2013

Application of a Layered Hidden Markov Model in the Detection of Network Attacks

Lawrence Taub
Nova Southeastern University, lawrencetaub@gmail.com

Application of a Layered Hidden Markov Model in the Detection of Network Attacks

by

Lawrence Taub

A dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in
Computer Information Systems

Graduate School of Computer and Information Sciences
Nova Southeastern University
2013

An Abstract of a Dissertation Submitted to Nova Southeastern University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Application of a Layered Hidden Markov Model in the Detection of Network Attacks

by
Lawrence Taub

March 2013

Network-based attacks against computer systems are a common and increasing problem. Attackers continue to increase the sophistication and complexity of their attacks with the goal of removing sensitive data or disrupting operations. Attack detection technology works very well for the detection of known attacks using a signature-based intrusion detection system. However, attackers can utilize attacks that are undetectable to those signature-based systems whether they are truly new attacks or modified versions of known attacks. Anomaly-based intrusion detection systems approach the problem of attack detection by detecting when traffic differs from a learned baseline. In the case of this research, the focus was on a relatively new area known as payload anomaly detection. In payload anomaly detection, the system focuses exclusively on the payload of packets and learns the normal contents of those payloads. When a payload's contents differ from the norm, an anomaly is detected and may be a potential attack. A risk with anomaly-based detection mechanisms is they suffer from high false positive rates which reduce their effectiveness. This research built upon previous research in payload anomaly detection by combining multiple techniques of detection in a layered approach. The layers of the system included a high-level navigation layer, a request payload analysis layer, and a request-response analysis layer. The system was tested using the test data provided by some earlier payload anomaly detection systems as well as new data sets. The results of the experiments showed that by combining these layers of detection into a single system, there were higher detection rates and lower false positive rates.

Acknowledgments

I would like to give thanks to my advisor, Dr. Cannady, and my entire committee, Dr. Ford and Dr. Mukherjee, for their support during this process. Through their guidance, advice, and insightful questioning of my work, I have grown tremendously while performing this research.

I would also like to give thanks to my family who supported me during my late night and early morning coding, testing, and writing sessions. My wife, Amanda, provided support and love while my children, Julia and Ethan, provided love and (very necessary) distraction. The support from everybody else in my extended family has been invaluable to keeping my motivation during this marathon.

I am deeply indebted to my friends and coworkers who provided a sounding board when I had a crazy, new idea. In particular, Nick Biasini let me bounce some of the craziest ideas off of him and helped me to think my ideas through. Todd Blackwell provided sanity checks and helped me with his immense Java knowledge when I needed code efficiency improvement or had to debug and fix an edge case bug in one of the libraries.

Davide Ariu was a large help to me with his discussion of the McPAD and HMMPayl systems. I am especially grateful for him providing the code for both systems to me that allowed me to test my detection in comparison with his systems' capabilities.

I also am grateful to my employer who provided the data necessary for testing during this research as well as aiding with the tuition costs for my degree.

I dedicate this work to my grandparents who were unable to be here to see the completion of this journey.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Background

As modern society makes use of computing technologies in more ways, the usage of computers to store sensitive or critical data continues to increase. Computers are used for operations critical to the functions of modern society such as the power grid and air traffic. Similarly, computers are used to store sensitive government documents and national security information such as a design for a new fighter plane (Souren, 2013). These computers are often networked to other computers and these networks may be connected to external networks through the Internet. With this increasing connectivity comes the added risk of attackers making use of this connectivity to tamper, disrupt, or gain access to this sensitive or critical data.

The most basic attackers are often thwarted because of their use of familiar attacks. Familiar attacks can be detected through the use of attack signatures. Attack signatures are a mechanism that specifies key components of the attack in order to accurately identify the attack with as little chance as possible for a false positive (Denning, 1987). However, more advanced attackers that use unfamiliar attacks are not so easily stopped by the use of signatures. The attacks used by an advanced attacker will be specifically designed to avoid detection and will not be found by signatures.

A common advanced attack methodology used is that of the 0-day attack (Wang, Cretu, & Stolfo, 2005). A 0-day attack is named such because it is an attack that has not yet been disclosed and thus cannot be adequately mitigated or detected through signature methods. All programs have the possibility for code-based vulnerabilities with the possibility increasing as the complexity of the code increases. Because of this, the continued discovery of new 0-day attacks is a virtual certainty. Another advanced attack methodology may consist of familiar attacks that are restructured in some way to evade detection from attackers. These attacks are named variant attacks (Wagner & Soto, 2002). At a holistic level, the attack may be the same attack as before but the variation in its construction allows it to become undetectable by standard signature-based methods.

The typical means for detecting attacks, whether familiar attacks, 0-day attacks, or variant attacks, is by using an intrusion detection system (IDS) (Heberlein et al., 1990). A Network-based IDS (NIDS) scans network traffic for potential attacks. Should the IDS detect an attack, it creates an alert that notifies the operator for further investigation. A variation on the IDS allows for it to sit in-line with the network traffic and add the functionality of immediately blocking any attack detected. This variation is called an intrusion prevention system (IPS) (Plato, 2004). At its core, an IPS behaves similarly to an IDS with the addition of the active mitigation features that allow an IPS to block the detected attacks.

An IDS typically performs attack detection using one of two methods. The first method is known as a signature-based method (Kemmerer & Vigna, 2002). The signature-based method matches an operator-created signature in traffic to identify attacks. The other method is an anomaly-based method (Marin & Allen, 2006). The

anomaly-based method examines all traffic to learn what is typical for a network then will detect deviations or anomalies from the normal traffic. When one of these anomalies is detected, the system creates an alert for an operator to investigate.

A recent offshoot of anomaly detection is the subfield payload anomaly detection in which the payload contents of packets are analyzed to detect attacks at the application layer. A model for comparing the payload anomaly detection system against the traditional NIDS is the OSI Model. The OSI Model is a model which characterizes network traffic into different logical layers (Day & Zimmermann, 1983). In terms of the OSI Model, a typical NIDS will operate at the third layer, the network layer. A payload anomaly detection system will operate at the seventh layer, the application layer.

The first major payload anomaly detection system was the PAYL algorithm. The PAYL algorithm created an intrusion detection system focused on detecting 0-day worms essentially using a histogram of bytes or n-grams of size 1 (Parekh, Wang, & Stolfo, 2006; Wang et al., 2005; Wang, Parekh, & Stolfo, 2006). Anagram followed up on PAYL by granting the ability to handle n-grams greater than 1 (Wang et al., 2006). The next major system was the McPAD system (Perdisci, Ariu, Fogla, Giacinto, & Lee, 2009) which built upon the n-gram model using a size of 2 and added the capability for space of varying size between the bytes while also modeling the collected n-grams in a Support Vector Machine (Scholkopf, Platt, Shawe-Taylor, Smola, & Williamson, 2001). The next two systems, Spectrogram (Song, Keromytis, & Stolfo, 2009) and HMMPayl (Ariu, Tronci, & Giacinto, 2011), were developed in parallel and focus similarly on using Markov Models (Markov, 1971) to detect attacks. A major difference in the two is that Spectrogram has knowledge of the structure of an HTTP payload and learns the specific

parameters while HMMPayl learns the entire request payload. As a whole, these detectors have been shown to be far more capable of detecting application layer attacks than the prior approaches taken in anomaly detection.

Additionally, other variants of network-based IDS exist that use other models for detection, such as biologically inspired IDS. Some examples of the biologically inspired model are provided by the human body's immune system (Hofmeyr, Forrest, & Somayaji, 1998) or the concept of danger theory (Aickelin, Bentley, Cayzer, Kim, & McLeod, 2003). Researchers have created many different models for attack detection. Each model has its own successes or lessons learned. These lessons have been incorporated in subsequent IDS to refine and improve upon the concept of attack detection.

Problem Statement

Current network-based anomaly detection systems suffer from high false positive rates and high false negative rates and thus are unreliable in production usage, either providing analysts with too many false alarms or failing to alert analysts to significant attacks. High false positive rates create significant amounts of noise for an analyst to investigate and determine if the alert is a legitimate attack or not. This amount of extra work can lead to missed true attacks. Low detection rates means true attacks have no corresponding alert and are not properly classified as anomalies therefore it is highly unlikely the analyst will have the ability to detect them (Axelsson, 2000).

With each generation of IDS and refinement of the underlying models, the false positive rates are reduced and the detection rates are improved. While the users' expectation is not for an oracle that provides a perfect assessment of every packet, there is a reasonable expectation for high levels of accuracy. Though current systems have advanced significantly over prior systems in their accuracy, they have not yet reached a high enough level of accuracy (Ingham & Inoue, 2007).

Goal

The goal of this research was to construct a payload anomaly detection system by combining multiple techniques of detection in a layered approach combining the requested URLs, request payloads, and a combination of the requested URLs and the responded payloads, demonstrate the system's viability, and demonstrate the increased detection capabilities the layered model had over current state of the art detection systems that typically use just one of the layers. Experiments using both the system created through this research and state of the art payload anomaly detection systems allowed for a direct evaluation of the system created through this research in comparison to other systems. The systems were evaluated using the receiver operating characteristic model which presents a simplified graph showing the detection capabilities of a system by plotting the false positive rate against the detection rate (Green & Swets, 1966).

Relevance and Significance

As long as there are assets of value, there will be attackers who want to take those assets. This axiom applies to physical objects of value such as jewelry as well as digital objects of value such as a person's bank account information or sensitive government communications. These examples are a small subset of all possible digital assets that might be of value to an attacker. To protect those digital assets, network owners employ many different attack detection and defense mechanisms. The focus of this work is on anomaly detection systems specifically detection within the payload.

Earlier anomaly detection systems focused on characteristics of the traffic contained within the header or high level statistics about the packet such as number of bytes (Wenke, Stolfo, & Mok, 1999). This was a logical decision at the time as many of the attacks were present in the header of a network packet or not well disguised. However, as the avenues of attack in the header were closed, attackers shifted their method from the header to the content of the packet in the payload. The payload is the communication that is being delivered from one host to another specifically at the application layer: allowing an application on one host to communicate to the application on the other. Attackers found a fertile ground for attacks in the application layer as each application introduces a new set of potential vulnerabilities.

Anomaly detection systems shifted their focus from the headers to the payloads and began detecting application layer attacks to match the attackers' shift. The payload anomaly detection systems evolved from the statistical modeling of single bytes (Parekh et al., 2006; Wang et al., 2005; Wang et al., 2006) to multiple bytes (Wang et al., 2006) to byte pairs separated by some distance (Perdisci et al., 2009). The next and current

generation of payload anomaly detection systems added complex statistical models to the detection mechanism favoring the Markov models (Markov, 1971) whether in Markov Chains (Song et al., 2009) or Hidden Markov Models (Ariu et al., 2011). Payload anomaly detection systems are the state of the art in anomaly detection intrusion detection as they represent the best detection mechanism for application layer attacks.

This research aimed to improve upon those state of the art systems by combining multiple layers of detection. The layers are a high-level navigation layer, a request payload analysis layer, and a request-response payload analysis layer. The high-level navigation layer is an approach that has been proven in the context of distributed denial of service attacks (Yi & Shun-Zheng, 2009). The request payload analysis layer has also been proven in the detection of attacks (Ariu et al., 2011). The request-response payload analysis layer is a novel detection mechanism that was tested through this research. Further, the combination of these three layers is also a novel combination and was tested through this research.

As attackers increase in sophistication, focus, and resources, the need for more accurate detection mechanisms protecting networks increases. Anomaly detection systems must improve their detection performance by lowering false positives and increasing true positives in order to best protect digital assets.

Barriers and Issues

The area of anomaly-based intrusion detection is a challenging area of study due to the expansiveness of the possible attacks. Accurately and consistently detecting the

elusive parameters that characterize an attack with perfect accuracy for all attacks remains an unachieved goal. The possibility of ever achieving this goal was posited as a question in the infancy of the study area (Denning, 1987). In 2007, Gates and Taylor followed up on Denning's goals and questioned whether Denning's goal of accurate and consistent attack detection is even achievable (Gates & Taylor, 2007). Similarly, Axelsson (2000) has indicated that accurate intrusion detection is a very hard problem due to the difficulty in minimizing the false alarm rate. The challenge of building good enough intrusion detection as outlined by Axelsson is not one to be taken lightly. The lesson taken from Axelsson that minimizing false positive rates is a significant criterion for usefulness for all anomaly-based intrusion detection systems. As stated previously, the goal of this research was to minimize the false positives and maximize the true positives meeting or bringing the state of the art closer to Axelsson's criterion for successful intrusion detection.

Testing an intrusion detection system is a difficult task. The testing of anomaly detection systems is a disputed area. There is no agreed upon approach or data set that allows for consistent testing. The data sets from the KDD Cup in 1999 (Lippmann et al., 2000; Newman, 1999) were used for test purposes at the time of the competition but have since been refuted as poor representations of actual data (Brugger, 2008; Mahoney & Chan, 2003; McHugh, 2000). This leaves a gap with no widely accepted benchmark data in the intrusion detection research community. Consequently, the KDD Cup '99 test set is often still relied upon to provide test data and comparisons between detection methods. This research presented a testing approach that was consistent with the test approaches published for prior payload anomaly detection systems (Ariu et al., 2011; Perdisci et al.,

2009). Further, it made use of the same attack data as was used in testing those systems to provide a representative comparison between systems.

Assumptions, Limitations, and Delimitations

A limitation of this study is that the testing was not performed using real-time data. It instead used offline data. This is part of a natural evolution of the system and does not represent a permanent limitation of the system rather a limitation of this initial study. A possible extension of this work is to optimize and enhance the system for real-time data. When designing the system, the possibility of extension to a real-time detection system was kept in mind during the design process. It is expected that the modifications required to support real-time detection would be minor.

Another consideration for this system is the nature of attacks can be difficult to identify. This creates a limitation for the study in that traffic may appear to be normal and benign to the researcher performing classification but may actually be an attack. This can lead to false classifications in both training and testing results. Given this limitation, the traffic and attacks will be classified using the best available knowledge at the time of classification to determine if an attack is present in the traffic.

Summary

The focus of anomaly detection systems have evolved along with the evolution of attackers and their targets. Early attacks tended to be hidden in the headers of packets or detectable by broad statistical measures. Attackers have evolved to place their attacks in

the application layer and avoid detection by broad statistical measures. To combat this evolution, detectors have had to shift their focus to packet payloads. This evolution has led to a new subfield of anomaly detection called payload anomaly detection. This field is still relatively young but has benefited greatly from the advancements in the broader parent field of anomaly detection. Payload anomaly detection has evolved from its earlier focus on the statistical distributions of bytes to more complex statistical models such as Markov models. With each evolutionary step, the detection characteristics improved. This research continued the advancements of payload anomaly detection by increasing the detection performance of the system through the application of multiple layers of detection combined as well as considering response payloads to provide a more accurate result in detecting attacks.

Chapter 2

Review of the Literature

Overview

This chapter is organized into two sections. The first will be a review of major milestones in intrusion detection starting from the initial research in the field to recent milestones. The second section will be a review of literature directly relevant to this research.

History of Intrusion Detection

This section is a breakdown in chronological order of the history of intrusion detection from start to the current point in time.

Origin of Intrusion Detection Systems

The concept of intrusion detection was first introduced by Anderson (1980). Anderson proposed intrusion detection as a method to identify when a system was compromised. This focus of detection would later become known as host-based intrusion detection as opposed to what would be created later as network-based intrusion detection where the focus was on network traffic. To identify intrusions, the review of audit trails

could be used to identify behaviors that represented misuse of the system. The different forms of intrusions were enumerated as internal penetration, external penetration, and misfeasance (J. P. Anderson, 1980).

Internal Penetration was defined as the case when a user with legitimate access to internal resources attempts to gain access to additional resources they are not authorized to access. This attempt may occur through the usage of credentials belonging to other users. This illegal access could be identified through logs that would show a user account behaving in a manner inconsistent with its typical behavior.

External Penetration was defined as the case when an unauthorized user attempts to access protected resources. This could be a user accessing a system through direct physical access to the system or through networked connections. External penetrations could be detected by repeated failed access attempts to protected resources.

Misfeasance was defined as the case when a user attempts to circumvent protections in place on the system to gain additional access. This might be making use of incorrect file permissions or errors in the operating system that grant additional access. Misfeasance could be detected through audit logs that show users gaining access to files or systems that they should not have access to.

These intrusions could be statistically identified as deviating from the observed ordinary behavior through the review of audit logs. For example, an external penetration attempt might be identified by a series of failed login attempts. An internal penetration attempt or misfeasance might be identified by noting users that vary from their normal

activities. This concept, though visionary as the original publication of intrusion detection, was labor intensive and an inefficient method to detect intrusions.

Intrusion Detection Expert System (IDES)

The concept of intrusion detection was expanded upon, refined, turned into a model at SRI International, and then published by Denning (1987). At the time of Denning's publication, networks consisted of far fewer components and a smaller variety of components than they do in modern networks. The system created, the Intrusion Detection Expert System (IDES), used an expert system to evaluate log data and determine potential intrusions or misuse. Like Anderson's work, the focus on specific logs and audit data from servers is representative of what was later named host-based intrusion detection. Denning's system consisted of five primary elements. Those elements follow:

- subjects - the users of the system or processes initiated by the users
- objects - the various pieces of the system that the subjects interacted with
- profiles - characterizations of the interactions between subjects and objects
- anomaly records - records describing the abnormal behaviors not contained in the profile as detected by the system
- activity rules - actions taken by the system in reaction to some stimulus

Analysis of all these elements allowed the system to detect anomalies in behavior that may have represented an intrusion.

While Anderson's work described the concept in an ideological form, Denning's work created a system that implemented the concept. This concept created a model that was built upon for all future anomaly detection systems.

Haystack

The Haystack IDS was developed by Smaha (1988). Among other things, Haystack had innovative and effective data reduction techniques. Haystack was designed for use in the U.S. Air Force to help System Security Officers detect and investigate intrusions. The Haystack system was specifically designed for use in the Air Force but it was designed with the intention that its applicability could be extended to other systems. Being designed for the Air Force allows the system an advantage in detecting intrusions over other types of networks: Air Force systems have rigidly defined acceptable use scenarios. Other networks tend to be more open and the administrators have less ability to specify and enforce rules over their users. The goal in designing Haystack was to augment not replace the existing security tools and personnel. The system analyzes the raw audit trail of the target system, processes that audit trail into canonical events, and then generates a security incident report to present to the security officer. The security officer then investigates the incident and determines if it is a legitimate alert or false alarm. Examples of security incidents that are reported by the system are variations from

normal behavior such as spikes in usage or known bad behaviors such as printing password files.

To handle the large amounts of throughput necessary to be processed by the Haystack system, the designers implemented data reduction techniques. Specifically, the Haystack analysis system is significantly less powerful than the mainframe system it was intended to monitor. To maintain the throughput necessary to monitor audit logs in a reasonable timeframe as well as maintain a historical trail of audit logs, the system performs data reduction techniques on those logs. Rather than maintaining the full audit trail, the system extracts the significant components of the audit trail and preserves those for detecting intrusions. This feature extraction methodology made sense and increased efficiency. It was adopted in most intrusion detection systems that followed to reduce the storage necessary to record logs and provide throughput improvements.

Network Security Monitor

In 1990, the concept of intrusion detection was modified to look at networks creating the network-based intrusion detection model (Heberlein et al., 1990). This was done through a program called Network Security Monitor (NSM) that evaluated network traffic for intrusion attempts. This program differed from the IDES model in that the focus of this IDS was analyzing the traffic flowing across the network and identifying that portion of the traffic that was anomalous traffic with the assumption that anomalous traffic is worth investigating further. This represented a novel approach to intrusion detection. No other system was focusing on the network data but instead focused on

auditing the data local to the systems. The goal of a network-based IDS is to identify an intrusion based on network traffic alone. This type of IDS focuses either on readily apparent characteristics of the traffic or the way the traffic differed from the normally observed traffic rather than the end effects as reported by the target systems.

The NSM work was later expanded upon and further described by Mukherjee, Heberlein, & Levitt (1994) to better define the concept of network-based intrusion detection. NSM collected network traffic and classified it into four layers. The four layers approach allowed the system to characterize the traffic. The first layer was at the packet level and analyzed the individual packets. The second layer built upon the first to create a unidirectional network traffic stream. The third layer built upon the second layer to create a bidirectional traffic stream. The fourth layer built upon the third layer to identify the activities of each host in the network. The collected outputs of the third and fourth layers as well as a profile of the typical traffic of the network are then fed into an expert system. The profile of the typical traffic is created by monitoring the network activity over time.

The concept of network-based intrusion detection is also the method used in this research. However, the method of traffic inspection differs greatly between the two systems. NSM used an approach demonstrated in previous host-based systems to inspect network traffic. This method consists of constructing a model of normal traffic from host to host. This concept, while still valuable, is prone to false positives. Attempting to catalog all standard network traffic flows in a network leads to a challenge due to the changing nature of networks. New hosts can be created and new users can make use of existing hosts. When these changes to the network occur, the traffic profile will be different creating a false positive. The method presented in this research is network-

agnostic to avoid these false positives. It focuses on the contents of the packets traversing the network rather than the source and destination of those packets.

Mukherjee, Heberlein, & Levitt urged the use of benchmarking methods to test and compare IDS as well as the development of better and more accurate models. Both goals remain relevant to this dissertation research. While there have been attempts at creating universally accepted test methods such as the DARPA data set (Lippmann et al., 2000), there still remains no widely accepted method by which every IDS can be compared in an unbiased manner. Additionally, while modern IDS have become more accurate in their detections, there is no single system that is accepted as the best detector. Each of the intrusion detection systems have different strengths and focus areas requiring tradeoffs and the evolution of attacks has led to the degradation of older systems.

Distributed Intrusion Detection System (DIDS)

Distributed intrusion detection systems, systems that collect data from multiple points in a network to combine data from different perspectives rather than from a single independent monitoring point, were a natural next step after network-based IDS. A specific distributed IDS known as Distributed Intrusion Detection System (DIDS) was created to expand the ability to monitor systems and networks from a single monitored entity to multiple monitored entities controlled from a single location (Snapp et al., 1991). DIDS had the capability to remotely monitor heterogeneous computer systems then perform data reduction to collect all that data at a central point for data processing and correlation. The distributed nature of the system was limited to a local area network

and did not include hosts located on different network segments. An interesting element of this system is that it is capable of uniquely identifying network users using traffic analysis then assigning each identified user a Network User Identifier (NID). This NID is capable of following the user despite any attempts to circumvent the tracking such as multiple accounts or lateral movement in the network.

DIDS consisted of several components: host-based monitors on each host to be monitored, a network-based monitor, and a director to coordinate the components (Snapp et al., 1991). The host-based monitors each track the audit trails of their host systems to identify anomalous behaviors. The network-based monitor observes the network traffic to detect what systems connect to each other. Both of these monitor types report their collected data to the director. The director consists of an expert system that analyzes the information provided by the monitors. Further, it can request additional data from each of the monitors as needed.

DIDS is interesting in the fusion mechanics of its data feeds – both host-based and network-based monitoring. The systems that existed prior to DIDS focused on the paradigms independently while DIDS combines its view of both the network and the hosts. This model is not one commonly seen in intrusion detection systems even since the creation of DIDS. Even at the time of performing this research, a more common deployment scenario is independent deployment of host-based and network-based intrusion detection systems allowing for networks to benefit from the best of class in both networks. This data from both types of intrusion detection systems, however, is typically collected in a central log collection and correlation system external to the detection

systems. This external system allows for correlation of the data it receives from all sources and can perform a similar fusion as was introduced in DIDS.

Next-generation Intrusion Detection Expert System (NIDES)

Recent research has created a new category of IDS that blends attributes of both signature-based detection and anomaly-based intrusion detection called hybrid intrusion detection. As an editorial note, other definitions exist for hybrid intrusion detection that include the blending of mobile, embedded, or visual detection mechanisms (Alam & Vuong, 2007; Lauf, 2007; Peng, Feng, & Rozenblit, 2006). Hybrid systems tend to focus on improving the accuracy of detection by both increasing the detection rate and reducing the false positive rate using a combination of the signature-based and anomaly-based methods.

The first true hybrid system was the Next-generation Intrusion Detection Expert System (NIDES) (D. Anderson, Frivold, & Valdes, 1995), which represents an evolution of IDES. NIDES is a hybrid in that it combines both the misuse detection that existed in IDES with statistical analysis to perform anomaly detection. NIDES was designed to detect intrusions, as identified by the data differing from the historical norm, and intrusion scenarios, known system vulnerabilities, and other violations of a systems security policy.

The statistical algorithms included in NIDES allow it to exceed the capabilities of a typical misuse system by detecting unknown attacks. A future extension of the NIDES system suggested by Anderson, Frivold, and Valdes would be to create a hierarchical

structure to allow the system to better detect collaborative attacks. Additionally, NIDES uses audit records like IDES did rather than including any form of network data.

Graph-based Intrusion Detection System (GrIDS)

A Graph-based Intrusion Detection System (GrIDS) was created at the University of California at Davis in 1996 with an interesting and unique approach to intrusion detection (Staniford-Chen et al., 1996). GrIDS makes use of activity graphs to define the permitted behaviors in a network. The activity graphs are created automatically by the system after analysis of activity on the hosts and network traffic.

GrIDS operates in a hierarchical fashion using reduction techniques allowing the system to scale to large networks. This is an enhancement on previous detection systems in that it can collaboratively detect attacks from large networks. GrIDS is also capable of incorporating data received from external monitoring sources. Further, the system is controlled by a graphical interface that allows drag-and-drop capabilities to reconfigure the system - even in real-time. While presenting many novel and unique detection capabilities, this model was not built upon much by later research.

Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD)

A system that was developed by SRI International for monitoring large-scale networks is the Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD) (Porras & Neumann, 1997). EMERALD is a highly-distributed system that makes use of the object oriented paradigm to perform network surveillance, attack

isolation, and automated response. EMERALD also includes a recursive framework that can share the analysis results from the distributed monitors to provide a global detection and response capability that can counter attacks occurring across an entire network enterprise.

EMERALD consists of three monitor types that service three different levels of the network hierarchy. The types of analysis are enterprise-wide, domain-wide, and service analysis. The information feeds up and is correlated from the service level to the domain level and finally to the enterprise level. The service monitors review audit logs and probe the domain for intrusion data. The domain monitor handles the reports from the service monitors and alerts the system administrators of threats. The enterprise-wide monitors correlate the reports from the domain monitors to detect distributed attacks.

A novel component introduced in EMERALD is its communication capabilities. EMERALD can have system monitors in one domain communicate with another domain. Additionally, the system monitors can communicate with third party analysis tools. The communication occurs using a subscription-based scheme where both a push and pull mechanism exists. For example, a client monitor can subscribe to receive analysis from a server monitor in order to provide better correlation. This allowed the intrusion detection systems to provide broader detection coverage.

Hummingbird

Frincke, Tobin, McConnell, Marconi, and Polla (1998) wrote about the lack of collaboration in intrusion detection lamenting that law enforcement investigation work

has used these techniques for a long time but they have not been implemented in an automated manner previously. To meet this goal, they proposed a framework for cooperative intrusion detection. The framework does not account for multiple distinct IDS but rather the interaction guidelines for multiple systems using the same IDS, Hummingbird, which uses the previously proposed Hierarchical Management of Misuse Reports (HMMR) protocol (Frincke, Evans, & Aucutt, 1996). The HMMR protocol funnels information from various sources under different administrative control to a central reporting system. This allows correlation among the various alerts at a higher level but does not allow various administrators to modify or configure systems that they do not control.

The Hummingbird work highlights the benefits that can be gained by sharing data between systems especially when there exists dependencies between the two systems. In relationships where the sharing benefits both sides, symbiotic relationships, there is a strong incentive to share data between the systems. Hummingbird facilitates these symbiotic relationships.

To protect the data flowing between systems, the framework implements a few key measures. The first measure is access control, specifically a weakened Bell-LaPadula (Bell & LaPadula, 1973) model and Biba (Biba, 1977) model. Further, the system protects the sensitivity of the data exchanged through data filtering techniques. The techniques used are data reduction and data sanitization. While these techniques create additional processing on the source host, they do provide the additional benefit of reduced communication load.

The introduction of sharing and collaboration introduced new risks to the paradigm of intrusion detection (Frincke, 2000). There is a local decision of how much trust is placed in the remote sites that share data with the local site. Additionally, there is a concern of how much data can be safely shared to a remote site without divulging secrets. Multiple security policies must be managed to address the varying security profiles of each partner site.

A major advantage in collaboration lies in detecting what is termed as widespread attacks (Frincke, 2000). Widespread attacks require collaborative efforts across multiple domains to detect them. Those domains could represent a single site, network, or an enclave. The categories of widespread attacks are defined as simple attacks, repeated pattern attacks, and multipoint attacks. Simple attacks are attacks that come from a single attacker or point of entry. Repeated pattern attacks are repeated simple attacks that are generated from multiple attackers independently. A multipoint attack is an attack that consists of many component actions that make up an attack. Each of those component actions are separated and performed by separate attackers. Through the use of multiple collaborative detection points, a detection system could detect the more complex attacks such as a multipoint attack whereas a non-collaborative system would be limited to simple and selected repeated pattern attacks.

Coordinated Attack & Response Detection System (CARDS)

Another collaborative system that was intended for a large-scale distributed system is called Coordinated Attack & Response Detection System (CARDS) (Yang,

Ning, Wang, & Jajodia, 2000). This system was created to detect signature-based alerts in a large distributed network. The monitored network was directly connected and centrally managed by a single administrator. The data was centrally correlated and the alerts jointly compiled and managed. This allowed the systems to perform correlated attack detection with centralized response. Additionally, the signatures could be created uniformly through the entire system.

CARDS consists of three main components. The first component is the signature manager. The signature manager generates specific signatures from generic signatures, decomposes specific signatures into intrusion detection tasks, and distributes these tasks to the involved monitors. Monitors are responsible for the observation of the target systems. The monitors perform the intrusion detection tasks and cooperate with other monitors if attacks are distributed. Coordinating all of these components is the directory service. The directory service provides information to each of the signature managers and monitors. Aside from the directory service, the signature managers and monitors are not required to collaborate and can operate independently. These elements only collaborate when investigating a distributed attack. The communication between elements uses Extensible Markup Language (XML) (Bray, Paoli, Sperberg-McQueen, Maler, & Yergeau, 2008).

Prior to CARDS, other collaborative intrusion detection systems had been created. However, those systems all were created with the assumption that they were under the control of a single administrator and reside in a single internal network but have different perspectives of that same network. Typically, the collaborative systems assume similar network structures such as subnets and hosts. Without this common reference point, the

data from most collaborative intrusion detection systems are incompatible with other collaborative IDS even those of the same type.

Extending the CARDS system, Ning, Jajodia, and Wang (2001)researchers created what they term abstraction-based intrusion detection. This is based on the earlier work of the Adaptable Real-time Misuse Detection (ARMD) system (J.-L. Lin, Wang, & Jajodia, 1998). The CARDS method is essentially a hybrid intrusion detection technique that combines abstraction with signature-based methods in order to increase the detection rate by allowing the system to detect unknown variations on known attacks. That abstraction is important because the systems that are being observed and monitored are disparate systems with different operating systems. Additionally, abstraction can be used to filter irrelevant details so that systems can ignore irrelevant details and focus on the relevant traffic information that could contain the anomalies.

Abstraction is frequently used as a preparation process and the abstraction is performed ad hoc. An example of how this usage of abstraction could create a problem lies in detecting an attack such as the Mitnick attack (Northcutt, 2002). In this attack, the attacker first creates a SYN flood attack to disable the incoming Transmission Control Protocol (TCP) (Postel, 1981) port then connects to another host spoofing the flooded host's IP address and TCP port as the source IP and port. If a user were to create a signature based on that attack, the signature might look for a SYN flood then the subsequent TCP connection. While this signature would detect the attack described above, a new method to disable the TCP ports other than a SYN flood would render the signature ineffective. To combat this, a better alternative would be to create a generic signature that replaces the SYN flood with any attack that disables TCP ports. No

detection system prior to CARDS could detect this attack. The best alternative to creating a generic signature would be to write programs that support this generalization but those programs would require constant updating as new methods were identified and observed. The application of abstraction in this system led to two conclusions: abstraction is an error-prone process and there is not enough support for effective and efficient abstraction in current IDS. Using Ning, Jajodia, and Wang's extension of the ARMD system, the above signature generalizing the disabling of TCP ports would be possible.

ARMD falls short when applied to a distributed environment because it does not account for distribution and coordination mechanisms. The design of this system did not make use of the existing hierarchical approaches as they were considered inefficient. Should two distant IDS need to communicate on a distributed attack in a hierarchal system, their communications would need to be forwarded through multiple levels of hierarchy before they reach the other host. It would be more efficient if the IDS could communicate with each other directly. The ARMD model allows IDS to communicate directly when that communication is essential to detecting attacks. To this end, a framework extending ARMD was created by Ning, Jajodia, and Wang that performs distributed attack specification and event abstraction. Further, the model can detect distributed attacks using a decentralized approach by decomposing signatures into "detection tasks". The end result is a system that can detect variants on known attacks but not detect completely unknown attacks. The variants on known attacks must share essential features of an existing signature. Abstraction was shown to be a powerful technique for increasing the detection rate of an IDS yet must be used carefully as it can

also increase the false-positive rate. While the feasibility of abstraction-based detection was demonstrated, additional research is necessary to refine this work.

Snort-based Hybrid Intrusion Detection Systems

A common platform used for hybrid intrusion detection systems is the Snort signature-based IDS (Roesch, 1999) with features of anomaly detection added to it (Aydin, Zaim, & Ceylan, 2009; Gómez, Gil, Padilla, Baños, & Jiménez, 2009; Hwang, Cai, Chen, & Qin, 2007). These modifications to Snort have increased the detection rate and false positive rate of Snort against their test data under controlled circumstances.

Snort is a signature-based IDS (Roesch, 1999). It is an open-source tool that is provided for free and is also available in a commercial form. Snort is often used for research as its open-source nature makes it very accessible and easy to extend. Snort, in its default state, operates in the standard signature-based method. Signatures are defined and provided to the system. Then the system monitors for traffic that matches the pattern defined in a signature. If a match occurs, the system will then create an alert and present it to the operator. It is also possible for Snort to operate in a blocking or IPS mode.

Aydin, Zaim, and Ceylan (2009) improved Snort by combining packet header anomaly detection (PHAD) (Mahoney & Chan, 2001) and network traffic anomaly detection (NETAD) (Mahoney, 2003). These anomaly detection protocols are inserted as preprocessor components to Snort. In the testing against the hybrid IDS, the test results show that the aggregated detection totals of the Snort detection combined with both PHAD and NETAD improves over the detection rates of any of the components on their

own. The hybrid IDS does not appear to be collaborating among the detection protocols though but rather appears to be multiple engines running on the same data in sequence.

Gómez, Gil, Padilla, Baños, and Jiménez (2009) created a system called H-Snort, a hybridization of Snort with anomaly detection. They also chose to add their anomaly detection capabilities as a Snort preprocessor. A statistical anomaly detection method was chosen though a future goal is to improve their system later with a data mining preprocessor. Two observations about H-Snort were noted through testing. The first observation is that a longer training period lowers the number of false alarms. The second is that as the number of elements increases, the sensitivity of the detection mechanism decreases thereby reducing the total efficiency as it misses detecting attacks.

Hwang, Cai, Chen, and Qin (2007) modified Snort to create a feedback loop allowing the anomaly detection piece to generate signatures which are then fed into the Snort signature detection system. In this scheme, they hope to leverage the ability of the anomaly detection to identify unknown attacks while relying on the signature detection to have low false positives. The scheme they use to detect attacks is that of episodes. Episodes are a sequence of network events with rare episodes representing the anomalous activity which is possibly an attack. When using their hybrid methods, the detection capabilities significantly increased averaging a doubling of attacks identified versus Snort alone.

Fuzzy Logic Hybrid Intrusion Detection

Other research has combined fuzzy logic methods with intrusion detection to allow signature-based methods to have an anomaly-based characteristic to them. Fries (2008) combines fuzzy logic and genetic algorithms (GA) to produce strong results on both the detection rate and false positive rate. This system takes advantage of the genetic algorithm's capability to identify solutions through unsupervised learning.

Fries uses the Intrusion Detection Based on Genetic Clustering (IDBGC) algorithm developed by Liu, Chen, Liao, and Zhang (2004). IDBGC provides a two-stage approach that establishes clusters of network traffic features and then detects intruders by classifying traffic into one of the clusters. The first stage establishes a set of clusters using the nearest neighbor method. In the second stage, the clusters are then combined using a GA to obtain a near-optimal result. Additionally, the rules are optimized using a principal component analysis (PCA) proposed by Bankovic, et al. (2007). PCA significantly reduces the number of features needed for rules by extracting the subset of features that preserves the most significant information. This is accomplished by identifying a few orthogonal linear combinations of the original variables with the largest variance. To improve the rules' detection capabilities, fuzzy logic is applied. This is done through the use of trapezoidal fuzzy sets. The test results show that the application of fuzzy logic to traditional GA intrusion detection methods has increased the detection capabilities.

Another method makes use of the ensemble feature selection technique of fuzzy belief *k*-NN classification algorithms to create a hybrid IDS (Chou & Chou, 2009). This ensemble method yields efficient results that perform better than individual feature

selection techniques. The hybrid model created uses an ensemble feature selecting classifier with four base classifiers. Each base classifier uses a subset of traffic features to derive an independent decision about the network traffic. The decision obtained from each classifier is combined to improve the detection capabilities of the system. The base classifier makes use of the fuzzy belief k-NN classification algorithm. The goal with this detection system was to improve detection of attackers gaining access to systems and further exploiting vulnerabilities of that system. To further improve the detection capabilities by reducing the false positive rate, a data mining technique, the C4.5 decision trees algorithm (Quinlan, 1993), created rules for normal behavior. Using the normal behavior rules as a filter, the ensemble classifier generated an output indicating if the behavior is anomalous.

A Collaborative Distributed Intrusion Detection System

Lin, Xiang, Pao, and Liu (2008) created an interesting distributed and collaborative method by which IDS in a large network, such as a single ISP, can share packet analysis data rather than replicating effort. To accomplish the notification and identification of previously analyzed packets, the system modifies the packets at the border routers to mark the packet as analyzed. It does so by using the Differentiated Services Code Point (DSCP) field, an IP header field typically used for Quality of Service (QOS) purposes. This field is initially populated with a value of zero. After an internal router analyzes the packet, that field is then set to one. This method of checking the packet only once internal to an autonomous system (AS) assumes that the routers and IDS inside a network can share signatures internal to that network. This methodology was

not done to increase the detection capabilities of the IDS but rather to improve the detection throughput throughout the network. The simulation of this technique indicated that an overall increase in throughput was observed.

A Correlation Extension to CounterStorm-1 (CS-1)

Cullingford (2009) describes a scheme by which IDS can work together within a single distributed network by sharing alerts but does not describe a scenario with disparate networks. The system is implemented by modifying an existing commercial IDS called CounterStorm-1 (CS-1) (Trusted Computer Solutions, 2001). Those modifications shared alerts between the different networks to correlate the various alerts. The modification introduced to the CS-1 system was extending the correlation idea to contain information from disparate engines possibly running on disparate networks. To this end, the system implemented a new anomaly detection algorithm, Rogue Ports (RP), in CS-1 to share data between CounterStorm's disparate internal detection engines, RP and Statistical Payload Anomaly Detection (PAYL). The two engines shared data at the Command Center level, the central control mechanism for the CS-1 system, though sharing between multiple CS-1 installations was also possible. Testing with CS-1 and the RP and PAYL engines showed that collaboration and cross-correlation among different engines was a powerful technique to reduce false positives. These correlation techniques were found to be effective and improve the CS-1 detection system so further efforts were expected to productize this capability into the system.

Intrusion Detection Interoperability

Multiple frameworks for creating intrusion detection interoperability have been constructed and shared. Those frameworks have all focused on creating a collaborative network of disparate intrusion detection sensors. This would be beneficial in the scenario of a business evolving in the choice of their typical IDS as time went on leading to multiple different IDS existing at the same time. The Common Intrusion Detection Framework (CIDF) is the earliest example (Kahn, Porras, Staniford-Chen, & Tung, 1998; Staniford-Chen, Tung, & Schnackenberg, 1998). As far as status on this work, Staniford-Chen most recently publicly posted that this work was never intended to create a standard and is currently dormant (Staniford-Chen, 2008).

Subsequent to CIDF, an Internet Engineering Task Force (IETF) working group, Intrusion Detection Exchange Format Working Group (IDWG), was created. This working group's mission was "to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems, and to management systems which may need to interact with them" (Erlinger & Staniford-Chen, 2002). This working group appears to be idle since 2002 with no further action taken. The results of the working group have been published as IETF Request For Comments (RFC) 4765 (Debar, Curry, & Feinstein, 2007), 4766 (Wood & Erlinger, 2007), and 4767 (Feinstein & Matthews, 2007).

Reviewing the documentation that is still available from both the CIDF and IETF intrusion detection working group, it is apparent that the scope of the attempted solution is very broad. The contributors to the proposed standards had attempted to create a method by which all IDS could work in a concerted manner while still processing data in

a distinct manner. Their specifications allowed the systems to command other systems and specify the actions and reactions that should occur. This goal would require the participation of all IDS creators, both commercial and academic, to ensure that the requirements of the standard were maintained and interoperability of the IDS was available. The two bodies achieved significant milestones in defining the IDS messaging scheme but never took the task to completion.

Areas of Research Directly Relevant to This Research

This section focuses on research that provides the foundation for the concepts and methodologies that will be employed in this research. First, detection systems using payload anomaly detection to detect attacks are discussed. This is followed by an analysis of some specific techniques that were incorporated into this dissertation research.

Early Payload Analysis Systems

In a system proposed by Kruegel, Toth, and Kirda (2002), a detector can make use of application layer protocols to identify anomalous traffic. By monitoring protocols such as HTTP, DNS, SMTP, and FTP, that covers a large portion of all anonymous traffic available on the internet. The prototype system specifically modeled traffic from the DNS and HTTP protocols. In the evaluation of this approach, one of the factors considered was the payload distribution. It was hypothesized that the distribution of a payload is not evenly distributed and each protocol has a distinct distribution that can be evaluated against. This approach was shown to have limited success.

In the system LERAD, anomaly detection was extended to the payload (Vargiya & Chan, 2003). These researchers discussed the rationale as to why it is more reliable to evaluate payloads without making use of explicit knowledge of the protocol specifications. A primary motivation for this lays in the challenge of hard-coding the specification for each protocol. Further, the nature of protocols and their constant change means that this specification must be maintained and regularly updated. The usage of four distinct algorithms on marking boundaries within the payload was evaluated in this work. Within each pair of boundaries lies a token that can be used to model the behavior of the payload. It is these tokens that are analyzed for anomalous behavior.

These two early payload analysis systems set the foundation for later payload analysis systems. Prior to this point, as discussed in the previous subsection, most network-based intrusion detection was focused on headers rather than the application layer content or payloads. The focus on payload analysis represents a shift in detection strategies.

Payload Analysis

The PAYL algorithm created an intrusion detection system capable of detecting 0-day worms and creating signatures based on those worms for detection elsewhere (Parekh et al., 2006; Wang et al., 2005; Wang et al., 2006). Their system detects worms through anomalous behavior, namely the propensity for worms to repeat uncommon behaviors on certain ports in a nature that is unusual for the network being monitored.

Further, worms tend to have the same traffic repeated on both ingress and egress. Once an attack is identified, the algorithm creates a static signature for detection of this attack.

PAYL has a mechanism used to share attacks without oversharing and violating the privacy of the sharing parties. To do so, the payloads are compared between different sites to confirm the existence of an attack rather than purely sharing information about an attack. Should two sites share payloads that are sufficiently similar, the attacks are considered to be confirmed and the signature is more trusted and reliable. The methods discussed for the comparison of attacks are String Equality (SE), Longest Common Substring (LCS), Longest Common Subsequence (LCSeq), Manhattan Distance (MD), and LCS of Z-String (Zstr) (Wang et al., 2005).

String equality is a method by which two strings are compared for an exact match (Wang et al., 2005). As applied to the problem of worm detection, the strings compared would be the payload coming in and the payload going out. If the payloads are the same, then a worm is likely detected. The challenge with SE is that it is a very strict detection method. If a single byte differs between the two payloads or the packet is fragmented differently, then the match will not be found.

The Longest Common Substring method behaves in a similar manner as SE but rather than requiring the full string to match, there can be substrings matching (Wang et al., 2005). This makes the method less exact than SE but allows it to overcome the problem of fragmentation. With longer matching substrings comes higher confidence in the match. However, the challenge with this methodology is that it is computationally expensive to execute.

The Longest Common Subsequence method is similar to LCS with the difference that the subsequence does not have to be a contiguous block of characters as in the substring (Wang et al., 2005). The advantage of this approach over LCS is that it can detect a polymorphic worm. However, there is also the risk of increased false positives.

To prevent the sharing of private information and only allow sharing of potential worm information, the two sites compare the potential attacks to look for confirmation using a reduced form. This removes the first three metrics, SE, LCS, and LCSeq, from consideration as useful comparisons using those metrics because they require actual full content to do a comparison. The remaining methods are MD and Zstr.

The MD method requires a byte distribution of the payloads be exchanged between the two sites (Wang et al., 2005). The distributions are considered similar if they have a small MD. The actual MD is computed as the distance between two points measured by only taking paths consisting of right angles (Black, 2006). This distance measure originated from considering the grid-like streets of Manhattan. When computing a path between two points, since one cannot traverse city blocks diagonally, the distances between two points are equivalent across many distinct paths consisting of only right angle turns.

The Zstr method is actually an LCS against the Z-String of the two payloads (Wang et al., 2005). The Z-String is defined as the 1-gram frequency distribution ordered by rank. Essentially, this is a descending ordered frequency list of distinct bytes. The LCS is performed against the Z-string providing the longest substring in common

between z-strings. This in turn provides a similarity amongst the distributions of the two payloads.

To perform the payload analysis, PAYL extracts n-grams from the payloads. An n-gram is a sequence of *n* sequential bytes (Shannon, 1951). The n-grams are statistically analyzed to determine the distribution. This mechanism is language independent and requires no parsing, no interpretation, and no emulation of the content. In other words, this mechanism can function as the detection mechanism on its own. The original usage for n-grams was in language independent categorization of text, essentially determining the similarity of one block of text to another. The concept of n-grams continues to be a common theme in payload anomaly detection and will be seen again in the systems discussed later as well as in the system introduced in this dissertation research.

In PAYL, the n-gram is specifically extracted from the payload and the value of *n* is set to 1, essentially making 1-grams or single bytes. Then a basic statistical analysis is performed on the 1-grams storing their mean and variance each in a 256 element vector (256 being the full set of symbols used to represent the content of a payload). To make use of this data, PAYL trains on these values with normal traffic and establishes thresholds. If those thresholds are exceeded during detection, an alert is generated. The conclusion is that content-based alert correlation is a promising direction for intrusion detection. By sharing attack data between sites, stronger defense against attacks and better identification of attacks is possible.

Anagram

Anagram (Wang et al., 2006) was built by the same creators as PAYL. The goal of Anagram was extending the capabilities of payload anomaly detection to defeat a weakness in PAYL identified by Kolesnikov, Dagon, & Lee (2006). This weakness was a mimicry attack, a way by which attackers can cause their attack traffic to mimic the byte distributions of normal traffic. To defeat this attack, Anagram makes use of n-grams with values of *n* greater than 1.

To illustrate the value of using n-grams with values of *n* greater than 1, there is a discussion about the construction of buffer overflow attacks. Considering the nature of a buffer overflow attack, the buffer is typically overflowed with a single character repeated many times (J. P. Anderson, 1972). However, the choice of a single character repeated many times is one of convenience and not necessity. The characters repeated are simply needed to fill space and can be replaced with randomized characters if desired or even a construct such as a NOP sled, a sequence of no operation byte code commands. To avoid the propensity of looking for the buffer overflow mechanism, the detection mechanism used invariants in the packet payload that will always be present for the attack to work. These invariants are exploit code, a sequence of commands, or a URL. By identifying these invariants through the higher value of *n* in n-grams, the system will mark the packets as anomalous and create an alert.

A risk of using higher values of *n* in the n-grams is that the space needed grows exponentially with *n*. To reduce this large amount of space, the system makes use of a Bloom filter (Bloom, 1970) to store the features of the n-grams. A Bloom filter operates by having an array of bits initially with a value of 0 and a set of hash functions with

results that correspond to the elements of the array. For every element of the set to be recorded in the Bloom filter, it is run through each of the hash functions and the corresponding array bit is set to 1. This leads to a situation where multiple hash bits are set to 1 and there may be overlap. In cases of overlap, the bits remain set to 1. To determine if an element is part of the set, the element is run through the hash functions and the corresponding array bits are queried. If any bits are 0, the element is not present. If all bits are 1, the element may be present though there is a chance for false positives in this method. A visual example of a Bloom filter can be found in Figure 1. In this example, the Bloom filter is being checked for the presence of elements x, y, z, and w. The element z is not present in the original set because it hashes to a bit set to 0. However, both elements y and w hash to the same bit in the filter. This may be an indicator of a false positive or both elements may be present in the original set. The rate of false positives is dependent on the size of the array. A larger array is less likely to have false positives but utilizes more system memory while a smaller array reduces the memory footprint but is more likely to have false positives.

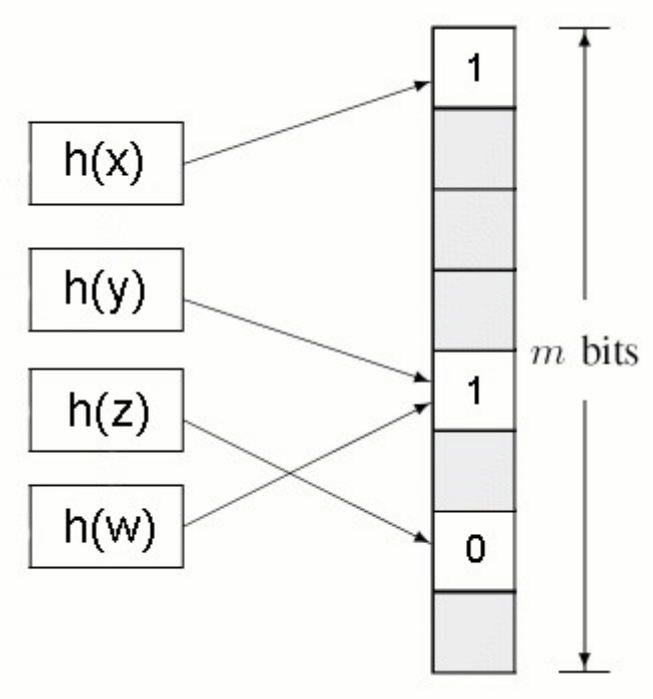

Figure 1. A sample Bloom Filter.

While testing Anagram, it became apparent that traffic tended to follow a distribution where a small amount of data was very common and a large amount of data was very infrequent as shown in Table 1. Their expectation was that a 0-day attack would have no occurrences prior to its detection by their system and that would indicate its presence. The reasoning behind this expectation is that the 0-day attack is exploiting some error condition in the application and this data has never been processed by the application prior to the introduction of the 0-day attack. This assumption creates risk outside of the controlled testing environment in that the 0-day attack could be pre-existing and part of the training data. This would lead to it being learned as normal traffic.

freq count	3-grams	5-grams	7-grams
>=5	58.05%	39.13%	32.53%
2 to 4	22.17%	28.22%	28.48%
1	19.78%	32.65%	38.99%

Table 1. Anagram traffic mix (Wang et al., 2006).

To test Anagram, network traffic data was collected from outside of the authors' department's web servers. This data serves as the normal traffic that is used to train the system. For the purpose of test data, they use worm samples either collected by the research team or obtained from third-party repositories.

Anagram not only detects anomalies but can perform a known attack detection filter as well. Wang et al. used a list of rules obtained from Snort as well as a collection of virus samples to construct a Bloom filter of n-grams contained in this malicious set. This creates a profile of "known bad" traffic that they term a "bad content model". To enhance their analysis, the system analyzed the bad data to identify known and expected n-grams and removes them from the set prior to being stored in the Bloom filter. This prevents the system from classifying a standard aspect of the protocol as an attack. For example, an exploit may be delivered through a standard HTTP command consisting of the text "HTTP GET". They would filter the "HTTP GET" command out of the bad content model while focusing on the parameters of the command.

The bad content model is used to enhance the accuracy of the training results. The system considers any packets that match at a level of 5% or higher to the bad content model to be potentially malicious and discard it from the training set. The system makes use of the bad content model again during detection. If the system detects an n-gram that

has never been seen during detection and that n-gram is present in the bad content model, that detection is weighted higher in considering it malicious.

Due to the nature by which the system identifies malicious traffic, by identifying suspicious n-grams, a side effect of the system is that it can generate signatures of the malicious traffic at no additional cost. The system can make use of the suspicious n-grams to generate a signature identifying those suspicious n-grams. To create these signatures, it makes use of a wildcard and create a packet specification that has each suspicious n-gram identified separated by the wildcard. Further, the Bloom filters could be shared between networks allowing for the detection mechanism to be shared with a reasonable expectation of privacy. This expectation of privacy is derived from the use of one way hashes and the set of all possible n-grams being so large. A collaborating site could make use of those Bloom filters to identify if suspicious n-grams have been detected elsewhere.

To combat the criticism against PAYL regarding its susceptibility to mimicry attacks, Anagram was tested against a polymorphic engine (Kolesnikov et al., 2006). The polymorphic technology they used was intended to have malicious traffic blend in with normal traffic. It accomplished this by mimicking normal traffic through padding the malicious strings with benign strings. Anagram performed well in detecting these attacks because the presence of the malicious strings meant that the n-grams of interest were still present despite the added noise of the benign strings. However, the bad content model approach did not benefit in this area of detection because of the modification to the n-grams contained in that model. Additionally, Anagram was tested against CLET-generated worms (Detristan, Ulenspiegel, Malcom, & Underduk, 2003). It was found that

the system performed well in detecting the CLET-generated worms because CLET is simply encrypted content rather than different content.

Multiple Classifier Payload-based Anomaly Detection (McPAD)

The McPAD system (Perdisci et al., 2009) refines the n-gram detection approach presented in PAYL and Anagram. The goal of anomaly detection using payload analysis remains the same. Building upon the PAYL and Anagram method of detecting n-grams, McPAD makes use of the n-grams differently. Rather than using a fixed size of 1 for *n* or varying sizes of *n* as in Anagram, McPAD uses a fixed value of *n* at 2. Again differing from the prior model, the bytes used to construct the n-gram are not necessarily consecutive. They are separated by some distance that is referred to as *v* which represents the distance between the two bytes. This construction is called a 2_v-gram. Using multiple values of *v*, a set of 2_v-grams is useful in that it can represent a richer feature space than the standard 2-gram by providing data on the structure of the payload beyond the adjacent bytes. Further, this method uses a fixed amount of space for each value of *v* of 256^2 with 2 representing the value of *n* in use. Using a higher value of *n* would increase the space used exponentially.

To reduce the dimensionality of the feature space, the system used a clustering algorithm. The inspiration for this algorithm was drawn from the paradigm of text classification in selecting an approach. To fit this paradigm, n-grams are considered to be the words and the payload is considered to be the full document. Typically, in the text classification paradigm, training only uses the words that are present. However, the

presence of an n-gram in the training set is not sufficient to exclude the n-grams not part of the training set. This is due to the expansiveness of the set of all n-grams and the variability of training data. In other words, it is very likely that any given training set will not contain all possible “normal” n-grams. If this were the case, there would be no need for complex classification algorithms and IDS could simply measure for the presence of certain n-grams.

To classify each payload as benign or anomalous, the system makes use of the varying parameter *v* and the dimensionality reduction clustering algorithm to have different representations of each payload in different feature spaces. Each of those representations has a model and classifier constructed. The set of all those classifiers is combined to form a Multiple Classifier System (MCS) as shown in Figure 2. In an MCS, each of the classifications has a vote on whether the analyzed payload is benign or anomalous (Xu, Krzyzak, & Suen, 1992). The votes are combined by using the average, product, minimum, and maximum probabilities. Once combined, the determination comes from comparing the result with a threshold. If the probability exceeds the threshold, then the packet is considered anomalous.

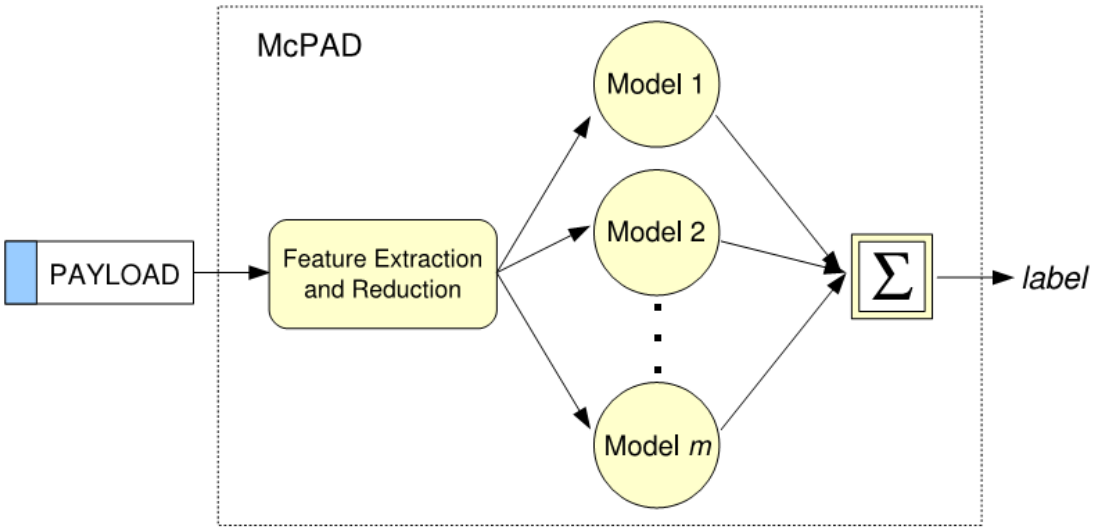

Figure 2. Overview of McPAD multiple classifier system (Perdisci et al., 2009).

The usage of an MCS for McPAD can be understood by decomposing it into its component parts. The concept of anomaly detection is a two-class classification problem where one class is well known, the normal traffic, and the other is poorly sampled, the anomalies. In this case, the approach best used is one of one-class classification. In one-class classification, the well-known group is used to identify those members not part of that group. This paradigm best fits the typical anomaly detection mechanism of taking unlabeled training data (data that does not have the attacks identified) and detecting anomalies on the network. The specific one-class classification technique in use in McPAD is the one-class Support Vector Machine (SVM) (Scholkopf et al., 2001). This technique was selected by Perdisci et al. because of empirical evidence about the strength of this technique in text classification problems. The SVM technique takes a set of data points and constructs a hyperplane intended to be used to separate the set of data points into two distinct sets.

Perdisci et al. provides a very thorough description of their experimental approach. A large benefit to this dissertation research is that they chose to share their test data. Their test data includes a full set of attacks and the variations of those attacks. They limited their testing to the HTTP protocol because of the difficulty they observed in obtaining comprehensive sets of attacks for other protocols. For training data, this experiment used the typical DARPA set, but also makes use of a specific data set consisting of traffic to and from their college's web server.

For the McPAD attack data, the data set began with non-polymorphic HTTP attacks made available by Ingham and Inoue (2007). They add to the attack data set an additional attack that exploits a vulnerability in the Windows Media Service (Perdisci, Gu, & Lee, 2006). In total, the set has 66 HTTP attacks. Of the attacks, 11 are shell-code attacks. Other attacks fall into other categories such as denial of service. Further, 8 of the 11 shell-code attacks were used to create 96 CLET polymorphic versions of those 8 chosen shell-code attacks. Of the shell-code attacks, 3 were chosen to be used in a polymorphic blending attack (PBA) (Fogla & Lee, 2006; Fogla, Sharif, Perdisci, Kolesnikov, & Lee, 2006). A tabular form of this data has been replicated as Table 2.This data set is fully available through a McPAD project web page (Perdisci, 2009). These attack sets were incorporated into this dissertation research's experiment discussed later.

Type	Attacks	Attack Packets
Generic	66	205
Shell-code	11	93
CLET	96	792
PBA	6,339	71,449
Total	6,512	72,539

Table 2. The attacks dataset characteristics of McPAD (Perdisci et al., 2009).

The results of testing showed that McPAD outperformed PAYL in detecting the shell-code attacks and polymorphic shell-code attacks. In general, McPAD was able to detect attacks better than PAYL as the false positive ratio decreased. For the PBA, McPAD was able to detect the attacks when they were spread over a low number of packets. For larger numbers of packets, neither detection system was capable of detecting the attacks. Computational results showed that the performance of PAYL was significantly better than McPAD. However, McPAD was a proof of concept written in Java that could be optimized for better performance.

Spectrogram

Spectrogram takes a slightly different approach to attack detection and payload analysis than previous payload anomaly detection systems (Song et al., 2009). Spectrogram focuses on defending a specific web server against attack. To do so, it trains a specific model against that server to learn the legitimate traffic types. However, Spectrogram was intended to improve upon PAYL's inability to scale to higher n-grams and to improve upon Anagram's inability to handle dynamic content as is the case in web

traffic. To achieve these goals, it uses Markov Chains (Markov, 1971) to handle the detection of attacks.

Spectrogram operates by focusing on the application layer, specifically the HTTP requests. Spectrogram is protocol-aware and thus can parse HTTP and break down the requests into their component pieces. Because Spectrogram focuses explicitly on HTTP traffic, Markov Chains were chosen as the analysis mechanism because of its ability to consider the position of the content in the HTTP traffic. Markov Chains are a set of states that are memoryless in the sense that they only depend on the current state and do not depend on a past state. For each potential state change, there is a probability associated with it. To enhance their detection capability, mixtures of Markov Chains are utilized which is a fundamentally similar approach to MCS as described in the McPAD system.

Song et al. tested Spectrogram against the PAYL and Anagram systems. In testing, it was observed that a larger value of *n* in the n-grams increases the detection capabilities of the system. Test results showed that the system performed better than the prior systems, detecting more attacks at lower false positive rates. However, even more notable is that it succeeded at detecting the semantic attacks, such as cross-site scripting, better than the other systems. This strength in detection is due to the system's HTTP-aware design. An interesting design consideration that can be used to improve results is that the system is capable of whitelisting, allowing certain known good traffic through without marking it as a possible attack.

Hidden Markov Model Payload Analysis (HMMPayl)

The Hidden Markov Model Payload Analysis (HMMPayl) (Ariu et al., 2011) builds off of the methods used in McPAD and takes it in a different direction. Where McPAD used SVM in the classification of packets, HMMPayl makes use of Hidden Markov Models (HMM) (L. E. Baum & Petrie, 1966; Rabiner, 1990).. Both models make use of an MCS to combine their classifiers to create a higher confidence in the classification results. Also of interest to this dissertation research is that the testing of HMMPayl is directly compared against McPAD even using the same data. The testing methodology of HMMPayl was adapted for use in this dissertation research.

The HMM is a model that defines states and their transitions (L. E. Baum & Petrie, 1966; Rabiner, 1990). As the name implies, some of those states are hidden or unobserved. From each state, there is a transition to the next state and each transition has an associated probability. The only information that factors into that probability of the next state is the current state. The probability in a HMM is only affected by the current state and not by a past state or a future state. An advantage of using an HMM in the application of anomaly detection is that they are robust against noise allowing the overall detector to be robust against noise.

As a tangent, a prior detection system called HMM-Web (Corona, Ariu, & Giacinto, 2009) was created by members of the same lab as HMMPayl. HMM-Web makes use of HMM and payload analysis with a goal of detecting web application attacks. This detector uses knowledge of the HTTP protocol to detect specific attacks such as Cross-site scripting (XSS) (Fogie, Grossman, Hansen, Rager, & Petkov, 2007) and SQL Injection (Litchfield, 2005). This detector lays some of the foundation for

HMMPayl but is not considered deeply in this research because of its concrete nature of detecting attacks through knowledge of the protocol specification rather than abstract investigation of the payload traffic. The end result is similar to a web application firewall (Cheswick, Bellovin, & Rubin, 2003) that is also capable of detecting some 0-day attacks.

Ariu et al. tested the HMMPayl system in a manner similar to the way Perdisci et al. tested the McPAD system. Ariu et al. used the same datasets as Perdisci et al. as well as an additional dataset that included XSS and SQL injection attacks. The results of the HMMPayl system were compared to the McPAD, HMM-Web, and Spectrogram systems. In testing against the McPAD system, HMMPayl proved to be a more accurate detector. When compared against Spectrogram, HMMPayl performed better at low false positive rates but was outperformed by Spectrogram at higher false positive rates. An example of the results of all four systems when tested against the XSS-SQL injection attack dataset can be found in Figure 3. However, the HMMPayl approach had with it a high computational cost. One way to overcome this was to make use of a low level language such as C (Kernighan & Ritchie, 1978) to implement the system rather than the scripting language chosen for the proof of concept. Additionally, employing the concept of random sampling

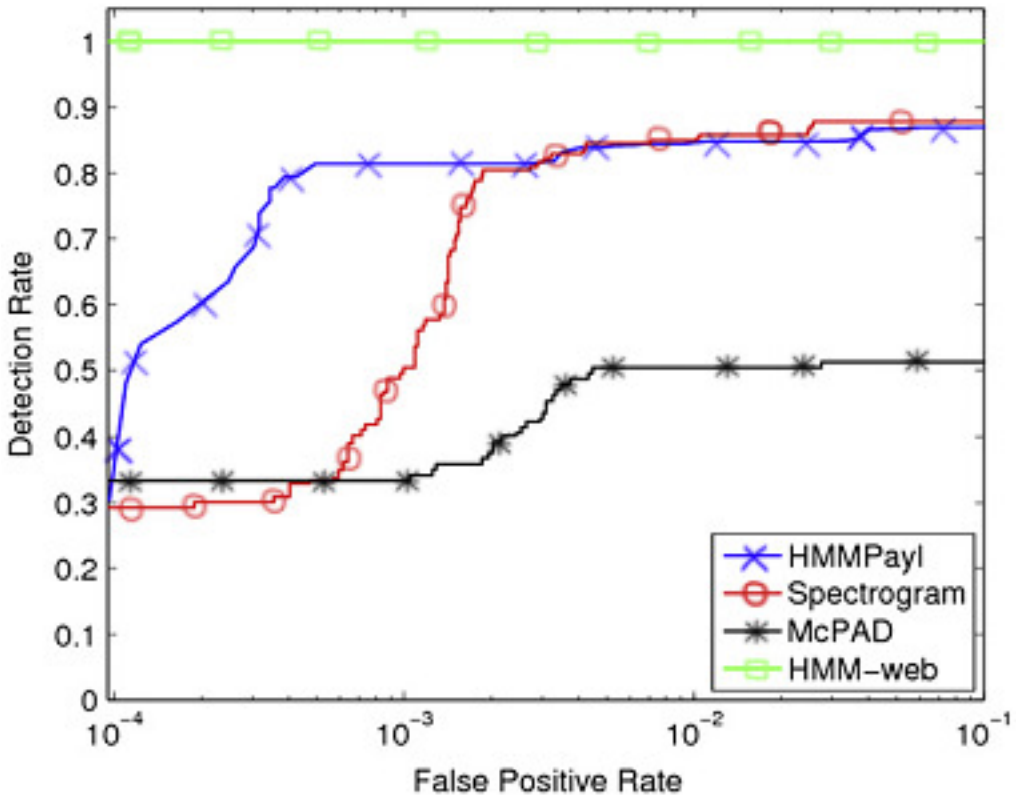

Figure 3. HMMPayl test results against the XSS-SQL Injection test set (Ariu et al., 2011).

could aid in reducing the computational cost. Another suggestion for improvement was considering the payload length in the analysis. Since payloads can vary in length, maintaining different models for different lengths may create a more accurate detector.

An important note considered in this dissertation research regarding HMMPayl is that it functions exclusively at the packet level and does not reconstruct HTTP sessions. This method of evaluation must be considered when evaluating the detection rate. An example might be that all benign requests are a single packet while all attacks span multiple packets. Compared against a detector that does reconstruct HTTP sessions but

has the same detection capabilities, the results would be skewed in favor of the packet-based detector over the session-based detector. This is due to the packet detector reporting multiple detections for the same session while the session detector only reports a single detection. Similarly, a packet detection mechanism that does not reconstruct sessions could be evaded through evasion attacks such as those presented by Ptacek and Newsham (1998).

Anomaly Detection of User Browsing Behaviors

User browsing behaviors across a web site as a whole tend to be fairly predictable. Using the hidden Markov model, it was shown that it is possible to model the browsing behavior and detect anomalies when the browsing behavior varies from the normal behavior (Yi & Shun-Zheng, 2009). The model was used for detecting distributed denial of service attacks utilizing the application layer they termed “App-DDoS attacks”.

A user’s browsing sequence is described by three elements: the HTTP request rate, the page viewing time, and the requested sequence of pages. The system assumes that an attacker can mimic a user’s request rate and page viewing time but not the sequence. The sequence of a user can be very dynamic and an accurate simulation of that behavior in scale is unlikely. Thus, the model focuses on modeling the sequences. By modeling typical users’ sequences, a Markov model can be constructed allowing for the behavior that does not fit the model to stand out as anomaly. The model yielded results of approximately 98% detection with a false negative rate of approximately 1%. This

approach to attack detection was used in this dissertation research to create a layer of the attack detection system.

Kolmogorov–Smirnov Test

The Kolmogorov-Smirnov test is a statistical test that can be used to determine if a sample matches a distribution (Kolmogorov, 1933; Smirnov, 1948). Additionally, a two-sample version of the test exists which can be used to determine if two samples are derived from the same distribution. The Kolmogorov-Smirnov test is a very useful test for comparing samples because of certain characteristics. One significant characteristic of the Kolmogorov-Smirnov test is that it makes no assumptions about a distribution whether the distribution is normal or not normal. However, if data can be assumed to be normal, there are tests that can provide more sensitive results. Because of the nature of the test, it compares more than just a mean and variance and actually compares the entire distribution by examining the location and shape of the empirical cumulative distribution functions of the two samples.

The Kolmogorov-Smirnov test has applications in intrusion detection previously (Caberera, Ravichandran, & Mehra, 2000; Estevez-Tapiador, Garcia-Teodoro, & Diaz-Verdejo, 2004). In Caberera et al. (2000), the test was used to determine if statistical values related to both normal and attack data have the same distribution. The test was used not as an intricate part of the detection mechanism but rather as a characteristic of the design of the detection system. In a system by Estevez-Tapiador et al. (2004), the Kolmogorov-Smirnov test was used to determine if the mean and standard deviation for

each symbol in HTTP request payloads was significant. Additionally, the Kolmogorov-Smirnov test was used to perform the final classification if a packet was normal or anomalous.

The usage of the Kolmogorov-Smirnov test in the system created in this dissertation research used it in a distinct and novel way to evaluate if test response payloads for a given URL had the same distribution as trained response payloads. This differs from the prior usage of the test in literature where it is a methodology evaluation tool and not a detection tool. This statistical test was specifically selected because of its non-parametric characteristics and its capability to perform a comparison of distributions, in this case against payloads.

Chapter 3

Methodology

Overview

The research method that was used to perform this research was experimental design. This research created an anomaly detection system named the Layered Hidden Markov Model Payload Anomaly Detection System (Layered) then tested it against network traffic consisting of both attacks and benign normal traffic. This methodology of creating a system and testing it is typical of how a new intrusion detection system is created and evaluated. Further, the methodology can be expanded to increase functionality or add additional detection capability to the system. To compare and improve the system created through this research, the newly created system was executed in parallel with existing peer-reviewed and accepted systems considered to be the state of the art using the same test data.

The intrusion detection system was created as a network-based system. It is a system created in software that resides on a single host machine. At the highest level, the system behaves like previous network-based intrusion detection systems in detecting attacks and creating alerts when that attack detection happens. The system follows the general detection mechanisms as described by the Spectrogram (Song et al., 2009) and

HMMPayl (Ariu et al., 2011) systems focusing on the payload content for anomaly detection.

System Design

The system consists of three detection layers with the results of each combining to classify traffic as an anomaly or normal. The first layer is a high level navigation model similar to that described in Yi and Shun-Zheng (2009). The second layer is a request payload analysis method similar to the one implemented in the HMMPayl system (Ariu et al., 2011). The third layer is an analysis that will consider both request and response traffic rather than just request traffic. The final piece is the combination layer that takes the output from all of the layers and yields a final determination as to whether the traffic being evaluated is anomalous or normal. All of these layers were constructed for this dissertation research and either represent new analysis methods or the enhancement of previously used analysis methods.

The general form for the analysis layers is the same for all three layers. Each layer has a training and testing procedure. For the training procedure, the algorithm will process through all the network flows provided from the training data and then perform analysis once the end of the training data has been reached. For the testing procedure, the algorithm will process each flow upon receipt. However, the Navigation Layer and the Request Layer both use the Hidden Markov Model (L. E. Baum & Petrie, 1966; Rabiner, 1990) for analysis while the Request-Response Layer uses the Kolmogorov–Smirnov test (Kolmogorov, 1933; Smirnov, 1948).

For each layer, an element of the packet is being analyzed. For the Navigation Layer, that element is the requested URL. For the Request Layer, it is the request

payload. For the Request-Response Layer, the elements are the requested URL and the associated response payload. The element under analysis in the layer is turned into a unique integer. For example, a URL of “index.html” has a value of 0 while “about.html” has a value of 1. For the Request Layer, that element is a byte within the payload which can be readily converted to an integer value. Repeated instances of an element will use the previously established value. These integer values are collected in a sequence across the flow. For the Request-Response Layer, since it is not utilizing the Hidden Markov Model, the elements being analyzed are preserved in their original form: a URL and a payload consisting of multiple bytes.

The sequences for the Navigation Layer and Request Layer are collected in a set of sequences for the training procedure and used to train a Hidden Markov Model using the Baum-Welch algorithm (L. Baum, Petrie, Soules, & Weiss, 1970). At the conclusion of the training procedure, the trained model is written to disk. The general form of a trained model is shown in Figure 4.

Descriptive line indicating the number of states

State number

Pi (π) value – The probability of this being the initial state

Aij value array – The probability of transitioning from state i to state j.

Probability distribution array – the type of distribution followed by the probability of an element belonging to this particular state

Figure 4. The general form of a trained Hidden Markov Model.

An actual example of a trained Hidden Markov Model is in Figure 5.

```
HMM with 5 state(s)

State 0
 Pi: 0.2
 Aij: 0.2 0.2 0.2 0.2 0.2
 Opdf: Integer distribution --- 0.001 0 0 0 0 0 0 0 0 0 0.018 0 0 0.018
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.052 0 0 0 0 0.003 0.001 0 0.004
0.004 0.004 0 0.005 0.013 0.045 0.022 0.023 0.03 0.024 0.017 0.018 0.016
0.007 0.007 0.009 0.009 0.017 0.015 0 0.012 0 0 0 0.019 0.003 0.014
0.008 0.012 0.006 0.005 0.006 0.008 0.002 0.004 0.008 0.008 0.009 0.004
0.008 0.003 0.007 0.01 0.014 0.003 0.001 0.002 0.001 0 0 0 0 0 0 0.009 0
0.03 0.004 0.033 0.011 0.051 0.008 0.014 0.011 0.03 0.001 0.004 0.014
0.016 0.025 0.029 0.021 0.002 0.031 0.022 0.035 0.012 0.004 0.015 0.002
0.001 0.004 0 0.002 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 5. An example of a trained Hidden Markov Model.

The Layered Hidden Markov Model Payload Anomaly Detection System tests for the probability that a given sequence was produced by the model. Consider a sample sequence of 1, 2, 3, 4, 5, 6, 7, 8. The general form of a set of sequences of n-grams where n equals 5 follows:

{(1, 2, 3, 4, 5), (2, 3, 4, 5, 6), (3, 4, 5, 6, 7), (4, 5, 6, 7, 8)}

Each element of the set yields a probability and the presented result is the average of the probabilities across the sequence. The probability of a set will be skewed lower if the individual sequences have probabilities of 0 indicating that the training model has never seen them. The actual n value used in the n-grams of the system is 10 for the Request Layer and 2 for the Navigation Layer. The value of 10 was shown in HMMPayl to be optimal with the best tradeoff between detection accuracy and memory utilization across a payload (Ariu et al., 2011). This was validated through empirical testing. The value of 2

for the Navigation Layer was chosen because it best captures the nature of a transition between web pages. In other words, the Navigation Layer represents the likelihood of a user browsing from one web page to the next given the trained likelihoods of web browsing. This layer is not analyzing a full sequence of web pages as the Yi and Shun-Zheng system did. The Request Layer is the likelihood of a flow containing a given sequence of 10 bytes in a particular order in a sliding window across an entire payload given the contents of payloads trained into the model. If a transition between two web pages does not exist in the training data, the probability of that transition will be 0. Similarly if a sequence of bytes does not exist in the training data, the probability of that sequence will be 0. However, since a probability of 0 could be a training problem, the averaging helps to eliminate any false positives caused by bad training data thus requiring multiple low probability events to bring the average down and cause an anomaly to be identified.

The probability of a given sequence coming from a model is derived from the information contained in the model combined with the sequence itself. Considering the sample model displayed previously, the Opdf value is the probability of a specific value belonging to that state. Recall that all values are discrete numbers uniquely assigned to some element of a flow so these probabilities are also discrete and not a continuous function. The Aij value is the probability of a transition from state i to state j. The Pi value (π) is the probability of starting in a given state. Because the set of values provided to the model is finite and discrete, it is safe to assume that the typical probability of a normal sequence tested against a model with a lower number of states will be larger than that of a normal sequence tested against a model with a larger number of states. Therefore

it is not inherently meaningful to compare the probability values of two different state size models against each other.

Navigation Layer

The first layer, the navigation analysis, focuses on analyzing the order in which web pages are requested. A depiction of the scope of this layer is in Figure 6. The URLs of web pages requested are fed into a Hidden Markov Model with the goal of constructing a model of the typical navigation patterns throughout the web site. The output is the likelihood of a user having browsed in that manner. If the likelihood is low, the indication is that this is an atypical browsing pattern and therefore anomalous. That feedback value is fed into the combination layer. This layer differs from Yi and Shun-Zheng's work in a few key ways.

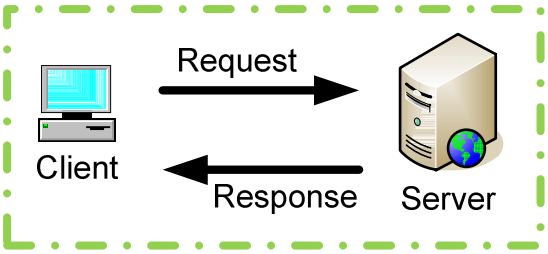

Figure 6. The high level navigation analysis layer looks at the order of URLs requested by the client.

For this layer's analysis, the URLs are stripped of any command parameters such as "?getdoc=25". This is done to generalize the URLs and aid in training and testing. For the example "?getdoc=25", assume the purpose of the page is to retrieve some sort of

document and each integer value provided retrieves a different document. In this case, the base URL is a valid URL and the differentiation of each parameter value is overly specific creating the likelihood of poor results whereas generalizing the URL by removing the parameter keeps with the spirit of the layer. Yi and Shun-Zheng did not explicitly state that they generalized the URL so it is assumed they used the full URL.

Further, this layer is only considering primary web pages such as HTML pages or ASP pages and not considering child objects such as images or scripts. This decision was made to eliminate false positives created by the various caching schemes in use such as proxies, browser-based caching, web accelerators, and content distributors. Depending on the placement of the sensor, the sensor may see all of the child object requests, none of the child object requests, or some fraction in between and all of those scenarios are potentially legitimate because of caching. However, the sensor will always see the primary web pages particularly in the case of a new request that would lead to an anomaly. Yi and Shun-Zheng considered all elements of web pages and not just the primary pages in their analysis.

Request Layer

The second layer, the request payload analysis layer, focuses on analyzing the contents of the payload in the requests. This layer follows the HMMPayl model and feeds the payloads into a Hidden Markov Model by breaking them up into segments derived from a sliding window. However, the key difference between the Request Layer and HMMPayl is that the payloads used in the Request Layer are combined across TCP payloads while HMMPayl used each individual packet's payload. A depiction of the

scope of this layer is in Figure 7. The size of the sliding window is a configurable parameter. Parameter values determined empirically through the HMMPayl research were used as a starting point for this development (Ariu et al., 2011). The output of this layer is the likelihood that the payload could have come from the constructed model. That value is fed into the combination layer.

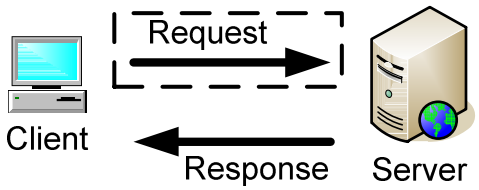

Figure 7. The request analysis layer looks at the payloads of the request.

Request-Response Layer

The third layer is an analysis of the request and response payloads. For this analysis, the input is the requested URL and the response payload in its entirety. A depiction of the scope of this layer is in Figure 8. The intention of this layer is to determine the likelihood that the response is a typical response and not the result of an attack such as a SQL Injection attack. For this layer, the only URLs considered are those that contain user-based input such as GET requests with parameters or POST requests. This layer also strips off any parameters for analysis. For example, a URL such as "/user_profile.php?user=john" would be considered for analysis but would be evaluated equivalent to "/user_profile.php?user=bob" because the parameters are stripped.

The Request-Response layer differs significantly from the other two layers because it does not rely on the Hidden Markov Model for its analysis. Thinking about it

abstractly, analyzing an isolated sequence of two elements is not an effective use of the Hidden Markov Model. The rough equivalent of doing so would be asking, “Given URL X, is the response payload Y’ equal to trained response payload Y?” The answer to this question is very unforgiving and cannot account for dynamic web pages or unique user accounts. A more thorough analysis was created by the use of the Kolmogorov–Smirnov test performing an analysis of the bytes in the trained response payload against a test response payload. Using this test is equivalent of asking “Given URL X, do the bytes of response payload Y’ have the same distribution as trained response payload Y?” In the case of an attack that would cause a web server to provide new, different, or unexpected response data, this data would create a different distribution of bytes in the response payload that would be detected through this test. The result of this layer is fed into the combination layer.

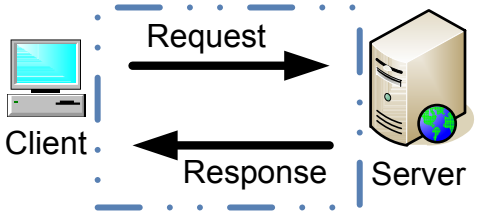

Figure 8. The request-response analysis layer looks at the payloads of both the request and response.

Combination Layer

The combination layer takes the results of all three detection layers and combines them to yield a final determination whether a packet is anomalous or normal. A depiction of the combination of all three layers is in Figure 9. The combination layer was chosen to

yield a determination of anomalous if any layer identified the given packet as anomalous. This provides the maximum sensitivity for the detection system and fits logically because the three layers do not significantly overlap in detection focus.

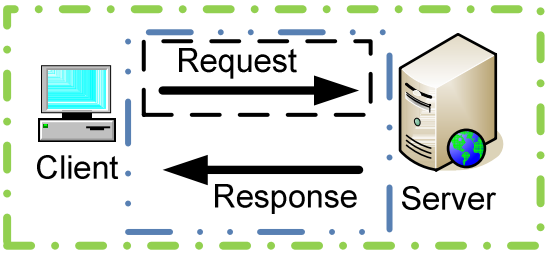

Figure 9. The combination of all three layers yields the final determination whether traffic is anomalous or normal.

The Combination Layer's output is a single determination of normal or anomalous. To achieve this determination, the system must interpret the inputs from each of the three analysis layers, the system requires thresholds for the analysis layers. For the Request Layer, the system performs a validation with normal labeled data to determine the threshold probability required to achieve a false positive rate specified by the user. For the Navigation Layer, the analysis is binary so a specific threshold is not necessary. For the Request-Response Layer, the threshold was determined empirically through analysis of data.

Software Design

The Layered Hidden Markov Model Payload Anomaly Detection System was created in software using the Java programming language (Gosling, Joy, Steele, & Bracha, 2005). Java was chosen from the perspective of simplicity and portability

allowing it to be easily used on different hardware platforms. The system created is not dependent on the underlying hardware with the stipulations that the hardware is capable of performing the processing requirements of the line speed parsing of the selected network traffic and capable of receiving data into the system at a modern line speed's data rate. No special computer hardware was needed and all development and testing occurred using standard retail components.

A significant component of the system is that of network traffic processing. The network traffic processing component was handled by the standard libpcap traffic capture library (Jacobson, Leres, & McCanne, 1994) as translated to Java to become Jnetpcap (Bednarczyk, 2012). This library allows for both reading and parsing of live network traffic from an Ethernet interface as well as offline traffic stored in the pcap format as well as standard network interfaces. The system is capable of taking network attack traffic from any source device or system so long as it is in the standard pcap format. The pcap format is a frequently used standard for traffic capture data, therefore it allows for greater flexibility and sourcing of attack traffic among a wide variety of capture sources.

For performing the analysis, the system uses the Jahmm software library (François, 2006). Jahmm is a Java Hidden Markov Model library implemented following the theory of Hidden Markov Models. Jahmm was designed with the goal of ease of use and code readability making it very useful for research purposes. For the Kolmogorov–Smirnov test, the Java Statistical Classes were used (Bertie, 2004). This library includes a wide variety of statistical methods.

The system design is clearly delineated into four layers making it a natural candidate for execution on a quad-core system. The implementation took that into account to ensure that the system was efficient and timely. The system was designed to avoid deadlocks and allow for parallelization of the multiple layers of analysis. However, due to memory limitations in the test system, large training sets were executed through the analysis layers training routines in series rather than in parallel.

Experiment

The test approach used in this research was similar to that of previous research of payload anomaly detection systems, specifically following the methodology laid out for HMMPayl. The experiment made use of two datasets consisting of normal baseline data and a third that was custom generated using WebGoat (OWASP, 2012). The first was the publicly available DARPA 1999 dataset. Despite the criticisms lodged against this dataset, it is a useful evaluation tool because it represents a commonality amongst the testing of most intrusion detection systems since it was released. The second dataset was captured from a Fortune 500 company. This dataset was captured through a network tap located outside of the firewalls and in line with the primary web server. Due to privacy reasons, the company has requested no attribution of the dataset. The third dataset was constructed using the WebGoat server.

To test the detection capabilities of the system, multiple attack datasets were used. These attack datasets are targeting the specific detection capabilities of the different layers. The Navigation Layer specific set of attack traffic was constructed by manually creating navigation-based attacks and capturing the traffic. The Request Layer specific

set of attack traffic was sourced from the data provided by the McPAD project team (Perdisci, 2009). Their freely available attack dataset consists of multiple attacks as captured and the variants of those attacks. The base attacks in the McPAD dataset were sourced from a comprehensive study that constructed and tested various different attack detection mechanisms proposed in literature using a consistent data set of attacks (Ingham & Inoue, 2007). The Request-Response Layer specific set of attack traffic was constructed by manually querying web servers with a variety of inputs and capturing the traffic.

To construct a comprehensive attack test set that would test all three layers of analysis in a natural browsing session, particularly the Navigation and Request-Response Layers, WebGoat was used to create a new attack data set. WebGoat is an intentionally vulnerable web server used for learning web-based penetration testing. It has various lessons that allow the user to learn attack skills across the spectrum of web attacks. The WebGoat data set consists of a normal training set and an attack set. Both sets were constructed by navigating the WebGoat site following the same script. For the training set, legitimate inputs were provided. For the attack set, successful attack inputs were provided. For example, the “String SQL Injection” lesson had a successful legitimate input of “Erwin”. The successful attack input was “Erwin' OR '1'='1”. Both data sets are bidirectional filtered to consist only of the web application traffic. Table 3 presents a tabular view of the WebGoat data set. Attack packets were determined through manual analysis of the packet capture. Attacks in this data set consist of navigation-based attacks and input validation attacks (SQL injection and cross-site scripting).

	Training Set	Attack Set
Total Packets	2252	2644
Request Packets	1160	1322
Attack Packets	0	140

Table 3. The packet breakdown of the custom WebGoat dataset.

The training and test normal data were constructed from the Fortune 500 company data. This data is approximately 8GB per day of traffic. Because of the challenges in handling a data set of this size using standard tools, the data set was then split into chunks of 1,000,000 packets. Those chunks were then used for training or test data. When used for training data, the first 50,000 packets of a chunk were used to train the system (50,000 was selected as a number that allowed the Request Layer to fit within memory constraints of the test system). The test data used the first 500,000 packets of a different chunk. The experiment results showed that these sizes were satisfactory for the testing performed. The results of the analysis layers were combined into the combination layer for a determination of whether or not the session represented anomalous behavior or normal behavior.

The system was compared against the state of the art payload anomaly detection system, HMMPayl, which was shown to generally outperform all prior systems. Since HMMPayl is a request only system, the datasets required a request only copy be made. To ensure that testing was equivalent, the datasets of request only was reduced from the 50,000 training packets and 500,000 test packets mentioned above to be only the appropriate portion of those packets that were requests. This number tended to be approximately a third of the total packets though this number was always determined exactly for each dataset.

To evaluate the system, the principle of Receiver Operating Characteristic (ROC) was employed (Green & Swets, 1966). The ROC is a graphical display of the sensitivity of a receiver. For a binary classifier, it plots the true positive rate against the false positive rate as the discrimination threshold varies. The worst case classifier is indicated by a line from the 0, 0 point to the 1, 1 point and represents the random chance results of guessing. This line is presented as a dashed line in the ROC charts. To mirror the results presentation of HMMPayl, the ROC curve will not have a false positive rate range from 0 to 1, rather it will focus on the range from 0 to 0.1 or a false positive rate of 10%. This limits the presentation of results to those areas where the false positive rate is relatively low. In intrusion detection, the true positive is called the detection rate (DR) and the false positive rate is called the false alarm rate (FAR). The goal of intrusion detection is maximization of the DR and minimization of the FAR.

Resources

This research required the usage of a computer capable of executing the software developed for this research and processing that data at a rate that is close to real-time. The system used consisted of an Intel Core i5-2500k processor and 8GB of RAM. Because the availability of hardware was limited, it was often necessary to improve the efficiency of the algorithms running on the system. One limit often encountered was the size of RAM. The size of RAM available created a limit on the size of the training set that could be used in the system. However, the results of testing proved to be satisfactory despite this limit.

To complete the testing aspect of this research, valid network traffic was necessary to provide the basis for testing the detection capability of the system. This traffic must include both attacks and valid network data. The attacks were either sourced from the McPAD test data or created manually. Valid network data came from the Fortune 500 dataset described previously in the testing section and the DARPA 1999 dataset which provides a comparison against other intrusion detection methods.

Chapter 4

Results

Data Analysis

The analysis of the results looks at the individual layers of the layered system in turn. Then the analysis focuses on the combined results of the three layers to analyze the overall performance of the system and the synergy of the three layers. The following results were all taken directly from the layered system created in this research.

Navigation Layer

The Navigation Layer focuses on the analysis of the browsing patterns a user follows when browsing the web site. Web sites have links that allow users to traverse from one page to another. When an anomaly is identified by this layer, it is an indication that the path from one page to the next is unlikely according to the training data used to train the model.

Consider the following example constructed using actual data and model training with a sequence size of 2 using the system created through this dissertation research. Suppose a user visits the web site home page followed by viewing a company press release then visits the company about page. The probability derived from this model is 0.022 of this sequence being legitimate. While this is a low probability, compare this

against a user now attempting to find a directory traversal vulnerability (Holub, 1986) and is trying arbitrary variations on a URL such as "/../../passwords.html". Assuming good training data, these obvious attack URLs would not have been trained into the system during the training phase as they are not legitimate web pages that a user would visit through standard means. The user's sequence of URLs visited in this case is two non-existent web pages or potential directory traversal attacks. The probability of this sequence being legitimate is 0.0 representing that this sequence could never have been generated from the trained model.

Should this user desire to hide their actions, they could have repeatedly visited the home page by hitting refresh in their browser. Suppose the attacker performs refresh five times on the home page prior to attempting the directory traversal. The probability of simply refreshing the home page five times is 0.056 while adding in the directory traversal at the end only brings it down to 0.045. These seemingly illogical probabilities are a result of the navigation analysis taking place in sequences of 2. A model configured for larger sequences would identify the behavior of constantly reloading the home page as unlikely rather than taking each occurrence as independent. Suppose that the model is trained to a sequence length of 5. The probability of the five refreshes becomes 9.90E-4 while the probability of the full sequence including the two directory traversals at the end drops to 3.30E-4.

Considering the two previous probabilities, the ratio of 3 can be explained very neatly in that a sequence of 5 refreshes of the home page has a probability of 9.90E-4 while adding 2 instances of browsing to non-existent web pages increases the total of number of sequences of 5 in the total sequence of 7 to 3. In other words, the probability

of the first 5 URLs, URLs 0 through 5, being browsed is 9.90E-4 while the probability of URLs 1 through 6 (including the first directory traversal) and URLs 2 through 7 (including both directory traversals) are both 0. To average out the probability, the total probability is divided by 3 leading to a probability of 3.30E-4. Essentially, for every non-trained web page visited in a sequence, the probability is 0 which is averaged in with the overall probability.

Considering the probabilities discussed as a result of the Navigation Layer, it is apparent that this usage of the Hidden Markov Model is not sufficient for the purposes for anomaly detection. The probabilities are not sufficiently distinct between normal and anomaly and between different instances of sequences to make accurate determinations. However, with confidence that the training data encompasses all paths in which a user might legitimately traverse the web site, the model can be used in another manner which gives very accurate results. The system is configured to alert on all sequence probabilities of 0 which means those alerts are any attempts for a user to visit a web page that is not present in the training data. With incomplete training data, this analysis approach runs the risk of creating a very noisy alerting mechanism rife with false positives. With good training data, this layer is highly accurate and precise with its alerts. This layer is highly dependent on good coverage in the training data to be successful in detecting attacks.

Request Layer

The Request Layer focuses on the analysis of the requests made to the web server. Standard HTTP requests follow a few patterns based on the request method. However, an

attacker might introduce unusual or atypical symbols into the request in an attempt to create an attack on the server. This layer focuses on the request at the individual byte layer and an anomaly indicates that the sequence of bytes in the request is unlikely according to the training data used to train the model.

The behavior of the request layer follows a similar methodology as the Navigation Layer but on a larger scale as most payloads tend to be relatively very large as compared to URL navigation sequences. Consider the simple case of a buffer overflow attack where the request is padded by 100 instances of the character A. Using a sequence segment size of 10, this means that there will be 91 instances of 10 character segments consisting entirely of "AAAAAAAAAA". Assuming the padding sequence was never seen in the training data, the result would be 91 counts of a probability of 0. These results would be averaged with the probabilities of all the other segments of the request payload. This would significantly reduce the overall probability of the payload.

Using the approach discussed in the Navigation Layer of relying on good training data and alerting on any 0 probability sequences is unlikely to be successful simply because the state space of all sequences of bytes is so broad that not all valid sequences may be trained into the model. In other words, legitimate sequences of bytes may exist that are not indicators of attacks but were not trained into the model meaning alerting on that sequence alone would be a false positive.

Since the Request Layer is so comparable to the analysis performed in HMMPayl, a direct comparison against HMMPayl can be performed. The HMMPayl detector was graciously provided by the creators of the system for use in comparison testing. To perform this comparison, the testing methodology of HMMPayl was adopted, using the

same data derived from the DARPA ’99 dataset as was used in HMMPayl testing for both training and normal data. The attacks used in comparison testing were the same as used in the Generic and Shellcode attack sets. However, when comparing the two systems as is, it became obvious that a direct comparison of the results was not meaningful because of the different interpretations of packets. The HMMPayl system treats each packet as an individual element while the Request Layer in this system handles reassembled TCP payloads as an individual element. This caused a disparity between the results in that HMMPayl reported results for attacks or normal packets multiple times when the Request Layer only reported one. This disparity was remedied by normalizing the HMMPayl results to have only one result per TCP payload as identified by the Request Layer. This normalization tends to lower the apparent performance of the HMMPayl system as compared to the performance reported in literature. However, this is not a criticism of the system’s detection mechanism but rather normalization to a different scale.

Once normalized, the results between the two systems tend to be comparable with the primary distinction that the Request Layer has a better true positive rate at a slightly lower false positive rate than HMMPayl. For the generic attack set, at lower false positive rates, the Request Layer shows an 86% detection rate performance improvement. At the higher false positive rates, the results tend to become more equivalent and are only approximately a 4% detection rate performance improvement over HMMPayl. For the shellcode attack set, the results follow a similar trend with the lower false positive rates having a greater performance gain. At the higher false positive rates, the shellcode attack set data has an equal detection rate of 0.91 at a false positive rate of 0.002. The results of

the comparison testing can be seen for the dataset referred to as Generic Attacks in Figure 10 and for Shellcode Attacks in Figure 11. The convention of HMMPayl reporting is adopted with the Receiver Operating Characteristic graphs here in that the graph is presented with a logarithmic scale for the false positive rate and only for the range of 0.0001 to 0.1 with 0.1 representing an arbitrary point where false positives outweigh the usefulness of the system. The “Layered” data series references the Request Layer in this system while the “HMMPayl” data series references the HMMPayl results. For Figure 10, the chart shows the Layered system outperforming the HMMPayl system in the generic attack dataset with the difference in detection rate greater in the lower false positives range and the gap narrowing as the false positive range approaches 0.1. This performance gain can be attributed to the Request Layer recombining TCP streams into single payloads.

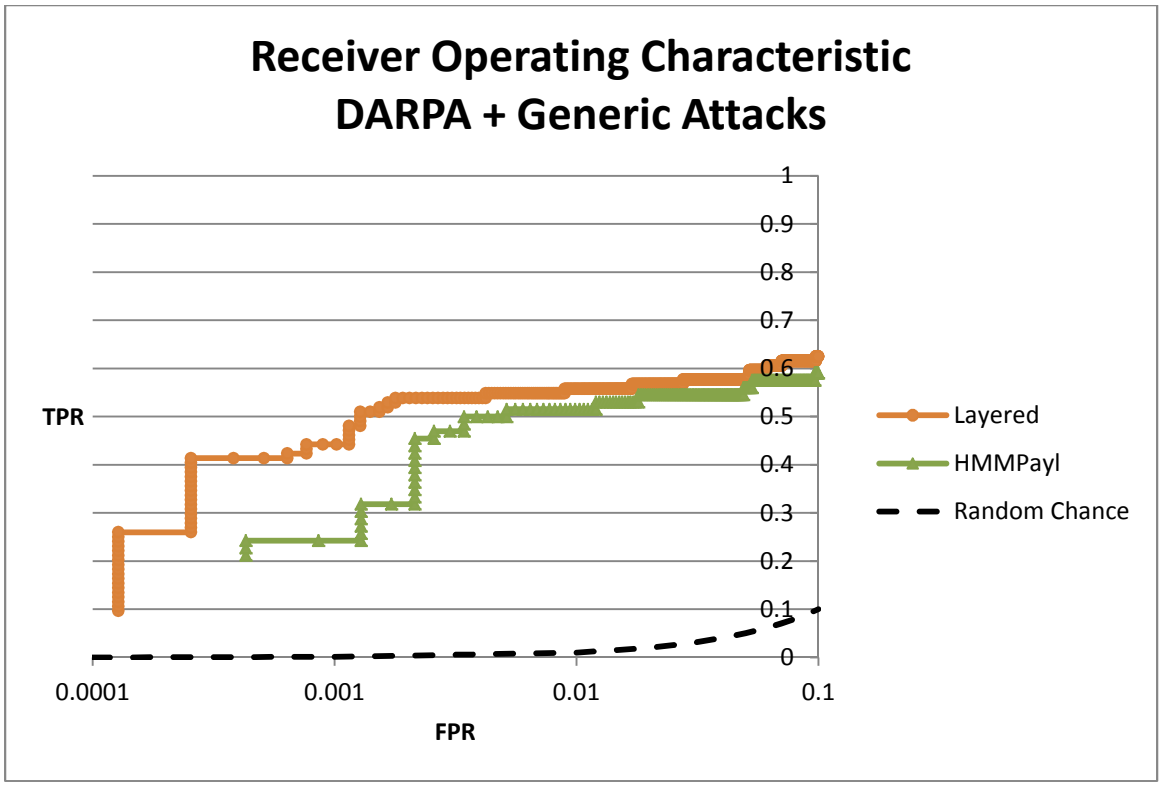

Figure 10. The ROC graph of the Layered system Request Layer as compared to HMMPayl for the Generic attacks dataset.

For Figure 11, the Layered system outperforms the HMMPayl system in the lower false positive range and the two systems have equivalent performance at false positive rates of approximately 0.002 and higher. This performance gain can also be attributed to the Request Layer recombining TCP streams into single payloads.

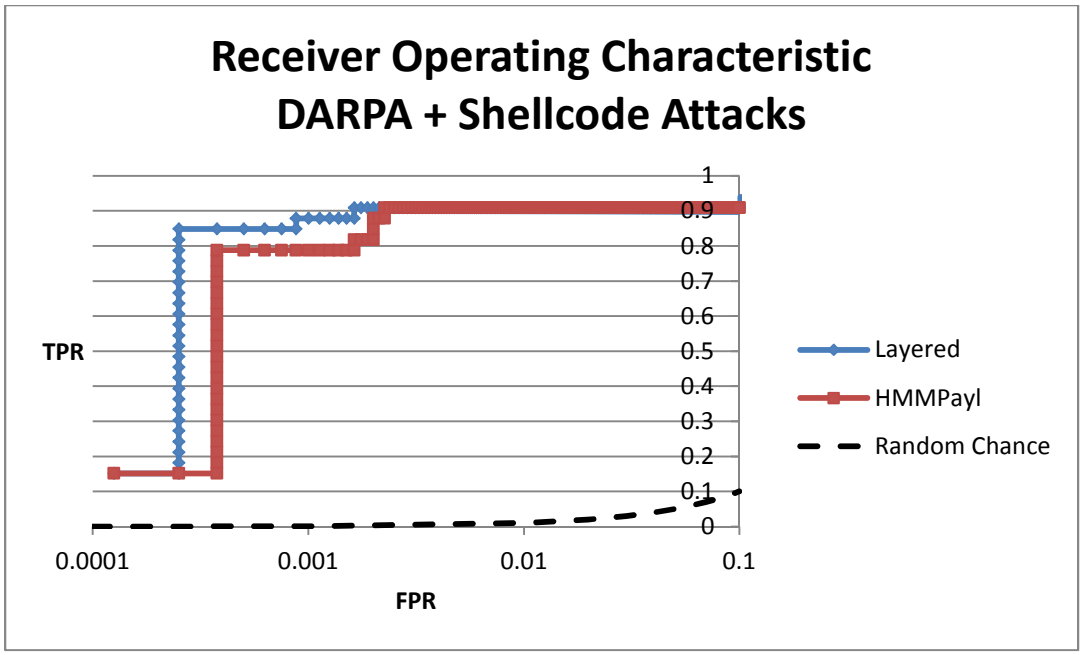

Figure 11. The ROC graph of the Layered system Request Layer as compared to HMMPayl for the Shellcode attacks dataset.

Request-Response Layer

The Request-Response Layer focuses on the individual transaction of a requested URL followed by a returned web page. In a normal, static web page case, requesting a URL will always return the same page. With a typical dynamic web page, results for a requested URL may differ. However, in the case of an attack, such as a SQL Injection attack in which an attacker enumerates the contents of a table in a database, the page returned may differ significantly from the normal pages returned. An anomaly in this layer indicates that the returned web page for a given URL differs significantly from the trained web page according to the training data used to train the model.

The Request-Response Layer behaves differently from the Navigation and Request layers. It does not utilize the Hidden Markov Model for its training and testing. Rather, it trains to learn a distribution of bytes for a payload based on the requested URL. For every tested response payload, a probability is generated that the tested payload's distribution of bytes is similar to the trained payload's distribution of bytes. Consider the case when the trained and test web page response for the URL are identical. In this case, the distributions are exactly identical. The probability returned from the test is 1.0. That result is saying that the two distributions are equal which is intuitive since the data sets are the same. Consider the opposite end of the spectrum where a script is queried to randomly select and return a CAPTCHA image. When two different particular CAPTCHA images' byte distributions are compared, the result is very unlikely with a probability of 5.32E-24 which means that the distributions have a low probability of similarity.

Using valid attacks against WebGoat (OWASP, 2012), a deliberately insecure J2EE web application, tests for Cross-Site Scripting and SQL Injection were created. These tests followed the lessons of "Stored XSS" and "String SQL Injection". For the Stored XSS lesson, a JavaScript alert is stored in a user profile and then viewed by another user. The viewing of the infected profile containing the alert by the second user was measured to have a probability of 2.33E-8 as compared to the original viewing of the benign profile without the attack present. The String SQL Injection lesson has a field in which a user can view credit card numbers by last name when providing a valid last name but can be injected to display the whole table of credit card numbers. The viewing of the entire table as opposed to just the valid individual has a probability of 0.02. Alternatively,

viewing the data for a valid last name but not the same last name as was trained into the system yields a probability of 0.94.

The next step for evaluating this layer was with blind SQL injection attacks. Blind SQL injection attacks are SQL injection attacks that do not provide direct results but rather provide a logical indicator if the SQL query is true or false. The web page is a simple login prompt that is vulnerable to SQL injection but does not provide the results of the query. The only indicator of success is that it allows a user to log in if the query evaluates to true. The trained case was a successful login. In the case of a successful login, the response page matches the trained case and the result is 1.0. In the case where a single tick mark (‘) is used to test for a potential SQL injection condition, the response is a database error message that significantly deviates from the standard login messages and yields a probability of 0.26. In the case of an unsuccessful login, a simple message stating there was an error logging in yields a probability of 0.33. In the case of a successful login through SQL injection, the probability is 0.99 which seems alarming because this layer failed to detect the attack. However, a blind SQL injection returns only success or failure, in this case by logging a user in or indicating an unsuccessful login. Therefore, the returned page for a successful SQL injection is virtually indistinguishable from a successful valid login and this result is expected. Further, when checking the results of the request layer, the requests containing SQL injection attempts are identified as the least probable requests meaning that the two layers are able to complement each other in this form of detection.

Figure 12 shows a ROC curve for the Request-Response Layer as tested on the WebGoat attack data set. This curve was used for the selection of a probability threshold

value that can be used to determine if the result is anomalous or normal. The selected true positive rate of 0.5 yielded a threshold probability value of 0.40. All results with similarity probabilities less than 0.40 are classified to be anomalous. This threshold choice was used in the overall combination layer for classification of the results of the Request-Response Layer. Based on the goals of detection, this threshold might be modified to a different value and might not necessarily be the proper threshold value for all detection scenarios.

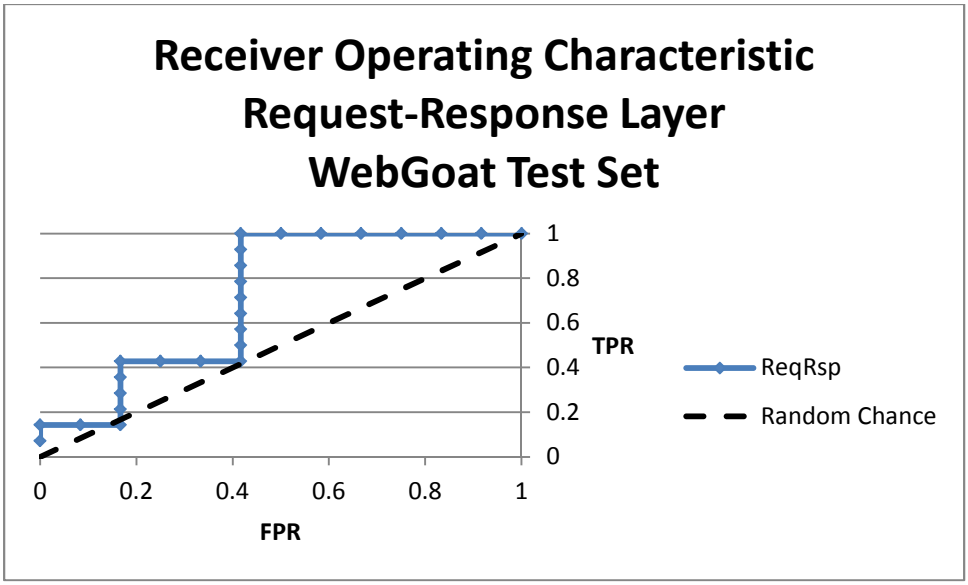

Figure 12. The results of testing the Request-Response Layer with the WebGoat test set.

Combination Layer

The Combination Layer represents the combination of the results of all three analysis layers into a single determination of whether the TCP stream is anomalous or not. For maximum sensitivity and because the layers have limited detection overlap, any

stream indicated as an anomaly by any layer is reported as an anomaly by the combination layer.

The most comprehensive experiment executed against the system created in this dissertation research was using packet captures generated by testing against WebGoat. The captures consist of a training set where normal queries are executed against the server and a test set consisting of normal navigation of the server interspersed with various SQL Injection attacks executed against the server and some navigation-based attacks. The SQL Injection attacks followed the scripts provided by the solutions pages of the server. In this dataset, many of the attacks provided limited deviation compared to the normal responses as determined both visually and through the request-response layer's analysis. However, for those attacks that had a large variation in the response, such as enumerating tables or executing commands on the web server and printing the output, the request-response layer did identify the attacks. Finally, the request layer was provided with significant deviation between the normal requests and the attack requests. The attack requests had uncharacteristic symbols representative of a SQL injection attack, such as apostrophes, semi-colons, and SQL query language. The results can be seen in Figure 13. For comparison, the results of HMMPayl against the same data sets are provided on the same graph. The graph shows that at a 0.008 false positive rate, the layered system has a detection rate of 0.91 while the HMMPayl system has a 0 detection rate until reaching a false positive rate of 0.12. For this reason, the false positive rate is reported in a linear scale on this graph. The results of HMMPayl are poor with this test set not as a reflection of the system's capabilities but rather a reflection of the test set chosen as a combination of the layers that extend beyond the detection mechanisms included in HMMPayl.

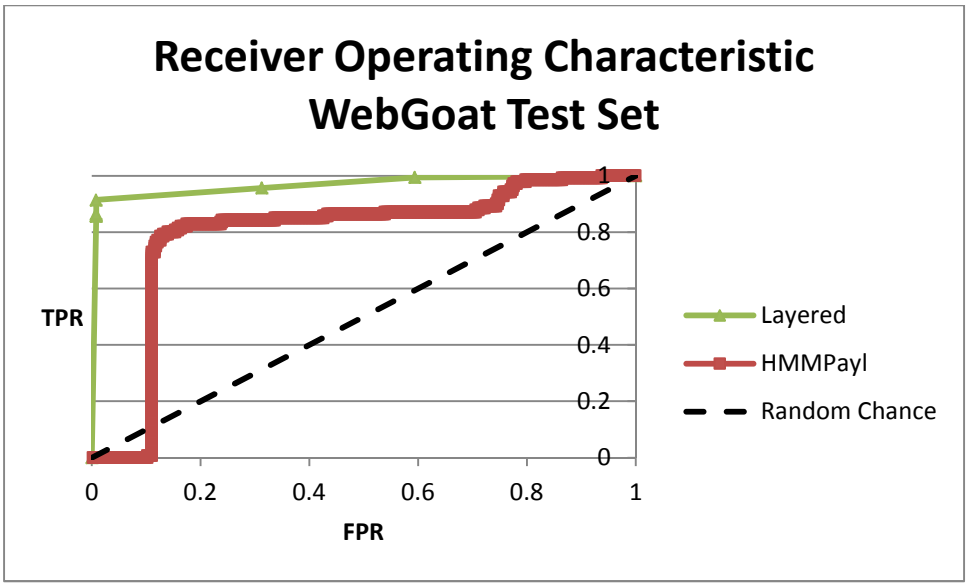

Figure 13. The results of testing against a WebGoat test set.

The next experiment used the Fortune 500 data in concert with the McPAD attack sets. In this experiment, the detection system was put up to a challenging task because it was heavily dependent on the detection mechanisms of only the Request Layer. The attacks of the McPAD attack sets only exist as requests and have no response components making the Request-Response Layer unable to contribute to the detection of attacks. Further, there were no navigation-based attacks present. Any Navigation-layer attacks identified were false positives due to bad training data. In this comparison, the layered system significantly outperformed the HMMPayl system as well. The results of this experiment can be seen in Figure 14 for the Generic attack set and Figure 15 for the Shellcode attack set. In Figure 14, for the generic attack set, the graph shows the Layered system having a detection rate of approximately 0.46 at a false positive rate of 0.015 while the HMMPayl system has a detection rate of 0.18 at the same false positive rate.

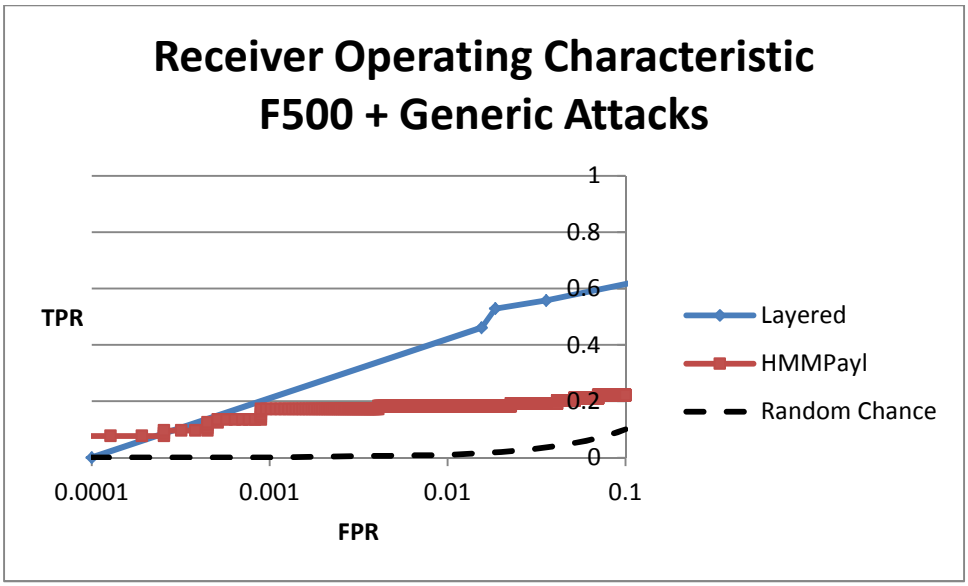

Figure 14. The results of testing against the McPAD Generic attack set and the Fortune 500 normal data set.

In Figure 15, for the shellcode attack set, the graph shows the Layered system having a detection rate of approximately 0.85 for a false positive rate of 0.015 while the HMMPayl system has a detection rate of 0.27 at the same false positive rate. For both attack sets, the Layered system shows higher detection results over HMMPayl.

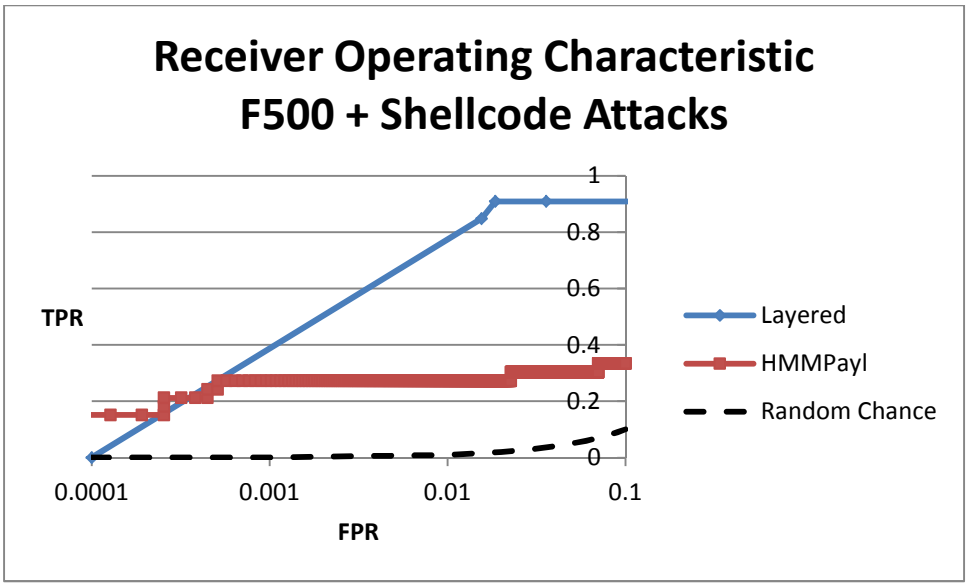

Figure 15. The results of testing against the McPAD Shellcode attack set and the Fortune 500 normal data set.

Findings

Navigation Layer

The results have shown that the Navigation Layer can successfully identify deviations from standard browsing patterns. When an attacker attempts to identify non-public portions of a web page or otherwise traverses a web page outside of the traditional links, this creates an anomaly that is reported to the combination layer. This layer is very sensitive to those attacks and with comprehensive training data can identify all attacks accurately. This layer has limited breadth though and cannot detect a subtle attack against a normal web page or otherwise identify a payload-based attack. However, this layer is not intended to detect those sorts of attacks.

Request Layer

The results have shown that the Request Layer can successfully identify deviations from standard request payloads by analyzing the individual bytes in the requests. Through this layer, attacks such as buffer overflow, shellcode attacks, cross-site scripting, and SQL injection attacks can be identified. When an attacker alters the contents of the request payload so that it no longer fits the profile of a normal request, this layer detects that attack. However, in the case of an attack that blends in to the normal byte patterns of request payloads, this layer will not detect the attack.

Request-Response Layer

The results have shown that the Request-Response Layer can successfully identify when the response payload for a given requested URL deviates significantly from the response payload that was trained. Through this layer, attacks such as SQL injection, cross-site scripting, and other parameter manipulation attacks can be detected. However, these attacks can only be detected when they cause a substantial deviation in the behavior of the web application and its response payload. For example, it was shown that a blind SQL injection, when successful, may not be detected by this layer but this could be compensated for by the Request Layer.

Combination Layer

The combination layer provides a combination of all three previous layers to effectively detect a large set of attacks. Working in isolation, the layers are each capable of detecting a subset of attacks. However, working in concert through the combination

layer, the system is capable of detecting broader attacks and even compensating for the weakness of layers toward certain attacks. As an example, blind SQL injection was referenced previously as an attack that is often not detected by the Request-Response Layer but often is detected by the Request Layer.

Summary of Results

The Layered Hidden Markov Model Payload Anomaly Detection System has been shown to be successful in detecting attacks and outperform the current state of the art detection system, HMMPayl, in the experiments performed. The individual layers were all tested with their results analyzed in turn. Then the combination of the layers was tested as well and demonstrated to be successful in detecting attacks.

Chapter 5

Conclusions, Implications, Recommendations, and Summary

Conclusions

Through the results reported in the prior chapter, it has been shown that the system created for this research has met the goals of this study. In the case of the request layer, where a parallel with detailed testing results exists in published literature, HMMPayl (Ariu et al., 2011), that layer was shown to always perform at least equivalently and often outperform existing systems. When combining the three layers into a single anomaly determination of a packet, the Layered system was shown to outperform the current state of the art intrusion detection system, HMMPayl, in detection performance.

Implications

This work has expanded the state of the art in payload anomaly detection, an area which is still very young in its study. Most significantly, this work has opened new doors as far as examining the response packets. Prior intrusion detection systems have typically focused on the request packets when looking for attacks and eschewed the response from the server. Through this dissertation research, the response from the server has been

shown to be a worthwhile area to point sensors at for the identification of attacks. While the detection of an attack in the response packet might be too late to save data if it is a destructive attack, it is still better to detect the attack than not detect it. Further, if the attack is some sort of exfiltration or information gathering activity, an attack detected in a response packet can still be blocked (assuming the system is an active system and properly placed in the network path) from going out of the network borders and thus hinder or possibly prevent the attacker's activity.

Another contribution through this work is in exploring the reassembly of TCP packets' payloads into a single stream. Prior payload analysis systems have been individual packet based in their analysis and have not performed the reassembly of TCP payloads. However, not combining payloads into a stream makes those systems susceptible to well-crafted evasion attacks (Ptacek & Newsham, 1998). The design principles of the Layered Hidden Markov Model Payload Anomaly Detection System were created with the stance that the analysis of packets should be performed with the same content as presented to the web server application.

Further, a layered system has been shown to improve detection capabilities in the payload anomaly detection field. Each layer has a focus on a different piece of the packet's payload and complements the other layers in detection capabilities. Through this complementary behavior, the different layers can work together to improve detection capabilities over any one layer independently. Additionally, a one size fits all approach to modeling is not the answer as demonstrated in the Layered system. Attempting to use one panacea for the solution to detecting all attacks would leave the Layered system crippled in the Request-Response layer.

Recommendations

Though the Layered system performs quite well in the experiments devised to test its performance capabilities, there are still opportunities to improve on its performance in future research. Some of the following recommendations will be specific enhancements to this detection mechanism while others will be more general to the anomaly detection field of research. This list of recommendations is not intended to be fully comprehensive or all-encompassing but rather a small slice of lessons learned from the development and testing of this system.

While the Layered system was deliberately built as a network-based detection system, a possible enhancement to this system would be to insert its functionality into a popular web application server and having it behave as a module in-line to the server rather than an external network-based sensor. This modification allows the intrusion detection system to see exactly what the server sees without the risk of evasion attack trickery or the challenges of implementing the TCP packet reassembly method to mimic the assembly performed by the server. Addtitionally, there could be distinct but subtle differences in how network devices and application servers handle the TCP packet reassembly (perhaps due to an attacker's wile). This could lead to a network-based detection mechanism incorrectly reassembling an attack and therefore not detecting the attack.

The layers selected in this study are by no means a comprehensive set of all potential layers. However, as shown by the combination layer, they do work well in harmony to identify different attacks. Other layers might be added to the system that would complement the existing layers or even overlap the existing layers in detection

capabilities to further reduce false positive rates in the system. One layer worth exploring in a future system would be focused on the time of the traffic. Consider that a simple denial of service attack might merely attempt to exhaust all available bandwidth or processing power on the server by repeatedly making rapid requests at a rate far faster than a human could click links or even mash the refresh button on their keyboard. In this case, the time between requests would be at computer speeds (order of milliseconds) rather than the typical human speeds (order of seconds).

A major challenge with the execution of this study was in the gathering of data for testing the system and comparing it against previous systems. Captured network data has a sensitivity to it that prevents researchers from sharing live network data. Fellow researchers were unable to share the captured data used to test their systems just as the captured data used to test this system cannot be shared. On the other hand, simulated or artificial network data is very difficult to make legitimate. The end result is that intrusion detection systems are difficult to test and particularly difficult to test in comparison with other systems. Further, it becomes challenging to compare systems when evaluation methods vary. Specifically, comparing a system that combines TCP payloads into a stream for analysis against systems that analyze individual TCP payloads requires normalization before a valid comparison can be made.

Evaluating the state of the art models, there are various enhancements to detection accuracy that exist though they typically include tradeoffs to gain that increased accuracy such as complexity, memory consumption, or performance. Some examples of those enhancements include using the Multiple Classifier System paradigm as in HMMPayl, deeply understanding the structure of the application layer requests as in Spectrogram,

the combination of the TCP payloads across packet boundaries as presented in this system, or the addition of multiple parallel layers of analysis as presented in this system. All of these enhancements have been shown to provide detection benefit. As future researchers consider the problem of anomaly detection, they should evaluate the various available enhancements to detection accuracy and consider their place in future systems. While these enhancements may not be necessary to prove the viability of a particular paradigm of detection, they might allow a system to better make the leap from laboratory experiments to real world detection usage.

A final recommendation is a word of caution to developers utilizing external libraries to implement functionality in their systems. Many of the external libraries encountered for potential inclusion in this system were found to be abandoned with no further development years ago. While their base implementation might be sound, there were challenges of edge case defects that created minor obstacles to identify. Identification of these defects can be challenging when having to debug someone else's code, particularly when they often do not provide source. Keep a decompiler such as JAD (Varaneckas, 2001) handy along with a debugger. Some of the libraries included in this system's code required minor fixes to ensure they worked as expected. On the other hand, a library that should be considered for any network-based system is the Jnetpcap (Bednarczyk, 2012) library which is regularly updated, actively supported, and very feature rich.

Summary

Attack detection has evolved to meet the parallel evolution of attacks in an arms race between the attackers and the defenders. Attack detection can take the form of signature-based detection where a specific signature of a known attack is written and traffic is compared against the signature. It can also take the form of anomaly detection where a detector learns the normal baseline of traffic and then creates alerts when traffic deviates significantly from that baseline. Attack detection in its earliest form involved system logs and characteristics. As attackers expanded to take advantage of the networking of computers, attack detection had to focus on detection of attacks in the network layer. Even within the focus on the network layer, detection has had an evolution of focus. Initial detection mechanisms focused on the header and evolved to look at areas like statistical aspects of the traffic. The most recent evolution is to look at the payload of network packets as the attackers have evolved their attacks to hide in the application layer.

This evolution has led to an area of study termed payload anomaly detection. This is an area that focuses on attacks at the application layer, often but not exclusively web or HTTP attacks. Payload anomaly detection is very concerned with the bytes of the payload and their specific ordering. A parallel can be drawn between the field of payload anomaly detection and the field of natural language processing. Many of the techniques used in payload anomaly detection are derived from earlier work in the field of natural language processing.

This research was performed with the goal of improving upon the current state of the art in payload anomaly detection. Payload anomaly detection has been shown

empirically to be very effective at detecting attacks making the margin for improvement very slim. To achieve this improvement, a system composed of multiple layers of analysis was proposed. The analysis layers all analyze the same data independently and then provide a result to a combination layer that combines the layers' results into a single determination of normal or anomalous traffic. The individual layers proposed for this system were a high level Navigation Layer, a Request Layer, and a Request-Response Layer. An important distinction between this system and prior systems is that the Layered system was constructed with the design principle to combine TCP payloads into the full stream ideally in the same manner as would be presented to the web server. Prior systems treated each TCP packet as an independent entity. This allows the system to be more accurate in its detection of attacks in the Request Layer. Without this mechanism, the system would not be effective in the Request-Response Layer.

The Navigation Layer focuses exclusively on the order of navigation of a web site. An ordinary web site can be mapped thoroughly through automated means or it can be mapped passively by monitoring the aggregate traffic of many users accessing the site. This mapping comprises the training data of the layer. Attacks are detected when a user attempts to browse to pages of the site that do not exist in the training data, such as a directory traversal attack or an automated mapping attempt to detect hidden pages like an administrator control panel. For the passive mapping method, this can result in false positives. For the automated mapping method, this should be comprehensive and not yield any false positives. This layer was shown to be very accurate when provided with all-inclusive training data. However, with limited training data, this layer presented false positives in its results.

The Request Layer focuses exclusively on request payloads. Request payloads in the terms of a web page are the manner in which different pages or components of a web site are retrieved for viewing on a local computer. Additionally, request payloads can contain data that a local computer is providing to the web site in the manner of POST requests. The best way to obtain this training data is through a large amount of actual request packets. These payloads are analyzed in the form of n-grams which break the bytes of the payload up into n-sized chunks then feed them into a Hidden Markov Model. The model learns the typical contents of payloads and their ordering to allow it to detect anomalous requests. Some examples of anomalous requests might be a SQL injection attack where the request includes various symbols and parameters that are not typically found in a request or a buffer overflow attack where the request includes many repeated characters that are not found in a typical request. The end result is a system that is capable of identifying various different application layer attacks that are carried in the request payload.

The Request-Response Layer focuses on a requested URL and a responded payload pair. Whenever a user requests a valid URL, the server responds with the contents of that page. The system tracks those pairs in training. As with the Navigation Layer, this information can be obtained through automated means or through passive means. However, this layer behaves differently than the Navigation Layer when faced with inadequate training data. If a requested URL has not been seen in the training data, this layer will not report an anomaly. If a requested URL has been seen in the training data, the test response will be compared against the trained response using the Kolmogorov-Smirnov test. This test determines if the distribution of the bytes in the test

response payload is equal to the distribution of the bytes in the trained payload. This layer was shown to be good at detecting deviating responses as the result of a SQL Injection attack or Cross-Site Scripting attack while allowing for normal deviations such as two different users providing valid credentials.

Finally, the three analysis layers provide their analysis results to the Combination Layer for the final determination of normal or anomaly. This layer collects the layers from the analysis layer and then makes a determination based on the values provided. If the results of any analysis layer indicate that a packet is anomalous, the determination is that the packet is an anomaly. If none of the analysis layers indicate that the packet is anomalous, then the packet is determined to be normal. This characterization behavior is the most sensitive and takes advantage of the complementary nature of the three layers. There was some overlap in detection capabilities of the three layers that was observed during testing as a natural benefit of the layered system.

The Layered system was tested against multiple data sets. For the sake of comparison against prior systems, this system was tested against a DARPA ’99 data set. For the sake of realistic normal data, it was tested against a private data set provided by a Fortune 500 company that is comprised of normal traffic from the internet to and from their primary web server. For the sake of comprehensive attack data, a custom dataset was constructed against the WebGoat test server. This comprehensive attack data includes attacks that span the detection capabilities of all three layers of the system, including navigation-based attacks and SQL Injection attacks. In the instances of the DARPA data and the Fortune 500 data, the attacks presented are request only attacks that were provided by the McPAD team and included shellcode-based attacks, buffer

overflow attacks, and various web server specific attacks. A more comprehensive description of the attacks in the McPAD data set is referenced previously in this document.

In the cases of the DARPA and the Fortune 500 datasets, the system was observed to be effective at detecting the McPAD attacks at a rate exceeding that of the prior state of the art systems. In the case of the WebGoat dataset, the system was observed to be superior in detecting the attacks than the prior state of the art systems.

The Layered Hidden Markov Model Payload Anomaly Detection System has expanded the state of the art in payload anomaly detection. The system has been shown to exceed the detection performance characteristics of prior systems and is capable of detecting additional attacks that prior systems could not detect. Further, the system has introduced two important characteristics of detection in the TCP stream reassembly and the response payload analysis.

Reference List

Aickelin, U., Bentley, P., Cayzer, S., Kim, J., & McLeod, J. (2003). Danger Theory: The Link between AIS and IDS? Artificial Immune Systems. In J. Timmis, P. Bentley, & E. Hart (Eds.), (Vol. 2787, pp. 147-155): Springer Berlin / Heidelberg.

Alam, M. S., & Vuong, S. T. (2007). APHIDS++: A Mobile Agent Based Intrusion Detection System. COMSWARE 2007, 2nd International Conference on Communication Systems Software and Middleware.

Anderson, D., Frivold, T., & Valdes, A. (1995). *Next-generation Intrusion Detection Expert System (NIDES): A Summary* (SRI-CSL-95-07): SRI International.

Anderson, J. P. (1972). *Computer Security Technology Planning Study* (ESD-TR-73-51). Bedford, MA: Electronic Systems Division, Air Force Systems Command, Hanscom Field.

Anderson, J. P. (1980). *Computer Security Threat Monitoring and Surveillance*. Fort Washington, PA: James P. Anderson Co.

Ariu, D., Tronci, R., & Giacinto, G. (2011). HMMPayl: An intrusion detection system based on Hidden Markov Models. *Computers & Security, 30*(4), 221-241.

Axelsson, S. (2000). The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security, 3*(3), 186-205.

Aydin, M. A., Zaim, A. H., & Ceylan, K. G. (2009). A hybrid intrusion detection system design for computer network security. *Computers & Electrical Engineering, 35*(3), 517-526.

Banković, Z., Stepanović, D., Bojanić, S., & Nieto-Taladriz, O. (2007). Improving network security using genetic algorithm approach. *Computers and Electrical Engineering, 33*(5-6), 438-451.

Baum, L., Petrie, T., Soules, G., & Weiss, N. (1970). A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics, 41*(1), 164-171.

Baum, L. E., & Petrie, T. (1966). Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics, 37*(6), 1554-1563.

Bednarczyk, M. (2012). jNetPcap [Computer Software]. Retrieved from http://jnetpcap.com/

Bell, D. E., & LaPadula, L. J. (1973). Secure Computer Systems: Mathemetical Foundations. *Mitre Corporation Technical Report 2547*.

Bertie, A. J. (2004). Java Statistical Classes (JSC) [Computer Software]. Retrieved from http://www.jsc.nildram.co.uk/

Biba, K. J. (1977). Integrity Considerations for Secure Computer Systems. *MTR-3153, The Mitre Corporation*.

Black, P. E. (2006, May 31, 2006). *Manhattan distance*. Retrieved October 15, 2011, from http://www.nist.gov/dads/HTML/manhattanDistance.html

Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Commun. ACM, 13*(7), 422-426.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F., &. (2008). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*: World Wide Web Consortium (W3C).

Brugger, S. T. (2008). *KDD Cup '99 dataset considered harmful*. Retrieved February 7, 2010, from http://www.bruggerink.com/~zow/GradSchool/KDDCup99Harmful.html

Caberera, J. B. D., Ravichandran, B., & Mehra, R. K. (2000). Statistical traffic modeling for network intrusion detection. 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems.

Cheswick, W. R., Bellovin, S. M., & Rubin, A. D. (2003). *Firewalls and Internet Security: Repelling the Wily Hacker* (2nd ed.): Addison-Wesley Longman Publishing Co., Inc.

Chou, T.-S., & Chou, T.-N. (2009). Hybrid Classifier Systems for Intrusion Detection. CNSR '09, Seventh Annual Communication Networks and Services Research Conference.

Corona, I., Ariu, D., & Giacinto, G. (2009). HMM-Web: A Framework for the Detection of Attacks Against Web Applications. ICC '09. IEEE International Conference on Communications.

Cullingford, R. E. (2009). Correlation and Collaboration in Anomaly Detection. Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications & Technology.

Day, J. D., & Zimmermann, H. (1983). The OSI reference model. *Proceedings of the IEEE, 71*(12), 1334-1340.

Debar, H., Curry, D., & Feinstein, B., &. (2007). *RFC 4765: The Intrusion Detection Message Exchange Format (IDMEF)*: IETF.

Denning, D. E. (1987). An Intrusion-Detection Model. *IEEE Transactions on Software Engineering, SE-13*(2), 222-232.

Detristan, T., Ulenspiegel, T., Malcom, Y., & Underduk, M. (2003). Polymorphic Shellcode Engine Using Spectrum Analysis. *Phrack, 0x0b*(0x3d).

Erlinger, M., & Staniford-Chen, S. (2002, April 4, 2002). *Intrusion Detection Exchange Format (idwg)*. Retrieved December 6, 2009, from http://www.ietf.org/proceedings/55/192.htm

Estevez-Tapiador, J. M., Garcia-Teodoro, P., & Diaz-Verdejo, J. E. (2004). Measuring normality in HTTP traffic for anomaly-based intrusion detection. *Computer Networks: The International Journal of Computer and Telecommunications Networking, 45*(2), 175-193.

Feinstein, B., & Matthews, G., &. (2007). *RFC 4767: The Intrusion Detection Exchange Protocol (IDXP)*: IETF.

Fogie, S., Grossman, J., Hansen, R., Rager, A., & Petkov, P. (2007). *XSS Attacks: Cross Site Scripting Exploits and Defense*: Syngress.

Fogla, P., & Lee, W. (2006). Evading network anomaly detection systems: formal reasoning and practical techniques. 13th ACM conference on Computer and communications security. Alexandria, Virginia, USA.

Fogla, P., Sharif, M., Perdisci, R., Kolesnikov, O., & Lee, W. (2006). *Polymorphic blending attacks*. 15th conference on USENIX Security Symposium. Vancouver, B.C., Canada.

François, J.-M. (2006). Jahmm [Computer Software]. Retrieved from https://code.google.com/p/jahmm/

Fries, T. P. (2008). *A fuzzy-genetic approach to network intrusion detection*. 2008 GECCO conference companion on Genetic and evolutionary computation. Atlanta, GA, USA.

Frincke, D. (2000). Balancing cooperation and risk in intrusion detection. *ACM Transactions on Information and System Security, 3*(1), 1-29.

Frincke, D., Evans, J., & Aucutt, D. (1996). Hierarchical management of misuse reports. *Journal of Computing Informatics*.

Frincke, D., Tobin, D., McConnell, J., Marconi, J., & Polla, D. (1998). A framework for cooperative intrusion detection. 21st NIST-NCSC National Information Systems Security Conference.

Gates, C., & Taylor, C. (2007). *Challenging the anomaly detection paradigm: a provocative discussion*. 2006 workshop on New security paradigms. Germany.

Gómez, J., Gil, C., Padilla, N., Baños, R., & Jiménez, C. (2009). Design of a Snort-Based Hybrid Intrusion Detection System. *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living, 5518/2009*, 515-522.

Gosling, J., Joy, B., Steele, G., & Bracha, G. (2005). *The Java Language Specification, Third Edition*: Addison-Wesley Professional.

Green, D. M., & Swets, J. A. (1966). *Signal Detection Theory and Psychophysics*: Wiley.

Heberlein, L. T., Dias, G. V., Levitt, K. N., Mukherjee, B., Wood, J., & Wolber, D. (1990). A network security monitor. IEEE Computer Society Symposium on Research in Security and Privacy.

Hofmeyr, S. A., Forrest, S., & Somayaji, A. (1998). Intrusion detection using sequences of system calls. *J. Comput. Secur., 6*(3), 151-180.

Holub, A. (1986). Directory traversal, trailing ^Zs, and horrifying experiences. *Dr. Dobb's Journal, 11*(9), 14-20.

Hwang, K., Cai, M., Chen, Y., & Qin, M. (2007). Hybrid Intrusion Detection with Weighted Signature Generation over Anomalous Internet Episodes. *IEEE Transactions on Dependable and Secure Computing, 4*(1), 41-55.

Ingham, K. L., & Inoue, H. (2007). *Comparing anomaly detection techniques for HTTP*. 10th international conference on Recent advances in intrusion detection. Gold Goast, Australia.

Jacobson, V., Leres, C., & McCanne, S. (1994). libpcap [Computer Software]. Lawrence Berkeley Laboratory. Retrieved from http://www.tcpdump.org/

Kahn, C., Porras, P., Staniford-Chen, S., & Tung, B. (1998). *A common intrusion detection framework*. Submitted to the Journal of Computer Security.

Kemmerer, R. A., & Vigna, G. (2002). Intrusion Detection: A Brief History and Overview (Supplement to Computer Magazine)*, 35,* 27-30.

Kernighan, B. W., & Ritchie, D. M. (1978). *The C Programming Language*. Upper Saddle River, NJ: Prentice-Hall, Inc.

Kolesnikov, O., Dagon, D., & Lee, W. (2006). *Advanced polymorphic worms: Evading IDS by blending in with normal traffic*. 15th USENIX Security Symposium.

Kolmogorov, A. N. (1933). Sulla Determinazione Empirica di una Legge di Distribuzione. *Giornale dell'Istituto Italiano degli Attuari, 4*, 83-91.

Kruegel, C., Toth, T., & Kirda, E. (2002). *Service specific anomaly detection for network intrusion detection*. 2002 ACM symposium on Applied computing. Madrid, Spain.

Lauf, A. (2007). *HybrIDS: Embeddable Hybrid Intrusion Detection System (Master's Thesis)*. Vanderbilt University, Nashville.

Lin, J.-L., Wang, X. S., & Jajodia, S. (1998). Abstraction-based misuse detection: high-level specifications and adaptable strategies. 11th IEEE Computer Security Foundations Workshop.

Lin, W., Xiang, L., Pao, D., & Liu, B. (2008). Collaborative Distributed Intrusion Detection System. FGCN '08, Second International Conference on Future Generation Communication and Networking, 2008.

Lippmann, R. P., Fried, D. J., Graf, I., Haines, J. W., Kendall, K. R., McClung, D., ... Zissman, M. A. (2000). Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. DISCEX '00, DARPA Information Survivability Conference and Exposition.

Litchfield, D. (2005). *Data-mining with SQL Injection and Inference* (An NGSSoftware Insight Security Research (NISR) Publication). Retrieved from http://www.ngssoftware.com/research/papers/sqlinference.pdf

Liu, Y., Chen, K., Liao, X., & Zhang, W. (2004). A Genetic Clustering Method for Intrusion Detection. *Pattern Recognition, 37*(5), 927-942.

Mahoney, M. V. (2003). *Network traffic anomaly detection based on packet bytes*. 2003 ACM symposium on Applied computing. Melbourne, Florida.

Mahoney, M. V., & Chan, P. K. (2001). *PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic* (CS-2001-04): Florida Institute of Technology Technical Report.

Mahoney, M. V., & Chan, P. K. (2003). An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection *Springer Berlin / Heidelberg*. Recent Advances in Intrusion Detection 2003.

Marin, G. A., & Allen, W. H. (2006). Needles in Haystacks: Practical Intrusion Detection from Theoretical Results. 31st IEEE Conference on Local Computer Networks.

Markov, A. (1971). Extension of the Limit Theorems of Probability Theory to a Sum of Variables Connected in a Chain. In R. Howard (Ed.), *Dynamic Probabilistic Systems (Volume I: Markov Models)* (pp. 552-577): John Wiley & Sons, Inc.

McHugh, J. (2000). 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.

Mukherjee, B., Heberlein, L. T., & Levitt, K. N. (1994). Network intrusion detection. *IEEE Network, 8*(3), 26-41.

Newman, D. (1999, October 28, 1999). *KDD Cup '99 Data Sets*. Retrieved February 7, 2010, from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Ning, P., Jajodia, S., & Wang, X. S. (2001). Abstraction-based intrusion detection in distributed environments. *ACM Trans. Inf. Syst. Secur., 4*(4), 407-452.

Northcutt, S. (2002). *Network Intrusion Detection* (Third ed.). Indianapolis: New Riders Publishing.

OWASP. (2012). WebGoat [Computer Software]. Retrieved from https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project

Parekh, J. J., Wang, K., & Stolfo, S. J. (2006). *Privacy-preserving payload-based correlation for accurate malicious traffic detection*. 2006 SIGCOMM workshop on Large-scale attack defense. Pisa, Italy.

Peng, J., Feng, C., & Rozenblit, J. W. (2006). A hybrid intrusion detection and visualization system. ECBS 2006. 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems.

Perdisci, R. (2009). *McPAD*. Retrieved October 12, 2011, from http://roberto.perdisci.com/projects/mcpad

Perdisci, R., Ariu, D., Fogla, P., Giacinto, G., & Lee, W. (2009). McPAD: A multiple classifier system for accurate payload-based anomaly detection. *Computer Networks, Special Issue on Traffic Classification and Its Applications to Modern Networks, 53*(6), 864-881.

Perdisci, R., Gu, G., & Lee, W. (2006). Using an Ensemble of One-Class SVM Classifiers to Harden Payload-based Anomaly Detection Systems. ICDM '06, Sixth International Conference on Data Mining, 2006.

Plato, A. (2004). *White Paper: What is an Intrusion Prevention System*: Anitian Corporation.

Porras, P. A., & Neumann, P. G. (1997). EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. 20th National Information Systems Security Conference.

Postel, J., &. (1981). *RFC 793: Transmission Control Protocol*: IETF.

Ptacek, T., & Newsham, T. (1998). *Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection* (Technical Report): Secure Networks, Inc.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*: Morgan Kaufmann Publishers Inc.

Rabiner, L. R. (1990). A tutorial on hidden Markov models and selected applications in speech recognition. In W. Alex & L. Kai-Fu (Eds.), *Readings in speech recognition* (pp. 267-296): Morgan Kaufmann Publishers Inc.

Roesch, M. (1999). *Snort - Lightweight Intrusion Detection for Networks*. 13th USENIX conference on System administration. Seattle, Washington.

Scholkopf, B., Platt, J. C., Shawe-Taylor, J. C., Smola, A. J., & Williamson, R. C. (2001). Estimating the Support of a High-Dimensional Distribution. *Neural Comput., 13*(7), 1443-1471.

Shannon, C. (1951). Prediction and Entropy of Printed English. *The Bell System Technical Journal, 30*, 50-64.

Smaha, S. E. (1988). Haystack: an intrusion detection system. Fourth Aerospace Computer Security Applications Conference.

Smirnov, N. V. (1948). Table for estimating the goodness of fit of empirical distributions. *Ann. Math. Stat., 19*, 279-281.

Snapp, S. R., Brentano, J., Dias, G. V., Goan, T. L., Heberlein, L. T., Ho, C.-l., ... Mansur, D. (1991). DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype. 14th National Computer Security Conference.

Song, Y., Keromytis, A. D., & Stolfo, S. J. (2009). Spectrogram: A Mixture-of-Markov-Chains Model for Anomaly Detection in Web Traffic. The Network and Distributed System Security Symposium (NDSS).

Souren, J. (2013). Security threats towards nations' critical infrastructures. *SC Magazine*. Retrieved from http://www.scmagazineuk.com/security-threats-towards-nations-critical-infrastructures/article/282187/

Staniford-Chen, S. (2008, February 18, 2008). *Intrusion Detection FAQ: What open standards exist for Intrusion Detection?* Retrieved December 6, 2009, from http://www.sans.org/security-resources/idfaq/id_standards.php

Staniford-Chen, S., Cheung, S., Crawford, R., Dilger, M., Frank, J., Hoagland, J., ... Zerkle, D. (1996). GrIDS - A Graph-Based Intrusion Detection System for Large Networks. 19th National Information Systems Security Conference.

Staniford-Chen, S., Tung, B., & Schnackenberg, D. (1998). *The Common Intrusion Detection Framework (CIDF).* Information Survivability Workshop, Orlando, FL.

Trusted Computer Solutions. (2001). CounterStorm-1 [Computer Software]. Herndon, Va

Varaneckas, T. (2001). JAD [Computer Software]. Retrieved from http://www.varaneckas.com/jad/

Vargiya, R., & Chan, P. (2003). Boundary Detection in Tokenizing Network Application Payload for Anomaly Detection. Workshop on Data Mining for Computer Security. Melbourne, FL.

Wagner, D., & Soto, P. (2002). *Mimicry attacks on host-based intrusion detection systems*. Proceedings of the 9th ACM conference on Computer and communications security. Washington, DC, USA.

Wang, K., Cretu, G., & Stolfo, S. J. (2005). *Anomalous payload-based worm detection and signature generation*. 8th International Symposium on Recent Advances in Intrusion Detection.

Wang, K., Parekh, J. J., & Stolfo, S. J. (2006). Anagram: A Content Anomaly Detector Resistant to Mimicry Attack. 9th International Symposioum on Recent Advances in Intrusion Detection.

Wenke, L., Stolfo, S. J., & Mok, K. W. (1999). A data mining framework for building intrusion detection models. Proceedings of the 1999 IEEE Symposium on Security and Privacy.

Wood, M., & Erlinger, M., &. (2007). *RFC 4766: Intrusion Detection Message Exchange Requirements*: IETF.

Xu, L., Krzyzak, A., & Suen, C. Y. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics, 22*(3), 418-435.

Yang, J., Ning, P., Wang, X. S., & Jajodia, S. (2000). *CARDS: A Distributed System for Detecting Coordinated Attacks*. IFIP TC11, Fifteenth Annual Working Conference on Information Security for Global Information Infrastructures.

Yi, X., & Shun-Zheng, Y. (2009). A Large-Scale Hidden Semi-Markov Model for Anomaly Detection on User Browsing Behaviors. *IEEE/ACM Transactions on Networking, 17*(1), 54-65.