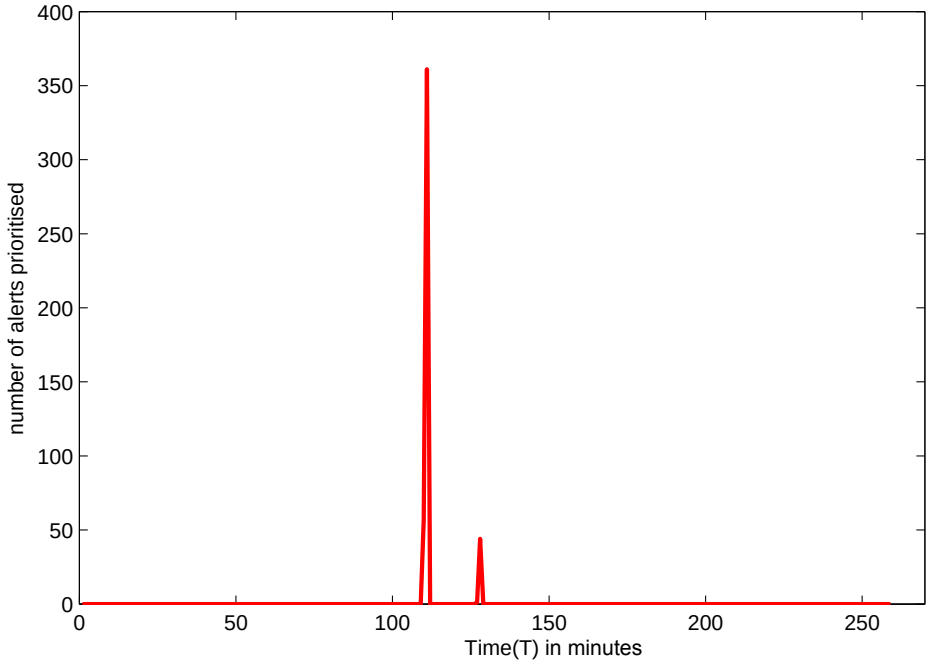

Figure 4.10: Expected Priority Distribution for Northrop Grumman II DMZ Attack Dataset

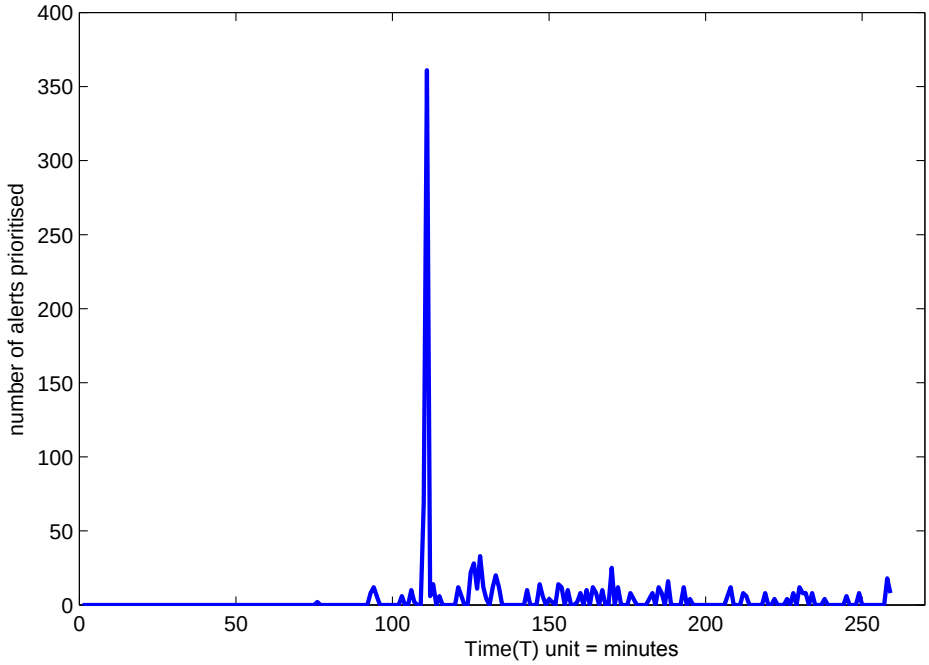

Figure 4.11: Actual Priority Distribution for Northrop Grumman II DMZ Attack Dataset

which only triggered a low volume of alerts. This caused the intrusion alerts to bypass the frequency metric. In addition, the attacker uses an off-site host thus the outgoing rate

metric fails to prioritise the alerts triggered by the attacker. The attacker also performs the web exploit attack within short time periods of 1-2 minute bursts, bypassing the attack duration or attack interval metrics.

Outmet however, achieved a True Positive Rate of 96.8% by successfully prioritising the alerts as expected. It was observed that Outmet successfully prioritised the web-exploit alert correlation graphs because they were anomalies in comparison to many other alert correlation graphs in the dataset which represented benign network activity. This is shown in Figure 4.11. It was also observed that a set of alerts which were expected to have low priorities were highly prioritised. In this scenario, all alerts related to the web-exploit were grouped into the same alert correlation graph. Thus while a TPR of 96.8% was achieved, a false positive rate of 12.3% was also achieved.

4.4.4 Evaluation Three: Northrop Grumman 2012 Internal Attack

This attack consisted of high network activity where only 0.25% of the alerts generated depict the attack described in the case study.

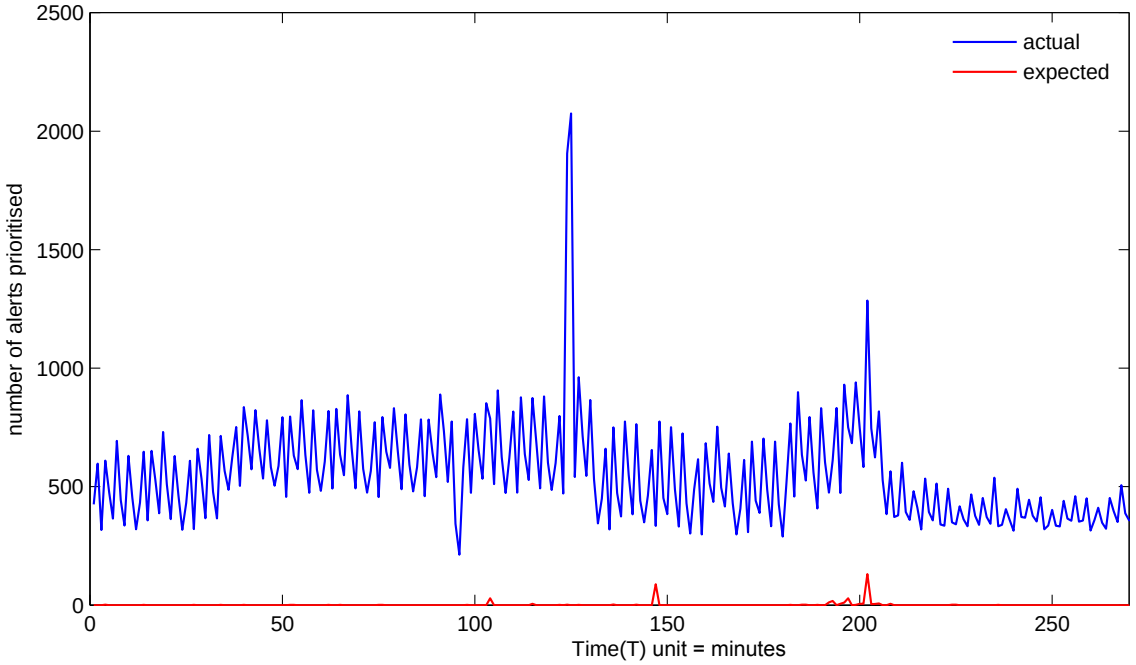

Figure 4.12: Prioritisation Results using Outgoing rate Metric on the Northrop Grumman II Internal Attack Dataset

Overall, the outgoing rate, attack duration metric and Outmet prioritised approximately 150,000, 2000 and 3000 alerts respectively. This resulted in a true positive rate of 78%, 38% and 57% respectively. The number of alerts assigned high priorities by the outgoing rate and attack duration metrics greatly exceeds the ideal quantity. This resulted in a high false positive rate of 98% for both outgoing and attack duration metric. The false positive rate for Outmet was significantly lower at 1.98%.

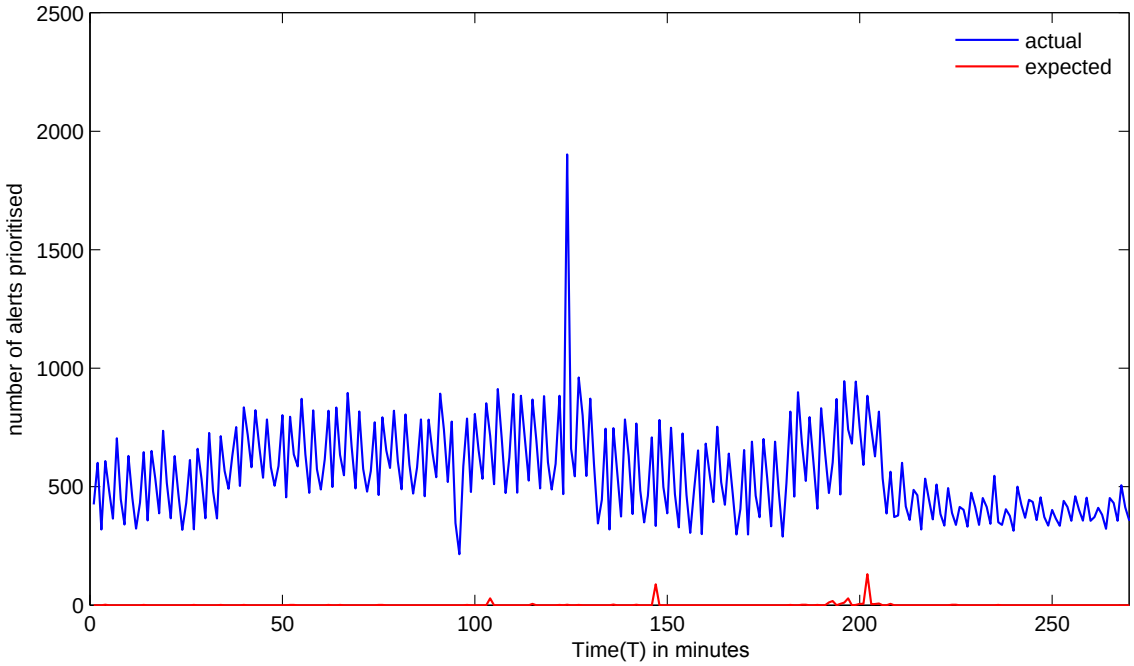

Figure 4.13: Prioritisation Results using ADM on the Northrop Grumman II Internal Attack Dataset

A Comparison between Outmet and Prior Art

In this scenario, the comparative frequency metric which was used in work by [16] failed to prioritise any alerts correctly. It however, prioritised a large number of irrelevant alerts resulting in a true positive rate of 0% and a false positive rate of 53%. This aligns with the argument in Section 4.1 that alerts representing stealth attacks may not occur in large volumes and that furthermore, alerts which occur in large volumes are not always real threats.

From the results, Outmet has the most accurate performance. More specifically, Outmet showed a low false positive rate and a high true positive. Similarly to the scenario in

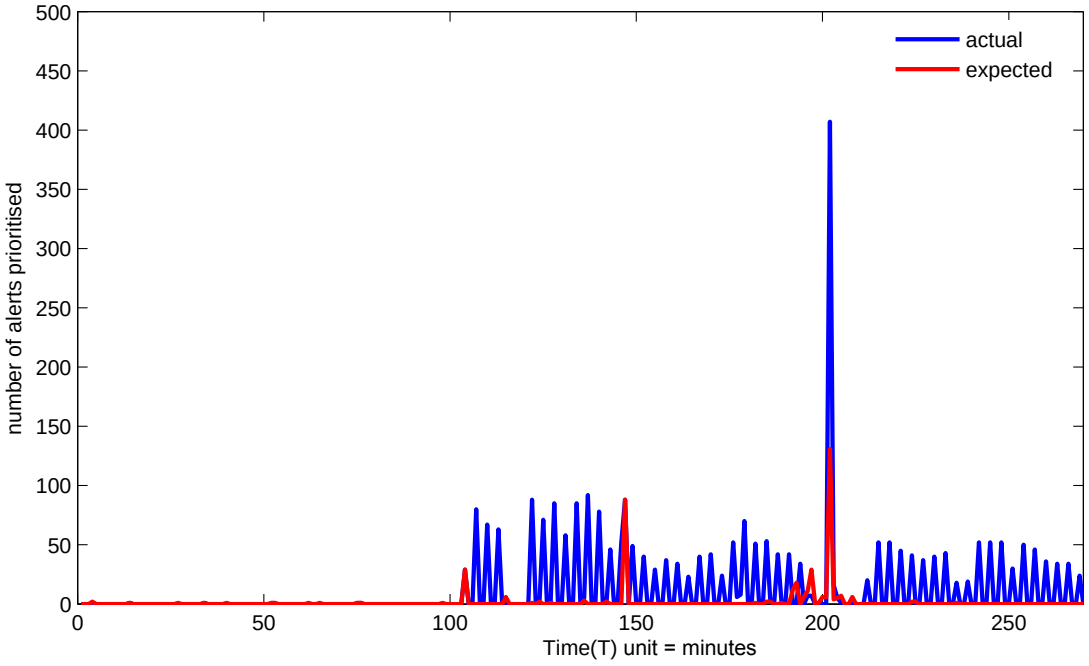

Figure 4.14: Prioritisation Results using Outmet on the Northrop Grumman II Internal Attack Dataset

Section 4.4.3, it was observed that the alerts which represented the attack in this scenario were aggregated into 3 - 5 alert correlation graphs. Since the attack occurred over a short time period and was performed by an external attacker metrics such as the interval metric and outgoing rate metric were not suitable.

4.5 Conclusion

In this Chapter a novel statistical prioritisation technique for quantifying the temporal and spatial characteristics of an attack were proposed for discovering malicious and attack behaviour. The first metric, attack duration metric (ADM) was based on a sigmoid function which assigns high priority values for a set of related alerts or events occurring on the network over a long period of time. Similarly the attack interval metric used in conjunction with the ADM provides additional information of the volume of an attack. Both metrics were proposed to work effectively for detecting denial of service activity. Next, a heuristic metric for detecting potential botnet activity or any attack which involves hosts on the

victim networks as the source of the attack was proposed and finally, a prioritisation metric based on probabilistic outlier analysis was proposed. This metric, Outmet categorises anomaly network activity as malicious attack behaviour.

The metrics were individually tested on datasets collected from three case studies consisting of denial-of-service attacks, a brute force web-exploit and a stealth host compromise. In general, it was observed that the most effective metric was Outmet which showed the ability to prioritise a range of alerts representing different attacks. The least effective metric was the interval rate metric. In addition, the metrics were compared to a simple pre-existing metric. The results showed that in stealth attacks, the proposed metrics in this chapter outperformed the pre-existing metric.

A potential bottleneck to the prioritisation process is that it is highly dependent on alert correlation and the generation of a set of alert sequences of graphs which each fully or partially depict an attack. Given poor quality correlation, the prioritisation process may be significantly affected negatively.

In summary, the primary objective of this Chapter was to prioritise intrusion activity in order to (a) eliminate false positive intrusion activity and (b) to reduce the challenges involved in gaining attack insights due to the volume and complexity of intrusion activity. Chapter 5 takes a different approach to improving attack insights. In Chapter 5, intrusion activities are grouped into clusters in order to provide attack discoveries. In order to achieve this, a partition-based clustering approach for grouping intrusion activity into clusters in real-time is explored. In addition to providing attack information, this approach is aimed to be used as an aid in the discovery of attack patterns which are commonly observed in known and unknown network attacks. The primary objective is to be able to gain further attack insights.

Chapter 5

Real-Time Clustering of Attack Pattern Graphs for the Discovery of Related Attack Behaviour

5.1 Introduction

Ning et al. [115] identified the importance of developing techniques to deal with the overwhelming information contained in a set of alert correlation graphs. In the previous Chapters(3 and 4) it was shown that an intense volume of intrusion activity on a network could result in a high volume of complex alert correlation graphs each potentially containing a high volume of nodes and high node interconnectivity. Since such a large volume of data is difficult for a system analyst to interpret, a prioritisation approach for filtering trivial graphs was proposed in Chapter 4. However, as the results showed, false positive filtering could lead to the filtering of valuable graphs mislabelled as trivial. In order to avoid this issue, a different approach to addressing the overwhelming information is proposed in this Chapter. Generally, organizing data into clusters reveals internal structures of the data. In this Chapter, it is hypothesised that the application of cluster analysis will be suitable for organising similar alert correlation graphs into groups. This could also successfully aid in discovering consistent characteristics which are commonly observed in specific attacks, for example *"which attack features best describe a denial-of-service*

attack or worm propagation?".

Ning and Xu [120] first explored the potential of clustering alert correlation graphs by using Graph Edit Distance to perform basic similarity analysis between graphs. However, in the last decade since their approach was proposed, very little research has been done in clustering alert correlation graphs. This is likely as a result of the difficulties presented in graph clustering. Aggarwal et al. [5] identified that the underlying structure of graphs introduces major challenges during cluster analysis due to the fact that many traditional clustering algorithms which are suitable for vector based objects require statistical measures such as vector sums, euclidean distances, probability distributions and averages. While these can be easily derived for vectors, these cannot be easily extracted from graphs.

In this Chapter, two approaches were investigated for performing cluster analysis on alert correlation graphs. Firstly, adapting the graphs to fit traditional clustering algorithms was considered. Bunke and Neuhaus [37] described in their work that while most traditional clustering algorithms are purpose built for vector structures, by converting graphs to vectors, many clustering algorithms become applicable. Thus, a method for embedding Graphs in Vector Spaces by Means of Prototype Selection which was studied extensively in [37, 138, 36] was explored. In this approach M graphs are reduced to M $N-$dimensional vectors where N represents a uniformed dimension for describing M graphs. This dimension reduction can be described as a multi-dimensional scaling of the $M \times M$ distance matrix between M graphs into an $M \times N$ matrix. While their work shows that clustering using graph embedding in vector space outperforms some traditional approaches, we identified that graph embedding in vector space will not scale over an evolving stream of graphs. Since alert correlation is a real-time application, a key objective in this research is to be able to simultaneously perform alert graph clustering in real time.

Thus, in the second approach, adapting a real-time traditional clustering algorithm to fit the graph data was considered. Prior art [37] showed that traditional clustering algorithms such as KNN classifiers which use minimal statistics can be successfully adapted

to the graph domain.

The contributions of this Chapter are as follows: Firstly, we focus on extending an algorithm - "CluStream" [3] for clustering an evolving continuous stream of data to suit the graph domain. This Chapter also explores the possibility of using alert correlation graphs to build attack patterns. An attack pattern is a generic representation of an attack that commonly occurs in specific contexts [108]. Fernandez et al. [58] describes that attack patterns provide a coherent benchmark for identifying attacks on the network. We hypothesise that a cluster representative is a potential attack pattern.

In the next section the adaptation of the *CluStream* algorithm is described. We experiment with two datasets - Northrop Grumman data and a competition challenge, we measure the quality of the clusters and how the clusters evolve over a period of time. Following, the proposed clustering analysis described above, we propose a heuristic method to extracting attack patterns from a set of clustered alert correlation graphs.

5.2 Proposed Approach: Online Clustering of Alert Correlation Graphs

Figure 5.1 illustrates the clustering process. The graph refining and clustering process are described in this section. The graph refining component receives a stream of alert correlation graphs from the correlation model described in Chapter 3.

5.2.1 Graph Refining

Given a stream of alert correlation graphs which each represent one or more attack scenarios, we propose a method to automatically extracting the attack pattern of an alert correlation graph. The attack pattern is a *"more abstracted/generalised version of the alert correlation graph"* which provides an easier approach to comprehending the attack the alert correlation graph represents. In this proposed approach, a more abstract representation of an attack pattern learnt from alert correlation graphs is developed.

Attack Steps

Let an Attack Pattern Graph be represented as $pG = (pV, pE, pI)$. In the set of vertices,

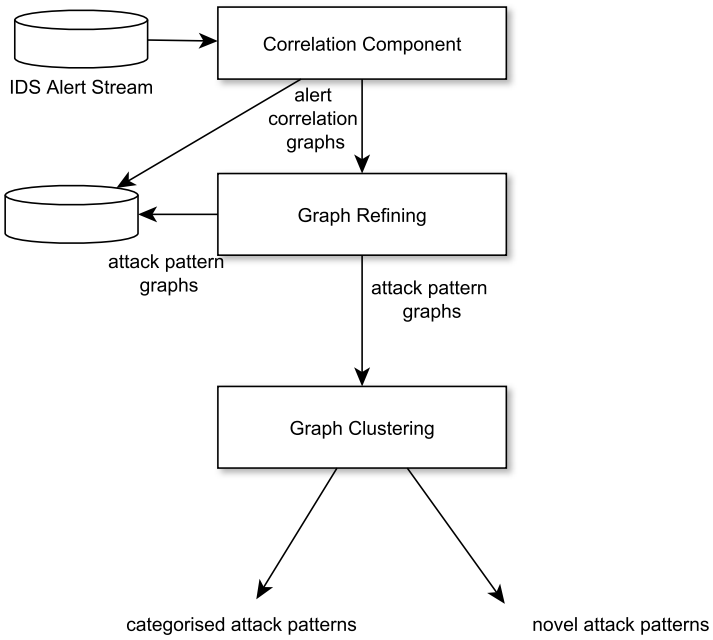

Figure 5.1: Clustering Process

pV in the attack pattern graph, each vertex $pv \in pV$ represents a step in an attack and each step is represented by a type of intrusion. We revisit the definition of an alert as a 6-tuple where α_6 is the intrusion type of the alert. Let an alert correlation graph $aG = (aV, aE)$ be considered. The set aV is an ordered set of all 6-tuple alerts which are contained in aG. Then, an attack pattern graph vertex pv is created for every one or more consecutive alerts in aV with the same source IP, target IP, target port and intrusion type.

While a vertex in an attack pattern is only represented by the intrusion type, the remaining attributes are used to aggregate and label the edges. For every alert vertex av_i in an alert correlation graph there is a bijective mapping to a vertex, pv_i in its respective attack pattern graph. Therefore for every edge ae_{av_i,av_j} there is a mapping to the source vertex pv_i and the target vertex pv_j. Two edges are aggregated if they share the same endpoints. The set pE contains all aggregated edges from the alert correlation graph.

In an alert correlation graph each edge e_{av_i,av_j} is labelled based on a quantitative measure of the strength of the correlation between av_i and av_j. In its respective attack pattern graph, each edge is labelled using a qualitative measure of the similarity between its end points. Each edge label is a 3-tuple (S_1, S_2, S_3) where S_i is a piece of information

made up of a *qualitative measure* and an *attribute*. The source IP, target IP and target port are the attributes considered where S_1 is associated with source IP attribute, S_2 with the target IP and S_3 with target port. In addition, three quantitative measures are considered.

1. *Identical:* S_i is given the value "identical" if the i_{th} attribute value for both endpoints are identical. This is represented by concatenating the qualitative value with the attribute such as "Identical target IP", "Identical source IP" or "Identical target port".

2. *Similar:* If the i^{th} attribute value for both endpoints is not identical, however, the similarity is greater than a defined threshold such as 0.5, it is said that both endpoints have a "similar" attribute. Thus, S_1, S_2, S_3 would be represented as "similar target IP", "similar source IP" or similar target port" respectively. We use the similarity metrics defined in [157].

3. *Different:* If however, the i^{th} attribute value for both endpoints is less than a defined threshold, the *different* qualitative measure is used.

Attack Features

In an attack pattern graph $pG = (pV, pE, pI)$, pI is the set of useful attack information. This describes the conditions under which the attack pattern may be observed. Temporal and relevant attacks related features, in particular, those used to define the priority metrics in Chapter 4 are automatically extracted from the respective alert correlation graph of the attack pattern graph.

5.2.2 Clustering Similar Attack Patterns

Task 1: Given a set of attack pattern graphs, $G = \{g_1, g_2, \ldots, g_n\}$, the initial task is to split G into k Clusters, $C = \{C_1, C_2, \ldots, C_k\}$ where $k \leq n$ such that $C_i \in C$ contains only similar attack pattern graphs.

The process is initialised with an adaptation of the k-means++ algorithm [20]. In order to partition G into k clusters, two key steps are performed:

Random initialisation of centroids

Let $M = \{M_1, M_2, \ldots, M_k\}$ be the set of centroids in the cluster space. Each cluster, C_i is represented by a centroid, M_i. Given G, k attack pattern graphs from G are selected at random using the k-means++ seeding technique to fill the set M.

Iterative member and centroid selection until convergence

This step is iteratively performed until k-means++ converges. In this implementation of k-means++, k-means++ is said to converge when the selected centroids no longer change. The iterations until convergence involves two steps:

- Assign each graph $g \in G$ to its closest cluster: For a given $g \notin M$, we assign g to the cluster with the closest centroid to g:

$$\arg\min\{dist(g, M_j) : g \notin M\}_{j=1}^{k} \tag{5.1}$$

  The distance between any two attack pattern graphs is given by computing the euclidean distance between their attack duration, interval rate, outgoing rate and their structural difference. For simplicity purposes as well as to minimise complexities, the structural difference between any two attack pattern graphs is given using the Jaccard similarity coefficient [134]. This metric measures the number of nodes and edges common in both graphs. As mentioned in Chapter 4, more accurate metrics such as Graph Edit Distance can also be used, however, the time required to compute the distance exponentially increases given a large set of vertexes and edges. Thus, there is often a trade-off between time and accuracy. Since the aim is to perform real-time analysis, we aim to minimise lagging in the analysis process as much as possible. Thus, the structural distance is defined as:

$$structDist(g_1, g_2) = 1 - \frac{g_1(pV, pE) \cap g_2(pV, pE)}{g_1(pV, pE) \cup g_2(pV, pE)} \tag{5.2}$$

- Re-compute the centroids: Typically, the centroid of a cluster is represented by computing the mean of the cluster [149]. While this is easy to compute for vector

data structures, the mean of a set of graph data structures is not straight forward to compute [149]. Let us assume a Graph Set of n graphs $= \{G_1, G_2, \ldots, G_n\}$ where $G_i = (V, E)$. The mean of the graph would be based on the vertices and edges in the graph set. Given that V and E are the set of vertices and edges represented by categorical data (not numerical), the mean cannot be computed.

A method proposed by Schenker et al. [149] is adopted whereby the mean graph of a Graph Set is successfully substituted for the median graph of the Set. This is more logical to compute. For a given cluster C of size m, we define the median graph, M as the centroid of the cluster which is the member in C with the minimum average distance to all other graphs in C:

$$M_i \leftarrow \arg\min \left\{ \frac{\sum_{j=0}^{m} dist(g_i, g_j)}{m} \right\}_{j=1}^{m} : g \in C \tag{5.3}$$

In the case where there is more than one median graph one is selected at random.

Task 2: We consider a situation where new attack pattern graphs may be received in real-time. For every new attack pattern graph, g' received in real-time which has not been clustered, the task is to update the cluster space C by allocating g' to a cluster.

We create an adaptation of the CluStream [6] and GMicro algorithms for clustering of graphs in real-time [3]. Given a new attack pattern graph, g' and a set of previously existing clusters $C = \{C_1, C_2, \ldots, C_k\}$, g' is either 1) absorbed by an existing cluster or 2) placed in a new cluster.

Firstly, we attempt to add g' to an existing cluster, $C_i \in C$. Similarly to the approach defined in Equation 5.1, we attempt to add g' to the cluster with the minimum distance between g and its centroid. We refer to this cluster as C_{min}. In certain cases, g may be unfit to be absorbed by C_{min} if despite C_{min} being the closest cluster to g', the members of g' are still highly dissimilar to g'. We define g' as being *unfit* for C_{min} if the distance

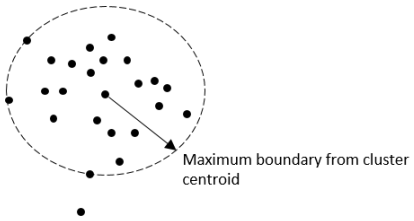

Figure 5.2: Maximum boundary of a cluster

between g' and C_{min} is greater than the maximum boundary value C_{min}. The maximum boundary of any cluster, C_i is computed as the root mean square deviation (rmsd) of all the graphs in C_i from its centroid M_i. Figure 5.2 illustrates this process intuitively.

$$maximum\_boundary(C_i) = \sqrt{\frac{\sum_{j=0}^{m} dist(M_i, g_j)^2}{m}} \tag{5.4}$$

We note that for a cluster with a single graph, the rmsd would be 0 thus, a heuristic is applied. Since C_{min} has no members, we look for the closest graph to C_{min} in the cluster space C. Let the closest graph to C_{min} be denoted as g_{min}. Note that g_{min} *cannot* be a member of C_{min}. Intuitively, given that g_{min} was at an unknown point unsuitable to be added to C_{min}, then g' should only be added to C_{min} if and only if 1) the distances between g' and C_{min} is less than the distance between C_{min} and g_{min} and 2) the nearest cluster is active (explained further below).

Given that g' cannot be added to a pre-existing cluster, then the following conclusions can be drawn:

- g' is a novel type of attack pattern whereby no other attack pattern similar to g' will be discovered. In this case, g' is just an outlier.
- g' is a novel type of attack pattern whereby other attack patterns similar g' may be discovered in future. In this case g' is the beginning of a new cluster in the stream of graphs.

In either scenario, since it will be unsuitable to add g' to an existing cluster, a new cluster is created. Similarly to [3], to minimise redundant clusters a 'maintenance' process is adopted by the algorithm. This involves: 1) De-activating in-active clusters and 2) Merging highly similar clusters.

De-activating inactive clusters

A cluster is de-activated if the time difference between its last added graph and the last received graph in the stream (in this case g)is less than a user defined threshold θ_t.

Merging clusters

If a cluster cannot be deactivated, an attempt to merge the two closest clusters is made. While merging the two closest clusters may be straight forward, this is generally a naive approach and may lead to high errors in the clusters. Let Figure 5.3(a) be considered where C_1 and C_2 are the two closest clusters. From visually inspecting the cluster space it can be concluded that C_1 and C_2 are not a natural cluster and should therefore not be merged. On the other hand, consider Figure 5.3(b) where C_1 and C_2 are still the closest clusters however the density of C_3 and C_4 is lower. In this case both C_1 and C_2 appear to be a natural cluster and a merge appears suitable. Note that the decision to merge the two clusters in either scenario can be drawn from whether the merge would significantly increase the square sum errors in the cluster space.

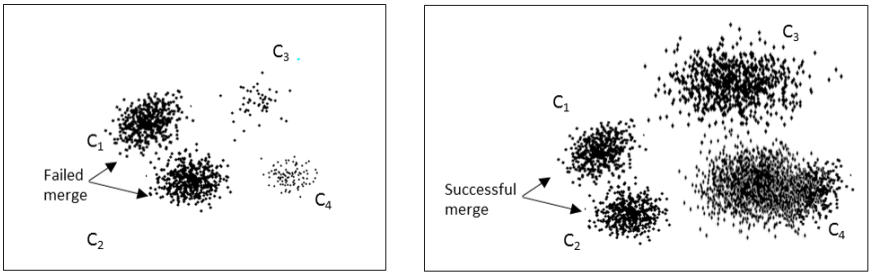

Figure 5.3: Suitable Merge Process

Therefore, to decide if two clusters C_i and C_j should be merged, we perform a temporary merge and call the new cluster C_{ij}, the centroid of the cluster is re-selected and the density of C_{ij} is computed as D_{ij}. The average density of the remaining clusters is also computed, $D_{\bar{ij}}$. If D_{ij} is less than $D_{\bar{ij}}$ clusters C_i and C_j will be merged.

If neither de-activating or merging can be performed, then no 'tidy-up' process is required. Thus, the number of clusters becomes $k+1$. It is noted that in an extreme case where many of the attack patterns are highly dissimilar, the number of clusters may increase significantly and at most equivalent to the number of attack pattern graphs received

on the stream. Thus, we propose a new parameter k_{max} which defines the maximum number of clusters allowed. If the number of clusters equals k_{max}, the merge procedure will be forced.

Extracting Novel Attack Pattern Graphs

For each cluster of attack patterns, the centroid is extracted as the most representative pattern of the cluster.

Time and Memory Complexities

We review the time and memory requirements needed to (1) Perform graph refining (2) Initialise the clustering process and (3) Dynamically update the cluster space in real-time upon receiving new graphs from the stream. For graph refining the time complexity required to refine a single graph is $O(|V|+|E|)$. At worst case, each attack pattern graph has the same number of nodes and edges. This requires an additional $O(|V|+|E|)$ memory storage. For the initialisation process, let I be at worst case, the number of iterations required before k-means++ converges during the initial task of initialising k clusters with n graphs. In addition, let d be at worst case, the largest graph size. Since the cluster space evolves over time, a value less than 10 for k is suitable. We tested the initialisation process with $k = 2$ and $k = 10$ in order to evaluate the time complexity where the average graph contained approximately 100 nodes and edges. Figure 5.4 shows that the time complexity for the initialisation process is linear. Following the initialisation process, each graph received in real-time is clustered in at worst case, $O(k \times (d))$ time.

In frequent pattern mining, the task of finding all frequent sequences in a huge database with m attributes may at worst case discover $O(m^f)$ frequent sequences of length f [111]. A well-known frequent graph mining algorithm, *gSpan* [176] has time complexity of $O(kfn+rf)$ where k is the number of occurrences of a frequent subgraph in a graph in the database, f is the number of frequent subgraphs, n is the number of graphs, and r is related to the internal computations of gspan [39]. Given this knowledge, we believe that the complexity of our approach is far less than frequent sequence or subgraph mining.

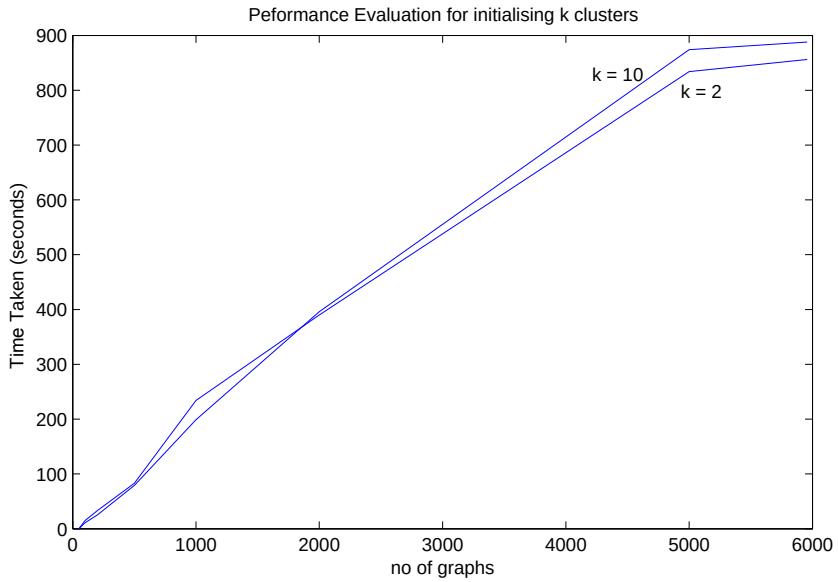

Figure 5.4: Complexity of Cluster Initialisation

5.3 Evaluation

The main objective of clustering attack pattern graphs in real-time was to discover novel attack patterns which would robustly describe specific types of attacks. Two alert datasets which capture DDoS, network infiltrations, worm propagation and network reconnaissance are analysed. We aim to be able to use real-time cluster analysis to show the categories of attacks occurring on the network as well as to show how the attacks vary over time.

5.3.1 Metrics

The correlation model described in Chapter 3 is used to transform a set of alerts into a set of meta-alerts represented as alert correlation graphs. Despite the alerts being historical events, the alerts are analysed in a streaming approach. In order to evaluate the quality of the clusters, the fundamental cluster separation and cohesion are measured. The definition for both metrics are provided.

- *Cluster Cohesion:* This measures the distance between the points in the cluster. Thus, a lower value indicates that items within the cluster are highly similar and that the cluster is tighter. For the purpose of the experimentations, the Cohesion for Cluster C_i is defined as the mean distance between the points in the cluster.

- *Cluster Separation:* This measures the distance between clusters. Thus a higher value indicates that clusters are highly dissimilar to other clusters. Hence, a high value is more desirable. In the experimentations, given a set of clusters $C = \{C_1, C_2, C_3, \ldots, C_n\}$, the cluster separation for C_i is defined as the mean distance between the points C_i and the points in every other cluster $\forall C_j \in C : C_j \neq C_i$.

In addition, the performance of the clustering process is measured. The datasets and their results are described next.

5.3.2 Environment

The clustering algorithm is implemented in Java and tested on a 64-bit Windows System with an Intel(R)Core i5 CPU processor with 2.40GHz, JVM 1.4.2 and 6GB for maximum heap memory.

5.3.3 Evaluation One: NG 2012 Internal Attack

160,000 Snort IDS alert logs were selected from the dataset described in Section 4.4.3. These alerts were collected over a 24 hour period from a computer network with 200 clients. The logs contain alerts which represent one major attack which occurred over 17 steps. In this experiment, these steps are aggregated into 5 main phases across the 24 hour period. These phases are described in Table 5.1. In addition, the logs contain alerts which represent network activity that was identified as malicious activity. Such activity included email content which appeared potentially malicious, abnormal host pinging and abnormal pinging of unreachable servers and ports. These activities repeatedly occurred over the 24 hour period.

Using the phases of the attack, we define 6 time periods where T_0 represents the time period before the main attack occurred, $T_1 T_4$ represents the time periods of the 5 phases of the main attack and T_5 represents the time after the main attack. It is to be noted that not all phases of the alerts are captured in the alert logs. Some alerts are missing for Time periods T_1 and T_{2b}.

The aim is to investigate if cluster analysis aids in distinguishing repetitive activity such as network host and port scanning from the main attack which consists of network

Table 5.1: Details of Network Activities and Remote Attack

T_i	Time Periods	Network and Attack Activity.
0	00:00:00 - 09:30:00	Normal network activity.
1	09:31:00 - 10:30:00	Quick and Intense Network scanning by attacker i.e. reconnaissance from the internal attacker to the main network.
2a	10:31:00 - 12:45:00	Failed exploit attempts on branch client workstations from internal attacker using shellcode embedded in email content.
2b	10:31:00 - 12:45:00	Successful exploit on internal vulnerable machine from internal attacker using shellcode embedded in email content.
3	12:46:00 - 13:35:00	Gain of full control and admin access of vulnerable machine from offsite attacker .
4	13:46:00 - 16:00:00	Offsite attacker uses vulnerable machine to explore internal network.
5	16:00:00 - 23:59:00	Normal network activit.y

infiltration and a DDoS attack. Furthermore, we aim to discover characteristics for defining correlated alerts which represent either type of attack.

First, we use the correlation model to generate alert correlation graphs. Each correlation graph groups alerts which are part of the same attack. From the 160,000 alerts collected from the logs, 146 alert correlation graphs were generated by the correlation model were the largest graph contained 84164 nodes and 84163 edges and the smallest graph contained 1 node and 0 edges.

Results and Discussion

The configuration for the Clustering algorithm are as follows: $k = 2$, $k_{max} = 20$, $init\_val = 10$ and $max\_iteration = 100$. Figure 5.5 shows the time duration of clustering each alert correlation graph. Initialising the clustering process with 10 alert correlation graphs took 260 secs. In comparison to the performance evaluation presented in Figure 5.4 which took 200 secs to analyse 500 graphs, this time duration is significantly higher. It was however noted that the sizes of the graphs in this scenario were significantly larger and given that the time complexity of the clustering process is dependent on the graph size, the time taken for the initialisation in this experiment was within reason.

In Table 5.2, the distribution of the alert correlation graphs over these time periods are shown. The results show that on average a cohesion and separation of 0.33 and 0.99 were achieved respectively. This indicates that the quality of the clusters derived was

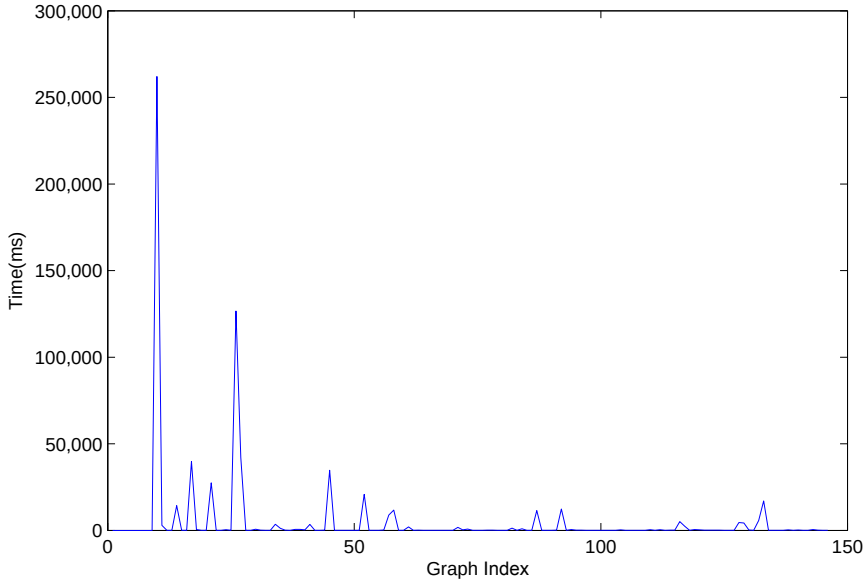

Figure 5.5: Time Taken During Real-time Cluster Analysis of NG Dataset.

good. The final state of the clusters which were derived at T_5 are detailed in Table 5.3. This includes all the graphs generated across T_0 to T_5.

Table 5.2: Summary of Clustering Results of Northrop Grumman 2012 Day 3 Dataset at various time periods

	T_0	T_1	T_2	T_3	T_4	T_5
no of clusters	20	20	20	20	20	20
no of graphs	43	60	87	91	114	143
seperation(mean)	0.97	0.99	0.99	0.99	0.99	0.99
cohersion(mean)	0.27	0.33	0.34	0.36	0.34	0.34

From Table 5.3, our investigation shows that larger clusters such as C_{12}, C_{19} and C_{20} contain graphs which represent the scanning of hosts, ports and graphs which represent suspicious mail content. These types of intrusions are the repetitive network attacks which were low priority and (in some cases) caused as a result of the network configuration.

We investigated smaller clusters as potential outliers or new evolving attack patterns. Investigation shows that the set of clusters namely - C_1 - C_6, C_9 - C_{10} and C_{13} & C_{14} contained fewer than 3 graphs. Inspecting some of the graphs contained in these clusters showed that approx. 80% of these graphs contained intrusion activity from the main attack. Figure 5.6 illustrates two alert correlation graphs found.

A more detailed inspection revealed that the attacker repeated the same attack against

Table 5.3: Detailed Clustering Results of Northrop Grumman 2012 Day 3 Dataset at Timeperiod T_5

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
Duration Rate										
low	50%	100%	100%	100%	100%	100%	50%	50%	100%	100%
medium	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
high	50%	0%	0%	0%	0%	0%	50%	50%	0%	0%
Interval Rate										
low	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
medium	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
high	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Outgoing Rate										
low	0%	0%	0%	50%	100%	0%	0%	0%	100%	100%
medium	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
high	100%	100%	100%	50%	0%	100%	100%	100%	0%	0%
Seperation	1	1	0.98	1	1	1	1	0.99	1	1
Cohesion	1	0	0	0	0	0	0.40	0.90	0	0
Cluster size	2	1	1	2	1	1	2	12	1	1

	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}	C_{19}	C_{20}
Duration Rate										
low	100%	79%	100%	100%	58%	0%	0%	100%	47%	32%
medium	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
high	0%	21%	0%	0%	42%	100%	100%	0%	53%	68%
Interval Rate										
low	100%	95%	100%	100%	100%	100%	100%	100%	100%	100%
medium	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
high	0%	5%	0%	0%	0%	0%	0%	0%	0%	0%
Outgoing Rate										
low	0%	14%	0%	100%	0%	0%	100%	0%	87%	91%
medium	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
high	100%	86%	100%	0%	100%	100%	0%	100%	13%	9%
Separation	0.99	0.97	1	1	0.99	0.99	1	0.99	1	0.96
Cohesion	0.27	0.52	0.10	0	0.66	0.58	0.61	0.37	0.72	0.72
Cluster Size	6	42	2	1	12	9	3	10	15	22

multiple targets. For example, Cluster C_{13} contained two graph members. Both graphs are highly similar i.e. both graphs depict the same attack steps, were triggered by the same intruder, however, targeted at different workstations. This indicates that an attacker is likely to repeat the same behaviour multiple times. Both graphs are depicted in Figure

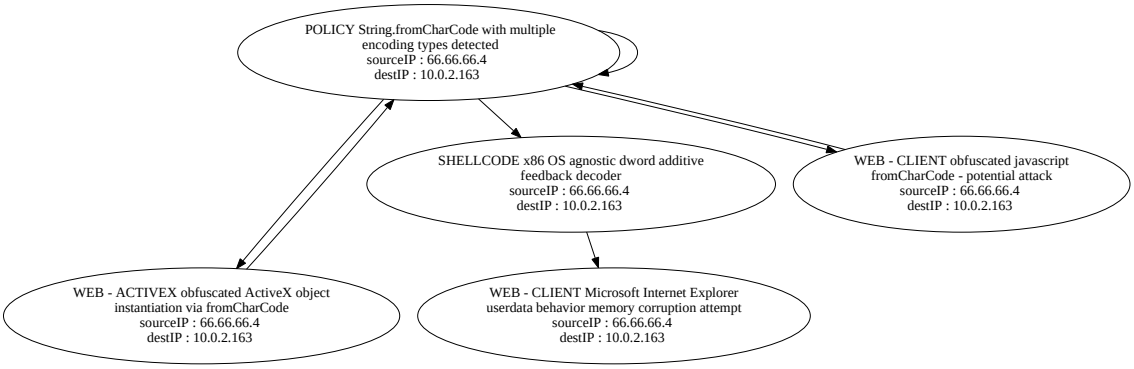

Figure 5.6: Main Attack from Northrop Grumman 2012 Dataset

5.7 and occurred within a short time period of each other.

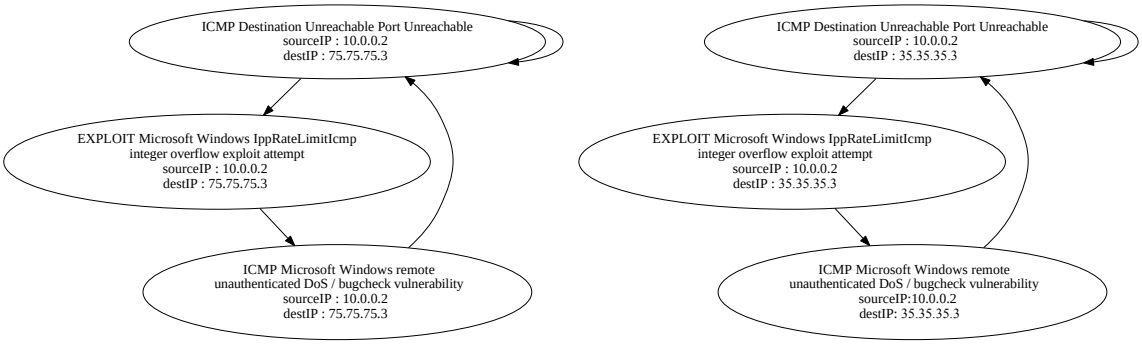

Figure 5.7: Similar Attack Patterns Discovered in NG Internal Attack Dataset.

5.3.4 Evaluation Two: VAST 2012 Mini-Challenge 2 Scenario

In this challenge a set of computers in the computer network of a regional office are infected with spyware and viruses. The computer network consists of 4000 workstations and 1000 servers. A Snort Intrusion Detection System (IDS) was used by the computer network and over a 2 day period of the challenge, 51073 intrusion alerts were triggered. As part of the challenge the task was to identify any noteworthy events, and security trends which took place for the time period covered in the IDS logs. A full description of the datasets and attack scenario is available from the Visual Analytics Community [1].

The alert correlation model described in [157] was used to aggregate all alerts into alert correlation graphs. 842 alert correlation graphs were generated. It was observed that the largest graph contained 20816 nodes and 20815 edges while the smallest graph contained 1 node and 0 edges.

[1] Visual Analytics Community Website : http://www.vacommunity.org/

Results and Discussion

The configurations for the real-time clustering approach were as follows: $k = 2$, $k_{max} = 20$, $init\_value = 10$, and $max\_iterations = 100$. The toolkit was deployed on the same machine as described in Section 5.3.3.

The alerts (and alert correlation graphs) are analysed in a streaming approach. Figure 5.8 shows the time performance for clustering each alert correlation graph. The spike in Figure 5.8 at the 10th graph index indicates that the initialisation of the clustering process was completed after 2.72s. Thereafter, the mean time for clustering a graph was 31.44ms.

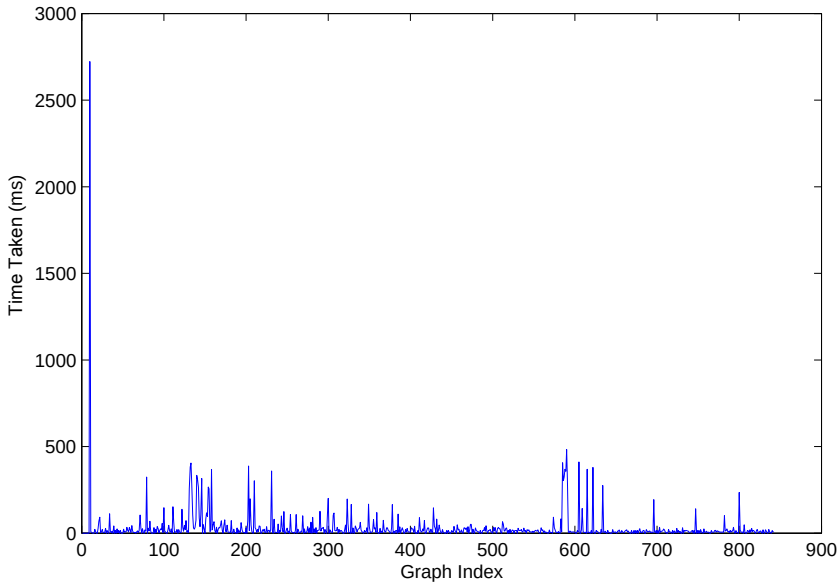

Figure 5.8: Time Taken During Real-time Cluster Analysis of VAST 2012 Dataset.

In order to evaluate the progression of the clusters over time, we take snapshots of the clusters' states approximately every 3 hours over the 39 hour time period. This resulted in 13 time periods. Table 5.4 shows a summary of the cluster over these time periods.

The results in Table 5.4 show that since only a single graph is established during the 1st timeperiod, 0 clusters are created. However, in the subsequent timeperiods T_1, T_2 102 additional alert correlation graphs had been generated. In addition, these graphs had been clustered into k_{max} clusters. Thereafter, k_{max} clusters are maintained. Despite this, a high cluster separation and low cluster cohesion is still achieved indicating good cluster quality.

Table 5.4: Summary of Clustering Results of VAST MC 2012 Dataset at various time periods

	no of clusters	no of graphs clustered	seperation (mean)	Cohersion (mean)
T_0	0	1	-	-
T_1	11	40	0.83	0.2
T_2	20	103	0.81	0.17
T_3	20	212	0.87	0.2
T_4	20	286	0.87	0.2
T_5	20	354	0.87	0.21
T_6	20	423	0.87	0.21
T_7	20	497	0.87	0.21
T_8	20	575	0.87	0.17
T_9	20	652	0.87	0.17
T_{10}	20	703	0.88	0.17
T_{11}	20	787	0.88	0.17
T_{12}	20	799	0.88	0.17
T_{13}	20	842	0.88	0.16

Table 5.5 shows the details of the clusters after all 842 alert correlation graphs had been analysed at the final time period T_{13}. In our investigation, 4 clusters stand out. These are clusters C_5 - C_8 each Containing 2, 2, 1, and 1 graphs respectively. Based on our hypothesis in Section 5.2.2, a cluster containing a single item is described as either 1) an outlier cluster 2) a new evolving cluster. In the case of these clusters, each cluster had existed from as early as time period T_3. This indicates that from time period T_3 from T_{13}, which is a 30 hour time interval, no similar graphs were added to these clusters. It can therefore be concluded that these clusters and the graphs they contain, are highly likely outliers.

Two of the clusters C_5 and C_6 are investigated further. Figure 5.9 and Figure 5.10 shows their centroids (i.e. median graphs). The duration rate of these graphs is 0.81 and 1, both have an interval rate of 0 and an outgoing rate of 1 respectively. These properties are indicative of an internal attack (measured by the outgoing rate) which is carried out over a long period of time. These properties are potentially properties of the spyware. However, while both graphs have similar structure and attack properties, each graph represents the intrusion activity targeted at different workstations. This potentially explains their placement in separate clusters.

In Section 5.1, the primary hypothesis is that multiple sequences of correlated alerts (i.e. alert correlation graphs) which are instances of the same type of attack will likely

Table 5.5: Detailed Clustering Results of MC2 VAST Challenge 2012 Dataset at Timeperiod T_{13}

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
Duration Rate										
Low	0%	100%	72%	51%	0%	0%	0%	0%	40%	4%
Medium	0%	0%	15%	8%	100%	0%	100%	100%	60%	0%
High	100%	0%	13%	41%	0%	100%	0%	0%	0%	96%
Interval Rate										
Low	100%	100%	72%	92%	100%	100%	100%	100%	100%	100%
Medium	0%	0%	14%	5%	0%	0%	0%	0%	0%	0%
High	0%	0%	13%	3%	0%	0%	0%	0%	0%	0%
Outgoing Rate										
Low	0%	100%	100%	100%	0%	0%	0%	0%	0%	100%
Medium	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
High	100%	0%	0%	0%	100%	100%	100%	100%	100%	0%
Cohersion	0.86	0	0.01	0	0.60	0	0	0	0.16	0.16
Seperation	1	0.87	0.74	0.70	1	1	0.99	0.99	1	0.89
Cluser Size	5	228	127	76	2	2	1	1	5	24

	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}	C_{19}	C_{20}
Duration										
Low	0%	0%	0%	26%	0%	0%	24%	0%	0%	5%
Medium	0%	0%	0%	6%	0%	0%	3%	0%	4%	2%
High	100%	100%	100%	68%	100%	100%	73%	100%	96%	93%
Interval Rate										
Low	100%	100%	100%	100%	100%	100%	97.09%	100%	100%	100%
Medim	0%	0%	0%	0%	0%	0%	2.91%	0%	0%	0%
High	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Outgoing Rate										
Low	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Medium	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
High	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Cohersion	0.16	0.13	0.17	0	0.15	0.10	0.22	0.10	0.15	0.15
Seperation	0.84	0.84	0.92	0.66	0.96	0.90	0.73	0.81	0.79	0.75
Cluster size	7	35	11	19	25	18	103	43	51	59

contain similar characteristics or features. The similarities between Figure 5.9 and Figure 5.10 validate this. Figure 5.11 also shows an attack pattern representation of Figure 5.9. The abstract representation of this graph shares similar characteristics with Figure 5.10 as well as other graphs in the graph set. Such an attack pattern can be used to characterise a type of internal attack.

5.4 Conclusion

In this Chapter, a clustering algorithm for categorising multi-attributed direct graphs such as alert correlation graphs is proposed. The algorithm, which is an adaptation of [3] has been tested with recent datasets and our results showed that it discovers naturally

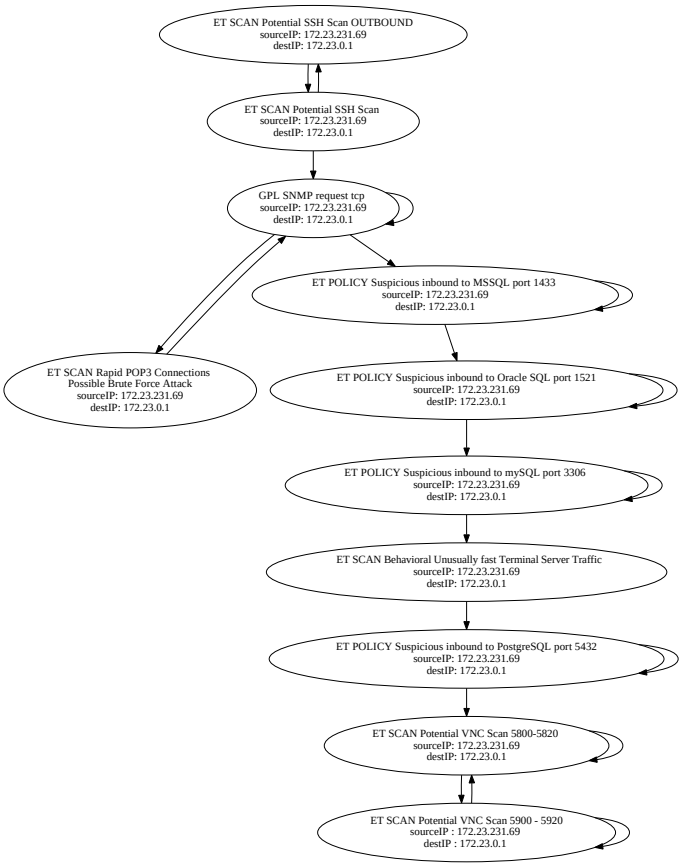

Figure 5.9: Centroid of C_5

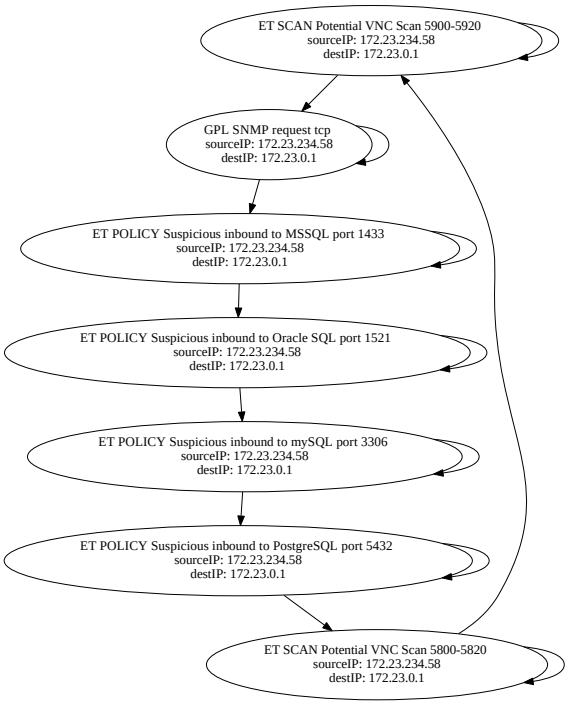

Figure 5.10: Centroid of C_6

evolving attack patterns, produces high quality clusters and at the same time performances is memory and time efficient. For future directions of this research, we believe that with little user validations, the discovered attack patterns can be used to mitigate future attacks by correlation network events to the steps in the attack pattern with the intentions of predicting an attackers next steps.

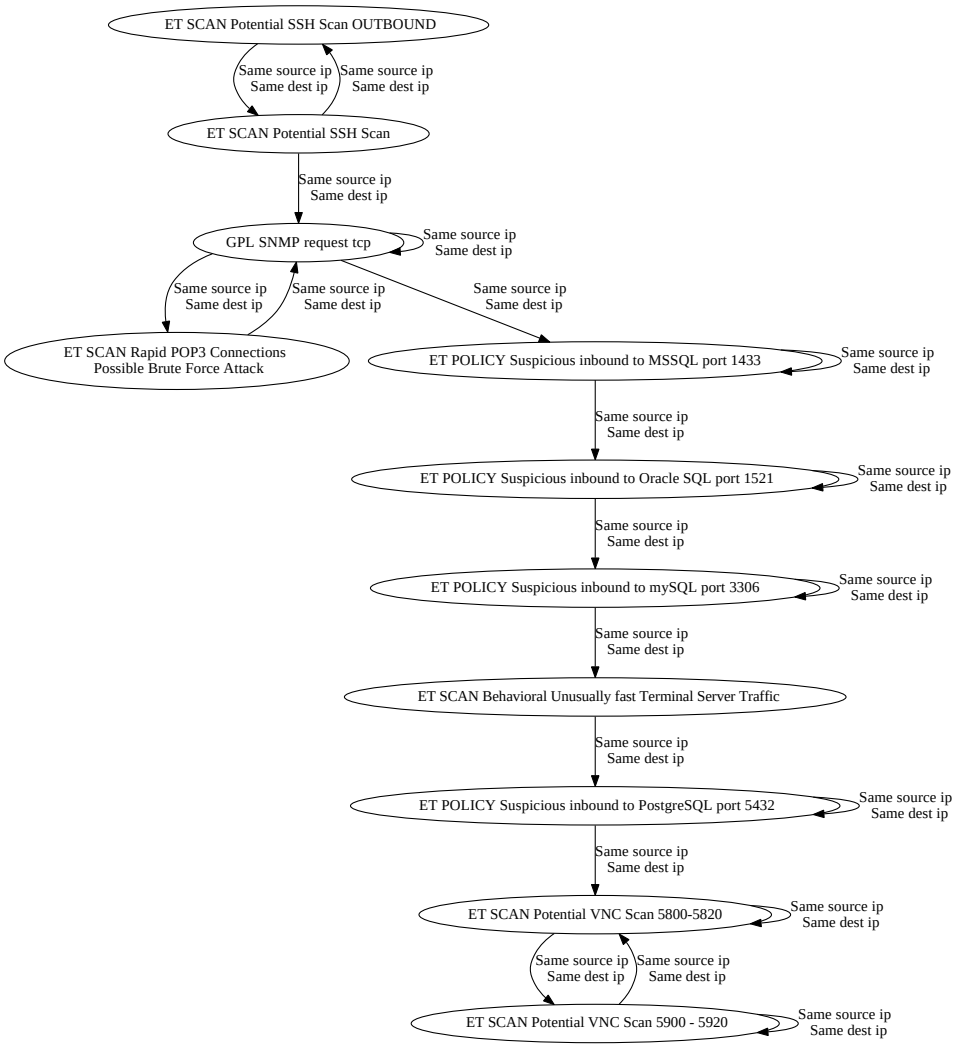

Figure 5.11: Attack Pattern Graph representaiton of Figure 5.9

Chapter 6

Visual Analytics for Investigating Attack Behaviour and Pattern Graphs

6.1 Introduction

Fully automated algorithms alone are often not capable of providing actionable insights about threat landscapes. [60]. Fischer et al. [60] further described that this is due to the complex results outputted by automated approaches. Keim et al. [80] also identified that certain domain problems which require data analysis are complex problems which an automated approach cannot fully solve. These problems such as understanding the scope and dynamics of a threat through data analysis require a combination of human analytical capabilities as well as automated approaches.

Goodall [64] also asserted that while automated approaches are a potential solution to combating threats, these approaches undervalue the strong analytical capabilities of humans. This re-instates the importance of including human analytical capabilities in the process of analysing security data.

Earlier on in Section 2.6.5 of Chapter 2, we introduced Visual Analytics as *the science of analytical reasoning facilitated by interactive visual interfaces* [165]. Visual Analytics allows the inclusion of human capabilities at various stages of the process of data analysis. Such capabilities are used to validate the results, discover knowledge unidentifiable by automated approaches, and to make actionable decisions based on the results.

Keim et al. [80] proposed a guide to the process of effectively including human intelligence during data analysis. This process, identified as the information seeking mantra includes four steps: *1) Analyse first, 2) Show the important, 3) zoom, filter and analyse further, and 4) Details on demand.*

Therefore, while the algorithmic approaches proposed in Chapters 3 to 5 effectively analyse and identify relevant features for discovering and measuring the complexity of an attack, each of these approaches require a security analyst to analyse further. Take for instance, Chapter 3 can be mapped to the first step in the information seeking mantra - *"Analyse First"*. In Chapter 3, alert correlation was used to achieve data aggregation whereby groups of aggregated alerts corresponded to the same attack. While this knowlege is highly useful a security analyst's input is still required to evaluate the results, identify the important alert groups and act on the results. In Chapter 4, the process of *"showing the important"* was automated by using a prioritisation metric to show only alert groups which may be relevant to an attack of interest such as a denial of service, or worm outbreak. Despite this, a security analyst's capabilities are still required to validate the prioritised results. This in other words corresponds to Step 3 - *"Analyse Further"* for the purpose of validating the results and invoking actions such as threat mitigation.

In this Chapter, a set of visualisation capabilities are proposed and developed for analysing security alert data. In addition, a set of visualisations are investigated for effectively visualising the outputs from Chapter 3 to Chapter 5. Shiravi et al. [154] identified five applications for visualising security data - *host/server monitoring, internal/external monitoring, port activity monitoring, routing behaviour monitoring, and attack pattern analysis*. The proposed and developed visualisation techniques are for achieving host/server monitoring as well as attack pattern analysis. The proposed techniques include of a time series and correlation graph visualisation. These visualisations are described by [103] as methods for effectively analysing historical security alert data. We also explore how the proposed methods can be used in real-time analysis by integrating the developed techniques into a pre-existing commercial visual analytics software system.

The rest of this Chapter is organised as follows: Section 6.2 describes a dataset

collected from a network attack experiment. The goal is to evaluate how *effectively* the proposed visualisations aid in presenting attack information to a security analyst. Section 6.3 details a set of visualisation methods which we proposed for effectively achieving *"host and server monitoring"* as well as discovering and understanding attack patterns. We develop three visualisations for security analysis. In Section 6.3.1, a time series visualisation method commonly referred to as the *Streamgraph (ThemeRiver)* is developed and applied to analysing security alert data. In particular, this visualisation is applied in order to analyse host and server activity. In Section 6.3.2, a radial correlation graph visualisation for identifying correlations between alert attributes is proposed. Finally, in Section 6.3.3 a correlation graph visualisation is proposed for visualising the alert correlation graphs produced in Chapters 3 - 5. Section 6.4 describes how these visual methods were integrated into **SATURN**, a commercial visual analysis tool developed by the Security Futures Practice, Research & Innovation, British Telecom. Finally, we conclude this Chapter in Section 6.5.

6.2 Datasets

To evaluate how effective the proposed visual methods speed up the process of attack detection, two datasets are used to evaluate the developed visualisation methods. The first dataset is collected from a private network while the second is a publicly available dataset. A description of the datasets is given next.

6.2.1 Private Network Alert Data

Due to the sensitivity of this dataset, there is a high anonymity in its description. However, it can be said that the intrusion alert logs were captured over a five day period on a network which had been infiltrated by a worm outbreak virus from 21st November to 27th November 2010. Through studying the dataset it was discovered that the self replicating and propagating worm was successfully propagating across the network. In total, 46,911 intrusion alerts were triggered. Additional network intrusion activity such as attempts to exploit web vulnerabilities also occurred simultaneously. For anonymity reasons, the network topology cannot be detailed.

6.2.2 USMA CDX2009

The USMA Cyber Defence Exercise (CDX) was held in April 2009 by teams from the U.S. Military Academy (USMA) and other military colleges. The exercise consisted of multiple connected networks, a red team and a set of defence teams. Each defence team was responsible for a network which was configured to run consistent network services including a web application. In addition, each network was required to integrate three untrusted workstations. The networks were then attacked by a remote red team. In this study, we evaluate the network of the USMA team with respect to the attacks from the red team. In the USMA network, a Snort IDS is placed at the perimeter of the network. In total, 25,741 alerts were triggered.

6.3 Proposed Security Visual Analytics

6.3.1 Time Series Analysis using Streamgraphs

Streamgraphs are time series graphs which are highly related to the conventional stack graphs [38]. Byron and Wattenberg [38] proposed that this type of complex layered graph is effective for displaying large datasets. It solves the problem where the distributions and occurrences of a large range of entities need to be observed across the same time scale on the x-axis. Its first application was for visualising trends in music by representing the music listening behaviour of a user over time (Shown in Figure 6.1). As shown in Figure 6.1 the x-axis intersects with the centre of the graph's y-axis and *stripes* are stacked symmetrically on both sides of the graph. Each stripe represents the number of times a user listened to a particular artist during a given time period.

While Streamgraphs are commonly used to visualise Open Source Intelligence and Social Media data, streamgraphs can be effectively used for gaining attack insight from security alerts. A common scenario where streamgraphs can be applied is in monitoring the communication history of a single computer which has frequent communication with a large number of hosts internal to the network as well as external.

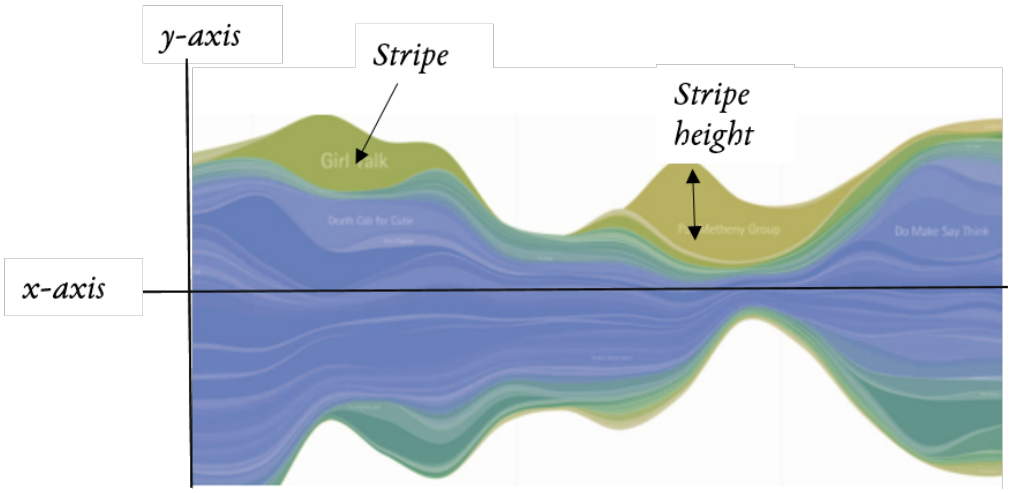

Figure 6.1: Application of streamgraph for visualisation trends in user's music listening behaviour [38]

USMA CDX2009 Alert Analysis using Streamgraph

Figure 6.2 shows a streamgraph which was developed in this study for security alert visualisation. We use this streamgraph to analyse the events that occurred during CDX2009. The streamgraph in Figure 6.2 shows the distribution of IDS alerts triggered on the first day of the attack. In this Figure, each stripe represents a host's alert activity while the vertical height of the stripe at a given time point, T_x, along the x-axis is a visual representation of the volume of alerts that occurred at T_x. Two 1-hour spikes can immediately be observed in Figure 6.2. The streamgraph shows that both spikes represent alerts targeted at IP address of the the network's Apache Web Server.

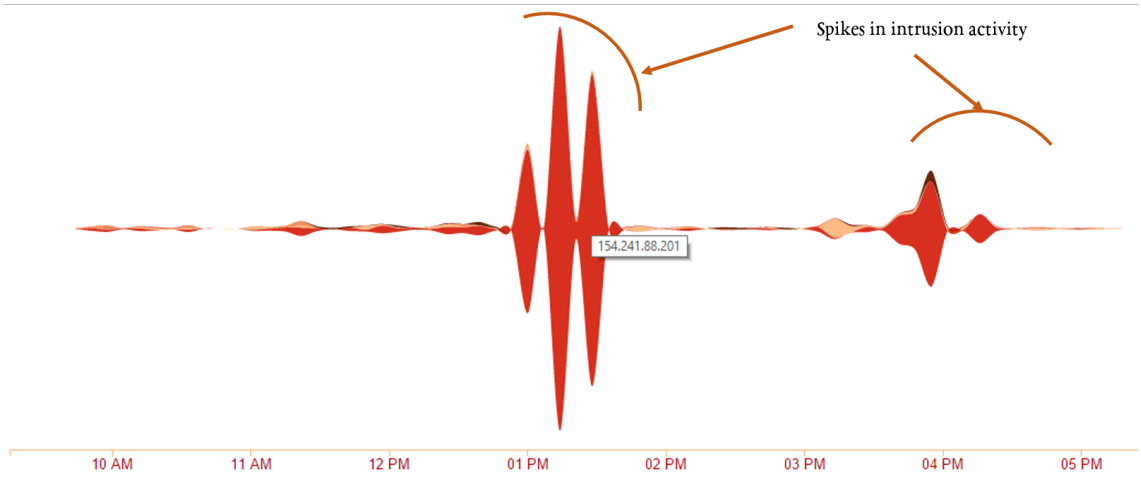

Figure 6.2: Streamgraph Visualising USMA CDX 2009 Dataset - x-axis = Time, y-axis = target IP address

In order to evaluate further, the streamgraph is used to show the distribution of alert types targeted at the Apache Web Server. This is shown in Figure 6.4. In this Figure, each stripe on the graph represents an intrusion type and the width of the region at a point in time represents the volume of an alert type targeted at the Apache Web Server. From Figure 6.4, it can be observed that there is a 15 minute exploit attempt by trying to run malicious executables on the web server. This is followed by intense scanning of the web server (Reconnaissance activity) followed by more exploit attempts.This was potentially an attempt to compromise the Web server.

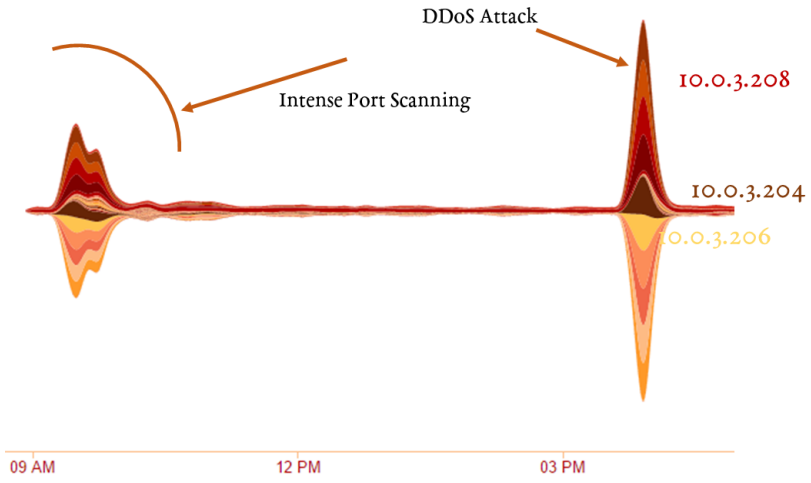

Figure 6.3: Streamgraph Visualising NG 2012 Internal Attack Dataset - x-axis = Time, y-axis = targeted IP address

Figure 6.4: Streamgraph Visualising USMA CDX 2009 Dataset - x-axis = Time, y-axis = Intrusion Type targeted at the Apache Web Server.

NGINT 2012 Alert Analysis using Streamgraph

This attack scenario consists of a full days attack experiment starting with reconnaissance network activity by an external attacker followed by stealth attacks and concluding with a denial of service launched from an external attacker on the defending network. A full description of the attack is provided in Section 4.3.1 of Chapter 4. Figure 6.3 shows a streamgraph of the 261,916 alerts triggered from 9.00 - 18.00. Similarly to Figure 6.2, each stripe in the graph represents the number of alerts targeted at a specific host on the network. In this Figure, two spikes which depict the increase in the intrusions targeted at a set of network hosts at two time periods can be observed. When both time periods are cross referenced with respect to the attack documentations, each spike represents intense port scanning activity and a denial of service attack respectively. A snippet of these attacks are shown in Figure 6.1 for a detailed description of the attack, see Table 4.1 of Chapter 4.

Table 6.1: Snippet of NG 2012 Internal Attack Description showing Intense Scanning and DDoS activity

Steps	NG Internal Attack Activity (NGINT)
1	Backtrack started on Branch client 10.1.0.201 (insider)
2	Quick scan from 10.1.0.201 of local subnet
3	Intense scan from 10.1.0.201 of local subnet
4	Quick scan from 10.1.0.201 of 10.0.0.0/24
5	Quick scan from 10.1.0.201 of 10.0.2.0/24
...	.....
...	.....
...	.....
15	Several short DOS attacks from Attack2 against 4.4.4.2.
16	DDOS from traffic generator against 4.4.4.2
17	DDOS from traffic generator against 4.4.4.2

Evaluating Streamgraph Alert Analysis

Aigner et al. [9] described that a streamgraph, which they refer to as a ThemeRiver visualisation provides an overview of what is important in a data at a given time. This can be re-affirmed since in both case studies described above, the streamgraph immediately provides an overview of the attack exercise allowing an observer to immediately isolate major attacks such as Denial-of-Service, intense port scanning and brute force web exploits.

Limitations

The ability to draw effective conclusions using this visual method is dependant on the volume of data, the layout of the streamgraph and visual attributes such as the coloring of the graph's categories. It was observed that the streamgraph is more suitable when the number of categories along the y-axis are limited to a small or medium set. In some scenarios, it may be more suitable for visualising intrusion categories but less suitable for visualising a large set of Source IP addresses. The ordering of the graphs along the y-axis is also crucial. In our case, the graph is ordered *inside out* i.e. by placing the stipes with the highest volume on the outer in the middle of the graph and the stripes with the least volume on the outer regions of the graph. Thus, attention is drawn to the frequent elements at the centre of the graph. On the other hand smaller elements may be difficult to observe. To illustrate this issue, Figure 6.5 is a filtered version of Figure 6.2 which only shows the intrusion activity on the network after filtering the activity related to the Apache Web Server. Note that due to the volume of traffic targeted at the Web Server, the intrusion activity targeted at other hosts cannot be easily seen in Figure 6.2. In the filtered graph however, periodic trends involving the intrusion activity of other targeted hosts can be observed. In practice, the application of the streamgraph in security visualisation was only suitable for overviews and not for advanced attack analysis. To achieve this, a set of visual methods, referred to as correlation graphs in work by Marty [103] are designed and applied.

6.3.2 Attack Analysis using Radial Graphs

A Radial Visualisation is the representation of data in an elliptical circular pattern [53]. In addition, radial visualisations are suitable for visualising relationships (and therefore, correlations), amongst disparate entities [53]. Typically, a traditional correlation graph such as the graph shown in Figure 6.6 is a matrix diagram which nests multiple scatterplots. Each scatterplot shows the extent to which two continuous data dimensions are correlated [103]. For example in Figure 6.6, it is shown that *Employee Hours* and *Cost* are positively correlated while *Incidents* and *Cost* are negatively correlation. Note however, that these

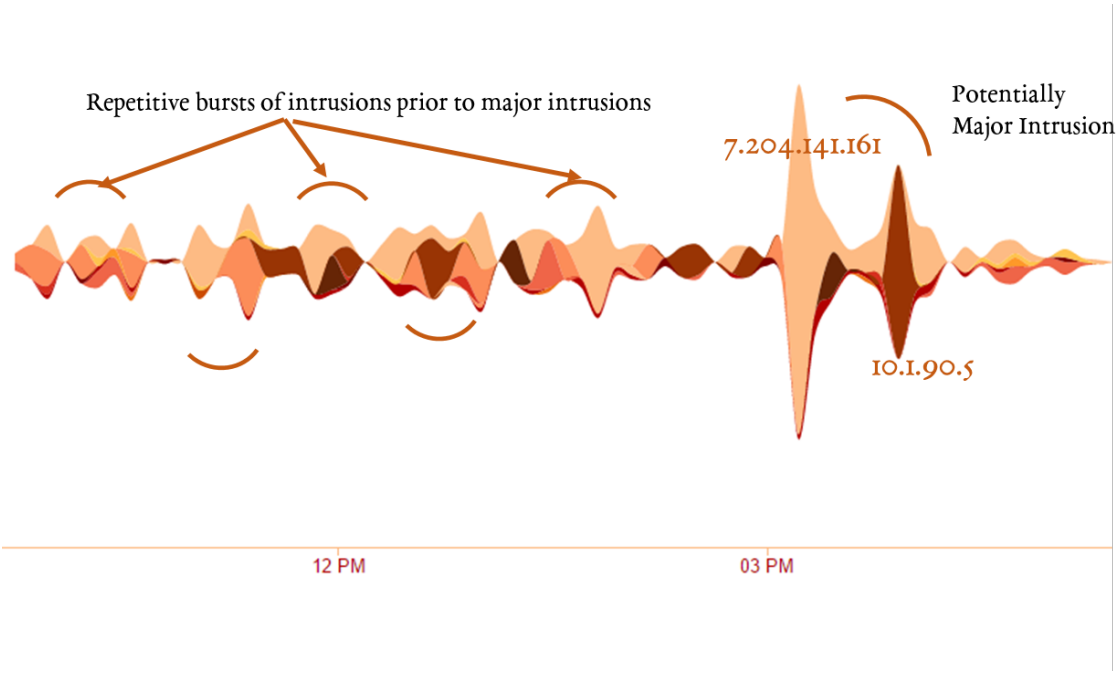

Figure 6.5: Filtered Streamgraph of Figure 6.2 showing USMA CDX 2009 Day 1 Attack

fields are numerical and therefore measurable. Since most security alert fields are nominal, such traditional diagrams are not useful in this domain. Marty [103] identified that in security alert analysis, there are two ways to use a correlation graph for analysis. In the first approach, two data dimensions of the same security alert data log are correlated with each other. In the second approach, the same data dimension of multiple security alert logs are correlated. In this section two radial graphs are proposed for achieving the first type of correlation.

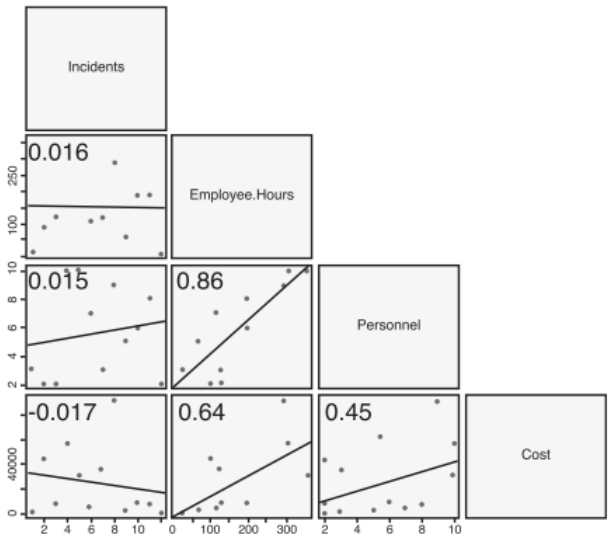

Figure 6.6: A traditional correlation matrix showing the relationship among multiple numerical data dimensions [103]

Radial Communication Graph

The designed radial communication graph is a network visualisation which consists of nodes and edges. Each node represents a value from the chosen dimension and an edge represents a communication link between the two nodes. This type of graph is particularly suitable for visualising IP Address communication on the network. Figure 6.7 shows the radial communication graph that was designed. In this graph, all IP addresses from a security alert log (i.e. all source IP addresses and target IP addresses) are collected and aligned as nodes in a circular pattern. Each edge represents one or more alerts which show that an alert from IP_a triggered an alert targeted at IP_b. The thickness of an edge represents the number of alert instances between two IP addresses.

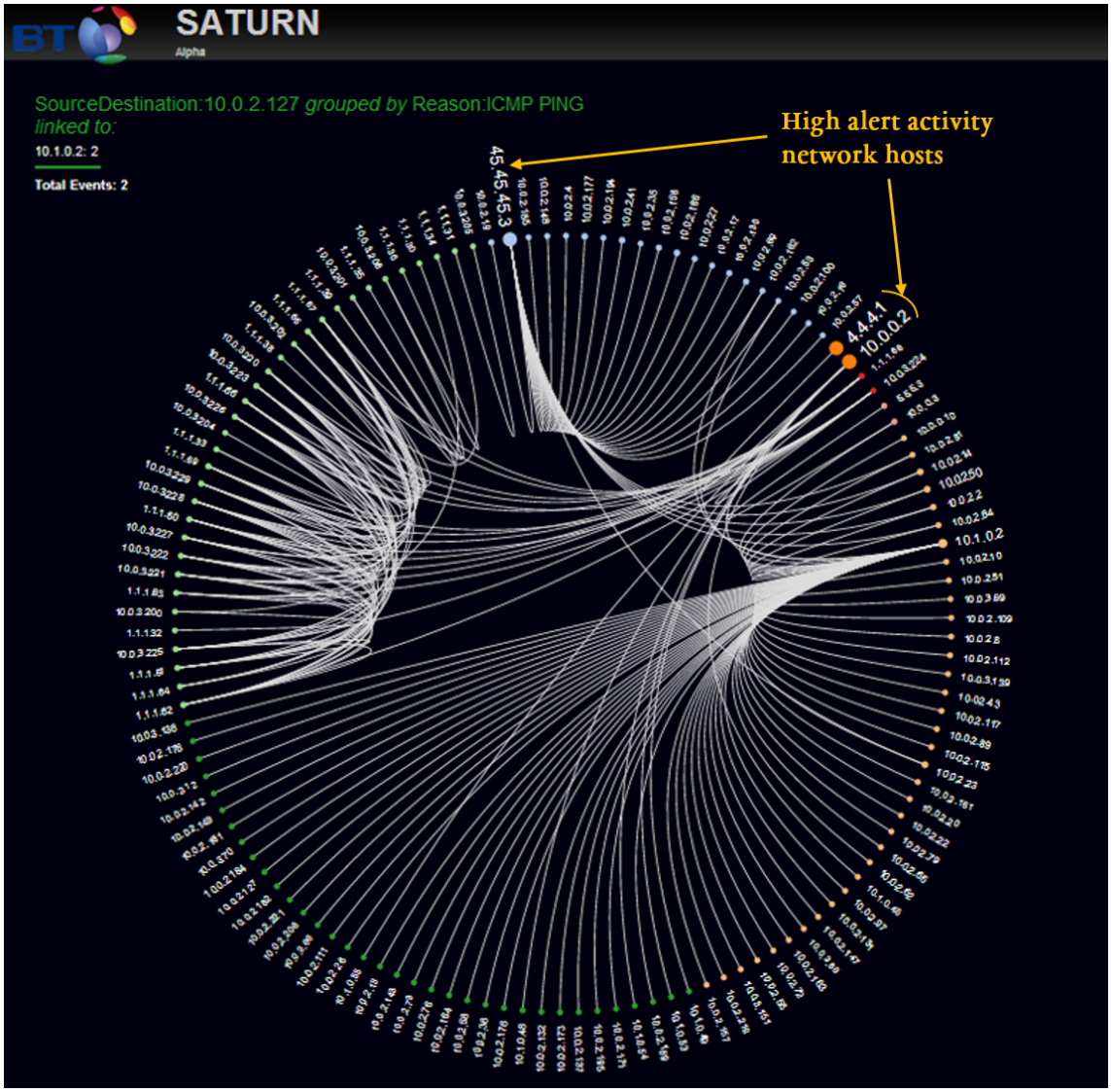

Figure 6.7: Radial Communication Graph proposed during this research study.

The proposed radial communication graph can also be used to visualise hierarchical information such that nodes are ordered according to various groupings (Shown in Figure 6.7 where nodes represent IP addresses and are grouped according to their logical locations).

2D Radial Coordinates Graph

The first graph is extended to visualise other arbitrary dimensions from security alert logs. For example, it was observed that the graph proposed in Figure 6.7 is only suitable for IP addresses. The radial coordinates graph is an adaptation of the traditional parallel coordinates which is a common form of visualising high dimensional datasets. In the parallel coordinates, n dimensions are represented by n lines aligned in parallel. Each point along a single line represents an attribute of the dimension and a line L_{a_i,b_j} between two points represents an entry in the dataset which has value i for dimension a and value j for dimension b. Figure 6.8 shows a traditional parallel coordinates visualisation.

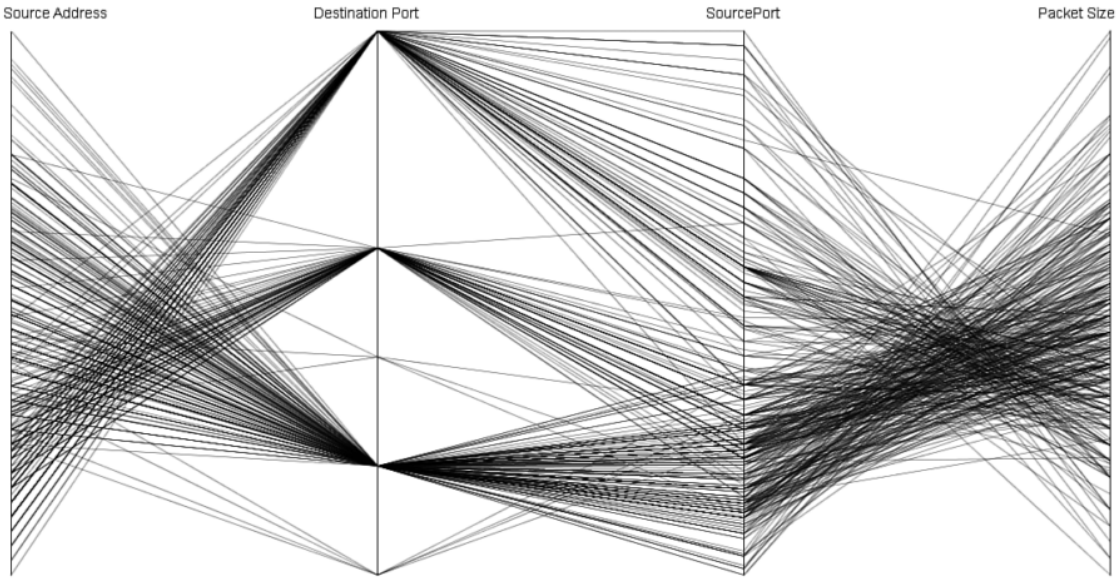

Figure 6.8: A Simple Parallel Coordinates [103]

Figure 6.9 shows the components of the proposed radial coordinates visualisation. Unlike the parallel coordinates, the radial coordinates is limited to visualising two dimensions and each dimension is ordered along the circumference of the radial view. While this means fewer dimensions are visualised, the radial visualisation avoids *information overload* - a common problem in visual analytics [80]. For example, a dimension in a

large dataset such as a security alert log may contain a large number of attributes (for example, IP Addresses). this may result to *"over plotting"* in the parallel coordinates causing the visualisation to become cluttered and complex to interpret [76].

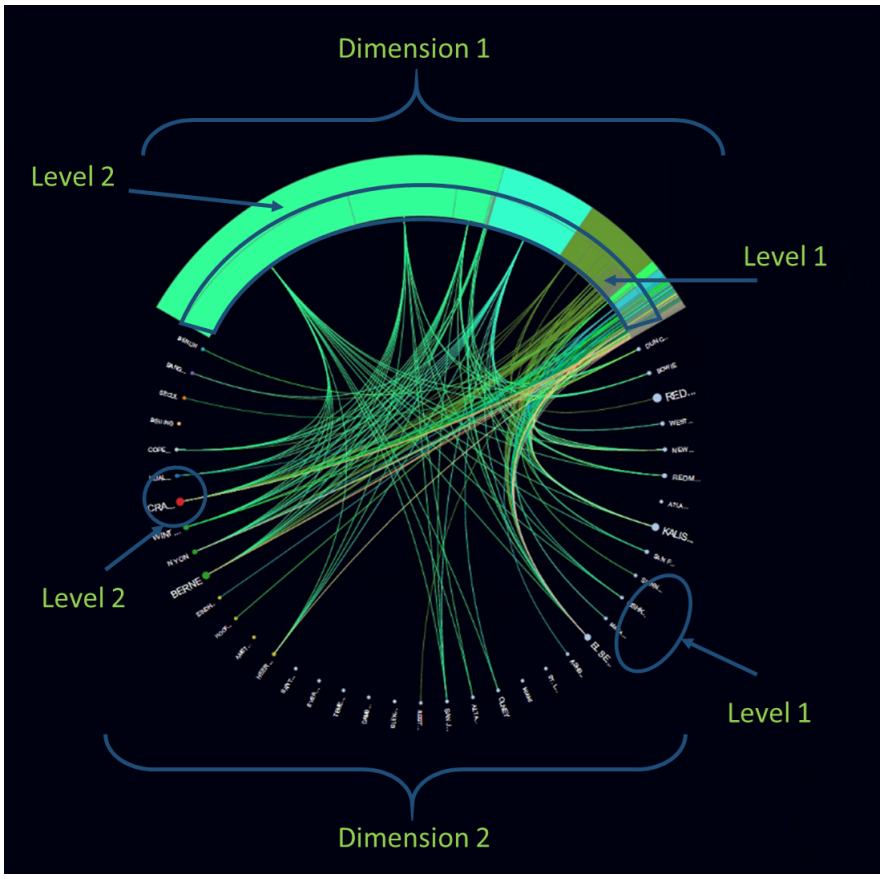

Figure 6.9: Components of the 2D Radial Coordinates proposed during this research study

In the radial coordinates, each dimension consists of two levels and each level represents the attributes of the dimension. The attributes of the first dimension are represented as *arcs* while the attributes of the second are represented as *circular nodes*. *Level One* represents the main dimension to be observed while *Level Two* is another dimension in the dataset which allows a grouping and ordering of the attributes in the first dimension. For example, this is useful when IP addresses are represented on the first level of dimension one. Rather than a random ordering of IP addresses, dimension two can be used to order level one according to their physical or logical locations (if this dimension/information is available in the dataset). Another example is using the *intrusion category* to order *intrusion types*. In other words, the second level of each dimension allows hierarchical

structuring. To add clarity to the visualisation, nodes are further coloured according to their groupings. An example of the components of dimension one are shown in detail in Figure 6.10.

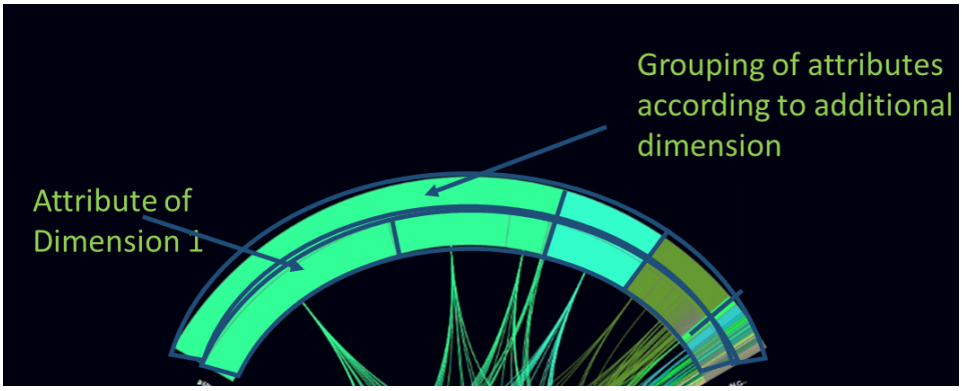

Figure 6.10: Hierarchical Structuring of alert attributes using 2D Radial Coordinates Graph

Private Network Alert Analysis using Radial Coordinates

In Figure 6.11, the radial coordinates shows the links between the IPs of potential attackers (i.e. intrusion source IPs) and victims on the private network (target IPs). The source of the intrusions are represented on dimension two where level one represents the source IP addresses and level two orders the IP addresses according to their geographical location. Similarly, the targets of the intended intrusions is represented on dimension two with IP addresses on level one and geographical locations on level two. As shown in Figure 6.11, the interactive capabilities integrated into the visualisation supports the inspection of each attribute in each dimension by fading out non-focussed elements. In Figure 6.11 it is seen that a large number of intrusion are targeted at IP addresses within the US. Highlighting this attribute in the visualisation enables the user to inspect the source IP addresses which triggered the intrusions. Also, Figure 6.11 shows that most of the intrusions targeted at IPs based in the US were triggered by hosts machines situated in Seoul and Beijing.

Figure 6.12 shows a radial coordinates plot of the same dataset with a different dimension selected for dimension two. In this Figure, dimension two is represented by the intrusion alert type, however, no grouping is available in the dataset. Figure 6.12 combined with Figure 6.11 shows that there is a strong correlation between intrusions

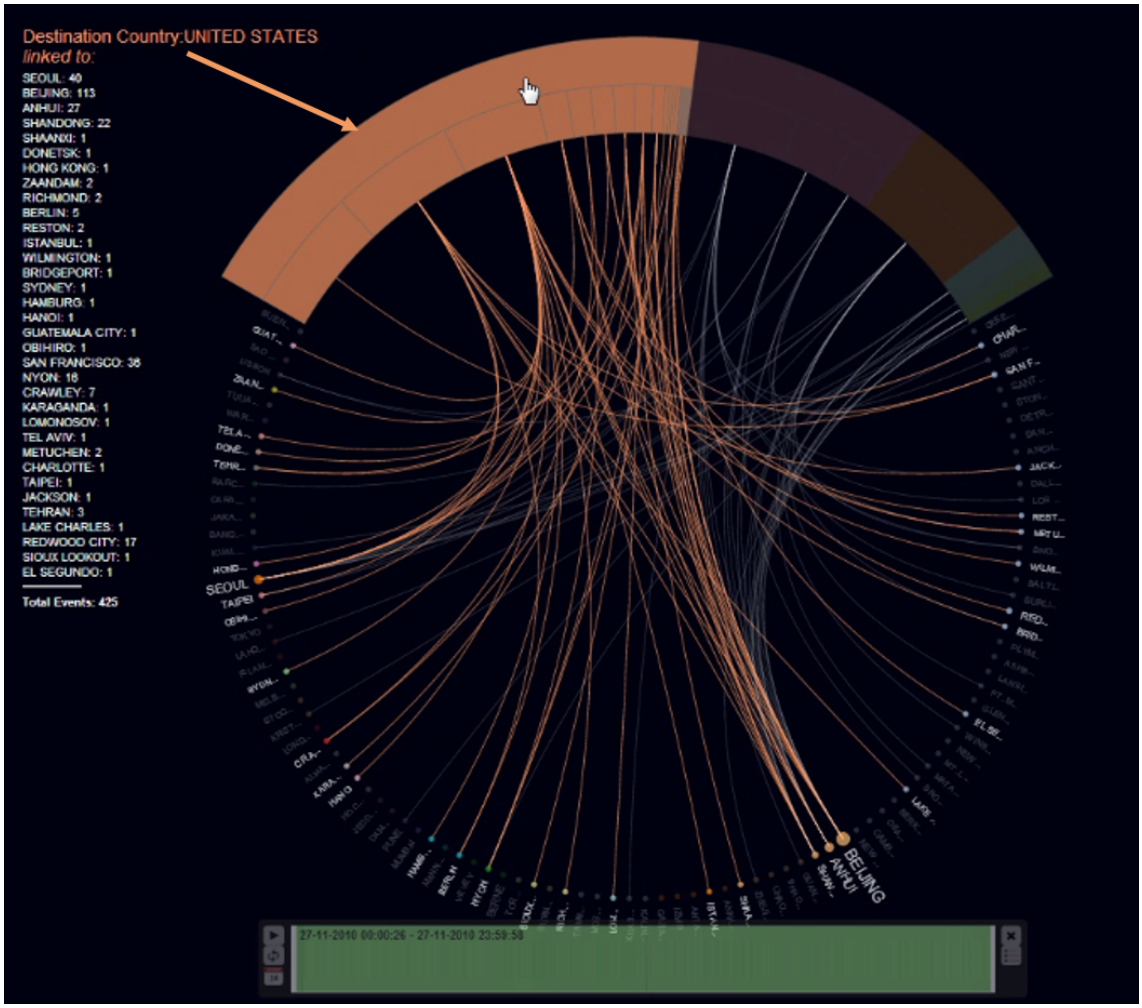

Figure 6.11: 2D Radial Coordinates showing Private Network Alert Capture (Dimension One = Alert Target, Dimension Two = Alert Source).

targeted at *US* from *BEIJING* and worm outbreaks - *SQL_SSRP_SLAMMER_WORM* and cross site scripting intrusions -*HTTP_CROSS_SITE_SCRIPTING*.(See Figure 6.12).

Evaluating Radial Graph Alert Analysis

While neither the radial communication graph or radial coordinates visualisations are suitable for visualising time series data, both graphs are highly suitable for finding attribute correlation or associations which may reflect attack patterns.

Similarly to many visualisations, a key challenge is that an attempt to visualise a high volume of data would result in a cluttered visualisation which will be difficult to interpret. Therefore, as the information seeking mantra proposes, pre-analysis is required first to filter out the important information. In the next section, a set of visualisations for representing the alert correlation graphs generated by the correlation models in Chapter 3

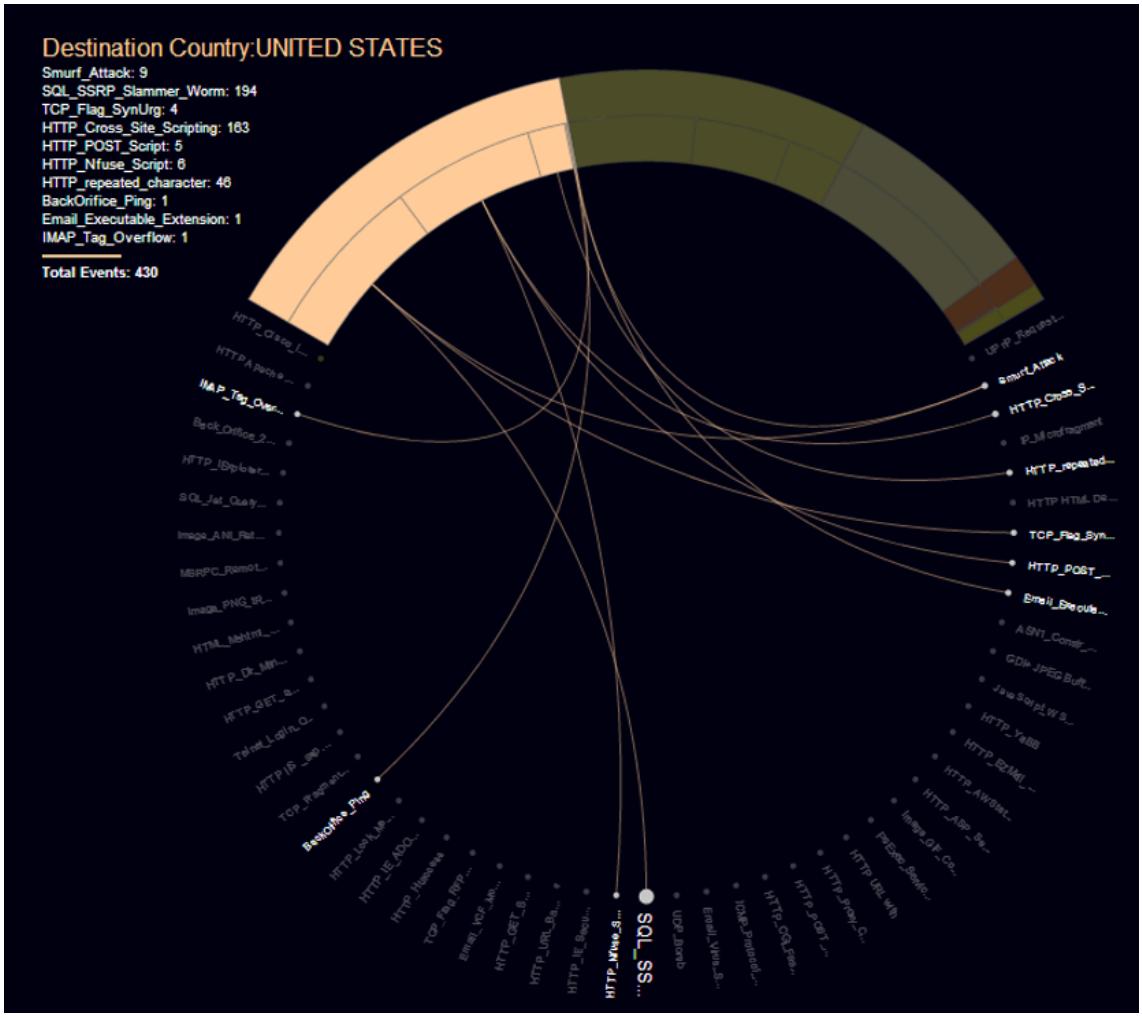

Figure 6.12: 2D Radial Coordinates showing Private Network Alert Capture (Dimension One = Alert Target, Dimension Two = Intrusion Type)

and prioritised by the automated analytical models proposed in Chapter 4 are explored.

Radial Graph Alert Analysis in Comparison to prior state of the art

The Radial coordinates visualisation proposed in this study is compared to similar recent visualisations proposed in [153] and [183]. *Avisa* [153], is closest to our Radial Coordinates. Similarly to radial coordinates, Avisa supports visualising two dimensions in a circular representation. Figure 6.13 shows Avisa (for a detailed description, see Section 2.6.5 of Chapter 2).

A key difference between the radial coordinates and Avisa is that our radial coordinates provides higher user interaction by supporting user selection of dimensions as well as details on demand (this is shown in Figure 6.11 where non-focussed elements are defocussed and details are provided inset). To our knowledge, our proposed radial approach

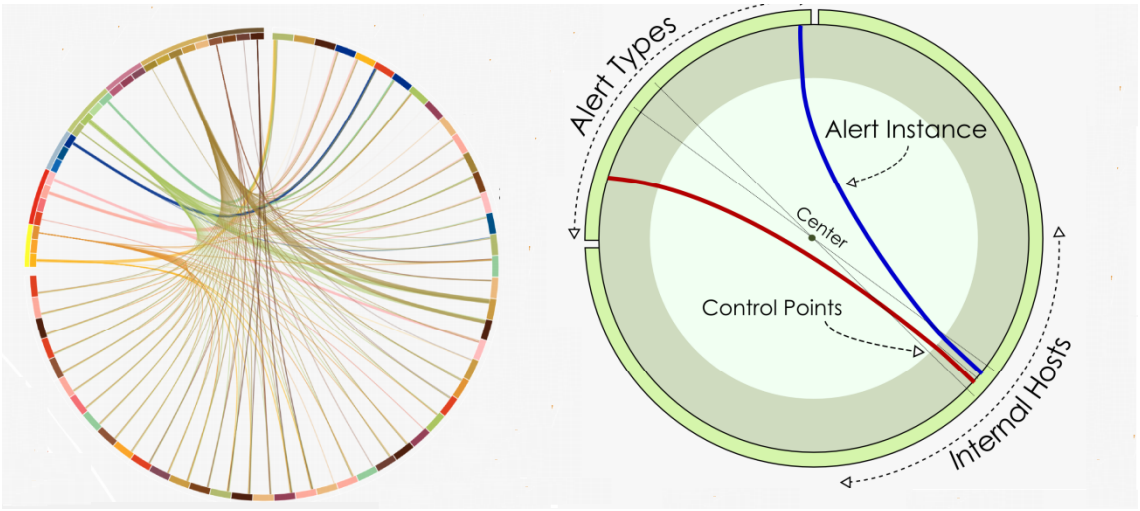

Figure 6.13: Avisa - A radial network visualisation proposed by Shiravi et al. [153]

is more dynamic. Nonetheless, Avisa supports additional features not supported by the proposed radial coordinates. This includes a component which filters the visualisation of hosts and alerts using on a set of heuristic metrics and standard statistical measurements such as standard deviations. While similar to the proposed approach, NetSecRadar [183], supports the visualisation of a multiple security logs. When compared with VisAlert, it was observed that at a single time, VisAlert supports visualising multiple alert logs including Snort alerts, FTP alerts and HTTP alerts. While the design for our 2D Radial Coordinates differs to VisAlert, our approach can be configured to visualise multiple alert logs in a single view. Pre-processing however may be required to collect the various alert logs into a single unified log file. Although VisAlert supports a set of features which are not supported by the 2D Radial Coordinates, these features can however be easily integrated. Unlike the 2D Radial Coordinates which does not represent the Time attribute, the timestamps of the alerts are represented in VisAlert using a set of inner cycles where each inner cycle equally represents a fixed time period. For clarification purposes, this is illustrated in Figure 6.14. This feature allows the user to quickly discover periodic patterns if configured properly. In addition, VisAlert visualises hosts based on their logical or physical proximity. While this can also be done in the 2D Radial Coordinates using *grouping*, the spatial representation is different.

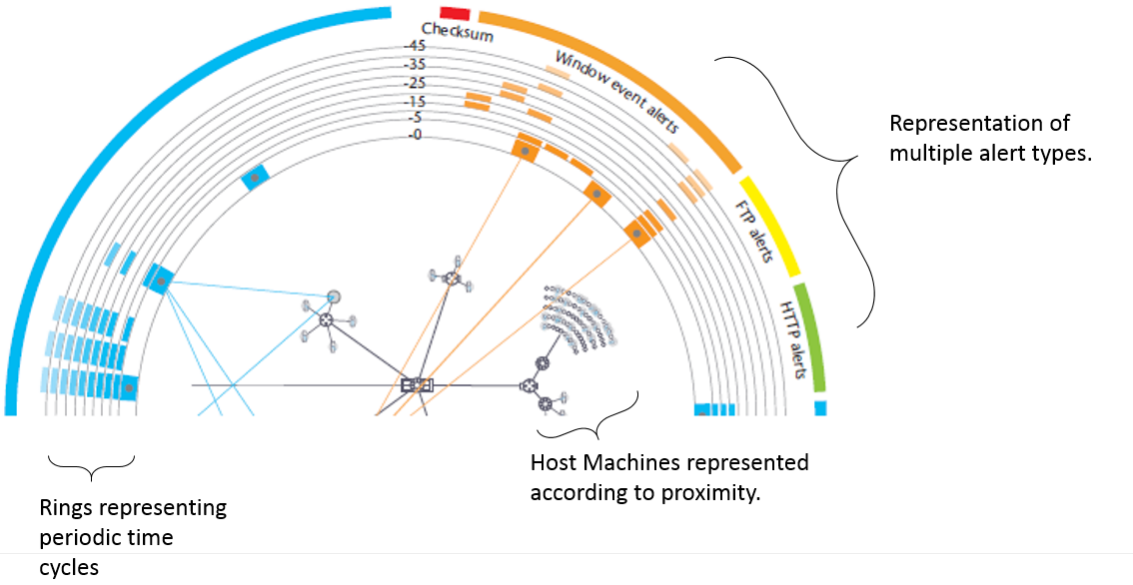

Figure 6.14: VizAlert - A radial network visualisation proposed by Livnat et al. [98]

6.3.3 Attack Analysis using Alert Correlation Graphs

Graph based visualisations (referred to as network graphs in [30] semiology of graphics) are arguably the most common visualisation technique for representing attack graphs. Many research studies including [123, 128, 144, 135, 162, 141] and many more, visualise attack graphs and alert correlation graphs using graph based visualisations. While the alert correlation graphs generated in Chapter 3 and the attack pattern graphs from Chapter 5 are not attack graphs, they share strong similarities.

Typically the graph based visualisation consists of at least two components. A set of nodes, where each node represents an entity. In an attack graph, a node may represent a phase in an attack while in an alert correlation graph, a node represents one or more aggregated alerts. An edge, in either case, represents the transition from one phase/alert to another.

Alert Correlation Graph Timeline

First, we proposed a time series visualisation which shares similar properties to a Gantt view [99]. In this visualisation, each alert correlation graph is represented as a node-edge graph ordered along the time axis, x. In order to position nodes effectively along the y-axis, a force directed algorithm [62] which pulls nodes with edges together is used. Figure 6.15 shows this visualisation where example alert correlation graphs are highlighted.

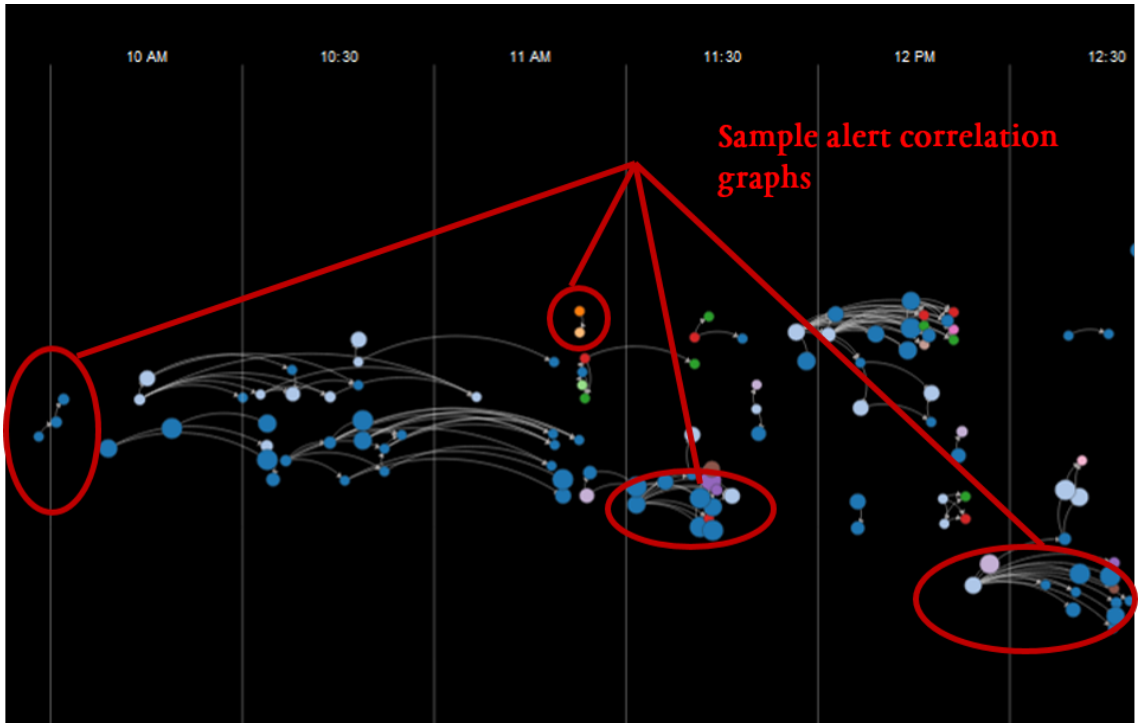

Figure 6.15: Initial Alert Correlation Graph Visualisation proposed during this research study

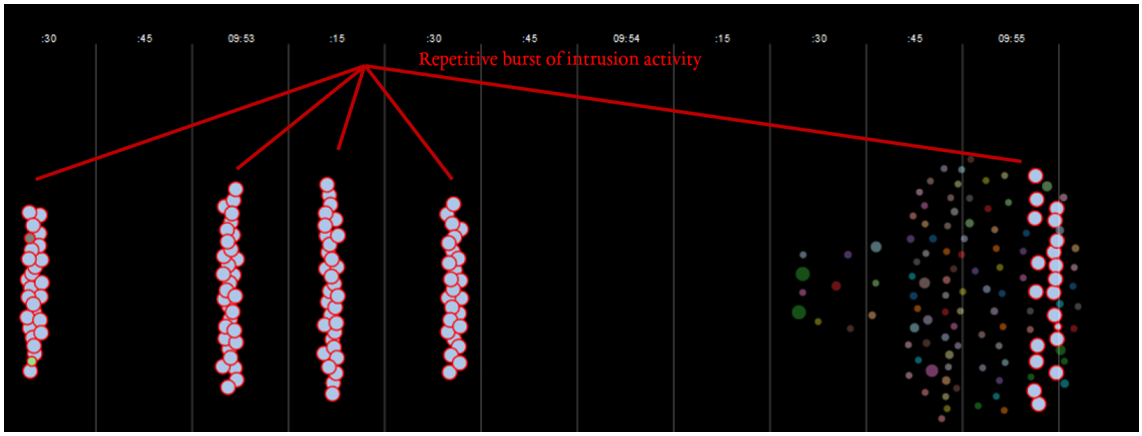

Figure 6.16: An alert correlation graph of interest containing repeative bursts of alerts. Graph is shown without edges to improve clarity.

USMA CDX 2009 Alert Analysis using an Alert Correlation Graph Timeline

Various limitations were identified. The most significant limitation or challenge was the high volume of clutter attributed to the visualisation resulting in graph overlapping which made it highly difficult to effectively isolate alert correlation graphs of interest. During the investigation however, it was possible to isolate a single alert correlation graph with interesting properties. Figure 6.16 shows this alert correlation graph. In order to show the properties of interest clearly, the edges are removed from the graph. By plotting the alert

correlation graph along the time-axis, it was possible to observe a repetitive burst of an intrusion type consistently occurring every 15 minute interval. Nonetheless, while it was possible to identify this graph, in general the overview of alert correlations graphs using the graph represented in Figure 6.15 is inefficient.

An Improved Alert Correlation Graph representation

A second and more effective approach was to visualise each alert correlation graph individually. This significantly reduced the clutter of the visualisation. In addition, in order to minimise and effectively organise the alert correlation graph nodes and edges, the GraphViz algorithm [63] was used to calculate the optimal positions of nodes and edges which minimised edge and node overlapping. Figure 6.17 shows the improved alert correlation graph visualisation. Each node in the visualisation represents a single or set of alerts and the y-axis positioning of the node is influenced by the order of the node i.e. the timestamp of the alerts the node represents. The node size indicates the number of alerts aggregated into a single node and the label and colour of each node represents the intrusion type. The arrowed edges between two nodes indicate the temporal correlation between the alerts.

Since the visualisation capability can only visualise a single alert correlation graph at a time, we developed an interactive feature for ordering alert correlation graphs according to a set of criteria. This feature allows the security analyst to order a set of alert correlation graphs by their timestamps, size, priority level (which is gained using the prioritisation metric proposed in Chapter 4), and cluster-id (which is gained using the clustering approach in Chapter 5). A navigation component was also integrated into the visualisation which allows the security analyst to navigate the alert correlation graphs in ascending or descending order. Zoom and Scroll features were also integrated into the visualisation to improve interactivity. Figure 6.18 shows these components.

Alert Correlation Graph Visualisation in Comparison to prior state of the art

To the author's knowledge, the most closest comprehensive system document in research papers which supported similar visual analytics was proposed by [117] over a decade ago. The advanced techniques and capabilities in modern visual analytics such as *interactivity,*

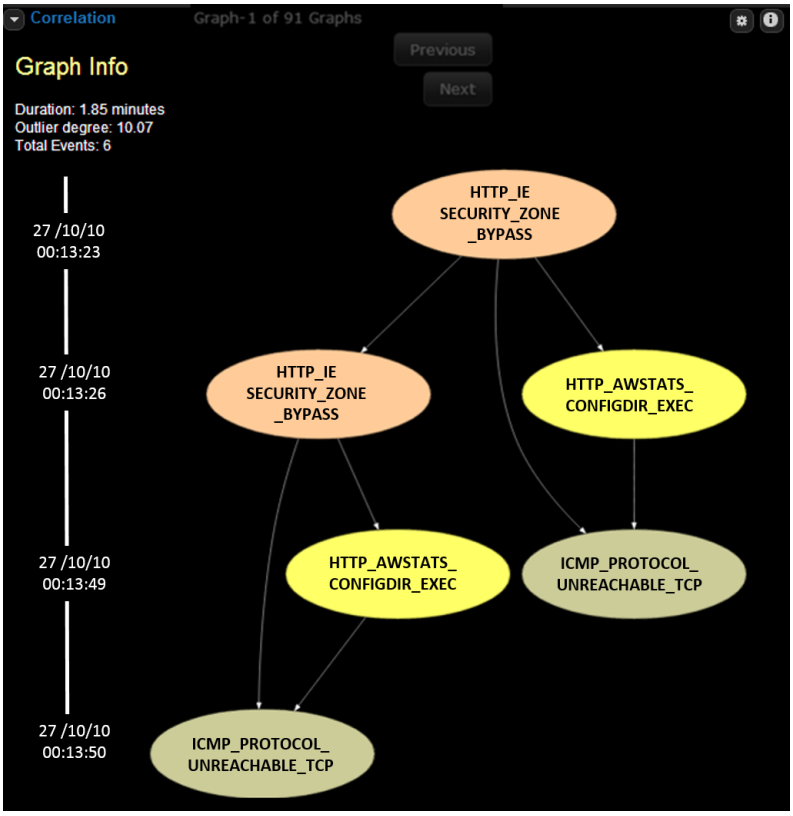

Figure 6.17: Alert Correlation Graph Visualisation showing Graph generated from Private Network Alert Data

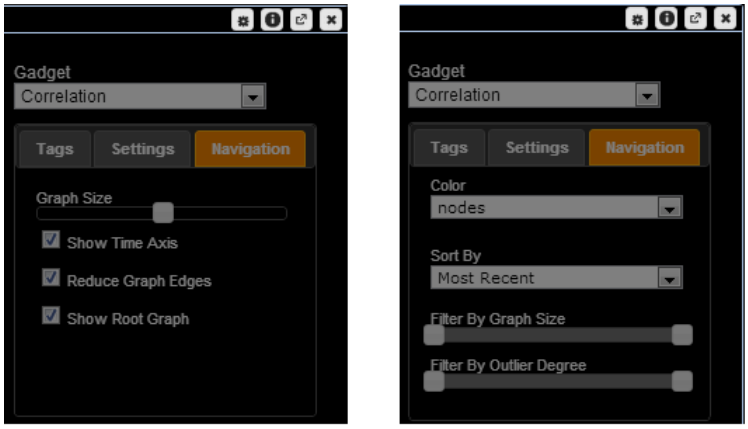

Figure 6.18: Settings Enabling Effective Options for Sorting Alert Correlation Graphs

filtering, linking, brushing and *responsive visual rendering* which were applied during the development of our proposed alert correlation graph visualisation allow our approach to support more effective alert investigation.

6.4 Integrating into Pre-existing system

The visualisations described in this Chapter and analytical methods in Chapter 3 - 5 were integrated into an experimental version of the SATURN Visual Analysis commercial web application developed by the Security Futures Practice, Research & Innovation, British Telecom. A brief description of SATURN is given.

SATURN is an AJAX enabled JavaScript web application. The Client-side application is a visualisation application which consists of components that allow a user to interactively load, query, visualise, and analyse a structured dataset while the Server-side component of the application allows the user to run data analysis on a given queried structured data. This may include data enriching, filtering, or categorising. These results are further visualised in the client-side application.

The stream graph, radial and alert correlation graph visualisations were developed in javascript using the d3.js library [33] and JQuery API [1]. All visualisations were integrated into the client-side application as shown in Figure 6.19.

The algorithms described in Chapter 3 - 5 were developed in Java as back end components which are exposed as web services using a range of Web Server APIs. The architecture in Figure 6.19 illustrates how the correlation view in the client-side requests for analysis of a security alert log using the visual interface. This prompts the launch of an XMLHttpRequest sent to the web service via the SATURN API. The XMLHttpRequest includes a flag to receive a response which contains the results of the correlation. i.e. one or more alert correlation graphs. The request also contains configuration parameters selected by the user,this includes correlation thresholds, windows and various configurations required by the correlation model, prioritisation model and clustering components. The analysis process is launched and starts a flow of analysis to be performed. The final output of the analysis process transforms the data into *json*[2], a format readable by the correlation view. The components in Figure 6.19 which are highlighted are the contributions of this research study.

[1] JQuery Source - https://jquery.com/

[2] JavaScript Object Notation (JSON) - http://www.w3schools.com/json

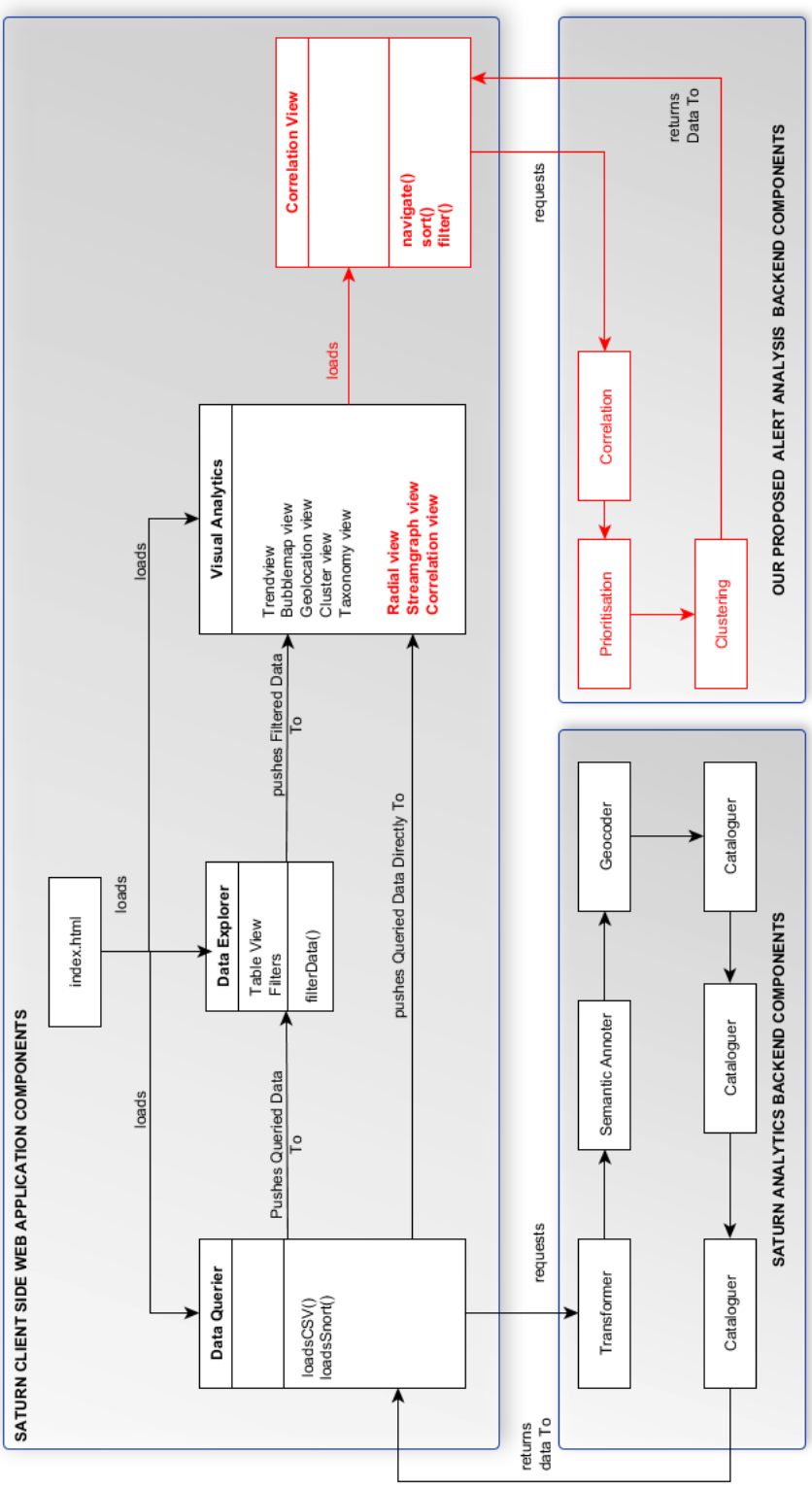

Figure 6.19: High Level Components of SATURN Visual Analytics tool showing integrated compnoents contributed during this research study

6.5 Conclusion

In this Chapter, visualisations for improving how a user gains insight into security alert data through the use of visualisation methods were explored. Three visualisation methods were proposed. We illustrated that visualisation alone is not sufficient for effective security data analysis. Furthermore, the proposed alert correlation visualisation shows that interactive visualisations improve the information conveyed by our designed automated approaches. Finally, we described how the visualisation techniques and methods described in Chapter 3, 4, and 5 were integrated into a visual analytics tool used for attack analysis.

Chapter 7

Conclusion

7.1 Overview

This study was set out to improve attack detection by proposing analytical methods for analysing security event logs. In Chapter 1, we stated that key pioneering research studies [116, 46, 86, 49, 166, 72] identified that an understanding of intrusions themselves, not as singular events, but rather as phased progressions, is required. These research studies argued that alert correlation enhances the context of an intrusion and subsequently, lead to the motivation of our research - *"To utilise the knowledge gained from alert correlation to eliminate false positive alerts and increase attack insight"*. As a result we formulated the following research question:

> *How can the information learnt from alert correlation be used to reduce false positive alerts? Furthermore, how can this knowledge be transformed into attack insights in order to increase the attack detection rate of a network?*

In order to achieve this, we explored statistical based methods for achieving alert correlation. To the authors knowledge, this approach provides the best trade-offs between the speed of the correlation model *(measured using time complexities)* and the accuracy of the correlation model, *(measured using the false positive and true positive rates)*. Therefore, we developed two correlation models based on Bayesian inference. Our findings described in Chapter 3.4 showed that the accuracy of an alert correlation model, based on a non-rule set method such as Bayesian inference can successfully identify a high volume

of true correlations amongst alerts if 1) the correct attributes and features are evaluated and 2) the model is not over-fitted. On the other hand, we observed that under-fitting the model would result in a high volume of correlations with very low accuracy. While this is a crucial issue, parameter estimation and optimisation for alert correlation was outside the scope of this research. The main focus of this Chapter, was to derive high quality alert correlation information and explore if, given this data, alert reduction and attack insight could be achieved and furthermore, if this novel approach was more effective than traditional approaches which do not consider alert correlation knowledge. Thereafter, in Chapters 4 and 5 we addressed how to utilise the output of the correlation analysis to achieve false positive reduction and increase attack detection respectively. Our findings in these chapters were as follows:

In Chapter 4, we aimed to answer the first part of the research question:

How can the information learnt from alert correlation be used to reduce false positive alerts?

We identified that the most common (and arguably naive) approach to eliminating false positives was to predominately analyse our correlation data against network and domain knowledge such as vulnerability scanner logs. However, during the preliminary study, we covered use-cases whereby such data is unattainable or is in unknown heterogeneous formats which require additional knowledge to translate. This is typically the case in a large scale computer network. As a result, we took a novel approach. We defined four prioritisation metrics for capturing attack dimensions. Our results showed that while the Outmet metric was useful and consistently filtered out false positive alerts in varying attack scenarios, metrics used to quantify the attack duration, attack interval and outgoing attack rate were less effective. Overall, when compared against other an existing approach, our results showed that the Outmet approach significantly reduced false positive.

In Chapter 5 we addressed the second part of the research question:

How can this knowledge be transformed into attack insights?

To address this we focussed on gaining attack insight by developing analytical methods which discover consistent attack patterns and features used by an attacker by extracting attack features from alert correlation graphs and mining these features to discover attack clusters. Our key finding in this Chapter was that the evaluated datasets derived from our case studies contained repetitive attack behaviour as a result of either real attacker activity or simulated attacker activity due to automated scripting. These activities were successfully identified frequently during the evaluation. We acknowledge that accuracy trade-offs were made to utilise heuristic distance functions in order to improve the speed of clustering such that it can be performed in near real-time. However, we showed that due to the nature of the alert correlation graphs, error-tolerant graph matching sufficiently captured the required features. A similar claim is also supported by related work [82].

In order to ensure that the identified attack insights are turned into actionable intelligence, we explored how to present this knowledge to a security analyst using a range of visualisation methods in Chapter 6.

The next section highlights the key achievements and contributions of this study to the domain of security applications and research. In Section 7.3, we suggest other domains where the methodologies in this research may be found useful. Finally, in Section 7.4 we discuss potential future work and some key open ended research questions which may be relevant to future research studies.

7.2 Contributions

Semi-automated probabilistic correlation model: Two probabilistic correlation models were presented. We showed that the models require little pre-configuration by a human-user and can automatically learn relationships between events. The main bottleneck in the proposed model is that it applies pairwise correlation resulting in *quadratic time complexity with respect to n - the number of alerts*. This however is a common shortcoming in alert correlation techniques and can be addressed by using sampling and selection methods to reduce the data volume. We tested the proposed models with two datasets and

provided two evaluation discussions. Our results showed that, in comparison to prior art [135], the Bayesian correlation approach (described in Section 3.2.3), showed on average a reduced false positive rate.

New Prioritisation Metrics: The four prioritisation metrics proposed in Chapter 4 present a new perspective to prioritising intrusions based on characteristics of a high-level attack. We tested each metric using three attack datasets. Of the metrics, *Outmet* showed the most promising with a best performance of a 100% true positive rate and a 1.98 % false positive rate with a relatively lower average of 84% and 5% respectively. In comparison to an approach proposed by [14], our findings showed that while *Outmet* demands more time and memory requirements, the simplistic and less complex metric which [14] proposed for prioritisation alerts with high correlations and similarities as high priorities is not a best approach. Evaluating their model using one of our datasets, we discovered that their metric produced a false positive rate of 53%. This reduction is not as significant as *Outmets*. The design and experimental evaluation of *Outmet* is published in [159]

An effective algorithm for clustering multi-attributed graphs in real-time: We proposed an adaptation of CluStream, (see Chapter 5 for details) for clustering alert graphs and sequences. The key achievement in the proposed adaptation is that it allows the clustering algorithm to perform cluster analysis graph data structures. For general purposes, this algorithm can also be applied to clustering generic multi-attributed directed graphs. During the evaluation of our adaptation of Clustream, we showed its effectiveness in successfully clustering attacks from different attack groups such as denial-of-service attacks and worm propagations. This is to be published in [158].

A Comprehensive System for Intrusion Alert Analysis: The proposed correlation models, prioritisation metrics and real-time clustering of alert correlation graphs algorithm can be collectively integrated as a framework for intrusion analysis or can be individually utilised. Through support provided by the sponsors and collaborators of this research, the individual components were integrated into a visual analytic tool developed by the research team at British Telecoms (described in Chapter 6 and published in [159]).

7.3 Other Applications

In this study there have been a range of contributions all aligned to the improvement of attack detection. Given that the methods were specifically proposed and tailored for attack detection, it may be difficult to apply them all together to other domains. The individual components, however, can be applied. We discuss how some of the proposed analytical techniques can be applied to other domains.

7.3.1 Network Traffic Analysis

This involves the monitoring and analysis of network traffic data which could be performed for other reasons aside attack analysis. In addition the data sources range from network traces to layer 7 application logs. Figure 7.1 shows an illustration of how the radial visualisation proposed in Chapter 6 can be used to present network data for monitoring purposes. In this visualisation, two dimensions of network traffic are shown, it shows how visualising the source and destination of network traffic can aid in understanding the flow of monitored traffic.

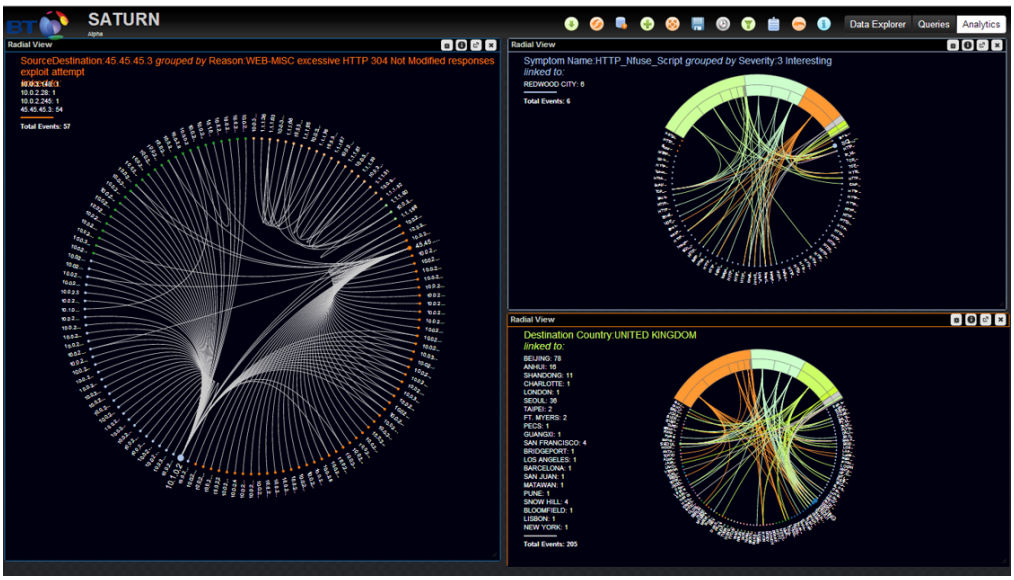

Figure 7.1: Radial Visualisation developed in this research after the SATURN integration[155]

7.3.2 Social Network Analysis

Social Network Analysis (SNA) is the study of relations between individuals including the analysis of social structures, social position, role analysis, and many others [4]. In

SNA key entities are represented as nodes in a graph and the interactions between the entities are represented as edges. Visual methods such as the proposed radial view are well suited for visualising social network data. Aside visual methods, graph theories such as graph clustering, outlier detection, prediction, and classification are commonly applied in social network analysis. Therefore, we believe that the clustering algorithm described in Chapter 5 may be suitable for SNA challenges.

7.3.3 Bioinformatics

Similarly to SNA, the clustering algorithm may be applicable to solving graph related problems in bioinformatics. In the field of bioinformatics, protein structures can be represented as graphs. Atoms, chemical compounds and genetic data can also be modelled using graphs.

7.4 Ongoing Challenges and Future Work

> *"The challenge is to develop a system that is able to detect all attacks and produce a minimal number of false positives..."* [85].
>
> Kruegel et al. 2005

Ten years on and developing such a system is still a major challenge. Today, the most well known deployed intrusion detection systems are signature based IDSs [12]. Despite their limitations, they are simplest and most effective in detecting known attacks and while there is an opportunity for hybrid and new detection methods such as state-ful protocol analysis to arise [93], successful anomaly based methods which can be independently deployed on a network are difficult to attain for two key reasons. Firstly, we argue that it is impossible to collect network data which *fully* reflects the differences between all attacks and normal network behaviour therefore, data modelling performed on insufficient training data will result in insufficient models. Take for instance, many approaches are being evaluated using the DARPA 1998 - 2000 datasets. Such datasets are over a decade old and no longer represent the scales and complexities involved in a modern network and attack. Given, the availability of sufficient and adequate data, the next challenge is

to model the differences between attack and normal network behaviour using algorithmic methods ensuring minimal false positives.

In this study, we focussed on improving signature based intrusion detection systems by applying post-intrusion analysis methods. We propose that ongoing, research effort continues to address how to effectively improve intrusion detection in some of the following areas.

Adequate Data: Many of the novelties and contributions of this research were made possible because the case studies and datasets used were based on newer and modern realistic scenarios. However, too few publicly available datasets with sufficient ground truth exist and according to [147], no modern publicly available replacements exist for the DARPA dataset. The collection of adequate datasets for building improved intrusion detection methods is required. In addition, evaluation techniques, particularly, in alert correlation are yes to be standardised [85].

Graph Theory Application: Aside attack scenarios and alert correlation graphs, many entities in network and intrusion detection such as IP-communication which represent host interactivity and network topologies can be represented as graph structures. Therefore, graph mining and theories may be suitably applied to this domain. In this research, only graph clustering was explored. Methods for measuring graph properties for example measuring the centrality of a graph or performing graph-node clustering for discovering key hosts (or perhaps botnet controllers) on the network may be suitable.

Parameter Estimation: In Chapter 3, a probabilistic bayesian correlation model which required the configuration of time-windows, thresholds and the selection of alert features was proposed. Our results showed that the effectiveness of the selected time-window and correlation threshold parameters were specific to datasets. While we did not address how to dynamically select model parameters, this is a key area which may be addressed. In addition, for post-intrusion analysis, we faced feature selection issues. Which features are best used? Are port addresses useful? Other parameter estimation issues arose. The parameter k was also reoccurring in Chapter 4 for *outmet* and Chapter 5 for the adaptation of *Clustream*. Dynamic approaches for selecting parameters which produce quality

clusters has been studied extensively in machine learning.

Prediction Capabilities: Human validated attack patterns can be used for future predictions. In this research, we have presented a dynamic method for generating attack patterns. Once validated, such patterns represented as graph structures can be used to predict future attacks using heuristic, statistical or graph mining methods. Such research is currently being explored in [181].

This study was set out to improve attack detection by proposing analytical methods for reducing false positive alerts and increasing attack insight through analysing security event logs. It has provided a critical argument and, used empirical evidence to show that event correlation is a key analytical method to understanding high-level attacks which further aids in the identification of false positive alerts. It has shown that contrary to prior art the output of automated statical alert correlation methods, (though challenging) can be effectively utilised in the discovery of attack patterns employed by attackers.

Bibliography

[1] A, T., K, G. R., and Agrawala, D. P. (2010). A survey of frequent pattern mining: Current Status and Challenging issues. *Information Technology Journal*.

[2] Abdullah, K., Lee, C., Conti, G., Copeland, J., and Stasko, J. (2005). IDS RainStorm: Visualizing IDS Alarms. *VizSEC*.

[3] Aggarwal, C. C., Ctr, T. J. W. R., Han, J., Wang, J., and Yu, P. S. (2003). A Framework for Clustering Evolving Data Streams. *Proceedings of the 29th international conference on Very large data bases-Volume 29. VLDB Endowment*.

[4] Aggarwal, C. C. and Wang, H., editors (2010). *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*. Springer US, Boston, MA.

[5] Aggarwal, C. C., Zhao, Y., and Yu, P. S. (2010a). On Clustering Graph Streams. *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 478–489.

[6] Aggarwal, C. C., Zhao, Y., and Yu, P. S. (2010b). On clustering graph streams. *Sdm*, pp:478–489.

[7] Agrawal, R., ImieliÅĎski, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, 22(2):207–216.

[8] Ahmadinejad, S. H. and Jalili, S. (2009). Alert Correlation Using Correlation Probability Estimation and Time Windows. *2009 International Conference on Computer Technology and Development*, (1):170–175.

[9] Aigner, W., Miksch, S., and Schumann, H. (2008). Visual Methods for Analyzing Data. 14(1).

[10] Al-Mamory, S. O. and Zhang, H. (2007). A Survey on IDS Alerts Processing Techniques. *ISP'07 Proceedings of the 6th WSEAS international conference on Information security and privacy*, pages 69–78.

[11] Alessandri, D. (2004). Attack-class-based analysis of intrusion detection systems. (May):244.

[12] Alhomoud, A., Munir, R., Disso, J. P., Awan, I., and Al-Dhelaan, A. (2011). Performance evaluation study of Intrusion Detection Systems. *Procedia Computer Science*, 5:173–180.

[13] Alserhani, F. (2013). A framework for multi-stage attack detection. *2013 Saudi International Electronics, Communications and Photonics Conference, SIECPC 2013*.

[14] Alsubhi, K., Aib, I., and Boutaba, R. (2012). FuzMet : a fuzzy-logic based alert prioritization engine for intrusion detection systems. *International Journal of Network Management*, 22(4):263–284.

[15] Alsubhi, K., Al-Shaer, E., and Boutaba, R. (2008). Alert prioritization in Intrusion Detection Systems. *NOMS 2008 - 2008 IEEE Network Operations and Management Symposium*, pages 33–40.

[16] Anuar, N. B. (2012). *Incident Prioritisation for Intrusion Response Systems*. PhD thesis, University of Plymouth.

[17] Anuar, N. B., Furnell, S., Papadaki, M., and Clarke, N. (2011). A risk index model for security incident prioritisation. *Australian Information Security Management Conference*, pages 24–39.

[18] Anuar, N. B., Papadaki, M., Furnell, S., and Clarke, N. (2013). Incident prioritisation using analytic hierarchy process (AHP): Risk Index Model (RIM). *SECURITY AND COMMUNICATION NETWORKS*, 6(December 2012):1087–1116.

[19] April, P. (2013). Internet Security Threat Report Government 2013. Technical Report April.

[20] Arthur, D., Arthur, D., Vassilvitskii, S., and Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 8:1027–1035.

[21] Ashley, M. (2006). White paper LAYERED NETWORK SECURITY 2006 : A best-practices approach. Technical Report January.

[22] Axelsson, S. (1999). The base-rate fallacy and its implications for the difficulty of intrusion detection. *Proceedings of the 6th ACM conference on Computer and communications security - CCS '99*, pages 1–7.

[23] Axelsson, S. (2000). The Base-Rate Fallacy and the Difficulty of Intrusion Detection. 3(3):186–205.

[24] Bai, H., Wang, K., Hu, C., Zhang, G., and Jing, X. (2010). Boosting performance in attack intention recognition by integrating multiple techniques. *Frontiers of Computer Science in China*, 5(1):109–118.

[25] Bateni, M. and Baraani, A. (2013). Time Window Management for Alert Correlation using Context Information and Classification. (September):9–16.

[26] Bateni, M., Baraani, A., and Ghorbani, A. A. (2013). Using Artificial Immune System and Fuzzy Logic for Alert Correlation. 15(1):160–174.

[27] Baumerucker, T., Burton, J., Dentler, S., Dubrawsky, I., Osipov, V., and Sweeney, M. (2003). *Cisco security professional's guide to secure intrusion detection systems*. Syngress Publishing.

[28] Benferhat, S., Boudjelida, A., Tabia, K., and Drias, H. (2013). An intrusion detection and alert correlation approach based on revising probabilistic classifiers using expert knowledge. *Applied intelligence*, 38(4):520–540.

[29] Benferhat, S., Kenaza, T., and Mokhtari, A. (2008). A Naive Bayes Approach for Detecting Coordinated Attacks. *32nd Annual IEEE International Computer Software and Applications Conference*, pages 704–709.

[30] Bertin, J. (1983). Semiology of graphics: Diagrams, networks, maps (WJ Berg, Trans.). *Madison, WI: The University of Wisconsin Press, Ltd*.

[31] Bertini, E., Hertzog, P., and Lalanne, D. (2007). SpiralView: Towards Security Policies Assessment through Visual Correlation of Network Resources with Evolution of Alarms. *2007 IEEE Symposium on Visual Analytics Science and Technology*, pages 139–146.

[32] Bilge, L., Balzarotti, D., Robertson, W., and Kruegel, C. (2012). DISCLOSURE : Detecting Botnet Command and Control Servers Through Large-Scale NetFlow Analysis. *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 129–138.

[33] Bostock, M. (2012). D3. js. *Data Driven Documents*.

[34] Breunig, M. M., Kriegel, H.-p., Ng, R. T., and Sander, J. (2000). LOF : Identifying Density-Based Local Outliers. *Proceedings Of The 2000 Acm Sigmod International Conference On Management Of Data*, pages 1–12.

[35] Browne, H. K., Arbaugh, W. a., McHugh, J., and Fithen, W. L. (2001). A trend analysis of exploitations. *Security and Privacy, 2001. S\&P 2001. Proceedings. 2001 IEEE Symposium on*, pages 214–229.

[36] Bunke, H. and Ã, K. R. (2011). Improving vector space embedding of graphs through feature selection algorithms. *Pattern Recognition*, 44(9):1928–1940.

[37] Bunke, H. and Neuhaus, M. (2007). *Mining graph data*. John Wiley & Sons.

[38] Byron, L. and Wattenberg, M. (2008). Stacked graphs - Geometry & aesthetics. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1245–1252.

[39] Cakmak, A. and Ozsoyoglu, G. (2008). Taxonomy-superimposed graph mining. *Proceedings of the 11th international conference on Extending database technology: Advances in database technology. ACM*.

[40] Cédric Michel, L. M. (2001). Adele: An Attack Description Language For Knowledge-Based Intrusion Detection. *Trusted Information*, pages 353–368.

[41] CERT-UK (2014). CERT-UK Quarterly Report. Technical report.

[42] Chen, S., Leung, H., and Dondo, M. (2014). Characterization of computer network events through simultaneous feature selection and clustering of intrusion alerts. In *SPIE Sensing Technology + Applications, International Society for Optics and Photonics*.

[43] Cheng, B. C., Liao, G. T., Huang, C. C., and Yu, M. T. (2011). A novel probabilistic matching algorithm for multi-stage attack forecasts. *IEEE Journal on Selected Areas in Communications*, 29(7):1438–1448.

[44] Cheswick, W., Bellovin, S., and Rubin, A. (2003). *Firewalls and Internet security: repelling the wily hacker*.

[45] Cheung, S., Fong, M. W., Ave, R., and Park, M. (2003). Modeling Multistep Cyber Attacks for Scenario Recognition. *In DARPA Information Survivability Conference and Exposition (DISCEX III)*, (DISCEX III):284–292.

[46] Cuppens, F. and Miège, A. (2002). Alert Correlation in a Cooperative Intrusion Detection Framework. *Proceedings of the 2002 IEEE Symposium on Security and Privacy*.

[47] Cuppens, F., Miège, A., Belin, E., and Cedex, T. (2002). Alert Correlation in a Cooperative Intrusion Detection Framework.

[48] Cuppens, F. and Ortalo, R. (2000). LAMBDA: A Language to Model a Database for Detection of Attacks. *Recent advances in intrusion detection. Springer Berlin Heidelberg*, pages 197–216.

[49] Dain, O. and Cunningham, R. K. (2001). Fusing a Heterogeneous Alert Stream into Scenarios. *In Proceedings of the 2001 ACM workshop on Data Mining for Security Applications*, pages 1–13.

[50] Debar, H. and Wespi, A. (2001). Aggregation and Correlation of Intrusion-Detection Alerts. *Recent Advances in Intrusion Detection.*, pages 85–103.

[51] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society,*, 39(1):1–38.

[52] Denning, D. E. (1987). An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, 13(2):222–232.

[53] Draper, G. M., Livnat, Y., and Riesenfeld, R. F. (2009). A survey of radial methods for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):759–776.

[54] Du, H. and Yang, S. J. (2014). Probabilistic Inference for Obfuscated Network Attack Sequences. *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*.

[55] Ebrahimi, A., Zad Navin, A. H., Kamal Mirnia, M., Bahrbegi, H., and Alasti Ahrabi, A. A. (2011). Automatic attack scenario discovering based on a new alert correlation method. *2011 IEEE International Systems Conference, SysCon 2011 - Proceedings*, pages 52–58.

[56] Elshoush, H. T. and Osman, I. M. (2010). Reducing false positives through fuzzy alert correlation in collaborative intelligent intrusion detection systems - A review. *International Conference on Fuzzy Systems*, pages 1–8.

[57] Falliere, N., Murchu, L. O., and Chien, E. (2011). Symantec Report: W32 . Stuxnet Dossier. Technical Report February.

[58] Fernandez, E., Pelaez, J., and Larrondo-petrie, M. (2007). Attack Patterns : a New Forensic and Design Tool. *Advances in digital forensics III. Springer New York*, 242:345–357.

[59] Ferrer, Z. and Ferrer, M. C. (2010). In-depth Analysis of Hydraq The face of cyber-war enemies unfolds.

[60] Fischer, F., Davey, J., Fuchs, J., Thonnard, O., Kohlhammer, J., and Keim, D. A. (2014). A Visual Analytics Field Experiment to Evaluate Alternative Visualizations for Cyber Security Applications.

[61] Fredj, O. B. (2015). A realistic graph-based alert correlation system. *Security and Communication Networks*, 8(15):2477–2493.

[62] Fruchterman, T. M. J. and Reingold, E. M. (1991). Graph Drawing by Force-directed Placement. *Software-Practice and Experience*, 21(NOVEMBER):1129–1164.

[63] Gansner, E. (2011). Drawing graphs with Graphviz. *Graphviz Drawing Library Manual*.

[64] Goodall, J. R. (2008). Introduction to VIsualization for Computer Security. *VIzSEC*, pages 1–18.

[65] Group, T. S. (2009). Tracking GhostNet : Investigating a Cyber Espionage Network. Technical report.

[66] Gu, G., Porras, P., Yegneswaran, V., Fong, M., and Lee, W. (2007). BotHunter: detecting malware infection through IDS-driven dialog correlation. *USENIX Security*

'07 Proceedings of the 16th USENIX Security Symposium, page 12.

[67] Gula, R. (2002). Correlating IDS Alerts with Vulnerability Information. *Tenable Network Security http://www. tenablesecurity.*, 2011(January):10.

[68] Hansman, S. and Hunt, R. (2005). A taxonomy of network and computer attacks. *Computers & Security*, 24(1):31–43.

[69] Heady, R., Luger, G., Maccabe, a., and Servilla, M. (1990). The architecture of a network-level intrusion detection system. *Zhurnal Eksperimental'noi i Teoreticheskoi Fiziki*, 0.

[70] Heckman, S. and Williams, L. (2008). On Establishing a Benchmark for Evaluating Static Analysis Alert Prioritization and Classification Techniques. *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement. ACM*.

[71] Hofmann, A. and Sick, B. (2011). Online Intrusion Alert Aggregation with Generative Data Stream Modeling. *IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING*, 8(2):282–294.

[72] Hutchins, E. M. (2008). Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. (July 2005):1–14.

[73] Incorporated, T. M. (2012). Activity with Network Traffic Analysis.

[74] Jha, S., Sheyner, O., and Wing, J. (2002). Two formal analyses of attack graphs. *Proceedings 15th IEEE Computer Security Foundations Workshop. CSFW-15*.

[75] Jinghu, X., Aiping, L., Hui, Z., and Hong, Y. (2012). A Multi-Step Attack Pattern Discovery Method Based on Graph Mining. *Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on. IEEE*.

[76] Johansson, J., Ljung, P., Jern, M., and Cooper, M. (2006). Revealing structure in visualizations of dense 2D and 3D parallel coordinates. *Information Visualization*, 5(2):125–136.

[77] Julisch, K. (2000). Dealing with false positives in intrusion detection. *extended abstract at RAID*.

[78] Karim Ganame, A., Bourgeois, J., Bidou, R., and Spies, F. (2008). A global security

architecture for intrusion detection on computer networks. *Computers & Security*, 27:30–47.

[79] Kavousi, F. and Akbari, B. (2012). Automatic Learning of Attack Behavior Patterns Using Bayesian Networks. *6'th International Symposium on Telecommunications (IST'2012)*, pages 999–1004.

[80] Keim, D. a., Mansmann, F., Thomas, J., and Keim, D. (2010). Visual Analytics : How Much Visualization and How Much Analytics ? *ACM SIGKDD Explorations Newsletter*, 11(2):5–8.

[81] Kent, K. and Souppaya, M. (2006). NIST: Guide to Computer Security Log Management. Technical report.

[82] Khan, A., Yan, X., and Wu, K.-L. (2010). Towards proximity pattern mining in large graphs. *Proceedings of the 2010 international conference on Management of data - SIGMOD '10*, page 867.

[83] Kintzel, C., Fuchs, J., and Mansmann, F. (2011). Monitoring large IP spaces with ClockView. *Proceedings of the 8th International Symposium on Visualization for Cyber Security - VizSec '11*, pages 1–10.

[84] Koike, H. and Ohno, K. (2004). SnortView. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security - VizSEC/DMSEC '04*, page 143, New York, New York, USA. ACM Press.

[85] Kruegel, C., Valeur, F., and Vigna, G. (2005). *Intrusion Detection and Correlation: Challenges and Solutions*.

[86] Krügel, C., Toth, T., Kerer, C., Kr, C., and Group, D. S. (2001). Decentralized Event Correlation for Intrusion Detection. *Information Security and Cryptology - ICISC*, LNCS 2288/:1–21.

[87] Lagzian, S. (2012). Frequent Item set mining-based Alert Correlation for Extracting multi-stage Attack Scenarios. *IEEE Telecommunications (IST), 2012 Sixth International Symposium*, pages 1010–1014.

[88] Lee, K., Kim, J., Kwon, K. H., Han, Y., and Kim, S. (2008). DDoS attack detection method using cluster analysis. *Expert Systems with Applications*, 34(3):1659–1665.

[89] Lee, S., Chung, B., Kim, H., Lee, Y., Park, C., and Yoon, H. (2006). Real-time analysis of intrusion detection alerts via correlation. *Computers and Security*, 25(3):169–183.

[90] Li, D., Li, Z., and Ma, J. (2008). Processing intrusion detection alerts in large-scale network. *Proceedings of the International Symposium on Electronic Commerce and Security, ISECS 2008*, pages 545–548.

[91] Li, W., Zhi-tan, L., Dong, L., and Jie, L. (2007a). Attack scenario construction with a new sequential mining technique. *Proceedings - SNPD 2007: Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, 1:652–657.

[92] Li, Z. T., Lei, J., Wang, L., and Li, D. (2007b). A data mining approach to generating network attack graph for intrusion prediction. *Proceedings - Fourth International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2007*, 4:307–311.

[93] Liao, H.-J., Richard Lin, C.-H., Lin, Y.-C., and Tung, K.-Y. (2012). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24.

[94] Lin, Z., Li, S., and Ma, Y. (2010). Real-time intrusion alert correlation system based on prerequisites and consequence. *2010 6th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2010*.

[95] Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., and Das, K. (2000). Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation. *In International Symposium on Recent Advances in Intrusion Detection*, pages 162–182.

[96] Liu, L., Zheng, K., and Yang, Y. X. (2010). An intrusion alert correlation approach based on finite automata. *Proceedings - 2010 International Conference on Communications and Intelligence Information Security, ICCIIS 2010*, pages 80–83.

[97] Liu, Z., Wang, C., and Chen, S. (2008). Correlating Multi-Step Attack and Constructing Attack Scenarios Based on Attack Pattern Modeling. *2008 International Conference on Information Security and Assurance (isa 2008)*, pages 214–219.

[98] Livnat, Y., Agutter, J., Moon, S., and Foresti, S. (2005). Visual correlation for

situational awareness. *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, 1:95–102.

[99] Mackinlay, J., Hanrahan, P., and Stolte, C. (2007). Show me: automatic presentation for visual analysis. *IEEE transactions on visualization and computer graphics*, 13(6):1137–44.

[100] Maggi, F. and Zanero, S. (2010). On the Use of Different Statistical Tests for Alert Correlation - Short Paper. *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 167–177.

[101] Marchetti, M., Colajanni, M., and Manganiello, F. (2011a). Framework and Models for Multistep Attack Detection. *International Journal of Security and Its Applications*, 5(4):73–92.

[102] Marchetti, M., Colajanni, M., and Manganiello, F. (2011b). Identification of correlated network intrusion alerts Pseudo-Bayesian. *Cyberspace Safety and Security (CSS), 2011 Third International Workshop on*, pages 15–20.

[103] Marty, R. (2008). *Applied Security Visualizaton*. Upper Saddle River: Addison-Wesley.

[104] McAfee Labs (2013). McAfee Labs Threats Report.

[105] Mell, P., Scarfone, K., and Romanosky, S. (2006). Common Vulnerability Scoring System. *IEEE Security & Privacy Magazine*, 4(6):85–89.

[106] Mirheidari, S. A., Arshad, S., and Jalili, R. (2013). Alert correlation algorithms: A survey and taxonomy. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8300 LNCS:183–197.

[107] MITRE (2008). Common Event Expression. Technical Report June.

[108] Moore, A. P., Ellison, R. J., and Linger, R. C. (2001). Attack modeling for information security and survivability. Technical report, Carnegie-Mellon University Software Engineering Institute, Pittsburgh.

[109] Morin, B., Mé, L., Debar, H., and Ducassé, M. (2002). M2D2: a formal data model for IDS alert correlation. *5th international conference on Recent advances in intrusion*

detection, pages 115–137.

[110] Morin, B., Mé, L., Debar, H., and Ducassé, M. (2009). A logic-based model to support alert correlation in intrusion detection. *Information Fusion*, 10(4):285–299.

[111] Motegaonkar, V. S. and Vaidya, P. M. V. (2014). A Survey on Sequential Pattern Mining Algorithms. *International Journal of Computer Science and Information Technologies*, 5(2):2486–2492.

[112] Musa, S. and Parish, D. J. (2008). Using Time Series 3D AlertGraph and False Alert Classification to Analyse Snort Alerts. *Visualization for Computer Security*, pages 169–180.

[113] Neumann, P. and Parker, D. (1989). A summary of computer misuse techniques. *Proceedings of the 12th National Computer Security Conference*.

[114] Nguyen, T. H., Luo, J., and Njogu, H. W. (2014). An Approach to Verify , Identify and Prioritize IDS Alerts. 7(6):395–410.

[115] Ning, P., Cui, Y., and Reeves, D. S. (2002a). Analyzing Intensive Intrusion Alerts Via Correlation. *5th International Symposium In Recent Advances in Intrusion Detection*.

[116] Ning, P., Cui, Y., and Reeves, D. S. (2002b). Constructing Attack Scenarios through Correlation of Intrusion Alerts. *Proceedings of the 9th ACM conference on Computer and communications security*, pp:10.

[117] Ning, P., Peng, P., Hu, Y., and Xu, D. (2003). TIAA : A Visual Toolkit for Intrusion Alert Analysis.

[118] Ning, P., Reeves, D. S., and Cui, Y. (2001). Correlating Alerts Using Prerequisites of Intrusions. Technical report, North Carolina State University, Raleigh NC,.

[119] Ning, P. and Xu, D. (2003a). Learning attack strategies from intrusion alerts. *Proceedings of the 10th ACM conference on Computer and communication security - CCS '03*, page 200.

[120] Ning, P. and Xu, D. (2003b). Learning attack strategies from intrusion alerts. *Proceedings of the 10th ACM conference on Computer and communication security - CCS '03*, page 200.

[121] Ning, P. and Xu, D. (2004). Hypothesizing and reasoning about attacks missed by intrusion detection systems. *ACM Transactions on Information and System Security*, 7(4):591–627.

[122] Njogu, H. W., Jiawei, L., Kiere, J. N., and Hanyurwimfura, D. (2013). A comprehensive vulnerability based alert management approach for large networks. *Future Generation Computer Systems*, 29(1):27–45.

[123] Noel, S. and Jajodia, S. (2004). Managing attack graph complexity through visual hierarchical aggregation. *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security - VizSEC/DMSEC '04*, page 109.

[124] Noel, S. and Jajodia, S. (2007). Attack Graphs for Sensor Placement , Alert Prioritization , and Attack Response. *Cyberspace Research Workshop*, pages 1–8.

[125] Noel, S., Robertson, E., and Jajodia, S. (2004). Correlating intrusion events and building attack scenarios through attack graph distances. *Proceedings - Annual Computer Security Applications Conference, ACSAC*, pages 350–359.

[126] Nyarko, K., Capers, T., Scott, C., and Ladeji-Osias, K. (2002). Network intrusion visualization with NIVA, an intrusion detection visual analyzer with haptic integration. *Proceedings 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. HAPTICS 2002*, pages 277–284.

[127] Ogle, D., Kreger, H., Salahshour, A., Cornpropst, J., Chessel, M., Gerken, J., Horn, B., Labadie, E., Schoech, J., and Wamboldt, M. (2002). Canonical Situation Data Format : The Common Base Event V1.0.1.

[128] Ou, X., Boyer, W. F., and McQueen, M. a. (2006). A scalable approach to attack graph generation. *Proceedings of the 13th ACM conference on Computer and communications security - CCS '06*, page 336.

[129] Ou, X., Rajagopalan, S. R., and Sakthivelmurugan, S. (2009a). An Empirical Approach to Modeling Uncertainty in Intrusion Analysis. *Annual Computer Security Applications Conference*, (Observation 1):494–503.

[130] Ou, X., Rajagopalan, S. R., and Sakthivelmurugan, S. (2009b). An Empirical Approach to Modeling Uncertainty in Intrusion Analysis. *2009 Annual Computer Security*

Applications Conference, pages 494–503.

[131] Peter Mell and Grance, T. (2002). Use of the Common Vulnerabilities and Exposures (CVE) Vulnerability Naming Scheme. Technical report.

[132] Porras, P. A., Fong, M. W., and Valdes, A. (2002). A Mission-Impact-Based Approach to INFOSEC Alarm Correlation. *Recent Advances in Intrusion Detection*, (Springer Berlin Heidelberg):95–114.

[133] Qin, X. (2005). *A Probabilistic-Based Framework for INFOSEC Alert Correlation*. PhD thesis, Georgia Institute of Technology.

[134] Rajaraman, A. and Ullman, J. D. (2011). *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA.

[135] Ren, H., Stakhanova, N., and Ghorbani, A. A. (2010). An Online Adaptive Approach to Alert Correlation. *Proceedings of the 7th international conference on Detection of Intrusions and malware, and vulnerability assessment (DIMVA)*, pages 153–172.

[136] Ren, P., Gao, Y., Li, Z., Chen, Y., and Watson, B. (2007). IDGraphs: intrusion detection and analysis using stream compositing. *IEEE computer graphics and applications*, 26(2):28–39.

[137] Reza, S. and Ghorbani, A. (2006). Alert Correlation Survey : Framework and Techniques. *Alert Correlation Survey : Framework and Techniques*, pages 1–10.

[138] Riesen, K., Neuhaus, M., and Bunke, H. (2007). Graph Embedding in Vector Spaces by Means of Prototype Selection. *Graph-Based Representations in Pattern Recognition*, (Springer Berlin Heidelber):383–393.

[139] Roesch, M. (1999). Snort: Lightweight Intrusion Detection for Networks. *LISA '99: 13th Systems Administration Conference*, pages 229–238.

[140] Roschke, S., Cheng, F., and Meinel, C. (2010). Using vulnerability information and attack graphs for Intrusion Detection. *2010 6th International Conference on Information Assurance and Security, IAS 2010*, (Ias):68–73.

[141] Roschke, S., Cheng, F., and Meinel, C. (2011). A New Alert Correlation Algorithm Based on Attack Graph. *International Conference on Computational Intelligence in*

Security for Information Systems (CISIS 2011), Torremolinos, Spain, June 2011 A, (June):58–67.

[142] Roschke, S., Cheng, F., Schuppenies, R., and Meinel, C. (2009). Towards unifying vulnerability information for attack graph construction. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5735 LNCS:218–233.

[143] Rowlingson, R., Healing, A., Shittu, R., Matthews, S. G., and Ghanea-Hercock, R. (2013). Visual Analytics in the Cyber Security Operations Centre. *Proceedings of The Information Systems Technology Panel Symposium on Visual Analytics*.

[144] Saad, S. and Traore, I. (2013). Extracting attack scenarios using intrusion semantics. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7743 LNCS:278–292.

[145] Sadoddin, R. and Ghorbani, A. a. (2009). An incremental frequent structure mining framework for real-time alert correlation. *Computers & Security*, 28(3-4):153–173.

[146] Salah, S., Maciá-Fernández, G., and Díaz-Verdejo, J. E. (2013). A model-based survey of alert correlation techniques. *Computer Networks*.

[147] Sangster, B., Connor, T. J. O., Cook, T., Fanelli, R., Dean, E., Adams, W. J., Morrell, C., and Conti, G. (2009a). Toward Instrumenting Network Warfare Competitions to Generate Labeled Datasets. *18th USENIX Security Symposium, 2nd Workshop on Cyber Security Experimentation and Test (CSET)*.

[148] Sangster, B., Cook, T., Fanelli, R., Dean, E., Adams, W. J., Morrell, C., and Conti, G. (2009b). Toward Instrumenting Network Warfare Competitions to Generate Labeled Datasets. *USENIX Security's Workshop on Cyber Security Experimentation and Test (CSET)*.

[149] Schenker, A., Last, M., Bunke, H., and Kandel, A. (2003). Comparison of Distance Measures for Graph-Based Clustering of Documents. *Graph Based Representations in Pattern Recognition*, pages 202–213.

[150] Sheyner, O. M. (2004). Scenario graphs and attack graphs. *Architecture*.

[151] Shiaeles, S. N., Katos, V., Karakos, A. S., and Papadopoulos, B. K. (2012). Real

time DDoS detection using fuzzy estimators. *Computers and Security*, 31(6):782–790.

[152] Shiravi, H., Shiravi, A., and Ghorbani, A. A. (2010a). IDS Alert Visualization and Monitoring through Heuristic Host Selection. *Information and Communications Security*, pages 445–458.

[153] Shiravi, H., Shiravi, A., and Ghorbani, A. A. (2010b). IDS Alert Visualization and Monitoring through Heuristic Host Selection. pages 445–458.

[154] Shiravi, H., Shiravi, A., and Ghorbani, A. a. (2011). A Survey of Visualization Systems for Network Security. *IEEE transactions on visualization and computer graphics*, 1(1):1–19.

[155] Shittu, R. (2013). An Automated Framework for Detecting Unusual Host Behaviour Using Intrusion Detection Logs. Technical report.

[156] Shittu, R., Healing, A., Bloomfield, R., and Muttukrishnan, R. (2012). Visual analytic agent-based framework for intrusion alert analysis. *Proceedings of the 2012 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2012*, pages 201–207.

[157] Shittu, R., Healing, A., Ghanea-hercock, R., and Bloomfield, R. (2014). OutMet : A New Metric for Prioritising Intrusion Alerts using Correlation and Outlier Analysis. *19th IEEE Conference on Local Computer Networks*.

[158] Shittu, R., Healing, A., Ghanea-hercock, R., Bloomfield, R., and Muttukrishnan, R. (2015a). Real-time Graph Clustering for automatically discovering novel attack patterns. *Network and Computer Applications (Under Review)*, pages 1–25.

[159] Shittu, R., Healing, A., Ghanea-Hercock, R., Bloomfield, R., and Rajarajan, M. (2015b). Intrusion alert prioritisation and attack detection using post-correlation analysis. *Computers & Security*, 50:1–15.

[160] Stallings, W. (2007). Network security essentials: applications and standards.

[161] Steven Eckmann, G. V. (2002). STATL: An Attack Language for State-based Intrusion Detection. *Journal of Computer Security*, 10(1):71–103.

[162] Sundaramurthy, S. C., Zomlot, L., and Ou, X. (2011). Practical IDS alert correlation in the face of dynamic threats. *International Conference on Security and Management (SAM'11)*.

[163] Taha, A. E. and Ghaffar, I. A. (2010). Agent Based Correlation Model For Intrusion Detection Alerts. *Learning*, pages 89–94.

[164] Tedesco, G. and Aickelin, U. (2008). Real-time alert correlation with type graphs. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5352 LNCS:173–187.

[165] Thomas, J. and Cook, K. (2005). Illuminating the path: The research and development agenda for visual analytics.

[166] Valdes, A. and Skinner, K. (2001). Probabilistic Alert Correlation. *In Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pages 54–68.

[167] Valeur, F., Vigna, G., Kruegel, C., and Kemmerer, R. (2004). Comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on Dependable and Secure Computing*, 1(3):146–169.

[168] Verizon (2013). Data Breach Investigations Report. Technical report.

[169] Vincent Zhou, C., Leckie, C., and Karunasekera, S. (2009). Decentralized multidimensional alert correlation for collaborative intrusion detection. *Journal of Network and Computer Applications*, 32(5):1106–1123.

[170] Wang, J. (2012). *Advanced attack tree based intrusion detection*. PhD thesis, Loughborough University.

[171] Wang, L., Ghorbani, A., and Li, Y. (2010). Automatic Multi-step Attack Pattern Discovering. *International Journal of Network Security*, 10(2):142–152.

[172] Weber, D. J. (1998). *A Taxonomy of Computer Intrusions*. PhD thesis, Massachusetts Institute of Technology.

[173] Winter, H. (2012). System security assessment using a cyber range. *7th IET International Conference on System Safety, incorporating the Cyber Security Conference 2012*, pages 41–41.

[174] Worlein, M., Meinl, T., Fischer, I., and Philippsen, M. (2005). A Quantitative Comparison of the Subgraph Miners MoFa, gSpan, FFSM, and Gaston. *Knowledge Discovery in Databases: PKDD*, (Springer Berlin Heidelberg):392–403.

[175] Xiao, L., Gerth, J., and Hanrahan, P. (2006). Enhancing Visual Analysis of Network Traffic Using a Knowledge Representation. *2006 IEEE Symposium On Visual Analytics And Technology*, pages 107–114.

[176] Yan, X. (2002). gSpan: graph-based substructure pattern mining. *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, (d):721–724.

[177] Yang, L., Gasior, W., Katipally, R., and Cui, X. (2010). Alerts Analysis and Visualization in Network-based Intrusion Detection Systems. *IEEE Second International Conference on Social Computing*, pages 785–790.

[178] Ying, Z., Fangfang, Z., and Xiaoping, F. A. N. (2013). IDSRadar : a real-time visualization framework. 56(August):1–12.

[179] Yu, J., Reddy, Y. V. R., Selliah, S., Kankanahalli, S., and Reddy, S. (2004). TRINETR : An Intrusion Detection Alert Management System 1.

[180] Zali, Z., Hashemi, M. R., and Saidi, H. (2013). Real-Time Intrusion Detection Alert Correlation and Attack Scenario Extraction Based on the Prerequisite-Consequence Approach.

[181] Zhan Cui, Ian Herwono, P. K. (2013). "Multi-Stage Attack Modelling". *Proceedings of Cyberpatterns 2013 (The Second International Workshop on Cyberpatterns: Unifying Design Patterns with Security, Attack and Forensic Patterns), Abingdon,*.

[182] Zhou, C. V., Leckie, C., and Karunasekera, S. (2010). A survey of coordinated attacks and collaborative intrusion detection. *Computers and Security*, 29(1):124–140.

[183] Zhou, F., Shi, R., Zhao, Y., Huang, Y., and Liang, X. (2013). Netsecradar: A visualization system for network security situational awareness. *Cyberspace Safety and Security*.

[184] Zhuang, X. Z. X., Xiao, D. X. D., Liu, X. L. X., and Zhang, Y. Z. Y. (2008). Applying Data Fusion in Collaborative Alerts Correlation. *2008 International Symposium on Computer Science and Computational Technology*, 2:124–127.

[185] Zomlot, L., Sundaramurthy, S. C., Luo, K., Ou, X., and Rajagopalan, S. R. (2011). Prioritizing intrusion analysis using Dempster-Shafer theory. *Proceedings of the 4th ACM workshop on Security and artificial intelligence - AISec '11*, page 59.