HEURISTICS FOR IMPROVED ENTERPRISE INTRUSION DETECTION

A Dissertation
Presented to
the Faculty of Engineering and Computer Science
University of Denver

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

by
James J. Treinen
June 2009
Advisor: Ramakrishna Thurimella

The dissertation of James J. Treinen is approved.

Dr. Ramakrishna Thurimella

Dr. Chris GauthierDickey

Dr. Christian Grothoff

University of Denver

June 2009

Author: James J. Treinen
Title: HEURISTICS FOR IMPROVED ENTERPRISE
INTRUSION DETECTION
Advisor: Ramakrishna Thurimella
Degree Date: June 2009

Abstract

One of the greatest challenges facing network operators today is the identification of malicious activity on their networks. The current approach is to deploy a set of intrusion detection sensors (IDSs) in various locations throughout the network and on strategic hosts. Unfortunately, the available intrusion detection technologies generate an overwhelming volume of false alarms, making the task of identifying genuine attacks nearly impossible. This problem is very difficult to solve even in networks of nominal size. The task of uncovering attacks in enterprise class networks quickly becomes unmanageable.

Research on improving intrusion detection sensors is ongoing, but given the nature of the problem to be solved, progress is slow. Research simultaneously continues in the field of mining the set of alarms produced by IDS sensors. Varying techniques have been proposed to aggregate, correlate, and classify the alarms in ways that make the end result more concise and digestible for human analysis. To date, the majority of these techniques have been successful only in networks of modest size. As a means of extending this research to real world, enterprise scale networks,

we propose 5 heuristics supporting a three-pronged approach to the systematic evaluation of large intrusion detection logs. Primarily, we provide a set of algorithms to assist operations personnel in the daunting task of ensuring that no true attack goes unnoticed. Secondly, we provide information that can be used to tune the sensors which are deployed on the network, reducing the overall alarm volume, thus mitigating the monitoring costs both in terms of hardware and labor, and improving overall accuracy. Third, we provide a means of discovering stages of attacks that were overlooked by the analyst, based on logs of known security incidents.

Our techniques work by applying a combination of graph algorithms and Markovian stochastic processes to perform probabilistic analysis as to whether an alarm is a true or false positive. Using these techniques it is possible to significantly reduce the total number of alarms and hosts which must be examined manually, while simultaneously discovering attacks that had previously gone unnoticed. The proposed algorithms are also successful at the discovery of new profiles for multi-stage attacks, and can be used in the automatic generation of meta-alarms, or rules to assist the monitoring infrastructure in performing automated analysis. We demonstrate that it is possible to successfully rank hosts which comprise the vertices of an Alarm Graph in a manner such that those hosts which are of highest risk for being

involved in attack are immediately highlighted for examination or inclusion on hot lists. We close with an evaluation of 3 sensor profiling algorithms, and show that the order in which alarms are generated is tightly coupled with whether or not they are false positives. We show that by using time based Markovian analysis of the alarms, we are able to identify alarms which have a high probability of being attacks, and suppress more than 90% of false positives.

Acknowledgements

I would like to thank my advisor, Dr. Ramakrishna Thurimella, and my thesis committee, Dr. Chris GauthierDickey, and Dr. Christian Grothoff for your advice and guidance during the course of my research. I would like to thank my mentor, Dr. Joan Mitchell, for her inspiration and sage wisdom.

To my mangers at IBM, Gary Grizzard, Chris Calvert, Chris Triolo, Mary Karnes and Doug Smith thank you for providing a flexible work environment. To my colleagues, Chris Bielinski, and Ken Farmer, thanks for assisting with data issues on top of your already full schedules. To Nick Essner, Jeff Lahann, Jesse Emerson and Darren Lawless, thank you for reviewing and validating my results.

To my parents, Tish and Steve, I would like to thank you both for teaching that learning is a life long process.

Finally, I would like to thank my wife, Tricia, for all of her support and sacrifice. Completing a PhD is tremendously time consuming, even more so when done while while holding down a full time job. There is no way I would have been able to do both had it not been for you.

Dedication

To Tricia

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The goal of information security is to protect the so-called “Big Three” tenets of a secure environment, namely C.I.A, or Confidentiality, Integrity, and Availability. In support of these goals, significant resources must be dedicated to the protection of assets which house sensitive data. A subset of this problem is the detection of attempts to reverse the C.I.A triad to D.A.D, or Disclosure, Alteration, and Destruction [44].

The problem of network security continues to receive increased coverage in the global media. Cyber terrorism, information warfare, and extortion are replacing the script kiddies and relatively benign hackers from the early days of the Internet. Theft of personal information has become a cottage industry, proving tremendously lucrative for skilled thieves, and expensive, if not devastating to the individual victims, as well as the the corporations who are responsible for the loss. The cost in reputation to a company who falls victim to a highly publicized attack is enough to send their stock price tumbling. The increased frequency of high profile security breaches has prompted a corresponding increase in regulations requiring improved cyber-security measures. This in turn has forced institutions of all sizes to increase their investment in defending themselves from the constant onslaught of attacks. The investment is generally made in the form of deploying complex intrusion detection infrastructures, the core of which are network or host based Intrusion Detection Sensors (IDSs).

The unfortunate reality is that existing intrusion detection technologies produce a disproportionately high volume of false alarms. While this is troublesome for small networks, the problem becomes intractable in large networks. The task of separating false alarms from alarms representing genuinely malicious traffic in a collection of large networks, most notably at large Managed Security Service Providers (MSSPs), can quickly become overwhelming.

Intrusion detection, as a field, is not an exact science. At best, intrusion detection in its current state is comprised of a set of heuristics implemented by a staff of well trained, highly skilled analysts, assisted by competent tooling [28]. Given this fact, the best we can hope for is to provide incremental improvements in the heuristics and tools that are available to the Security Operations Center (SOC) staff. It is this task that we have undertaken during our research. As a result of our efforts we have developed a set of tools which improve the capability of the SOC analysts in their fight to protect our networks.

1.2 Problem Statement

A major problem faced by those who deploy current intrusion detection technology is the large number of false alarms generated by IDSs, which can be well over 90% [38, 39, 40].

As noted by Lippmann, et al. in [50], the deployment of an inaccurate Intrusion Detection System (IDS) can have undesirable effects in addition to simply missing certain types of attacks. The first of these is the potential to reduce the level of vigilant monitoring by security operations staff, due to the false sense of security provided by the IDS. Secondly, using operations staff to examine all of the alarms produced in a day can make the deployment of a typical IDS system extremely expensive in terms of support and labor costs. These issues are further compounded in large intrusion detection infrastructures where the number of managed sensors can easily reach into the thousands, *generating millions of alerts per day.*

Large intrusion detection systems warrant full time staff dedicated to defending the network against compromise. The security staff is generally deployed in a Security Operations Center (SOC), where all aspects monitoring the IDS infrastructure are centralized.

The context for our experiments is the SOC of a large Managed Security Service Provider (MSSP). Generally, this environment is comprised of many thousands of IDS sensors installed across a large number of customer networks. Our experiments were conducted on a production data set that was generated by roughly 1,000 IDS sensors. The sensor technologies used to generate the data set represented multiple vendors and versions of their

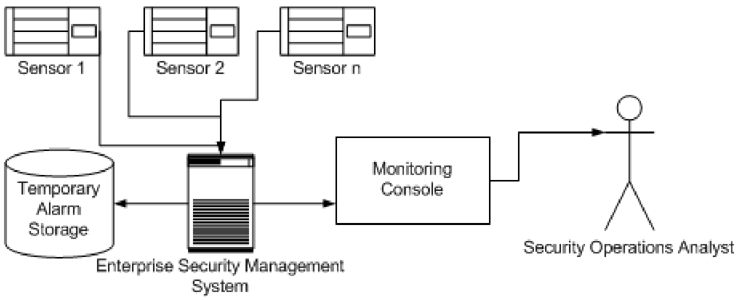


Figure 1.1: A typical security operations center

software, and were installed across 135 distinct customer networks. The alarm logs generated by the sensors were consolidated at the SOC which employed a third party Enterprise Security Manager (ESM). In addition to consolidating the alarms as they arrived at the SOC, the ESM had the ability to perform automated analysis of the alarm stream. The ESM included engines to perform correlation and aggregation of alarms, as well as rule based analysis. Rule based analysis is a type of meta-signature based system to seek out patterns of alarms which have a higher probability of being genuine attacks.

The tool set developed during the course of our research is meant to augment the capabilities of the ESM, extending both its ability, and the ability of the analysts who work in the SOC, to suppress false alarms while highlighting true attacks. We concentrate on illuminating full attack profiles, as the majority of attacks are not comprised of single actions, but rather actions which result in multiple alerts being generated and routed back to the SOC.

1.3 The Knowledge Discovery Process

In order to be truly effective, the use of data mining techniques must be one step in an overall Knowledge Discovery in Databases (KDD) process. This case is made repeatedly in the literature, e.g. [58] who use cluster analysis solely as the initial step in their data exploration. It is reiterated in [37, 38, 39, 40] that although the research tends to focus on the mining algorithm employed, it is only one step in the overall KDD process. It is also noted that without all of these steps, data mining runs a high risk of finding meaningless or uninteresting patterns. It is for this reason that [83] propose their end-to-end KDD architecture. Julisch outlines the basic KDD steps as follows in [38], as condensed from their original definition in [24] :

1. Understand the application domain
2. Data integration and selection
3. Data mining
4. Pattern evaluation
5. Knowledge presentation

A similar outline is made in [58], who also note that once a group of domain experts is consulted, the entire process should be automated to the extent that is possible.

The KDD process highlights the importance of the SOC analyst. Throughout the course of our experiments we worked continuously with SOC personnel to ensure that the findings produced during our experiments were of high quality, and that they provided information which was useful and actionable by the security staff. By refining our techniques based on the SOC feedback we were able to greatly improve the quality of our results.

1.3.1 Data Collection

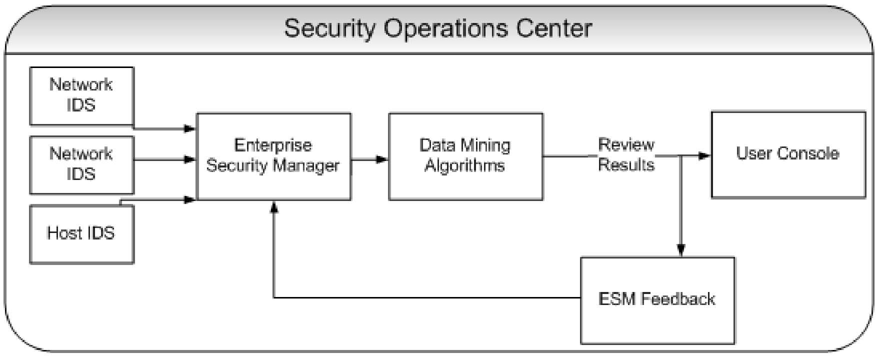


Figure 1.2: Data flow for IDS data mining

The data flow required for our analysis is illustrated in Figure 1.2. Alarms are generated by a set of intrusion detection sensors (IDSs) and are collected at a central Enterprise Security Manager (ESM) which consolidates them for display in the Security Operations Center (SOC). Alarms are stored temporarily in a database on the ESM, and are periodically extracted and stored permanently in a data warehouse. The data warehouse was custom built to facilitate off-line analysis. The set of alarms used during our analysis is automatically loaded via a query from the ESM to the data warehouse, eliminating any need for manual intervention. Data is fed automatically to the data mining algorithms via a set of queries which are executed against the data warehouse. The results are then displayed on a user console for use by the SOC analyst.

1.4 Data Structures

Since their introduction, Attack Graphs have received considerable attention as a way to model the vulnerabilities of an enterprise network. These graphs model the paths that an attacker could take in order to successfully compromise a target. Naïve representations typically result in models that grow exponentially in the number of possible states. Because the resulting graphs are unwieldy even for small networks, recent research has focused on reducing their visual complexity and making them tractable for computational purposes [33, 63].

We propose *Alarm Graphs*, an alternative to Attack Graphs, that are built from the alarms produced by the sensors comprising the monitoring infrastructure. Alarm Graphs are used as the basic data structure for our first two algorithms. In their basic form, they are used to assist in the creation of high quality rules to be installed in the ESM for increased accuracy of automated monitoring. By augmenting the alarm graphs with knowledge of known attacks, we are then able to perform a more complete analysis regarding the extent of attacks, and predict which hosts have increased likelihood of being involved in future attacks.

1.4.1 Alarm Graphs : Modeling Alarms as Directed Graphs

Definition 1.4.1. *The set of all intrusion detection alarms A is a set of 5-tuples $a = \langle t, s, d, g, n \rangle$ which capture the information contained in an IDS*

Sensor Type	**Source IP**	**Destination IP**	**Signature**	**Count**
Network	10.0.0.1	10.0.0.3	Share Enumeration	500
Network	10.0.0.1	10.0.0.3	Remote Buffer Overflow	300
Network	10.0.0.2	10.0.0.3	Remote Buffer Overflow	300
Network	10.0.0.3	10.0.0.4	Share Enumeration	100
Host	10.0.0.4	10.0.0.4	Brute Force Login Attempt	1

Table 1.1: Typical intrusion detection alarms

alarm.

Each $a \in A$ is comprised of the sensor type t, either host based or network based; the source IP address of the attack s; the destination, or target IP of the attack d; the alarm signature g which describes the perceived malicious activity; and a count n describing the number of times this combination repeats. This information is stored as a table in the data warehouse, and is easily retrievable.

Definition 1.4.2. *An* Alarm Graph *models the set of alarms A as a directed graph $G = (V, E)$. The set of vertices represents the IP space of A, and the set of edges models the set of detected alarms between the various IP addresses.*

Using the set of alarms A, we generate a directed graph $G = (V, E)$. We define S as the set of distinct source IP addresses, and D as the set of distinct destination IP addresses. The set of vertices $V = S \cup D$, such that each $v \in V$ represents an IP address from the set of alarms A. It is important to note that S and D are not disjoint, and in fact $S \cap D$ can make up a large percentage of the overall IP space. A directed edge $e \in E = (s, d)$ is drawn corresponding with the direction of the perceived attack. We deduce the direction of each alarm from the source IP to the destination IP address. The directed graph $G = (V, E)$ is then generated such that each IP address in the alarm set is represented as a vertex in the graph, and each edge represents the detection of one or more detected alarms between the two vertices. Alarms which are triggered by Host Intrusion Detection Sensors (HIDS), where the sensor resides on the machine being attacked, are denoted as self-loops, as the source IP address is not captured by this type of sensor. For the purposes of our analysis, the raw alarm data shown in Table 1.1 is summarized by the adjacency function $f_G : S \times D \rightarrow \{0, 1\}$. We define the adjacency function f_G such that if for any $s \in S, d \in D$ an alarm is triggered by the IDS, a corresponding entry exists $f_G(s, d) = 1$, representing the directed edge $e = s \rightarrow d \in E$. Or,

$$f_G(s, d) = \begin{cases} 1 & \text{if an alarm is triggered from s to d;} \\ 0 & \text{otherwise.} \end{cases}$$

The alarms are summarized such that independent of how many alarms are triggered between distinct pairs of hosts, only one edge is drawn. The rationale behind this approach is that given the high volume of false alarms, the structure that describes the alarm flow is more important than the actual volume. This sentiment echoes Chakrabarti, et al. [14] who note during their analysis of web graphs that the link structure of the web implies underlying social networks. We extend this concept to the social structures implied by the connections illuminated by Alarm Graphs. Understanding this link structure provides an effective means of discovering attacks that would have otherwise gone unnoticed. The results are such that the IDS alarms which are shown in Table 1.1 are modeled as the directed graph shown in Figure 1.3.

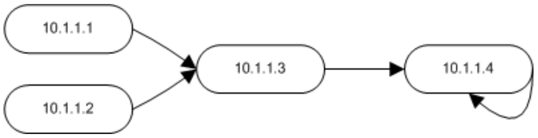


Figure 1.3: Intrusion detection alarms from Table 1.1 as a directed graph

1.5 Thesis Statement

Based on our research, we have found the following to be true:

1. The majority of attacks are comprised of multiple phases. By discovering recurring patterns in IDS alerts, it is possible to create meta-alarms which reflect the overarching structure of an end-to-end attack.

2. The alarms generated by an IDS have an underlying structure which is useful in creating improved intrusion detection heuristics. It is possible to exploit this structure to discover previously overlooked portions of an attack.

3. IDS sensors by their nature produce a baseline of false positive noise. By modeling this baseline and detecting deviations from it, it is possible to direct resources to determine the cause of the change.

4. The order in which alarms are generated by an IDS sensor is tightly coupled to whether those alarms are false positives, or whether they are reflective of a genuine attack.

1.6 Novel Contributions

Specifically, this research supports the following goals:

- **Objective Risk Assessment.** When faced with the task of monitoring large networks, it is easy for human analysts to develop tunnel vision, narrowing their attention to a subset of hosts such as web servers which are commonly known to be involved in attacks. In comparison, by ranking alarm graphs we provide a facility for an analyst to objectively assess the risk of all hosts and not lose sight of the "big picture". We accomplish this by calculating the effect known attacks have on the remaining portions of the graph.

- **Systematic Identification of Missed Attacks.** As mentioned above, it is easy for security analysts to fixate on a subset of hosts during their manual analysis. In contrast, our approach provides a comprehensive analysis of the network, and reports likely extensions of a known attack. This data is invaluable for forensics and intrusion prevention. It is worth noting that when our algorithm was run against historic intrusion data, it identified compromised nodes that were missed by security personnel.

- **Automated Watch List Generation.** The output generated by our ranking analysis is a list of those hosts which have higher probability of being involved in future attacks. By paying close attention to the hosts on this list, security monitoring teams can increase their efficiency. Here again, our algorithm predicted surprisingly high number of attacks when run against historic intrusion data. For exact numbers, see Section 4.4.4.

- **Sensor Tuning.** During the course of our analysis, we found that certain hosts and sensors were repeatedly flagged for inspection, but were not involved in true attacks. These nodes generally produced large volumes of false alarms, creating a high noise level in the alarm stream. Our algorithms provide cues that are useful in creating filters to remove this noise, thus decreasing the overall cost associated with running the monitoring infrastructure, while increasing the overall fidelity of the alarm stream.

- **Visualization.** Alarm Graphs can be visualized using tools such as GraphViz [7]. Because the alarms are reduced to a single link between distinct hosts, as opposed to full enumeration of the alarm log, visualizations produced are compact and intuitive to a human analyst. Using the proposed sensor profiling tools in Chapter 5, we are able to provide near real time visualizations of sensors which deviate significantly from their normal behavior.

- **Alarm Suppression.** By modeling the order in which a sensor typically generates its baseline false alarm traffic, we show that we can detect actual attacks as deviations from this profile. This has the net effect of suppressing over 90 % of the false alarms that were generated by the sensors, and greatly reducing the workload on the SOC staff.

1.7 Outline

The remainder of this dissertation is organized as follows. Chapter 2 presents background information on the application of data mining to the field of intrusion detection, as well as an overview of the state of the art regarding intrusion detection sensors. Chapter 3 outlines a novel technique for the automatic generation of meta-rules based on recurring patterns of alarms in the IDS data stream. Chapter 4 outlines a novel technique for determining the extent to which an attack effects the other hosts in the network by ranking the nodes in Alarm Graphs. In chapter 5 we evaluate a set of sensor profiling techniques that act as a meta-anomaly detector over the set of alarms generated by a sensor and show that we can accurately detect attacks while simultaneously suppressing over 90% of the false alarms. Concluding remarks are presented in chapter 6.

Chapter 2

Background

2.1 Overview of Intrusion Detection Systems

Many attempts have been made to improve the accuracy of intrusion detection sensors. Ultimately, these can be placed in one of two categories:

1. Misuse Detection Systems. Misuse based IDS relies heavily on signatures of known malicious activity, and are deployed both as host and network based IDS, as well as virus detection systems.

2. Anomaly Detection Systems. Anomaly based IDS typically create a baseline for the normal behavior of a host, and attempt to classify any deviation from this baseline as either benign or malicious.

The main difference between anomaly and misuse based systems is that anomaly based systems train the model to reflect the normal behavior of the system and misuse detection systems define malicious behavior a priori, and then scan for instances of known malicious activities. Both types have their strengths and weaknesses. Misuse detection systems have the advantage of knowing what they are looking for up front. Their major downfalls revolve around the fact that they do not know to look for new threats until a signature has been defined and installed on the system. Misuse detection systems are prone to creating large volumes of false alarms because in order to code rules or signatures that are not too restrictive to be bypassed easily by an attacker, they must be coded in a way that makes them prone to flagging legitimate activity as malicious, when in fact, it is not. Anomaly based IDS has a higher probability of flagging a new class of attacks based solely on the fact that it is, in fact, new. The major difficulty with anomaly based IDS is providing a clean data set to train the model. Anomaly based IDS also have a propensity to create huge volumes of false alarms because often things that are anomalous in nature are, in fact, legitimate activity.

2.2 Supervised vs. Unsupervised Learning

There are two main approaches to training the data mining models employed by intrusion detection systems.

Supervised learning involves using a set of examples to train a positive response to a set of inputs. This approach is typically used in misuse detection systems. Generally a set of data which are known to contain examples of a specific type of attack are used to train the algorithm which is used in the IDS. The algorithm is trained to classify future data with the same attributes as the training data as malicious. While many modern algorithms have the ability to generalize the training data in a way that allows them to be more flexible during their analysis of future data, the major drawback of this approach continues to be the need to provide ample training data for each potential category or type of attack in order to make the IDS as accurate as possible. Keeping the sensors up to date with current attack profiles becomes nearly impossible as the strategies and techniques available to an attacker change almost daily.

Unsupervised learning does not require labeled training data. This approach is often used in anomaly based systems where a baseline behavior profile for a host is modeled during a time when attacks are believed to be absent from the system. Any deviation from this profile is then flagged as potentially malicious. The major drawback of this method is that there exists a greater potential for the creation of false positives.

2.3 Mining Raw System and Network Data

Numerous data mining techniques have been applied to the field of intrusion detection. The vast majority of prior research has concentrated on mining various types of system audit data or raw network traffic in order to build more accurate IDS devices [8, 31, 45, 46, 47, 48, 58, 70, 74, 77].

2.3.1 Anomaly Based IDS

Anomaly detection systems are generally trained using unsupervised machine learning algorithms. The intrusion detection group at Columbia University has done extensive research into anomaly based intrusion detection systems. Most notably, they pioneered the use of system audit data to build profiles of normal behavior on Unix hosts, and attempt to detect attacks based on deviations by the system from its normal behavior profile [45, 47]. A major contribution of this work is a proposed scheme in which newly discovered anomalies would be saved as potential signatures for misuse based systems. The use of association rules for intrusion detection on host systems is also introduced in this research, which we will cover in more detail in chapter 3.

A system for building anomaly detection sensors using unlabeled data is proposed by Portnoy et al. in [70]. They note that two approaches can be used to training classifiers, notably supervised, using labeled data, and unsupervised, using unlabeled data. This technique is useful because they provide an improved method for detecting new anomalies without intervention of a human to train the new attack profile into the system. The major downfall with this approach is that training the system requires a significant amount of data which is free from attacks. This type of data is generally difficult to obtain, as is discussed in detail in [54].

Apap et al. present a method for detecting attacks on Windows system by monitoring the Windows Registry for anomalous access patterns in [4]. This model is trained using a clean set of access patterns, and any deviation from this baseline behavior is flagged as suspicious.

Anomaly based IDS is typically confined to host systems, but a proposed technique for anomaly based IDS on networks is given in [64] using Bayes classifiers. This approach is limited in its efficacy given the tremendous difficulty involved in finding suitable training data. A method for unsupervised learning using random forests in is presented by Zhang in [87].

2.3.2 Misuse Based IDS

The Intrusion Detection Project at Columbia University has completed extensive exploration of data mining techniques which are useful in mining system audit data. They have used the RIPPER Machine Learning algorithm from AT&T Bell Labs [17] to learn rules which can used to automatically classify system call traces as malicious or benign. They have implemented both misuse based classifiers as well as anomaly classifiers, the difference being that the first required training samples of malicious behavior in order to identify future instances of known attacks. The second implementation used RIPPER to create a normal class based on a baseline of normal system activity, any deviation from which was classified as a potential compromise of the system being monitored [45, 47]. Both of these implementations fall into the category of host based IDS and data was used from both system call audit data as well as tcpdump [45]. The use of association rules and frequent episode mining techniques was also introduced by Lee in [48], we will discuss this further in section 3. The Columbia IDS Project is further summarized in [45, 77], with descriptions of techniques for performing unsupervised cluster analysis on raw network traces in [70], and a technique for discovering malicious executable code in [74].

MadamId extracts features from network connections and builds models over connection records that represent a summary of the traffic from a given network connection [70].

2.3.3 Graph Based IDS

GrIDS is an intrusion detection system based on directed graphs presented by Cheung, et al. [15]. This approach does not fall cleanly into the category of misuse or anomaly based IDS, but is mentioned here as its graphical nature is relevant to our research. GrIDS uses directed graphs known as activity graphs to model activity between hosts in an attempt to discover attacks. Their approach differs from ours in that they monitor the base network data and build the activity graphs based on connection information coupled with temporal rules which define when an edge should be drawn and how long it is retained. This approach showed some promise at detecting large scale attacks such as worm outbreaks or brute force port scanning, however, the original paper was never extended.

2.3.4 Offline Analysis of Raw Data

One of the earliest attempts to use data warehousing techniques to facilitate off line analysis of raw data for intrusion detection is described in [58]. In this paper, a data warehouse is used to store historic tcpdump data, and clustering techniques are applied to detect instances of similar attacks. Basic measurements for the top 10 noisiest sensors are defined, but given the high volumes of false alarms, it has been found that measurements of this type have limited use in real world deployments. The problem of managing the large amounts of data associated with data mining based IDS is explored by Honig in [31], who propose an end to end architecture for managing and mining data related to anomaly based intrusion detection systems. Both of these approaches differ significantly from the approach used in our work as they attempt to store the raw syslog or tcpdump data for off line analysis. In large environments this is impossible.

2.4 Mining the Alarm Stream

The scope of research on the application of data mining techniques to IDS alarms has received significantly less attention. Cluster analysis has been used to attempt to classify alarms into attack and benign categories [39, 47] and to perform root cause analysis regarding the cause of false alarms [37, 38, 39, 40]. The results obtained using cluster analysis can vary widely depending on which algorithm and distance measure is used. These issues are discussed at length in [26, 32, 39, 45, 47, 58, 70, 83].

2.4.1 Alarm Correlation

At the beginning of the 21st century, significant effort was spent researching effective means of reducing the alarm load on SOC personnel via alarm correlation. The main goal of this activity was to group alerts that were significantly alike, and present them as a single alarm, or set of alarms, producing a more concise and understandable view for the analyst. Correlation of events is significantly more difficult than it sounds at first pass, especially if the network is monitored by a homogeneous set of sensors. In order to perform any type of analysis, the alerts must first be normalized into a standard format via a pre-processing routine, and then examined by the correlation engine. Many techniques have been proposed to solve the alert correlation problem. Alert Fusion is described by Valeur, et al in [81]. This is a multi-phase system which performs all normalization procedures internally to standardize the format of the alarms and attempt to fill in any missing fields that are required for analysis based on a set of well defined heuristics. Subsequent phases of this procedure attempt to determine the focus of the alert under inspection, as well as assigning a value as to the perceived impact the alert could have on the system if the attack were successful. Finally, an engine performs multi-step correlation based on known patterns as a means of illustrating the full scope of an attack as the sum of its parts.

A probabilistic measure of similarity is proposed by Valdes and Skinner in the EMERALD system [80]. The technique described uses a minimum measure of similarity for each attribute in an alarm to calculate the probability that they can be fused into a single event. If these conditions are met, the alarms are fused and a condensed view of the alarm stream is given to the analyst.

The concept of Attack Scenarios is introduced by Ning in [59]. The solution proposed is to discover scenarios comprised of multiple alerts in time sequential order, based on a prerequisite - consequence model. This technique attempts to match consequences of an alert to prerequisites of subsequent alerts, thus determining whether all conditions have been met for a particular scenario to be successful. By using a graph based data structure known as hyper-alerts, the authors are able to discover some new attack scenarios. This is an advantage over earlier work that relied heavily on predefined scenarios against which to perform signature based pattern matching. This work is extended in [61, 60, 62] to include more accurate measures of similarity based on a novel graphing technique known as an Attack Strategy Graph in addition to the prerequisite - consequence model defined in their earlier work.

2.4.2 Alarm Classification

In their basic form, alarm classification systems aim to classify alerts as either false positives, or true attacks. As the field has advanced, progress has been made in classifying sequences of alerts into specific subclasses, or categories, of attacks. The foundational work for these problems comes from IBM's Watson and Zurich Research Labs.

The earliest attempts were based on the creation of Association Rules, which were used to discover frequently recurring patterns of alarms [2, 52]. Once these patterns were identified, they were deemed either malicious, or benign, and used as a type of misuse based IDS system whereby as new alarms flowed into the system, if sets of the alarms matched the previously defined rules, the alarms were marked as having a high probability of being true attacks and placed in the corresponding category for examination by a human analyst. This approach has the drawback that a large number of training examples are required, and some skill is required to select which alarm attributes should be used to train the model. It is noted that complete classification is highly improbable, and that this partial classification, while not ideal, is a significant step forward in managing high volumes of IDS alarms.

Julisch introduced a novel alarm clustering engine known as CLARAty, which uses topologies to allow classification of alert attributes at various levels of abstraction. He calls these topologies *Generalization Hierarchies*. Using this approach, the attributes of each alarm are generalized based on a predefined set of criteria in the clustering engine until clusters begin to emerge [37, 38, 41, 39, 40]. Over the course of these papers, Julisch shows that using cluster analysis is an effective means of determining the root causes of alarms, be they an attack, or a system error that is resulting in high numbers of false positives. He then argues that by using this knowledge it is possible to improve the overall quality of the alarm stream by removing the root causes of false alarms. By performing iterative analysis of the alarm streams, coupled with removing the factors that cause the spikes in false alarms, he shows that cluster analysis is effective at reducing the overall load on the monitoring system. An alternative to Julisch's clustering engine is given in [53] which describes a technique to cluster alarms by attack phase. The major drawback of this work is that is requires manually assigning each alarm to stage category before executing the mining algorithm, making the system difficult to maintain over the long term.

Julisch's clustering work is extended by Pietraszek in [69] who couples the results from CLARAty with machine learning algorithms based on the RIPPER algorithm [17]. This system is known as ALAC, or Adaptive Learner for Alert Classification. ALAC works by observing the classifications made by a human analyst and iteratively training the classifier based on the human assignments so as to enable future autonomous analysis of incoming alarms. Given the nature of the data contained in IDS alarms, heavy preprocessing of the alarm attributes is required so that they can be evaluated by the machine learning algorithms. Pietraszek concludes with a discussion on how tools such as CLARAty and ALAC can be used to facilitate more efficient work by human analysts.

2.5 On Data

A recurring trend in the IDS literature over the past 10 years, especially that which deals with training anomaly based IDS systems, is that clean data to train the models is tremendously difficult to obtain. The most commonly used data set is the 1999 DARPA Off-Line Intrusion Detection data set, which was generated by MIT Lincoln Laboratory as a baseline to compare the performance of proposed intrusion detection systems [50]. The 1999 DARPA data is comprised of both syslog and tcpdump traces that are then given as input to the various systems which are to be evaluated. This data has been used extensively, but has received some criticism due to the fact that it was synthetically generated to model a small military network, and there is some concern as to whether it accurately portrays the real world environment [54]. Regardless of these concerns, which are legitimate, the fact remains that this data set is 10 years old, and can hardly be considered indicative of modern network and system behavior. As such, we have chosen to conduct our experiments on the previously discussed production networks. Because the set of IDS alarms generated is from production networks, and has been reviewed by highly trained analysts assisted by state of the art automated analysis systems, we have access to extensive data for the evaluation of our algorithms.

During the course of their daily activity, the SOC analysts produce incident logs which result in a labeled subset of the alarms which are *known* to be genuine attack activity. It is impossible to guarantee that this list is exhaustive of all of the attacks carried out in the monitored networks. In fact, we prove that this is sometimes not the case as our algorithms have uncovered attacks that were overlooked by both the human and machine assisted analysis. However, we believe that the data available has put us in a unique position to advance the state of the art in mining large alarm data sets.

2.6 Conclusions

A wide variety of approaches have been explored to building the better IDS. Anomaly and misuse based systems both have their strengths and weaknesses. Various techniques involving off line data mining have been proposed in an attempt to improve the accuracy of the IDS sensors, or at the very least provide a secondary view into attacks that may have been overlooked. Intrusion detection continues to be a difficult problem. In the following chapters we provide a set of heuristics assisted by data mining and machine learning techniques which provide an incremental step forward in improving the reliability of intrusion detection systems, specifically in the context of enterprise class security operations.

Chapter 3

Association Rules Mining

3.1 Automated Rule Discovery

The advent of *rules only monitoring* has provided a means of reducing the number of false alarms which must be processed in the Security Operations Center, while simultaneously providing a mechanism for monitoring for known attack patterns in the incoming alarm streams. The use of rules only monitoring is becoming common place in large intrusion detection infrastructures, and is supported by many of the commercially available Enterprise Security Management (ESM) solutions [18, 19]. This is a strategy in which the ESM infrastructure has the ability to consolidate the alarms generated by a large number of IDS sensors for display on an operations console in a Security Operations Center (SOC). These solutions consist of a complex monitoring infrastructure which reduces the number of alarms that the SOC personnel must examine by either aggregating alarm streams in a near real-time manner and displaying summarized alarms for inspection

[22], or by examining the alarm stream for predefined patterns and subsequently triggering meta-alarms as a result of pattern matches [18, 19]. This pattern matching solution has the same inherent problem as signature-based intrusion detection sensors. If the signatures, or in this case the rules which define the alarm patterns, are not of high quality as well as current with emerging attacks, the ESM engine runs a very high risk of missing true attacks by simply not detecting them. In this chapter we show that although a large percentage of the alarms generated by an intrusion detection system are false positives, subtle patterns can be uncovered in the logs which are indicative of certain types of attack activity. We provide a technique which can be used in an off line analytical environment to automate the discovery of previously unknown rules for the ESM rule engine, thus helping to mitigate the risk of an obsolete rule base. Our technique allows for more timely discovery of new patterns, while at the same time reducing the labor costs associated with keeping the ESM rules engine up to date.

The time from the appearance of new attack profiles to the time when new rules describing them are implemented is *critical*. Any delay in updating the rule base could result in potentially undetected attacks. The amount of manual inspection currently required to discover new rules makes staffing to meet these time demands very expensive. We have found that using our framework to automate this task drastically decreases the amount of manual inspection required. This in turn has the net effect of decreasing the time from discovery to implementation as well as decreasing the overall cost of maintenance.

3.2 Related Work

The concept of association rule mining for intrusion detection was introduced by Lee, et al. in [45, 48], who used the rules returned by the association rule algorithm to prove that causal relationships exist between a user, and the type of entries that are logged in the audit data as a result of their actions on the system. Based on this finding, Lee was able to use the rules produced from the system audit data to aid in the definition of attack models for an online IDS system. We extend this work to the area of mining IDS alarm streams as opposed to raw system data. Our findings mirror those presented in this work, namely that we are able to show the existence of causal relationships between an attacker and the legitimate alarms produced as a result of their actions. We discover these patterns using the association rules algorithm [1].

Prior use of association rules to to mine IDS alarms can be found in [52]. In this work, Manganaris et al. created an anomaly detection system using association rules to baseline the normal behavior of entire networks of sensors, and subsequently, individual sensors. They then use this model to decide whether an alarm, or set of alarms should be filtered or logged for review by the SOC. Their second goal was to create client specific profiles and segment them into categories based on the monitoring needs that were illustrated by the association rules generated from their system. The insight behind this approach was that if a group of alarms matches a known rule with a high confidence value, that set of alarms has a high probability of being part of the baseline noise of a sensor and can be disregarded. Our work differs from this approach as we are not building an anomaly detection system using association rules, but are proposing an automated framework for producing new rules for the misuse based monitoring system in use at the SOC. Our framework proposes new rules which can be coded into the ESM, and are a type of meta-signature which looks for patterns in the alarm logs which have a high probability of being malicious.

3.3 Preliminaries

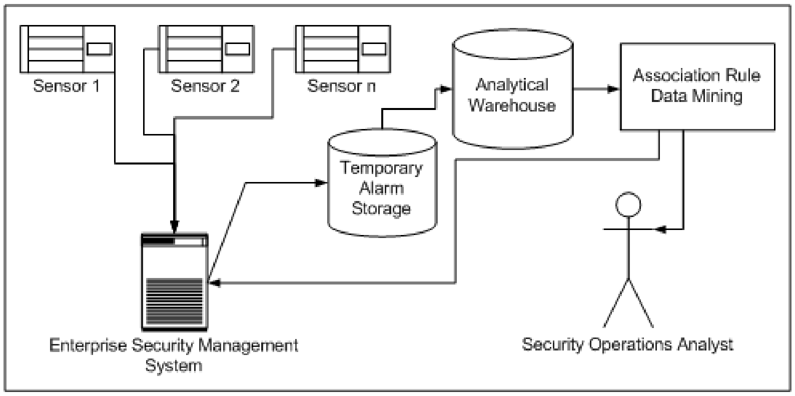

Sensor 1
Sensor 2
Sensor n
Analytical Warehouse
Association Rule Data Mining
Temporary Alarm Storage
Enterprise Security Management System
Security Operations Analyst

Figure 3.1: The association rules data mining architecture

Figure 3.1 illustrates the place of our rules generation engine in the overall SOC environment. The IDS alarms are retrieved from our data warehouse and analyzed by the association rules engine. The results are then examined by an analyst prior to installation in the ESM.

3.3.1 Multi-Phase Attacks

Attempts to compromise networked computing resources generally consist of multiple steps. The first of these is the reconnaissance phase, consisting of the identification of target operating systems, port scanning, and vulnerability enumeration. This is followed by the exploitation of the weaknesses discovered during the initial intelligence gathering process. A successful attack often ends with the installation of back door channels so that the attacker can easily gain access to the system in the future [55].

If an intrusion detection infrastructure is in use at the victim network during this process, each action by the attacker has the potential to raise an alarm, alerting the security staff to the presence of malicious activity in the network. Generally speaking, intrusion detection sensors do not have the ability to aggregate the alarms for the discrete activities into an end-to-end attack profile. Given that an alarm is raised for each perceived malicious action, the typical intrusion detection sensor can generate many thousands of alarms per day. Unfortunately, the vast majority of these alarms are false positives [39], and the task of separating the real attacks from false alarms quickly becomes daunting, especially in large IDS environments.

Our research has shown that in the same manner that Lee was able to demonstrate the existence of causal relationships between users and the entries logged in system audit data as a result of their actions [45, 47], it is possible to show causal relationships between an attacker and the combination of alarms which are generated in intrusion detection logs as a result of their behavior in a network. We were then able to use the patterns which were discovered using our data mining technique to configure new rules for the ESM system in a rapid and economical way. As a means of demonstrating this, we include examples of attack activity which answer the following questions:

1. What techniques did the attacker employ?
2. How were these techniques manifested as patterns in the IDS alarm logs?
3. Was our framework able to detect these patterns?
4. How did the discovered patterns result in a new rule in the ESM?

As with all data mining solutions, much up-front work must be done adjusting the parameters for the algorithm so that optimal results are obtained. There is no silver bullet configuration, and it is noted throughout the literature that when using association rule mining, the features which are chosen for examination are critical to the success of the algorithm [47, 58].

3.3.2 Association Rules Terminology

The main goal of association rule mining is to locate non-obvious relationships between members of a large data set [20]. The goal of our analysis is to find associations between the various attack signatures and IP addresses which constitute true attacks on the network, and capture them as rules in the ESM rule engine so that the SOC can easily detect future instances of the attack. The association rules algorithm generates rules in the following form, as well as some statistics which describe their strength and quality.

$$[x][y] \rightarrow [z]$$

$$Support = 50$$

$$Confidence = 80$$

This rule indicates that a relationship exists between the items x, y and z. Specifically, the rule states that whenever x and y were present in a given grouping, known as a transaction, then z was present as well. The Support value states that this specific grouping of three items represents 50 percent of the transactions which were examined. The Confidence value states that 80 percent of the time that the items x and y were found together, the item z was also found [20].

Formally, let $I = \{i_1, i_2, ..., i_n\}$ be a set of items. Given a set of transactions D, where each transaction is defined as a set of items $T \subseteq I$, a transaction T contains X if $X \subseteq T$. An association rule is an implication $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. The association rule $X \Rightarrow Y$ holds in the transaction set D with a Confidence c if c percent of transactions in D which contain X also contain Y. The association rule $X \Rightarrow Y$ has a Support value s in the transaction set D if s percent of the transactions in D contain $X \cup Y$ [1].

In our results, the Support values are typically less than 5 percent. This is due to the fact that thousands of signatures exist in the alarm database, and generally the rules which are discovered cover only a small percentage of the total signature set for a given day.

3.3.3 Modeling Alarms as Directed Graphs

Network ID	**Source IP**	**Destination IP**	**Signature**
Network A	10.0.0.1	10.0.0.4	Signature 1
Network A	10.0.0.2	10.0.0.4	Signature 1
Network A	10.0.0.3	10.0.0.4	Signature 2
Network A	10.0.0.5	10.0.0.7	Signature 2
Network A	10.0.0.6	10.0.0.7	Signature 2
Network A	10.0.0.7	10.0.0.8	Signature 2
Network A	10.0.0.9	10.0.0.13	Signature 3
Network A	10.0.0.10	10.0.0.13	Signature 4
Network A	10.0.0.11	10.0.0.13	Signature 5
Network A	10.0.0.12	10.0.0.13	Signature 6

Table 3.1: IDS alarms resulting in 3 connected components in an Alarm Graph

In order to facilitate a novel technique for filtering the number of alarms which must be analyzed during the mining process, we generated an Alarm Graph using the definition in Section 1.4.2. Each entry in the data warehouse included both the source IP address and destination IP address for which the alarm was raised. We deduced the direction of each potential attack from this information. We then generated a directed graph $G = (V, E)$ such that each IP address was represented as a vertex in the graph, and each edge was represented by a detected alarm. The edge was drawn from the source IP address toward the destination IP address, corresponding to the direction of the alarm.

The results are such that the IDS alarms which are shown in Table 3.1 are modeled as the directed graph shown in Figure 3.2.

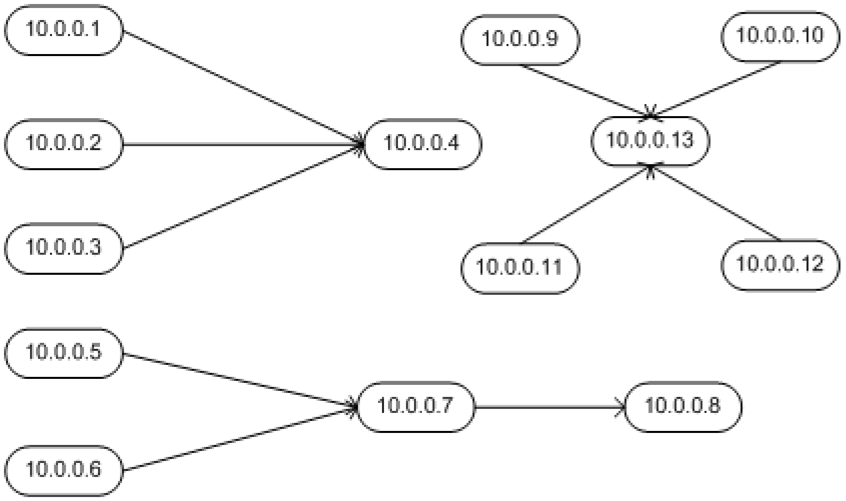


Figure 3.2: Intrusion detection alarms as a directed graph with three connected components

3.3.4 Data Set Reduction Using The Connected-Component Algorithm

The number of alarms produced in large intrusion detection environments can easily be on the order of millions of rows per day. We have observed raw event counts approaching 10 million events per day. It is a given that most of these alarms were false positives, however it was not possible to label precisely which alarms were of genuine concern [37, 38, 39, 40]. Because of the large volumes of data that required analysis, it was beneficial from a performance perspective to trim away any data that we knew to be irrelevant before starting the mining activities. In order to facilitate this, we represented the alarm logs as directed graphs, which allowed us to employ the use of graph algorithms to limit the scope of our inquiry. This process was only possible if we had a priori knowledge of a signature for which we wished to discover new rules.

When considering the problem of finding rules which exist between distinct signature and IP address combinations, it was important to note that there were alarms in the overall data set that could not be related to one another. For example, while examining one set of alarms, if we knew that another set of alarms could not be related to it, we removed the second set from consideration.

Drawing on our earlier discussion of alarm logs as directed graphs, we could translate the set of alarms in Table 3.1 into the directed graph shown in Figure 3.2, which displays three easily identified connected components. Limiting our mining activity solely to alarms in the same connected component allowed us to explore only those relationships between alarms which could legitimately exist. A complication arose in the case of slave nodes which were controlled by a master who was not represented in the graph. We designated this scenario to be out of scope for our experiments.

When attempting to discover rules for a specific signature, a natural question arises as to why we did not simply limit the alarms to those which were produced by a source IP address that also produced the signature undergoing analysis. Reducing the data set in this manner was possible if we were interested only in the detection of single-source attacks for a specific signature. We would then examine the set of all alarms generated by a source IP

Network ID	**Source IP**	**Destination IP**	**Signature**
Network A	10.0.0.5	10.0.0.7	Reconnaissance
Network A	10.0.0.5	10.0.0.7	Exploit 1
Network A	10.0.0.7	10.0.0.8	Exploit 2
Network A	10.0.0.6	10.0.0.7	False Alarm

Table 3.2: Intrusion detection alarms for a multi-stage attack

address which triggered the signature in question. However, trimming the data in this way would severely limit any further analysis that we wished to perform on the set of alarms. By carrying the other relevant alarms from the connected component, we have access to a greater number of signatures and IP addresses for analysis. We also preserved the ability to perform further analysis by grouping on fields other than the source IP address if we found that a more extensive exploration of the data was warranted.

For example, consider a multi-stage attack consisting of a reconnaissance event which discovered a vulnerability on the target and exploited it in a way that in turn attacked a third system. Table 3.2 lists alarms which would constitute such a scenario. These alarms are shown graphically in Figure 3.3. The reconnaissance and subsequent exploit occur between 10.0.0.5 and 10.0.0.7. A successful compromise of 10.0.0.7 by 10.0.0.5 is then used to further attack 10.0.0.8.

If we had specified the reconnaissance signature as the input to the mining process and trimmed away all IP addresses which did not trigger that

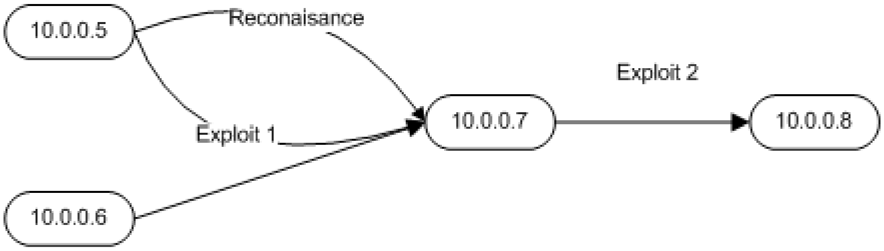


Figure 3.3: A multi-stage attack scenario

signature, we would have missed the second half of the attack. As such, limiting the alarms that we examine only to those which occur in the same connected component provided the appropriate balance of efficiency without interfering with our ability to perform complex analysis of the relevant data. On average, we were able to reduce the amount of data that required analysis by 30 percent. However, our ability to reduce the amount of data we inspected was sometimes diminished in the case of graphs which were nearly fully connected. Because this type of graph produced one large connected component comprised of the majority of the alarms, the amount of data which we were able to trim away prior to executing the association rule algorithm was in some cases reduced to less than 5 percent.

3.4 The Approach

Our experiments were conducted on the set of alarm logs generated by network-based intrusion detection sensors over a 24-hour period for 135 distinct production networks. The alarms were loaded into a data warehouse specifically engineered to facilitate efficient off line analysis of intrusion detection alarms using association rule mining techniques. We repeated the experiments on a daily basis for 30 days.

3.4.1 Generation of Signature Specific Rules

Our first set of experiments were conducted with the goal of discovering new rules for a signature which was thought to be exhibiting suspicious behavior. We accomplished this by first selecting the set of connected components in which the suspected signature was present, and discarding all alarms that were not members of these these connected components. Once we had filtered the data in this way, we then executed the association rule algorithm to see if any rules for this signature were generated. Algorithm 1 describes this technique.

Algorithm 1 Find-Signature-Rules(G,s)

```
Require: $G = (V, E)$, a directed graph of IDS Alarms, $s$ a subject signature
$C$ ←Connected-Components($G$)
for all $C' \in C$ do
    if $s \in C'$ then
        copy all alarms in $C'$ to $T$
    end if
end for
R ← Association-Rules(T)
Return R
```

Of the scenarios that we discuss, signature specific rule generation experienced the lowest occurrence of success. One of the reasons for this was that rather than being identified algorithmically, the signature examined was generally chosen by a human operator who was simply curious as to whether any correlations involving this signature were hidden in the data.

The subject signature was most often chosen for analysis based on an abnormally high volume of that signature over a specific time period, or its appearance as a new signature where it had not been previously detected. These scenarios might occur due to the introduction of a previously unforeseen attack scenario into a network, or simply because of software updates on the sensors themselves.

Over the course of our experiments, we were able to successfully generate rules for specific signatures roughly 10 percent of the time. However, given that data mining always requires manual evaluation and exploration of its results, we still believe this to be an effective tool for operations staff to have at their disposal. The skill of the user conducting the analysis had a great impact on the quality of the results, which is consistent with the views expressed in [38, 58, 83]. We found that as the user's experience with the technique grew, their ability to choose signatures for which rules would be generated grew as well.

Approximately half of the experiments uncovered patterns involving signatures other than those which were the original subject of our exploration. In some cases, the rules algorithm would produce more than 100 rules for a single run. This appeared at first glance to be overwhelming, however, the rules which exhibited very strong Confidence values floated to the top on their own merits, and were easily identifiable.

If we were unable to safely remove significant numbers of rows from consideration by filtering on connected component, the time required for the mining algorithm to generate results grew rapidly. A side effect produced by this complication was the generation of a very large number of rules by the algorithm. In some cases we observed rule counts as high as 8000 for a single network's data. This number of rules on its own is of limited value, as it does not solve the problem of limiting the amount of data which must be examined manually by operational staff. However, the vast majority of the time, the count of rules for a single network on a single day was below 100. When a spike occurred, we found it to be indicative of significant phenomena in the network being monitored. We discuss these findings in a later section of this chapter.

A useful means of tuning the number of rules returned by the association rule algorithm was to adjust the minimum values for the Support and Confidence parameters for the mining algorithm, which had the net effect of limiting the number of rules which were produced. The obvious risk in limiting the rules to those with a very high Support value is that any signature which generated low volumes when compared to the volume of alarms in a single day will simply be lost. It is for this reason that we generally set the Support value at a relatively low setting, while enforcing a constraint of high Confidence values on the results. By doing this, we were able to limit the results to rules which were found to hold the majority of the time.

3.4.2 Generation of Single Source Rules

Our framework generated the greatest number of high Confidence rules when we grouped the transactions in the database by source IP address. When using this approach it was not necessary to limit the rows we examined using the connected components algorithm, though it was beneficial from a performance perspective if we knew the signature for which we wished to perform the analysis. When performing single-source analysis, we also found that setting the minimum values for the Support and Confidence parameters to 0 was useful. Intuitively, providing these low values for the Support and Confidence parameters would produce an overwhelming number of rules. However, over the course of our experiments we found that on average, a single source IP address will trigger less than two signatures in any 24 hour period. Because we were looking for correlations between signatures which were generated by a single source, it was obvious that no rules would be generated for these IP addresses. Because of this, 87 percent of our single-source experiments generated zero rules for a given day's data.

3.5 Efficacy of the Framework

The Confidence value given for a new rule was critical in determining how effective the rule would be in the production monitoring environment. On average, 66 percent of the rules we produced had a confidence value of 100, and rules with a Confidence value over 80 were produced 86 percent of the time. We found that certain types of attack activity generated very high volumes of rules with a Confidence value of 100 percent. While these rules were not false positives, they skewed the statistics. Disregarding them, the percentage of rules with a Confidence value above 80 percent was 63 and the percentage of rules with Confidence values of 100 was 43.

When applying our technique, we were able to detect attacks that did not trigger meta alarms on the operational console. In one case, we were able to detect an attack on a day where the ESM system received 1,543,997 alarms.

The detected attack was comprised of only 6 alarms, and did not result in a meta alarm firing on the operational console. This is of great consequence as this attack would otherwise have been lost in the noise of the 1.5 million other alarms that flowed through the infrastructure that day. It was then possible to code a rule describing this scenario into the ESM system so that future instances would be detected.

3.5.1 Rule Examples

1. **Web Server Attack**:

Network ID	**Source IP**	**Destination IP**	**Signature**
Network A	24.9.61.170	192.168.2.4	AWStats configdir Cmd
Network A	24.9.61.170	192.168.2.5	XMLRPC PHP Cmd
Network B	24.9.61.170	192.168.2.16	AWStats configdir Cmd
Network B	24.9.61.170	192.168.2.17	XMLRPC PHP Cmd
...	...	...	...

Table 3.3: IDS alarms for a multi-stage web server attack

Rule for Multi-Stage Web Server Attack
[AWStats configdir Cmd]$\Rightarrow$ [XMLRPC PHP Cmd]
Confidence = 100
Support = 3.45

Our first example does not indicate the reconnaissance approach which was used to determine the list of web servers that underwent the detected attack, as no reconnaissance signature was present in the alarm log that generated this rule. It is possible that the technique used did not trigger an alarm, or that the reconnaissance phase of the attack was carried out many days in advance in an attempt to prevent detection. The alarms which were present in the database which generated this rule are indicated in Table 3.3. The IP addresses have been sanitized to prevent identification of the customer network for which the analysis was performed.

This rule involves two signatures generated by an attacker who was attempting to locate a vulnerability to exploit on a web server. The first stage of the attack appeared in the alarm logs as multiple instances of the signature, [AWStats configdir Command Exec], which fired as the attacker attempted to execute an unauthorized command using the *configdir* variable of the *awstats.pl* CGI script. The second phase of the attack appeared in the alarm logs as the signature, [XML RPC PHP command Execution], which was triggered as attempts were made to exploit an XMLRPC vulnerability via SQL injection [78].

We were able to detect this pattern by grouping alarms by the source IP address, and looking for repetitive combinations. When grouped together, these two signatures, when triggered by the same source IP address, are indicative of an attacker who attempted multiple exploits before either compromising the target server, or moving to another victim. Further, because these were the only rules generated for this network on the day in question, we can be almost certain that the activity was legitimate attack activity and not part of an automated vulnerability scan. We observed this same pattern on two distinct monitored networks on the same day, which indicates further that the detected activity was a real attack.

2. **Reconnaissance Attack**:

Rule for Reconnaissance Activity

[TCP Port Scan][TCP Probe HTTP]⇒ [LANMan share enum]
Confidence = 66.66
Support = 1.7

This rule was generated using data from a network where an attacker was attempting to locate vulnerable file shares to attack. A pattern was found in the alarm logs for this customer which described a frequently occurring pattern of two TCP-based reconnaissance signatures followed by a LANMan share enumeration, which is a common means of locating vulnerable file shares for future exploitation.

3. **Scanning Activity**:

Rules for Scanning Activity

[RPC Race Condition Exploitation]⇒ [TCP SYN Port Sweep]
Confidence = 51
Support = 1.8

[SQL Query in HTTP Request]⇒ [TCP SYN Port Sweep]
Confidence = 43
Support = 1.7

[FTP RealPath Buffer Overflow]⇒ [TCP SYN Port Sweep]
Confidence = 100
Support = 0.2

Rules of this type frequently materialized when a network experienced a series of exploit and probing attempts. This type of brute force attack results in a set of rules where the actual attacks span a wide range of signatures, and are associated with a reconnaissance event in the form of a TCP port scan. The goal of the attacker in these situations was to discover open vulnerabilities on a system to be exploited in future attacks. A special case which had to be considered when searching for these types of attacks was whether or not the scanning activity was legitimate traffic generated as part of a policy verification procedure. This was most commonly caused by the use of an automated scanning appliance under the control of the network security staff as a means of ensuring that the hosts under their control had been updated with the most recent security patches.

4. **Worm Related Rules**:

   Worms propagate by exploiting vulnerabilities to gain control of a victim server, subsequently scanning the network for other vulnerable machines, as to guarantee rapid and widespread infection before a patch can be implemented. The following example rules define a multi-stage worm attack which took advantage of file sharing vulnerabilities which exist in a widely deployed operating system. The first rule correlates an overflow exploit of an SMB vulnerability, and subsequent access. The existence of the [ICMP L3 Retriever Ping]alert is indicative of Black/Nyxem worm activity.

Rule for Black/Nyxem Worm
[NETBIOS SMB–DS IPC unicode share access][ICMP L3retriever Ping]⇒ [NETBIOS SMB–DS Session Setup And request unicode username overflow]
Confidence = 100
Support = 41

Another example of worm related patterns which we detected describes correlations relevant to the SQL Slammer worm which ravaged the Internet in 2002, and is still frequently detected. This worm exploited a buffer overflow vulnerability to execute malicious code and install itself on the victim machine, after which it scanned for other hosts to which it could propagate. Two mature signatures exist for this worm in our monitoring environment. The first signature describes the initial overflow attempt, followed by a propagation attempt. Our algorithm was able to determine that a strong correlation exists between these two signatures. Using this information, we can then code a new rule into the ESM which watches for this type of pattern, and raises a meta alarm when it is detected.

Rule for SQL Slammer Worm
[MS-SQL version overflow]⇒ [MS-SQL Worm Propagation]
Confidence = 100
Support = 35

While worms such as SQL Slammer are well known, we have shown that our method can consistently detect the patterns which are generated in the alarm stream by their propagation. Based on this, we feel that the techniques presented here can be applied to detect future instances of emerging worm traffic, *independent* of whether the intrusion detection sensors supply worm specific signatures, or if the newly emerging worm manifests itself as a combination of existing signatures.

3.5.2 Identification of High Risk Networks

As mentioned previously, we found that on average, 87 percent of our experiments generated no rules for a given network over a 24-hour period. This translates to the total number of networks for which rules were produced in a single 24-hour period being 17 out of 135. Figure 3.4 shows a typical count of rules generated per monitored network. In this case, 19 out of the 135 monitored networks produced rules. Of these 19 networks, 12 produced 10 or less rules for that particular day, while one network produced 117 and one produced 2295. Graphing these counts highlights the anomalous networks, which provides a useful tool for operational personnel to see which networks require immediate attention.

3.5.3 Facilitation of Sensor Tuning and Root Cause Analysis

Much in the same way that Julisch describes the use of cluster analysis for the identification of the root cause of false positive alarms in [38, 39, 40], we have found that we can facilitate the determination of root causes of certain alarms using our data mining framework.

Figure 3.5 shows a 30-day trend of rule volumes broken out by day for a selected network. The spikes represent the generation of 4854 and 7926 rules on two separate days, respectively. When we inspected these rules, they appeared to describe a denial of service attack on an electronic commerce site.

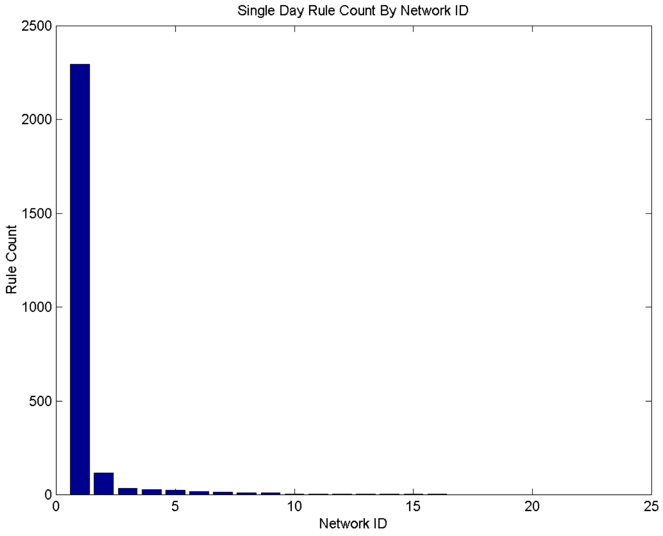


Figure 3.4: Anomalous network activity as shown by a count of rules produced per network for a selected day

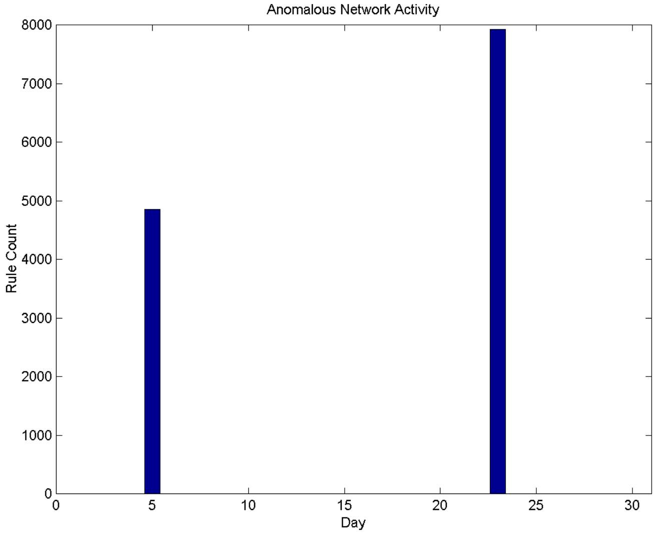


Figure 3.5: Spikes indicating anomalous activity for a single network

The rules covered 47 percent of the alarms which were generated on the corresponding days, and were comprised of a flood of Half Open SYN signatures, coupled with various other attack signatures. After some investigation, it was discovered that the actual cause of the alarms was a misconfigured IP route between a web application server and an LDAP server. Every time that a user attempted to authenticate to the application, the request was lost due to the IP stack's inability to complete the TCP handshake. The intrusion detection sensors interpreted this as a spoofed source IP address, which resulted in a flood of the corresponding alarms to the security operations center. By fixing this IP routing problem, the corresponding reduction in alarms would provide increased fidelity in the alarm stream for that network as well as increased chances that legitimate attack traffic would not be overlooked.

3.6 Conclusion

We have outlined a novel framework for the application of association rule mining techniques on the millions of alarms which are received daily at large Managed Security Service Providers. As new attack strategies emerge, our framework is successful at discovering the associated patterns of alarms which occur as a result of the attacker's actions in the victim network. By highlighting these patterns, we reduce the time required for SOC personnel to implement meta rules which ensure the detection of future instances of emerging attacks.

Our framework provides a reliable means of closing the time gap between the appearance of new attack profiles in the alarm logs and the configuration of rules in the ESM. We accomplished this while reducing the human-error factor, as well as the costs associated with manually inspecting large alarm logs.

In addition to the ability to discover new rules for the ESM, we have also shown that our framework can be used to flag suspicious network activity for in-depth analysis by operations staff in an off line environment. The use of our framework can detect a variety of classes of attacks which may have been lost in the large data volumes due to processing time constraints in the on-line monitoring system.

Chapter 4

Ranking Alarm Graphs

4.1 Watch Lists

In addition to its ability to use meta-rules to monitor the incoming alarm stream for known malicious patters, the ESM also has the ability to maintain watch lists of suspicious IP addresses. These lists are generally created manually by an analyst who wishes to monitor a specific IP, or set of IPs more closely. If an alert is received for an address on this list, the alert is assigned a higher review priority by the ESM. A need exists for generating watch lists automatically, based on an objective ranking algorithm that determines which IP addresses have the highest probability of being involved in attacks.

In this chapter, we describe an automated ranking procedure that uses the underlying structure created by an Alarm Graph to rank nodes as a function of their proximity in the graph to known malicious hosts, or victims.

4.2 Related Work

Our research draws inspiration from the field of Attack Graph generation. Attack Graphs are used to model the set of possible actions that could result in a compromised network. As described by Lippmann and Ingols in [51], research on Attack Graphs has focused on three areas. The first is the modeling of network connectivity and known vulnerability findings as a means of enumerating the options available to an attacker to successfully compromise a target host [3, 6, 33, 34, 57, 59, 61, 76]. The second is the definition of formal languages used to describe these graphs, as well as the conditions under which state transitions within them are allowed [21, 79]. The third thrust of research has focused on grouping large numbers of intrusion detection alerts by compiling end-to-end attack scenarios or strategies based on Attack Graph analysis as discussed by Ning, et al. in [59, 60, 61]. Although various works [63, 76] have discussed methods for the use of probabilistic processes to analyze Attack Graphs, they generally make the assumption that the values which describe the probability of a state transition are predefined. This is addressed by Mehta, et al. in [56], who provide a method for ranking Attack Graphs using link analysis techniques to find the values algorithmically. After the ranking values are computed for an Attack Graph, the nodes with the highest ranks are highlighted as those which have the greatest probability of being involved in an attack. Starting with these marked nodes, an analyst can then focus their attention on the most important portions of the Attack Graph, and use the information contained therein to develop mitigation strategies. It is this concept that we extend in our work by applying a similar analysis technique. Our approach differs from previous work in that rather than use Attack Graphs, we construct an Alarm Graph using the set of intrusion detection alarms triggered for a specified time period. A second key difference between our approach and the previous work is that we augment this graph with data on known attacks, and use link analysis techniques to gain deeper understanding as to how the known attacks influence other nodes in the graph.

4.3 The Ranking Algorithm

We employ Page and Brin's PageRank algorithm [13, 68] to analyze the Alarm Graph. The PageRank algorithm was originally designed to rank the relative importance of a web page among the set of all pages in the World Wide Web. PageRank utilizes the link structure provided via hyperlinks between web pages to gauge this importance. Each hyperlink from a page to a target page is considered a vote, or endorsement of a page's value by the page which links to it. PageRank is computed recursively, and as such, any page that is linked to from a page that has high rank will itself receive a higher rank due to the fact that an important page has linked to it. A random surfer model is assumed, in which a user selects a random starting point and navigates the web via random clicks to other pages. If a surfer lands on a page with no outbound links, known as a dangling state, they are assumed to start the process again from a new random location. It is also assumed that at any point, a surfer can randomly jump to a new starting point. This random re-entry is captured via a damping factor γ, which is divided by the number of nodes in the graph, and added to all other nodes equally. This model yields Equation 4.1.

$$PR(v_i) = \frac{(1-\gamma)}{N} + \gamma \sum_{v_j \in IN(v_i)} \frac{PR(v_j)}{|OUT(v_j)|} \tag{4.1}$$

The first term of this equation represents the probability of a node being reached via a random entry into the graph, either through a bookmark or the surfer typing a known URL into the browser. The second term is the summation of the probabilities given to a state from all nodes that link into the node. As such, $\{v_1, v_2, v_3 ... v_n\} \in V$ are the vertices in the web graph, $IN(v_i)$ is the set of pages that link in to v_i, $|OUT(v_j)|$ is the number of links out of v_j, and N represents $|V|$ [13, 68].

The output of the PageRank function is given by the vector $PR = (pr_1, pr_2, ...pr_n)$ where pr_i represents the rank of vertex v_i. The values of PR correspond to the entries of the dominant eigenvector of the normalized adjacency matrix of G. This eigenvector is defined as:

$$PR = \begin{pmatrix} pr_1 \\ pr_2 \\ \vdots \\ pr_n \end{pmatrix}$$

where PR is the solution to:

$$PR = \begin{pmatrix} \frac{1-\gamma}{N} \\ \frac{1-\gamma}{N} \\ \vdots \\ \frac{1-\gamma}{N} \end{pmatrix} + \gamma \begin{pmatrix} \alpha(v_1, v_1) & \cdots & \alpha(v_1, v_N) \\ \alpha(v_2, v_1) & \ddots & \vdots \\ \alpha(v_N, v_1) & \cdots & \alpha(v_N, v_N) \end{pmatrix} PR$$

using the adjacency function:

$$\alpha(v_i, v_j) = \begin{cases} \frac{1}{|OUT(v_j)|} & \text{if an edge exists from } v_i \text{ to } v_j; \\ 0 & \text{otherwise.} \end{cases}$$

This algorithm models the probability that a user who is randomly surfing the Internet will land on a given page [13, 56, 68].

4.3.1 Extending PageRank to Alarm Graphs

We extend the concept of ranking web graphs to ranking Alarm Graphs in the following manner. Each alarm in the alarm set has the potential to represent a genuine attack. For the purposes of our analysis, we think of an attack as a state transition from the node representing the attacker to a successful compromise of the target IP of the alarm. Following this logic, each path in the Alarm Graph represents a potential path of compromise by an attacker through the monitored network.

Using Alarm Graphs, we model the potential paths that an attacker could take through the network, as detected by the intrusion detection sensors, in lieu of the web graph which is proposed in the original PageRank discussion. Using this model, we can then analyze which nodes in the graph have the highest probability of being visited by an attacker, given random entry into the Alarm Graph.

The use of the PageRank algorithm requires that we model the IDS alarms as an ergodic Markov model. Simply put, ergodicity of a Markov model means that every state in the graph is reachable from every other state, given sufficient time. Ergodicity also guarantees that the model will converge to a stable state given sufficient time [25]. The model generated using IDS alarms is not ergodic without some modification. We remedy this in the same manner as is proposed in the original PageRank paper [68], by creating a link from all dangling states to all other nodes in the graph, where a dangling state is defined as a state in the graph from which no outbound links originate. The intuition here is that if an attacker reaches a dangling state, or the end of a potential attack path as detected by the IDS, that they can begin a new attack by jumping randomly to another portion of the graph. The PageRank algorithm captures the effect of this random re-entry into the graph via the damping factor, as described in Equation 4.1.

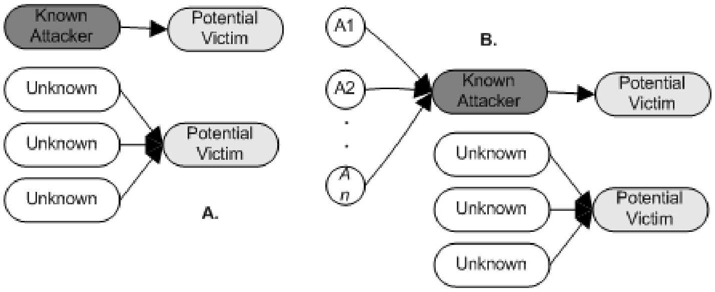


Figure 4.1: Ideal coloring of an Alarm Graph

Ideally, when using this approach we would produce rankings in which nodes undergoing genuine attacks receive the highest ranks, and as the level of risk for a host decreases, so does its corresponding rank. Using these ranks, we would like to produce visualizations that highlight nodes of highest risk as shown in Figure 4.1a. However, in order to accomplish this consistently, we must incorporate additional information into the graph prior to executing the ranking algorithm.

4.3.2 Incorporation of Known Attacks

The results of data analysis are known to improve if the analysts (or algorithm) are able to include additional up front knowledge of the data set [24]. The data warehouse that stores our intrusion detection alarms also contains a labeled data set of known attacks that have been identified by the SOC during the course of monitoring the network. We will refer to this data as the set of known security incidents. Prior to ranking the Alarm Graph G, we augment the graph with this data in a manner that improves the quality of the ranking output.

The graph augmentation occurs as follows. In the same manner that a link from one web page to another can be considered a vote or endorsement for the target page, the existence of an edge to a given node in the Alarm Graph can be considered a vote that the targeted node is involved in an attack. Extending this notion, if we know for certain that a given node is involved in an attack, we would like to observe how this fact influences the other vertices in the graph. We accomplish this by annotating the graph with a set of n auxiliary nodes, each of which casts a vote for a single known attacker or victim. The size of n is variable based on the size of the Alarm Graph as a whole. For the purposes of our experiments we uniformly set $n = 50$. Our primary goal is to evaluate the risk that other nodes are extensions of known attacks. Our analysis does not evaluate physical network connectivity, rather we examine the existence of traffic between pairs of hosts that has been perceived as malicious by the IDS. It is important to note that no edges are drawn toward auxiliary nodes, which ensures that no auxiliary vertex will appear as a highly ranked host. We illustrate this technique in Figure 4.1b. Given this annotated Alarm Graph we can now calculate the influence of known attackers and victims on the remaining vertices in the graph using the PageRank algorithm. PageRank is computed recursively, and once the model converges, we are able to observe the influence of these high ranking nodes on the network. The results provide us with a realistic representation of those nodes that have the highest risk of being extensions of known attacks.

4.4 Results

To test the efficacy of our approach, we conducted a series of experiments using intrusion detection data from a production network. The results show that our technique can be used to conduct a more complete analysis of the data produced by the intrusion detection infrastructure. The data consisted of all alarms produced within a 24-hour period. Our experiments were conducted over a 30-day period using data produced by 125 intrusion detection sensors. On average we observed 1,800 distinct source IP addresses and 1,000 target IP addresses per day. Note that for all examples, the true IP addresses have been obfuscated to protect the confidentiality of the subject network. The total number of alarms received at the SOC averaged 10,000 network IDS (NID) alarms, and 40,000 host IDS (HID) alarms per day. On average, computation of the ranks took between 2 to 5 minutes on a 1 CPU machine with 1Ghz processor and 2 Gbyte RAM, depending on the alarm volume for that day.

4.4.1 Emergence of Unseen Hosts and Forensic Analysis

During the course of our experiments we discovered that the vast majority of incidents were attributed to a small subset of the overall IP space. This has the adverse effect of causing the analysts to subconsciously focus on this familiar subset of IP addresses, and potentially overlook attacks occurring on other hosts. By using our algorithm, we were able to highlight newly emerging hosts for analysis. As the structure of the underlying Alarm Graph changed over time, new IP addresses moved to the top of the IP ranking automatically. Newly appearing hosts increased in rank and importance if they had a direct connection from an IP address that had been identified as a known attacker or target. This happened as a result of the new host inheriting a portion of the high rank associated with the known attacker or victim. A new host's rank also rose if it was the victim of a coordinated attack wherein it was targeted by multiple attackers. In either of these scenarios, our algorithm consistently marked these hosts as high risk.

4.4.2 Anomalous Alarm Pattern Recognition

By algorithmically identifying anomalous link patterns in the Alarm Graphs, we are able to highlight sets of alarms which have a higher probability of being genuine attacks. For example, the cluster of alerts shown in Figure 4.2 is an uncommon structure in the graph and represents the emergence of a Denial of Service (DoS) Attack.

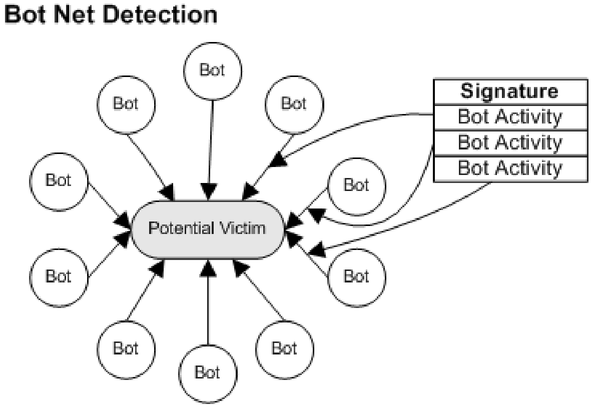
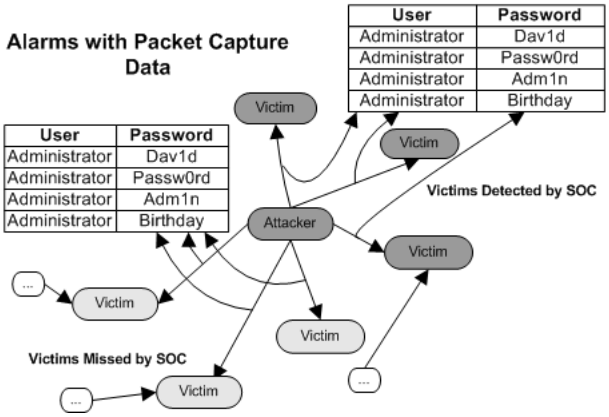


Figure 4.2: Probable denial of service attack

4.4.3 Identification of Missed Attacks

Figure 4.3 demonstrates the ability of our algorithm to discover attacks which were missed by the SOC. The darker nodes in the graph are those hosts for which a known incident had occurred. The ranks of these vertices were artificially inflated using the previously described technique. The lighter color nodes represent hosts which inherited these high ranks, and were marked for inspection by our algorithm, but had not been discovered by the SOC. This example shows a brute force dictionary attack against an FTP service running on multiple servers. The SOC detected a portion of this attack, and opened an incident record. However, the analyst only identified half of the victims of the attack. The upper half of Figure 4.3 illustrates those hosts which were marked as targets, while the lower left portion shows those which were missed. By elevating the rank of the attacking node, our algorithm highlighted the additional three hosts. Upon inspection, these were found to be victims of the same attack. We have included packet capture data from the alarms to further illustrate the attack.



User	Password
Administrator	Dav1d
Administrator	Passw0rd
Administrator	Adm1n
Administrator	Birthday

User	Password
Administrator	Dav1d
Administrator	Passw0rd
Administrator	Adm1n
Administrator	Birthday

Figure 4.3: Detection of partially identified dictionary attack

4.4.4 Automated Watch List Generation

Watch lists of suspicious IP addresses are maintained by the ESM and are used to monitor the alarm stream for any alerts generated by these hosts. Currently, these watch lists are populated manually. By using the results generated by our algorithm, it is now possible to build these watch lists automatically. By using the ranked output, we can successfully predict those IP addresses which have the highest probability of being involved in an attack during the subsequent day. Evaluation of our watch lists showed that on average we were able to successfully predict 83% of the security incidents that were manually flagged in a 30-day sample of historic alarm data. We calculated the threshold for which an IP was placed on the watchlist by determining the 97th percentile of the IP rankings for the day. Any IP address which was ranked at or above the threshold was automatically included on the watch list for the following day.

We define successful prediction of an incident as the inclusion of either the source or destination IP address of the alarms comprising that incident on a watch list produced by our algorithm. Using our algorithm, we were able to produce a list of those IP addresses which were suspicious based on the number distinct attackers, or because they were close to hosts which held high rank in the Alarm Graph and inherited a portion of this high ranking based on the recursive calculation of the PageRank algorithm. Figure 4.4 illustrates the performance of the watch lists generated via the ranking algorithm over a 30-day period.

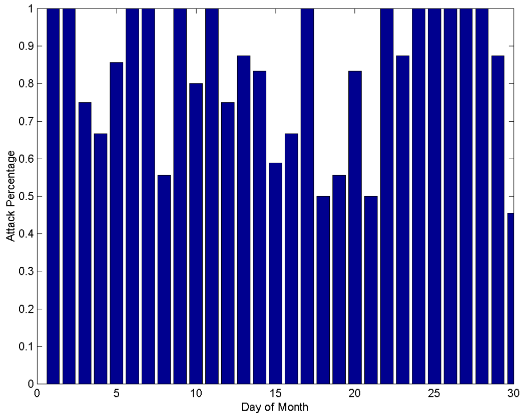
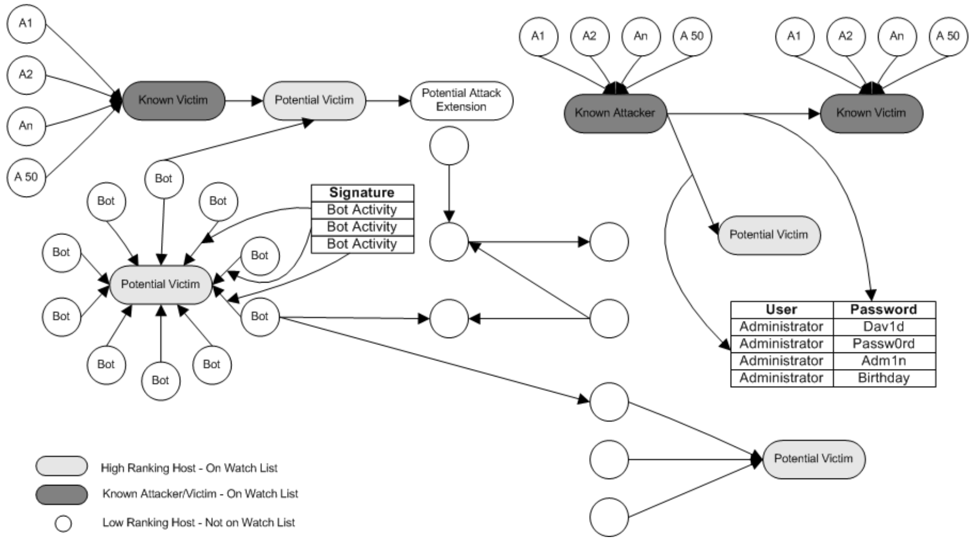


Figure 4.4: 30-day incident prediction trend

4.4.5 Facilitation of Sensor Tuning

The ranking algorithm sometimes repeatedly identified hosts that received a high rank, but were not involved in genuine attacks. When this behavior was observed over a period of time, we were able to use the patterns identified by the algorithm to filter the alarms that were causing the fictitious spikes. This type of filtering improves the overall effectiveness of the IDS infrastructure as it reduces the load on the ESM and the analysts, and improves the overall quality of the incoming alarms, resulting in a higher number of genuine attacks being detected.


A1
A2
An
A 50
Known Victim
Potential Victim
Potential Attack Extension
Known Attacker
Known Victim
Potential Victim
Bot
Potential Victim
Potential Victim
High Ranking Host - On Watch List
Known Attacker/Victim - On Watch List
Low Ranking Host - Not on Watch List

Signature
Bot Activity
Bot Activity
Bot Activity

User	Password
Administrator	Dav1d
Administrator	Passw0rd
Administrator	Adm1n
Administrator	Birthday

Figure 4.5: Colored Alarm Graph from production network, including auxiliary nodes and attack signatures

4.4.6 Visualization

Figure 4.5 shows a subgraph of an Alarm Graph generated from production IDS data. The full Alarm Graph is too large to display in a readable manner in print. This figure illustrates two known attacks. The nodes are colored so that the darker the color of the vertex, the higher its rank. The darkest vertices in the graph are those hosts which are known to be involved in attacks, and are shown with the corresponding auxiliary nodes added. Those vertices which are a lighter shade of gray have inherited high rankings, and will appear on the watch list generated at the end of the ranking routine. Additional gray nodes exist in the form of hosts which have received IDS alarms from multiple sources. These atypical patterns are caught by our ranking algorithm, and these hosts will appear on the watch list as well.

The visual representation of the colored Alarm Graphs provides a compact model that can be used by a human analyst to quickly triage the monitored network, providing visual cues as to which systems require immediate attention. Because the alarms are summarized into a single edge per pair of hosts for which an alarm was raised, the graphs grow slowly as compared to the overall alarm volume, and are easily understood for realistic networks.

4.4.7 Limitations

Certain type of attacks cannot be detected using our technique. These can be classified into the following categories.

1. **Atomic Attacks.** Attacks which are comprised of a single action are very difficult to detect using this technique. However, rules generally

exist in the ESM to automatically detect this type of attack. Once they are labeled in the data warehouse the ranking algorithm will detect any propagation of these attacks to other nodes.

2. **New Hosts.** In this situation, a new IP address appears in the alarm logs that has not been previously observed. Because the host was not previously in the alarm logs, it will not be included in any watch lists. This type of host can be detected using our technique for offline analysis if one of two conditions is true. First, if the host is a descendant of a node in the Alarm Graph which is known to be involved in an incident it will inherent a portion of the high rank and appear in the watch list. Secondly, the host will be flagged if it is linked to by a sufficient number of distinct attackers.

4.5 Conclusion

The PageRank algorithm, when applied to annotated Alarm Graphs, is a useful tool for efficiently and methodically analyzing large sets of intrusion detection alarms. Our technique provides an effective means of performing forensic analysis to uncover attacks which were overlooked during real-time monitoring. Additionally, we are able to generate watch lists of IP addresses which are known to have high risk of being involved in an attack. The watch lists are comprised of hosts that are in close proximity to a known attacker or victim, or that are a member of an anomalous structure in the Alarm Graph.

The incorporation of known attacks into our analysis allows us to drastically improve the quality of our results. Prior to annotating the Alarm Graphs with the incident data, the rankings produced were of minimal value, as the distributions reflected the random nature of the underlying graph. However, by including the attack data we are now able to highlight those hosts that deserve a higher rank. By forcing these high ranks, we are able to observe the ripple effect of malicious hosts throughout the network. This provides an effective means of decreasing the likelihood that an attack will be lost in the noise of the false alarms.

Chapter 5

Sensor Profiling

5.1 Intuition

Managed security service providers (MSSPs) must manage and monitor thousands of intrusion detection sensors. The sensors often vary by manufacturer and software version, making the problem of creating generalized tools to separate true attacks from false positives particularly difficult. Often times it is useful from an operations perspective to know if a particular sensor is acting out of character. Over time, IDS sensors show a consistent operating characteristic in terms of volume and types of alarms which are triggered. When a sensor departs from its normal operating mode, it is indicative of significant phenomena on the network, which is often the presence of an attack. For the following set of experiments, we define normal behavior as the sensor emitting its typical stream of false alarms, and attempt to detect departures from this profile as a means of attack detection.

We propose a solution to this problem using anomaly detection techniques over the set of alarms produced by the sensors. Similar to the manner in which an anomaly based sensor detects deviations from normal user or system behavior, we establish the baseline behavior of a sensor and detect deviations from this baseline. We show that departures from this profile by a sensor have a high probability of being artifacts of genuine attacks. We evaluate a set of time-based Markovian heuristics against a simple compression algorithm and show that we are able to detect the existence of all attacks which were manually identified by security personnel, drastically reduce the number of false positives, and identify attacks which were overlooked during manual evaluation.

During the course of our experiments, we evaluated the performance of three alarm evaluation heuristics comprised of single step Markov Chains, Hidden Markov Models, and a simple heuristic based on the GNU gzip utility. The underlying intuition to our approach is that intrusion detection sensors are inherently noisy, and although the false positives they generate appear random, the behavior of a given sensor will exhibit a "normal" behavior which can be modeled over time. We further hypothesize that deviations from this "normal" behavior have a high probability of being attacks. It is important to note that a model must be created for each sensor as the software versions, signature databases, and placement of the device can vary significantly across the installation base. As such, no general model can be created to cover the set of sensors for the entire network. A potential weakness of this approach is the likelihood that the alarms generated by a particular sensor will vary over time, especially in the case of a major software update to the device. Events of this nature will require retraining of the models which are used for the sensors.

To support our hypotheses, we adapt earlier work from the field of applied statistics. Schonlau, et al. evaluate the efficacy of statistical heuristics in detecting masqueraders via the statistical analysis of system call traces [73]. We adapt their approach to the analysis of IDS alarms, and show that Markov Chains and Hidden Markov Models prove to be very effective at detecting all types of attacks by acting as an anomaly detector over the set of IDS alarms. We also evaluate the compression technique described in [73] and show that while it is effective at intrusion detection, it yields a significantly higher percentage of false positives for this type of analysis. We do not evaluate the "Uniqueness" approach described by them because we do not perform our analysis on a per user or per IP address basis. Neither do we evaluate IPAM or the "Sequence Match" methods described in this paper, for similar reasons.

We chose to evaluate the alarm sequences on a per sensor basis as opposed to a per IP address basis. The IP space on any given network is extremely large, and analysis of the alarms generated per IP not only has the potential to require the modeling of millions of distinct IPs, but the number of alarms generated per IP address is not large enough to lend itself to training a model of any kind. In addition to this fact, we chose to model at the sensor level of aggregation to fix a potential weakness in Ourston's work [65, 66, 67]. While Ourston, et al. made a significant contribution to the field of intrusion detection by introducing the concept of Markovian modeling to the field of IDS alarm analysis, it is ineffective when an attacker changes his IP address. By aggregating alarms at the sensor level, our techniques are immune to this type of evasion.

5.2 Related Work

While it is well established that it is possible to detect attacks based on deviations from normal system behavior by modeling system calls, relatively little research has been performed in the area of profiling IDS sensors by modeling the alarms they emit. Previous work consists mainly of research which was performed by the IBM Zurich Research Lab on techniques for creating sensor profiles using Association Rules [2]. Our approach differs from this work in that our models take into account the order in which the alarms are generated by the sensor.

Krügel et al. propose a system for improving the accuracy of anomaly based IDS sensors using Bayesian networks [43]. Our approach differs from this in that Bayesian networks do not model the inter-dependence of alarms using time based sequencing. We show that there is a strong relationship between the order in which alarms are generated, and whether they in fact are the result of a genuine attack.

Hidden Markov Models have been applied in various ways to the problem of learning normal user or process behavior based on system call traces in Unix. These generally have extended earlier work on modeling traces of Unix system calls using N-grams [30], or Markov Chains [35]. The application of Hidden Markov Models to Unix system calls generated by operating system processes is explored in [23, 85]. Ju presents research on using HMMs to model user generated system calls in [36]. The application of HMMs to network data is explored in [5, 84].

Ourston, et al. [65, 67] present a technique for detecting multi-stage attacks using Hidden Markov Models and IDS alarms. The main weakness of their approach is its reliance on connection records, which are trivial to compromise if an attack originates from multiple source IPs or if an attacker spoofs their IP address. A secondary limitation of this approach is that they train the HMMs for positive response. If a new category of attack emerges, until a new HMM is trained for that attack, the alarm sequences falling into this category have a high probability of going undetected.

Haslum et al. present a technique for quantifying risk to a network based on the set of alarms from multiple intrusion detection sensors in [29] and use this merged alarm stream to calculate a risk score using a Hidden Markov Model. They extend their work in [27, 28] to build an intrusion prevention sensor which predicts whether an alarm sequence has a high probability of being followed by an alarm which will complete an attack scenario, and takes preventative action to mitigate the threat.

A general framework for the application of Markov Chains in anomaly based intrusion detection systems is given by Jha in [35] using system call data. Jha's framework is frequently adapted in subsequent research. Sallhammar presents a method for applying Markov Chains in conjunction with a cost and reward system for computing the probability of an attack based on game theory [72]. Khanna presents a novel approach for detecting attacks on mobile ad-hoc networks using Hidden Markov Models in [42]. Zanero uses a combination of Hidden Markov Models based on system call traces, and theory from the field of Ethology to create formalized characterizations of system interactions resulting in what he calls a “Behavioral Intrusion Detection System” [86].

The primary difference between our work and that of the prior art is that we build profiles which are intended to model normal behavior for a particular sensor. The majority of the prior research models specific attacks, or attack profiles, and attempts to make predictions based on those models [28]. In contrast, we model the baseline false positive noise of a sensor as “normal” behavior. By detecting deviations from this baseline, we are able to detect a change in the sensor state. We then show that this anomalous behavior has a high probability of representing malicious activity on the network.

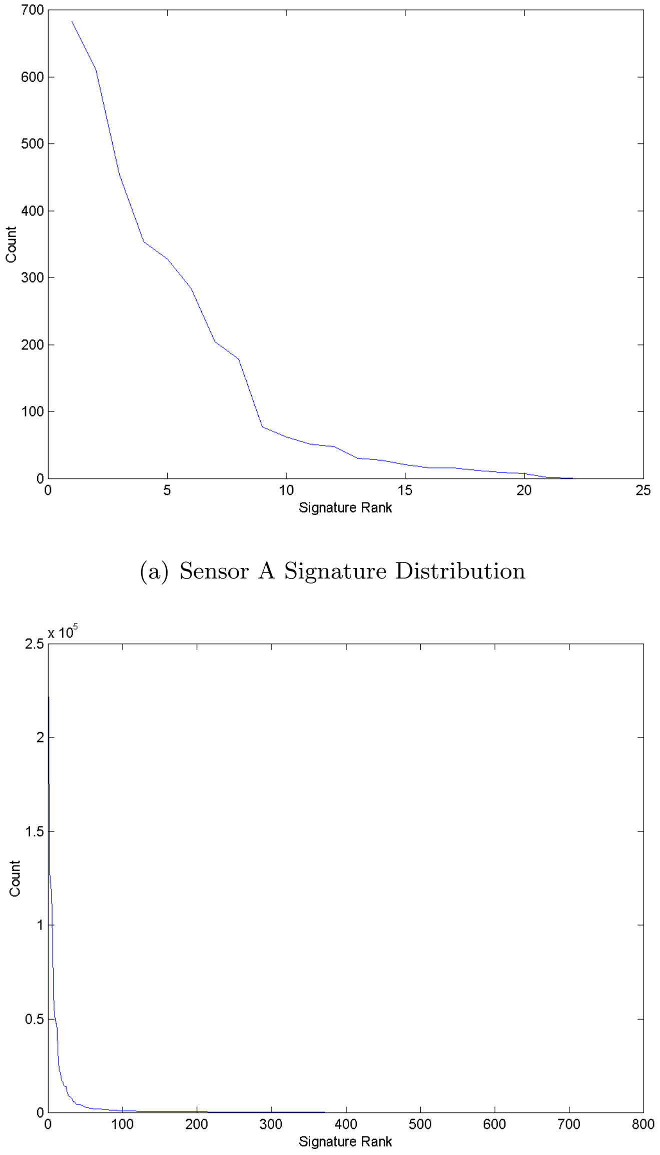
5.3 On Data and Experimental Design

Using supervised training techniques as described in [65, 67] is extremely difficult. Significant portions of these two papers are dedicated to preprocessing routines which generalize the base alarm data to a form where creating abstract models of attacks is feasible. Our approach is fundamentally different.

Because we have access to a large repository of known security incidents, we generate training sequences of observations based on periods of data which contain no known attacks. This approach carries the inherent risk that the SOC personnel overlooked an attack that may be present in a training sequence. We mitigate this risk as much as possible by using a large number of training sequences, and closely examining false positives that are detected on the training data to ensure that no attacks were inadvertently introduced during the training period.

We conducted our experiments on the set of alarms generated by two IDS sensors running in production mode on large corporate networks. We label these sensors sensor A and sensor B. Sensor A was a Cisco NetRanger network IDS. Sensor B was a SourceFire network IDS Sensor. We selected these two sensors for our experiments to demonstrate that our techniques were technology agnostic, and to compare our results on sensors which had received differing levels of filtering and tuning. We also wished to conduct our experiments on sensors which are representative of typical technologies in use in current corporate environments. We evaluated the set of alarms generated by each sensor for the 30-day period starting May 1, 2008 and ending May 30, 2008. Sensor A was tuned to be relatively quiet, and generated 3,483 alerts for this time period. Sensor B was not tuned as aggressively and monitored a larger network. Sensor B generated 172,839 alerts during the test period. During the test period, 17 of the 3,483 alarms generated by sensor A were reflective of true attacks. For sensor B, 308 alarms represented genuine attacks. For both sensors, the false positive rate was well over 99%, making the discovery of genuine attacks extremely difficult.

Table 5.1 shows a typical set of IDS alarms consisting of the IP address of the attacker, the IP address of the victim, the numeric signature ID, and the name of the signature for which the alarm was raised. The only column of significance for our analysis is Signature ID. In order to analyze the alarms we map the Signature ID field from the alarm to an identity field in a database which we custom built to facilitate this analysis. The mapped signature IDs are monotonically increasing integers, ranging from 1 to m



(a) Sensor A Signature Distribution

(b) Sensor B Signature Distribution

Figure 5.1: Signature frequency distributions

Source IP	**Destination IP**	**Signature ID**	**Signature**
10.0.0.1	10.0.0.4	**1**	TCP Port Scan
10.0.0.2	10.0.0.4	**1**	TCP Port Scan
10.0.0.3	10.0.0.4	**2**	Buffer Overflow
10.0.0.5	10.0.0.7	**3**	ftp brute force login

Table 5.1: Intrusion detection alarms for sensor profiling

where m represents the number of distinct signatures, i.e. the number of different attack types, for which an alarm was raised during the 30-day test period. $m = 22$ for sensor A and $m = 800$ for sensor B.

Figure 5.1(a) is a plot of the signature frequency distribution for sensor A over 30 days. Figure 5.1(b) is the same plot with alarm frequencies for sensor B. In both cases, the vast majority of the alarm traffic for the 30-day period is comprised of a relatively small number of signatures, and drops of quickly for the remainder of the signature set. We have evaluated many other production sensors and found this to be typical for any given IDS. Given this phenomenon, it is normal to see that each of the sequences of alarms that we test look very similar in composition. This is the main inspiration for our research. Given that the majority of the alarms generated by an IDS are the same signatures over and over, it makes sense that deviations from these alarms, and the order in which they appear, are indicative of a change in state of the sensor, i.e. from emitting false positives, to detecting genuinely malicious activity on the network. It is important to note that the set of alarms present in the training data, and those in the test data, are not mutually exclusive. If this were the case, separating legitimate alarms from false alarms would be the trivial exercise of simply filtering the "noisy" signatures. In fact, all signatures from the test data were represented in the training data as well. This further demonstrates that the order in which the alarms are generated is a significant indication of whether the alarms are false positives, or manifestations of an attack.

In order to build the data sets that were used in our experiments, we constructed training and test sequences in the following manner. Let $A = \{a_1, a_2, \ldots, a_n\}$ be the complete set of alarms generated by the sensor over the 30-day experimental time period. We subdivided A into two subsets consisting of training data R and test data S. We selected a period of days during which no known attacks were identified by the SOC and generated a set of training sequences R such that each $r_t \in R$ is a sub-sequence $\{r_t \ldots r_{t+k}\}$ beginning at time t. The set of sequences was generated using a sliding window of length k. For both sensor A and sensor B, this training data was comprised of the first 5 days of the month. We defined the set of test data as $S = \{A - R\}$ and generated test sequences $s_t \in S$ in the same manner as the training sequences. Originally we attempted to model the sensors in a state of silence by inserting a signature id of "0" for each second of the day during which no alarm was generated. Given that there are 86,400 seconds in a day, this had the effect of diluting the signal produced by the sensors to the point where analyzing the signal produced by the sensor became ineffective. As such, we made the decision to model only the actual signal, and not introduce the notion of silence to the models.

On average sensor A generated an alarm every 12 minutes, and sensor B generated an alarm every 17 seconds. This is the same approach used in [65, 66, 67, 73]. A good topic for future research would be to introduce continuous time Markov Chains to the set of experiments, and model the absence or presence of alarms as a Poisson process. This would provide a facility for analyzing bursty behavior by a sensor, or a normally noisy sensor which suddenly goes quiet, both of which are potential indicators of malicious activity on either the network, or the sensor itself.

5.3.1 Experimental Design

We evaluated three different techniques during the course of experiments, "Compression", "Single Step Markov Chain", and "Hidden Markov Model". All three of these methods attempt to detect anomalies in a stream of alarms generated by production intrusion detection sensors as a means of detecting attacks based on deviations from normal sensor behavior.

The methods share a common data foundation, in that the set of alarms is segmented into training data and test data, each of which are further subdivided into training and test alarm sequences. To facilitate discussion for the remainder of this chapter, we define the following notation:

$A = \{R \cup S\}$	The set of alarms generated by a sensor
R	The training data
r_t	The training sequence starting at time t
S	The test data
s_t	The test sequence starting at time t
M	The set of distinct signature IDs
m	The size of M
k	The length of the training and test sequences.

The sequence length evaluated during all experiments was 10. This choice appears somewhat arbitrary, but it was determined during the course of the experiments that a sequence length of 10 yielded the best results. We also evaluated sequence lengths of 2, 5, 15, 25, and 50, all of which yielded inferior results for both sensors.

5.4 Overview of Methods and Model Construction

5.4.1 Compression

Insight

The underlying intuition behind the compression method [73] is that test data which are appended to a training data set will yield a higher compression ratio if they are similar to the training data than if they vary significantly. This is due to the nature of the compression algorithm used in the gzip utility, as defined in [82]. The underlying Lempel-Ziv algorithm builds compression rules starting from the beginning of the file to be compressed. Given this fact, it makes sense that as these rules are built from the front of the file, data appended to the end of the file will compress more readily if it is similar in nature to the data which was used to build the rules. If the appended test data differs significantly from the training data, the compression ratio will suffer. Informally, this method tries to capture changes in entropy [75] as test data is appended to the training data.

Approach

To score this approach we define a score x_{s_t} for each test sequence $s_t \in S$ as the number of additional bytes required to compress the test sequences when appended to the training data R

$$x_{s_t} = gzip(R + s_t) - gzip(R)$$

Thresholds

The threshold used in the compression experiments was determined by calculating a set of cross validated scores x_t^{cv} for each sequence in the training data for both sensor A and sensor B. For each training sequence $r_t \in R$ we compute

$$x_t^{cv} = gzip(R + r_t) - gzip(R).$$

We fixed our target detection rate at 100% for known attacks and experimentally determined the appropriate threshold for each of the two sensors. The resulting thresholds were the 97th percentile for Sensor A, and the 89th percentile for Sensor B. When evaluating the test sequences $s_t \in S$, any sequence receiving a score $x_{s_t} > threshold$, was marked anomalous. The compression technique required the most relaxed thresholds of our three sets of profiling experiments. This is due primarily to the relatively small variance in the cross validated training scores. Tightening the thresholds resulted in significant degradation in detection accuracy.

Results

The compression algorithm, when applied to sensor A, generated 1021 meta-alarms, yielding an alarm reduction rate of 71%. When applied to sensor B, 19231 meta-alarms were generated, yielding an alarm reduction rate of 88%. Overall, the use of the gzip utility yielded the worst results of the three profiling techniques. Rather than relying on the gzip utility to perform the calculations, a formal investigation of the efficacy of entropy based anomaly detection on IDS alarms may yield better results, and warrants further exploration.

5.4.2 Markov Chains

Motivation

Markov Chains and Hidden Markov Models come from the field of signal processing, and have been used extensively in various speech recognition and machine learning applications. The benefit of these two techniques lies in the fact that they model the the order in which events occur in a training data set, and can be used to evaluate the probability of a sequence of events from a test data set. It is intuitive that the order in which alarms occur is important in the detection of attacks, and that this order will differ from the order in which alarms are generated as false positives. We show that detecting these changes is a very effective means of detecting attacks in a network, with a low rate of false positives, and a high rate of alarm reduction.

Model

Markov Chains are stochastic processes which are effective at modeling the behavior of a system over time. A complete discussion of Markov Chains is provided in [16]. We define a Markov Process as

$$\left\{X^{(n)}, n = 0, 1, 2, \ldots\right\}. \tag{5.1}$$

which take a finite or countable set M, in this case the integer signature IDs emitted by the sensors.

As in the compression technique, we define A as the total set of alarms emitted over the 30-day experimental time period. We further divided A into two subsets $R \subseteq A$, the attack free training data, and $S \subseteq A$, the test data. R and S are decomposed into sub-sequences using the same sliding window technique described for the compression experiments such that $r_t \in R$ is the training sequence starting at time t and $s_t \in S$ is the test sequence starting at time t. s_t and r_t are of the same predetermined length k. As such,

$$M = \{0, 1, 2, 3, 4, \ldots, m\}$$

which may be realized as the following, when generating sequences using a sliding window:

$$\begin{aligned} s_t &= \{5, 7, 5, 6, 6, 6, 2, 4, 7, 7\} \\ s_{t+1} &= \{7, 5, 6, 6, 6, 2, 4, 7, 7, 3\} \\ s_{t+2} &= \{5, 6, 6, 6, 2, 4, 7, 7, 3, 2\} \\ s_{t+3} &= \{6, 6, 6, 2, 4, 7, 7, 3, 2, 9\} \end{aligned}$$

$$\vdots$$

Definition 1. Suppose a fixed probability P_{ij} independent of time exists such that

$$P(X^{(n+1)} = j | X^{(n)} = i, X^{(n-1)} = j_{n-1}, \ldots, x^{(0)} = j_0) = P_{ij}, \quad n \geq 0$$

where $\{j, i, j_0, j_1, \ldots, j_{n-1}\} \in M$. Then this is called a Markov Chain process.

This probability can be interpreted as the conditional distribution of any future state $X^{(n+1)}$ given the past states

$$X^{(0)}, \quad X^{(2)}, \quad \ldots, \quad X^{(n-1)}$$

and present state $X^{(n)}$ is independent of the past states and depends solely on the present state. The probability P_{ij} thus represents the probability that the process will transition to state j given that it is currently in state i.

The transition probability P_{ij} is contained in a transition matrix, which holds the transition probabilities between all states in the Markov Chain.

$$P = \begin{pmatrix} P_{00} & P_{01} & \cdots \\ P_{10} & \ddots & \vdots \\ \vdots & \vdots & \vdots \end{pmatrix}$$

We use the technique of maximum likelihood to fit our data to the Markov Chain Model, and estimate the values of P. P is known as the one-step transition Matrix, and holds the probabilities of transition from one state to another state in a single step.

In order to determine the probability of being in a certain state n steps from now, we must calculate the n-step transition matrix. We call these probabilities the outlook probabilities. Using the transition values from P and an initial probability vector X, we are then able to calculate an "outlook" probability as follows

Definition 2. Let $X^{(n+1)} = PX^{(n)}$ be the probability distribution of the states one step from time n. We then know that $X^{(n+1)} = P^{(n+1)}X^{(0)}$ and $X^{(n+1)}$ holds the probabilities of being in a given state at time $n+1$, given the initial probability distribution $X^{(0)}$ and the one-step transition matrix P.

We are then able to determine the probability of a sensor emitting an alarm n steps from the current time (t).

Definition 3. Let X, the initial probability distribution vector be constructed in such a way that given a sequence of alarms $s_t \in S$ beginning with the signature id s_{t_0}, let $X^{(0)}_{s_{t_0}} = 1$ and all other $x \in X^{(0)} = 0$ indicating that the known starting state of the test sequence is s_{t_0} with probability 1. Given the one-step transition matrix P, we define the alarm outlook measurement to be

$$O_{s_t} = \prod_{n=1}^{k} P^{(n)} X^{(0)} \tag{5.2}$$

where t is the time of the first alarm in the sequence being evaluated, n

is the n^{th} element of the sequence, and $k = 10$ is the sequence length, as before.

Threshold

The threshold for the set of experiments using Markov Chains was calculated in the following manner. Given the one step transition matrix P, and an initial state probability vector $X^{(0)}$, for each training sequence $r_t \in R$ calculate $\left\{P(r_t)|P, X^{(0)}\right\}$ using equation (5.2). We calculated the 99.9th percentile of these scores, sorted highest to lowest, and marked any sequence as anomalous which had a probability lower than the threshold determined by the training data.

Results

For this set of experiments we were able to detect 100% of those attacks which were manually identified by the SOC using $k = 10$ as the sequence length for both sensors. In addition to accomplishing the automation of attack detection in the alarm logs, we were able to successfully identify multiple attacks and reconnaissance events which had gone unnoticed during manual inspection of the alarms. Over the 30-day period, the Markov Chain anomaly detector raised 482 meta-alarms for sensor A, yielding a suppression rate of 86%. For sensor B 1230 meta-alarms were generated, yielding a suppression rate of 99%.

5.4.3 Hidden Markov Models

Model

Hidden Markov Models were first proposed by Baum in [9, 10, 11, 12]. A Hidden Markov Model (HMM) is a doubly embedded stochastic process which models a set of symbol observations. Hidden Markov Models differ from basic Markov models in that the state which emits the observation is invisible, i.e. *hidden* from the observer. In a standard Markov process, the states themselves are visible to the observer. The observations in Hidden Markov Models are dependent on observation probability distributions at each hidden state, and transitions between the hidden states are governed by a secondary, hidden, stochastic process.

Rather than use the simple Markov Model described in the previous section, where each observation corresponds to a single state, the Hidden Markov model allows increased flexibility by modeling a set of observations as a probabilistic function of the current state, followed by a state change to either a new state, or the ability to remain in the current state prior to emitting the next observation, based on a state transition probability distribution. An in depth tutorial on Hidden Markov Models is presented by Rabiner in [71].

A Hidden Markov Model is defined by the following.

1. N. Let N denote the number of physical, hidden states of the model. This number is significant to some reality of state change in the real world which is represented in the model. We modeled the set of IDS alarms observed by the SOC in terms of a two state Hidden Markov

Model. We experimented with other numbers of states, but found that the use of a two state model yielded the best results.

2. m. Let m denote the number of distinct observations that can be emitted per state. m is thus the size of the alphabet M of symbols which are actually observed by the user of the system. For the sensor profiling problem, m is the number of distinct IDS alert signatures which are produced by the sensor. The observation symbols are given as $V = v_1, v_2, \ldots, v_m$.

3. α_{ij}. Let α_{ij} denote the transition probability distribution for the hidden states $\{H_1, H_2\}$ such that

$$\alpha_{ij} = P[q_{t+1} = H_j | q_t = H_i], \quad 1 \leq i, j \leq N. \tag{5.3}$$

4. $B = b_j(l)$ Let $B = b_j(l)$ denote the observation symbol probability distribution in a given state j such that

$$b_j(l) = P[v_l \quad at \quad t | q_t = H_j], \quad 1 \leq j \leq N, \quad 1 \leq l \leq M. \tag{5.4}$$

5. Let $\pi = \pi_i$ denote the initial state probability distribution such that

$$\pi_i = P[q_1 = H_i], 1 \leq i \leq N. \tag{5.5}$$

Given this set of parameters a Hidden Markov Model can be fully specified as $\lambda = (N, m, \alpha, B, \pi)$.

Once fully specified, Hidden Markov Models can be used to answer three canonical questions :

1. Given a trained HMM, what is the probability of a sequence of observations ?
2. What is the optimal sequence of hidden state changes to produce observations which emulate the training data?
3. How can we adjust the model to best achieve a set of training observations?

As in the Markov Chain experiments, the set of IDS alarms, A is divided into two sets of sequences R and S where $r_t \in R$ and $s_t \in S$ represent the sequence of length k at time t. The parameters α, B, π are all estimated using the Baum-Welch algorithm using the set of training sequences $r_t \in R$ [71]. We train the HMM using 5 days of IDS alarms for which no attacks are known to have occurred. Once trained, we are able to determine the probability score of a test sequence $s_t \in S$, the probability of a sequence of alarms, using the Forward Verterbi Algorithm [71] as

$$x_{s_t} = Verterbi(s_t) \tag{5.6}$$

Thresholds

To determine the threshold for marking test sequences as anomalous we calculated the score $x_{r_t} = Verterbi(R_t)$ for each sequence r_t in the training data R. As in the previous experiments, we fixed our target detection rate at

Technique	Sensor	Detection Rate	Alarm Reduction	Threshold
Compression	A	100%	71%	97
Compression	B	100%	88%	89
Markov Chain	A	100%	86%	99.9
Markov Chain	B	100%	99%	99.9
HMM	A	100%	93%	99.7
HMM	B	100%	95%	99.9

Table 5.2: Summary of findings for sensor profiling heuristics

100% for known attacks and adjusted the threshold to achieve this goal. In order to detect 100% of known attacks we set the threshold for Sensor A at the 99.7th percentile. For Sensor B, we were able to tighten this threshold to the 99.9th percentile and still achieve total attack detection. Any sequence from the test data s_t was marked as anomalous if $x_{s_t} < threshold$.

Results

Over the 30-day experimental time period, 239 meta-alarms were created for sensor A using the Hidden Markov Model approach, yielding an alarm suppression rate of 93%. For the same time period, 7813 meta-alarms were generated, yielding a 95% alarm reduction rate. The Hidden Markov Models successfully detected 100% of the genuine attacks that were manually identified by the SOC for sensor B, and 82% of the attacks for sensor A. As with the Markov Chain approach, we were able to detect attacks and reconnaissance activity which had gone unnoticed by the SOC.

5.5 Conclusions

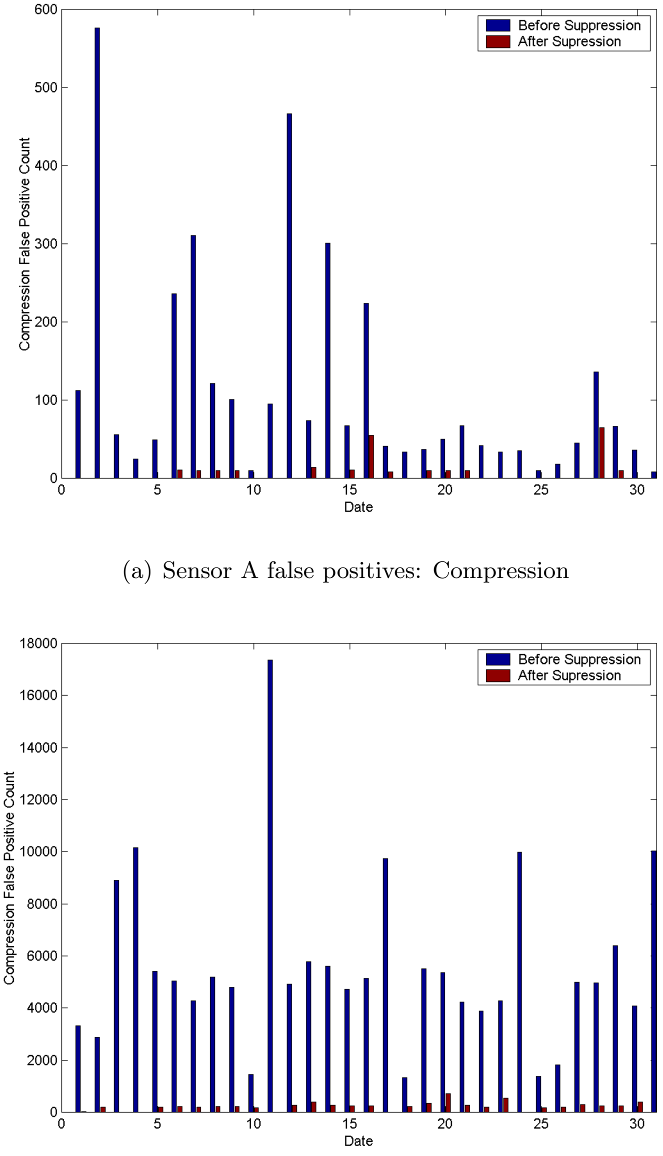
Table 5.2 summarizes the results of our experiments. In order to provide a stable point over which to compare the performance of the three profiling heuristics, we set the threshold for each technique at a value where we were able to detect 100% of the known attacks from the test data. As expected, Markov Chains and Hidden Markov Models out performed the use of compression to detect anomalies in sensor behavior. Markov Chains suppressed the greatest percentage of false alarms for the noisy sensor, Sensor B, eliminating 99% of the false positives. The use of Hidden Markov Models was more successful in eliminating false alarms on the quieter sensor, Sensor A, yielding a suppression rate of 93%.

The relatively small number of alarms produced by Sensor A, overall, made it more difficult to train the models. As such, the performance of the tested techniques for Sensor A is not as good as for Sensor B. This can be attributed to the smaller amount of training data, and the greater mean time between alarms emitted by this sensor.

It is interesting that the compression algorithm performed as well as it did, given the small sequence lengths which were evaluated. For example, compression missed only one attack comprised of a single alarm on a day where the other 4284 alarms were all false positives. By definition, these "one shot, one kill" attacks are extremely difficult to detect due to the small footprint they leave in the data. Because of this, it is not surprising that simple compression was not enough to detect the existence of such an attack. This attack was detected by both the Markov Chain and HMM techniques, solely because it represented an anomalous state transition in a sequence of alarms that would otherwise be representative of normal system behavior on the part of the sensor.

Overall, we were able to suppress a very large number of the false alarms which were generated by both sensors. This has the net effect of reducing the work load of SOC personnel, while increasing the accuracy of the monitoring infrastructure as a whole. Figures 5.2, 5.3, 5.4 show the base false alarm rates graphed against the false alarm rates of the Compression, Markov Chain, and Hidden Markov Model techniques, respectively.

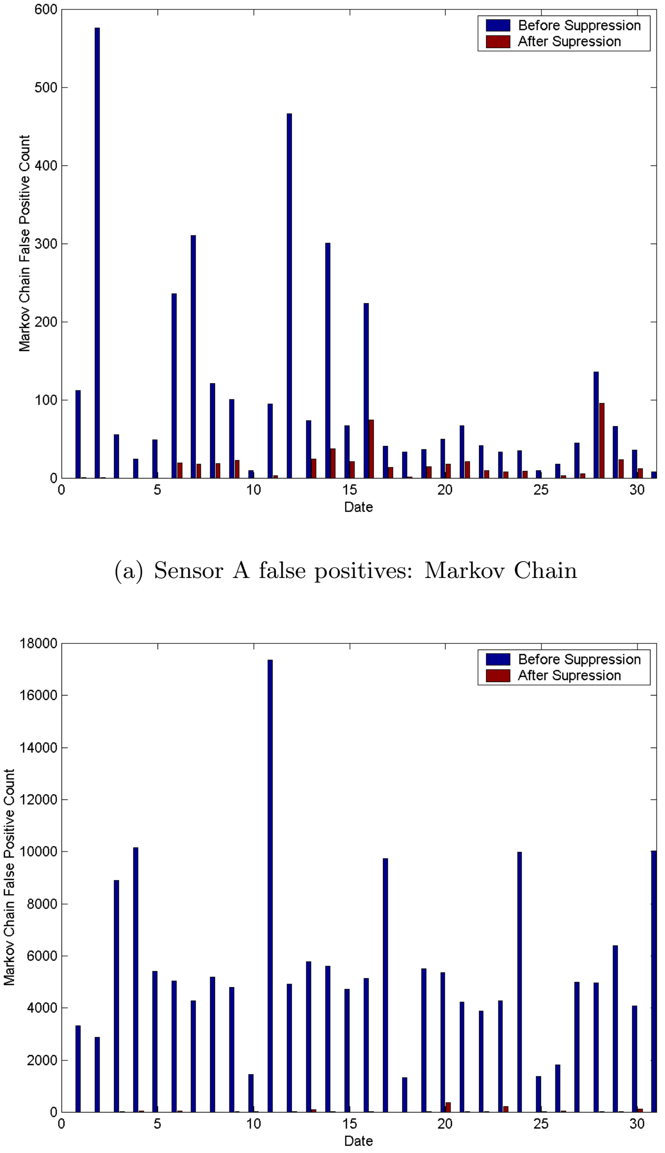
Interesting further research on this topic would involve exploration of the alarm rate produced by a sensor. It is intuitive that significant changes in the rate in which an IDS emits alarms could be indicative of attacks. The authors suggest exploring this problem in terms of Poisson processes and continuous time, multi-step Markov Chains.



(a) Sensor A false positives: Compression

(b) Sensor B false positives: Compression

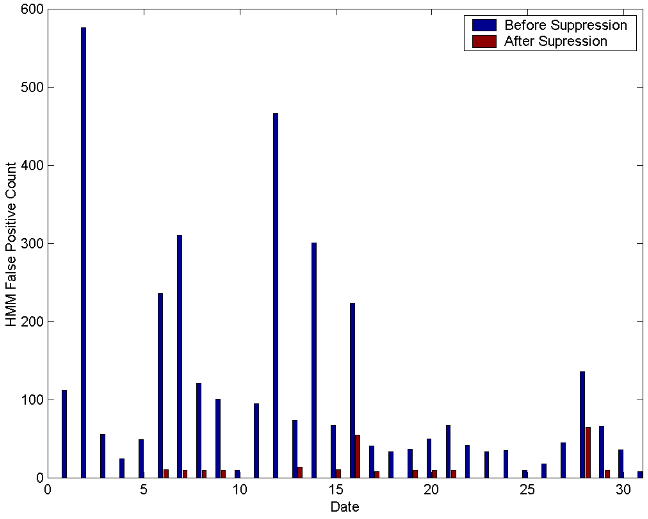
Figure 5.2: Compression false positive rates before and after suppression, by date



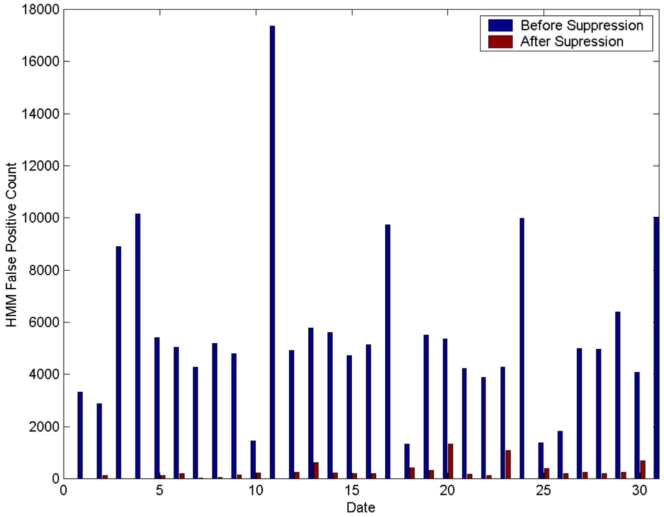
(a) Sensor A false positives: Markov Chain

(b) Sensor B false positives: Markov Chain

Figure 5.3: Markov chain false positive rates before and after suppression, by date



(a) Sensor A false positives: HMM



(b) Sensor B false positives: HMM

Figure 5.4: HMM false positive rates before and after suppression, by date

Chapter 6

Conclusion

6.1 Summary

The task of identifying malicious activity in large networks remains daunting. The complexity and skill of attackers continues to grow, fueled by the quest to turn cyber crime into a profitable enterprise. As cyber criminals continue to organize and set their sites on targets of increasing value, the stakes will become perpetually higher.

The rate at which devices are added to networks will continue to increase as the world becomes more and more connected. As the number of networked assets continues to grow, the threat of compromising these devices will rise simultaneously. Pervasive computing will lead to more monitoring appliances, which will lead to more alarms. While it is likely that the accuracy of intrusion detection devices will slowly improve over time, we have shown that the processing of IDS alarms continues to be an extremely difficult problem. Unfortunately, to date, very little progress has been made in reducing the false alarm rates of even the most widely tested and deployed sensors.

Competent tooling is the only reasonable defense that the SOC has in defending networks against the constant onslaught of attacks. Without it, alarms which are triggered by legitimate attacks run a high risk of being overlooked. Due to the high level of skill required to identify attackers, there is an increasing trend in the industry toward the hiring of Managed Security Service Providers (MSSPs) to monitor large corporate networks. MSSPs universally rely on full featured ESM systems to assist the SOC staff in analyzing the high volumes alarms that are generated each day. The heuristics presented in this dissertation have been proven to provide vital steps forward in increasing the capability of the analytical tools which are available for use by the SOC.

Our tools have been tested and validated in production environments at one of the world's largest MSSPs. During these tests, we demonstrated the ability to significantly reduce the over all number alarms which must be examined manually. More importantly, we have demonstrated the ability to uncover attacks that had previously gone undetected. We have also shown that given data on known security incidents, we were able illuminate portions of attacks that previously were unknown to the security staff.

6.2 Future Work

We suggest the following directions for future research.

Extending the association rules algorithm in Chapter 3 to include a notion of the order in which the alarms were generated may help to improve rule quality. A first attempt at this was made by Li, et al. in [49] who extended our earlier work to include a basic notion of time. Further exploration of these and other techniques for automated rule generation is warranted.

The exploration of alarm arrival rates is a natural extension to the sensor profiling problem. We suggest modeling the set of IDS alarms using continuous time Markov Chains and Poisson Processes to create additional capabilities for sensor profiling tools.

As mentioned in Chapter 5, a formal exploration of the changes of entropy in the alarm stream may provide useful insights into the ratio of the baseline false positive noise to the actual signal created when a sensor is detecting a genuine attack.

Bibliography

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 207–216, 1993.

[2] K. Ali, S. Manganaris, and R. Srikant. Partial classification using association rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 115–118, 1997.

[3] P. Ammann, D. Wijesekera, and S. Kaushik. Scalable graph based network vulnerability analysis. In *Proceedings of the 9th ACM Conference On Computer and Communications Security*, pages 217–224. New York: ACM Press, 2002.

[4] F. Apap, A. Honig, S. Hershkop, E. Eskin, and S. Stolfo. Detecting malicious software by monitoring anomalous windows registry accesses. In *Proceedings of Recent Advances in Intrusion Detection, 5th International Symposium*, pages 36–53, 2002.

[5] D. Ariu, G. Giacinto, and R. Perdisci. Sensing attacks in computers networks with hidden markov models. In *MLDM*, pages 449–463, 2007.

[6] M Artz. Netspa: A network security planning architecture. Master's thesis, Massachusetts Institute of Technology, 2002.

[7] AT&T. Graphviz. http://www.graphviz.org.

[8] D. Barbara, J. Couto, S. Jajodia, and N. Wu. Adam: A testbed for exploring the use of data mining in intrusion detection. *SIGMOD Record*, 30(4):15–24, 2001.

[9] L. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3:1–8, 1972.

[10] L. Baum and J. Egon. An inequality with applications to statistical estimation for probabilistic functions of a markov process. *Bulletin of the American Meteorological Society*, 73:360–363, 1967.

[11] L. Baum, G. Petrie, T. andSoules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Ann. Math. Stat.*, 41:164–171, 1970.

[12] L. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Ann. Math. Stat.*, 30:1554 – 1563, 1966.

[13] S. Brin and L. Page. The anatomy of a large scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[14] S. Chakrabarti, B. Dom, D. Gibsony, J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Mining the link structure of the world wide web. *IEEE Computer*, 32(8), 1999.

[15] S. Cheung, R. Crawford, M. Dilger, and et al. The design of grids:a graph-based intrusion detection system. *U.C. Davis Computer Science Department Technical Report CSE-99-2*, 1999.

[16] W. Ching and M. Ng. *Markov Chains Models, Algorithms, and Applications*. Springer Science+Business Media, Inc., New York, 2006.

[17] W. Cohen. Fast effective rule induction. In *ICML*, pages 115–123, 1995.

[18] Arcsight Corporation. Arcsight esm product brief, 2005.

[19] Arcsight Corporation. Arcsight pattern discovery product brief, 2005.

[20] IBM Corporation. *IBM DB2 Intelligent Miner Modeling Administration and Programming Guide v8.2*. IBM Press, New York, NY, second edition, 2004.

[21] F. Cuppens and R. Ortalo. Lambda: A language to model a database for detection of attacks. In *Proceedings of the 3rd Annual International Symposium On Recent Advances in Intrusion Detection*, Berlin, Germany, 2000.

[22] H. Debar and A. Wespi. Aggregation and correlation of intrusion-detection alerts. In *Proceedings of Recent Advances in Intrusion Detection, 4th International Symposium*, pages 85–103, 2001.

[23] Y. Du, H. Wang, and Y. Pang. Hmms for anomaly intrusion detection. In *CIS*, pages 692–697, 2004.

[24] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, pages 27–34, 1996.

[25] G. Grimmet and D. Stirzaker. *Probability and Random Processes.* Clarendon Press, Oxford, 1992.

[26] Y. Guan, A. Ghorbani, and N. Belacel. Y-means : A clustering method for intrusion detection. In *Proceedings of Canadian Conference on Electrical and Computer Engineering*, 2003.

[27] K. Haslum, A. Abraham, and S. Knapskog. Dips: A framework for distributed intrusion prediction and prevention using hidden markov models and online fuzzy risk assessment. In *IAS*, pages 183–190, 2007.

[28] K. Haslum, A. Abraham, and S. Knapskog. Fuzzy online risk assessment for distributed intrusion prediction and prevention systems. In *Tenth International Conference on Computer Modeling and Simulation, UKSIM*, pages 216–223, 2008.

[29] K. Haslum and A. Årnes. Multisensor real-time risk assessment using continuous-time hidden markov models. In *CIS*, pages 694–703, 2006.

[30] S. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. *Journal of Computer Security*, 6(3):151–180, 1998.

[31] A. Honig, A. Howard, E. Eskin, and S. Stolfo. *Adaptive Model Generation : An Architecture for the Deployment of Data Mining-based Intrusion Detection Systems*, pages 153–194. Kluwer Academic Publishers, Boston, 2002.

[32] V. Hosel and S. Walcher. Clustering techniques : A brief survey. *Technical Report, Institute of Biomathematics and Biometry*, 2000.

[33] K. Ingols, R. Lippmann, and K. Piwowarski. Practical attack generation for network defense. In *Proceedings of the 22nd Annual Computer Security Applications Conference*, Miami Beach, FL, 2006.

[34] S. Jajodia, S. Noel, and B. O'Berry. *Topological Analysis of Network Attack Vulnerability.* Kluwer Academic Publisher, Dodrecht, Netherlands, 2003.

[35] S. Jha, K. Tan, and R. Maxion. Markov chains, classifiers, and intrusion detection. In *CSFW*, pages 206–219, 2001.

[36] W-H. Ju and Y. Vardi. A hybrid high-order markov chain model for computer intrusion detection. *National Institute of Statistical Science Technical Report Number 92*, 1999.

[37] K. Julisch. Mining alarm clusters to improve alarm handling efficiency. In *Proceedings of the 17th Annual Computer Security Applications Conference*, pages 12–21, 2001.

[38] K. Julisch. *Data Mining for Intrusion Detection A Critical Review*, pages 33–62. Kluwer Academic Publishers, Boston, MA, 2002.

[39] K. Julisch. Clustering intrusion detection alarms to support root cause analysis. *ACM Transactions on Information and System Security*, 6(4):443–471, 2003.

[40] K. Julisch. *Using Root Cause Analysis to Handle Intrusion Detection Alarms*. PhD thesis, Universitat Dortmund, 2003.

[41] K. Julisch and M. Dacier. Mining intrusion detection alarms for actionable knowledge. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 366–375, 2002.

[42] R. Khanna and H. Liu. Distributed and control theoretic approach to intrusion detection. In *IWCMC*, pages 115–120, 2007.

[43] C. Krügel, D. Mutz, W. Robertson, and F. Valeur. Bayesian event classification for intrusion detection. In *ACSAC*, pages 14–23, 2003.

[44] R. Krutz and D. Russel. *The CISSP Prep Guide*. Wiley Publishing, Inc., Indianapolis, 2003.

[45] W. Lee and S. Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, pages 79–94, 1998.

[46] W. Lee, S. Stolfo, P. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop, and J. Zhang. Real time data mining-based intrusion detection. In *Proceedings of the 2nd DARPA Information Survivability Conference and Exposition*, 2001.

[47] W. Lee, S. Stolfo, and M. Kui. A data mining framework for building intrusion detection models. In *IEEE Symposium on Security and Privacy*, pages 120–132, 1999.

[48] W. Lee, W. Stolfo, and K. Mok. Mining audit data to build intrusion detection models. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 66–72, 1998.

[49] Z. Li, A. Zhang, D. Li, and L. Wang. Discovering novel multistage attack strategies. In *ADMA*, pages 45–56, 2007.

[50] R. Lippmann, J. Haines, D. Fried, J. Korba, and K. Das. The 1999 darpa off-line intrusion detection evaluation. In *Computer Networks*, volume 34, pages 579–595, 2000.

[51] R. Lippmann and K. Ingols. An annotated review of past papers on attack graphs. Technical Report ESC-TR-2005-054, MIT Lincoln Laboratory Technical Report, 2005.

[52] S. Manganaris, M. Christensen, D. Zerkle, and K. Hermiz. A data mining analysis of rtid alarms. In *Proceedings of the Second International Workshop on Recent Advances in Intrusion Detection*, West Lafayette, IN, 1999.

[53] S. Mathew, R. Giomundo, and S. Upadhyaya. Real-time multistage attack awareness through enhanced intrusion alert clustering. In *Proceedgins of SIMA 2005*, Atlantic City, NJ, 2005.

[54] J. Mchugh. Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security*, 3(4):262–294, 2000.

[55] S. McLure, J. Scambray, and G. Kurtz. *Hacking Exposed Fifth Edition: Network Security Secrets & Solutions*. McGraw-Hill/Osborne, 2005.

[56] V. Mehta, C. Bartzis, H. Zhu, E. Clarke, and J. Wing. Ranking attack graphs. In *Proceedings of the 9th Annual International Symposium On Recent Advances in Intrusion Detection*, pages 127–144, Hamburg, Germany, 2006.

[57] A. Moore, R. Ellison, and R. Linger. Attack modeling for information security and survivability. *CMU Software Engineering Institute Techinical Report CMU/SEI-2001-TN-01*, 2001.

[58] K. Nauta and F. Lieble. Offline network intrusion detection: Mining tcpdump data to identify suspicious activity. In *Proceedings of the AFCEA Federal Database Colloquium*, 1999.

[59] P. Ning, Y. Cui, and D. Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, Washington, D.C., 2002.

[60] P. Ning, Y. Cui, D. Reeves, and D. Xu. Techniques and tools for analyzing intrusion alerts. *ACM Transaction on Information and System Security*, 7(2):274–318, 2004.

[61] P. Ning and D. Xu. Learning attack strategies from intrusion alerts. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, pages 200–209, New York, NY, 2003.

[62] P. Ning, D. Xu, C. Healey, and R. St. Amant. Building attack scenarios through integration of complementary alert correlation method. In *NDSS*, 2004.

[63] S. Noel and S. Jajodia. Managing attack graph complexity through visual hierarchical aggregation. In *IEEE Workshop on Visualization for Computer Security*, 2004.

[64] S. Noel, D. Wijesekera, and C. Youman. *Modern Intrusion Detection, Data Mining, and Degrees of Attack Guilt*, pages 1–31. Kluwer Academic Publishers, Boston, MA, 2002.

[65] D. Ourston, S. Matzner, W. Stump, and B. Hopkins. Applications of hidden markov models to detecting multi-stage network attacks. In *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS)*, 2003.

[66] D. Ourston, S. Matzner, W. Stump, and B. Hopkins. Applications of hidden markov models to detecting multi-stage network attacks. In *HICSS*, page 334, 2003.

[67] Dirk Ourston, Sara Matzner, William Stump, and Bryan Hopkins. Coordinated internet attacks: responding to attack complexity. *Journal of Computer Security*, 12(2):165–190, 2004.

[68] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web, 1999.

[69] T. Pietraszek. Using adaptive alert classification to reduce false positives in intrusion detection. In *Proceedings of the 7th Annual International Symposium On Recent Advances in Intrusion Detection*, pages 102–124, Sophia Antipolis, France, 2004.

[70] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering. In *ACM Workshop on Data Mining Applied to Security (DMSA 2001)*, 2001.

[71] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[72] K. Sallhammar, B. Helvik, and S. Knapskog. On stochastic modeling for integrated security and dependability evaluation. *JNW*, 1(5):31–42, 2006.

[73] M. Schonlau, W. DuMouchel, W. Ju, A. Karr, M. Theus, and V. Yehuda. Computer intrusion: Detecting masquerades. *Statistical Sciences*, 16(1):1–17, 2001.

[74] M. Schultz, E. Eskin, E. Zadok, and S. Stolfo. Data mining methods for detection of new malicious executables. In *Proceedings of IEEE Symposium on Security and Privacy*, 2001.

[75] C. Shannon. A mathematical theory of communication. In *Bell System Technology Journal*, pages 379–423,623–656, 1948.

[76] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. Wing. Automated generation and analysis of attack graphs. In *IEEE Symposium on Security and Privacy*, 2002.

[77] S. Stolfo, W. Lee, P. Chan, W. Fan, and E. Eskin. Data mining-based intrusion detectors: An overview of the columbia ids project. *SIGMOD Record*, 30(4):5–14, 2001.

[78] Cisco Systems. Network security database, 2005.

[79] S. Templeton and K. Levitt. A requires/provides model for computer attacks. In *Proceedings of New Security Paradigms Workshop*, pages 30–38, 2000.

[80] A. Valdes and K. Skinner. Probabilistic alert correlation. In *Proceedings of Recent Advances in Intrusion Detection, Third International Workshop*, pages 54–68, 2001.

[81] F. Valeur, G. Vigna, C. Krügel, and R. Kemmerer. A comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on Dependable and Secure Computing*, 1(3):146–169, 2004.

[82] T.A. Welch. A technique for high performance data compression. *IEEE Computer*, pages 8–18, 1984.

[83] D. Yang, C. Hu, and Y. Chen. A framework of cooperating intrusion detection based on clustering analysis and expert system. In *Proceedings of the 3rd international conference on Information Security*, 2004.

[84] Y. Yasami, M. Farahmand, and V. Zargari. An arp-based anomaly detection algorithm using hidden markov model in enterprise networks. In *ICSNC*, page 69, 2007.

[85] N. Ye, Y. Zhang, and C. Borror. Robustness of the markov-chain model for cyber-attack detection. *IEEE Transactions on Reliability*, 53(1):116–123, 2004.

[86] S. Zanero. Behavioral intrusion detection. In *ISCIS*, pages 657–666, 2004.

[87] J. Zhang and M. Zulkernine. A hybrid network intrusion detection technique using random forests. In *ARES*, pages 262–269, 2006.