2011

APECS: A Dynamic Framework for Preventing and Mitigating Theft, Loss, and Leakage of Mission Critical Information in Trust Management Networks

W. Owen Redwood

THE FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

APECS: A DYNAMIC FRAMEWORK FOR PREVENTING AND MITIGATING THEFT, LOSS, AND LEAKAGE OF MISSION CRITICAL INFORMATION IN TRUST MANAGEMENT NETWORKS

By

W. OWEN REDWOOD

A Thesis submitted to the
Department of Computer Science
in partial fulfillment of the
requirements for the degree of
Master of Science

Degree Awarded:
Spring Semester, 2011

The members of the committee approve the thesis of W. Owen Redwood defended on December 8, 2010.

Mike Burmester
Professor Directing Thesis

Sudhir Aggarwal
Committee Member

Xin Yuan
Committee Member

The Graduate School has verified and approved the above-named committee members.

To my parents. Thanks for your support, guidance, and everything you directly and indirectly taught me.

To my brother. Thanks for being my best friend and advisor.

"If I'd asked my customers what they wanted, they'd have said a faster horse."
-Henry Ford

ACKNOWLEDGMENTS

I am deeply indebted to my advisor, Dr. Mike Burmester, for his valuable guidance and support during the course of my research. I would also like to thank my friends and classmates for their honest input regarding the research. Finally, I would like to thank the members of my committee and the faculty of the Florida State University Department of Computer Science for their support and inspiration.

-W. Owen Redwood

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Existing solutions that address data loss, theft, and corruption of information and resources in networks rely on deep content analysis, central policy management, and attempt to achieve broad content protection across multiple platforms and locations [47]. These solutions unfortunately are designed to protect against careless users and very unsophisticated malicious insiders. Consequentially they are trivial to defeat with simple obfuscation. This thesis outlines the flaws with existing approaches and borrows lessons and techniques from related security systems in order to propose a novel approach on policies and mechanisms that are more ideally suited for addressing this problem.

This thesis describes the design, implementation, and analysis of *real-time statistical (Markov chain and Bayesian) analyzers* (extending work of [24, 14, 28, 33, 40, 55]) for network anomaly detection to trigger *novel policy-based* temporal resource access-disruption mechanisms (extending the work of [12, 13, 41]). These temporal resource access-disruption mechanisms (aka *Rollback-access* mechanisms) dynamically mitigate the risk of security-critical file distribution by rolling back the granted access to the aforementioned files upon detecting that the user is a perceived threat. The analyzer design goals are: to minimize the consequences of anomalous behavior, to make the analyzer resist Denial of Service(DoS) attacks, to have a real-time response time to anomalies, and to deal with network threats without seriously disrupting services. The resultant temporal access-disruption mechanisms provides for an unprecedented resilience to resource-centric attacks. Additionally, we present some experimental results, which demonstrate the potential of the aforementioned mechanism. Finally, it is important to note that while this thesis extends existing work [12, 13], it addresses only one of many aspects that are necessary to actually implement such systems.

CHAPTER 1

INTRODUCTION

With the advent of the computer security modeling era in the 1970's-80's, the concept of the Mandatory Access Control and Discretionary Access Control layers (MAC and DAC respectively) were first envisioned by David Elliott Bell and Leonard J. La Padula [7] to provide secure mechanisms to control access to resources/information. This concept established the foundation for secure multi-user computing, and secure network file systems. Several integrity models emerged in that era (most notably Biba, Lipner, Clark-Wilson), and almost all models showed some inherent similarity to the Bell-La Padula confidentiality model [32]. Steve Lipner conceived of the use of an access control matrix to address file system protection and security to ensure integrity in his famous integrity model which built upon the work of Bell-La Padula's model [32]. However, the matrix model introduced by Lipner did not scale well with the number of users in a network, and this motivated the innovation of Role Based Access Control (RBAC) [21, 45]. RBAC systems assign to users different roles, and each role has specific permissions. Conceptually, RBAC systems can work in conjunction with MAC and DAC systems, and offer a more efficient mechanism to manage access control. Many systems today are based on RBAC mechanisms. A notable weakness in all of these solutions originally was that these models required some form of a centralized architecture to manage access control.

Trust Management (TM) systems emerged to provide decentralized solutions. TM systems are infrastructures designed to provide scalable and efficient access control management to the network resources. A TM system can be formally described by a lattice structure in which authorization for access corresponds to trusted flow path, and access requests are established through policies [52].

Various TM systems have been developed over the past twenty years. Some systems focused on the logic of the TM engine [2, 5, 35], others on general purpose authorization for the system [10, 11, 50], and others on specialized purposes (*e.g.* combining ACL and PKI) [6, 16, 25]. However, over the past twenty years, network administrators often chose to implement simpler TM systems such as PGP and variants of SDSI [57, 42]. Today, the systems that have attracted the most focus are SPKI[50], KeyNote[10], RT*[36], and still RBAC based systems [21, 45].

While these systems offer adequate solutions for resource/information confidentiality, integrity, and availability for *normal* users, the need for protection against *adaptive malicious* outsiders (outside users) spawned the proliferation of Intrusion Detection or Intrusion

Prevention Systems and anti-virus/malware software. Intrusion Detection Systems (IDS) and anti-virus/malware software offer reactive defenses against outsider attacks. However as the attack detection mechanisms become more accurate and responsive, and automated response mechanisms became common, IDS eventually evolve into Intrusion Prevention Systems (IPS).

The attack detection mechanisms in IDS have recently become fairly sophisticated, with the leading contemporary systems using a combination of signature-based and anomaly-based detection. However, the open problem of protecting the network and its resources against *malicious* insiders has had few sophisticated solutions. Unfortunately, malicious insiders require little sophistication to defeat most existing mechanisms that address this problem. Malicious insiders can also be seen as infected terminals or hacked accounts of normal users, that are preprogramed or remotely controlled in an undetected manner, to carry out attacks that violate the integrity, confidentiality, and even availability of network resources and sensitive information.

1.1 Motivation

The theft (or compromize) of sensitive digital information is highly lucrative for cyber criminals, and is a high priority goal for some powerful nations. Recently there have been a number of such events that have had the potential to inflict incalculable damage [44, 43]. The most notable being the Wikileaks classified information dump in October 2010 which was enabled by a malicious insider that leaked several hundreds of thousands of classified documents [1].

This thesis aims to open the door to a new realm of solutions to address such security issues. In order to accomplish this, we require a mechanism that can predict such aattacks in real time.[1]

A common property of the *observable behavior* of attacks discovered by various malware, cyber defense, and vulnerability analysts throughout the years, is that the overwhelming majority of attack behavior is anomalous when compared to the normal behavior of any network/system [56, 23, 22, 34]. The behavior of attacks is most commonly observed via network traffic or system calls. However, it is important to note that not all anomalous behavior is malicious. Consequentially, building a resource defense that is triggered in some sort by anomalous behavior will have false positives in any network. This will indubitably result in disruption of resource availability in the network due to non-malicious anomalies. Nevertheless, there certainly exist networks where the need to maintain the utmost confidentiality of resources outweighs the need for perfect availability. This research is intended for such networks.

This thesis proposes a solution to this problem by implementing Markov chain analyzers that monitor outgoing network traffic and report when the traffic appears anomalous, which in turns triggers a Network Policy to protect the confidentiality of the resource. Since Markov chain modeling has a high success rate for predicting future events in various industries, predictive applications utilizing Markov chains are usually known to work well when

[1]For example, uploading an unusually large number of confidential files by a user with secret clearance should be prevented before the task is completed.

correctly applied. However, most applications are for physical processes that operate under strict laws. When it comes to monitoring human behavior for anomalous actions (*e.g.,* Byzantine), the application is similar regardless of the fact that humans are not bound by such behavioral laws. Here, our metrics for detecting anomalous behavior are dynamically influenced by the system threat level, existing behavioral profile(s), and the information resources currently allocated to the user.

Recently two TM systems were proposed which support a "Rollback Access" fucntionality intended to dynamically mitigate the system risk of mission-critical information loss/leakage under perceived threats [12, 13]. As a part of this research, we proposed a dynamic trigger for the Rollback Access mechanism using Markov chain profiling [41]. These proposed solutions utilize a dynamic form of Markov anomaly detection to trigger the Rollback Access mechanism. Existing Markov anomaly detection systems develop a system profile that captures typical system behavior and looks for events, or series of events, that are not supported by the profile [14, 54, 28, 55] by comparing the support to a static threshold. Essentially, when the user's behavior appears too anomalous for the system to tolerate, the system engages a resource rollback mechanism to safeguard the system. This mechanism is flexible and is controlled by the trust policies, thus protecting resources for those users that are most trusted. Here "behavior" refers to the outgoing network traffic of that user. Rollback can involve erasing, preserving, or branching off a user's changes to a file when a rollback occurs, which is further supported by the existing MAC and DAC related mechanisms [12, 13]. It is important to note that while Rollback Access can be implemented in a number of ways, and has already been detailed in the literature, our approach in this thesis is to show how it can be managed in an efficient and scalable way.

1.2 Formal Problem Statement

The majority of existing security solutions are unable to defend against the most damaging security threats posed by the attacks of the future. Signature-based (aka knowledge-based) intrusion detection/prevention systems constantly look towards defending against the attacks of today and the past. While Anomaly-based intrusion detection/prevention systems have the capability to detect and prevent future attacks and zero-day attacks, the few systems that implement response mechanisms utilize anti-virus, firewalls, process-priority throttling, and network-traffic throttling mechanisms [30, 22].

In security critical networks that contain mission critical information, the defense of said information also requires an automated mechanism. While numerous solutions here already exist [47], all are trivially defeated by unsophisticated means. We require a sophisticated mechanism that can preserve the integrity and confidentiality of mission critical information that has been allocated to trusted/qualified users, whom have afterwards become a perceived threat to the system.

CHAPTER 2

BACKGROUND

In 1984 Fred Cohen published a paper that theoretically proved that the problem of detecting computer viruses is NP-hard, and that finding the *witness* for it is indeed undecidable [17]. Specifically Cohen demonstrated that detection by "appearance" of viruses is an infeasible and undecidable problem. It is important to note that this is proved with the example of a virus that can detect that it has been or can be detected. In other words, it is impossible to detect every type of intrusion that might occur in every scenario. Furthermore, his work essentially illustrates that the resources needed to detect (in a reasonable amount of time) intrusions grows with the amount of traffic on the network. These are important facts to consider when analyzing existing security systems (specifically Intrusion Detection/Prevention Systems and Data-Leak-Protection systems).

Binary responses are typical in the realm of computer security: file access is either permitted or not (access control mechanisms); files are either encrypted or not (cryptography); and connections are either permitted or blocked (firewalls and gateways). When computer viruses were observed to exhibit biological behavior in 1972 [48], security researchers began focusing on existing natural biological defenses. Biological defenses employ a graduated response to attacks, rather than a binary one. Naturally, the human immune system inspired much research in the areas of virus/intrusion detection and network security [22]. The most cited resulting security mechanisms that employed graduated responses typically throttle traffic [30] and/or process-priority [23]. In these systems, small disturbances result in small responses, and large disturbances result in large responses. Architectures that employ graduated responses tolerate imperfect detection of threats, in order to achieve more dynamic security.

2.1 Intrusion Detection/Prevention Systems

Intrusion Detection Systems (IDS) have two types of detection modes: 1) signature-based (aka misuse-detecting, knowledge-based, pattern-matching-based, rule-based) and 2) anomaly-based (aka statistical-based) intrusion detection. Intrusion Detection Systems and the remaining systems in this chapter can all (at least) be host-based (HIDS) and/or network/gateway-based (NIDS). These systems traditionally focus on incoming attacks against the host or network. Eventually systems were developed to automatically respond to some (but not all) detected attacks in order to prevent them. These systems are termed

Intrusion Detection/Prevention Systems (IDPS or IPS).

The area of detecting and preventing attacks originating inside the network that are directed against it is loosely referred to as *extrusion detection*. Extrusion detection, as a category evolved into Data-Leak-Prevention systems (and similarly termed systems).

2.1.1 Signature-Based Intrusion Detection and Prevention

The NIST (National Institute of Standards and Technology) formally defines intrusion detection as *"the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices"* [38].

Signature-based detection is the primary mode of almost every existing Intrusion Detection System that is publicly available. This mode offers high-accuracy in detecting attacks that it has the signature base to detect. However, signature-based IDS effectiveness is based on the IDS's signature knowledge. Signatures can consist of packets' header information and/or payload information. All signature-based IDS by definition, will fail to detect attacks that it does not have a signature (or similar signature) to match to the attack (e.g. novel attacks, zero-day attacks, or tweaked attacks) [38].

Furthermore in certain scenarios, sophisticated attackers can fool almost all known IDS systems by altering the signature of the traffic via computationally cheap obfuscation techniques (e.g. the simplest compression of packet payloads fools existing IDS that are DPI-reliant). Attackers primarily use such a technique to obfuscate the incoming and outgoing network traffic to an infected terminal inside the network. Another common problem is that most IDS solutions fail to implement Deep Packet Inspection (DPI) mechanisms that reconstruct the traffic flow to the application layer. Even some commercial IDS solutions fail to reconstruct the traffic flow at any level. Other solutions discussed later in this chapter also commonly make the same or similar fatal mistakes.

2.1.2 Anomaly-Based Intrusion Detection and Prevention

The NIST formally defines anomaly-based intrusion detection as *"the process of comparing definitions of what activity is considered normal against observed events to identify significant deviations"* [38]. To accomplish this, the IDS forms a profile (or a baseline) of normal activity. The first comprehensive anomaly-based intrusion detection model dates back to 1987 [18]. Other early anomaly-based intrusion detection research explored various statistical methods and models (i.e. Bayesian, Markov chain, and Hidden Markov Models) to achieve an anomaly-based detection system that yielded high accuracy and low false-positive rates in attack reporting [19, 29, 37, 22, 46].

The computational overhead of anomaly-based detection systems includes the time it takes to develop the profile and the number of passes the detector performs while processing the input. A sufficient, but not necessary property of a detector to perform in real time, is that it is limited to a single pass over the input during anaylsis [22].

Anomaly-based attack detection has been enormously popular in system-call based intrusion detection [23, 22]. Developing the normal behavior profile for system call based

detectors is relatively simple, since the average program makes over a million system calls per execution.

For anomaly-based detection in network traffic analysis, the system must learn the profile/baseline over time by observing: what protocols are typically used, what ports and devices communicate with each other, and the amount of bandwidth typically used. Generally, if the profile is designed to distinguish the application responsible for each traffic stream, the cost and complexity of developing the normal behavior profile increases. This will require the analyzer to identify the host's network-applications in real time from traffic analysis, in order to use the appropriate profile for more accurate anomaly-detection. However, this has been a difficult problem for researchers, as few such existing mechanisms are considered "reliable" [51]. Reliable application-identifying methods cannot rely on payload inspection as it can be easily obfuscated (e.g. encryption, compression, and padding). Furthermore, reliable methods must also be able to quickly identify the network application, as demonstrated in typically under 6 packets in [8].

Consequentially as of this writing, the security community has yet to reach a consensus on an acceptable real-time application-identifying traffic analyzer method that would be the standard of reliability, expediency, and accuracy in such methods. Peer review of proposed methods in this area is unfortunately sparser than other fields of security research. The identification of an *acceptable* method would greatly benefit the research of this thesis, and thus remains as future work.

Nevertheless, real-time anomaly-based traffic analyzers that do not require identifying the application exist and have been shown to be effective. The most cited anomaly-based intrusion detection example is SPADE (Statistical Packet Anomaly Detection Engine) [9]. SPADE was funded a DARPA funded project in 2003 and was promptly abandoned after funding stopped. SPADE used simplistic Bayesian statistics to track the frequency of packets. Another example is a recently proposed system that utilizes Markov Chains to achieve more accurate prediction in real-time [56].

2.1.3 Signature vs Anomaly based Intrusion Detection

Signature-based detection systems establish a set of illegal states via its collection of signatures for known attacks or misuse. Whereas, anomaly-based detection systems establish a set of legal states via its learning process that captures the normal behavior of the network. The illegal state set in the anomaly-based system is the compliment of the legal state set established by the profile (and the defined probability threshold(s)). Consequentially, anomaly-based systems have the potential to detect attacks not yet discovered by signature-based systems.

By the previously mentioned property that virus/attack appearance/signature-based detection is undecidable and NP-hard [17], it is easy to hypothesize that for all systems: the illegal state set in a signature-based detection system can never be larger than the illegal state set for a anomaly-based detection system. In order for a signature-based detection system to detect all the attacks that its anomaly-based component can detect, it must first observe all possible attacks that are anomalous. This clearly presents an untenable tautology.

However, anomaly-based detection systems are only as good as their profiles, and by

definition permit attacks that do not appear significantly different from normal behavior. Accurately developing profiles depends on the application at hand, and can be established (and maintained) in a variety of methods. If a system decides that the learning-process for a profile is complete when the resultant profile can capture all normal behavior for the system (or application), it is easy to see that this also is an untenable process for today's ever-evolving Internet. Consequentially, this decision is a qualitative one.

Next we must consider the maintenance of the profile, since applications, user behavior, and even network topology are subject to change. Automatically detecting exactly when it is correct to update the profile is clearly an undecidable problem: as it requires the knowledge that the current profile is insufficient to capture all the normal behavior of the system. Consequentially, the the evolution (or update) of the profile is often triggered on a regular time interval in systems. So, unlike signature-based detection, the collection and evolution of knowledge about the system is rather imprecise. Furthermore in anomaly-based systems, there remains the possibility that the system *"learn"* that an attack is normal behavior during the learning-process or the profile evolution process.

Given this imprecision, complexity, and uncertainty, why are anomaly-based systems imperative for security-critical networks? There is a simple but loaded answer: as long as these networks are the targets of highly-motivated and highly-sophisticated adversaries, they will be subjected to attacks that have not been discovered by signature-based detection systems. Therefore, the novel-attack detection capabilities of anomaly-based detection systems are essential in defending against such attacks. This clearly assumes that the vast majority of network traffic is normal, and not malicious. When this assumption does not hold true, the security provided by anomaly-based detection systems will indubitably deteriorate. However, if such attack-traffic-dominated networks exist, they clearly are poor choices for handling the secure distribution of mission-critical information. It is certainly worth noting that establishing anomaly-based detection security systems, intangibly exacerbate the difficulty of approach for any would-be adversary. Whether this increase in difficulty yields appreciable results, is an undecidable question in itself.

2.1.4 Intrusion Prevention Systems

Nevertheless, the signature-based IDS have been the most popular variant. Consequentially, IDS evolved into Intrusion Prevention Systems (IPS), with the following distinction: "An intrusion prevention system (IPS) is software that has all the capabilities of an intrusion detection system and can also attempt to stop possible incidents" [38]. IPS systems commonly interact with firewalls and gateways to immediately attempt to stop suspicious incoming/outgoing network traffic. NIST formally explains that IPS defenses involve the following mechanics[38]:

- Terminiation of the suspicious network connection (along with any or all user sessions being utilized for the attack).
- Shut off access to the target(s) from the attackers user account or IP address (or potentially other address/identifier form).
- Completely shut off access to the targeted host(s), service(s), application(s), or resource(s).

2.1.5 State of the Art Systems

Finally, it is impossible to discuss IDS and IPS systems without touching the Einstein 1 and 2 systems developed by the US-CERT (United States-Computer Emergency Readiness Team) to protect the United States of America's government and military critical networks. It is important to note for complexity and scalability analysis, these systems need to secure the over 2000 known networks (each with numerous gateways and access points) controlled in some form by the US government. Many of these networks are controlled by distinct agencies, which further poses significant problems for timely intrusion detection response. The NSA (National Security Agency) is officially moving towards developing Einstein 3.0 which would have intrusion prevention capabilities. It is reasonable to assume that these systems possess the most advanced capabilities to date. However, in a speech to the 2008 RSA Conference, Department of Homeland Security Secretary Michael Chertoff revealed the three limiting constraints to their IDS/IPS systems and efforts [15]:

- There are so many access-points and gateways to the multi-domain government network, that it is impossible to centrally monitor everything with gateway-based systems.
- Intrusion alerts trigger a human-based response which is not instantaneous, and many agencies do not have 24/7 response capabilities.
- "[T]he architecture of EINSTEIN is fundamentally a backward-looking architecture. We identify anomalies by looking at traffic that has penetrated or entered certain federal domains. We analyze that to see whether we believe there is something nefarious going on. We contact the agents and we ask them to look at their own networks. And then when they come back to us, if we determine there's an attack, we try to take steps to identify the signature and characteristics of the attack so we can identify it if it comes in the future. This doesn't happen instantly, and the time delay in all the steps of the process I've told you is time we cannot afford to lose in a world in which attacks come literally in microseconds and from all points of the globe. So these constraints have limited the effectiveness of our ability to deal with an increasing tempo of attacks on our federal civilian domains" [15].

2.2 Data-Leak Prevention (DLP) Systems

2.2.1 Overview

Data-Leak Prevention (DLP) has a plethora of alternative names due to initial market confusion (To name a few: Data Loss Prevention/Protection, Information Loss Prevention/Protection, Information Leak Prevention/Protection, Extrusion Prevention, Content Monitoring and Filtering, Content Monitoring and Protection). DLP systems are essentially security solutions that focus on sensitive information escaping or being intentionally leaked out of secure networks. *Securosis* defines DLP as: *"Products that, based on central policies, identify, monitor, and protect data at rest, in motion, and in use, through deep content analysis"* [47]. As of this writing, there are dozens of DLP solutions, and all of them utilize the following techniques [47]:

- The perform deep content analysis (aka DPI), in order to identify sensitive information leaving the network. This requires clear-text payloads at the gateway level, and multi-pass pattern matching to perform analysis.
- They require central policy management.
- They observe broad content coverage across multiple platforms (mainly Windows and Red-Hat Linux) and locations (multi-domain).

2.2.2 Disadvantages of existing DLP solutions

Since the core component of DLP solutions is deep packet inspection (aka deep content analysis), they are ill-equipped to protect against malicious insiders with any sophistication. Furthermore, they cannot respond in real time if the DLP analyzers parse the traffic payloads for all combinations of sensitive information patterns, as this requires multiple passes.

Furthermore, in a multi-domain setting with distributed resources to protect, the fundamental mechanisms behind these solutions become computational infeasible as the domains and resources scale upwards. All deep packet inspection techniques perform slower as the resource pool grows.

CHAPTER 3

MATHEMATICAL BACKGROUND

3.1 A Formal Introduction to Markov chains

Definition 9.1: [20, 39] A discrete-time random process $X = \{X_1, X_2, \ldots\}$ is a *Markov chain* if it satisfies the Markov property:

$$Pr(X_{t+1} = x_{t+1} | X_t = x_t) = \quad Pr(X_{t+1} = x_{t+1} | X_t = x_t, \ldots, X_1 = x_1), \tag{3.1}$$

where the x_i belongs to the support S.

The *transition probability (also known as: "Conditional Discrete Density Function")*

$$p_{ij}(t, t+1) = Pr(X_{t+1} = x_j \mid X_t = x_i)$$

is the probability that the state of the process at time $t+1$ will be x_j, given that at time t its state was x_i. In Markov chains, if the transitional probabilities do not depend on time, the process is *time-homogeneous*. We will *not* be dealing with time-homogeneous Markov chains.

More generally the transitional probability that after n-steps the state will be x_j, given that the start state was x_i is denoted by $p_{ij}^{(n)}$. It is easy to see that:

$$p_{ij}^{(n)} = \sum_{r=1,\, 0<k<n}^{n} p_{ir}^{(k)} p_{rj}^{(n-k)}. \tag{3.2}$$

This formula is derived from the *Chapman-Kolmogorov* equations [24].

With this, we can illustrate the 1-step and n-step *transition probability matricies*. P^1 is a square, stochastic matrix of notation:

$$P^1 = [p_{ij}(1)] = \begin{pmatrix} p_{00} & p_{01} & \cdots & p_{0n} \\ p_{10} & p_{11} & \cdots & p_{1n} \\ \cdots & \cdots & \cdots & \cdots \\ p_{n0} & p_{n1} & \cdots & p_{nn} \end{pmatrix}$$

Calculating P^n is achieved by applying formula (3.2). Markov chains can be modeled by finite state machines since all that is necessary to predict the next state is the current state.

In many applications (in particular ours) random processes have memory. A Markov chain of order m predicts the next state based on the past m states.

3.1.1 Properties of Markov chains

There are several key properties of Markov chains that require discussion. Markov chains can be decomposable or non-decomposable, periodic or aperiodic. The state space, can be finite or countably infinite. These properties determine how transition probabilities are calculated, the calculation of P^n, as well as the convergence of P^n as $n- > \infty$ [39] (Convergence is also referred to as the "long-run distribution" in literature).

A Markov chain is said to be decomposable if there exists a state within it that it cannot reach another state in the machine within a finite number of steps. Conversely, a Markov chain is non-decomposable if every state in the machine can reach every other state in a finite number of steps. Formally, stated: for all non-negative integers i and j, there is an integer $n \geq 1$ such that $p_{ij}(n) > 0$. An important property of non-decomposable Markov chains is that all states are homogenous, such that if one state is aperiodic – then all states are aperiodic (Thereby making the Markov chain aperiodic) [24, 39].

When the state space I of the Markov chain X_n is finite, then the transition calculation is simple. A Markov chain X_n is finite when it has K states and the number of possible values of the random variables X_n is finite and equal to K. However, when I is countably infinite, then obtaining P^n requires asymptotic analysis [24, 39]. We will not be dealing with countably infinite state spaces in this paper.

3.1.2 Markov chains of order M

Definition 9.2: [20] A discrete-time random process $X = \{X_1, X_2, \ldots\}$ is a *Markov chain of order* m if:

$$Pr(X_n{=}x_n \mid X_{n-1}{=}x_{n-1}, \ldots, X_1{=}x_1) = Pr(X_n{=}x_n \mid X_{n-1}{=}x_{n-1}, \ldots, X_{n-m}{=}x_{n-m}) \tag{3.3}$$

for any integer $n > m$.

In our variable-threat application, the size of m is determined by the security policy of the TM system: when the threat level is elevated, the value of m should be high; whereas when it is low the threat level m can be low. Additionally, m is influenced by the resources currently accessed by the user, as will be detailed later.

For Markov chains of order m the transition n step probability (3.2) is expressed by:

$$p_{ij}^{(n)} = \sum_{r \in S,\, O<k<n} p_{ir}^{(k)} p_{rj}^{(n-k)}. \tag{3.4}$$

We can use the transition probabilities (3.4) to compute the *temporal Markov distribution* $Pr(X_n{=}x)$ for the system states at time t, given an initial distribution $Pr(X_1{=}x)$. Markov chains of order m are modeled by Turing machines, since they require the use of a memory tape.

The Markov Evolution Function. For dynamic Markov chains, the transition probabilities are updated by an *evolution function* upon an observed state change. Any state change can trigger the evolution. In our model evolution is triggered after a variable length time interval. This interval and the evolution function are crucial in maintaining the accuracy of our stochastic Markov behavior analyzer.

The evolution of the process through one step is given by:

$$Pr(X_n = x_j) \;=\; \sum_{r \in S} p_{ir} Pr(X_{n-1} = x_r), \tag{3.5}$$

in terms of the earlier transition probabilities (3.4).

Markov chains are commonly described by directed graphs in which the edges are labeled by the probabilities of going from one state to the other states.

The following result is given, to support our case for Markov profiling [24]: *Non-decomposable, aperiodic, Markov chains typically converge to a fixed profile; that is,*

$\lim_{n\to\infty} p_{ij}^{(n)} \;=\; q_i$.

regardless of the initial probability distribution. This property is *extensively* proven in [39].

3.2 Bayesian Inference

In probability theory, Bayesian inference is a method by which some observation or evidence is used to: 1) calculate the probability, with a degree of confidence, that a hypothesis regarding the event may be true, and 2) to update the previously calculated probability for the event [46, 20, 39].

For anomaly detection applications, Bayesian inference describes the use of a learned (or *prior*) probability over a hypothesis to determine the chance of that hypothesis given observed events. Specifically, the hypothesis would be either that the entity generating the event is behaving anomalous or normally. The chance of the hypothesis holding true given the observed events and the prior probability, is termed the *posterior probability of the hypothesis* [46, 20]. This can also be seen as the confidence that the Bayesion application has in the given hypothesis.

Following the calculation of the posterior probability of the hypothesis, the prior probability is updated by the following formula:

$$Pr[H_j|X] = \frac{Pr[X|H_j]Pr[H_j]}{Pr[X]} \tag{3.6}$$

Where:

- H_j refers to hypothesis j.
- X refers to the observed event.
- $Pr[H]$ refers to the *prior probability* of H that was inferred before observing X.
- $Pr[X|H_j]$ refers to the conditional probability of X occuring when H_j is true.
- $Pr[X]$ refers to the *a priori* possibility of observing X under all hypothesis, as given by the following:

$$Pr[X] = Pr[X|H_1]Pr[H_1] + \ldots + Pr[X|H_n]Pr[H_n] \tag{3.7}$$

3.3 Comparing Bayesian models, Hidden Markov models and Markovian chain models

One may model an access control system by a stochastic finite state machine $\mathcal{F}$ with state set $\mathcal{S} = \{s_i\}$. A Bayesian behavior model is captured by the triple: $< \mathcal{N}, \mathcal{SP}, \mathcal{T} >$ where $\mathcal{P} = \{Pr(s_i|s_j), s_i, s_j \in \mathcal{S}\}$ is the set of transition probabilities, and $\mathcal{T}$ a set of thresholds for the transition probabilities (used to determine anomalous behavior). In any realization of the system, the transition probabilities can only be approximated over learning period, and then used for future behavior analysis, which will involve the thresholds in $\mathcal{T}$. With this approach the Bayesian analyser averages out over time the the transient probabilities and then compares this with the sampled behavior using the thersholds to decide whether the system behavior is anomalous (or client behavior, if the analysis focusses on a particular client). Without a *prior probability* updater, the Bayesian analyzer can be quite crude in its decisions, in the sense that actual behavior is compared with a fixed expected "normal" behavior.

Many researchers use Hidden Markov Models (HMM) for behavioural analysis; however, Markov chains can be regarded as a simplified version of HMM and require fewer parameters to be set, thus simplifying the training phase, and reducing the way of supplementing Markov chains with additional setup phase. The standard task of a HMM is to understand a hidden action or property based on available input. In the Markov chain models there is no hidden information so the task of understanding and interpreting information is achieved within the model.

CHAPTER 4

APECS: ANOMALY-(BASED)-POLICY ENFORCEMENT CONTROL SYSTEM

4.1 Architecture Overview, Requirements, and Details

APECS is designed for multi-domain, variable threat environments that value the confidentiality and integrity of mission-critical information over the availability of such information *(extending the R-TM system in [12])*. Specifically, APECS is designed for enforcing policies that focus on reacting to anomalous behavior of the domain, the user(s), and other trusted domains. Fundamentally, anomaly-based-policies can be expanded to focus on securing parts of the domain other than confidentiality and integrity for information. Conceptually, these policies can be expanded to cover already existing traffic and process throttling mechanisms. However, re-conceptualizing existing work under the APECS framework is not quite novel, and is outside the scope of this thesis. Thus we only present resource-centric anomaly-based policies in this work, and simply demonstrate that this framework is compatible with existing gradual response security mechanisms, by extending .

APECS uses a gradual response mechanism (similar to throttling) to dynamically prevent and mitigate the theft, loss, and leakage of sensitive information. To achieve this, APECS employs rollback-access mechanisms [12, 13], which temporarily disrupts the access of resources provided to users when those users are perceived as a potential threat to the system. The graduated nature of the rollback-access response is achieved via the Threat Level policies and the Threat Level Control layer, which allows for the specification of variable sensitivity (i.e. dynamic thresholds) to anomalies, at each Threat Level [41].

This framework is fundamentally designed to perceive users as a threat when their current behavior significantly deviates from their normal behavior. Here we focus on outgoing network traffic for a user's behavior, while largely ignoring DoS prone protocols. While anomalous behavior is not always malicious behavior, this approach offers the broadest security coverage against attacks, and can be deployed on top of existing IDS/IPS solutions. APECS requires the deployment of secure operating systems compatible with rollback-access mechanisms and secure network file systems. Furthermore, host-based hardware analyzers must be designed to be tamper-proof.

4.2 The Rollback-Access TM Model and Threat Levels

This section provides a brief overview of the Rollback-Access TM model outlined in [12], and explains the notation for the reader. In this model, the primary functionality of the TM is to *authorize* actions of entities. We denote TM^{auth} as the authorizing functionality, and that TM^{auth} *realizes* TM.

The authorizations of TM^{auth} are restricted by the threat level in our model. Thus, if the threat level is θ, then the possible authorizations of TM are denoted by TM^{auth}_{θ}. Threat levels are a construct of the network's security policy, that describe the degree of perceived threat posed by entities related to the trust management network. Threat levels can be local or global, and linear or non-linear. For instance, the United States Department of Homeland Security implements a global, 5-stage threat level which is a linearly ordered set (low/green, guarded/blue, elevated/yellow, high/orange, and severe/red), in which each stage restricts the possible authorizations available (in addition to the implied increased security measures and threat awareness).

Generally, threat level systems are modeled by partially sets $(\Theta, \succeq)$. It is denoted that the set of trust management systems that Θ induces on TM by:

$$TM_{\Theta} = \{TM_{\theta}\}_{\theta \in \Theta} \tag{4.1}$$

This is called a *multi-domain* trust management system with *rollback access* (R-TM) [12]. Furthermore TM_{Θ} has the natural dominance relation $\succeq_{auth}$ such that $TM^{auth}_{\theta 1} \succeq_{auth} TM^{auth}_{\theta 2}$. This denotes that all actions authorized under $TM^{auth}_{\theta 2}$ are also authorized under $TM^{auth}_{\theta 1}$. This requires that policies that span across multiple threat levels have their restrictions be monotonically increasing as the threat level increases.

Users, groups of users (for role-based access control [21, 45]), the entire domain, and other trusted domains each can have their own threat level. Conceptually, threat levels can be seen as the inverse of the level of trust the system has towards an entity. If an entity's threat level is low, the system asserts that there is a low probability of threatening behavior from that entity in the immediate future. Conversely, if an entity's threat level is high, the system asserts that there is a high probability of threatening behavior from that entity. Therefore, when the threat level is elevated in our model to θ and the resulting TM^{auth}_{θ} no longer authorizes the current (or ongoing) actions of the user under the previous TM^{auth}_{0-1}, then the current unauthorized actions of that user are *rolled-back*. For example, if a user was authorized to access some resource(s) and then the threat level elevates while that user is accessing the resource, access to resource(s) may be revoked if the TM^{auth} no longer authorizes such access. This occurs independently of other trust mechanisms that may apply.

Each threat level specifies policies that are to be enacted under that degree of perceived threat. Naturally, these policies will differ for each entity type, and these policies can be either heterogeneous or homogeneous across the threat levels. For example, a single homogeneous policy for the threat levels of users could be:

- Threat level 0 (lowest) $\rightarrow TM^{auth}_{0}$: *Zero penalty against the user's clearance.*
- Threat level 1 $\rightarrow TM^{auth}_{\theta 1}$: *Penalize the user's clearance by 1 security level.*

- Threat level 2 $\rightarrow TM^{auth}_{\theta 2}$: *Penalize the user's clearance by 2 security levels.*

$\vdots$

- Threat level N $\rightarrow TM^{auth}_{\theta n}$: *Penalize the user's clearance by N security levels.*

Here the penalized clearance would have a lower bound of the lowest clearance (e.g. unclassified).

To model the policies of a TM system one may use *information flows*: the policies specify which flows are permitted and which are forbidden. For example in the Bell-LaPadula [7] privacy model, an information flow from a *classified* resource to a user with *secret* clearance is permitted, while a flow from the same resource to a client with *unclassified* clearance is forbidden.

In our model, the threat levels for entities can be manipulated manually and/or automatically, via anomaly-based threat level analyzers as outlined in [41] (and as described later). Furthermore, the threat levels for the various entities can all impact single users. To coordinate all the policies into a single layer, we propose the *Threat Level Control layer*. For instance, imagine a domain threat level policy similar to the above user threat level example, where the penalty affects all users within the domain. Thus if a user has threat level 0, but the administrator has raised the domain threat level to 1, the threat level control layer will reflect that the user's clearance has been penalized by 1 security level.

4.2.1 Rollback-access

This section explains the rollback access mechanism notation outlined in [12] For resources, each access action α is assigned an *access threat threshold value* ($\theta(\alpha)$), which denotes the highest threat level at which α is authorized. When the threat level is elevated to θ^+, $TM^{auth}_{\theta+}$ is invoked and rollback-access (RA) is triggered. Current actions that are not authorized under $TM^{auth}_{\theta+}$ get suspended (termed "rollback: withdrawal mode"), and a record of their partially executed state is temporarily stored. The record(s) for α are retrieved when the threat level lowers to θ from $\theta+$, allowing for the state of α to be restored. The key characteristics of the RA mechanism are that: (i) in withdrawal mode, α is suspended, (ii) the RA is transitory, (iii) the RA is segregated (from other rollbacks), (iv) and α can be restored when authorized by a lower threat level (under rollback: restore mode)

These characteristics are facilitated by the use of *information compartments* (IC), which exist for each threat level. Thus IC_θ exists for threat level θ. IC_θ is a memory block that stores the partial-execution states for actions that have been rolled back (due to threat level elevation). If the execution of action α is suspended because the threat level has elevated from θ to $\theta+$, then the record of its state is stored in $IC_{\theta(\alpha)}$. Intermediary information compartments are allowed, since several actions may be rolled back upon threat level elevation, allowing for RA segregation.

4.3 Threat Level Control Layer

The threat level control layer exists to collect the policies specified by the various threat levels for the network, in order to dynamically respond to perceived threats in a systematic

and consistent manner. Our approach provides adequate flexibility with existing MAC, DAC, and/or RBAC systems (such as RT [36, 21, 45]) TM systems. However other TM systems such as credential-based KeyNote [10] and SPKI/SDSI [42] require a significantly different approach, which we aim to address in future work.

The domain threat level influences a proposed Threat Level Control (TLC) layer that logically sits above the Mandatory Access Control (MAC) layer [7], which in turn sits above the Discretionary Access Control layer. Thus, should the domain threat level raise, previously accessible resources can be barred from users, as in [12]. Here the established rollback mechanism could take safety measures to ensure that the data is secured until the network behavior is restored to normal. Such rollback measures can involve access withdrawal via resource encryption or secure network deletion, access freezing, reduction to vanilla access, or simple resource permission changing. Real time access control, remains an ongoing challenge for the computer security field, and developing a robust, secure, and flexible architecture to support this with implementation details is also intended for future work.

4.3.1 Architecture Model

In decentralized multi-domain trust management networks, the discovery and verification of user credentials and network policies is an important problem. Distributing the updates for the TLC layer require similar solutions. Based on the TM architecture (see figure 4.1) outlined in [12, 13], the TLC should reside where the ACLs for the domain are located. The ACL and TLC are manipulated directly by the Sensor Agent(s). Then the Resource Custodian is responsible for all rollback actions.

To illustrate how our model can support the *need-to-know* principle [49], which is the foundation of many real world security-critical networks, we consider an example presented in [12]. In this example, the military provides maps containing information regarding insurgent locations, and dangerous areas (*e.g.*, the locations of improvised explosive devices (IEDs)), to non-governmental organizations (NGOs) in order to facilitate the coordination and cooperation in all facets of tactical operations. This cooperative strategy ensures the nonmilitary efforts are not caught up in these operations (which could result in civilian casualties). Meanwhile, it is equally important under the need-to-know principle the loss or leakage of these intelligence resources be prevented or mitigated to likewise avoid military or civilian casualties, and preserve the value of the intel. This scenario offers insight into how a TM system under our model would be established, and furthermore outlines the remaining details of our model that are discussed in later sections.

Suppose that an NGO client has such access, and repeatedly requests maps covering different operation areas within a short time-frame. Assume that this is undesirable, and that the military cooperation policy forbids any one client from amassing fresh intelligence in such a fashion. In the absence of an automatic trigger that would detect such a violation for this scenario, the security manager in our model [12, 13] can manually raise the threat level for the NGO domain and/or client to β, enacting the next TM_{β}^{auth}. To enact the policy that forbids the amassing of fresh intel at threat level β, TM_{β}^{auth} can restrict the quantity, frequency, and/or clearance allowed to access information that the system recognizes as being *fresh*.

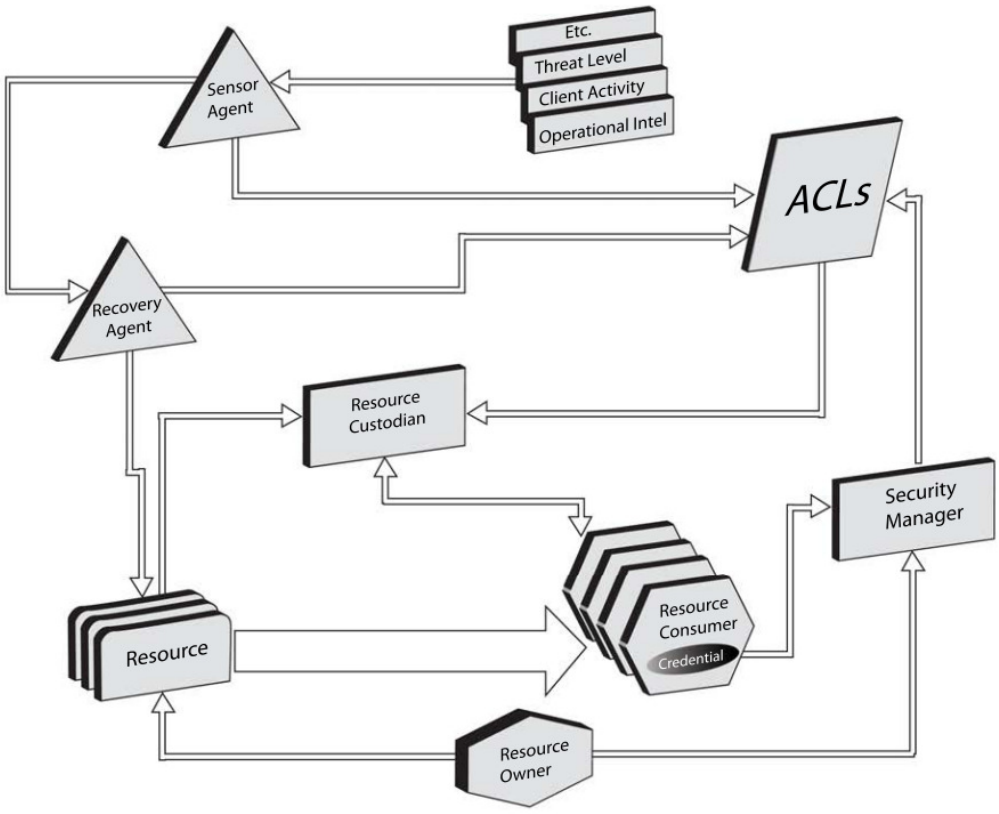

Figure 4.1: The R-TM Architecture Model from [12, 13].

Conversely, in the presence of an anomaly-based automatic trigger for this policy, the statistical analyzer would observe the properties of the resources requested by the NGO (domain or client) and the frequency of the requests. The analyzer is not necessarily limited to these metrics. Clearly, the implementation of the analyzer is important, and we will refer back to this example in later sections to explain such details. Yet regardless of implementation, the analyzer must be designed such that it predicts requests from the NGO for *fresh* intel as having low probabilities: such that individual or infrequent requests are normal; while frequent requests are regarded as anomalous. Otherwise, the analyzer cannot provably *realize* the policy. When the analyzer observes that the events are anomalous, it reports the anomaly to the sensor agent(s). The network sensor agent(s) observe the anomaly alarm(s) raised by the statistical analyzer and raise the threat level automatically (see figure 4.1).

Thus the TM will regard such requests as a need-to-know violation, and a potential threat. We stress: (a) the temporal nature of this violation need-to-know applies only to maps which detail current or future activities, or fresh information, and, (b) that the threat is only potential it is possible that the mission of the NGO client necessitates such actions.

CHAPTER 5

NETWORK PROFILING FOR ANOMALY DETECTION

Network profiling is well suited for limited-functionality networks that are designed for little past securely sharing and storing classified resources. Under the Internet Protocol Suite model, the vast majority of networks in widespread deployment support hundreds of application layer (layer 4) protocols. It is widely know that the difficulty of securing a network is directly proportional to the complexity of the network. Consequentially, due to complexity, it is computationally infeasible to develop an accurate, statistical-based analyzer for such application-rich networks. This chapter discusses techniques and algorithms to avoid combinatorical explosion (to ensure real-time performance) and the design and implementation of anomaly analyzers at the host and gateway levels.

5.1 Overview

Statistical analysis for network anomaly-detection can be performed via Markov chain or Bayesian inference. For a Markov chain implementation, the profile analyzes the set of all events and monitors the transitiions from event to event. For a Bayesian implementation, the profile will observe the frequency of events over the learning period. At the conclusion of the learning period, the events are assigned their *a priori* probability, all hypotheses are assigned their *prior probability*, and finally conditional probabilities of every event for all hypotheses are calculated.

For our purposes, the hypotheses for an anomaly detection Bayesian analyzer would be (i) The entity is behaving normally and (ii) the entity is behaving anomalous. A hidden Markov model can be established over these two states, associating the probability of each state with the hypotheses that (a) the entity is not malicious and (b) the entity is malicious [28]. Hidden Markov models (HMM) help to conceptualize the overall process; however HMM's have been shown to generate little additional accuracy and have high computational overhead to maintain [22].

In any implementation, avoiding combinatorical explosion is key in developing a normal behavior profile that can support a real-time application. Therefore, lumping, clustering, or categorizing events in order to reduce the size of the profile is essential to ensuring a real-time response. *Lumpability* was first introduced by Kemeny and Snell in 1960 as a

method of reducing the state space (be it finite or countably infinite) for *continuous-time* Markov chain applications [31]. Jernigan and Baran explain in [27] that: when the state space of a finite Markov chain can be partitioned, and each partition can be treated as a single state in a smaller chain that retains the Markov property — then the original chain is "lumpable" with respect to that partition (see theorem 6.3.2 in [31]). Kemeny and Snell formally state the necessary and sufficient condition for lumpability (in [31]): "*A Markov chain is lumpable with respect to the partition if and only if the distribution of the next subset is the same for every state in the present subset.*"(quoted from [27])

Thus, a continuous-time Markov chain $X = X_0, \ldots, X_n$ is lumpable with respect to partition $T = T_{[0]}, \ldots, T_{[K]}$ for K equivalence classes, if and only if, for any two classes $T_{[n]}, T_{[m]} \in T$, and for any two states $X_i, X_j \in T_{[n]}$:

$$p_{(X_i, T_{[m]})} = p_{(X_j, T_{[m]})} \tag{5.1}$$

where

$$p_{(X_i, T_{[m]})} = \sum_{X_a \in T_{[m]}} p_{(X_i, X_a)} \tag{5.2}$$

In other words, $p_{(X_i, T_{[m]})}$ refers to the one-step transitional probability of going from state X_i to all of the states in the equivalence class (or partition) $T_{[m]}$ [31, 26].

Clustering is the process of combining data into sets based on some sort of similarity. It is a common technique in statistical data analysis applications, and is used widely in many fields. Some of the most common fields that utilize this technique are: data mining, pattern recognition, image analsys, data compression, and machine learning. The anomaly-detection applications for our model are closely related to data mining, pattern recognition and machine learning fields.

Generally, clustering can be distance-based or conceptual-based. Distance-based algorithms require that the state space follow have comparable elements. Often-used distance measures for low-dimension data sets include Hamming, Euclidian, Manhattan, entropy distance, and Pearson correlation. There are numerous clustering algorithms and techniques for high-dimensional data sets. Some involve Monte Carlo sampling, simulated annealing, and population-based methods. In essence, some dimension(s) of the state space must have some structure that allows clustering of like (or nearby) data into sets. Conceptual-based algorithms group elements into clusters if those elements *define* a concept that is shared by all elements in the cluster. Therefore, the applicability of clustering algorithms depend on the structure of the data set (for distance-based), or the information expressed by the data (for conceptual-based).

5.2 Host-Based Profiling

Client behavior is captured by a random process that describes the temporal client activities and the services utilized by the client. The client analyzer can operate on the client end in a secure network, or at the network and gateway. In order to perform anomaly analysis at any level, it is necessary to develop a normal behavior profile for the monitored client.

To develop the normal behavior profile for the client, the analyzer monitors client behavior over a training period. In this way the analyzer captures on a regular basis the daily network activities of the client. Furthermore, should a detectable internal/ external attack against the client occur during training, this is omitted from the training data.

Host based statistical analyzers can utilize either Markov chain anomaly detection or Bayesian anomaly inference. Existing Bayesian anomaly detection models have had the advantage of being the easiest to implement and manage. However, Bayesian anomaly inference has been demonstrated to be less accurate than Markov Chain anomaly detection techniques [56].

The statistics of this profile are used to initialize the stochastic client analyzer. Then, a (possibly preassigned) probability threshold θ_{cl} : $0 \leq \theta_{cl} \leq 1$ is used to partition the client states into normal and anomalous. Again, if the client's behavior becomes too anomalous for the system to tolerate, the system will penalize the client. Here, the penalty is implemented in the proposed TLC layer, and penalizes the client's clearance level without changing the existing ACL. Later, the client's TLC penalty is reduced back to zero provided the client's behavior is estimated to be normal to the system for a sufficient temporal period. A confidence factor c_{cl} : $0 \leq c_{cl} \leq 1$,, is used to assess the stochastic prediction.

5.3 Domain/Gateway-Based Profiling

5.3.1 Overview

Domain behavior is defined by a random process whose states capture basic aspects of the temporal domain traffic, generated from inside (domain) Monitoring network behavior is coordinated among the network's gateway server(s).

To develop a normal behavior profile, the system is trained over a period of time, during which the domain traffic is monitored while performing under "normal behavior". Should a detectable internal/external attack occur during the training period, it is omitted from the training data. The statistics of this behavior are used to initialize a stochastic Markov domain analyzer. A (possibly preassigned) probability threshold θ_{do} : $0 \leq \theta_{do} \leq 1$ is used to partition the process states into normal and anomalous as in the case of the client analyzer. This is used to define the behavior of the next state of the domain. Likewise, a confidence factor c_{do} : $0 \leq c_{do} \leq 1$, which is a domain parameter, is used to assess the stochastic prediction.

5.3.2 Statistical Detectors

As an illustration suppose that the set of states of the domain traffic to be analyzed is S and that the statistics of the behavior profile defines a temporal Markov distribution μ on S. The states of S get partitioned by θ_d into: $S_{\mathsf{normal}} \uplus S_{\mathsf{anomalous}}$. If the (value of the) current state X_{t+1} belongs to $S_{\mathsf{anomalous}}$, then the μ-prediction of the domain analyzer is:

$$Pr(\mathsf{traffic\ is\ anomalous} \mid X_{t+1} = s \in S_{\mathsf{anomalous}}) = c_{do}.$$

If the value of X_{t+1} belongs to S_{normal} then,

$$Pr(\mathsf{traffic\ is\ normal} \mid X_{t+1} = s \in S_{\mathsf{normal}}) = 1.$$

In our variable-threat TM model, if the domain behavior becomes too anomalous, then it is regarded as a potential threat and the threat level is raised accordingly.

The domain threat level influences a proposed Threat Level Control (TLC) layer that sits above the Mandatory Access Control (MAC) layer [7] , which in turn sits above the Discretionary Access Control layer. Thus, should the threat level raise, previously accessible resources can be barred from users, as in [12]. Here the established rollback mechanism could take safety measures to ensure that the data is secured until the network behavior is restored to normal. Such rollback measures can involve access withdrawal via resource encryption or secure network deletion, access freezing, reduction to vanilla access, or simple resource permission changing.

Thus, typical network loads would constitute the normal behavior profile, which would become broad enough to support the wide array of network loads that happen on a daily basis. Network anomalies, such as DDOS-like traffic, or man-in-the-middle-like (*i.e.*, Ettercap) traffic constitute anomalous enough behavior that would not be sufficiently supported by the domain's normal behavior profile. In our system, the threshold for determining the tolerance of anomalous domain behavior is influenced by the current threat levels, the users' threat level, and the currently allocated resources.

We have not discussed the particular mechanisms that the Markov domain analyzer will use to profile domain traffic. For *reactive* profiling we can use traditional Intrusion Detection mechanisms. Such mechanisms have been extensively researched in the literature and there are several solutions that are very effective in detecting anomalous traffic—see *e.g.,* [40, 3, 4, 53].

Intrusion Detection is certainly a consideration that should impact on the Markov domain analyzer. However here we are more concerned with *real-time proactive* defense, which will address extreme threats (for which there may be no "signature" and which may not be distinguishable from authorized behavior), which can only be captured as anomalous or atypical behavior. In our model the domain analyzer monitors several metrics that are used to profile network traffic. These include among others: a) the application, b) the destination of the traffic, and c) the permissions required for the application (if any). Determining the service from the traffic flow has well established solutions in the field of real time traffic analysis [33], while b) and c) are trivial to determine (from the ethernet headers and the MAC/DAC policies).

Other resource-centric metrics that influence the analyzers involve policy restrictions include (but are not limited to): suspicious resource access pairings (that possibly violate need-to-know restrictions—see below for a more detailed discussion), average access time per resource (which would be used to detect a resource-spider/crawler attack), the type of resource, and the average access statistics of each type. These metrics are used to supplement the Markov analyzers in highlighting anomalous behavior by incorporating policy in the process, and serve to optimize the statistical specificity and sensitivity of the model.

CHAPTER 6

MOTIVATING SIMULATION

6.1 Simulation Overview

To simulate anomaly-based Threat Level analyzers, we developed a Python application that would build a Markov based normal behavior profile given seed traffic for the client. The analyzer would then observe outgoing network traffic for anomalies, given a set of predefined thresholds. The dimensions of the traffic involved: The destination network, the inferred application-layer protocol, and the packet size. Simple clustering algorithms were utilized to minimize the state space for each dimension of traffic. Here, categoriziation of destination IPs into networks became necessary, as avoiding clustering results in exponential memory growth to store the domain profile. Finally, a lagrange approximation was employed to ensure that the Markov chain profiles were nondecomposible.

With the normal profiles initialized, we simulated attack traffic flows that are typical of large file transfers. The attack flows were randomly addressed to the outside networks in the data set. Attack flows emulated data leakage attacks from unsophisicated malicious insiders, and sophisiticated malicious outsiders with remote access (who would not have physical access to the machine). All attacks were simulated in line with simulated normal traffic.

6.1.1 Simulation Specifics

The simulation was performed on the following platform:

Ubuntu 10.04 (32 bit)

Python 3.1.3 (released Nov 27 2010)

Intel Core 2 Duo - E6750 @ 2.66GHz, 2.66GHz

ASUS P5N-E SLI LGA 775 NVIDIA nForce 650i SLI ATX Intel Motherboard

4GB DDR2 PC2-5300

750GB HDD (ST375064 0AS SCSI)

On average, the simulation allowed for an outgoing datarate of 4.5MB per second. The Python simulation application is a sequential program, and conversion to a parallel or concurrent version would allow for higher datarates.

6.1.2 First Design

The first design for the thesis simulation involved a Markov chain based threat level analyzer which monitored client network activity. Order 2 Markov chains were used in the anomaly-detection of the client's outgoing traffic, and was performed on the clientside. Markov chains were used to analyze each dimension of the traffic, and thus normal behavior profiles were constructed to detect anomalies for each dimension. Markov TM agent takes as input the distribution μ_{cl} of the Markov client analyzer and the distribution μ_{do}

Domain anomalies were reported whenever the overall distribution of the outgoing network traffic was detected to be anomalous in regard to μ_{do}. Likewise, *client* anomalies were reported when the distribution of network application flows were detected as anomalous in regard to μ_{cl}. There are four cases to be considered—see Figure 6.2.

Cases	μ_{do}	μ_{cl}
1	1	1
2	1	c_{cl}
3	c_{do}	1
4	c_{do}	c_{cl}

Figure 6.1: Scenario 1: The cases reported to the Markov agent

In Case 1, when both analyzers report normal behavior the threat level is not raised, and a domain counter cnt_d (initially set to 0) and client counter cnt_c (initially set to 0) are each decreased by 1. We use two thresholds, one for each counter: t^d_{lower} and t^c_{lower} (a domain and client parameter, respectively). When the counter cnt_d reaches the threshold t^d_{lower}, the domain threat level is lowered (a variant would use different thresholds for each threat level). Similarly, when the counter cnt_c reaches the threshold t^c_{lower}, the client threat level is lowered, and in turn the clearance penalty of the client is lowered. Also, cnt_c and cnt_d are bounded below by the lowest t_{lower}.

In Case 2 the analyzers have reported only a client anomaly. The objective is to lower the domain threat level counter and raise the client threat level counter. The counter cnt_d is decremented by 1 and when it reaches t^c_{lower}, domain threat level is lowered by one. Similarly, cnt_c is incremented and when it reaches the threshold t^c_{raise} the client threat level is increased (by one).

In Case 3, the counter cnt_d is incremented by 1 and when it reaches the threshold t^d_{raise} the domain threat level is raised (by one). Similarly, the counter cnt_c is decremented and when the threshold is reached the client threat level is lowered. Finally, in Case 4 both analyzers report anomalous behavior. In this case the both counters are raised, and if the corresponding thresholds are reached, we use the approaches in Case 2,3.

6.1.3 Simulation Results

Initial results indicate that a base-case implementation of the client-based Markov chain traffic analyzer is moderately effective at detecting large malicious file transfers, given the normal behavior profile. With a anomaly detection threshold of $\theta = 5.00e-30$ and a

Markov chain order of 2, the analyzer detected approximately $34.24 - 44\%$ of simulated attacks, and had a mean false positive rate of 4.225%. Attacks are considered detected when they trigger the threat level to raise through anomaly detection reports handled by the Markov TM agent.

However, this system is fundamentally limited: it does not fully exploit the confidence levels c_{do} and c_{cl}. There are several ways in which the confidence level of the monitoring procedure can impact on the Markov process. One way would is to use the confidence levels c_{do} and c_{cl} to decrease (increase) the corresponding counters by fractional values. This, and similar considerations, is part of ongoing work into fine-tuning the decision making aspects of the Markov TM agent. The false alarm rate for the domain was significantly higher than the false alarm rate for the client analyzer.

6.1.4 Second Design

The second design for the thesis simulation also involved a Markov chain based threat level analyzer for the client. Markov chains were used in the anomaly-detection of the client, and were performed on the clientside.

Similarly, the Markov TM agent takes as input the distribution μ_{cl} of the Markov client analyzer and the distribution μ_{do}. There are four cases to be considered—see Figure 6.2.

Cases	μ_{do}	μ_{cl}
1	(1 - c_{do})	(1 - c_{cl})
2	(1 - c_{do})	c_{cl}
3	c_{do}	(1 - c_{cl})
4	c_{do}	c_{cl}

Figure 6.2: Scenario 2: The cases reported to the Markov agent

In Case 1, when both analyzers report normal behavior the threat level is not raised, and a domain counter cnt_d (initially 0) and client counter cnt_c (initially 0) are each decreased by (1 - c_{do}), and (1 - c_{cl}) respectively. Like before, there are two thresholds, one for each counter: t^d_{lower} and t^c_{lower} (a domain and client parameter, respectively). When the counter cnt_d reaches the threshold t^d_{lower}, the domain threat level is lowered. Similarly, when the counter cnt_c reaches the threshold t^c_{lower}, the client threat level is lowered, and in turn the clearance penalty of the client is lowered. Again, cnt_c and cnt_d are bounded below by the lowest t_{lower}.

In Case 2 the analyzers have reported only a client anomaly. The objective is to lower the domain threat level counter and raise the client threat level counter. The counter cnt_d is decremented by (1 - c_{do}) and when it reaches t^c_{lower}, domain threat level is lowered by one. Similarly, cnt_c is incremented by c_{cl}, and when it reaches the threshold t^c_{raise} the client threat level is increased (by one).

In Case 3, the counter cnt_d is incremented by c_{do}, and when it reaches the threshold t^d_{raise} the domain threat level is raised (by one). Similarly, the counter cnt_c is decremented by (1 - c_{cl}), and when the threshold is reached the client threat level is lowered. Finally,

in Case 4 both analyzers report anomalous behavior. In this case the both counters are appropriately raised by c_{do} and c_{cl}, and if the corresponding thresholds are reached, we use the approaches in Case 2,3.

6.1.5 Simulation Results

Initial results for this version indicate that a base-case implementation of the client-based Markov chain traffic analyzer is moderately effective at detecting large malicious file transfers, given the normal behavior profile. With a anomaly detection threshold of $\theta = 5.00e-30$ and a Markov chain order of 2, the analyzer detected approximately $43.33-50\%$ of simulated attacks, and had a false positive rate of 2.5155%. Initial simulation tests using higher Markov chain orders revealed lower false positive rates, clearly due to increased contextual awareness of the model. Attacks are considered detected when they trigger the threat level to raise through anomaly detection reports handled by the Markov TM agent. These preliminary results indicate that the the framework holds promise for TM networks. However, further testing remains to determine the effective bound on the Markov chain order for the anomaly analyzer, as all Markov chain applications yield diminishing returns on increased accuracy.

CHAPTER 7

CONCLUSION

7.1 Concluding Remarks

We have presented a new security application in the APECS framework to dynamically protect against loss, theft and leakage of mission-critical information. We demonstrated how existing solutions that address data loss, theft, and leakage of information fail to prevent sophisticated attacks, since they are designed to protect against careless users and very unsophisticated malicious insiders. This thesis outlined the flaws with existing approaches in similar areas and proposed a novel approach on policies and mechanisms that are more idealy suited for preventing and mitigating the theft, loss, and leakage of mission critical information.

This thesis described the design, implementation, and analysis of *real-time statistical (Markov chain and Bayesian) analyzers* for network anomly detection to trigger *novel policy-based* rollback-access mechanisms (extending the work of [12, 13, 41]). The rollback-access mechanisms in the APCES framework dynamically mitigates the risk of security-critical file distribution by rolling back the granted access to the aforementioned files upon detecting that the user is a perceived threat. Finally, this thesis presented some experimental results, which illustrate the potential for the APECS framework.

7.2 Publications Related to this Thesis

The following papers were accepted/published with the following conferences:

- *(Published:)* W. O. Redwood and M. Burmester. *Markov anomaly modeling for Trust Management in variable threat environments.* In ACM-SE 2010: Proceedings of the 48th annual Southeast regional conference, New York, NY, USA, 2010. ACM.

- *(Accepted:)* W. O. Redwood and M. Burmester. *Markov Reputation In Variable Threat Level Trust Management Systems For Resource Allocation Risk Assessment.* IEEE SoutheastCon 2010.

7.3 Future Work

The proposed APECS framework identifies a new realm for computer security research. Much of the hardware and software required for the architecture proposed here remains to be seen in known developments. Operating system redesign is required for customized file I/O system calls to perform rollback-access for mission-critical files. New security architectures and embedded systems can be designed to support the various aspects of the APECS framework.

7.3.1 Compatability with other forms of TM systems

The described APECS framework had been demonstrated to be compatible with MAC, DAC, and RBAC based TM systems. In the future, we intend to extend this framework to demonstrate how it can be applied to important alternative TM systems, such as KeyNote, RT, SDSI.

7.3.2 Real-time Application-Identifying Traffic Analyzers

A method that quickly and reliably identifies network applications in real-time from simple traffic analysis, would enable the Markov chain based anomaly analyzers to develop profiles on a per-application basis. This could significantly reduce false alarms, and potentially increase anomaly detection accuracy.

7.3.3 Inter-Domain Threat Level Policy Propagation

A method that propagates threat level policy updates to threat level control layers, reliably in other domains. This would allow for a more expressive trust management framework, as the domain would be able to incorporate in the respective threat levels for other trusted domains.

7.3.4 Secure Analyzer Communication Protocol

In this section we describe a secure network protocol for passing messages and commands over a network between Markov TM agents. A suitable protocol would employ standard secure protocol mecahanisms such as: public/private key encryption, salt, last-message hash, and keep-alive control messages. The goal of the protocol is to ensure that TLC updates are atomic, and that physical tampering of the APECS system hardware is automatically detected.

BIBLIOGRAPHY

[1] Wikileaks Iraq War Diary. http://wikileaks.org/, April 2010.

[2] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. In *Advances in Cryptology - CRYPTO '91: 11th Annual International Cryptology Conference*, pages 1–23. LNCS 576, 1991.

[3] Alder, Carter R., E. F., J. Foster, M. R. Jonkman, and M. Poor. *Snort IDS and IPS Toolkit*. Syngress Publishing, 2007.

[4] R. Anderson. *Security Engineering*. Wiley, New York, 2001.

[5] A. W. Appel and E. W. Felten. Proof-carrying authentication. In *6th ACM conference on Computer and Communications Security*. ACM, 1999.

[6] D. Balfanz, D. Dean, and M. Spreitzer. A security infrastructure for distributed Java applications. In *21st IEEE Symposium on Security and Privacy*, 2000.

[7] David Elliott Bell and Leonard J. La Padula. Secure Computer Systems: Mathematical Foundations. Technical report, MITRE Corporation, Bedford, Mass, 1973. MTR-2547.

[8] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatain. Traffic classification on the fly. *ACM SIGCOMM Computer Communication Review*, 2006.

[9] Simon Biles. Detecting the Unknown with Snort and the Statistical Packet Anomaly Detection Engine (SPADE). Technical report, Computer Security Online Ltd. http://www.computersecurityonline.com/spade/SPADE.pdf.

[10] M. Blaze, J. Feigenbaum, and A. D. Keromytis. *KeyNote: Trust management for public-key infrastructures*. 1999.

[11] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 164 –173, may. 1996.

[12] Mike Burmester, Prasanta Das, Martin Edwards, and Alec Yasinsac. Multi-domain Trust Management in Variable Threat Environments Using rollback-access. In *Proc. Military Communications Conference (MILCOM 2008)*. IEEE, 2008.

[13] Mike Burmester, Prasanta Das, Martin Edwards, and Alec Yasinsac. Multi-domain Trust Management in Variable Threat Environments —a user-centric model. In *Proc. Military Communications Conference (MILCOM 2009)*. IEEE, 2009.

[14] Yu-Shu Chen and Yi-Ming Chen. Combining incremental Hidden Markov Model and Adaboost algorithm for anomaly intrusion detection. In *CSI-KDD '09: Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*, pages 3–9, New York, NY, USA, 2009. ACM.

[15] Michael Chertoff. *Remarks by Homeland Security Secretary Michael Chertoff to the 2008 RSA Conference*. Department of Homeland Security, 2008.

[16] Yang-Hua Chu, Joan Feigenbaum, Brian LaMacchia, Paul Resnick, and Martin Strauss. REFEREE: trust management for Web applications. *Computer Networks and ISDN Systems*, 29(8-13):953 – 964, 1997. Papers from the Sixth International World Wide Web Conference.

[17] Fred Cohen. Computer Viruses: Theory and Experiments. In *7th DOD/NBS Computer Security Conference*, September 24-26 1984.

[18] D.E. Denning. An Intrusion-Detection Model. In *IEEE Transactions on Software Engineering*, volume 13, Issue:2, pages 222–232, Februrary 1987.

[19] D Endler. Intrusion detection Applying machine learning to Solaris audit data. In *Proceedings of the Computer Security Applications Conference*, 1998.

[20] W. Feller. *An Introduction to Probability Theory and its Applications*. John Wiley & Sons, 1968.

[21] D.F. Ferraiolo and D.R. Kuhn. Role Based Access Control. In *15th National Computer Security Conf*, pages 554–563, Oct 13-16. 1992.

[22] Stephanie Forrest, Steven Hofmeyr, and Anil Somayaji. The Evolution of System-Call Monitoring. In *ACSAC '08: Proceedings of the 2008 Annual Computer Security Applications Conference*, pages 418–430, Washington, DC, USA, 2008. IEEE Computer Society.

[23] Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. A Sense of Self for Unix Processes. In *SP '96: Proceedings of the 1996 IEEE Symposium on Security and Privacy*, page 120, Washington, DC, USA, 1996. IEEE Computer Society.

[24] Gary D. Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Markovian Analysis of Large Finite State Machines. *IEEE Transactions on CAD*, 15:1479–1493, 1996.

[25] Amir Herzberg, Yosi Mass, Joris Michaeli, Yiftach Ravid, and Dalit Naor. Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers. *Security and Privacy, IEEE Symposium on*, 0:0002, 2000.

[26] Jane Hillston. Compositional Markovian Modelling Using a Process Algebra. In *in the Proceedings of the Second International Workshop on Numerical Solution of Markov Chains: Computations with Markov Chains*. Kluwer Academic Press, 1995.

[27] Robert W. Jernigan and Robert H. Baran. Testing lumpability in Markov chains. *Statistics and Probability Letters*, 64(1):17 – 23, 2003.

[28] Shrijit S. Joshi and Vir V. Phoha. Investigating hidden Markov models capabilities in anomaly detection. In *ACM-SE 43: Proceedings of the 43rd annual Southeast regional conference*, pages 98–103, New York, NY, USA, 2005. ACM.

[29] Y. F. Jou, F. Gong, C. Sargor, S. F. Wu, H.C. Chang, and F. Wang. Design and Implementation of a Scalable Intrusion Detection System for the Protection of Network Infrastructure. In *DARPA Information Survivability Conference and Exposition*, 2000.

[30] Justin Lewis Balthrop. RIOT: A Responsive System for Mitigating Computer Network Epidemics and Attacks. Master's thesis, University of New Mexico, 2005.

[31] J. Kemeny and J. L. Snell. *Finite Markov Chains*. D. Van Nostrand, 1960.

[32] Micki Krause and Harold F. Tipton. *Handbook of Information Security Management*. CRC Press LLC, Auerbach Publications, 1997.

[33] Mihails Kulikovs and Ernests Petersons. Real-Time Traffic Analyzer for Measurement-Based Admission Control. *Advanced International Conference on Telecommunications*, 0:72–75, 2009.

[34] Wenke Lee, Salvatore J. Stolfo, and Philip K. Chan. Learning Patterns from Unix Process Execution Traces for Intrusion Detection. In *In AAAI Workshop on AI Approaches to Fraud Detection and Risk Management*, pages 50–56. AAAI Press, 1997.

[35] Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Trans. Inf. Syst. Secur.*, 6(1):128–171, 2003.

[36] Ninghui Li and John C. Mitchell. RT: A Role-based Trust-management Framework, 2003.

[37] P. G. Neumann and P. A. Porras. Experience with EMERALD to Date. In *1st SENIX Workshop on Intrusion Detection and Network Monitoring*, 1999.

[38] NIST. *Guide to Intrusion Detection and Prevention Systems (IDPS)* . Recommendations of the National Institute of Standards and Technology. Special Publication 800-94, Gaithersburg, MD. February 2007.

[39] E. Parzen. *Stochastic Processes*. Holden-Day, 1962.

[40] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23-24):2435 – 2463, 1999.

[41] W. O. Redwood and M. Burmester. Markov anomaly modeling for Trust Management in variable threat environments. In *ACM-SE 2010: Proceedings of the 48th annual Southeast regional conference*, New York, NY, USA, 2010. ACM.

[42] R. L. Rivest and B. Lampson. *SDSI A simple distributed security infrastructure.* http://theory.lcs.mit.edu/ cis/sdsi.html., 1996.

[43] S. Gorman. Electricity Grid in U.S. Penetrated By Spies. http://online.wsj.com/article/SB123914805204099085.html, April 2009.

[44] S. Gorman and A. Cole and Y Dreazen. Computer Spies Breach Fighter-Jet Project. http://online.wsj.com/article/SB124027491029837401.html, April 2009.

[45] R. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-Based Access Control Models. In *IEEE Computer (IEEE Press) 29*, pages 38–47, August 1996.

[46] S. Scott. *A Bayesian paradigm for designing intrusion detection systems.* Computational Statistics and Data Analysis, 2003.

[47] Securosis. Understanding and Selecting a Data Loss Prevention Solution. Technical report, Websense, October 21 2010.

[48] E. H. Spafford. Virus. *Encyclopedia of Software Engineering*, 1994.

[49] N. S. I. TCSEC. Trusted Computer System Evaluation Criteria. Technical report, DoD 5200.28-STD, 1983.

[50] Status Of This, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. 2007.

[51] Yuji Waizumi, Yuya Tsukabe, Hiroshi Tsunoda, and Yoshiaki Nemoto. Network Application Identification Based on Communication Characteristics of Application Messages. *Issue 60*, pages 754–759, December 2009.

[52] Stephen Weeks. Understanding Trust Management Systems. In *SP '01: Proceedings of the 2001 IEEE Symposium on Security and Privacy*, page 94, Washington, DC, USA, 2001. IEEE Computer Society.

[53] M. Whitman and M. Mattord. *Principles of Information Security.* Thomson, Canada, 2009.

[54] Yi Xie and Shun-Zheng Yu. A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors. *IEEE/ACM Trans. Netw.*, 17(1):54–65, 2009.

[55] Nong Ye. A Markov Chain Model of Temporal Behavior for Anomaly Detection. In *Proceedings of the 2000 IEEE Workshop on Information Assurance and Security*, pages 171–174. IEEE, 2000.

[56] Qingbo Yin, Rubo Zhang, and Xueyao Li. An new intrusion detection method based on linear prediction. In *InfoSecu '04: Proceedings of the 3rd international conference on Information security*, pages 160–165, New York, NY, USA, 2004. ACM.

[57] P. Zimmerman. *The Official PGP User's Guide.* MIT Press, Cambridge, 1995.

BIOGRAPHICAL SKETCH

William "Owen" Redwood was born on April 29, 1986, in Baltimore Maryland to Dr. William Raymond Redwood (Ph.D Biophysics, M.D Radiation Oncologist) and Dr. Nancy P. Dalos (M.D. Neruology). In the summer of 2008, he completed his Bachelors degree in Computer Science from the Georgia Institute of Technology. There, he also completed the *Multidisciplinary Certificate Program in Information Assurance*. Under the advisement of Dr. Mike Burmester, he obtained his Masters degree in the fall of 2010 after defending this thesis at the Florida State University Department of Computer Science. He enrolled for the Ph.D program at FSU for the spring 2011 semester.

Owen's research interests include network (static, ad-hoc, and mobile/vehicular) security, cryptography, and trust management systems.

He is currently living in Tallahassee with his brother Chris, who is also enrolled at FSU under the honors biochemistry undergraduate degree. He is considering graduate school enrollment at FSU to work on photochemistry under the advisement of Dr. Jack Saltiel, and has already published influential work as an undergraduate.