Correlating IDS alerts with system logs by means of a network-centric SIEM solution

Andreas Bråthen

Master's Thesis
Master of Science in Information Security
30 ECTS
Department of Computer Science and Media Technology
Gjøvik University College, 2011

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Correlating IDS alerts with system logs by means of a network-centric SIEM solution

Andreas Bråthen

2011/07/01

Abstract

This thesis concerns the need for a network-centric Security Information and Event Management (SIEM) solution that correlates data based on network topology and traffic flow, and which takes into account the continuous change in such networks. The research question is raised based on the fact that current SIEM solutions are device-centric with minimal understanding of the causal relationship between log events. Furthermore, the used approaches are suboptimal in correlating data collected from scattered security systems (e.g. IDS, firewall), which requires security personnel to analyze larger data sets with potentially high false positive rate, rather than having the incidents validated, prioritized, and presented in a unified view.

We have in this thesis proposed a conceptual model based on a network-centric approach, and performed a case study of this model using Cisco NetFlow. We observe the model through a series of attacks, and analyze whether the model is a more viable approach to deal with incidents in comparison to current approaches, and whether the approach makes it possible to reduce the number of alerts requiring follow-up and in prioritizing incidents more accurately. The study identifies several network characteristics that may influence the practical implementation of such a model and proposes a set of requirements that a network-centric model should fulfill.

Sammendrag

Denne oppgaven omhandler behovet for en nettverkssentrisk Security Information and Event Management (SIEM) løsning som korrelerer data basert på nettverkstoplogi og nettverksflyt, og som tar hensyn til den kontinuerlige endringen i slike nettverk. Forskningsspørsmålet er basert på det faktum at gjeldende SIEM løsninger er enhetssentriske med minimal forståelse av det årsaksmessige forholdet mellom logg-innslag. Videre er gjeldende tilnærminger suboptimale i korreleringen av data som er samlet fra spredte sikkerhetssystemer (f.eks. IDS, brannmur), som krever sikkerhetspersonell til å analysere større datasett med potensielt høy falsk positiv rate, istedenfor å få hendelsene validert, prioritert og presentert i en enhetlig visning.

Vi har i denne oppgaven foreslått en konseptuell modell basert på en nettverkssentrisk tilnærming, og utført en case study av denne modellen ved bruk av Cisco NetFlow. Vi observerer modellen i en rekke angrep, og analyserer hvorvidt modellen er en mer levedyktig tilnærming for å håndtere hendelser i sammenligning med gjeldende tilnærminger, og hvorvidt en slik tilnærming gjør det mulig å redusere antallet alarmer som behøver oppølging og prioritere hendelser mer nøyaktig. Studien identifiserer flere nettverks-karakteristikker som kan påvirke den praktiske implementeringen av en slik modell og det foreslås et sett med krav som en nettverkssentrisk modell bør oppfylle.

Acknowledgements

The idea for this thesis emerged while I was working at the European Organization for Nuclear Research (CERN) a few years ago, and concretized through my Msc studies at Gjøvik University College. Although the study has been interesting and meaningful by itself, it has been so particularly because of all the people that have been involved and contributed to it. I'm astonished by all the time many people have been willing to spend, for which I'm forever thankful.

My supervisor, Slobodan Petrović, deserves a thanks for his great feedback and guidance. He has extended to me numerous of hours of advice and been steadfast in his role as a supervisor.

My fellow students deserve to be thanked for all the good times and companionship we have had during the studies. Without them the studies would not have been as interesting and enjoyable as they have been.

I would like to thank my family and friends for their understanding and support. They have motivated me in times of need, and always encouraged me to look ahead.

I would also like to express my utmost gratitude to my girlfriend, Stine Andresen, for her love, sacrifice and support through my Bsc and Msc studies. She has always believed in me and maintained unwavering faith in my abilities.

There are so many others whom I may have inadvertently left out and I sincerely thank all of them for their help.

Andreas Bråthen, 2011/07/01

Contents

List of Figures

List of Tables

1 Introduction

1.1 Topic Covered by the Thesis

A modern network infrastructure often consists of a multitude of security systems protecting the digital assets of the business. These systems, as well as the end-devices connected to the network, maintain detailed history of activities that have taken place. The history, which is recorded into log files, may be used to detect and investigate security incidents, policy violations, fraudulent activity and operational problems [4]. From a network perspective, correlated log data makes it possible to determine the exact path a packet has taken through the network, how it was treated along that path, and in many cases even the packets legitimacy based on its behaviour when reaching its final destination (albeit several challenges exist [5]).

The field of Security Information and Event Management (SIEM) is based on the idea of collecting log data from network-connected devices, reflecting network activity and/or device-operation [6], and systematically using it to enhance intrusion detection capabilities and investigating security-related incidents. This process consists of multiple steps: preparation) the data is collected, aggregated and normalized (i.e. consolidated); detection) the data is analysed from different views and correlated by putting together different parts of an attack into a complete picture [7]; and reporting) alerts and reports are generated from the detection phase. Because systems in use today are so versatile, it is possible to normalize log data from almost any type of source and correlate data based on own choice.

The correlation process is the most crucial part of SIEM. It is the association of different but related events to provide broader context than a single event can provide [8]. For the correlation model to trigger on a particular pattern, all the necessary data needs to be collected. An accurate model will furthermore only use the absolute subset of data related to a particular pattern - which requires the model to be adaptable to system and network changes. This work discusses the limitations behind the static approach used by SIEM systems today and proposes an accurate and contextual network-centric model that complements the process of a networked Intrusion Detection System (IDS).

1.2 Keywords

Keywords covered by this thesis in accordance to the taxonomy provided by IEEE Computer Society: **3.2.0.**{Data communication, Network-level security and protection}, **3.2.3.**{Network monitoring}, **11.6.5.**{Unauthorized access}.

1.3 Problem Description

IDS' and SIEM systems are expert systems that have limited view of what is going on. An IDS, or sensor, is an expert of network communication that only knows what can be learned from traffic passing through it. When the sensor detects an attack, it has no way of knowing whether the

attack did succeed, whether the attack was blocked by another security system or whether the attack had any impact on the destination at all. A SIEM on the other hand, has a much better idea of what is going on as it interprets events through log data that have taken place on the devices. The problem with SIEMs are exactly the opposite – because they have no understanding of details and do not focus on network data being transmitted, they do not know the root cause behind log entries, the causal relationship between log entries and the attack vector resulting in a log entry.

Because SIEM operates on a higher abstraction layer, it trusts other devices (i.e. sources) to send information (i.e. through log data) that the devices themselves, or the applications generating events, consider relevant. When events are not generated, the SIEM will be unable to detect attacks, as it is neither aware of an activity taking place nor able to reconstruct or fill the gap to correlate successfully [1]. Examples of this could be when the source purposely does not record events or when events occur at a layer below or outside the applications working sphere. The SIEM is then exposed to circumvention where the state of the source is manipulated in such a way that no trace can be found - which can be the case for network traffic (e.g. Denial Of Service (DoS), protocol anomaly, encrypted payload), low level system alterations or even after a device has been compromised.

Traditional SIEM approaches however, do not emphasize on any particular type of logs and may therefore be considered to operate on a flat structure, treating logs from systems, network devices and applications on an equal level. SIEM systems try, however, to integrate some network awareness through the use of vulnerability management, asset database, and network change and configuration control systems (NCCM). Some systems are also supplemented with contextual information such as network environment and threats [7]. Despite of this, SIEMs' remain static systems that do not take into account the continuous changes in a network infrastructure and lack the necessary network details to detect new, or analyse incidents sufficiently. Current solutions are therefore unable to correlate events requiring a holistic view and understanding of network dynamics, including context, topology, packet payload, protocols, session flows and more. We will in this project try to determine what requirements are needed to make a SIEM system that has a holistic view of the network.

1.4 Justification, Motivation and Benefits

Because of the vast amount of data passing through networks and considering the large amount of data generated by devices, detecting intrusions is a significant challenge. From a network perspective, the number of alerts generated by IDS' has shown to overwhelm security personnel [11] making it crucial to reduce the number of alerts and prioritizing the remaining ones. From a device perspective, SIEM systems lack the necessary details that enables security personnel to prevent similar attacks from reoccurring, and from integrating the continuous stream of network information to discover, verify and reduce security-related incidents.

It has been recognized that detecting attacks requires data from various sources such as fire-

[1] In some domains, techniques in the pre-processing phase have been proposed such as *path completion*. Path completion refers to inclusion of important page access records that are missing in the access log due to browser and proxy server caching [9]. Other studies in the same domain have proposed a network monitor system instead [10].

wall, Web server, IDS, end-devices and so on [12, 13]. Furthermore, correlating information found in multiple logs allows IDS' to improve the effectiveness of alerts [13]. By studying alerts generated by IDS' and analysing what log data is relevant for those alerts, we would able to make more educated decisions when dealing with incidents. This would also allow us to have a holistic view of what happened with minimum information. By having complete information, security personnel can conduct more careful analysis and will have more time doing so as false positives and real attacks that are stopped, would never reach the view of the analyst in real time.

1.5 Research Questions

The main research question is '*Can we improve SIEM by making it network centric?*'. To be able to answer the question, we need to study how packets are routed through a network, identify what types of devices are involved in this process and what information we are able to extract from these devices. The packets we are going to look at need to represent a diverse set of intrusions considering the variety of network devices and security systems that may influence the packets. The concept of connection tracking (i.e. combining packets with related characteristics) is central to a network-centric approach, and needs to be studied as well.

In answering our main question, we should address the following sub-questions:

- What network-devices are needed to investigate security incidents?
  - What log data from these devices are relevant?
  - How can log data be analysed and used in a correlation process?
- What methods can be used to perform connection tracking?
- How can intrusion alerts be mapped to connection trackers (i.e. flows)?

1.6 Summary of Contributions

In this thesis we study a network-centric approach to SIEM that differs from today's approach as it focuses on the network aspect instead of the devices. We have proposed a conceptual model for correlating and providing contextual information to log data, based on network traffic flow. We have implemented the model using Cisco NetFlow as the chosen connection tracker, and observed this method in a case study simulating various types of attacks.

We discuss characteristics of the model and the method used in Chapter 6, and present a list of requirements that we believe the model should fulfill in Section 6.6. The requirements outline what we believe is necessary for the model to be used in practice, based on what was observed during the case study. The observations are largely influenced by the method, but contributions reflect the practical implementation of the generic model in relation to the research question.

2 Related Work

This chapter reviews current research in the fields of event correlation, topology discovery and network flow analysis, as well as drawing parallels to context integration to build a foundation for the research.

2.1 Event Correlation

Event correlation is a widely accepted approach to manage the complexity of modern telecommunications and data networks [14]. It addresses the problems of having large volumes of isolated events by producing a succinct overview of security-related activity on the network [15]. It also binds together events coming from one more log files within a context in order to provide a more complete picture of what has happened in a system [16]. This enables security incidents to be efficiently prioritized and reduces the burden on the security analyst that otherwise would become overwhelmed [4]. The complex process of correlation may be viewed as substituting a set of alarms that match a predefined pattern with a new alarm [17].

The authors of [1] refer to intrusion correlation as the process of interpreting, combining and analyzing information from all available sources about a target system activity, for the purpose of intrusion detection and response. According to them, there are two types of intrusion correlation: intrusion event correlation and intrusion alert correlation. The difference is that the former is concerned with correlating neutral events while the latter is concerned with misuse or anomalies. Their relationship is depicted in Figure 1. Our work is primarily concerned with intrusion event correlation as a mean to build network flows to be used in combination with intrusion alerts and contextual data.

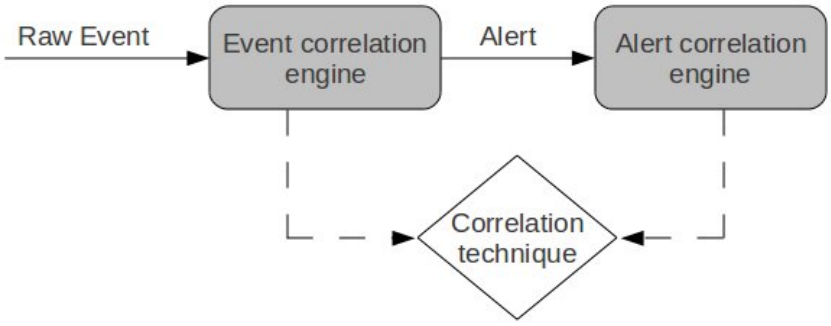

Figure 1: Intrusion correlation flow [1].

Before the event correlation takes place, three processes must be complete [16, 18]: collecting, filtering and normalization. The first process collects data in its raw form. This is followed by the filtering process, whose objective is to reduce the number of events and disregard those

that are not related to an attack [16]. It conducts four types of tasks to achieve its objective [17]: compression, counting, suppression and generalization. Finally, the normalization process translates data into a standardized format, which is understood by all components in the correlation process [15].

The event correlation process is subject to several problems, which are inherited by the previous components, related to the raw data set and the manipulation of it. In particular, it has to deal with issues related to logging (e.g. formats and vast amounts of recorded events) [16] and data that are inadequate (e.g. ambiguous, incomplete, inconsistent) [17]. In this respect, the lack of standardized log formats and an agreed protocol for generating events, may be considered major causes of these problems. Whenever the event correlation engine measures the strength between variables, it has to deal with the aforementioned issues.

2.1.1 Correlation Techniques

The correlation process is described by [16] as follows:

> Events are correlated by assigning relationships between multiple events related directly or indirectly with the system violation. The events related with the attack are generated by different devices and applications, and are written in different log files. The correlation process then, links a series of events in order to recreate the attack sequence. [**Ed.** Depending on the outcome of the correlation, an alert may be generated. The alert can be assigned a priority value according to the severity, impact or probability.]

The correlation engine uses rules to interpret incoming events. It is problematic to construct and to maintain correlation rules, as it requires continuous effort to identify problem patterns, which is time-consuming and error-prone [18]. The correlation techniques used by the correlation engine may fall into the following four categories [1, 19]:

Rule-based correlation. The relationships between alerts are specified in rules, which stipulate pre- and post-conditions that need to occur for a correlation to take place. The correlation method is based on predefined sequences of events that relate to known patterns and behaviors, which defines an attack [16].

Scenario-based correlation. Causality relationships between alerts are specified in terms of scenarios. A successful correlation (i.e. match) occurs when a combination of alerts form a predefined attack scenario.

Statistical correlation. The statistical relationship between alerts fall within a predefined threshold (i.e. statistically related). The threshold value is based on estimates for what is known good or known bad behaviour.

Temporal correlation. Correlation takes place according to the alerts or events temporal relationship (i.e. based on time-series).

Another type of correlation technique involves doing an impact analysis of the attacked system [20]. The idea of impact analysis is to determine what impact the threat has on the system in question. This can be done with the following two correlation methods: local correlation) The

impact of the attack is verified by a local agent running on the victim, which checks whether the attack succeeded or not; and Operating System (OS) correlation) The attack class is compared to the types of services running on the host (including OS), which deems the attack harmless if the host is not exposed to the particular attack class.

2.1.2 Reducing the Correlation Data Set

In an attempt to reduce the volume of logs relevant to an incident and improving the correlation process, [21] studied how distinct types of attacks were related to various types of log files. The study showed that three of the categories, Denial of Service (DoS), User-to-Root (U2R) and Remote-to-Local (R2L), constituting 44 attacks, had large similarities in terms of log traces and where some attacks could be reduced to as little as 5 log sources from the initial 15. Accordingly, by using this log-to-alert mapping they were able to improve IDS accuracy and effectiveness by correlating log data.

In a similar study [13], the researchers looked at the relationship between attacks and log types using a top-down approach. The idea was to study some known attacks to infer which logs would contain traces of them. They concluded that some attack classes had common behavior and that it was possible to identify logs that would be more likely to store useful information related to particular attack classes. The attacks they studied were categorized into two classes: Remote-to-User (R2U) and DoS. Out of the 15 attacks, 8 belonged to the former class while the remaining to the latter. When studying these classes, the researchers observed that the two most important logs were syslog and NetFlow [1]. Accordingly, correlation based on log content could improve IDS performance.

In relation to our study, these results are interesting from two aspects in particular. First, they support our assertion that log correlation used together with IDS may be advantageous in terms of reducing the number of alerts requiring follow-up and in investigating intrusions. Then, both studies showed high dependence on logs originating from network devices such as NetFlow and routing information. In the first study, 38 of the 44 attacks had traces found in NetFlow. In the second study, the figure was 12 out of 15 attacks. This supports our assertion that network flow and topology information have some importance in detecting or investigating attacks.

2.1.3 Context and Situational Awareness

To further improve the quality of the correlation techniques, several researchers have proposed to integrate additional information into the process [2]. In [22], the researchers discern that the wealth of information available to the security analyst may have the potential to contribute in detecting incidents and gaining confidence in the credibility of incidents' alarms. They propose a framework where alerts are combined with vulnerabilities, target topology and ranking alerts based on the interest of the asset owner. This is similar to a technique known as vulnerability correlation [23], where data from vulnerability scanners are compared to the observed alert. In [24], the researchers propose to integrate system monitoring or vulnerability scanning tools in order to increase the confidence in alerts. There is also a technique called susceptibility cor-

[1] NetFlow is a Cisco developed network protocol to report IP traffic data, which is the de-facto-standard for monitoring traffic flow in a network.

[2] This is also referred to as the process of *enriching data* in SIEM context.

relation [25], where the probability of an asset's exposure is calculated by using all available information about that asset, such as what services are running, what ports open, and the type of OS used on the machine.

In study [13], the researchers correlate log data based on traces after the Yara-virus and complement the correlation process by using IDS. Their idea is that correlating heterogeneous logs while simultaneously enabling IDS to identify attacks makes it possible to reduce the number of false positives and validating whether an attack has taken place. A shortcoming with their study is that they study only a single virus, which is known to leave distinct log traces and which uses methods easily detectable by an IDS.

In study [26], the researches discuss the combination of system logs and IDS alerts from a reverse perspective. First, alerts from IDS' are correlated, and then system events are integrated into the process. They state, as with [13], that the support of more detailed and precise information from event logs enable alert correlation from IDS to achieve higher accuracy. The system information being integrated in this case is known as *OS-level dependency tracking*, which is a method to track process forks [3] and file operations from event logs based on specific objects. The conclusion of the study is that the discussed integration greatly improves the correctness of the correlation process and in making hypotheses about possible missed attacks.

A similar study to [26] is conducted by [27] based on OS-level dependency tracking with IDS alert correlation using an event-processing engine called Coral8. The researchers make two statements when describing their approach. First, most attacks have operations on specific OS-level objects. Secondly, if an attack prepares for another attack, the later attack's corresponding operations would be dependent on the earlier ones. The study concludes that the technique can significantly reduce false correlations.

2.2 Determining Network Topology

It is well known that knowing the up-to-date physical topology of an Internet Protocol (IP) network is crucial to a number of critical network management tasks, including reactive and proactive resource management, event correlation, and root-cause analysis [28, 29]. Furthermore, knowledge of element interconnections is essential to filter out secondary alarm signals and correlate primary alarms to pinpoint the original source of failure in the network [28]. By having the complete topology of a network, it is possible to determine how packets travelled across the network and through which devices at what time. In a small and static environment, creating a topology map is more of an administrative issue. In modern networks on the other hand, which are complex and dynamic by nature, it poses several challenges.

Inferring the network layer topology (i.e. Open Systems Interconnection (OSI) layer-3) is relatively easy since routers are aware of their immediate layer-3 neighbors as well as attached subnets, which are published through their SNMP Management Information Base (MIB) [28]. This information is sufficient to determine layer-3 topology, but fails to capture the complex interconnections of the Ethernet LANs that underlie the logical links of layer-3. Unfortunately, layer-2 devices (e.g. switches, bridges and hubs) do not provide similar information of their immediate layer-2 neighbors, which complicate the discovery of the physical network topology.

[3] The act of initiating a new process (i.e. child) from within a running process (i.e. parent).

According to [28], any practical solution for discovering physical IP topology needs to deal with three fundamental difficulties: limited local information) Because of the difficulties of inferring a device's physical neighbors (layer-2 devices in particular), an algorithm should make minimal assumptions and utilize information stored locally; transparency of elements across protocol layers) Because layer-2 devices are completely transparent to layer-3 routers directing traffic between subnets, the algorithm should establish interconnections between network elements operating at different layers of the OSI model; and heterogeneity of network elements) Because a network is often comprised of different vendors, the algorithm should be able to gather topology information correctly from heterogeneous sources.

Another difficulty in this respect is determining the types of devices that packets are flowing through in the topology. Albeit existing techniques to gain information about hosts can be used (e.g. reconnaissance techniques performed by scanning tools), these techniques encounter a problem similar to the techniques utilized to perform topology discovery – increasing the load on the network and hosts when generating probing traffic. Although we will not be addressing issues related to discovering and building a network topology in this study, we will look into the aspect of determining host types when the topology is known by using log data. For a more comprehensive overview of current network discovery techniques and their limitations, look at survey [5].

2.2.1 Establishing Network Flow

In any large-scale network, event and alarm-producing systems are distributed across the entire network, comprising some (and possibly all) of the computing and infrastructure systems in the network [30]. These systems produce large amounts of information that easily overwhelm a security analyst as well as log management systems. As stated, two major problems having ramifications for event correlation are related to the volume of data and the lack of standardized log formats [16].

In terms of event sources, we can divide them into three categories: 1) Those events that influence how packets are traversed along the topology (i.e. generated by devices that manipulate the layer-2 and layer-3 content of network packets); 2) Those events that combined reflect the attack process; and 3) those events that provide context and impact analysis of the attack. Another view of event sources is the classification of them into distinct type of domains [1] as depicted in Figure 2. In answering our research question, we primarily need to look at the network domain of event sources.

In terms of log formats and network flow technologies, Cisco's NetFlow is the de-facto standard used by network manufacturers today. The newest version 9 was published in 2004 [31] and may be used in equipment such as switches, routers and firewalls. A flow in Cisco's NetFlow is structured into a seven tuples format: source IP, destination IP, source port, destination port, IP protocol, ingress interface and IP type of service. NetFlow is becoming superseded by a new format called IP Flow Information Export (IPFIX) [32]. Whereas NetFlow was developed by Cisco, IPFIX is developed by the Internet Engineering Task Force (IETF) and is expected to become the new de-facto standard. Both NetFlow and IPFIX consider a flow to be a set of packets being sent through a device within a specific timeslot that shares a number of characteristics.

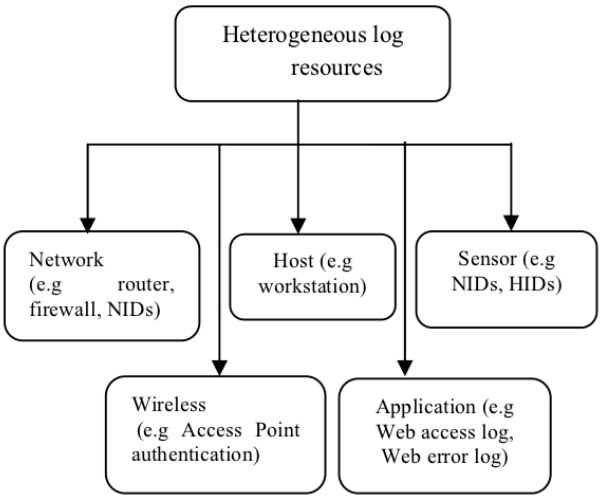

Figure 2: A domain perspective from heterogeneous event sources.

Flows are maintained in the cache of the monitoring device, and each set of characteristics not seen earlier, are inserted into this cache. Each entry in the cache is given an unique flow ID. Flows are considered complete, and the flow entry is flushed from cache (or *exported*, which is the terminology used by RFC standards), when one of the following criteria has been met [31]:

1. TCP flags indicate a completed flow (FIN or RST).
2. X seconds after the last packet has been seen, matching a specific flow ID. Time is configurable.
3. X minutes after the flow has been created. This is to avoid staleness. Time is configurable.
4. When the exporter encounter internal constraints, such as when the memory is full or when counters wrap around, causing the flow cache to rotate.

Flows are said to play a vital role in network security to detect DoS attacks, network-propagating worms, and other undesirable network events [33, 34]. There exist several commercial products, and Free and Open-Source Software (FOSS) that revolve around Cisco NetFlow, such as Cisco CS-Mars, IBM Aurora, NetQoS, Arbor Networks. There also exist alternatives to Cisco's NetFlow such as HP's sFlow or Juniper's jFlow. The latter two are, however, flow sampling technologies that by specification are problematic when dealing with the research question raised in this thesis. Sampling is described as follows in RFC 3917 [35]:

> Sampling describes the systematic or random selection of a subset of elements (the sample) out of a set of elements (the parent population). Usually the purpose of applying sampling techniques is to estimate a parameter of the parent population by using only the elements of the subset. Sampling techniques can be applied for instance to select a subset of packets out of all packets of a flow or to select a subset of flows out of all flows on a link.

3 Attack Classifications and Detection Capabilities

This chapter reviews current research on attack taxonomies, and applies the *Igure and Williams* taxonomy [3] with particular emphasis on traceability and detection capabilities.

3.1 Attack Classifications

A taxonomy is formally defined as the categorization of the relationships between the characteristics of objects [3]. A taxonomy of attack types in particular, is considered to contribute in dealing with attacks better [36]. The purpose of attack taxonomies as stated by [36], is to provide a useful and consistent mean of interpreting attacks, which allows attack-learning to be shared between organizations. An attack taxonomy also extends the capability to include all properties of an attack into account, including yet unknown attack classes. A taxonomy of computer system attacks has many useful applications in the security industry, such as:

Conducting security reviews and assessments: An attack taxonomy may be beneficial in the process of reviewing security posture and assessing relevant data.

Measuring detection capabilities: An attack taxonomy may contribute in estimating a security controls' detection capabilities and in determining strong and weak characteristics of such controls.

Improving detection capabilities: An attack taxonomy may be necessary to identify which attack classes should be tested against a security control and to understand which characteristics of the attack a security control needs to be improved on.

Evaluating the impact of an attack on a system or service: An attack taxonomy may enable an organization to estimate risk and probability of a particular type of attack class and prioritizing resources accordingly.

Avoiding common design flaws in product development: An attack taxonomy may enable software developers to enhance their effort on areas where a product should be particularly strong and robust.

Attack taxonomies are used in this study to understand what types of attacks are most relevant to the research, and in selecting wide and non-related attack types. This further allows us to gain better understanding of what types of components the network should consist of, which is driven by the components relevance in a particular attack class. This helps us determine what detection capabilities are needed, what log sources may be relevant to investigate attacks and what signatures are needed to detect them.

3.1.1 Classification Overview

The work on classifying attacks is largely an on-going process which has been worked on for several decades [37]. In study [36], the authors summarize some of the requirements that have been used to create attack taxonomies in the past. The study lists 8 characteristics in particular, which includes: completeness) account for all possible attacks; determinism) the procedure of classifying must be well defined; mutually exclusive) each attack must fall into one category only; terminology compliance) existing security terminology should be used; unambiguous) each category should be clearly defined; and usefulnes) the taxonomy must be able to be used in the security industry.

A survey on attack taxonomies [3] highlights some of the ambiguity with today's taxonomies and the difficulty of fulfilling all the characteristics summarized by [36]. In the survey, the authors have studied different attempts to create classifications in the period from 1974 until 2006. The problem with taxonomies as stated, is that attack classifications may be based on different goals, covering different aspects of systems or environments, and providing dimensions that are not directly transparent between subsequent attempts to classify. This is also supported by [38], which reviews 25 approaches to establish taxonomies and identifies the differences and relationships between them. The study shows that in terms of attribute description, dimensions and objective, most of the suggested classifications are semantically different. It was further shown that the studied classifications do not satisfy all the classification-principles as discussed by [36, 39].

In study [2], the researchers suggest a taxonomy based on a set of features that arguably can be found in all types of attacks. Every attack has a value in each of the 14 features. The taxonomy, which is depicted as a hierarchy in Figure 3, shows clearly that attacks have several characteristics that each can influence the capabilities of intrusion detection controls. Moreover, an attack that changes some of its features may be considered a different type of attack, which does not fall into the same classification. Hence, performing a case study that consists of a wide set of attack types with different features and characteristics, may turn out to be a bad approach when generalizing the results from the study. This is especially true when taking into account that a single attack may utilize multiple attack vectors with different features, and considering that some attacks may use arbitrary obfuscation techniques to outsmart a security control.

The 14 features as outlined in [2], are used in the same study to build an attack-severity-level-scheme that attempts to classify attacks based on one of the features in particular – the attack objective. Accordingly, objective achievement is the most influential factor in an attack. The researchers conclude that it is rational to use a 5 level classification based on the attack objective in order to evaluate the severity of an attack. The level system rates attack from the objective of gaining root privileges (highest level) down to the objective of retrieving information about the system to perform a more targeted attack (lowest level). The system is not concerned about the methods used to perform the attack or the means to do so, but rather evaluates the outcome of the attack if successful. The system is helpful in prioritizing what attacks to deal with first, which may be beneficial in the impact evaluation taking place after the correlation process is completed. It cannot, however, be related to detection controls as it focuses on the ends of the attack: source as in the desired goal and victim as the outcome of the attack. Our study needs to focus on the network aspects.

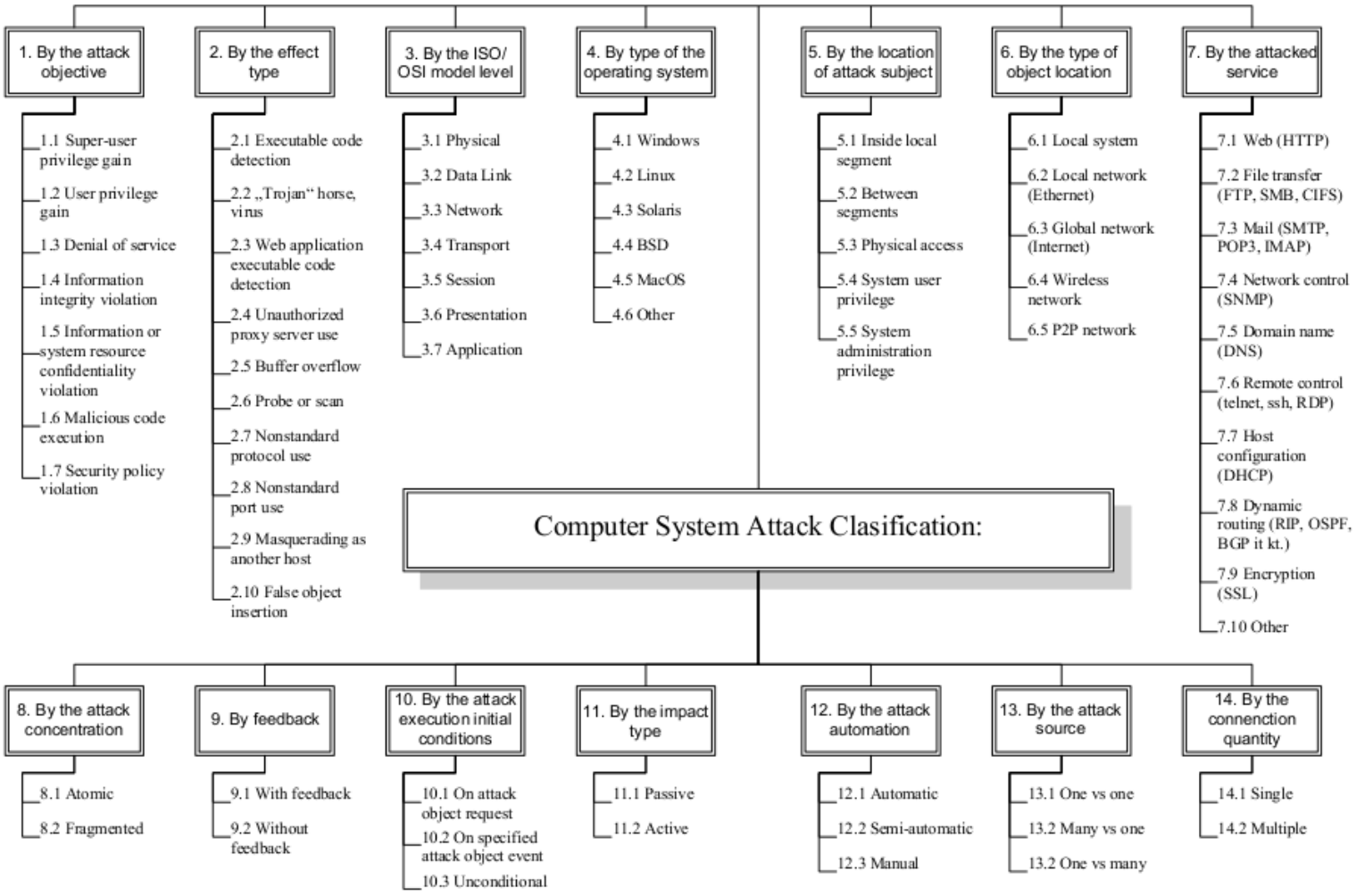

Figure 3: Hierarchical level of features found in attacks [2].

In terms of IDS/IPS [1] testing, the testing platforms rarely conform to the entire set of characteristics discussed earlier. The KDD cup 99, which had been used extensively in the academia for this purpose, is one example of this. The KDD cup 99 divides 4.900.000 single connection vectors where each contains 41 features, is labeled as either normal or attack with exactly one specific attack type [40]. The attacks fall into either of the following categories: DoS, User to Root (U2R), Remote to local (R2L) or Probing attack. There are several limitations behind using this data set in IDS/IPS testing [40, 41], and the use of it in the network intrusion detection domain has been highly discouraged [42].

The perhaps most comprehensive testing platform today is the one used by NSSlabs. It is well-acknowledged as it includes a large set of attacks. Their tests in 2010, using their *methodology v6.0* for instance [43, 44], include 1179 exploits composed of several different classifications. The tests do not argue for the choice of classifications, but are nevertheless used as a standard mean for comparing IDS/IPS', which includes the following categories: Threat vectors (who initiated the attack), Target type (e.g. Web-server, JavaScript), coverage by result (e.g. code injection, buffer overflow), coverage by vendor (e.g. Adobe), types of fragmentation (e.g. packet, stream), types of obfuscation (e.g. URL, HTML) and evasion-techniques (e.g. FTP). The tests also include performance measurements and claims to be using real-world simulated traffic.

A weakness with such industry-based testing platforms, is that they do not measure detection capabilities or accuracy in evaluating events, but whether a particular exploit was detected or not (i.e. the hit rate). An IDS vendor with significant amount of resources to develop signatures, or an IDS with a more aggressive default enabled signature-set, will be beneficial in such tests. An IDS focusing on a particular service or OSI-layer may be superior on its area, but weaker overall. If the attacks were less generic, such as targeting specific features of an attack or branches based on a taxonomy hierarchy, the test would be better suited to measure a security controls' detection capabilities as stated initially.

3.2 The Igure and Williams Taxonomy

In a study by [3], it was proposed a set of basic properties a taxonomy should be based on. The primary focus is that the taxonomy should be application- or system-specific, that the taxonomy should either be hierarchical or linear depending on its use, and that the classes do not need to be mutually exclusive. This taxonomy fits well into the characteristics that have been discussed earlier. The taxonomy is called attack-oriented [38] as it focuses on the goal and the outcome of an attack, and not on the actual methodology to perform the attack. It is also very flexible as it supports several dimensions where each dimension can be used to add a finer (i.e. more granular) specification of an attack.

Based on the mentioned properties, the taxonomy is structured into four initial layers:

Level 1 - attack impact: The immediate impact of an attack in terms of the basic security property it violates. Immediate means in this case the first security property being violated.

Level 2 - system specific attack types: The classes of impact from an attack that fall under a

[1] IPS is often considered the equivalent of an IDS with added blocking capabilities.

given security property.

Level 3 - system components (attack targets): The specific system component being targeted by an attack.

Level 4 - system features (source of vulnerability): The specific feature in the system component being targeted (i.e. exploited).

A challenge of using this taxonomy is deciding how granular one should be for each attack. By using a top-down approach one could narrow down to the very core of an attack, or one could stop at a higher level that makes more sense when testing completeness. This ensures that one selects attacks from a wide perspective that are more suitable for making generalizations. A potential problem with this kind of taxonomy is that complete attacks may not be isolated. Attack vectors often have dependencies and run sequentially, which means that a single attack may fall into multiple categories thus complicating the goal of generalizing.

A problem with taxonomies based on basic security properties as discussed by [45], is that it is not obvious what should be considered the immediate result of an exploit. The authors exemplify this with a password-guessing attempt. A password-guessing attempt may reveal the password of the user account, but the intentions behind the guessing of such an account would be up to the attacker. The attacker could continue to breach any of the three basic security properties, thus be the primary objective behind the actions. However, as the first consequence is that the attacker learns the password, the attack is considered a breach of confidentiality.

A similar pragmatic example is also given in the same study [45]. The researchers explain that by planting a trojan horse on an infected computer, the attacker has succeeded in executing unanticipated processes. The intensions behind planting such a backdoor may be to violate any of the three basic security properties. Using a simplified approach again, the trojan has primarily inserted data which may be considered an integrity breach. The attack vectors used to plant the trojan, which is concealed from the user, is a result of deliberately tricking the user into installing what is believed to be a trustworthy application.

Another point that the taxonomy does not discuss is the origin of the attack or exploit. The attacker may target a specific host to attempt to breach any of the three security properties, or the client may accidentally perform actions that may be considered a successful attack without the attacker knowing. The latter may be explained by the client becoming infected or attacked based on its behavior. Visiting a malicious Website for instance, may lead to the client being infected without the attacker directly targeting that client in particular. Regardless of who originates the attack, the intentions behind the attack may just as well be the same.

The taxonomy discussed here will be used in this thesis, and is depicted in Figure 4. The Figure shows the first and second layer mentioned earlier, which is considered adequate depth for the case study. We will discuss all of the layers in the sections that follow.

To adjust to the problems mentioned earlier, we will extend the use of the basic security principles as suggested by Meadows [46]. First and foremost, the use of the word *authorization* may be considered approval. Approval is the acceptance that some action may be performed within the boundaries of that approval. The term *confidentiality* will not only include unauthorized ac-

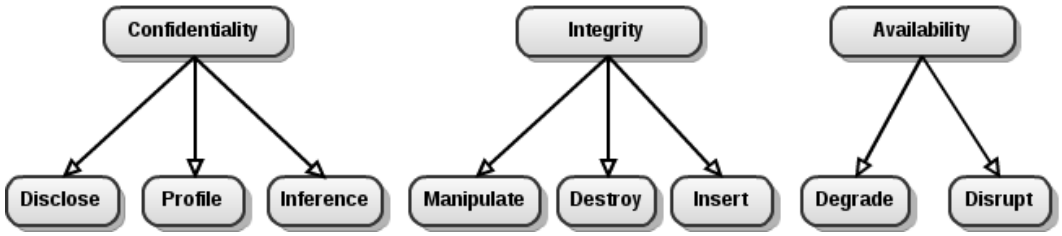

Figure 4: Attacks classified by intent based on the Igure and Williams taxonomy [3].

cess but also unauthorized use of a system. This may involve reconnaissance with the intent to gather information that is not considered public information. The term *integrity* will denote the change of a system without the authorization to do so. Approval may in this case be considered the act of a user deliberately installing an application with clear intentions. Breach of *availability* is the intent to influence a service in such a way that the quality of that service is significantly reduced or access to the service itself denied.

3.2.1 Confidentiality

The goal of confidentiality is to protect information from being disclosed or revealed to entities not authorized to have that information [47]. Protection is ensured by using security controls that authorizes access to data according to some criteria, and/or isolates the service from the data source in such a way that confidentiality may not be breached through the service alone. An attacker's goal may then be accomplished by direct or indirect means. Manipulating databases or encodings to retrieve data are examples of direct attacks, gaining shell access or increasing privileges through the service to perform additional attacks is a secondary type of attack. This thesis is only concerned with direct attacks, as the attacker may perform any type of attack once security has been breached [3].

Confidentiality has in this thesis three level two branches. Disclose is about gaining access to information being protected by security controls, by circumventing it in such a way that its integrity has not been breached. Typical attacks may be SQL-injections, directory traversals, manipulating encodings, semantic URL attacks and replay attacks. This branch of attacks is primarily on the application level and may require application-aware signatures.

The second branch, profiling (i.e. reconnaissance), is concerned with the act of collecting information about a target before attacking it. Profiling is often the legitimate use of a service with the sole purpose of collecting as much information about that service as possible. The actions do not need to be towards a particular service, it may also include port-scans and sweeps, vulnerability scanning and more. Profiling causes no harm to the systems [2], but often generates a lot of noise that may be detected by an IDS.

The third branch, inference, is concerned with the legitimate use of services to extract data that combined may form a new meaning [48]. For instance, imagine a database allowing SQL-views to get the average sum of salary per department but not per employee. If it was possible to create arbitrary queries with a reduced set of people, one could over a set of queries be able

[2]Unless discussing SCADA or PLC-systems, which are not considered in this study.

to extract information about a particular employee. Another class of attacks we define into this branch is brute-force. A brute-force attack is about repeatedly trying various combinations to find the correct key. The inference class may be very noisy due to it's nature of repeated attempts.

3.2.2 Integrity

Integrity refers to the trustworthiness of data or resources, and is usually phrased in terms of preventing improper or unauthorized change [48]. Integrity is verified by creating mathematical hashes of data and comparing them before and after an event has taken place. This makes sense when transferring data over a network to ensure that data is complete and not altered, but is impossible to impose in services where data is expected to change and the change cannot be predicted. An attacker's goal is then to alter data in such services either by circumventing some security control, or claiming an authorized identity, which is allowed to perform direct alterations.

We have divided integrity into three branches in this thesis. Manipulating refers to the actual change of data on the system. This may include services, file systems, registry or even directly into memory. Manipulating attacks may refer to legitimate use of a service in a non-authorized way, or it may refer to attacks that change the processes and action-flows in such a way that unexpected outcome occurs. Example of the latter category are buffer overflows and in general the use of exploits that takes advantage of improper input validation. Manipulating attacks in terms of buffer overflows are typically seen as strings of repeated characters, which need to be provided in order to hit the return address on a stack.

The second branch is about destroying, removing, scrambling or even hiding data in such a way that it is not present when expected. As with manipulation attacks, it may involve data located on the application level or directly on the storage medium itself. Examples of attacks in this branch may be typical database queries such as *drop tables*, or it may be direct shell commands performing deletion commands.

The third branch is about insertion attacks. It is not an attack in the typical sense, but is about inserting data into a data-storage with the purpose of either degrading the overall value of other data or adding unauthorized scripts and programs into the execution flow. A trojan is an example of the latter as the program code is inserted into the system, without approval, and which circumvents the human "security control" by hiding in, or masquerading as an authorized application.

3.2.3 Availability

Availability refers to the ability to use information or resources as desired [48]. Availability is basically both about being able to access a particular resource and being able to use it in the way it is intended. Large deviations in performance may lead to breach of availability. Security controls protecting availability are often not primarily about security aspects, but are service-management issues in the sense that traffic needs to be sent where expected, and the necessary resources need to be allocated for each connection. Breach of availability may often come from legitimate use of a service where the allocated resources do not meet the popularity (i.e. amount of connection attempts) of that service.

We have divided availability attacks into two branches, which follow the categorization for

DDoS attacks in particular [49]. Degrade attacks are about consuming some portion of the consumers' resource significantly, thus seriously degrading the service to legitimate users [49]. The goal for the attacker could be for instance, to cause the victim to loose some percentage of its customers due to them not getting access, or that the victim is falsely believed that additional expensive investment in resources is required to cope with the perceived low performance. Degrading attacks may target the full spectrum of the OSI-model [50]. DoS of applications in particular, is about tying the resources in terms of CPU-cycles or memory allocation to false requests. These requests may be large (i.e. network-intensive) queries for data, or requests that target system components requiring intensive processing.

The second branch is about disrupting the service in such a way that the service is denied to a majority of legitimate users [49]. As with degradation attacks, this may target the full spectrum of the OSI-model. Attacks that are disruptive are rarely as sophisticated as degrade attacks as they are about brute-force and flooding the victim with data that cannot be processed within reasonable time. Detecting attacks that are disruptive is then simple because they create a lot of noise on the network, which has significantly higher bandwidth utilization than the normal baseline. The challenge with DDoS attacks is stopping the large amounts of seemingly legitimate traffic from many sources, but this is not discussed in this thesis.

4 Network Event Correlation Model

This chapter discusses the underlying principles for the proposed network event correlation model, and describes the model in a multi-step approach from alert detection to impact analysis.

4.1 Network Routing

A network may be considered a number of pathways for communication between two or more hosts [51]. Network event correlation as discussed in thesis, is in many ways similar to graph-theory and the study of the shortest path problem [1]. The shortest path problem is the problem of finding a path between two vertices (or nodes) that constitutes the shortest distance between those vertices in terms of cost. The problem is surprisingly viable in many types of domains, where the domain may be constructed as a graph and where cost may be assigned to the edges (e.g. travel time over the edge), or assigned to the vertices (e.g. cost of visiting that vertex).

The shortest path problem is often characterised as a problem where the graph is completely known, and where all costs in that graph are set. In telecommunication, the problem is present when one wants to determine how packets can be most efficiently routed between devices (i.e. vertices) in an organizational network. A complicating factor in telecommunication however, is that the graph is in continuous change because of several external events, such as:

- The cost may be influenced by several criteria (e.g. bandwidth, jitter, latency, QoS).
- The cost between vertices changes according to external strategic factors (e.g. preferred route, load balancing [2]).
- The complete graph is not always known (e.g. crossing network domains, continuous reconfiguration and updates).
- There may co-exist multiple algorithms for calculating optimal path in the same network.
- The graph may expand or contract ad-hoc (e.g. link failure, equipment replacement, maintenance).

Telecommunication-networks address the shortest path problem by using routing protocols, whose purpose is to consider the aforementioned events and select the optimal route based on various cost-metrics (e.g. links, router status) [53]. Because routing protocols have different goals, distinct characteristics, and there are various cost-metrics that may be used in "solving" the problem most efficiently, different routing protocols are used in different types of environments.

[1] Also referred to as the Single-Pair-Shortest-Path (SPSP)-problem.
[2] Also known as the Equal Cost Multi-Path (ECMP) rule in RFC 2328 [52].

Within a single network domain (i.e. intra-domain), Interior Gateway Protocols (IGP) are used, such as: Interior Gateway Routing Protocol (IGRP), Enhanced IGRP (EIGRP), Open Shortest Path First (OSPF), Routing Information Protocol (RIP) and Intermediate System to Intermediate System (IS-IS).

All of the mentioned protocols use algorithms that consider a different set of network characteristics in their calculations, which makes them difficult to compare. One protocol may for instance be less computationally complex and have smaller message overhead, making it more suitable in smaller environments. Interior Gateway Protocols are divided into two classifications. Link-state protocols, such as OSPF and IS-IS, often uses the Dijkstra algorithm to calculate paths, where each node constructs a complete map of the network and calculate paths independently. Distance-vector protocols, such as RIP and IGRP, use the Bellman-Ford algorithm, where each node informs its neighbors periodically, in addition to when changes occur on the network.

Routing protocols have to relate to cost in the sense that cost is a vector of parameters where each may be weighted differently. EIGRP for instance, considers the following six tuples: bandwidth, path load, path delay, reliability, Maximum Transmission Unit (MTU) and hop count. Other more lightweight protocols such as RIP, relate to cost as a single value, the hop count between path and destination. A network using a distance-vector may then be simpler to deal with as it is more predictable in a network with less topology changes, but it may be problematic when used in large-scale networks [53].

Routing protocols are relevant in our thesis because it is the calculation that determines how packets are routed throughout a network. While the goal in the shortest path problem is to determine the optimal route between two vertices, our goal is to identify what exact route was used and what devices mediated along that path. Furthermore, traffic passes through barriers that treat packets differently, according to the barriers' decision criteria and rules. It is necessary to collect decisions through log data to determine what actions were taken on those packets and to identify relevant logs that enable the correlation model to raise educated and precise alerts.

4.2 Protection Domains

A physical network topology refers to the characterization of the physical connectivity relationships that exist among entities in a communication network [28]. Said differently, it means how different network devices such as switches and routers are interconnected, and how hosts are connected to them. Routing protocols deal with the connectivity complexity by dividing the shortest path problem into multiple smaller problems (i.e. domains), where each node calculates the optimal path according to its learned knowledge or in cooperation with other nodes. The understanding of this domain-based approach and in particular why and how domains impose different access controls, is necessary to evaluate the impact on targets as argued in this thesis.

A modern network infrastructure may be viewed as the concept of protection domains, where each domain contains a set of resources that are only available to those that are allowed to enter that domain. The access to a domain is protected by a barrier (i.e. a guard) that authorizes or blocks distinct traffic characteristics from entering or leaving. The purpose of protection domains, sharing similar concepts with network segmentation, is to restrict access to resources within areas of a network that only authorized hosts, sending authorized packets, should be allowed to access.

Protection domains are influenced by level-2 and level-3 of the OSI-model, shown as a logical representation of how a network is isolated in terms of user-interactions and resource-access.

Protection domains may be viewed as access to information from an attacker's perspective as shown in Table 1 [54] [3].

	Access information about NI device of a network	Access to computers of the targeted network	Access to NI devices of the targeted network
A remote computer	Yes	Maybe	Usually no
A computer in the targeted network	Yes	Yes	Usually yes
An NI device in the targeted network	Yes	Yes	Yes

Table 1: Access to information from an attacker's perspective.

The table shows the type of information that an attacker can access, or learn, depending on the attacker's location in relation to a given network infrastructure. It shows, for instance, that if an attacker only has a remote computer, that attacker can only access some information about the network infrastructure in most situations. If the attacker however, has access to a device located within the network infrastructure, that attacker would be able to directly access other resources on that network. The distinction of access to resources, depending on how much access the attacker has, is important as there are guards protecting the access to resources, which influences the correlation process and impact evaluation, based on intrusion alerts.

A modern network infrastructure often follows a hierarchical solution, or segmentation practice, as depicted in Figure 5 [48, 55]. The protection barriers or guards discussed until now are synonymous with the protection walls seen in the figure.

Figure 5 consists of four protection domains, which are located in front, behind or between the protection barriers. The first protection domain is located in front of the outer protection barrier, which is basically the representation of the Internet. Exterior Gateway Protocols (EGP), which is not considered in this study, resides here. The distinction is made because of ingress- and egress-filtering performed by each barrier. Behind the outer protection barrier, three intra-domains may be found where the aforementioned IGPs reside. One protection domain between the outer, inner and demilitarized zone (DMZ) [4] protection barrier, another behind the DMZ protection barrier, and the last behind the inner protection barrier. The latter domain most often consists of numerous additional domains, which are not represented here as it changes significantly between organizations [48].

The figure shows a typical hierarchical model (c.f. Table 1), where the inner protection domain inherits the weakest protection barrier that allows traffic to be reached from the previous protection domain [5]. This granular model is beneficial as it requires the attacker to breach multiple barriers before reaching the tradionally most sacred data owned by the organization is found.

[3] NI is the abbreviation for Network Infrastructure.

[4] A DMZ is a subnetwork within an organization that is considered less trustworthy than internal networks. Resources exposed to external networks normally reside here.

[5] For ingress traffic, and vice versa for egress traffic.

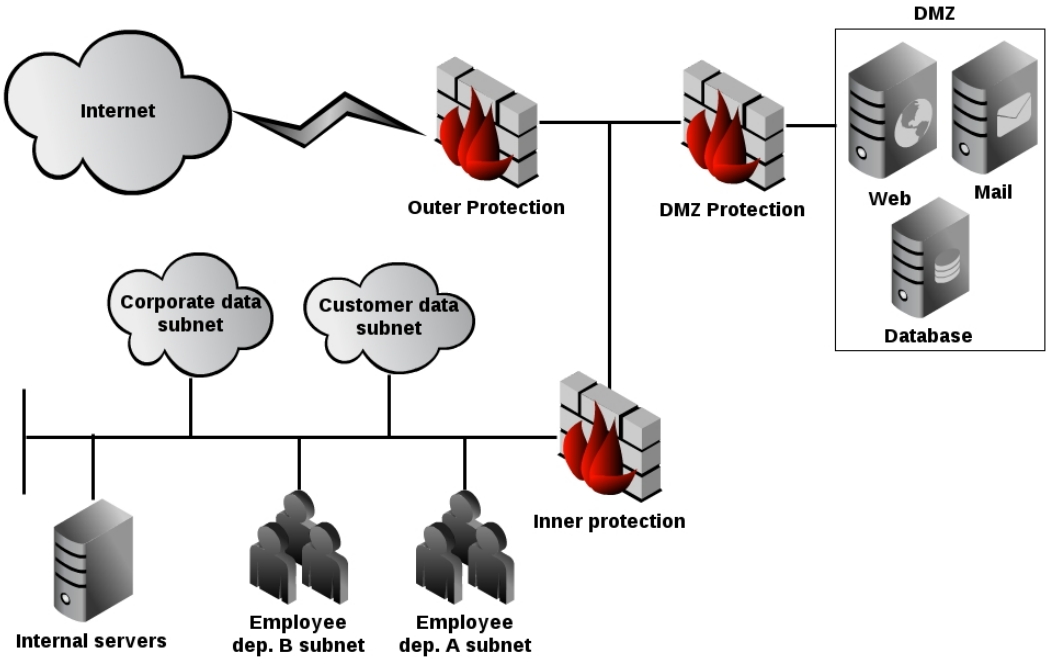

Figure 5: Composition of a traditional network topology.

A linear model ensures that the first guard blocks network-traffic from entering the next domain, and then the following guard blocks traffic from entering the next and so on.

The linear model is as stated, beneficial because it supports a multi-security scheme (i.e. defense-in-depth). It is strong for protecting the organization, but may be less supportive when investigating incidents such as when an internal host has been compromised. Imagine having an IDS in front of the outer protection barrier and assuming that an attacker successfully compromised an internal resource located on the DMZ network. Investigating the incident requires the victim be identified, that some state information from it can be obtained, and sometimes determining whether the attack was blocked or not by the DMZ protection gate. An IDS with no knowledge of the internal network may not evaluate the impact alone, and even multiple IDS' located in different protection domains where alert-correlation takes place, may have challenges on its own [56, 57].

4.3 Proposed Network-Centric Model

The proposed model is based on the idea of protection domains and relates to log data as a multi-tiered architecture. The purpose of such a model is to correlate alerts from a detection control such as an IDS located on the outer parts of a network, with log data on the destination device and intermediate network devices. The model itself does not generate alerts, although it may be extended to perform this, but acts on alerts received from others. The concept of protection domain is central to this model.

A protection domain consists of all network equipment located between protection gates,

which primarily are routing and switching equipment using routing protocols and technology as discussed in Section 2.2. This equipment generates traffic flow data, which is sent to a central collector on the network, being the SIEM solution. The traffic flow data is assembled to determine the exact path that traffic has taken between the gates, and traffic to be mapped when source- and destination addresses change. Assembling traffic flow data is asserted to be critical in order to understand what logs are relevant for evaluating incidents, and in understanding what decisions were made on the different protection gates that analyzed the traffic.

The protection domains in the model are evaluated independently as the domains have no direct relationship. Traffic routed within one domain does not affect the routing in the next domain and is assembled by direction (i.e. ingress or egress). Packet content (i.e. payload), even if encrypted, is irrelevant as only the headers from OSI-layers discussed earlier are needed [6]. The purpose of assembling network traffic in the domain is not to find out what is sent, but rather to determine where it was sent, how it was sent and whether some part of the transport-layers was changed during transmission.

The protection gate inspects the packets and determines whether the packets are allowed into the next protection domain or not. The decision made by a protection gate constitutes the most important evaluation to whether an incident should be followed up or not. If the gate blocked the traffic that raised the intrusion alert, the alert is considered a low priority, or even discarded. If the gate allowed the traffic that raised an intrusion alert, the alert needs to be investigated further according to some priority (e.g. probability of success, impact of attack, or even formal risk-models). Note that investigating further may mean that the traffic entered a new protection domain, and that the process needs to be repeated within that domain.

The types of log data used in this model may be classified into four distinct categories. Each of these categories is used in various steps of the model. The numbers denote the classes:

1. **Detection logs**. Generated at each detection control located in front of or behind a protection gate. Approach used in this study is an IDS, but may also include controls such as proxies, gateways, firewalls or Data Loss Prevention (DLP) systems.
2. **Network traffic logs**. Generated by all equipment within each protection domain. Typical examples are switches and routers, and even firewalls.
3. **Protection gate logs**. Generated on the border of each protection domain. Refers to both ingress- and egress-filtering decisions made by firewalls, proxies, gateways, DLP or others.
4. **Device logs**. Generated at each device being targeted by the attack. Includes resources such as user-clients and servers. Basically the final destination of the packet.

The logs are categorized into different groups based on a session-oriented approach, including devices as depicted in Figure 6. All logs are then assigned to a group being relevant for a particular session, which includes network traffic logs, protection gate and detection logs. When analysing a particular event, the specific session is investigated in order to determine whether

[6] Encrypted packets would cause problems particularly in two situations. First, the IDS would be unable to inspect the packets content to detect attacks. Secondly, data may be tunneled so that what is sent to a destination is re-sent to a different host and thus escaping detection mechanisms (and even circumventing the impact analysis). A general best-practice is however, that encrypted traffic is either denied, or inspected through the use of SSL-termination equipment.

the session was allowed through the gates, and to determine the destination device. When the destination device has been determined, and the session logs show that the session was allowed through all gates, the investigation process can begin.

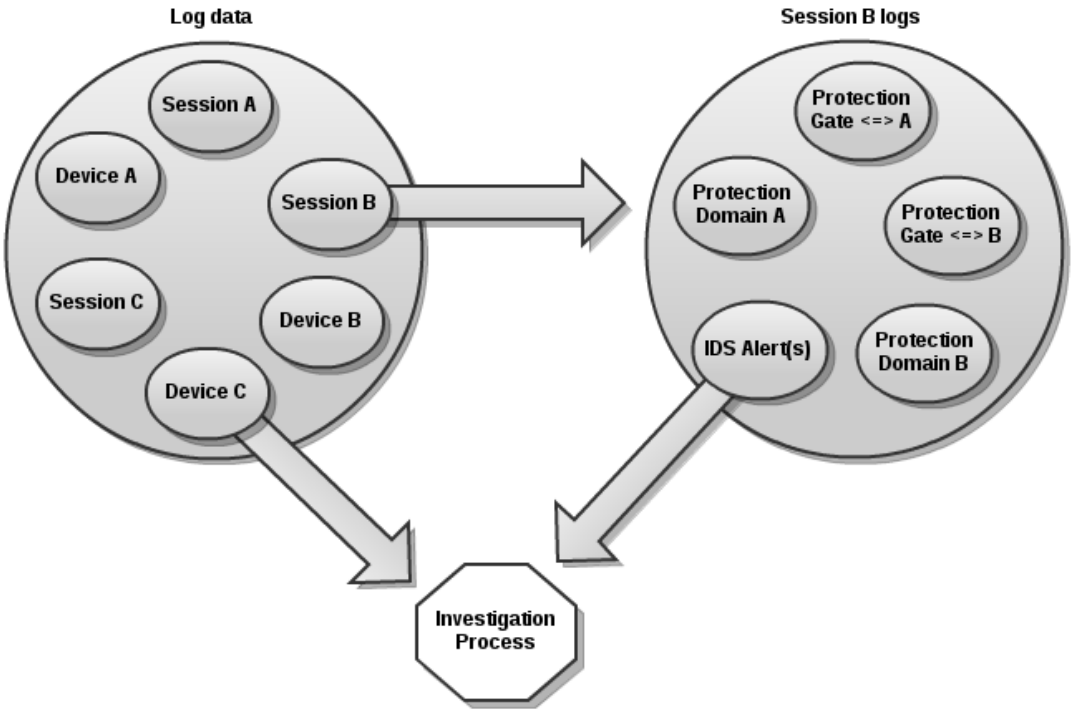

Figure 6: Log events and their relationship in the model.

The goal of the investigation process is to determine what impact the session had on the victim, which is done by correlating logs obtained from the device with the raised intrusion alert. The outcome of this investigation is the re-evaluation of the severity of the incident. For instance, if the intrusion detection system yields that the intrusion-severity is high, the investigation process based on logs may determine that the actual severity is much lower, and even adding the probability of success is low. Each investigation may then be a risk evaluation by itself, which is a more precise and all-encompassing evaluation than the individual intrusion detection system can provide. The evaluation process may be an impact analysis as studied by [20] or any other research technique discussed in Section 2.1.3.

4.3.1 Step 1: Detecting Intrusions

The first step in the model is to acquire alerts that have been generated by some security control, such as an IDS. The alert forms the basis for which the correlation process continues. When an alert is raised, the model interprets the alert in relation to some connection event. The connection event is the session, which forms the process of grouping all data that is related to that alert. The grouping may be based on the event's header such as source- and destination ports, or with any other identifiers derived from the network packets that are considered necessary to relate

connection events and intrusion alerts.

Figure 7 illustrates a design where there are two detection controls. This is a typical design used when one needs to differentiate between traffic that passed through the gate and allowed into the organization (i.e. ingress-filtering), and on the outside to determine whether internal traffic was allowed through the gate to external addresses (i.e. egress-filtering). If this design is used, and both detection controls have the same policy and detection model, they would both raise an alert when a suspicious event took place based on the same network traffic. The proposed model is not concerned with the correlation between alerts generated by two or more detection controls. This is addressed in other research such as [11, 58, 59].

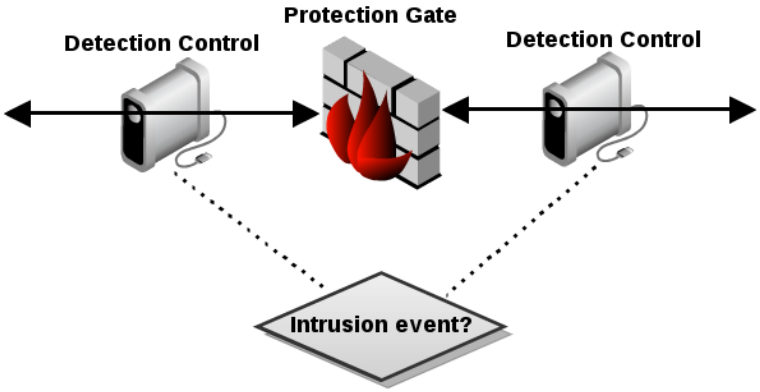

Figure 7: Intrusion detetection and correlation preparation.

The model is reactive in nature as it deals with traffic that has taken place and analyses alerts in a larger context for the security analyst. A potential issue with this is that an alert may be raised on sessions that have not completed, which could be the case when large amounts of data are transferred in a single session, or when a session does not terminate properly (a time-out will occur). Despite that sessions may not be completed in some situations, they may be complete enough to be analyzed in the context the intrusion alert was raised in.

4.3.2 Step 2: Tracking Flow Direction

The second step in the model determines the aggregated packet flow (i.e. sessions) path throughout the network domain. It does so by taking all flow events and assembling these to determine path from the source (i.e. from a protection gate) to the destination (i.e. next protection gate, server, host). Each piece of network equipment will generate flow events, and allow these to be combined and collected. The flow tuples *source and destination IP*, and *source and destination port* are combined for this (c.f. Section 2.2.1 for NetFlow tuples). The latter tuple will only be present when the traffic concerns TCP-sessions. Figure 8 illustrates the area of focus for step 2.

When determining the path for a session, we may encounter situations where the session is fragmented into multiple packets. Each of these fragments may take different routes, and normally be re-assembled at some central locations (e.g. intermediate network equipment) [7]. In

[7] Some protection gates and other types of network equipment, require the complete packet to be ressambled in order

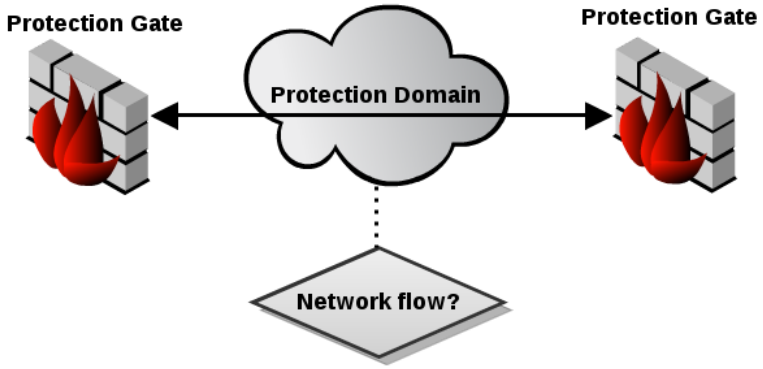

Figure 8: Determining flow direction between protection gates.

case the packets are re-assembled after a protection gate, the model needs to interpret how each of these packets was treated at the gate in step 3.

This step is repetitive for each protection domain identified to be a part of the traced session, and will only correlate logs that are related within that particular domain.

4.3.3 Step 3: Evaluating Protection Gate Decisions

The third step of the model will evaluate the decision made by the protection gate on the particular packet (or session) passing through it. The gate's decision influences whether the threat may materialize (i.e. the alert may impact) or if it is blocked before reaching its target. This is illustrated in Figure 9

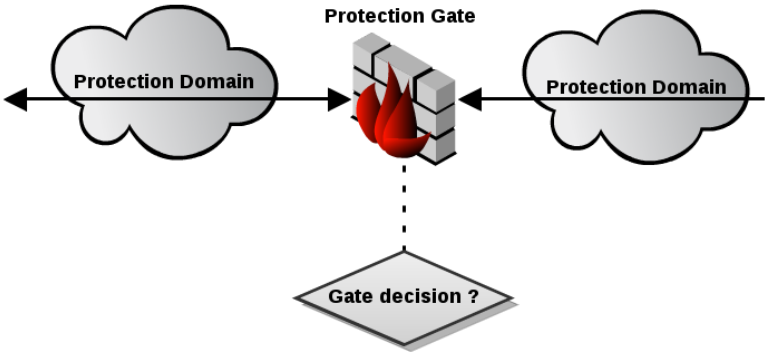

Figure 9: Evaluating decisions made by protection gate.

The protection gate may take many different actions on the packet, and generate log entries on its decisions. The actual evaluation process made by the protection gate is not of particular interest in the model as we are only interested in its decision in binary form – was it authorized or not? This requires the model to have some understanding of the actual type of device the gate is, and the ability to interpret log data collected from it. This is not straightforward for

to make a decision on how to proceed.

heterogeneous networks, which encompass numerous different vendors, models, systems and versions, but should not be a complicating factor once the log data are supported (i.e. structure defined).

Another important area the model needs to account for is whether the gate modifies the layer-3 address of the packet. This influences the subsequent correlation as for instance the IP-address 10.0.0.1 would have no relationship to the IP-address 82.34.11.245, unless the two addresses are mapped. This address-translation is performed by the gates (and sometimes routers and switches) that support Network Address Translation (NAT), and is common practice from external to internal network and vice versa. Once the binary answer about the packet has been determined, it needs to determine the outgoing address of the packet. Intuitively, if a packet pass through a gate with the same header and/or identifiers, the packet was allowed through by the protection gate.

The decision made by protection gates is evaluated in terms of ingress- or egress-filtering. If traffic is blocked ingress, it may not need to be addressed by the incident handler as the client was neither compromised nor put at risk. If it is egress, the traffic needs to be addressed as the cause may be a deliberate policy breach, attempt to circumvent policy restrictions, or a compromised client sending call-back traffic [8]. This is an important distinction in the model because egress-filtering has higher severity when it blocks. The model could then correlate device logs based on the gate decisions made to block the session. This is however, in contrast with our idea of having a model that only needs to understand the binary decision.

4.3.4 Step 4: Estimating Intrusion Impact

The final step in the model is related to the actual impact evaluation of an intrusion alert. The intrusion alert is related to some part of the targeted host, and the model should first understand what parts of the host are exposed and what logs are relevant. This correlation process requires some understanding of the intrusion alerts. This step is illustrated in Figure 10.

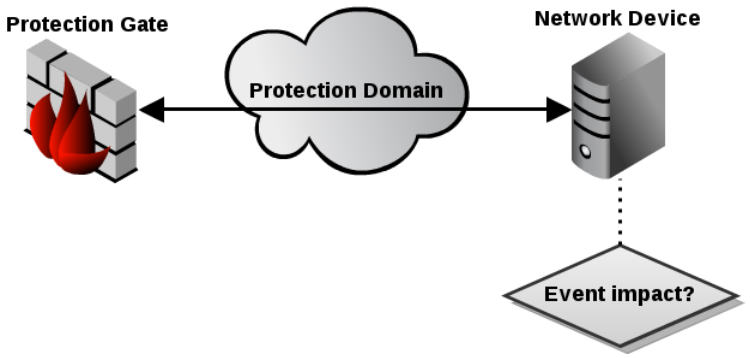

Figure 10: Determining intrusion alert impact.

The impact evaluation attempts to determine whether the host was compromised, and if its state were in any way altered because of the event that triggered the intrusion alert. Depending

[8] Call-back refers to malware that has succeeded in infecting a host, attempting to send information to its owner (c.f. backdoor, control-center).

on the outcome, the alert is prioritized accordingly. The model is not concerned about the severity level necessarily, but rather about determining whether those alerts constitute a real threat and in enabling security personnel to better prioritize which alerts to investigate. The model is neither concerned whether an intrusion alert is a false positive. As far as the model is concerned, all alerts are real. Its primary objective is to determine whether the alerts constitute an actual threat, and to what degree these require follow-up.

The alerts will be based on the severity level presented by the intrusion detection system, as the model does not relate to the actual context the alerts were raised in. What is important, however, is to classify alerts based on their probability of success. The model will operate with the following categories:

True: The host was negatively affected by the traffic alerted by the intrusion detection system.

False: The host was **not** negatively affected by the traffic alerted by the intrusion detection system.

Unknown: It cannot be verified or refuted that the traffic affected the host.

With these categories in mind, the incident handler would deal with alerts first located in the true category, then the unknown. The false category would not be considered, as these alerts have no impact on the target. The categories fall into a two-dimensional queue, where alerts are sorted by their respective severity level. The severity level is based on the evaluation made by the intrusion detection system, in combination with the evaluation process, as stated.

A potential problem is that multiple alerts being related to the same attack fall under different categories. As mentioned earlier, alert-correlation should also be considered to reduce the chance for ambiguity and avoid separate alerts that are related to the same incident. Having alerts related to the same attack grouped together is also beneficial as it increases the confidence level that the alerts are handled correctly.

5 Experiment

The first part of this chapter reviews the network topology that the experiment is based on and discusses various aspects of it such as composition, traffic flow, policy decisions, followed by a discussion about how to interpret data. The last part of the chapter is the actual experiment, where data is presented and observations discussed.

5.1 Experimental Setup

The case study has been performed with the environment depicted in Figure 11, using three physical machines running VMware Workstation 7.1, together with a Cisco 2600 series router connecting them. All hosts in the experiment were virtualized on these three machines: One machine virtualized the red and purple domain, one virtualized the orange domain, and the last one the green domain. The Network Interface Card (NIC) on each of the physical machines redirected all traffic to their respective virtualized protection gate, running Smoothwall Express 3.0 update 8, and ran a NetFlow service to collect network traffic flow.

The physical machines were set up with Linux Ubuntu 10.04, and three Free and Open-Source Software (FOSS)' related to NetFlow. The first, being *fprobe*, is a libpcap-based tool that listens on network traffic and emits NetFlow flows to a collector of own choice. The collector we used, being the second FOSS, was *nfcapd*. nfcapd is a NetFlow capture daemon that reads NetFlow data from the network and store it into files on the hard drive. fprobe was configured to use NetFlow ver. 5 as higher versions do not provide any additional tuples that we would need [1]. The third FOSS is *nfdump*, which is a NetFlow display and analyze program.

We encountered two problems when using the described virtualized environment, which had to be resolved. First, all network traffic being transmitted and received on the physical NICs was captured by our NetFlow emitter, including internal host traffic not transmitting external traffic (i.e. to router). This affected our data set as we would end up with duplicate flows for some networks. Our solution to the problem was to enable a filter to the running fprobe process, which made it to ignore any traffic that matched the filter. Here is an example of a filter we used on the physical machine hosting the orange protection domain:

```
(host 192.168.1.1 or not (src net 192.168.1 and dst net 192.168.1)) and not
host 10.10.10.4
```

Our second problem was that we observed our physical NICs transmitting duplicating traffic. It turned out that it was a known issue that existed for virtualized hosts configured as *bridged mode* in VMware [60]. There were several proposals on how to deal with the issue on Windows, but none applied to our environment, and there was no fix available from VMware. Bridge mode

[1] This is not entirely true as ver. 9 introduced the concept of templates and field type definitions. Our Cisco router did not support this however, due to old Cisco Internetwork Operating System (IOS).

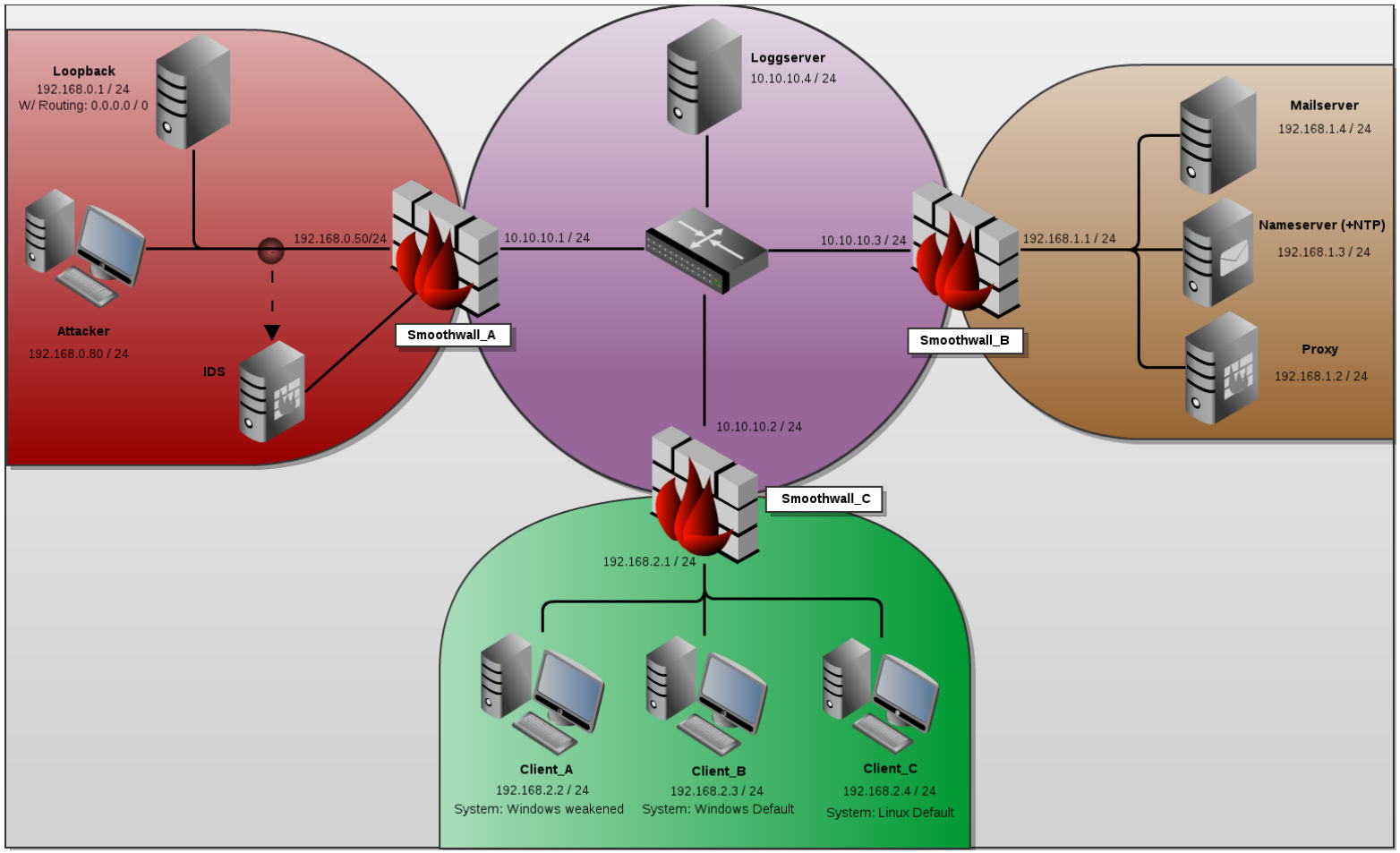

Figure 11: A link-layer overview of the experiment's network composition.

was used by the protection gates located on the 10.10.10.0/24 segment, as they needed to communicate with each other through the router and with the virtualized hosts configured as *host-only*. We were able to identify where this occured (NICs connected to the purple domain), and addressed the issue by halving some of the flow tuples in the algorithm (Packets, Bytes and Flows).

5.1.1 Network Composition

A physical network may be represented in various forms such as through link-layer, network layer or overlay topology. Representing a network through link-layer is a more suitable approach for us, as it is an intra-domain task, which is more suitable when discussing small-scaled networks [5]. The network topology used in the experiment adheres to the broad consensus of how a network should be structured (c.f. Section 4.2). It consists of four protection domains, whereof three may be considered internal and the fourth external (i.e. not in control by the organization, which simulates the Internet in our environment). Each protection domain has a color representation that denotes how restrictive the policy is within that particular domain. These are as follows:

Red: Completely open policy, constituting the most untrustworthy domain where the organization has neither insight nor control of what takes place. The protection gate in front of the red domain is primarily for blocking "Internet noise", non-targeted attacks, and general reconnaissance/probing traffic. It is expected that some traffic originating from the Internet are allowed through, as the organization is hosting services located on the internal protection domain being public accessible.

Purple: Is the "roundabout" of the organization. All traffic entering this domain is routed to the other respective domains (with the exception of the log server). The Cisco router ensures that external hosts cannot access internal resources without circumventing its access control and ensures that internal traffic is properly routed.

Orange: Enforces a semi-open policy, where only services located within that domain can be accessed. Only traffic going to and from these services is allowed through the gate. The domain constitutes a DMZ-zone where services cannot be fully trusted.

Green: The most restrictive policy of the domains, where all the internal clients are located and valuable information stored. It cannot be accessed directly from the external domain without the clients first initiating the communication channel [2], and where hosts on the inside has been configured to always send traffic through the proxy located on the DMZ. This is done through an OS-configuration and may easily be circumvented.

Each of the domains has been hardened according to their policy, based on best practices for firewall-configurations. The domains may be viewed as a linear model, where each domain down the chain is more secure (i.e. restrictive) than the previous. The lower the chain, the less are the resources in that domain allowed to do. In terms of the traffic flow, all outgoing traffic is allowed

[2] Stateful inspection is used on all protection gates to enforce this.

for the following directions: 1) purple towards the red domain; 2) Orange towards the purple domain; and 3) green through the purple and towards the orange domain, via proxy. Notice that resources located in the green domain should not access the red domain without passing through the orange domain (enforced through OS-configuration as stated). The orange domain contains security controls such as a proxy-server (Squid ver. 3.0 stable 19) and antivirus-filter (ClamAV ver. 0.97 with most recent signature base) for both Web- and SMTP-traffic.

Every protection domain is protected by a protection gate that performs Source NAT (SNAT). SNAT refer to the process of masquerading IP-addresses behind a protection guard and replacing IP-addresses with new addresses when passing through the protection guard, and vice versa. A table of all the on-going connections is maintained in the guard, where port number is the identifier for each session. This is typically used in large-scaled networks, which has the drawback that one needs to acquire log data or lookup connections in the table to determine what was accessed by a host in front of or behind the device performing SNAT.

5.1.2 Network Services

As stated, traffic originating from internal resources is only transmitted out of the network if their connection attempts are authorized by some security control. Our default policy on the network is to allow all traffic towards known services and hosts, unless the traffic pattern is considered a violation of policy. In the orange protection domain, we have prepared 3 servers and 4 services typically found in organisations.

Proxy: The proxy is the main gateway for primarily Web traffic through ports 80 and 443, but also used for FTP-sessions and the like. It retrieves requests from hosts located internally and externally, and evaluates each request according to a set of criteria. The proxy maintains a state with the requestor, it denies traffic based on port or address, it caches requests, and it performs antivirus scanning on all content that it mediates. The proxy uses port 3128 (Squid default) for all requests.

Mail: The E-mail-server acts both as a Mail Transfer Agent (MTA), IMAP- and SMTP-server for clients on the internal network. It can retrieve E-mails from the outside, and has enabled spam- and phishing-control in addition to running antivirus-checks on every E-mail.

DNS: The Domain Name System (DNS) is basically an electronic phone-book for IP-to-name mappings. For every request that the proxy receives containing names, a DNS-query is sent to the DNS-server. The DNS-server replies with the translated IP-address (discussed later). The DNS also plays a vital part of the security scheme as it is integrated with an IP-and-domain-name reputation list. The DNS responds with either typical answers such as Canonical Name (CNAME), A-records, and pointers to reverse IP-addresses (PTR), or it replies with a negative answer if the queried name is present in the reputation list.

NTP: The Network Time Protocol (NTP)-server is used to ensure accurate time through the entire organisation. All hosts on the network are configured to periodically synchronize with this server. NTPs are particularly important in log analysis as correct time-stamp on log records is necessary to investigate incidents properly, and placing events in the correct timeline.

The first three services maintain logs of each and every traffic-request being made. These log sources are used in the correlation process and when investigating intrusions, and constitutes in general important sources to evaluate whether an intrusion alert poses a real threat. The services will be able to stop or influence traffic related to their field of expertise if their security mechanisms consider this necessary.

5.1.3 External Traffic Simulation

The network used in this experiment needs to be isolated from the real-life Internet as the experiment involves malware that may break out of the environment. This makes it more difficult to simulate a real-world environment considering that the red protection domain is more confined than we would like. We have solved this problem by modifying packets on the routing level, and implementing static routes on each internal protection gate towards our desired destination. Let us explain how this takes place.

There are two aspects of this scenario: an IP-address outside the organisation and an unrecognized Fully Qualified Domain Name (FQDN). In terms of the latter, the internal DNS will translate all names, even if these are not present in the translation table. This is done by assigning all DNS-queries with a random IP, and maintaining a table of those IP-addresses with their respective FQDNs. This operation is performed in real-time. The DNS can then start with a table containing only hosts known from the internal network, and gradually build up the table as queries are made. This is particularly important when dealing with real malware that attempts to break out of the environment. For instance, the domain *www.bbc.co.uk*, which is not hosted within our network, will be resolved to a randomly-assigned IP-address within this network topology. This takes place even if the domain is not part of the initial translation table and that no external DNS have been queried.

So when only IP-addresses are remaining, the question is how to simulate their presence on the network and sending traffic to the designated IP. Routing as intended solves this problem. We have a default route that will match all IP-addresses that are not part of the network topology. The route is as always, towards the barrier of the network, which is the protection gate in front of the red domain. From there on, the protection gate has set the gateway of the default route to be to the host called *loopback*. The loopback host would normally reply to such requests with "Destination port unreachable", or just time out. We have however, altered all packets as mentioned with a pre-routing translation that alters all destination IP-addresses of the incoming interface with the IP-address of the loopback. This means that the loopback will see all traffic as if it is toward itself, and respond accordingly. All IP-addresses outside the RFC 1918-ranges [61] will then be answered by the loopback host.

The point of having arbitrary IP-addresses supported, is because of the attacks we perform in our experiment. With live network simulation, one always strives for traffic characteristics that reflect completeness and real-world appearance. Arbitrary requests come primarily from real malware, which have different interests in terms of target, exploitation vector and more as discussed in Chapter 3. This enables us to simulate real responses to traffic initiated by malware in terms of establishing sessions, and making initial responses to this traffic. We will review the malware later in this chapter.

5.1.4 Event-Sources and Collecting Events

Log sources in the experiment are narrowed down to only include the log sources that may be relevant to the study. We have done this by categorizing the log sources into three groups: a) services relevant for understanding traffic and evaluating risk. Examples of this are proxy and antivirus-solution; b) log data needed to complement network traffic where one cannot relate two independent flows; and c) clients where the malware will be introduced. Because the malware used in this experiment have not been analysed, we do not know the behavior of a client once infected. This prohibits us from narrowing the logs to specific applications or system-components. The collected log sources are as follows [3]:

Proxy: Requests and responses for all types of services, including antivirus.

Nameserver: Queries related to domain name resolution.

SMTP: Access to the E-mail box, E-mail-transfer and content filtering mentioned earlier.

Firewalls: Traffic, both ingress and egress, that was blocked or filtered.

IDS: Alerts raised based on traffic seen on the red protection domain, which is either to or from clients in the organisation. We have activated over 12 000 Snort rules, from Snort officials, Emerging threats and Bleeding Edge [4], as well as the ones developed by us to detect the attack classes discussed in Section 3.2.

NetFlow: NetFlow flows from all the NIC-interfaces seen in Figure 11. Firewalls have two NICs (outer and inner segment), so we need to capture flows on both. Hosts, with the exceptions of servers, will not capture flows.

User clients: Includes both Windows and Linux hosts. These hosts have been enabled with all the logging capabilities that are present in the systems by default. No attempts have been made to extend or to customize the default logging data set.

Log data generated by all of the aforementioned sources are transferred to the SIEM solution with the syslog protocol over default UDP port 514. The SIEM solution with IP-address 10.10.10.4, does not interact with hosts on the network, and only retrieves logs. It tags each log record it receives with source and timestamp. The NTP-server ensures correct timestamp.

5.1.5 Processing Data and Correlating Events

The network is designed so that client-traffic has to pass multiple paths to reach the Internet, where each domain will hide its origin source through the use of SNAT. This means that session-assignment has to be translated for each SNAT-enabled gate. We will use NetFlow for this purpose. When an IDS alert triggers, we will use the alerts details to trace the flow backwards to the client, and then the packet flow to reach the destination. We then have two types of log classes that need to be looked into, depending on how the traffic was initiated: 1) The logs that may be relevant to understand what events took place on the host prior to the alert. This could be for instance when malware on an infected client attempts to send call-back traffic to its controller;

[3] Note that none of the external devices located on the red protection domain, simulating Internet services and attackers, are logged.

[4] http://www.snort.org/, http://www.emergingthreats.net/ and http://www.bleedingsnort.com/.

and 2) The logs that determine how the receiving traffic is treated. Logs may then show whether the client is at risk based on the particular IDS alert that triggered.

Our algorithm for evaluating IDS alerts is as follows:

1. Determine whether the traffic originates from an internal or external client.
2. If the traffic originates from an internal client, trace back to the source of the traffic.
   - Investigate logs along that route.
   - Evaluate logs based on the behavior that was observed.
3. If the client retrieves traffic from the outside, follow the traffic to its destination.
   - Investigate logs along that route.
   - Evaluate the impact of the traffic and any subsequent change in client's behavior.

Each IDS alert, which constitutes an incident that needs to be looked into, contains minimal information about what took place, and requires human intervention in most circumstances. An IDS alert in Snort format is composed of the signature-header, which includes information such as ID and message, and then what source- and network socket triggered on the traffic. We will use this socket information in the algorithm to automate the trace and network traffic, as well as narrowing the needed logs along the way as mentioned.

5.2 Case Study

The experiment will simulate client-side attacks and attempt to evaluate the impact and cause of these attacks based on the obtained logs. We have 250 files that contain real malware samples obtained from various sources on the Internet. Most of these files raise multiple alerts when scanned through different antivirus solutions. The files contain recent malware up to 2011, and which targets different attack vectors. These files will be retrieved and opened by the clients one by one with one second's delay between each retrieval.

Our case study involves an attacker with a masqueraded IP-address sending phishing E-mails to an organization's employees, where the goal is to make the clients execute code (i.e. our malware samples). All of the E-mails contain links that will redirect the users to the attacker's Website, *www.evilsite.com*. The users will connect to the E-mail-server with IMAP, refresh their E-mail-client, and then click on the link, which redirects them as stated. When visiting this Website the users will unknowingly download malware and execute it.

Our observations in this study will focus on tracing the IDS alerts once they trigger, and look at what information and problems we encounter along the way. The study may be viewed from three perspectives: First, E-mails are sent into the network. Second, the user client is opening E-mails and attempting to acquire the malware, with subsequent download and execution. Third, the malware will try to break out of its environment, in which the IDS will observe the traffic going out of the network, again. We do not know what each piece of malware attempts to do in terms of the latter, but we will monitor the network for 60 minutes after the last piece of malware has been executed and collect data throughout that period [5].

[5]The reason for choosing this time duration is to make sure that as most data as possible is collected during the experiment. This is beneficial when studying the correlation process.

5.2.1 Results

During the experiment, it was collected a total of 9465 log entries (**excluding** flows) throughout the network. Most of these were related to the clients and the proxy server, which is expected considering our topology and choice of log sources. Logs from the firewalls were only collected when traffic was blocked, which turned out to be particularly much for the DoS-exposed guard. Because our topology was configured with an open policy in terms of outgoing traffic, few of these log entries were related to this. Table 2 outlines the number of collected logs, sorted by source.

Source	**Log entries**
192.168.1.2	1613
192.168.1.3	212
192.168.1.4	821
192.168.2.2	1054
192.168.2.3	569
192.168.2.4	397
10.10.10.1	1233
10.10.10.2	3566
10.10.10.3	0
Total	9465

Table 2: Log events collected during the experiment.

In addition to the log entries collected from the internal network, we have the IDS located on the external network. It raised 972 alerts during the experiment. This includes all alerts raised for when malicious files were downloaded to the client, for when the infected machines attempted to connect to the outside, and for the attempt to DoS an internal resource. The top 10 alarms, accounting for 72% of the triggered alerts, are seen in Table 3.

Signature	**Numbers**
1:15334384:5	233
1:2011938:3	106
3:15454:4	91
129:15:1	59
1:2010882:3	52
1:12592:3	50
1:17668:1	48
122:3:1	20
1:17543:1	19
1:17517:2	16
..	
Total (all alerts)	972

Table 3: Top 10 Snort alerts triggered during the experiment.

The top three IDS alerts from Table 3 are shown below.

```
[**] [1:15334384:5] Large number of ICMP ping  [**] [Classification: DOS-attack]
     [Priority: 3] {ICMP} 108.164.164.51 - > 192.168.0.50

[**] [1:2011938:3] ET MALWARE CryptMEN HTTP library purporting to be MSIE to PHP
     HTTP 1.0 [**] [Classification: A Network Trojan was detected] [Priority: 1]
     {TCP} 192.168.0.50:49870 - > 192.168.0.1:80

[**] [3:15454:4] WEB-CLIENT Microsoft Office PowerPoint malformed msofbtTextbox
     exploit attempt [**] [Classification: Attempted User Privilege Gain]
     [Priority: 1] {TCP} 114.87.23.221:80 - > 192.168.0.50:35924
```

When alerts were raised, we correlated them with NetFlow and log data, and evaluated the risk based on what had been raised. First, we used NetFlow to trace back the traffic source to the client, and the traffic flowing towards the client. Each time the NetFlow passed through a protection gate, we checked whether there were any blocking statements that described why it was blocked. Once the entry and exit flow had been identified, we could evaluate what timing information was relevant for the logs, based on what had been detected by our IDS. This process was automated in an implementation of the algorithm, where each IDS alert provided the output as shown in Table 4.

Based on NetFlow, we were able to determine whether traffic that had raised an IDS alert, was successfully received by the client. The traffic flow further enabled us to establish a precise timeline for the entire session, allowing us to narrow down the amount of log records for each passing device to an absolute minimum. In terms of outgoing traffic, we acquired all the log records on the client for a given threshold, up until the first flow was seen. For incoming traffic, we could look into the log data beginning from the last NetFlow reaching the client, based on the timestamp.

Results shows that of the 972 IDS alerts that triggered, all except the ones not based on UDP/TCP, could be traced accurately back to the client and protection guards. 367 of the IDS alerts could be mapped to log records that showed the sessions being blocked by the protection guard. 334 of the IDS alerts could be traced back to security mechanisms such as the proxy and antivirus-solutions that blocked the session. The high number of IDS alerts is caused by some malware that resulted in multiple IDS alerts, where all could be traced back to the exactly same proxy-entry due to identical network sockets.

```
################## ALERT START ##################
Time:  May  3 21:17:45 192.168.0.1 1304450333.351: 05/03-21:18:53.301462

IDS:  [1:17668:1] POLICY attempted download of a PDF with embedded JavaScript
      [Classification: Potential Corporate Privacy Violation] [Priority: 1] {TCP} 114.87.23.221:80 - > 192.168.0.50:42697

Date flow start          Duration Proto   Src IP Addr:Port      Dst IP Addr:Port   Packets   Bytes Flows  Location
   <Client start>
2011-05-03 19:18:53.259    0.249 TCP     192.168.2.2:1053  ->   192.168.1.2:3128     5     504   1   192.168.2.1
2011-05-03 19:18:53.259    0.249 TCP      10.10.10.3:1053  ->   192.168.1.2:3128     5     504   1   10.10.10.3
2011-05-03 19:18:53.710    0.249 TCP      10.10.10.3:1053  ->   192.168.1.2:3128     5     504   1   10.10.10.2
2011-05-03 19:18:53.711    0.249 TCP      10.10.10.3:1053  ->   192.168.1.2:3128     5     504   1   192.168.1.1
2011-05-03 19:18:53.791    0.021 TCP     192.168.1.2:42697 -> 114.87.23.221:80      12     976   1   192.168.1.1
2011-05-03 19:18:53.791    0.026 TCP      10.10.10.2:42697 -> 114.87.23.221:80      12     976   1   10.10.10.2
2011-05-03 19:18:54.240    0.026 TCP      10.10.10.2:42697 -> 114.87.23.221:80      12     976   1   10.10.10.1
2011-05-03 19:18:54.240    0.026 TCP    192.168.0.50:42697 -> 114.87.23.221:80      12     976   1   192.168.0.50
      <IDS-ALERT>
2011-05-03 19:18:54.241    0.026 TCP    114.87.23.221:80   ->  192.168.0.50:42697   12   12360   1   192.168.0.50
2011-05-03 19:18:54.241    0.026 TCP    114.87.23.221:80   ->    10.10.10.2:42697   12   12360   1   10.10.10.1
2011-05-03 19:18:54.592    0.026 TCP    114.87.23.221:80   ->    10.10.10.2:42697   12   12360   1   10.10.10.2
2011-05-03 19:18:54.593    0.026 TCP    114.87.23.221:80   ->   192.168.1.2:42697   12   12360   1   192.168.1.1
      <BLOCKED>

(proxy): May  3 21:17:45 192.168.1.2 1304450265.297: pport:3128 cip:10.10.10.3 cport:1053 dip:114.87.23.221 stat:403
 user:- reqply:963 resply:295 reqdata: GET http://www.evilsite.com/batch3/bd8ef15388213da62e6ad874079843c1.pdf
 1.1 size:TCP_MISS offset:DIRECT type:text/html tr:115
################## ALERT END ##################
```

Table 4: Correlation example for an IDS alert.

5.2.2 Combining Flows

A flow in NetFlow is a set of collected characteristics between any two end-points. Most networks consist of numerous end-points between source and destination. To use NetFlow for our purpose requires that flows between any two end-points can be combined. When there's a direct communication between the end-devices, the flows can be combined by comparing the network socket between all intermediate devices. In our case, we had protection gates performing SNAT, which meant that IP-sessions would be translated based on port numbers between each network segment. We combined the port number with traffic between the protection gates to combine flows (i.e. network sockets).

A problem with combining flows as observed, is that it is only possible as long as the session-traffic is not broken (i.e. forwarded). If for instance, network traffic is passed to a server or end-point, which took place in the experiment for clients connecting to the proxy and other servers on the DMZ, we were unable to combine flows along the complete path from source to destination. When traffic reached a server on the DMZ, the traffic was sent over the same segment with a new connection being independent from the previous connection. This is comparable to SNAT on the server, but without the possibility to compare NetFlow tuples. The scenario may be depicted in Figure 12.

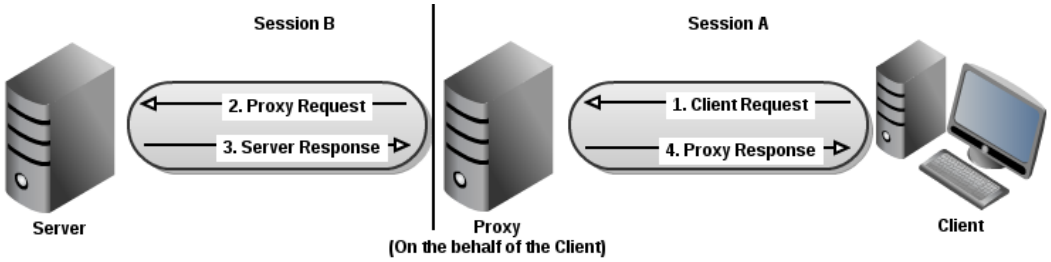

Figure 12: Session request mediated by the proxy on behalf of the client.

5.2.3 Persistent and Pipelined Sessions

Another problem we encountered was that it was in some cases impossible to track down IDS alerts to end-devices based on NetFlow with sufficient accuracy. Many types of protocols (e.g. HTTP, FTP) provide the possibility for something called persistent connections. Persistent connections allow the source and destination to maintain the connection (i.e. TCP state) once it has been established. The end-devices then continue to use the identical port number until the connection is terminated.

The purpose of persistent connections is well-described in RFC 2616 [62]:

- By opening and closing fewer TCP connections, CPU time is saved in routers and hosts (clients, servers, proxies, gateways, tunnels, or caches), and memory used for TCP protocol control blocks can be saved in hosts.
- Network congestion is reduced by reducing the number of packets caused by TCP opens, and by allowing TCP sufficient time to determine the congestion state of the network.

- Latency on subsequent requests is reduced since there is no time spent in TCP's connection opening handshake.

Whether a connection is persistent depends on how the end-points set up the connection. In HTTP 1.0, the line "Connection: Keep-Alive" can be seen in the packet header. In HTTP 1.1, this is no longer needed as it is enabled by default and must be explicitly turned off. As we were using SNAT, persistent connections prohibited us from breaking down in- and outgoing-connections and correlate them with precision to IDS alerts. We resolved the issue by forcing both the proxy and the attacker's service to use non-persistent connections.

We also encountered the problem of pipelining, which is the concept of allowing a client to make multiple simultaneous requests without waiting for each response. Single TCP connections are then used more efficiently, but it complicates the correlation process as a single session cannot be separated in terms of flows. By disabling persistent connections, we also prohibited pipelining from taking place.

5.2.4 Network Socket Reuse

The number of ports is limited to 16-bit, assigned to sessions and released when TCP connections have completed their TIME WAIT state. Because of this, port numbers are bound to be repeated over some time period. Windows in particular, incorporates a 120 seconds delay in their algorithm to avoid confusion with duplicated TCP segments of the old connection [63]. With low-traffic clients, which were the case in our experiment, it takes longer time than high-load hosts to repeat the same TCP segments. Port numbers were our unique identifier as mentioned, and considered an essential component to combine flows.

When running through our algorithm in real-time, we encountered no problems because of the low traffic load. However, when running the algorithm on data that was gathered after the experiment had ended, which is likely to be done on historical data, we observed several conflicts between end devices. One IDS alert resulted in two different network flows where two different clients were identified. The problem could be solved by looking at all the flows in conflict, and determining the minimum amount of time between them. It turned out that with our network traffic, 15 minutes was sufficient. The problem took place on the client with IP-address 192.168.2.2 and the proxy server in particular.

5.2.5 Port-Less Sessions

One of the attack-classes we used was targeting availability through the use of DoS-attacks. The attack itself was not particularly sophisticated as it only sent large amounts of ICMP-packets to the network, but provided some insight to the model in relation to the attack-class. The attack-class may in non-sophisticated cases be considered very noisy and easily detected by the IDS. The IDS had threshold-signatures that were raised each time a number of certain characteristics had repeated. Here is one example of the alert that was raised:

```
[**] [1:15334384:5] Large number of ICMP ping  [**] [Classification:
     DOS-attack] [Priority: 3] {ICMP} 108.164.164.51 - > 192.168.0.50
```

We attempted to DoS one of the servers on the DMZ, which was allowed by protection gate A but blocked by protection gate B. This explains the high-number of log entries from this source compared to the other sources. In NetFlow, we saw entries such as the following (c.f. Table 4 for field description):

```
2011-05-03 22:39:20.506     66.010 ICMP    108.164.164.51:0
    ->      192.168.2.4:0.0          2317      252232     1
```

ICMP packets, which are encapsulated as part of the level-3 on the OSI stack, do not have any port numbers (Introduced on level-4) [6]. Because we were unable to relate flows to IDS alerts, we could not determine the risk and impact of the DoS attacks, and neither trace the traffic to its source or destination. NetFlow would then be unsuitable for availability attacks as described here, as IDS alerts could not be related to the flow. In NetFlow, flows that are port-less, are denoted with a zero.

[6] In firewall-terms, ICMP traffic is filtered by the flags found in their header.

6 Discussion

This chapter contains a discussion about the experiment and our observations when conducting it. We will focus on the issues we encountered and discuss these in relation to our research questions. We start by discussing the model and its challenges, followed by some sections discussing various issues. The last section proposes a set of requirements that we argue a network-centric SIEM solution should fulfill.

6.1 Completeness of our Model

While the proposed model has the potential to serve as a basis for a network-centric SIEM solution, the actual efficiency of it relies on multiple factors that were not addressed. In the experiment, we encountered situations where multiple alerts were raised based on a single network session containing JavaScript and shellcode in particular. Even though these alerts were obviously related to the same malware, our model considered all alerts to be individual. This is undesirable in many situations, depending on how the protection gates treated the traffic flow and where the traffic flow was initiated.

The lack of correlation between IDS alerts influences two aspects in particular: First, we have the size of the alert set visible to the security analyst. As the security analyst will review each alert, the number of alerts should be reduced to an absolute minimum. This is related to Collaborative IDS (CIDS) techniques, concerning the correlation and clustering of alerts. Secondly, we have the precision of the correlation process. As IDS signatures trigger on different characteristics of the network traffic, the SIEM will need to make different correlations based on a single intrusion. We assert the correlation process may need to relate to attack vectors. The result of such a process may lead to contradictory results where one alert is verified, and another unverified or inconclusive. The security analyst may then end up with being presented a set of alerts that require extensive human interpretation, which creates complex and contradictory situation [64]. One approach to deal with the issue would be to add precedence to some types of alerts in the view. Another would be to weight various outcomes differently (e.g. verified intrusions).

Prioritizing Alerts

Prioritizing alerts is the concept of presenting alerts, or an evaluated incident summary, to an incident handler so that the alerts with highest criticality based on the organisations assets, are addressed first. The algorithm used in the experiment did not have any sophisticated alert-prioritization scheme, as the log correlation algorithm was not advanced enough. The experiment showed that we could significantly reduce the amount alerts by being network centric, and provide the incident handler with a view as needed to analyze incidents. A model as proposed, should be extended to include a summary after the SIEM correlation process, where incidents can be dealt with according to a stack-based approach. One could for instance, rank incidents according to a numerical value where each value is the calculation of various attributes, such as:

Traffic direction: In which direction was the alert raised? Outgoing traffic has a higher weighted value as the intrusion takes place behind the security perimeter (c.f. Table 1), and given that security policies are traditionally less restrictive for egress traffic. Incoming traffic on the other hand, are subject to Internet noise and have a higher false positive rate.

Protection gate action: What was the decision taken by the protection gates on the network traffic? Traffic passing through a protection gate is more severe than blocked traffic.

Outcome correlation process: What was the result of the SIEM correlation process? An alert, or cluster of alerts, should have precedence over other alerts with the same severity if the alerts are verified to be real threats (i.e. higher confidence level). The outcome of correlation phase does not necessarily need to be conclusive, it could also be a probability measurement based on asset characteristics (e.g. OS, services, versions).

Severity rating: How severe is the attack considered? Attacks with higher severity should be prioritized. The severity level should not be dictated by attack vector alone, but also based on the criticality of services and hosts (i.e. integrating business assets).

Attack categorization: What type of attack was observed? A policy violation for instance, should in general have lower attention than trojan-activity. Business goals may also dictate that confidentiality attacks as discussed in Section 3.2.1 have precedence over availability attacks (e.g. stock brokers vs. Internet banking).

6.2 Network Composition Challenges

Network composition is the result of a design process intended to support mission goals, with influence by secondary factors such as security and performance. Our network design was based on best-practice, as discussed by [48, 55]. Network composition does, however, vary greatly between organisations when it comes to products, techniques and configurations. We believe that the model is widely applicable to various network compositions, but there are several aspects that may need to be considered in this respect.

First, we want to discuss the amount of data that is needed in a network-centric model. In our experiment, we collected flows from all network segments in the environment, with the exception of client machines. It turned out that flows were only needed in the central points of the network, where the network layer was altered [1]. For instance, we saw that we did not need flows from the purple domain (i.e. Cisco router), as these flows were not critical in tracking sessions throughout the network. It was possible to compare the flow data set from the protection gates connected to the purple domain and find out the same information. In terms of performance, however, it may be necessary to include non-critical components such as the router constituted in our network, to improve efficiency of the correlation process. On the other hand, the amount of data generated by a network environment may be a limiting factor as the amount of exported flow records could overwhelm the collectors [65, 66]. Even if there exist techniques to cope with this such as sampling, aggregation and filtering, these may negatively impact the correlation process as the necessary level of detail is not exported.

[1] This may not be applicable for all types of environments.

Secondly, in larger organizations one would encounter issues related to routing-optimization and where traffic-load dictates the need to load-balance traffic between multiple devices. Our network did not have multiple paths to each destination (i.e. low edge degree per node) so correlating a complete traffic picture could be done by collecting flows on all relevant NICs with high accuracy. This influences both IP-fragmentation and flow-separation, which may cause problems in terms of an incomplete data set when executing the algorithm.

Thirdly, our model was able to determine the outcome of some IDS alerts by having an understanding of how the log data was structured (i.e. format). This was particularly the case when interpreting logs from the proxy. Without having structured data and understanding how to interpret it, we would not be able to combine the flow going from the external server to the proxy, with the flow from the proxy to the internal client. This is a critical point as the whole approach is based on the concept connection tracking, when isolating smaller set of log events to evaluate potential impact. This is however, not considered a difficult issue as SIEM-solutions today are already known to have a wide support for a multitude of log sources.

6.3 Applicability in Real-Time Analysis

Analysis and evaluation of intrusions in real-time are beneficial as they make it possible to minimize the potential impact for the intrusion to an absolute minimum. Real-time analysis in relation to the presented model will not be realistic, considering that the model is reactive by nature when correlating data based on log data (i.e. historical data). Although the correlation of flows took several minutes, the bottleneck is not a matter of finding an efficient algorithm for the model, but rather that flows are summaries of traffic characteristics collected over a certain amount of time. The time is adjustable in several aspects, as mentioned in Section 2.2.1. Minimizing the time-values, causing the flows to be exported more frequent, may contribute in getting the model as close to real-time as possible.

Another problem in real-time analysis is related to the uniqueness of flows. We observed that over a certain amount of time, there were duplicate flows in our collector. This duplication was only observed after the experiment had been completed, when analyzing all flows based on a full data set. This may as well cause problems in real-time analysis. One example for this are servers or hosts with high traffic load such as the proxy in our environment. Unless flows are exported and flushed from the NetFlow buffer before another identical flow is started, duplicates will occur. Typical timeout vales for TCP-segments, as discussed earlier, should be addressed in this context. Avoiding duplicates is, as we stated, matter of calculating the appropriate time for flushing the buffer and extending the flows to include as many unique tuples as possible.

6.4 Coverage of Attacks

We saw in the experiment that flows could be used when the IDS alerts contained port numbers. There are however, many types of attacks that do not contain any ports, and perhaps especially those attacks targeting availability. Availability attacks, and in particular those attacks that create high volumes of traffic on the network, deviate significantly from normal baseline in terms of number of packets or transferred bytes. While we could not track this traffic in our model, there are fields related to availability attacks that have been considered in this area. In study [34],

the authors performed flow level traffic measurement to detect such attacks. One could consider areas where the model included traffic flow measurement to track this type of traffic.

Coverage of attacks was also, in particular in our experiment, heavily dependent on the IDS' ability to detect attacks or irregularities in the network traffic. If the IDS did not detect an attack, the attack would bypass the model. To extend the attack coverage, the model should include complementary correlation methods, such as provided by todays SIEM solutions. The connection tracking could then be used afterwards to provide the necessary context of the detected intrusion. Other types of intrusion detection sources can include network flow anomaly, blocked events by protection gates, or application-level events such as viruses detected by an antivirus solution.

6.5 Protocol-Level Awareness

One of our biggest concerns for the model is the need to separate flows according to a particular set of protocol-behavior. We were able to create an accurate context of IDS alerts as each session passing through the IDS was related to a particular application-level request or response. When application protocols reuse established sessions, as with HTTP persistent connections and FTP, it is difficult to separate "causes" in network sessions and providing context on particular causes. This may be addressed by adjusting the NetFlow timers to make the protocol export sessions faster, but this does not really solve the problem.

Even if the timers could be adjusted to an acceptable level, the NetFlow tuple *timestamp* may be misleading in the correlation process. The timestamp is set for the first flow for each flow ID. This implies that the security analyst in particular forensic investigations, cannot provide accurate time-information for IDS alerts where the flow consists of multiple application level exchanges between the client and the destination. Timestamp may not be critical in terms of the correlation process, depending on the algorithm, but it certainly complicates the subsequent human process of analyzing alerts afterwards.

Another aspect with some services such as proxies, is that they separate traffic coming from in- and out- NICs, causing the two traffic sessions to have no apparent relationship. For instance, if an internal client makes a request to an external server through a proxy, that request does not go directly to the external server. Instead, the request is interpreted by the proxy, which then generates a new and independent request to the external server on behalf of the internal client. The issue here is that one needs to be able to map these independent requests to be able to establish complete flows throughout the entire network.

We saw in the experiment that NetFlow could not be used for this purpose as the keys between flows from in- and out- NIC were not preserved. The proxy did neither generate log entries that allowed this mapping to take place, requiring us to create a script to deal with the issue. The script interpreted traffic, connected network sessions that had a relationship together, and created a log entry that allowed the complete context to be established in the correlation process. Our correlation algorithm had to resort to these log entries when traffic reached servers that acted as mediators between internal clients and external servers.

6.6 Requirements for a Network-Centric Model

The proposed model has been implemented and tested in the experiment using Snort as the IDS and Cisco NetFlow as the connection tracker. Based on what was observed and our gained knowledge during the implementation of the model, we consider the following requirements necessary for a real-life model to be a viable approach.

6.6.1 The Model

- The model needs to consider network environments consisting of high edge degree nodes and cyclic paths.
- The model needs to consider network environments consisting of heterogeneous devices, and understand these in terms of their purpose and decisions these make.
- The model needs to acquire flows from all network devices necessary to track connections throughout the network environment.
- The model needs to build connection tracking (primarily) around a technology that do not utilize sampling techniques.
- The model needs to consider packet forwarding taking place on both OSI layer-2 and OSI layer-3 (e.g. Link-layer vs. IGPs/802.1Q (VLAN tagging)).
- The model needs to revolve around packet-flow and not session-flow as the latter may reduce the accuracy of the correlation process (c.f. persistent connections and parallel transfers).
- The model needs to relate to log sources that do not provide any apparent connection-related keys, which may break the entire connection tracking.
- The model needs to consider connection tracking on OSI layer-3 to extend the coverage of attack classes.
- The model needs to dynamically build an asset base to improve impact analysis, via traffic learning or external software support.
- The model needs to relate to TCP-fragments, and in particular intermediate network equipment that do not assemble all fragments.
- The model needs to relate to OSI layer-3 and OSI layer-4 translation techniques, such as NAT.

6.6.2 Connection Tracking

- Flows should have a 1:1 mapping to application-layer content such as request/response (or at least some relationship to the application-layer in general).
- Flows and intrusion alerts need to have shared keys and be uniquely related based on these keys.
- Flows and intrusion alerts need to be related even if the intrusion is only taking place on OSI layer-3.
- Flow-exporters should be configured to use all supported tuples (c.f. Cisco NetFlow), and

preferably extended to include additional tuples.

- Flows from independent sessions must not collide on high-load hosts, even in terms of historical data.
- Flow-exporters should utilize a connection-oriented transport protocol (i.e. TCP), and export only those flows that are needed to track an intrusion alert.
- Flow-timers need to be configured to an absolute minimum to export flows close to real-time.
- Flow-timers should be adjusted to match the traffic load and TCP-segment allocation to avoid conflicts.

6.6.3 Algorithm (Correlation Process)

- The algorithm must be able to either interpret IDS signatures or to be able to put IDS alerts into a context.
- The algorithm should be able to correlate or combine independent intrusion alerts that concern the same intrusion (at least in the summary).
- The algorithm must deal with contradicting correlation outcomes, showing correlated intrusions as both verified/inconclusive.
- The algorithm should understand the decision made by the protection gates, and not only the binary decision to network traffic.
- The algorithm needs to consider intermediate hosts/servers such as proxies, which may act as both source and destination for a single session.
- The algorithm should have some interpretation of the attack vector used in an intrusion or intrusion class.
- The algorithm needs to support numerous log formats, which is influenced by the types of devices used in the network environment.

7 Future Work

This chapter contains proposals for future work, which is related to the proposed model, the method used, and building a network SIEM solution in general.

- An important aspect that needs to be addressed, is the relationship between the alert intrusions raised by the central IDS and the actual log entries on the end clients. One approach is to look at the actual mapping between IDS alerts and specific log entries. Another is to understand what types of log entries are relevant for various types of attack classifications. Section 2.1.2 discussed some work on the area.
- In terms of log sources, we need to acquire a better overview of how application logs influence the model, and how application logs may be mapped to flows, or assigned identifiers that allow dynamic mapping to take place. A particular problem we encountered in this respect, was the lack of common identifiers between various levels of logs, making it difficult to relate them. Proxies for instance, acting as mediators, use two independent flows per session which cannot be related without identifiers.
- Other methods that may replace Cisco NetFlow ver. 5 should be reviewed in order to understand the characteristics of such methods in relation to the model (disregarding sampling technologies as discussed in Section 2.2.1). IPFIX, aka NetFlow ver. 10, should be reviewed in relation to the model as it provides functionality that may prove to be more suitable. The NetFlow Secure Event Logging (NSEL)[2] used by Cisco ASA 5580, also looks promising as it maintains state of flows between ingress and egress, and triggers on events that cause state change in the flows (i.e. event-driven).
- The proposed model was applied with a signature-based IDS, allowing 1:1 mapping between flows and IDS alerts to take place. It would be interesting to study the behavior of anomaly-based IDS in relation to the model, where multiple flows are extracted from a single anomaly. This may also include flow level traffic measurement to complement the signature-based attack coverage.
- Our network environment did not include routing protocols as the network environment was considered too small. It will be necessary to see how routing protocols and routing tables affect the model, both in terms of dynamic and static layout. We would for instance assume that static routing tables would replace the need for flows in those protection domains this applies to.
- We discussed the problem of network traffic only consisting of OSI level-3 and below, and how it was not possible to relate the traffic directly to the application or stream. We think methods should be reviewed to relate low-layered connections with streams.

- There are also several topics that have not been considered in this study, and which deserve attention. This includes load-balanced networks, IP-fragmented networks where fragments are not reassembled by intermediate routers, and the correlation of flows where either central data is missing (which may be the case when the amount of NetFlow overwhelms the receiver [66]) or where load-balancing has separated the traffic.

8 Conclusions

This thesis is concerned with the study of a network-centric log correlation model, whose purpose is to answer the research question *Can we improve SIEM by making it network centric?*. To substantiate our research, we proposed a conceptual model and then studied that model in a common network environment by using Cisco NetFlow. Our answer to the research question is largely influenced by this chosen method, in addition to the network composition and performed network attacks.

The proposed model has many common denominators with today's practice for a segmented network topology, and its sheer focus on the dynamic aspect of a network, supports the research question, which was also observed in the experiment. By applying this model in a simulated real-life environment, we were able to reduce the amount of log data to a more manageable size. We believe that this may be beneficial both in terms of efficiency for a SIEM solution and the level of work required by a security analyst to investigate a particular incident. Furthermore, it allowed us to establish an accurate timeline in terms of network traffic for a given event, providing a reliable trace of all network devices involved in the incident, as well as providing a context on the incident itself. We were also able to significantly reduce the number of IDS alerts that were raised, which consequently reduces the amount of alerts requiring human intervention. Research concerning CIDS or alert correlation may reduce this even further [15, 56, 59].

Despite of the promising prospects on the model itself, observations showed that actual method we used, Cisco NetFlow, may be a suboptimal approach for tracking network flow within a security context as discussed in this thesis. The method had several drawbacks in relation to a network-centric approach, which primarily is based on the characteristics of the protocol and its applicability in a wide set of attack classifications. With the former, we refer to the method for generating flows, variables that influences flows (e.g. timeout-values), and the number of traffic identifiers constituting a flow (e.g. collisions). With attack classifications, we refer to the issue that flows do not provide enough characteristics for attacks taking place on OSI layer-3 and below, and that some protocols such as HTTP and FTP, support techniques such as *persistency* and *pipelining*, making it difficult to precisely identify a particular attack-stream.

We also encountered transparency-issues between high-level logs and flows, which required us to facilitate mapping between application-logs and network-logs by providing a shared key. The inconsistency between log formats with different focus is expected and stipulates that the model needs to consider alternative approaches, or facilitate, on those central hosts this concerns. Proxies, DNS' and antivirus-solutions are typical examples of this.

Conclusively, we believe that our work shows that it is possible to improve SIEM by making it network centric in relation to our conceptual model, but that the applicability of Cisco NetFlow ver. 5 in this matter, may be problematic. To move the field forward, we have proposed several aspects of the model that requires attention as further work and presented a set of requirements that the model needs to fulfill.

Bibliography

[1] Yusof, R., Sulamat, S. R., & Sahib, S. 2008. Intrusion Alert Correlation Technique Analysis for Heterogeneous Log. In *International Journal of Computer Science and Network Security, Volume 8, No.9*.

[2] Paulauskas, N. & Garsva, E. June 2006. Computer System Attack Classification. *Electronics and Electrical Engineering 2*, 2(66), 84–87.

[3] Igure, V. & Williams, R. June 2008. Taxonomies of attacks and vulnerabilities in computer systems. *Communications Surveys Tutorials, IEEE*, 10(1), 6–19.

[4] Kent, K. & Souppaya, M. Guide to Computer Security Log Management. NIST Special Publication 800-92, National Institute of Standards and Technology (NIST), September 2006.

[5] Donnet, B. & Friedman, T. December 2007. Internet Topology Discovery: a Survey. *IEEE Communications Surveys and Tutorials*, 9(4), 2–15.

[6] Huang, W., Zhou, Y., Yu, B., Tang, W., & Beedgen, C. F. March 2010. Storing log data efficiently while supporting querying. WO/2010/028279, United States Patent and Trademark office.

[7] Karlzen, H. An Analysis of Security Information and Event Management Systems - The Use of SIEMs for Log Collection, Management and Analysis. Master's thesis, Chalmers University of Technology, Department of Computer Science and Engineering, Gothenburg, Sweden, January 2009.

[8] Lane, A. June 2010. Understanding and Selecting a SIEM/LM: Correlation and Alerting. http://securosis.com/blog/understanding-and-selecting-a-siem-lm-correlation-and-alerting. (Last visit: 14.12.2010).

[9] Prasad, G. S., Reddy, N. V. S., & Acharya, U. D. 2010. Knowledge Discovery from Web Usage Data: A Survey of Web Usage Pre-processing Techniques. In *Information Processing and Management*, volume 70 of *Communications in Computer and Information Science*, 505–507. Springer Berlin Heidelberg.

[10] Peng, S. & Cheng, Q. December 2009. Research on Data Preprocessing Process in the Web Log Mining. In *Information Science and Engineering (ICISE), 2009 1st International Conference on*, 942–945.

[11] Andersson, D., Fong, M., & Valdes, A. June 2002. Heterogeneous Sensor Correlation: A Case Study of Live Traffic Analysis. Presented at IEEE Information Assurance and Security Workshop, United States Military Academy, West Point, NY.

[12] Hätälä, A., Särs, C., Addams-Moring, R., & Virtanen. 2004. Event data exchange and intrusion alert correlation in heterogeneous networks. In *Proceedings of the 8th Colloquium for Information Systems Security Education (CISSE), Westpoint, NY, CISSE*, 84–92.

[13] Abad, C., Taylor, J., Zhou, Y., Sengul, C., Rowe, K., & Yurcik, W. 2003. Log Correlation for Intrusion Detection: A Proof of Concept. In *Proceedings of the 19th Annual Computer Security Applications Conference, Las Vegas*. ACSAC.

[14] Jakobson, G., Weissman, M., Brenner, L., Lafond, C., & Matheus, C. 2000. GRACE: building next generation event correlation services. In *Network Operations and Management Symposium (NOMS), IEEE/IFIP*, 701–714.

[15] Valeur, F., Vigna, G., Kruegel, C., & Kemmerer, R. September 2004. A Comprehensive approach to intrusion detection alert correlation. *Dependable and Secure Computing, IEEE Transactions*, 1(3), 146–169.

[16] Herrerias, J. & Gomez, R. April 2007. A Log Correlation Model to Support the Evidence Search Process in a Forensic Investigation. In *Systematic Approaches to Digital Forensic Engineering, 2007. SADFE 2007. Second International Workshop on*, 31–42.

[17] Grer, D. W., Khan, I., Ogier, R., & Keffer, R. 1996. An Artificial Intelligence Approach to Network Fault Management. SRI International, Menlo Park, CA.

[18] Hellerstein, J. L., Ma, S., & Perng, C.-S. 2002. Discovering actionable patterns in event data. *IBM Systems Journal*, 41(3), 475–493.

[19] Sadoddin, R. & Ghorbani, A. 2006. Alert correlation survey: framework and techniques. In *PST '06: Proceedings of the 2006 International Conference on Privacy, Security and Trust*, 1–10, New York, NY, USA. ACM.

[20] Beckers, J. & Ballerini, J. P. 2003. Advanced Analysis of Intrusion Detection Logs. *Computer Fraud & Security*, 2003(6), 9–12.

[21] Azarkasb, S. & Ghidary, S. June 2009. New approaches for intrusion detection based on logs correlation. In *Intelligence and Security Informatics, 2009. ISI '09. IEEE International Conference on*, 234–234.

[22] Porras, P., Fong, M., & Valdes, A. 2002. A Mission-Impact-Based Approach to INFOSEC Alarm Correlation. In *Recent Advances in Intrusion Detection*, volume 2516 of *Lecture Notes in Computer Science*, 95–114. Springer Berlin / Heidelberg.

[23] Cisco Systems Inc. 2005. Rationalizing Security Events with Three Dimensions of Correlation. http://www.cisco.com/en/US/prod/collateral/vpndevc/ps5739/ps5209/prod_white_paper0900aecd802c1c5b.html. (Last visit: 14.12.2010).

[24] Zhai, Y., Ning, P., Iyer, P., & Reeves, D. December 2004. Reasoning about complementary intrusion evidence. In *Computer Security Applications Conference, 2004. 20th Annual*, 39–48.

[25] Rinnan, R. Benefits of Centralized Log file Correlation. Master's thesis, Gjøvik University College, Department of Computer Science and Media Technology, Januar 2005.

[26] Zhai, Y., Ning, P., & Xu, J. 2006. Integrating IDS Alert Correlation and OS-Level Dependency Tracking. In *Intelligence and Security Informatics*, volume 3975 of *Lecture Notes in Computer Science*, 272–284. Springer Berlin / Heidelberg.

[27] Toprak, M. Intrusion detection system alert correlation with operating system level logs. Master's thesis, İzmir Institute of Technology: Computer Engineering, December 2009.

[28] Breitbart, Y., Garofalakis, M., Jai, B., Martin, C., Rastogi, R., & Silberschatz, A. 2004. Topology discovery in heterogeneous IP networks: the NetInventory system. *IEEE/ACM Trans. Netw.*, 12(3), 401–414.

[29] Bejerano, Y. 2009. Taking the skeletons out of the closets: a simple and efficient topology discovery scheme for large ethernet LANs. *IEEE/ACM Trans. Netw.*, 17(5), 1385–1398.

[30] Myers, J. A Dynamically Configurable Log-based Distributed Security Event Detection Methodology using Simple Event Correlator. Master's thesis, Air force Institute of Tech Wright-Patterson AFB OH graduate school of engineering and management, June 2010.

[31] Claise, B., Sadasivan, G., Valluri, V., & Djernaes, M. October 2004. Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational).

[32] Trammell, B., Boschi, E., Mark, L., Zseby, T., & Wagner, A. October 2009. Specification of the IP Flow Information Export (IPFIX) File Format. RFC 5655 (Proposed Standard).

[33] Cisco, I. oct 2007. Introduction to Cisco IOS NetFlow - A Technical Overview. http://www.cisco.com/en/US/products/ps6601/productswhitepaper0900aecd80406232.shtml, (Last visit: 29.05.2011).

[34] Zhenqi, W. & Xinyu, W. 2008. Netflow based intrusion detection system. In *Proceedings of the 2008 International Conference on MultiMedia and Information Technology*, 825–828, Washington, DC, USA. IEEE Computer Society.

[35] Quittek, J., Zseby, T., Claise, B., & Zander, S. October 2004. Requirements for IP Flow Information Export (IPFIX). RFC 3917 (Informational).

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

[37] Abbott, R. P., Chin, J. S., Donnelley, J. E., Konigsford, W. L., Tokubo, S., & Webb, D. A. Security Analysis and Enhancements of Computer Operating Systems. Technical report, Institute for Computer Sciences and Technology, April 1976.

[38] Jin, S., Wang, Y., Cui, X., & Yun, X. October 2009. A review of classification methods for network vulnerability. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, 1171–1175.

[39] Amoroso, E. G. April 1994. *Fundamentals of Computer Security Technology*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, ISBN 0-13-108929-3.

[40] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. 2009. A detailed analysis of the KDD CUP 99 data set. In *Proceedings of the Second IEEE international conference on Computational intelligence for security and defense applications*, CISDA'09, 53–58, Piscataway, NJ, USA. IEEE Press.

[41] McHugh, J. November 2000. Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Trans. Inf. Syst. Secur.*, 3, 262–294.

[42] Brugger, S. T. 2008. KDD Cup '99 dataset considered harmful. University of California - Davis, Department of Computer Science, http://www.bruggerink.com/~zow/GradSchool/KDDCup99Harmful.html, (Last visit: 21.05.2011).

[43] NSS labs, Inc. august 2010. Network Intrusion Prevention Systems - Individual Product Test Results - Palo Alto Networks PA-4020. Methodology Version: 6.0.

[44] NSS labs, Inc. march 2010. Network Intrusion Prevention Systems - Individual Product Test Results - NSFOCUS Network IPS 1200. Methodology Version: 6.0.

[45] Lindqvist, U. & Jonsson, E. 1997. How to Systematically Classify Computer Security Intrusions. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, SP '97, 154–163, Washington, DC, USA. IEEE Computer Society.

[46] Meadows, C. 1993. An outline of a taxonomy of computer security research and development. In *Proceedings on the 1992-1993 workshop on New security paradigms*, NSPW '92-93, 33–35, New York, NY, USA. ACM.

[47] Álvarez, G. & Petrovic, S. 2003. A new taxonomy of Web attacks suitable for efficient encoding. *Computers & Security*, 22(5), 435–449.

[48] Bishop, M. December 2002. *Computer Security: Art and Science*. Addison-Wesley Professional, 1st edition, 9th printing, ISBN 0-201-44099-7.

[49] Mirkovic, J. & Reiher, P. April 2004. A taxonomy of DDoS attack and DDoS defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34, 39–53.

[50] Wood, A. & Stankovic, J. October 2002. Denial of service in sensor networks. *Computer*, 35(10), 54–62.

[51] Burgess, M. 2004. *Principles of Network and System Administration*. John Wiley & Sons, Ltd, 2nd edition, ISBN 0-470-86807-4.

[52] Moy, J. April 1998. OSPF Version 2. RFC 2328 (Standard). Updated by RFC 5709.

[53] Altın, A., Fortz, B., Thorup, M., & Ümit, H. 2009. Intra-domain traffic engineering with shortest path routing protocols. *4OR: A Quarterly Journal of Operations Research*, 7, 301–335.

[54] Wong, A. & Yeung, A. April 2009. *Network Infrastructure Security*. Springer Publishing Company, Inc., ISBN 978-1441901651, 1st edition.

[55] Lodin, S. & Schuba, C. February 1998. Firewalls fend off invasions from the net. *Spectrum, IEEE*, 35(2), 26–34.

[56] Zhou, C. V., Leckie, C., & Karunasekera, S. 2009. Decentralized multi-dimensional alert correlation for collaborative intrusion detection. *Journal of Network and Computer Applications*, 32(5), 1106–1123. Next Generation Content Networks.

[57] Morin, B., Mé, L., Debar, H., & Ducassé, M. 2009. A logic-based model to support alert correlation in intrusion detection. *Information Fusion*, 10(4), 285–299. Special Issue on Information Fusion in Computer Security.

[58] Almgren, M., Lindqvist, U., & Jonsson, E. 2008. A Multi-Sensor Model to Improve Automated Attack Detection. In *Recent Advances in Intrusion Detection*, volume 5230 of *Lecture Notes in Computer Science*, 291–310. Springer Berlin / Heidelberg.

[59] Taha, A. E., Ghaffar, I. A., Bahaa Eldin, A. M., & Mahdi, H. M. K. May 2010. Agent based correlation model for intrusion detection alerts. In *Intelligence and Security Informatics (ISI), 2010 IEEE International Conference on*, 89–94.

[60] VMware community discussion. June 2010. Duplicate pings between guest and host when using bridged networking. http://communities.vmware.com/thread/274555. (Last visit: 21.05.2011).

[61] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., & Lear, E. February 1996. Address Allocation for Private Internets. RFC 1918 (Best Current Practice).

[62] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. June 1999. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard). Updated by RFCs 2817, 5785.

[63] TechCenter, W. S. May 2005. Transmission Control Protocol/Internet Protocol (TCP/IP), Windows 2003 with SP1. http://technet.microsoft.com/en-us/library/cc759700%28WS.10%29.aspx. (Last visit: 29.05.2010).

[64] Gorton, D. Extending Intrusion Detection with Alert Correlation and Intrusion Tolerance. Master's thesis, Chalmers University of Technology, Department of Computer Science and Engineering, Gothenburg, Sweden, October 2003.

[65] Kobayashi, A. & Claise, B. August 2010. IP Flow Information Export (IPFIX) Mediation: Problem Statement. RFC 5982 (Informational).

[66] Hu, Y., Chiu, D. M., & Lui, J. C. 2006. Adaptive flow aggregation - a new solution for robust flow monitoring under security attacks. In *In IEEE/IFIP Network Operations and Management Symposium (IEEE/IFIP NOMS 2006)*, 424–435. IEEE Computer Society.