8-2010

Cyber Profiling for Insider Threat Detection

Akaninyene Walter Udoeyop
audoeyop@utk.edu

To the Graduate Council:

I am submitting herewith a thesis written by Akaninyene Walter Udoeyop entitled "Cyber Profiling for Insider Threat Detection." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Engineering.

Gregory D. Peterson, Major Professor

We have read this thesis and recommend its acceptance:

Itamar Arel, Hairong Qi

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Akaninyene Walter Udoeyop entitled "Cyber Profiling for Insider Threat Detection." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Engineering.

Gregory D. Peterson

Major Professor

We have read this thesis
and recommend its acceptance:

Itamar Arel

Hairong Qi

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

CYBER PROFILING FOR INSIDER THREAT DETECTION

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Akaninyene Walter Udoeyop

August 2010

ACKNOWLEDGEMENTS

I would first like to thank the Lord. Without Him, none of my accomplishments would have been possible. I would also like to thank Dr. Gregory D. Peterson for being an excellent advisor. Next, I would like to thank Dr. Itamar Arel and Dr. Hairong Qi for taking the time to serve on my committee. I would also like to thank Dr. Frederick T. Sheldon for being a great mentor. Last, but definitely not least, I would like to thank my family and friends for their love and support.

ABSTRACT

Cyber attacks against companies and organizations can result in high impact losses that include damaged credibility, exposed vulnerability, and financial losses. Until the 21st century, insiders were often overlooked as suspects for these attacks. The 2010 CERT Cyber Security Watch Survey attributes 26 percent of cyber crimes to insiders. Numerous real insider attack scenarios suggest that during, or directly before the attack, the insider begins to behave abnormally. We introduce a method to detect abnormal behavior by profiling users. We utilize the k-means and kernel density estimation algorithms to learn a user's normal behavior and establish normal user profiles based on behavioral data. We then compare user behavior against the normal profiles to identify abnormal patterns of behavior.

TABLE OF CONTENTS

Chapter Page

CHAPTER I

Introduction

Motivation

Many companies and organizations, at some point, have knowingly or unknowingly been subject to a cyber attack [4, 9]. Many associate cyber attackers with entities that exist on the outside of a company or an organizational infrastructure hacking and breaking into information systems to execute these cyber attacks. Others associate cyber attacks with viruses or system/network intrusion. Until the past decade however, insiders were often overlooked as potential threats to commit cyber attack [58].

Although security policies and access control policies aim to prevent organizations from known threats, individuals that are trusted to follow these policies do not at times. When security and access control policies are not followed, it typically exposes organizations to both external and internal cyber threats. Although the majority of cyber attacks stem from external entities, insider attacks are often more damaging and costly due to the knowledge of and access to information systems [4]. More than a quarter of cyber attacks were traced back to insiders in 2009. The intricacies surrounding insider threat are more complex than those dealing with external entities. This is because internal cyber attacks do not always occur as a direct result of a breach of security or access policy. Some internal attacks occur without a breach of any security or access policy.

The 2010 Cyber Security Watch Survey conducted by CERT and the U.S Secret Service attributes 26% of the cyber security events in 2009 to insiders [4]. The 2009 Verizon Data Breach Investigations Report shows that insiders cause 20% of data breaches. Keep in mind that some cyber attacks go undetected. The Verizon report also shows that 32% of these attacks were caused by business partners [9]. We will discuss later why business partners and the concept of insiders are not always mutually exclusive in the realm of insider threat.

In this thesis, we aim to develop a method to identify abnormal cyber behavior by establishing user profiles based on cyber data that is collected from the user while active

on a computer. To do this, we implement supervised and unsupervised learning algorithms to establish "normal" behavioral profiles for users. We then monitor users for anomalies in their behavior patterns. It is important to state that our goal is to identify these abnormalities. Categorizing a set of behavior as suspicious or malicious is subject to interpretation and outside of the scope of this thesis. For example, we can identify that a user is deleting an abnormally large amount of files from a directory that they do not typically access. Although this is abnormal, declaring this behavior as a suspicious act or malfeasance includes additional variables that are based on the terms of the security policy; this can be explored as future work. This work will benefit, not only insider threat research, but also cyber behavioral research in that it provides a means to model user behavior in a way that we can recognize normal behavioral tendencies.

Insider Threat

The Insider

Many organizations such as CERT and RAND dedicate research to insider threat and the insider problem [16]. The definition of an insider however lacks consistency throughout the cyber security community and specifically insider threat research. Most studies elect a binary approach to defining an insider, claiming that an insider must fall within a definable parameter such as an employment category. The assumptions surrounding this approach however, become somewhat vague and uncertain with the increasing use of outsourcing, mobile computing, contractors, and business partners. In this thesis, we adopt a lattice approach that defines the insider based on access [3]. We define an insider as a trusted entity that is given the power to violate a security policy. The insider is determined in reference to an established security policy. Insider threat lies in the access and ability to:

1. Violate a security policy using legitimate access, or
2. Violate an access control policy by obtaining unauthorized access

The Insider Problem

The intricacies surrounding insider threat are more complex than those dealing with external cyber threats because of the target demographic. This is partially due to issues dealing with trust, privacy, and ethics, which we will discuss later. The *Insider Problem* can be summarized as the challenge of protecting an organization from internal cyber attacks. This problem is challenging in that it is difficult to predict and defend, but has the potential to be very damaging.

Ethics and Trust

One of the main challenges within the insider problem is trust. In order for an individual to be considered an insider, they must be granted a level of trust. This trust can be manifested in the form of a computer account, access to a restricted room, access to a system resource, or knowledge of sensitive information. In granting any form of trust to this individual, an organization is also granting this individual the power to abuse that trust.

We acknowledge the possibility that an insider has the ability to abuse the trust relationship between themselves and an organization. But we ask ourselves, what is a reasonable approach to mitigating insider threat? Most approaches explore monitoring the insiders. How can we do this in a way that is revealing, yet not unreasonably intrusive? For example, if a malicious insider is identified, it is common to investigate the insider's email account. In an attempt to prevent an insider attack, is it reasonable to peruse all insiders' email frequently while seeking malfeasance? Is that overly intrusive? This thesis does not answer these questions, but state that these are pertinent issues that exist within the context of monitoring and deep analysis of human activity and behavior.

Cyber Attacks

Cyber attacks occur in any combination of three forms: exfiltration, data corruption, and denial of service. These attacks can result in high impact losses for organizations including damaged credibility, exposed vulnerability, and financial losses. To gain a better understanding of what occurs during a cyber attack, we will take a closer look at each form of a cyber attack. We will then examine real insider attack cases,

analyzing how the attack took place and what occurred on the systems used during the attacks.

Exfiltration

Data theft, IP theft, or any purposeful extrusion of data is a form of an exfiltration attack. This attack usually occurs as a result of insiders collaborating with competitors and adversaries to an organization [9]. The insiders typically provide sensitive or confidential information to these adversaries for a form of compensation that is usually monetary.

The case of career FBI agent Robert Hanssen [59] is among the most notorious and damaging exfiltration and espionage cases in US history. As an agent for Soviet intelligence services, Hanssen downloaded reams of highly classified information to encrypted data storage devices and a laptop. He then used the laptop to communicate with Soviet intelligence officers and to transfer the information that he obtained. This attack resulted in Hanssen revealing a large amount of highly sensitive security information to the Soviet Union. In order to obtain the information, Hansen accessed the FBI automated records system. His queries on the system during the exfiltration attack often exceeded his need-to-know. As an authorized user of the databases that he exploited, Hanssen was able to commit the attack without any breach of the access control system in place. However, his abnormal search patterns, if monitored, could have raised suspicion to his malfeasance.

A similar example of an exfiltration attack can be seen in the case of 16 year CIA employee, Harold Nicholson [16]. He worked as an instructor in a CIA special training center from 1994 to 1996. During his time with the CIA, Nicholson passed a wide range of highly classified information to Russia. He obtained the information by hacking into the CIA's enterprise computer system and performing a wide range of queries on the CIA's databases. The queries exceeded his need to know, including US intelligence data on Chechnya, which he sold to Russian handlers. The CIA called in the FBI to investigate Nicholson after he failed a routine polygraph test. They found that he had transferred an extremely large amount of Top-Secret files from CIA computers onto

encrypted disks, rolls of film, and his PC hard drive. The abnormally large amount of files exported from the CIA system could have raised suspicion. It is unclear if any abnormal system activity occurred during the process of Nicholson hacking onto the CIA system.

Data Corruption

A data corruption attack is defined as an unwarranted change to authentic information or data. This includes not only modification, but also the deletion of information. This attack is usually evident in cases of cyber fraud or attempts to damage data.

An example of such an attack can be observed in the case of software developer Chris Harn [16], who was responsible for monitoring the computer network and servers at Autotone Systems. These servers are used to automate day-to-day betting. Harn abused his super-user privileges on Autotone's systems when he altered bets placed by his fraternity brothers for more than $3 million of illicit earnings. After altering the bets, he modified the system's audit trail to cover his tracks. Harn's system privileges allowed him to commit these actions. However, bets and audit logs were very rarely modified on Autotone's system. If monitored, these modifications could have been flagged as abnormal.

Another example of a data corruption attack is observed in the case of Marie Lupe Cooley [16], an administrative assistant of Steven E. Hutchins Architects. Assuming that she would be fired from her job, Cooley vengefully entered the architecture firm's premises at 11:00pm on a Sunday and used her own account credentials to log onto a server that had years of architectural drawings valued at $2.5 million. She then proceeded to delete them all. Cooley did not attempt to cover her tracks and was soon arrested. Cooley's system privileges allowed her to perform the actions that constituted the attack. However, the time that she accessed the server and the large amount of data that was deleted from the server could have been flagged as abnormal or suspicious.

Denial of Service

Any unwarranted attempt to make a computer resource unavailable to its intended users is a denial of service attack. This attack can be executed in a variety of forms such as malicious code, malware installation, overloading a buffer, or any other action that results in disabling a system resource.

Some denial of service attacks however are unique in that they are not always realized immediately. These attacks are implemented by the attacker inserting malware, followed by a delay in the execution of the malware. The delay between insertion and execution of the malware can span minutes, days, months, or even years. These attacks are triggered or 'detonated' by an event or a point in time. This is why these attacks are called 'bombs'. These 'bombs' are manifested in 2 forms. The first of these is a logic bomb, which is triggered by a system event. Logic bombs lay dormant and do not execute until a specific system event occurs. Unlike the logic bomb, the second type of 'bomb', called a time bomb, is triggered by a specific moment in time. These attacks are not triggered by an event. Rather, they lay dormant until a period of time predetermined by the attacker.

William Shae, a programming manager of Bay Area Credit Services, Inc., was a software programmer and managed the firm's financial computer network [16]. He was notified of adverse employment issues towards the end of 2002 and placed on a performance improvement plan in January 2003. Soon after, Shae planted a "time bomb" on the firm's network, and then proceeded to delete audit logs that could potentially reveal his activity. He was terminated on January 17th, 2003 when he didn't show up for work. The malicious code from the time bomb executed near the end of January, resulting in 50,000 financial records being either deleted or modified. This also led to a disruption of computer network's functionality. The total financial losses were estimated to exceed $100,000. Like previous attackers, Shea was authorized within the system to perform the actions that constituted the Denial of Service attack. However, there were events that could have warranted suspicion such as the deletion of audit file modifications.

Abnormal Activity

Many insider attacks similar to these real examples of exfiltration, data corruption and denial of service attacks occur on a daily basis and cause considerable damage and financial loss. We observe abnormal behavior on the part of the attackers in these examples. This abnormal behavior typically occurred during or before the attack. Hansen's abnormal search patterns, Nicholson's abnormally large data transfers, Harn's data and audit record modification, Cooley's late night access and deletion of $2.5 million worth of architectural drawings, and Shae's deletion of audit data are all abnormal actions that occurred during or before the attacks. A system that monitors typical behavior on a computer system could have identified these abnormalities and detected the attacks. This premise combined with the examples above and many more examples of real insider attacks are the motivation for this thesis.

CHAPTER II
Stakeholders

This chapter highlights concerns amongst various industries with regard to insider threat. This should give some insight to the scope of the insider problem. This chapter also looks at contributions to insider threat research, data, and mitigation approaches.

National Security Sector

Since its inception, the United States national security sector has dedicated counter intelligence efforts focused towards insider threat [26]. The scope of these efforts includes and extends beyond insider threats to IT systems. The main concerns of the national security sector includes the introduction of malicious code, unauthorized access to data, exfiltration of sensitive data, alterations in system activity, and alterations to system topology. Their concerns are mainly focused on denial of service and exfiltration attacks.

The national security sector utilizes counterintelligence (CI) and Internal Affairs programs as the means to protect against foreign intelligence and security services (FISS). This sector includes Defense Information Systems Agency, US Joint Forces Command (USJFCOM), and US Strategic Command (USSTRATCOM), as well as the intelligence and CI communities who all expend effort and strategies to monitor, detect, prevent, and recover from insider threats.

Critical Infrastructure Sectors

President George W. Bush's 2006 National Infrastructure Protection Plan (NIPP) stated that critical infrastructures include "the assets, systems, and networks, whether physical or virtual, so vital to the United States that their incapacitation or destruction would have a debilitating effect on security, national economic security, public health or safety, or any combination thereof [27, 28]." The National Strategy for Homeland Security identifies the specific entities that encompass critical infrastructures as agriculture and food, water, public health, emergency services, institutions of

government, defense industrial base, information technology, telecommunications, energy, transportation, banking and finance, the chemical industry, and the postal and shipping industries.

Insider threat is a growing concern within each of these industries because of their importance and the devastation that an attack could cause. Much of the nation's infrastructure has been, or is in the process of being upgraded and retrofitted with connections to a local area network (LAN) or wide area networks (WAN). This increased accessibility to the Internet also increases the number of attack vectors available to both external attackers and the internal attackers that we refer to as malicious insiders.

Due to the importance of this sector, there have been a variety of research activities and government sponsored studies conducted by the USSS, the Department of the Treasury, and selected financial and industry groups to document and investigate insider threats within the critical infrastructures sector [61].

Non-Critical Infrastructure Sectors

Insider threat is a concern for a number of sectors that are not involved in national security or critical infrastructure. Since 2006, The Department of the Interior's Office of Inspector General has focused on the Department's information system vulnerabilities to the insider threat, publishing insider threat evaluation reports for three Interior organizations: it's National Business Center ("Trusted Insider Threat Evaluation Report for the National Business Center, " August 2006), the Bureau of Land Management (BLM) ("BLM Trusted Insider Threat Evaluation"), and the Minerals Management Service ("Trusted Insider Threat Evaluation Report for the Minerals Management Service") [16].

Law Enforcement

The challenge for law enforcement regarding insider threats is complex in that they are tasked with not only identifying malicious insiders in other industries and organizations, but also protecting against potential insider attackers within their own

organizations [29]. The primary federal law enforcement agencies that investigate cybercrimes, including those involving malicious insiders, are the FBI, USSS, Immigration and Customs Enforcement, the Postal Inspection Service, and the Bureau of Alcohol, Tobacco, and Firearms. The Internet Crime Complaint Center, a partnership between the FBI and the National White Collar Crime Center, acts as a clearinghouse to receive, develop, and refer complaints about cybercrime to the appropriate investigative agency.

Law enforcement also compiles information and statistics on cyber crime and losses attributed to insider threat. Examples of these efforts can be seen in the studies and surveys below.

In a March 2006 IBM survey of 600 US Chief Information Officers regarding their views on cybercrime, three-quarters of the respondents stated that threats to corporate security were originating inside their own organizations. The findings published by IBM indicated a deep wariness about insider threats as well as attacks arising from outside the company.

Surveys conducted collaboratively by the Computer Security Institute (CSI) and the FBI reveal that insider threats have gradually become recognized as discrete forms of incidents in computer security. Polling more than 700 computer security practitioners in US corporations, government agencies, financial institutions, medical institutions, and universities, the 2005 survey showed that 46% of respondents had experienced insider security incidents at least 1–5 times that year. 7% of respondents stated they had experienced at least 6–10 incidents and 3% stated that they experienced at least ten [30].

The 2006 CSI/FBI survey indicated an increase of both insider incidents and awareness by victim organizations. The 2006 survey, based on 616 responses from the target group of the previous year, captured this growing awareness by shifting from asking complex questions about the frequency and source of security incidents to simply asking respondents to estimate the frequency of insider vs. outsider attacks. 39% of the respondents indicated that their organizations incurred losses greater than 20% to insiders. 7% thought that insiders accounted for more than 80% of losses [31].

A 2007 survey showed that insider abuse of network access (59% of respondents) or email (such as trafficking in pornography or pirated software, 52% of respondents) overtook virus incidents as the most prevalent security problem according to the annual CSI/FBI survey [62].

Other security incidents documented in the survey, including denial of service, laptop/mobile, theft, telecom fraud, unauthorized access to information, virus, financial fraud, system penetration, sabotage, theft of proprietary info, abuse of wireless, network, website defacement, and misuse of public information fall under the three forms of insider attacks discussed in Chapter 1. Insider web access abuses have become the most prominent security problem since 2007.

In October 2007, the Center for Identity Management and Information Protection (CIMIP) at Utica College published a DOJ-funded study that analyzed over 500 US identity theft cases closed between 2002 and 2006. In the study, insiders defined as "employees of entities housing the identity information/documents stolen", were found to be the perpetrators in almost 35% of the cases.

According to an article in the British magazine *Computing*, the City of London police estimates that one third of its cases now have an insider element to them [60].

The concern about insider threats in the private sector, especially in banking and finance, is particularly high in the UK. The Department of Trade and Industry (the UK counterpart to the US Department of Commerce) contracts PricewaterhouseCoopers to publish an annual survey that includes cyber security topics [32]. An entire section of the survey is devoted to statistics on insider misuse of information systems in UK businesses.

The DOJ Computer Crime and Intellectual Property Section (CCIPS) implements national strategies to counter computer and intellectual property crimes around the world [33]. These crimes include cyber intrusion, data theft, and cyber attacks on critical information systems. CCIPS investigates and prosecutes computer crimes by working with other government agencies, the private sector, academic institutions, and foreign counterparts.

Academia

Most of the recorded cases of insider attacks at universities have resulted in the loss of students' personal information and the dissemination of privileged or sensitive university information. Although the majority of these attacks are committed by outsiders that gain access to the universities information system, these attacks are often aided by university insiders such as faculty, staff, and students who are either knowledgeable or unaware of the attack.

For example, a 2006 phishing attack that targeted Indiana University students was a direct result of university employees storing revealing information in the publicly accessible /etc/passwd file of a university computer system that was later attacked by the phishers [34].

Another insider data breach can be observed in the April 2007 case where the US Department of Education restricted access to the student loan database, which contains the personal financial information of millions of student aid applicants. These access restrictions were put in place after a number of concerns arose about authorized users abusing the information in the database. The concerns were justified when it was discovered that a Department of Education employee had sold $100,000 in stock in a student loan company based on information he had obtained from the database. In addition, in an investigation by New York's attorney general [35], private lenders were found to have used the database to find potential borrowers. Insider threats to educational institutions affect more than personally identifying information. In March 2005, applicants to top-tier business colleges and universities learned of a procedure that allowed them to determine the status of their applications before the schools sent out their decision letters. While the information thus disclosed could hardly be considered sensitive, the fact that the attack was possible causes concern regarding similar techniques making it possible to leak more sensitive information [63].

Educational institutions are also concerned about masqueraders, which includes outsiders who gain insider privileges. A major cause for concern is that students often return to campuses after a break or vacation with their computers in one of two states [36]:

1. Having been connected to the Internet during an entire break, the computers contained a wide variety of downloaded software, had been used to browse hundreds or thousands of websites, and in a number of cases, had been infected by viruses and worms.

2. Having not been connected to any network, the computers had not had any security patches or updates installed during the entire summer.

In 2003, George Washington University's email filters received 177,000 viruses on 22 August, the day when students returned from summer break; this number exceeded the average number of viruses filtered each month by a factor of ten. At MIT, more than 750 machines were infected by a virus called Blaster within three weeks [37].

After these attacks, many universities began to implement systems to quarantine and scan computers when they are connected to the school network, granting access privileges only to those that pass various security tests. In some cases, returning students are automatically placed in a restricted virtual local area network (VLAN), from which their systems are scanned for vulnerabilities and patched before being granted access to the university's LAN or WAN.

Universities typically do not implement insider threat or anti-insider countermeasures for their information systems. They rely instead on traditional IT security technologies and practices such as network access control systems, virus scanners, and patch management systems to protect their systems against both outsider and insider attacks.

The growing concern for insider threat has led universities to introduce policies in an attempt to deter and prevent insider attacks [38]. One such act is a law passed by a number of states to prevent university use of social security numbers as student identifiers.

Despite the lack of implementation of insider threat prevention systems, a number of academic institutions are engaged in insider threat research. Insider threat is

considered a significant enough problem by some professors to warrant lectures and entire courses devoted to the topic [64, 65].

The entities, organizations, and infrastructures discussed in this chapter are all affected by the insider threat problem. Each stakeholder has vulnerabilities that, if exploited, could lead to devastating consequences. Although specific concerns with regard to insider threat may differ for these stakeholders, means to identify and prevent insider attacks would be beneficial to them.

CHAPTER III
Related Work

In this chapter, we look at related work focused on insider threat identification, tracking, and prevention. The systems discussed below utilize a combination of access control, system call monitoring, anomaly detection, and resource usage monitoring. Access control systems grant or deny access to files and resources on a computer or information system based on user privileges. The system call monitoring systems discussed below monitor operating system commands in an attempt to identify suspicious sequences of activity. The systems that utilize anomaly detection use both data mining and reinforcement learning techniques to identify abnormal instances in data based on a set of training data. Resource usage monitoring systems for insider threat observe the state of the host machine's components such as the hard drive and the processor and watch for states that warrant suspicion.

Existing Insider Threat Systems

Mckinney et al. developed an approach to identify masqueraders by monitoring process resource usage [5]. This approach was implemented by establishing a "normal" user profile based on typical user behavior. The Naïve Bayes machine learning algorithm was used to establish these normal profiles. Behavior that deviates from these normal profiles is flagged as anomalous and potentially suspicious. This approach showed promising results with a true positive rate of 96.7% and a false positive rate of 0.4% from data collected over a three week period.

Qiao et al. introduced a framework to combat insider threat by implementing Subject-Verb-Object (SVO) monitors [39, 40, 41]. This approach monitors users and processes, capturing information such as login times, user name, privilege levels, process IDs, and security levels. This approach also logged access types such as reads, writes, executes, and the "Object" monitor recorded file attributes such as owner, size, path, and type. Data was accumulated and evaluated from a user metadata repository that contains

user data such as file system I/O and process activity. Threats were assessed based on the STRIDE/DREAD threat model which includes Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Escalation of Privilege, Damage Potential, Reproducibility, Exploitability, Affected Users, and Discoverability. Results presented from this approach only pertain to a mechanism that groups correlating alerts into a single alert. No results were provided that gave insight as to the performance of the overall system.

Shavlik et al. developed a masquerade detection system that monitored system performance data, event logs, application data, and user behavioral information such as the number of running programs and typing rate [42, 43]. The Winnow-based algorithm was utilized for anomaly detection and produced better detection rates than the Naïve Bayes algorithm. This system detects abnormal behavior based on behavior that is abnormal for the entire population. This approach detected anomalous behavior by running the logged activity of one user through the profile of another user. Detection rates were generally in the mid to upper 90th percentiles with rates as high as 97.4 percent.

Spitzner et al. utilized honeypots and honeytokens as a means to combat exfiltration insider attacks [44]. Honeypots are computer systems that are essentially set up to attract and "trap" people who attempt to illegitimately utilize a computer resource. Activity on a honeypot is assumed to be malicious, and is recorded. Honey tokens are digital data or information, such a record, a credit card number, or a file which are set up to attract people who attempt to illegitimately use the resource. No results were presented pertaining to the effectiveness of this approach.

Yu Chiueh developed the Display-Only File Server (DOFS), a system designed to combat information theft in 2004 [45]. In addition to a centralized file server, The DOFS verified whether the user had access to a queried resource. If a user has adequate access privileges, then the server releases an image of the file, not the file itself. This prevents individuals from modifying or corrupting the original file.

Pramanik introduced a security policy and framework similar to Digital Rights Management for insider threat [46]. This approach implements an access control

framework that requires user verification before the user grants access to read/write/update files. This system however had undesirable effects such as preventing users from opening files that they typically have access to.

Park et. al. modified a Role Based Access Control system to account for insider threat [47, 48] by implementing Composite Role-Based Monitoring (CRBM). In addition to granting access based on user privileges, this approach allows or denies access to files and resources based on the user's currently assigned task. The CRBM uses three role structures to define access privileges: Application, Operating System, and Organization. Access to resources is granted or denied based on these three role structures. The results from these experiments however did not support claims of an insider threat system that can prevent insider attacks more accurately than existing approaches and mechanisms.

Symonenko et al. introduced a role based approach in which user roles, context, and semantics are factored into insider threat [49, 50, 51]. This approach uses natural language processing to determine topics and areas of interest within documents. These are then compared user assigned topics and areas of interest based on the context of a user role. Support Vector Machines (SVMs) and ontologies are utilized to categorize document and user topics and interests. The documents were then grouped using clustering. The size of clusters and the distance from other clusters were used to analyze the threat.

Cathey et al. utilized clustering algorithms based on documents, search queries, and relevance ratings in an attempt to determine misuse based on need-to-know [52, 53]. In this approach, a collection of documents are sorted into clusters. A user profile is then established based on frequently accessed clusters. These document clusters are compared to those in the user's profile. Need-to –know or relevance was determined by typical user search queries and prevalent terms within the documents accessed by the user. Threats are determined by calculations of dissimilarities in a requested document and the user's profile. Results for this approach showed "misuse detection" rates between 90-100 percent with false positive rates between 12 and 15 percent.

Aleman-Meza, et al. has research efforts focused on data breaches within the context of insider threat [54]. This research employs statistical, NLP, and machine

learning techniques to measure the relevance of accessed documents to insiders' tasks. Their focus is in the national security and terrorism domain. The approach ranks documents as highly relevant, closely related, ambiguous, not relevant, and undeterminable based on semantic correlations. This short paper presented results that seem promising with a test set of 1000 documents.

Liu et al. introduced an approach for insider threat detection in which they analyzed system calls and system call information and attributes [55]. They utilized a k-nearest-neighbor machine learning algorithm to find anomalous behavior based on system calls. Tests showed this method to show a high false-positive rate for insider threat scenarios. However, this algorithm showed promising results for intrusion detection.

Anderson et al. proposed an architecture for detecting insider attacks by monitoring file and program events [56]. The architecture consisted of sensors and content-based routing. The sensors were essentially monitors deployed inside applications (e.g. MS Word and Internet Explorer) and the operating system. These sensors monitored events such as file opens, closes, saves, and keyboard inputs. Once captured, these events are sent by a content-based-routing system to a set of dynamic access control and analysis components. The access control components can interact with the program and prevent suspicious actions from being taken.

Nguyen et al. developed a system to detect suspicious insider behavior by monitoring system calls associated with the file system and process [57]. This approach analyzed explored commands initiated by the user and system calls initiated by the host system. Tests showed that file system calls initiated by the user showed a lot of variance and would not be ideal for creating behavioral profiles. The file system calls initiated by the operating system on the other hand, showed very predictable behavior, and was recommended by the authors for user behavioral profiles. This study did not provide test results that compared different user behavioral profiles. This would be beneficial for this study in that it would provide insight as to whether the predictable behavior is based on user activity or whether the operating system behaves unpredictably, but similarly for every user. The system calls attributed to processes gave promising results for modeling

user behavior. Of all the processes traced, 92% of them had a fixed list of files they access, many of them accessed files in patterns, and 92% had a fixed list of possible child processes. This approach successfully detected all buffer overflow attacks except for those in which the processes did not have a fixed number of child processes.

CHAPTER IV
APPROACH

In the examples of exfiltration, data corruption, and denial of service attacks in Chapter 2, we notice that several actions occurred before or during the insider attack that deviated from the norm. In this chapter, we introduce an approach to identify abnormal behavior by establishing user profiles that model normal behavioral patterns based on data acquired from the user. After acquiring the data from process, hardware, network, and the file system, we utilize the k-means [19] unsupervised learning algorithm to "learn" a user's normal behavior patterns. We then establish normal profiles by applying the Kernel Density Estimation [21] (KDE) algorithm.

It is important to keep in mind that this research intends to identify abnormal behavior. The interpretation of this behavior however is subject to further research. This thesis intends to show when a user that typically behaves a certain way is not behaving normally. For example, if a user typically only works on files within a certain set of directories and sub-directories, then suddenly begins to copy a large set of files from another subdirectory, this can be classified as abnormal behavior. Our research aims to identify such occurrences. However, the criteria to label this as suspicious or malicious is subject to further research in the area of cyber security policies. These criteria may also vary based on the organization.

Profiling

Behavior Set

Our objective is to ultimately categorize user behavior as either normal or abnormal. Normal behavior is identified in behavior patterns that fall within a set of recognized typical behaviors. Abnormal behavior is identified in behavior patterns that do not fall within a recognized set of typical behaviors. This idea is represented in Figure 1. We aim to identify and flag instances of abnormal behavior.

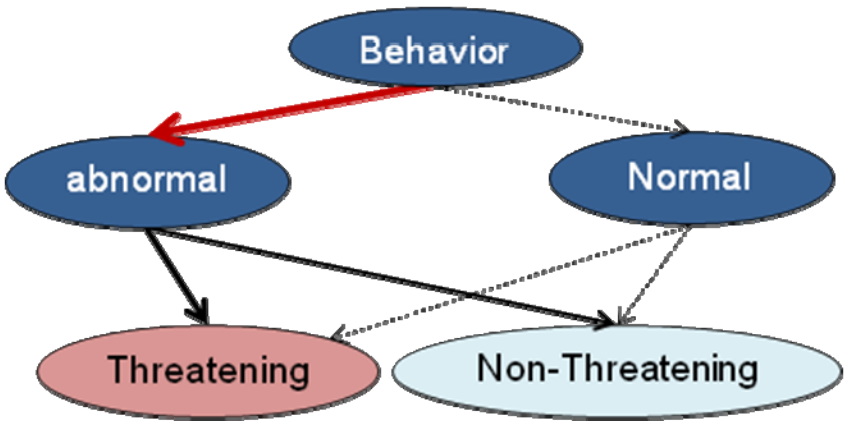


Figure 1: Decision tree modeling abnormal, normal, threatening, and non-threatening behavior.

We acknowledge the possibility of false alarms where behavior that is flagged as abnormal may actually be typical user behavior. We also acknowledge the possibility of missing alarms to occur where abnormal behavior may go undetected. This is because we identify normal behavior based on a set of training data. Therefore, the accuracy of categorized normal and abnormal behavior depends on how well the training data models normal user behavior.

Search Space

Let *B* be the set of all possible behaviors. This means, a combination of all possible system states and user actions. $\forall$ b ϵ B, b ϵ A if it is considered abnormal and b ϵ N if it is considered to be normal.

$A \cap N = \emptyset$

$A \cup N = B$

When we include a set of behaviors Q, which may be questionable, but are not able to be categorized as normal or abnormal, then,

$A \cup N \cup Q = B$

$A \cap N = A \cap Q = N \cap Q = \emptyset$

In the search for threatening and malicious behavior, we intend to remove normal behavior from the search space, effectively reducing the search space and the complexity of the search.

Assumptions

All of the test subjects are aware of monitoring software running in the background of their host machine. However, we assume that the test subjects are performing tasks that they typically would outside of the testing environment. It is possible that individuals may behave more cautiously or that test subject behavior may be more premeditated at times due to the knowledge of the monitoring software. A testing environment where the test subjects are unaware that their behavior is being monitored would be ideal, however, we were unable to establish such an environment due to ethical and privacy concerns.

Data sets from real cyber attacks are hard to find. This is because the organizations that are victim to these attacks are hesitant to release data and information. This may be due to apprehension that releasing this information may present an image of vulnerability for the organization, or that releasing this data may give other attackers a blueprint to perform a similar attack. However descriptions of various cyber attacks and knowledge of computer and information systems lead us to make assumptions about the behavior on systems utilized during an attack. These assumptions are typical throughout cyber security research due to the lack of real attack data, particularly research dealing with insider threat [5, 16, 66].

We also assume that the users did not tamper with the monitoring software and that the software ran continuously throughout the duration that they were logged onto the machine.

Cyber Data

Virtually every action by the user of a host machine triggers a system event or a change in the state of the system. Opening a file, deleting a file, visiting a website, closing a program, and even moving a mouse cursor are examples of some of these actions. These and many more actions trigger system events and/or state changes that we can pinpoint and track on a host machine. By monitoring the state of a host machine and system events over time, we can model a user's behavior on a host machine.

We utilized the Microsoft .NET Framework [67] to collect this data. The .NET Framework enables programming languages to include system event listeners that watch for system events. The .NET Framework also includes functions to query the state of various hardware and software components of a computer. We used C# along with the .NET Framework not only because the programming language has a wide array of sub-functions that make routine tasks simple to implement with less lines of code. C# also includes libraries with data structures and elements such as data grids to store and organize data. This is beneficial when dealing with large data sets.

Before discussing the components that we will monitor, we will clarify what we mean by the terms user and host. Some computer based research refers to the term *User* as the individual using a computer [5, 8]. However, the term *User* takes on added complexity in the realm of insider threat. Obviously, just because a person logs on to a computer with a set of credentials does not mean that the authenticated user on the system matches the physical person. Hackers and masqueraders often commit fraudulent authentications which result in scenarios where the individual using the system does not match the authenticated user. This is why we define the term *User* as the individual associated with a set of authentication credentials on a computer system. The *User* is recognized by their authentication credentials, not by their physical identity. This definition is useful in that a computer system can only identify a user by means of

authentication. A portion of our research aims to identify and track scenarios where a fraudulent authentication may occur.

The term *Host* or *Host Machine* refers to any computer, server, information system, or similar operating system based machine with a user authentication system, hardware (as opposed to a virtual machine), and network access.

Process Variables

A process is an instance of a computer program consisting of one or more threads that are executing simultaneously. A thread consists of an executable sequence of code. Processes can be invoked by a user or an operating system and have an associated unique identification number. We monitor the following attributes for every running process:

Privileged Processor Time – the ratio of time that a process utilizes the processor vs. the user time. This represents the percentage of the processor that is being utilized by a process.

Private Bytes – the current size, in bytes, of memory that a process has allocated that cannot be shared with other processes.

Thread Count – the number of active threads in a process. An instruction is the basic unit of execution in a processor, and a thread is the object that executes instructions. Every running process has at least one thread.

Creating Process – the parent process that created a specific process or set of processes.

We also collect the process identification number associated with each process. These variables show the programs and services utilized on a computer. They also give insight to the amount of system resources that each process uses. When monitored over time, these variables can model program, process, and thread activity on a computer. They can also show how extensively specific applications and services are utilized by the user.

Hardware Variables

The Microsoft .NET framework allows us to monitor the state of various hardware components. We track the following hardware components on the host:

CPU Usage – the percentage of the processor that is being utilized, and
Memory Usage – the amount of memory in use.

When tracked over time, these components allow us to establish a temporal model of the overall host machine hardware state.

Network Variables

Some examples of network activity include a user visiting a website, connecting to an FTP server, or using software that communicates with another host over a network connection. The IPGlobalProperties .NET library includes subroutines that access information about active network connections on a host machine. We monitor the following variables for every Transmission Control Protocol network connection:

Destination IP – the network address for the remote endpoint of a network connection,
Destination Port – the remote port on which a network connection is established,
Host IP – the network address for the local endpoint of a network connection**,** and
Host Port – the local port on which a network connection is established.

These variables represent instantiations of endpoint-to-endpoint network connections. When tracked over time, they allow us to establish a temporal model of endpoint-to-endpoint network activity on a host.

File System Variables

A computer stores and organizes files and data on a file system. Creating, deleting, or modifying files alter the state of the file system. The FileSystemWatcher

.NET class includes subroutines to monitor these file system events. We utilize a file system event listener that watches for the following file system events:

File Created – a new file is created on the file system of the host,
File Changed – a file is modified or resaved on the file system of the host,
File Renamed – the name of a file on the file system of the host is changed, and
File Deleted – an existing file is deleted from the file system of the host.

The filename, absolute path, and timestamp associated with each event are also tracked. These file system events may be triggered by the user, the operating system, or an active program. A collection of these file system events, over a period of time, model the file system activity on a host machine.

Learning Algorithms

Machine learning allows computational learning agents to establish or evolve assumptions based on empirical data. The focus of machine learning is to recognize complex patterns and make decisions based on the data and assumptions formed from previous data. We apply the k-means and the kernel density estimation algorithms to the cyber data to recognize patterns that represent a user's normal behavior.

Supervised vs. Unsupervised Learning Algorithms

Recognizing patterns in empirical data can be accomplished in several ways. One popular approach is to utilize machine learning techniques. Machine learning algorithms can be classified as supervised, unsupervised, or semi-supervised (which is a combination of supervised and unsupervised). Supervised learning relies on training data from which a predictor function is inferred. The goal of the predictor function is ultimately to classify or label new data based on pre-defined classes. Unsupervised learning algorithms, on the other hand, do not require training data. They model inferences based on the test data exclusively. We utilize a semi-supervised approach as we utilize a

combination of k-means, an unsupervised learning algorithm, and kernel density estimation, a supervised learning algorithm.

K-Means

The k-means clustering algorithm is an iterative method to partition a given dataset into a specified number of clusters [19, 20]. Given a set of data, the goal of this algorithm is to group each data value with a cluster of data values, then converge the cluster centroids to best represent correlations within the data points. A centroid is the average point or center of a cluster. The algorithm operates on a set of d-dimensional vectors $D = \{\mathbf{x_i} \mid i = 1, \ldots, N\}$, where $\mathbf{x_i} \in \Re^d$ denotes the i^{th} data point. The first phase of the algorithm is to select *k*, the number of clusters to associate with the data points. The next phase is to assign coordinates to these centroids. This can be accomplished in several ways. Some techniques for selecting these initial centroid locations include sampling at random from the dataset, clustering small subsets of the data, and perturbing the global mean of the data k times. Selecting the initial centroid locations is NP-hard and there is no known optimal method for this [23]. Perturbing the global mean of the data is optimal for data points with a relatively even density, which is not the case for this work. Setting the centroids as a result of clustering small subsets of data adds an immense amount of computational complexity and does not guarantee optimal results. Selecting centroids by randomly sampling the data set improves the chance that that each centroid converges with close locality to the data. It also requires a relatively low amount of computation when compared to other centroid selection techniques. This is why we chose to select the centroids by sampling at random from the data set.

The algorithm then iterates between these two steps until convergence:

Step 1: Data Assignment. Each data point is assigned to its closest centroid, with ties broken arbitrarily. This results in a partitioning of the data.

Step 2: Relocation of "means". Each cluster representative is relocated to the center (mean) of all data points assigned to it.

Convergence is achieved when the assignments of data points to clusters no longer change.

Kernel Density Estimation

In statistics, Kernel Density Estimation (KDE) is a technique used to estimate a probability density function (PDF) [21, 22]. The PDF infers a statistical distribution based on a set of data. After establishing the cluster centroids with the k-means algorithm, we apply KDEs to these centroids to produce a statistical distribution of these centroids. The probability density function (PDF) can be obtained by the following [24]:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i}K(\frac{x-x_i}{h_i})$$

where $x_1, x_2, ..., x_n \sim f$ is an distributed sample of a random variable, K is some Kernel, and h is a smoothing parameter.

In this thesis, we implement discrete KDEs, also known as Box KDEs, where the histogram produced by the KDE is separated into a finite number of sub-intervals [25]. Figure 2 shows an example of both discrete and continuous KDEs. We chose to utilize discrete KDEs due to the fact that continuous KDEs would require a probability calculation for every new data point. On the other hand, all probability calculations for discrete KDEs can be completed directly after the cluster centroids have been established. Although continuous KDEs are inherently more accurate than their discrete counterparts, it seemed more feasible to utilize discrete KDEs due to the high rate at which data is accumulated in this study.

The discrete KDE is essentially a histogram where we divide the intervals covered by the data values into sub-intervals based on their special density. An example of our implementation can be visualized in the discrete KDE in Figures 2. In this example, we include the following data points: 2.2, 2.4, 2.45, 2.5, 3.33, and 3.66. For every data point, we place on top of it a block of width 1, add them up, then normalize the result based on the highest sum.

Analysis

Our objective is to establish normal user profiles which are ultimately represented by KDE density distributions. We approach this in three phases: data acquisition, unsupervised cluster analysis, and kernel density estimation. We begin by acquiring the process, hardware, network and file system data described earlier. We store this data in log files and in memory using the Microsoft .NET framework data structure called Data Grids [29]. For visualization purposes, the data can then be plotted on two-dimensional graphs. Figure 3 is an example of the number of threads vs. parent processes.

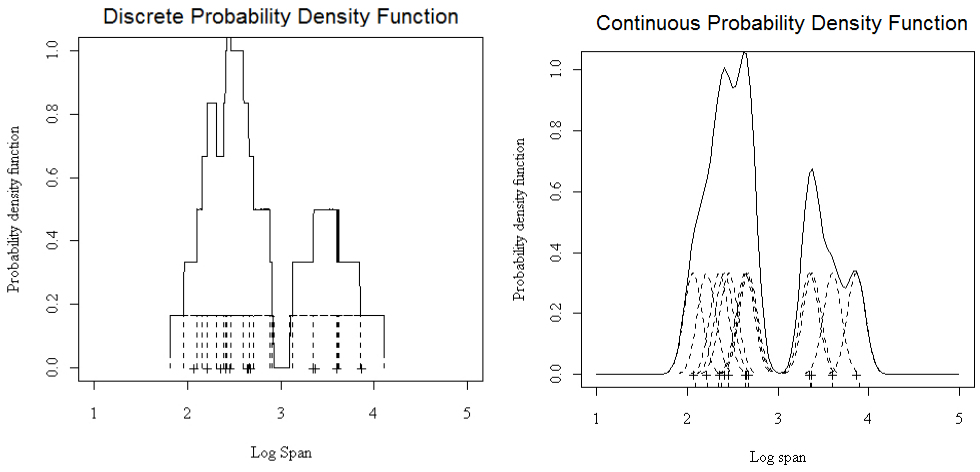


Figure 2: Continuous and Discrete Kernel Density Estimation

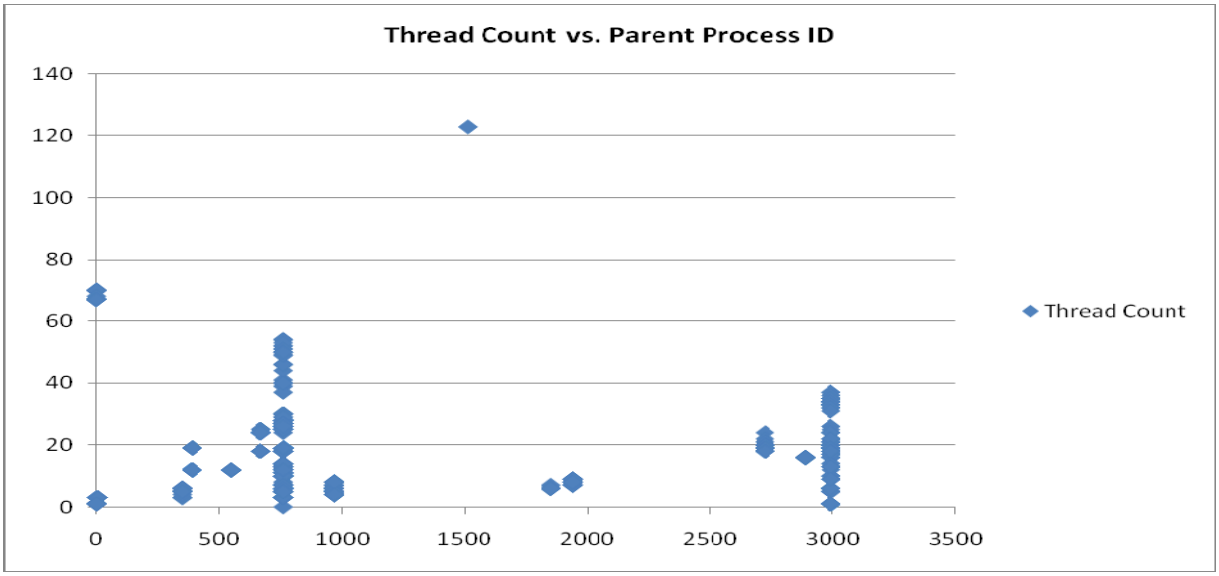


Figure 3: Number of Threads vs. Parent Process ID

We perform behavioral analysis on the following variables:

Number of File System Events vs. Directory – the number of file creations, modifications, and deletions, along with the files that are renamed based on the directory.

TCP Connections vs. IP Address – the amount of TCP connections established from the local host to a remote machine.

Number of Threads vs. Parent Process – the number of active threads running per process; each process is grouped by parent process.

Number of Handles vs. Process – the number of active handles per process. A handle is a token, typically a pointer that enables a program to access a resource, such as a library function.

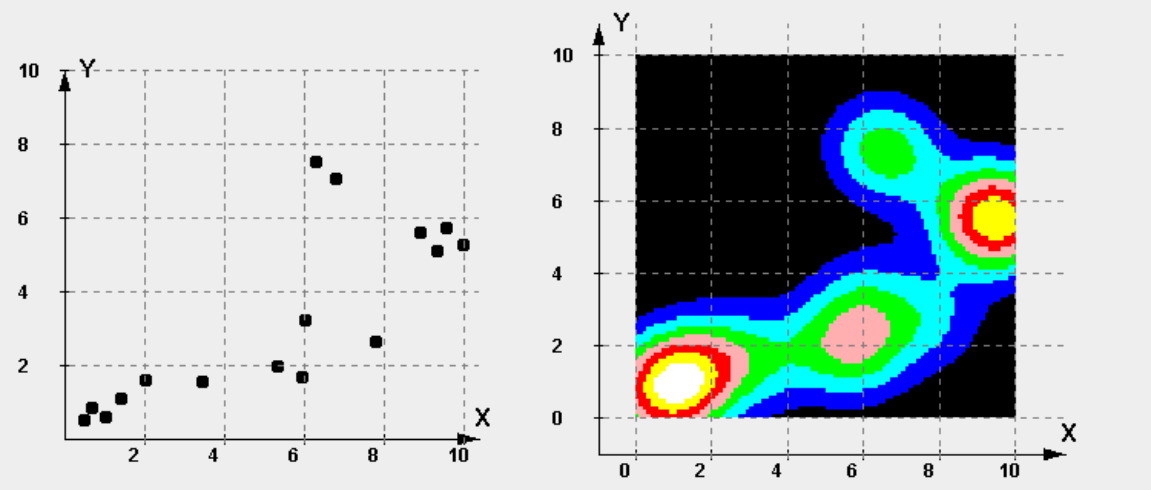
Hard Drive Usage – the ratio of bytes stored on the hard drive vs. the total number of bytes on the hard drive.

File Read Bytes/sec – the amount of memory that a read operation is performed on per second.

After assimilating the data, we apply the k-means clustering algorithm. We initially set the number of clusters (*k*) to equal half the range of the data. For example, if a total of 60 processes are utilized by the user during an experiment, we set the number of clusters to 30(60/2). We will now refer to the number of clusters as k. We then select k random values from our data set. The initial centroids are set to equal these values. The data assignment and centroid relocation steps discussed earlier are repeated until the centroids converge with respect to the data.

Once the centroids have converged, we implement the KDE algorithm to show a statistical distribution of the centroids. Figure 4 shows an example of a KDE distribution.

As we can see from the distribution graph in Figure 4, the centroids in Graph1 are more dense around the respective white, red, and yellow areas in Graph 2. The centroids are less dense around the pink, green, and blue areas. The lowest densities can be found around the black areas.



Graph 1 **Graph 2**

Figure 4: Visual example of a KDE (Graph 2) based on data points from Graph 1

Approach Summary

In this section, we look into what this density distribution means. Chapter 4 discusses how the process, hardware, network, and the file system data represent user actions. When collected over time, this data can model user behavior.

The *k*-means algorithm partitions this data into correlated groups called clusters with centroids that represent the mathematic "mean" of these groups. Within the context of this thesis, these clusters identify the center of correlated behavior.

Although k-means clustering allows us to identify this behavior, they do not categorize new data as normal or abnormal. The probability distributions from KDEs allow us to assign a probability to how 'normal' an action is. A flowchart of this approach can be observed in Figure 5.

This approach will be implemented for the experiments and results discussed in Chapters 5 and 6 respectively.

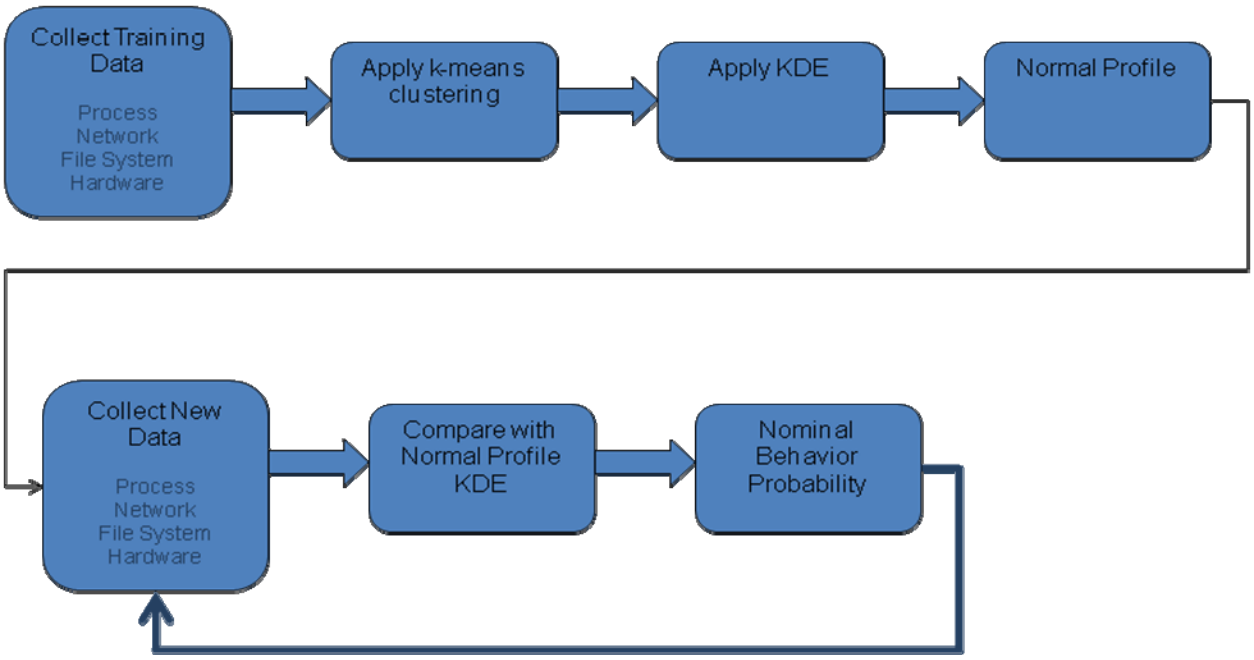


Figure 5: Flowchart of approach from collecting data to producing nominal behavior probability

CHAPTER V

Experimental Setup

Several staff, interns, and researchers at Oak Ridge National Laboratory (ORNL) agreed to be monitored for a pre-determined period of time. Anonymity will be maintained for all participants. However, their role within ORNL (staff, researcher, intern) may be specified. All experiments were conducted on the Windows XP operating system.

All participants were aware that their file system, network, hardware, and process information would be monitored and that data would be collected based on the user and system activity on their computer. Although the participants were aware of this, we asked that they conduct tasks the way that they typically would if their behavior was not being monitored. Issues concerning the validity of this assumption are discussed in Chapter 4 as well as other ethical and privacy issues relating to this thesis. The data was collected form participants over periods of time that spanned from a few days to a few weeks depending on the test.

Data Acquisition

Data was accumulated using a monitoring tool based on the Microsoft .NET framework. The tool monitors and collects data based on events and states on the operating system and in the hardware respectively. The data sets and variables associated with this study are discussed in Chapter 4.

Profile Training Phase

All experiments begin with what we call the profile training phase. This consists of accumulating and processing the participant's process, hardware, network and the file system data to establish a normal profile. This data is processed using the k-means clustering and kernel density estimation algorithms discussed in Chapter 4. Each normal profile is represented by the 7 KDE probability distributions below:

Processor Usage Profile – KDE probability distribution based on processor usage data during the profile training stage,

Memory Usage Profile – KDE probability distribution based on memory usage data during the memory training phase,

Hard Drive Usage Profile – KDE probability distribution based on the hard drive usage data during the profile training phase,

Process Threads Profile – KDE probability distribution based on the number of active threads for each process during the profile training phase,

File System Profile – KDE probability distribution based on file system activity data acquired during the profile training phase,

Network IP Profile – KDE probability distribution based on IP addresses and network connection data acquired during the profile training phase,

Network Port Profile – KDE probability distribution based on network port utilization data acquired during the network training phase.

Nominal Behavior Probability

A KDE in essence consists of a probability distribution. Within the context of this thesis, the probability distributions associated with the normal profiles represent the probability that a user is behaving normally. We will refer to this as the *nominal behavior probability*.

One participant's processor profile can be seen in Figure 6. We can see that during the profile training phase, the user's CPU usage was typically between 15 and 40 percent. After this profile is established, processor usage values between 15 and 40 percent will result in a relatively high nominal behavior probability. Processor usage values outside of the 15-40 percent range will result in relatively low nominal behavior probabilities.

Test Cases

None of the participants engaged in behavior that can be deemed malicious. We anticipated this, especially since the participants were aware of the monitoring software. Because of this, we performed tests by simulating common insider attacks with a goal of identifying abnormalities during the simulated attacks. The sub-sections below describe experiments that were performed and discuss the objectives and concerns with each experiment.

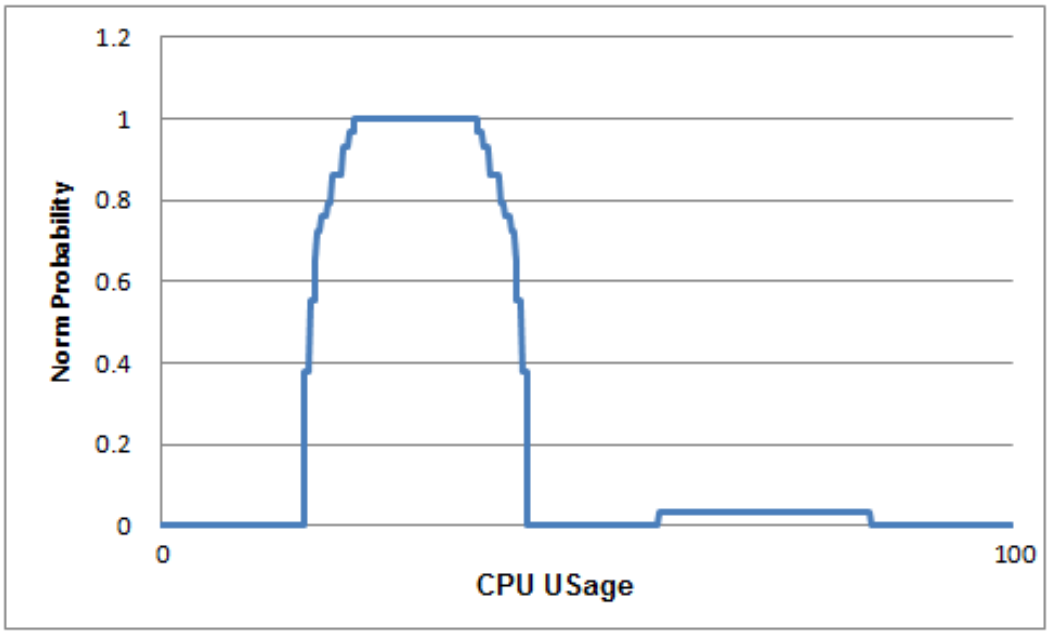


Figure 6: Processor profile nominal behavior probability distribution

Typical Behavior

This test compared users' typical behavior to their normal behavior profile. Participants initially establish a normal behavior profile, and then they are monitored along with their nominal behavior probabilities while they continue their typical daily tasks. This test does not simulate an insider attack, but it does provide valuable insight into expected nominal behavior probabilities for behavior perceived to be normal.

Thread Bomb

A thread bomb is a type of denial of service attack. In this test, a thread bomb hooks into an application such as Notepad or Internet Explorer and begins to spawn either a pre-defined number of threads or an unbounded number of threads. This occurs after the profile training phase. This attack, like many denial-of-service attacks, is sudden and can render a host machine unresponsive, inoperable, or expose the host to different forms of attacks. Detecting such an attack before a host becomes unresponsive allows for preventive or mitigation measures to be taken.

To perform this test, we simulate an attack that hooks into Notepad and spawns 2000 threads.

Abnormal File System Activity

The examples of exfiltration and data corruption involve access to and activity on the file system. In real life exfiltration cases, the attacker often accesses directories that they either do not typically access or that exceed their need-to-know. We performed the following tests for abnormal file system activity:

Abnormal File Deletions: Before establishing normal user profiles, we created arbitrary 40MB directories inside the user, programs, and system32 directories. After establishing a normal profile, each participant systematically deleted a combination of files from one or multiple of these arbitrary directories. Participants acknowledged that they rarely perform

tasks inside of the system32 and Programs directory because they use machines with least-user-privileges" (LUP).

Abnormal File System Activity: After establishing a normal profile, each participant created modified, deleted, and renamed files in directories that they typically do not access.

Abnormal Network Activity

In the following tests, we aim to detect abnormal endpoint-to-endpoint network activity by monitoring the IP addresses of network connections. We also aim to detect abnormal port access. The following tests were conducted after the profile training phase:

New Network Connection: Each participant made frequent visits to a foreign website that they typically do not visit.

New Port: Each participant connected to remote servers on ports that they typically do not utilize.

Abnormal Process Activity

Abnormal process activity can be an indicator that a malicious application, malicious code, or any other form of malware exists on a system. As discussed earlier, these attacks can be devastating if left undetected. We created several tests that we thought would be useful in detecting abnormal process activity. The following tests were conducted after the profile training phase:

New Program: Each participant began to utilize programs that they did not use during the profile training phase.

Abnormal Hardware States

Abnormal hardware states can be a sign of a variety of denial-of-service and data corruption attacks. The following tests were considered useful in detecting abnormal processor and hard drive activity:

Processor: Each participant executed and utilized programs to increase the processor's work load.

Hard Drive Test 1: Each participant loaded at least 10GB of data onto the hard drive after the profile training phase.

Hard Drive Test 2: Each participant loaded 15 GB of arbitrary data onto the hard drive of their computer before the profile training phase. After the training phase, the participants delete the 15GB of data.

CHAPTER VI

Results and Discussion

Typical Behavior

After establishing a normal behavior profile, participants were monitored while they continued their typical daily tasks. Table 1 contains the average nominal behavior probabilities associated with each profile.

In Figure 7, we observe a snapshot of a participant's hard drive usage with respect to their hard drive profile. The results below are formatted as follows:

"Nominal Behavior Probability: "NomBehProbability (#FreeMBytes) "Ave:" AveNormBehProb

NomBehProbability – The nominal behavior probability for a specific instance

#FreeMBytes – The amount of free memory on the hard drive in megabytes

AveNormBehProb – The average nominal behavior probability for the participant during the entire test

Table 1: Average nominal behavior probabilities associated with typical daily tasks.

Profile	**Average Nominal Behavior Probabilities**
Processor Usage	.720
Memory Usage	.694
Hard Drive Usage	.995
Threads	.614
File System	.593
Network IP	.328
Network Port	.688

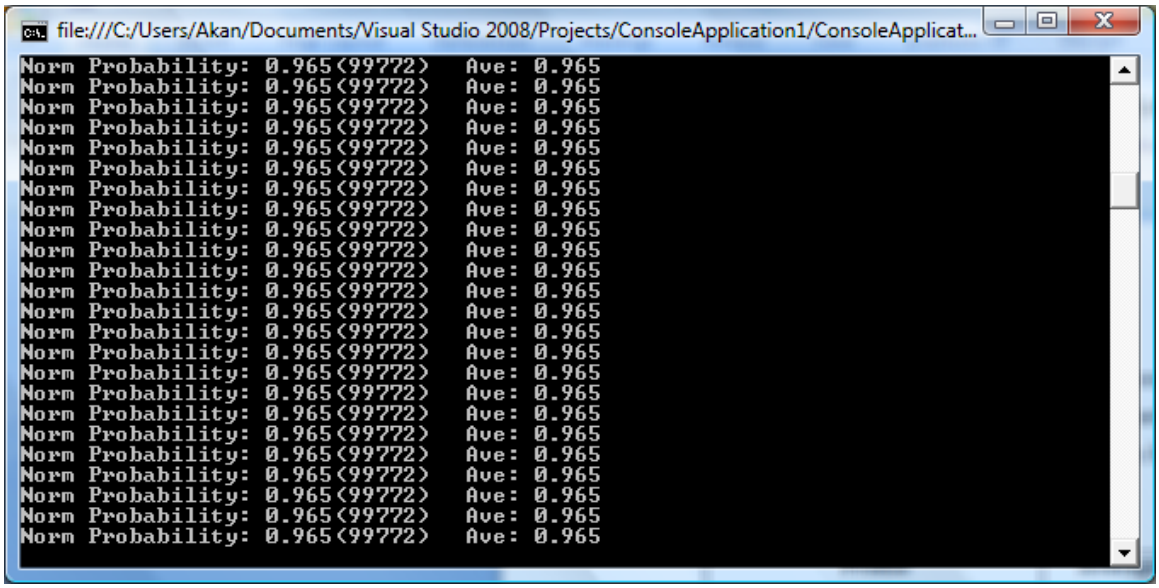


Figure 7: Screen print of an individual's hard drive usage over time in comparison the hard drive profile.

We observe a hard drive nominal behavior probability of .9655 from this screen shot. This implies that the participant's behavior associated with the hard drive is most likely normal.

Thread Bomb

With this test we attempt to detect abnormal activity in the active threads on the host. Table 2 shows the average nominal behavior probabilities associated with the users' thread profile before and after the deployment of the thread bomb. We see a drop of .143 in the nominal behavior probability. Although we expected a larger difference between the nominal behavior probabilities before and after the thread bomb, these results do show that less "normal" activity is taking place with regard to process threads. We assume that the moderate changes in the nominal behavior probabilities are due to the thread bomb only affecting one process. All other processes are unaffected by the thread bomb, thus do not significantly contribute to a decrease in the nominal behavior probabilities.

Table 2: Thread profile average nominal behavior probability before and after thread bomb deployment.

Thread Profile	Average Nominal Behavior Probability
Before Thread Bomb Deployment	.631
After Thread Bomb Deployment	.488

Rather, the unaffected processes would contribute to the nominal behavior probabilities approaching the value before the deployment of the thread bomb.

Abnormal Network Activity

The following network tests attempt to detect abnormal point-to-point network activity based on IP addresses and ports.

New Network Connection Test

Table 3 shows the average nominal behavior probabilities associated with the users' network IP profile and before and after each participant made frequent visits to a foreign website that they typically do not visit.

These results show virtually no change between the nominal behavior probabilities before and after the access to foreign websites. We attribute this to the nature of the IP address and the behavior of the participants. For example, the IP address for Google.com is 209.85.227.106 and is hosted on a server in Mountain View, California. The IP address for Amazon.com is 72.21.210.250 and is hosted on a server in Seattle, Washington. NorthKorea.com is hosted on a server in Germany with an IP address of 82.98.86.165. The network card on a host machine however does not know the URL of the websites that it visits. It also does not know the physical location of these websites. The network card only knows the IP address, the port, and other packet level information associated with a network connection.

Because of this, we cannot identify why 82.98.86165 would be abnormal when compared to 209.85.227.106 and 72.21.210.250 without location or domain information.

Table 3: Network IP profile average nominal behavior probability before and after abnormal network activity.

Network IP Profile	Average Nominal Behavior Probability
Before Foreign Website Access	.353
After Foreign Website Access	.349

There are IP geo-location databases that pair IP addresses with their physical address. Querying this information is relatively slow and is not inately supported in C#. Acquiring this location information about each IP address would benefit future work in this research. Internet users typically visit numerous new websites every day. Unless we can acquire information about the physical location of these network endpoints, then identifying abnormal endpoint-to-endpoint network IP behavior will be extremely challenging.

New Network Port Test

Table 4 shows the average nominal behavior probabilities associated with the users' network port profile before and after each participant utilizes ports for network connections that they typically do not use. These results were promising in that they show a decrease of .79 in the average nominal behavior probability. This significant decrease shows that abnormal port utilization can be easily detected.

Abnormal File Deletion

Table 5 shows the average nominal behavior probabilities associated with the participants' file system profile before each user performed the file deletions associated with this test. The following shows these values seconds after new ports have been utilized. With these results, we see a moderate decrease in the average nominal behavior probability. This decrease, although moderate, is still significant and can be attributed to the abnormal deletions.

Table 4: Port profile average nominal behavior probability before and after abnormal port utilization.

Network Port Profile	Average Nominal Behavior Probability
Before Access to Different Ports	.894
After Access to Different Ports	.104

Table 5: File system profile average nominal behavior probability before and after abnormal file deletions.

File System Profile	Average Nominal Behavior Probability
Before File Deletions	.561
After File Deletions	.321

Abnormal File System Activity

Table 6 shows average nominal behavior probabilities associated with the users' file system profile before each participant performed the file system actions associated with this test. These results show a slight decrease in the average nominal behavior probability.

Abnormal Process Activity

Table 7 shows average nominal behavior probabilities associated with the users' process profile before and after the participants began to utilize one or more program or service that was not utilized during the profile training phase. With a decrease of .345, we observe a significant enough decrease to identify abnormal process activity. There are many variables associated with processes that may be useful if monitored to detect abnormal behavior and can be explored in future work. With a decrease of .345, we observe a significant enough decrease to identify abnormal process activity.

Table 6: File system profile average nominal behavior probability before and after abnormal file system activity.

File System Profile	Average Nominal Behavior Probability
Before Abnormal File System Behavior	.512
After Abnormal File System Behavior	.429

These include run time, CPU usage, memory usage, handles, the parent process, and I/O data operations per process. This experiment shows that monitoring threads can be useful in identifying abnormal process behavior. Combining thread monitoring with some of the variables above may enhance these results.

Abnormal Hardware States

The following tests aim to detect abnormalities attributed to state of the processor and the hard drive.

Processor

Table 8 shows average nominal behavior probabilities associated with the users' processor profile and activity before and after each participant executed and utilized programs to increase the processor's work load.

The nominal behavior probability associated with the processor profile decreases an average of .514 from before to after the abnormal activity. This is a significant difference and shows that that abnormal processor states can be identified.

Hard Drive Test 1

Tables 9, 10, and 11 show information associated with the users' hard drive profile and activity before and after 10GB of data is added to the hard drive.

Table 7: Thread profile average nominal behavior probability before and after new program utilization.

Process Thread Profile	Average Nominal Behavior Probability
Before New Program Utilization	.603
After New Program Utilization	.258

This is a significant difference and shows that that abnormal processor states can be identified.

Hard Drive Test 1

Tables 9, 10, and 11 show information associated with the users' hard drive profile and activity before and after 10GB of data is added to the hard drive. These results show a drastic decrease in the average nominal behavior probability after GBs worth of data are added onto the hard drive. This is similar to the results of Hard Drive Test 2, so we will discuss these drastic nominal behavior probability differences with the results for the following test.

Hard Drive Test 2

Tables 12, 13, and 14 show average nominal behavior probabilities associated with the users' hard drive profile and activity before and after the 1GB, 5GB, and 15GB associated with this test are deleted from the hard drive. These results, like the results from Hard Drive Test 1 show an average decrease in the nominal behavior probability of almost 1. This is because of the nature of how the hard drives are used by the participants while behaving normally. During the profile training phase, the amount of each hard drive used by the participants did not fluctuate much. We noticed an average growth of used hard drive space of under 1 megabyte per day. This observation leads us to understand why a fluctuation of 1GB might cause such a large decrease in the nominal behavior probability. Figure 6 shows a participant's hard drive usage data monitored for almost 300 hours during profile training. Notice how the amount of free hard drive space does not fluctuate much more than 100MB in aver 12 days.

Table 8: Processor profile average nominal behavior probability before and after increased processor usage.

Processor Profile	Average Nominal Behavior Probability
Before Increased Workload	.659
After Increased Workload	.145

Table 9: Hard drive profile average nominal behavior probability before and after adding 1GB to the hard drive.

Hard Drive Profile	Average Nominal Behavior Probability
Before Adding 1GB	.993
After Adding 1GB	.013

Table 10: Hard drive profile average nominal behavior probability before and after adding 5GB to the hard drive.

Hard Drive Profile	Average Nominal Behavior Probability
Before Adding 5GB	.993
After Adding 5GB	.000

Table 11: Hard drive profile average nominal behavior probability before and after adding 10GB to the hard drive.

Hard Drive Profile	Average Nominal Behavior Probability
Before Adding 10GB	.993
After Adding 10GB	.000

Table 12: Hard drive profile average nominal behavior probability before and after deleting 1GB from the hard drive.

Hard Drive Profile	Average Nominal Behavior Probability
Before Removing 1GB	.997
After Removing 1GB	.008

Table 13: Hard drive profile average nominal behavior probability before and after deleting 5GB from the hard drive.

Hard Drive Profile	Average Nominal Behavior Probability
Before Removing 5GB	.997
After Removing 5GB	.000

Table 14: Hard drive profile average nominal behavior probability before and after deleting 15GB from the hard drive.

Hard Drive Profile	Average Nominal Behavior Probability
Before Removing 15GB	.997
After Removing 15GB	.000

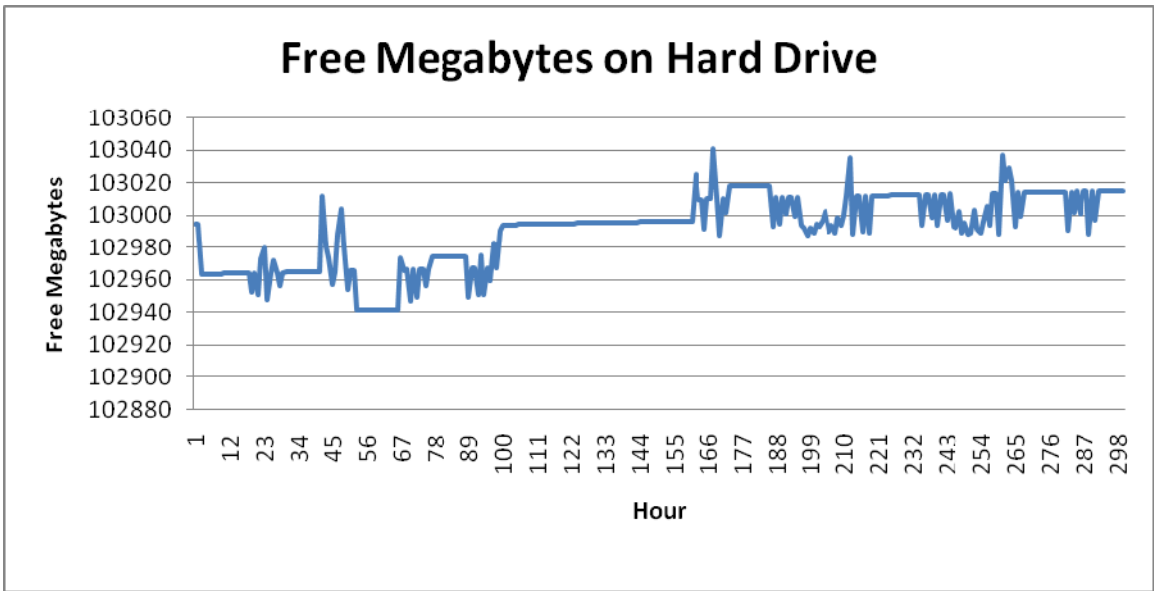


Figure 6: Free megabytes on a hard drive over time

Summary of Results

Every test resulted in a decrease in the average nominal behavior probability. Table 15 and figure 7 show a breakdown of nominal behavior probability changes for each test. The tests focused on the hard drive and the network ports showed the largest nominal behavior probability change ($p \geq .79$, where p is the nominal behavior probability). Tests focused on file system deletions, process activity, and processor usage resulted in moderate decreases in the nominal behavior probability ($.2 < p < .79$). The tests focused on endpoint-to-endpoint network connections and abnormal file system activity showed relatively small changes in the nominal behavior probability ($p < .2$). Possible reasons for these small changes are discussed in previous sections of this chapter.

Table 15: The changes in the average nominal behavior probabilities for each test.

Test	Δ Norm Probability
Thread Bomb	0.143
New Network Connection	0.004
New Port Access	0.79
Abnormal File Deletion	0.24
Abnormal File system Activity	0.083
Abnormal Process Activity	0.345
Processor Usage	0.514
Hard Drive – Add 1GB	0.98
Hard Drive – Add 5GB	0.993
Hard Drive – Add 10GB	0.993
Hard Drive – Delete 1GB	0.989
Hard Drive – Delete 5GB	0.997
Hard Drive – Delete 15GB	0.997

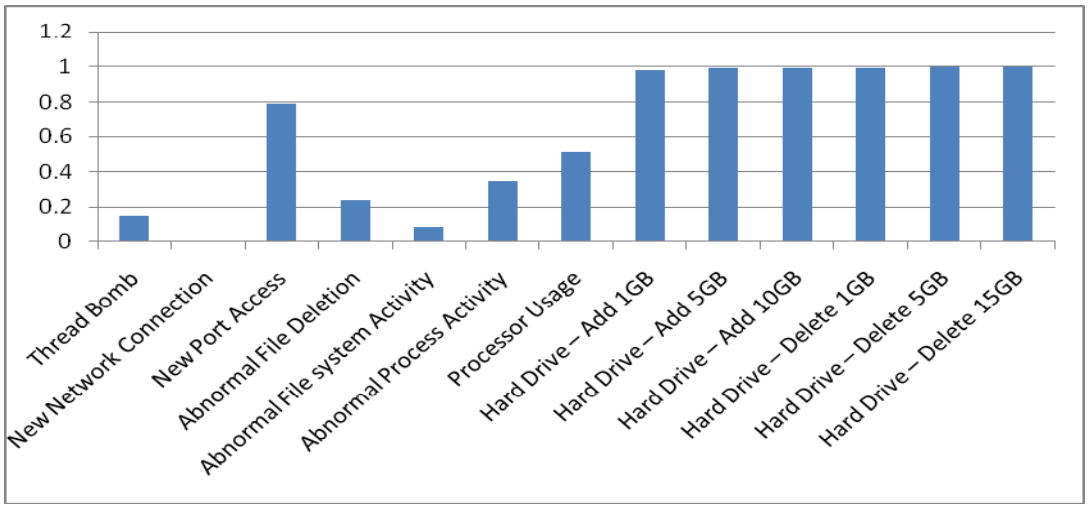


Figure 7: Bar Graph of the average nominal behavior probabilities difference for each test

CHAPTER VII

Conclusions and Future Work

These results show that is possible to identify abnormal behavior by monitoring user activity. The results are promising in that we see decreases in the nominal behavior probability for each simulated attack except for abnormal network IP addresses. We discuss possible reasons for this in Chapter 5. With these results, we provide a means by which to gauge, from a probabilistic view, how "normal" actions are. Future research can identify thresholds for which to categorize actions as threatening or non-threatening based on these nominal behavior probabilities.

To our knowledge, this is the first work that attempts to identify abnormal single/multi user behavior through learning the user's typical behavior patterns. Although previous work attempts to utilize user profiling to combat threats such as masquerading [5, 42, 43] or exfiltration [52,53], this work establishes a means to gauge the normality of user behavior, which also not explored by previous work.

Future Work

This work implements one and two dimensional analysis based on the target variables. We assign 7 KDEs that target variables exclusively to represent normal profiles. Instead of isolating individual variables (e.g. Processor Usage, File Deletions, and Threads) for behavioral analysis, combining datasets and increasing the dimensions for analysis may show interesting results and possible correlations between variables. For example instead of just analyzing the number of threads per process, we could analyze the number of threads per process per process CPU usage. This 3 dimensional analysis could draw show relations between the number of threads and the CPU usage of active processes. Because of this, we would like to explore increasing the dimensionality of data analysis.

We would also like to explore correlations between different events. Behavior between different components on a host machine may be related or may differ in sensitivity, importance, or perceived indication. This analysis may provide insight into how sensitive a specific sensor is and how much weight to place on a specific sensor.

The intent of this research was to monitor user behavior. However, the monitoring software showed that system activity occurs on a host while unoccupied by the user. The operating system creates deletes and modifies numerous log files continuously, which affects not only the file system, but also the hard drive. Operating system processes and services are running in the background, affecting the processor usage and the memory usage. It is also possible for applications that are not utilized by the user to make network connections. The monitoring software seems to indicate that these activities occur at a consistent rate when the host machine is not occupied by the user. We would like to applying similar concepts mentioned in this work to detect abnormalities on a server or host rarely occupied by physical users. This would be done by establishing a normal system profile. System behavior that deviated from the normal profile can be flagged as a possible attack.

As discussed in Chapter 5, identifying abnormal IP address connections is difficult without location or domain information. Because of this, we would like to explore an approach in which we attempt to identify abnormal network behavior based on the location and domain of the network connections.

In this work, normal profile development is completed before users are monitored for abnormal behavior. It is possible for user behavior to vary slightly as projects or tasks change over time. Because of this, we would like to explore a method in which the normal profile is gradually updated after the profile training phase. This includes analysis on data sampling across time and exploring behavior correlations over time.

BIBLIOGRAPHY

[1] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z. Zhou, M. Steinbach, D. J. Hand, D. Steinberg, "*Top 10 algorithms in data mining*", Knowledge and Information Systems, 2008.

[2] M. Bishop, C. Glass, "*Defining the Insider Threat*", In *Proceedings of the 4th annual workshop on Cyber Security and Information Intelligence Research*, pages 1–3, Oak Ridge, TN, 2008.

[3] M. Bishop, "Position. Insider is Relative", 2005. In *Proceedings of the 2005 New Security Paradigms Workshop*, pages 77–78, Lake Arrowhead, CA, October 20–23, 2005.

[4] CSO magazine, U.S. Secret Service, CERT, Deloitte, "*2010 Cyber Security Watch Survey*", 2010.

[5] S. McKinney, D. S. Reeves, "*User Identification Via Process Profiling*", In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research*, Oak Ridge, TN, 2009.

[6] F. Aiolli, "*A Preference Model for Structured Supervised Learning Tasks*", 2005. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, IEEE Computer Society, pages 557–560, Houston, TX, November 27-30, 2005.

[7] W. Eberle, L. Holder, "*Insider Threat Detection Using Graph-Based Approaches*", In *Cybersecurity Applications & Technology Conference For Homeland Security* , pages 237–241. IEEE Computer Society,, 2009.

[8] M. Kirkpatrick, F. Sheldon, "*An Architecture for Contextual Insider Threat Detection*", 2009.

[9] W. H. Baker, A. Hutton, C. D. Hylender, C. Novak, C. Porter, B. Sartin, P. Tippett, J. A. Valentine, "*2009 Data Breach Investigations Report*", Verizon, 2009.

[10] D. Anderson, D. M. Capelli, J. J. Gonzalez, M. Mojtahedzadeh, A. P. Moore, E. Rich, J. M. Sarriegui, T.J. Shimeall, J. M. Stanton, E. Weaver, A. Zagonel, "*Preliminary System Dynamics Maps of the Insider Cyber-threat Problem*", *22nd International Conference of the System Dynamics Society*, Oxford, UK, 2004.

[11] M. Bishop, C. Gates, D. Frincke, F. Greitzer, "*AZALIA: an A to Z Assessment of the Likelihood of Insider Attack*", *IEEE Conference on Homeland Security Technologies*, May 2009.

[12] K. S. Kiillourhy, R. A. Maxion, "*Toward Realistic and Artifact-Free Insider-Threat Data*", In ACSAC '07: 23rd Annual Computer Security Applications Conference, pages 87--96, Dec. 2007.

[13] A. Udoeyop, F. Sheldon, M. Kirkpatrick, "*Heuristic Identification and Tracking of Insider Threat Prospectus*", August 2009.

[14] M. Maybury, P. Chase, B. Cheikes, D. Brackney, S. Matzner, T. Hetherington, B. Wood, C. Sibley, J. Marin, T. Longstaff, L. Spitzner, J. Copeland, S. Lewandowski, "*Analysis and Detection of Malicious Insiders*", *International Conference on Intelligence Analysis*, McLean, VA, 2005.

[15] Q. Ni, J. Lobo, S. Calo, P. Rohatgi, E. Bertino, "*Automating Role-based Provisioning by Learning from Examples*", Symposium on Access control Models and Technologies, 2009.

[16] Information Assurance Technology Analysis Center, "*The Insider Threat to Information Systems*", State-of-the-Art Report, October 2008.

[17] C. Blackwell, "*A Security Architecture to Protect against the Insider Threat from Damage, Fraud and Theft*", In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research*, 2009.

[18] J. Myers, M. R. Grimaila R. Mills, "*Towards Insider Threat Detection using Web Server Logs*", In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research*, 2009.

[19] Andrew Moore, "K-*means and Hierarchical Clustering – Tutorial Slides*", accessed 19 January, 2010. http://www-2.cs.cmu.edu/~awm/tutorials/kmeans.html.

[20] Politecnico Di Milano, "*K-Means Clustering", A Tutorial on Clustering Algorithms*", accessed 17 January, 2010. http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html.

[21] T. Duong, "*An Introduction to Kernel Density Estimation*", accessed 5 February, 2010. http://school.maths.uwa.edu.au/~duongt/seminars/intro2kde/.

[22] M. Rousson, D. Cremers, "*Efficient Kernel Density Estimation of Shape and Intensity Priors for Level Set Segmentation*" In *MICCAI*, volume 1, pages 757–764, 2005.

[23] M. Mahajan, P. Nimbhorkar, K. Varadarajan (2009). "*The Planar k-Means Problem is NP-Hard*". In *Proceedings of the 3rd Int. Workshop on Algorithms and Computation*, volume 5431 of *Lecture Notes in Computer Science*, pages 274-285, 2009.

[24] B. Fisher, "Kernel Density Estimators", accessed 25 January, 2010 at: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0405/MISHRA/kde.html.

[25] B. W. Silverman, "*Density estimation for statistics and data analysis*", Chapman and Hall, London, 1986.

[26] M. Balakgie, "*Computer Security: Cyber Attacks—War without Borders*" Testimony before a hearing of the House Subcommittee on Government Management, Information, and Technology. 26 July 2010, accessed 31 December 2009 at: http://www.iwar.org.uk/cip/resources/congress/000726mb.htm.

[27] DHS, "*National Infrastructure Protection Plan*", 2006, accessed 18 February 2010 at: http://www.dhs.gov/xprevprot/programs/editorial_0827.shtm.

[28] US Office of Homeland Security, "*The National Strategy for Homeland Security*", 16 July 2002, p. 30, accessed 18 February 2009 at: http://www.whitehouse.gov/homeland/book/nat_strat_hls.pdf.

[29] Emery, Adam, IBM, "*US Businesses: Cost of Cybercrime Overtakes Physical Crime*", IBM Press Release, 14 March 2006, accessed 31 December 2009 at: http://www.ibm.com/press/us/en/pressrelease/19367.wss.

[30] Gordon, Lawrence, Martin Loeb, William Lucyshyn, and Robert Richardson, "*CSI/FBI Computer Crime and Security Survey*", 2005, accessed 20 December 2009 at: http://www.gocsi.com/press/20050714.jhtml.

[31] Gordon, Lawrence, Martin Loeb, William Lucyshyn, and Robert Richardson, "*CSI/FBI Computer Crime and Security Survey 2006*", accessed 21 December 2009 at: http://www.gocsi.com/press/20050714.jhtml.

[32] PricewaterhouseCoopers for the UK Department of Trade and Industry, "*Information Security Breaches Survey 2006*", accessed 16 December 2009 at:

http://www.pwc.com/ExtWeb/pwcpublications.nsf/docid/F9843CD3C8E0FB828025715A0058C63B.

[33] Young, Tom. "Lloyds *TSB tackles insider fraud threat Computing*", 6 June 2007, accessed 17 December 2009 at: http://www.computing.co.uk/computing/news/2191468/lloyds-tsb-tackles-insider.

[34] Goodlin, Dan. "*University admins lend phishers a hand*" The Register, 18 April 2007, accessed 20 December 2009 at: http://www.theregister.co.uk/2007/04/18/university_phish_hack.

[35] Glater, Jonathan. "*US Limits Access to Student Loan Database*", New York Times, 18 April 2007, 20 December 2009 at http://www.nytimes.com/2007/04/18/us/18loans.html?ex=1335412800&en=091151830d19d963&ei=5124&partner=permalink&exprod=permalink.

[36] Mayo, Tracy, "*Crash Course*", CSO Online, 1 August 2004, accessed 21 December 2009 at: http://www.csoonline.com/read/080104/school.html.

[37] Rolish, Michael, "*Blaster Worm, Sobig Hit MIT Computers*", MIT whitepaper, August 2003, accessed 16 December 2009 at: http://www-tech.mit.edu/V123/N34/34virus.34n.html.

[38] State of Virginia. "*Government Data Collection and Dissemination Practices Act*", Code of Virginia § 2.2-3800, accessed 9 February 2010 at: http://leg1.state.va.us/cgi-bin/legp504.exe?000+cod+2.2-3800.

[39] C. Feng, J. Peng, H. Qiao, and J. Rozenblit, "*Alert fusion for a computer host based intrusion detection system*", In *Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, pages 433–440, Washington, DC, USA, 2007. IEEE Computer Society.

[40] J. Peng, C. Feng, H. Qiao, and J. Rozenblit. "*An event-driven architecture for fine grained intrusion detection and attack aftermath mitigation*". In *Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, pages 55–62, Washington, DC, USA, 2007. IEEE Computer Society.

[41] H. Qiao, J. Peng, C. Feng, and J. Rozenblit. "*Behavior analysis-based learning framework for host level intrusion detection*". In *Proceedings of the 14th Annual IEEE*

International Conference and Workshops on the Engineering of Computer-Based Systems, pages 441–447, Washington, DC, USA, 2007. IEEE Computer Society.

[42] J. Shavlik and M. Shavlik, "*Selection, combination, and evaluation of effective software sensors for detecting abnormal computer usage*", In *KDD*, pages 276–285, 2004.

[43] J. Shavlik, M. Shavlik, and M. Fahland. "*Evaluating software sensors for actively profiling Windows 2000 computer users*". In *Fourth International Symposium on Recent Advances in Intrusion Detection*, 2001.

[44] L. Spitzner, "*Honeypots: Catching the insider threat*", In *Proceedings of the 19th Annual Computer Security Applications Conference*, page 170, Washington, DC, USA, 2003. IEEE Computer Society.

[45] Y. Yu and T. Chiueh, "*Display-only file server: a solution against information theft due to insider attack*", In *Proceedings of the 4th ACM workshop on Digital Rights Management*, pages 31–39, New York, NY, USA, 2004. ACM.

[46] S. Pramanik, V. Sankaranarayanan, and S. Upadhyaya, "*Security policies to mitigate insider threat in the document control domain*", In *Proceedings of the 20th Annual Computer Security Applications Conference*, pages 304–313, Washington, DC, USA, 2004. IEEE Computer Society.

[47] J. Park and J. Giordano, "*Role-based profile analysis for scalable and accurate insider anomaly detection*", In *Proceedings of the 25th IEEE International Performance Computing and Communications Conference*, pages 463–469, 2006.

[48] J. Park and S. Ho, "*Composite role-based monitoring (CRBM) for countering insider threats*", In *ISI*, pages 201–213, 2004.

[49] S. Symonenko, E. Liddy, O. Yilmazel, R. Zoppo, E. Brown, and M. Downey, "*Semantic analysis for monitoring insider threats*", In *ISI*, pages 492–500, 2004.

[50] O. Yilmazel, S. Symonenko, N. Balasubramanian, and E. Liddy, "*Improved document representation for classification tasks for the intelligence community*", In *Proceedings of the AAAI Spring Symposium Series*, 2005.

[51] O. Yilmazel, S. Symonenko, N. Balasubramanian, and E. Liddy, "*Leveraging one-class SVM and semantic analysis to detect anomalous content*", In *ISI*, pages 381–388, 2005.

[52] R. Cathey, L. Ma, N. Goharian, and D. Grossman, "*Misuse detection for information retrieval systems*", In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 183–190, 2003.

[53] L. Ma and N. Goharian, "*Query length impact on misuse detection in information retrieval systems*", In SAC, pages 1070–1075, 2005.

[54] B. Aleman-Meza, P. Burns, M. Eavenson, D. Palaniswami, and A. Sheth, "*An ontological approach to the document access problem of insider threat*", In *ISI*, pages 486–491, 2005.

[55] A. Liu, C. Martin, T. Hetherington, and S. Matzner, "*A comparison of system call feature representations for insider threat detection*", In *Proceedings from the Sixth Annual IEEE SMC*, pages 340–347, 2005.

[56] K. Anderson, A. Carzaniga, D. Heimbigner, and A. Wolf, "*Event-based document sensing for insider threats*", Technical Report CU-CS-968-04, University of Colorado Department of Computer Science, 2004.

[57] N. Nguyen, P. Reiher, and G. Kuenning, "*Detecting insider threats by monitoring system call activity*", In *IAW*, pages 45–52, 2003.

[58] M. McConnell, B. A. Hamilton, "*Information assurance in the twenty-first century*". Security & Privacy, Suplement to IEEE Computer Society, 2002, pages 16-19.

[59] Op. cit. Commission for the Review of FBI Security Programs, "*A Review of FBI Security Programs*", 2003.

[60] T. Young, "*Lloyds TSB tackles insider fraud threat*", *Computing Magazine*, accessed 18 February 2010 at: http://www.computing.co.uk/computing/news/2191468/lloyds-tsb-tackles-insider.

[61] DHS. "*Cyber Security Research and Development."* BAA 07-09, May 2007, accessed 18 May 2007 at: http://www.hsarpabaa.com/Solicitations/BAA07-09_CyberSecurityRD_Posted_05162007.pdf.

[62] R. Richardson, "*CSI/FBI Computer Crime and Security Survey 2007,*" accessed 21 December 2009 at: http://www.gocsi.com/press/20050714.jhtml.

[63] D. Bank, C. Conkey, "*New Safeguards for Your Privacy*," *The Wall Street Journal*, page D1. March 24, 2005.

[64] *Course webpage*, accessed 21 December 2009 at: http://homes.cerias.purdue.edu/~crisn/courses/cs590T/index.html.

[65] *Course webpage*, accessed 21 December 2009 at: http://www.cse.sc.edu/~farkas/csce727-2007/lectures.htm.

[66] S. Chi, J. S. Park, K. Jung, J. Lee, "*Network Security Modeling and Cyber Attack Simulation Methodology*," *Lecture Notes in Computer Science*, Vol.2119, 2001.

[67] *Microsoft .NET Framework Overview*, accessed 18 May 2010 at: http://www.microsoft.com/net/overview.aspx.

[68] Microsoft, *DataGrid Class,* accessed 18 May 2010 at: http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.datagrid.aspx/

APPENDIX

Microsoft .NET Variables

Each variable that was monitored using the .NET Framework for this study is listed below. This includes variables from experiments and variables that can be utilized for future work.

Format: “Component”, “Variable”, “Quantity”

"Cache", "Pin Reads/sec"
"IP", "Datagrams/sec"
"IP", "Datagrams Sent/sec"
"IP", "Datagrams Received/sec"
"LogicalDisk", "Free Megabytes", "_Total"
"LogicalDisk", "Disk Reads/sec", "_Total"
"LogicalDisk", "Disk Writes/sec", "_Total"
"LogicalDisk", "Disk Read Bytes/sec", "_Total"
"LogicalDisk", "Disk Write Bytes/sec", "_Total"
"Memory", "% Committed Bytes In Use"
"Memory", "Committed Bytes"
"Memory", "Page Reads/sec"
"Memory", "Page Writes/sec"
"Objects", "Events"
"Objects", "Processes"
"Objects", "Threads"
"PhysicalDisk", "Disk Reads/sec", "_Total"
"PhysicalDisk", "Disk Writes/sec", "_Total"
"PhysicalDisk", "Disk Read bytes/sec", "_Total"
"PhysicalDisk", "Disk Writes bytes/sec", "_Total"
"Print Queue", "Jobs", "_Total"

"Print Queue", "Total Pages Printed", "_Total"

"Processor", "% Processor Time", "_Total"

"RAS Total", "Bytes Received"

"RAS Total", "Bytes Received/Sec"

"RAS Total", "Bytes Transmitted"

"RAS Total", "Bytes Transmitted/Sec"

"RAS Total", "Frames Received"

"RAS Total", "Frames Received/Sec"

"RAS Total", "Frames Transmitted"

"RAS Total", "Frames Transmitted/Sec"

"RAS Total", "Total Connections"

"RAS Total", "Total Errors"

"RAS Total", "Total Errors/Sec"

"Redirector", "Packets/sec"

"Redirector", "Packets Received/sec"

"Redirector", "Packets Transmitted/sec"

"Server", "Logon Total"

"Server", "Errors Access Permissions"

"Server", "Errors Logon"

"Server", "Errors Granted Access"

"Server", "Errors System"

"Server", "File Directory Searches"

"Server", "File Directory Searches"

"Server", "Files Opened Total"

"Server", "Files Open"

"Server", "Sessions Timed Out"

"Server", "Sessions Errored Out"

"Server", "Sessions Logged Off"

"Server", "Sessions Forced Off"

"System", "% Registry Quota In Use"

"System", "Processes"
"System", "Exception Dispatches/sec"
"System", "File Control Bytes/sec"
"System", "File Control Operations/sec"
"System", "File Data Operations/sec"
"System", "File Read Bytes/sec"
"System", "File Read Operations/sec"
"System", "File Write Bytes/sec"
"System", "File Write Operations/sec"
"System", "Processor Queue Length"
"System", "System Calls/sec"
"System", "System Up Time"
"System", "Threads"
"TCP", "Connections Failures"
"TCP", "Connections Active"
"TCP", "Connections Established"
"TCP", "Connections Passive"
"TCP", "Connections Reset"
"TCP", "Segments Received/sec"
"TCP", "Segments Retransmitted/sec"
"TCP", "Segments Sent/sec"
"TCP", "Segments/sec"
"Terminal Services", "Active Sessions"
"Terminal Services", "Inactive Sessions"
"Terminal Services", "Total Sessions"
"UDP", "Datagrams Received/sec"
"UDP", "Datagrams Sent/sec"
"UDP", "Datagrams No Port/sec"

TCP Connections

"Destination IP"

"Destination Port"

"Host IP"

"Host Port"

FileSystem Event Listener

"File Created"

"File Changed"

"File Renamed"

"File Deleted"

Process Components

"Process", "% Processor Time"

"Process", "% User Time"

"Process", "% Privileged Time"

"Process", "Virtual Bytes"

"Process", "Working Set"

"Process", "Page File Bytes"

"Process", "Private Bytes"

"Process", "Thread Count"

"Process", "Elapsed Time"

"Process", "Process ID"

"Process", "Creating Process ID"

"Process", "IO Read Operations/sec"

"Process", "IO Write Operations/sec"

"Process", "IO Data Operations/sec"

"Process", "IO Other Operations/sec"

"Process", "IO Read Bytes/sec"

"Process", "IO Write Bytes/sec"

"Process", "IO Data Bytes/sec"

"Process", "IO Other Bytes/sec"

VITA

Akaninyene Walter Udoeyop was born in Calabar, Nigeria on August 11, 1986. After moving to England and then America, he graduated from Columbia High School in Maplewood NJ. He then went on to receive his Bachelors of Science in Computer Engineering from the University of Tennessee in Knoxville in May 2008. In August 2008, he proceeded to pursue a Master's of Science in Computer Engineering with a concentration in Computer Architecture at the University of Tennessee in Knoxville.