Holistic Network Defense: Fusing Host and Network Features for Attack Classification

THESIS

Jenny W. Ji, 1st Lieutenant, USAF

AFIT/GE/ENG/11-18

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/11-18

Holistic Network Defense: Fusing Host and Network Features for Attack Classification

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

Jenny W. Ji, BS

1st Lieutenant, USAF

March
2011

AFIT/GE/ENG/11-18

Holistic Network Defense: Fusing Host and Network Features for Attack Classification

Jenny W. Ji, BS

1st Lieutenant, USAF

Approved:

Dr. Gilbert L. Peterson (Chairman)	date
Dr. Robert F. Mills (Member)	date
Dr. Michael R. Grimaila (Member)	date

AFIT/GE/ENG/11-18

Abstract

Defending the cyberspace domain has taken a higher priority as the internet has become an enabler and integrated component of the defense strategy. At the same time, it has also opened additional channels for enemy exploitation. Current defensive systems focus primarily on network data, and are plagued by a high number of false positives and/or duplicate alerts with no ranking of their importance. This work presents a hybrid network-host monitoring strategy, which fuses data from both the network and the host to recognize malware infections. This research seeks to categorize systems into one of three classes: Normal, Scanning, and Infected. Normal is defined as a computer surfing the World Wide Web, with antivirus and malware shields up and running, any threats are assumed to be random. Scanning is defined as the perpetrator having a partial or complete map of the network and knowing trying exploits at targeted IPs or IP ranges. Infected is defined as an operational computer surfing the World Wide Web and having an active trojan infection as defined by the Antivirus alerting to an infection. The objective is accomplished by fusing data from multiple network host sensors and extracting features from network traffic using the Fullstats Network Feature generator and from the host using text mining, looking at the frequency of the 500 most common strings and analyzing them as word vectors. Testing on data collected at AFIT from the 2010 Cyber Defense eXercise and a controlled data collection of a Normal Windows

Vista host and an Infected Vista host. Hybrid method results outperformed host only classification by 31.7% and network only classification by 25%. The new approach also reduces the number of alerts while remaining accurate compared with the commercial Intrusion Detection System (IDS) SNORT. These results improve the relevance of alerts so that even the most typical users could understand alert classification messages. A key benefit of the hybrid data fusion methodology is mission relevant information by identifying and reporting the progression of an attack through the three stages.

Acknowledgements

To my advisor, course instructors, student colleagues and committee, thank you for your time and mentorship when I had my many technical roadblocks, and providing the motivation for me to get back up when I hit them.

To my friends and family, thank you for reminding me that there is a life beyond the ivory tower.

Finally, I'd like to thank the leadership from my previous assignments, who believed in and recommended me for a wonderful opportunity to further my academic credentials and develop skills that will help me wherever my career takes me.

Jenny W. Ji

Table of Contents

(Includes all Tables and Figures)

List of Abbreviations

ACM	Association for Computing Machinery
AIS	Artificial immune systems
ANN	Artificial Neural Network
AV	AntiVirus
C&C	Command and Control
CDX	Cyber Defense eXercize
CSV	Comma Separated Value
DARPA	Defense Advanced Research Projects Agency
DDoS	Distributed denial of service
DLL	Dynamic-Linked Library
DNS	Domain Name Server
DoD	Department of Defense
FOL	First-order Logic
GUI	Graphical User Interface
HIDS	Host-based intrusion detection system
HTTP	HyperText Transfer Protocol
I/O	Input/output
IANA	Internet Assigned Numbers Authority
IDS	Intrusion detection system
LVQ	Learning Vector Quantization
MIT	Massachusetts Institute of Technology

NIDS	Network IDS
NSA	National Security Agency
OS	Operating System
PCTCG	Principal-subordinate Consequence Tagging Case Grammar
PID	Process Identification Numbers
ROC	Receiver Operating Characteristic
SCADA	Supervisory Control and Data Acquisition
SIGKDD	Special Interest Group for KDD
SOM	Self Organizing Map
SVM	Support Vector Machine
TF/IDF	Term Frequency/Inverse Document Frequency
WEKA	Waikato Environment for Knowledge Analysis

I. Introduction

The industrial cyber security report produced by the British Columbia Institute of Technology, and the PA Consulting Group in the April 1st 2005 issue states that there has been a 10-fold increase in the number of successful cyber attacks on infrastructure Supervisory Control and Data Acquisition (SCADA) systems since 2000 [5]. The Department of Defense (DoD) officials have also observed that the number of attempted intrusions into military networks has increased, from 40,076 incidents in 2001, to 43,086 in 2002, to 54,488 in 2003, and to 24,745 as of June 2004 [5]. A newer report, the 2007 E-Crime Watch Survey from CSO Magazine found the number of security incidents increased for the majority of companies polled from the period between 2005 and 2007. These findings were based on the work of the U.S. Secret Service, Carnegie Mellon University Software Engineering Institute's CERT Coordination Center and Microsoft [6]. However, the consequences of these attacks on military operations are not as clear as financial valuations do not paint the whole picture when national security is at risk. This prompts a predominantly wait and see stance for much of policy development in this arena.

Without a doubt, over the last decade, malware has become a primary source of malicious cyber activity [1]. Malware encompasses a whole progression of escalating activities that include scanning, (distributed) denial-of-service (DOS), and direct exploit attacks, taking place across the Internet. Among its various forms, botnets in particular

have recently distinguished themselves to be among the premier threats, making up a high volume of total internet traffic and slowing the information superhighway for all other users [19]. Like the previous generation of computer viruses and worms, a bot is a self-propagating application that infects vulnerable hosts through direct exploitation or Trojan insertion. However, bots distinguish themselves from the other malware forms by their ability to establish a command and control (C&C) channel through which they can be updated and directed. Once collectively under the control of a C&C server, bots form what is referred to as a botnet. Botnets are effectively a collection of zombie computing assets employed for a variety of illicit activities, including information and computing resource theft, SPAM production, hosting phishing attacks, or for mounting distributed denial-of-service (DDoS) attacks.

Over the last decade, the term cyberwar has developed from a mere virtual threat to more a concern of national security between nation states as demonstrated by recent examples from Estonia's internet outage in 2008 [7], and the new software "worm" Stuxnet [17]. The 2008 Russian invasion of Georgia was reportedly accompanied by a parallel hit on their information networks [35]. Stuxnet spreads via infected memory sticks plugged into a computer's USB port. Stuxnet checks to see if WinCC is running. If it is, it tries to log in, to install a clandestine "back door" to the internet, and then to contact a server in Denmark or Malaysia for instructions. Microsoft said in August 2010 that Stuxnet had infected more than 45,000 computers. At Iran's uranium-enrichment plant at Natanz, inspections by the International Atomic Energy Agency found that about half Iran's centrifuges are idle and those that work were yielding little. Some say a fall in the

number of working centrifuges at Natanz in early 2009 is evidence of a successful Stuxnet attack. There is little downside to such an attack, because it would be virtually impossible to prove who did it, but new reports indicate [42] that the sophistication of the code suggests the effort was backed by a national entity and there is a slew of technically capable candidates.

1.1. Problem Statement

Network-based intrusion detection systems (NIDS) and intrusion prevention systems (IPSs) are currently the go-to technology for defending against external network attacks. Traditional NIDS, whether signature based [48] or anomaly based [3], focus on inbound packets flows for signs of malicious point-to-point intrusion attempts. Often, a NIDS is installed at an access gateway, to watch over a network consisting of a multitude of individual hosts and it is difficult to tell if any one of those is a target or gets successfully infected by an alert thrown from the NIDS. Network IDS have the capacity to detect initial incoming intrusion attempts, but at the sacrifice of a very high false positive rate and an overabundance of relevant true positives as seen by the prolific frequency with which they produce alarms in operational networks [19]. Furthermore, limited throughput often requires sampling of traffic; as it would overwhelm the NIDS to inspect every packet.

On host systems, antivirus (AV) software is relied upon to prevent malicious downloaded code from being installed. Unfortunately, AV scanning of executables for malware detection faces a number of significant problems, one being that current malware programs typically implement run-time packing and self-modifying code [28]. Therefore,

the instruction present in the binary on disk is typically different than those executed at runtime. Distinguishing and preventing a successful local host infection from the myriad scans, intrusion attempts and AV evasions is ultimately the real goal of effective and intelligent network security applications. Statistical based methods of intrusion detection should defend better against a zero day attack, which takes advantage of a bug that neither the software's creators nor users are aware of. By finding software vulnerabilities before the software's manufacturers, a programmer can create a virus or worm that exploits that vulnerability and harms computer systems in a variety of ways. While not every zero day attack truly occurs before software producers are aware of the vulnerability, developing a patch can take time. Alternatively, software producers may sometimes hold off on releasing the patch because they do not want to inundate customers with numerous individual updates. If the vulnerability is not particularly dangerous, software producers may choose to hold off until multiple updates are collected and release them together as a package. Still, this approach can potentially expose users to a zero day attack.

Although there are several NIDS and HIDS implementations, there is currently no commercial IDS and few if any research in this field which fuses data from both of these sources and applies Machine Learning algorithms for classification of the stages and the progression of a cyber-attack. There are a few commercial IDS that still use signature based detection but have augmented the signatures by incorporating host data, examples being McAfee Entercept and IBM's Proventia IPS. In the only published paper available from the literature search on a hybrid methodology by Depren [8], the hybrid methodology was tested on the flawed MIT Lincoln Labs KDD99 dataset which entirely

consists of TCP dumps and no host data. Depren does not explain how the host misuse detection module contributes to the test results, whereas this thesis will collect host data alongside network packet dumps. This work seeks to create a proof of concept methodology for the fusion of network and local host data which will enable an analyst to filter much of the extraneous network sensor alerts and alleviate the noise issues that are commonplace with traditional NIDS and HIDS. The method developed performs anomaly detection based on modeling patterns of features rather than matching a set of features used in more traditional signature based models. These latter models concentrate on the fixed features of a specific malware instance, and as a result, are often easily evaded by code obfuscation or polymorphism, making it straight forward to modify and alter their appearance without changing the behavior of the running script for successful infiltration [28].

1.2. Impact of Research

A signature-based IDS performs well for detecting known threats, but cannot properly identify a novel threat; it must have a signature in its database which exactly matches a given threat in order to detect it [19]. System administrators must constantly update threat-detection signatures to keep up with the ever expanding threat pool.

Additionally, detecting a network attack solely through signature-based systems has been shown to be unsustainable in both the short and long term [46]. As quickly as a signature is created for defense against an exploit, a hacker can unleash a slightly tweaked variant, requiring a completely new signature for detection. This is why zero day exploits

are so doubly damaging; a signature has not been created and cannot be deployed in time to prevent all possible damage.

There are several NIDS (SNORT, BRO, Fragrouter…) and HIDS (Prelude, OSSEC, Osiris…) available, some of these have aggregators but none combine data from both locations to classify the state of attack, the work presented in this thesis will demonstrate that having information available from both host and network increases classification of attack accuracy much more than each alone.

Given the DoD's heavy dependence upon cyberspace, the vulnerable nature of cyberspace, and the multitude of threats which aim to undermine its confidentiality, integrity and availability, continued research of this forward edge domain is paramount.

1.3. Research Overview

One major difficulty facing the cyber situational awareness research community relates to the hyper-dimensionality of the threat search space. Due to the sheer size of the cyberspace domain and the limitations of processing power and lack of data abstraction, the environment will always be partially-observable. Stated otherwise, it is infeasible to try all combinations of sensor data in real time, sensors need to be wisely chosen for the system based on its application environment. Theoretically, given infinite time, sensors, storage and computing power, perfect situational awareness can be derived. Practically, any situational awareness obtained will be constrained by the resources allocated, data available, abstraction ability of the system and the time bounds acceptable to solving the problem. Situational awareness is a computing application like any other, concerned not only with processing effectiveness, but also its efficiency.

There are also limits as to how much we should rely on software for information protection. At hand, there exists many other ways a hacker may try to enter our network, including social engineering lures to gain passwords from unsuspecting users, or an insider could completely bypass the network detection sensors and exfiltrate data with portable media devices by having physical access to the host.

1.3.1. Past Research Summary

Given that signature-based sensors are not feasible for detecting all threats, researchers must consider alternative solutions [47] [3][19]. Researchers have proposed many types of threat detection methods, a representative few of these are discussed further

in Chapter 2. The bulk of the research efforts for threat detection thus far focuses on developing methods relying solely on network traffic [21] [15], solely on event logs [34] [50], or solely on system calls[45]. Unsurprisingly, due to the pressures of finite resources and push to select the fewest features to process, little or no research efforts thus far has attempted to combine data from various system sensor categories, such as file I/O, network traffic and process meta data, in order to form a holistic picture of what data is most relevant for the identification of threats. Situational awareness research aims to bring together raw data to formulate higher level views. This requires transforming the data from its raw form into data which is capable of informing decisions.

1.3.2. Problem Statement and Hypothesis

The present bane of intrusion detection systems is the high incidence of false positive alerts, the alerting of malicious activity when it is not actually present [49]. There is a clear trade-off for a system that catches more potential threats; it means that it will also catch more benign anomalies [39]. More often than not, most alerts will be benign and everyone knows the story of the boy who cried wolf, the danger being that system administrators will eventually start disregarding even a true threat. A system administrator or security personnel must sift through these false alerts in order to locate true positive alerts which point to real threats. In order to improve the accuracy of IDS and ease the burden on system administration personnel, a methodology for reducing the incidence of false positive alerts, while accurately identifying malicious events is needed.

This thesis presents a methodology to utilize the information provided from two sources, host and network sensors to expand cyberspace data understanding for the

purpose of improving threat detection accuracy. By identifying relevant features from an array of sensor data sources, this methodology will identify not only if a system is under attack or not, but also what stage an attack is occurring.

The raw host data is obtained using Windows SysInternals tools and the network traffic is processed with the Fullstats Packet Feature Generator. The resulting features files are then classified using Weka's support vector machine algorithm to classify and test effectiveness. The data collection process involved two environments, The Cyber Defense eXercise and a Vista Machine surfing the internet. Both host and network attributes were formatted into CSV format, using 248 numerical packet metrics for the network data and 500 most frequent string n-grams as attributes in the case of SysInternals host data text files. Weka's results were supportive of the hypothesis that fusing host and network data is a more accurate classification method and detailed results are presented in Chapter 4.

1.3.3. Data Source

Two data sources are used in testing. The first is an isolated network established at AFIT during the 2010 Cyber Defense eXercize (CDX). The CDX challenges the student participants to build a network with all of the services required by the National Security Agency's (NSA) directive--including e-mail, file sharing, network printing, a Web server, and a bulletin board system. The mission is to keep those services running while thwarting attempts to compromise. This year, all teams built their service providing systems or servers from scratch, and received workstation virtual machines from the NSA. Participants are directed not to patch the workstations without prior approval. It was expected that the NSA would find their way into some of the systems regardless of how

tightly they were locked down, although it turned out not to be the case for the servers monitored. The threats tend to cover the full range from downloaded attachments and links to malicious Web sites, to direct scanning, enumeration, and attempts at exploitation.

A massive amount of data was collected of AFIT 1 team's servers and the analysis was ultimately done on the DNS server running Windows Server 2003 using memory dump utility win32dd and SysInternals Suite tools to be further detailed in Chapter 3. AFIT 1 succeeded in preventing NSA infiltration, but as a result, data was of only one category: scanning. To remedy the situation, data for the normal and infected classed were obtained on a test network consisting of a Windows Vista host connected to the Internet using the same collection tools. The host is infected via visits to known malicious websites as listed in the malware domain list and infection was verified by antivirus software.

1.3.4. Assumptions and Limitations

There three predominant limitations and assumptions of this research are that the CDX network was inconsistent and uncontrolled, that the sensors impacted the normal execution of the system, and there was only partial observability into the exercise conduct. The host features were created from data mining the raw text files and thus depends heavily on the application generating these files as they could change the output formats in later versions. Data collections were also of mixed sources (CDX and Vista internet networks) which made it artificially easier for classification than would typically occur.

1.4 Summary

As discussed, doing research to select the lightest sensors for needed performance metrics to extract the best features from is critically important. And by combining network and host sensors, the alerts can be made both more accurate and informative than at present, underlying the importance of this research. This chapter serves as an introductory look at both the current status and the problems of this field of study. Chapter 2 delves into the established research in the area of intrusion detection, classification methods and feature selection approaches. Chapter 3 outlines a methodology for data collection, preprocessing and analysis. Chapter 4 provides results of the experiments and an analysis of why text mining falls short of the hypothesis for this investigation. Chapter 5 summarizes this thesis' effort, details the contributions of this work, and offers recommendations for future work.

II. Literature Review

This chapter presents an overview of the existing technologies that are applied to solve the problem of increasing the accuracy of attack classification using host and network data in synergy. It provides background of the current research methodologies and approaches that identify malicious activity, either using only host data [55] or only network data [50]. It also underscores that approaches that use both are not identified in the literature. This is not to say that they do not exist as new papers are output almost on a daily basis and anyone searching this field may come across something that was not in the search list at the time this compilation was put together. The topics include a discussion of intrusion detection systems followed by an overview of the most common machine learning techniques. Finally, it concludes by a brief discussion of the machine learning techniques and features employed in this research.

2.1. Types of IDS

The following subsections will describe the types of Intrusion Detection Systems (IDS) and their functions.

2.1.1. Network Intrusion Detection Systems (NIDS)

Network-based intrusion detection systems are designed to monitor network traffic and are positioned in proximity to the firewall at the border of an intranet to the internet [48] [60] [18]. Network intrusion detection systems monitor internet packets for

suspicious content and either raises an alert about possible malicious content or in the case of active Network Intrusion Prevention, block those packets entirely.

A network-based IDS typically utilizes a network adapter running in promiscuous mode to monitor and analyze all traffic in real-time as it travels across the network [48] [60] [43]. Its attack recognition module uses four common techniques to recognize an attack signature:

- Pattern, expression or byte-code matching,
- Frequency or threshold crossing
- Correlation of lesser events
- Statistical anomaly detection
- Flow based

NIDS are vulnerable to being overwhelmed by the sheer number of packets that they are required to inspect [48]. Depending on the amount of analysis performed upon each packet, a NIDS could be incapable of running real-time for one computer without lag, yet in theory a NIDS could be tasked with a network of tens or even hundreds of computers. This issue can be relieved somewhat through optimization of the computer hosting the NIDS and by reducing the amount of analysis performed on each packet, by implementing techniques for shallow packet inspection.

The vast majority of NIDS are focused on packet headers [3] [38] [50] [37] because of the difficulties in sufficient computing resources otherwise needed in reading the packet payloads, but a rare few do exist [59]. There is a vast array of machine learning algorithms employed in analyzing packet headers which are detailed in the discussion on machine learning algorithms for IDS, Section 2.2.

One example of the creative use of NIDS research is alert correlation. Nearly all of the proposed alert correlation methods are based on syntax-oriented approaches. For example, Wei [60] exploits the semantics of attack behaviors, and presents the semantic vector space model to extract and classify the attack scenarios automatically. Wei [60] uses first order predicate logic (FOPL) and linguistics to classify DDoS computer attacks based on features derived from NIDS alert streams. FOPL is a form of automated reasoning and here, it aids in analysis of a large volume of data collected from an NIDS. Wei [60] presents a semantic vector space model that organizes the raw NIDS alerts to extract features using the common text mining term frequency and the inverse document frequency (TF/IDF) approach and categorizes the attack scenarios, improving upon current IDS alerts.

Principal-subordinate Consequence Tagging Case Grammar (PCTCG) [60] converts the aggregated NIDS alerts into uniform streams. PCTCG is based on Case Grammar, which has many advantages. First, Case Grammar structure specifies the semantic relations between a verb and its slots. Second, the Case Grammar cab is easily represented by a semantic network, which includes abundant semantic relations to express the alerts associations. Third, unlike the syntactic level, Case Grammar theory is deeply semantic, which means it does not change under grammatical transformation. PCTCG uses First-order Logic (FOL) as the alert representation and reasoning language. In simulations, the backward chaining language Prolog is used for logic programming of predicate logic. First-order attack resolution works under the Principal-subordinate relation. When one alert is in the subordinate phase and its subordinate keywords are in a

specific relationship with the principle alert, these two alerts are correlated. Based on the attack ontology and alert contexts, alerts are represented as attack semantic space vectors. Text categorization techniques are then applied to categorize the intrusion stages. In order to classify which intrusion category an alert *a* belongs to, the similarities of *a*'s semantic vector and all category vectors are measured. The alert *a* is then assigned to the category with the highest value.

In simulations, the number of alerts had decreased to 29.1%, and 12.2% respectively [60]. The correct classification rates for three intrusion stages: gather information, making enable, and launching attack are also presented. The method is fast, the reasoning time is far less than the alert inter-arrival time. Simulation results also show the scheme not only performs as well as the traditional alert correlation technique, but also facilitates intelligent semantic reasoning. There are limitations which Wei identifies as a compromise between computational power/speed and accuracy of categorization in the form of the Alert Semantic Context Window Size.

2.1.2. Host Intrusion Detection Systems

Host-based intrusion detection systems are installed directly on a computer and they monitor its processes for evidence of possible intrusions or intrusion attempts into the computer system [28]. Examples of such evidence include, but are not limited to log files, system processes, internet usage, file operations, Windows registry operations, and any correlation schemes the designer of the IDS believes will provide information about possible attacks and intrusion attempts.

Host-based intrusion detection started in the early 1980s before networks were as prevalent, complex and interconnected as they are today [24]. In this simpler environment, it was common practice to review audit logs for suspicious activity.

Host-based IDS have grown to include other technologies. One popular method for detecting intrusions checks key system files and executables via checksums at regular intervals for unexpected changes [24]. The timeliness of the response is in direct relation to the frequency of the polling interval. Finally, some products listen to port activity and alert administrators when specific ports are accessed. This type of detection brings an elementary level of network-based intrusion detection into the host-based environment

Host-based intrusion detection systems also have limitations to balance their successes. These include a lack of portability, storage requirements, and vulnerability. Host-based intrusion detection systems are usually not portable because they are designed to protect a certain type of machine with a specific operating system. For example, providing the same service to a computer running Apple OS X without significant modifications to a system written for MS Windows 7 is still very much a challenge. This limits the portability of a given host-based intrusion detection system. HIDS are also limited by storage requirements, since they are installed directly on the machine they are tasked to protect, so a trade-off must be made between the detection capabilities of the IDS and the functionality of the host machine [18]. This presents a challenge to the next generation melding of Network and Host IDS, as these will likely include the limitations imposed by the specificity of the host OS.

2.1.3. Hybrid Network Host Intrusion Detection Systems

Both network- and host-based IDS solutions have unique strengths and benefits that complement each other [24]. A next generation IDS, therefore, must include tightly integrated host and network components. Combining these two technologies will greatly improve network resistance to attacks and misuse, enhance the enforcement of security policy and introduce greater flexibility in deployment options.

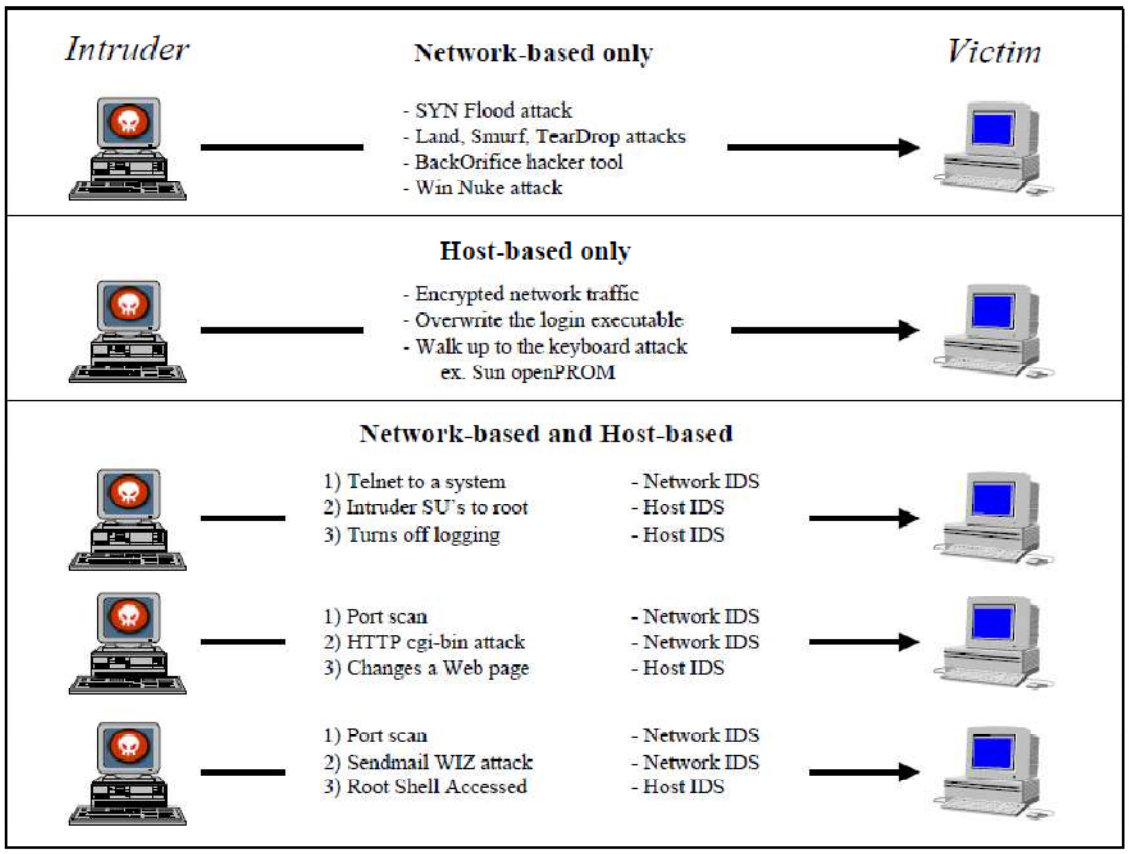

Figure 2.1: Types of IDS - ISS.net White Paper [24].

Indeed, there is significant research in the area of consolidating network security alarms into coherent incident pictures [24]. One major vein of research in intrusion report correlation is that of alert fusion, clustering similar events under a single label [58]. The

primary goal of fusion is log reduction, and in most systems similarity is based upon either attributing multiple events to a single threat agent or providing a consolidated view of a common set of events that target a single victim. The botnet infection problem does not satisfy either criteria and makes the feasibility and usefulness of such a goal questionable. The botnet infection process spans several diverse transactions that occur in multiple directions and potentially involves several active participants. A more applicable area of alert correlation research is multistage attack recognition, in which predefined scenario templates capture multiple state transition sequences that may be initiated by multiple threat agents [45]. While botnet infections do regularly follow a series of specific steps, it is rare to accurately detect all steps, and just as difficult to predict the order and time-window in which these events are recorded.

Erskine [11] used data mining techniques to create a clearer picture of the events in escalating stages of attack. His work parses the information generated by host sensors and packet captures to uncover relationships between data, leading to a fuller understanding of the critical events occurring on a computer under normal, scanning and exploit conditions while reducing the burden to the system administrator with more accurate alerts and fewer false positives. A sampling of features used include summary process data, such as memory utilization, user and system time, number of threads and handles; packet-level metrics and network volumetric data such as number of bytes sent/received per protocol; DLL counts and changes, port changes and states, process numbers and logon types, event traces and counts, just to name a few. The classification is performed with an ANN using Matlab. The process used to select these features was via manual analysis of the data

collected. In contrast, this work is based on data that more closely resembles an operational environment, and uses network features from research that is popularly cited to show proven performance in application identification [43]; this is in conjunction with text mining of host data, something that has not been studied in existing literature.

2.2. Network Intrusion Detection Datasets

Production of accurate and realistically representative training and testing sets for IDS research is a real and ongoing challenge [38]. Several standard datasets have been found to be faulty [38] and real effort needs to be put into the data collection process if results from new algorithms and approaches that claim to be improvements are to be trusted.

McHugh [38] provides a critique of the Lincoln Lab's procedures in the creation of the 1998 (and some of the 1999) IDEVAL dataset. His main criticisms included their assumptions made without corroborating evidence or descriptions, the traffic density and uniformity of the dataset, their odd ROC curves used for analysis, and their procedures for scoring IDSs that make little or no sense with respect to the intrusion detection field. In conclusion, McHugh finds the IDEVAL dataset a step in the right direction in providing intrusion detection testing data, but finds many errors in the process that make the dataset only of limited utility for testing intrusion detection systems.

Maxion and Tan [36] propose a system for benchmarking anomaly-detection systems, investigating whether the regularity of a data set influences the effectiveness of

the IDS. Artificial datasets were generated with a given entropy value between 0 and 1. Results confirm that regularity of the data set affects the effectiveness of the IDS.

The majority of research performed on intrusion detection datasets has focused on those consisting of network packet traces of packet header information. Testing of methodologies at least requires a dataset containing two types of packets: a clean dataset for training to establish a baseline and a dataset including labeled attacks for testing performance. Such a dataset is difficult to obtain due privacy laws; Google is a prime example of a public company that has paid penalties for running aground of this issue, for example, with the violation of New Zealand privacy law for its mapping project in 2010 [10]. Packet traces can contain personally identifiable information in its payload and must be scrubbed of user names and passwords as a minimum measure. This is currently a tedious process which could easily overwhelm any research effort, trying to build a program that could automate the process of filtering out users, passwords, street addresses, and phone numbers in the payloads of packets. There is also a fine line between cleaning data while preserving its usefulness. It is difficult to completely and reliably anonymize these traces without destroying the value of the packets for researchers and testers of intrusion detection systems.

Often, storing personally identifiable information is necessary to identify the perpetrator of an internet attack just as how it is necessary for law enforcement to keep a national database of fingerprints and DNA. For many IDS, even storing such information in a short term database could heavily influence the success of detection, adding the capability to correlate certain combinations of common usernames and passwords to an

intruder trying to guess access requirements as an example. The anonymity of the internet serves as both a sanctuary to those who wish to disrupt and as a hostile battleground to those seeking to prosecute illegal criminal activities.

Because most prior research has focused on either network features [3] [59] [38] [50] [37]] or host features [28] [18] [34] [50] [45], there are no publically available data collections that have network trace packets accompanied by memory images, process monitor information, or host logs of any sort let alone labeled data. With the most commonly used internet packet sets already known to be tainted [38] [36], NIDS research papers have accuracy ratings that should be questioned. Hybrid systems will be further stunted with results that will be hard to compare and reproduce if there are no publically available data sets that all researchers could test against.

Due to such severe limitations, data for this thesis was collected from a simulated exercise and may not be representative of a real world enterprise network and the actual work conducted on the network. In addition to that handicap, the red team, the NSA for this exercise, did not provide details as to the type of attack and the times they were attempted, leading to questions of accurate labeling. There was also a divergence in the mission of the exercise and the mission of this research; the exercise was a resounding success in that NSA failed to penetrate and disrupt the team's services but that also meant the collected data is lopsided in terms of having a representative set that would contain samples of both normal and infected hosts. Thus, data had to be supplemented to be able to provide additional classes using a setup that differed quite significantly from the environment and host machine in CDX. The results therefore, are highly subject to

fluctuation if tested in any other scenario. For example, in real world usage, IPSEC is cumbersome and is not widely deployed as network protection but it was the AFIT team's primary means of security so results and classification models applied on this dataset are not expected to carry across differing platforms. It is a proof of concept and relative trends are expected to hold in a variety of situations based on the fact that Moore developed his packet features generator from extensive testing on a real university network [42].

2.3. Limitations of Statistical Methods

The advantages of anomaly detection systems include lower storage requirements than a signature detection system. The library it requires for everything that is normal is orders of magnitudes smaller than the signature library, or everything that is known to be abnormal. This reduces the amount of space required by an anomaly detection system. However, the limitations of the datasets previously mentioned also have a huge affect on the efficacy of pattern matching and statistical methods. Pattern matching relies on 'learning' the probability weightings for correlation of subtle differences to models of malicious abnormal activities. There are many cases where this is not the most optimal learning strategy. Consider the simple situation of a spam filter blocking. Some spam filters will block emails that contain words or phrases that have been deemed common to spammers but that also means that a friend you know not to be spammer will get blocked even if they write an email to say, only discuss spamming, and that you actually want to read. Unless you add them to your safe list, a list that may continue to grow over the life of your account, you may miss many emails that get misclassified as spam. Machine learning methods are still many generations behind humans in determining intent, better

known as semantics. Machine learning is also very heavy on finding correlations that provide the most information gain, so that the subtlety in the variety of relationships connecting two things together becomes a concern. Say, you are in an environment where Google Chrome is the dominant browser and JavaScript is enabled. An attack is underway that is aimed at a vulnerability in JavaScript independent of browser type but what the algorithm learns is that that the attack is correlated to Chrome users because it does not understand the underlying mechanism and is fooled by the prominence of Chrome browsers in that particular environment. The JavaScript correlation is more subtle and is overwhelmed by the information gain statistics of the browser type. Thus the new alert database update that is released to the general public does not help those using Internet Explorer even though they too can be targeted and does not help users of Chrome with JavaScript disabled, generating a lot of false positives.

The process to ‘unlearn’ a bad correlation can take as long as it did to ‘learn’ it in the first place, wasting precious time and resources. While humans also learn empirically, we also have other methods, such as discovery processes like the scientific method, troubleshooting charts or sleuthing out the underlying causes of unwanted behavior, mostly avoiding the more time consuming route of unlearning by waiting for the probability weights to change. Even the most advanced Artificial Intelligence algorithm cannot yet determine when best to switch methods of problem solving so that this results in the necessity of having exceptions lists, which could very quickly grow to be unwieldy and require regular vigilant maintenance due to the pace of new applications gaining prominence and old ones growing obsolete. It does not help information security postures

that consumer electronics is a trending toward the melding of multiple features and functions onto a single device. One method of establishing baseline statistics is to develop the profile of a normal user; the normal user however becomes more complicated as the model requires more flexibility. Smart-phones now feature many additional bands of communication like 4G, WIFI, GPS and FM radio to stream video, voice, music and text. There are ever increasing operating system options (Android, Linux, Windows Mobile to name just a few) that it presently seems near impossible to distinguish what the typical profile for a user ought to be. There is a mobile computing security challenge looming on the horizon with a clear trade-off between security and convenience. A push in advancing AI research will be required as current available approaches will be hard pressed to meet those new challenges. In the meantime, human operator training for both administrators and users, and adherence to policy continues to remain one of the best defenses against cyber malfeasance. Malware authors could become craftier, using delays, time-triggered behavior, or command and control mechanisms to try to prevent malicious scripts from executing during analysis. These options indeed make it very difficult to detect and identify all threats for even the most advanced forms of intelligence, the eye of a seasoned human IT professional.

2.4. Statistical Machine Learning Method Application to IDS

The following subsections will describe some common Machine Learning Algorithms as applied to network anomaly detection.

2.4.1. Cluster Analysis

Unsupervised methods, also called clustering, group unlabeled patterns into clusters based on similarities [7]. Patterns within the same clusters are more similar to each other than they are to patterns belonging to different clusters. Data clustering is very useful when little a priori information about the data is available. Clustering methods can be classified into two categories: hierarchical clustering algorithms and partitioned clustering algorithms. The degrees of similarity are dependent on the metric selected and thus the metric must be chosen with care. For example, for an IDS clustering normal, scanning and infected data. The best features to select are those that differ based on the presence of malware and suspicious network communication. However, based on how the data is collected, it could become corrupted by differences in the network environment and host OS and applications. An unsupervised algorithm could latch on to those differences which are not pertinent to the goal of the research. Partitioning clusters split a data set into a user-defined (either a defined number or based on defined distance threshold) set of separate partitions. Hierarchical clusters split the data set into a data "tree" that begins with the complete data set and moves down to individual data.

Clustering is applicable to anomaly and intrusion detection because, in theory, 'normal' events should be more similar to other 'normal' events and 'abnormal" events should either be similar to other "abnormal" events or not similar to anything at all [22]. Based upon this principle, abnormal events (in other words intrusions and attacks) can be defined by anything that does not sufficiently resemble known types or normal events. Cluster analysis helps to detect intrusions by sorting actions, packets, files, etc. into

groups and then flagging those that are not members of any group or who for a group that does not meet a certain size threshold (assuming that normal events are common and abnormal ones are rare). Knowledge of the field that clustering is applied to is necessary as the human must input the number of clusters, as well as a threshold for separation, as well as determine realistic ratios of representative samples seen in real world situations; or the danger will be that the clustering algorithm will split groups that should be together into neighboring clusters or the opposite scenario could occur as well.

2.4.1.1. Self Organizing Maps

SOMs are a method for visualizing data of high dimensionality [59]. The output of the algorithm form clusters of similarity. Thus, it can be a way to help analyze data when knowledge of how many classifications there should be is not available beforehand.

One important component of SOM is the weight vectors or “neurons”, these vectors contain the input data as well as its location in the lattice space. Then, via a very simple algorithm, the neurons compete for which ones best represent the data:

1. Initialize Map
2. For *i* from 0 to 1
3. Randomly select a sample
4. Get best matching unit
5. Update winner and neighbors
6. Increase *i* a small amount
7. End

There are various ways to initialize the weight vectors. One could just assign initial weight vectors randomly. Calculating SOMs can be computationally expensive depending on the size of the lattice and the dimensionality of data, so there are some methods of

initializing the weights such that samples which are known to be different start off far away. This can save a significant number of iterations in order to produce a good map.

The next step is to go through all the weight vectors and calculate the distance from each weight to the chosen sample vector. The weight with the shortest Euclidean distance is the winner. If there is more than one with the same distance, the winning weight is chosen randomly among the weights with the shortest distance.

Then, update the weight vectors by determining which weights are considered neighbors and how much each weight can become more like the sample vector. The neighbors of a winning weight can be determined using a number of different methods. Some use concentric squares, others hexagons, or a Gaussian function where every point with a value above zero is considered a neighbor.

The amount of neighbors decreases over time. This is done so samples can first move to an area where they will probably be, then refine their position. The function used to decrease the radius of influence often doesn't really matter as long as it decreases, so a linear function is typical.

An attribute of this learning process is that the farther away the neighbor is from the winning vector, the less it learns. The rate at which the amount a weight can learn decreases and is typically a Gaussian function. Then as time progresses, the winning weight becomes slightly more like the sample where the maximum value of i decreases. The rate at which the amount a weight can learn falls of linearly.

2.4.1.2. Learning Vector Quantization

The basic LVQ approach is based on a standard trained SOM with input vectors and weight vectors [4]. The new factor is that the input data points have associated class information. This allows us to use the known classification labels of the inputs to find the best classification label for each weight vector. For example, by simply counting up the total number of instances of each class for the inputs within each classification cell, a new input without a class label can be assigned to the class of the cell it falls within.

The problem with this is that, in general, it is unlikely that the Voronoi cell boundaries will match up with the best possible classification boundaries, so classification generalization performance will not be optimized. The obvious solution is to shift the Voronoi cell boundaries so they better match the classification boundaries.

The LVQ algorithm is as follows:

1. If the input x and the associated weight vector w (i.e. the weight of the winning output node) have the same class label, then move them closer together by $\Delta w = \beta(t)(x - w)$ as in the SOM algorithm, where $\beta(t)$ is a learning rate that decreases with the number of iterations/epochs of training. This way we get better classification than by SOM alone.
2. If the input x and associated weight vector w have different class labels, then move them apart by $\Delta w = -\beta(t)(x - w)$.
3. Weights corresponding to other input regions are left unchanged with $\Delta w = 0$.

A second, improved, LVQ algorithm known as LVQ 2.1 is sometimes preferred because it comes closer in effect to Bayesian decision theory.

The same weight/vector update equations are used as in the standard LVQ, but they only get applied under certain conditions, namely when all three below are met:

1. The input vector x is incorrectly classified by the associated weight vector $w_I(x)$.
2. The next nearest weight vector $w_S(x)$ does give the correct classification.
3. The input vector x is sufficiently close to the decision boundary between $w_I(x)$ and $w_S(x)$.

In this case, both vectors $w_I(x)$ and $w_S(x)$ are updated (using the incorrect/correct classification update equations respectively). Other variations on this theme also exist (LVQ3, etc.).

Hendry and Yang [22] proposed a method of automatic signature creation using clustering. Data is run through a clustering algorithm that ideally forms normal and anomalous clusters. Those attributes that are shared by members of each anomalous cluster can be used as signatures to detect possible intrusions. Upon testing, the algorithm was able to correctly identify 70%-80% of malicious clusters as anomalous. Several methods of clustering have been applied to the NIDS problem, including Self-Organizing Maps [52], Principal Component Analysis [54], Y-means [20] among numerous others.

2.4.2. Artificial Neural Network Analysis

The Artificial Neural Network analysis method for intrusion detection assigns "weights" to anomalous, perhaps questionable computer usage indicators [11]. As a computer performs a series of actions due to a given user command, program, etc., these

weights are fine tuned as the computer learns from training on known, labeled suspicious or intrusive actions. Each class has its own set of weight combinations and once learned, these are used to determine which class a novel sample falls into. If deemed suspicious, the process is terminated and the proper action is taken by the detection system.

The limitations of this technique are related to the size and accuracy of the training set. System administrators and managers may perform "questionable" actions on the computer during routine maintenance and upgrading and so over time, the algorithm slowly can be trained to accept intrusive activities as normal [47], or sometimes otherwise known as drift. Radial basis function (RBF) network [63], Boltzmann machine [30], Kohonen self-organizing network [52] are just a few examples of ANNs employed in this field.

2.4.3. Non-Stationary Models

Non-stationary models are best represented by Bayesian probability models [33]. Adjustment of probability weightings are based mostly on frequency of occurrence. For example, an action that is performed daily or hourly is much more likely to occur at any given time than an action that was last performed several months or years previously. Early email SPAM prevention algorithms are a prime example of an application [40].

In non-stationary models, the probability of an action occurring is defined as the inverse of the time since the action was last performed. If an action is performed whose probability exceeds a set threshold then that action is labeled intrusive.

The main limitation of this system is that there exist legitimate actions that are only performed infrequently and a pure probability model is thus insufficient [33]. The model does not allow for any form of understanding of the underlying behaviors and mechanisms that a system administrator is trying to allow or prevent.

2.4.4. Heuristic-based Analysis

Nearly all current commercial antivirus software utilizes heuristic approaches via rule-based systems to detect malicious software that does not match an existing signature [36]. This means, that the component of the heuristic engine that conducts the analysis extracts certain rules from a file and these rules are compared against a set of rule for malicious code like a simple event correlator. If a rule matches, an alarm is triggered. A heuristic scan checks to determine if a defined series of actions is occurring together within a defined window (for example a port sweep) that would indicate that an attack is in progress.

The major drawbacks of heuristic-based analysis are the space requirements for storage of the algorithm, the tuning and maintenance to fit the host, and the CPU requirements for the scans of the heuristic algorithm [44].

2.4.5. Entropy-based Analysis

In entropy-based analysis, intrusions are detected through analysis of the uniformity of the data set [50]. As each packet, action, etc. is processed by the detection system, the entropy of the data set as a whole is recalculated. If the new packet, action, etc. is normal, it should not change the entropy of the data set significantly or even not at

all. However, if the new item is anomalous, it should significantly increase the entropy of the data set by a proportion relative to the size of the data set (the larger the data set, the less a single item will be able to affect its entropy). If an item is found to be anomalous, it is marked for proper action and removed from the data set to decrease the total entropy. Another option is to keep the data set the same size by removing the oldest item as each new normal item is added. This would allow for growth of the data set over time to match a user's potentially changing habits.

The major issues with this approach involve the uniformity of the data set in general. If the data set is not chosen to be sufficiently uniform, the entropy of the normal items alone will be large enough to mask most intrusions. However, if the data set is too well defined, even intrusions may have a low enough entropy to go undetected simply because an unimportant or constant value is chosen as the basis for the entropy measurements.

Plattner and Wagner [50] describe a methodology for the detection of large-scale scanning worm outbreak using an analysis of the entropy of IP networks. The authors state that when a worm outbreak occurs, the entropy of source IP addresses decreases (due to a few, infected addresses having a great increase in traffic) while the entropy of destination IP addresses increases (due to multiple addresses being scanned by the infected computers). Likewise, source port entropy decreases and destination port entropy increases, although this is a less reliable measure. The authors measure the entropy of a system by using the metric of the size of a compressed object as a value for basing its entropy. The authors tested their system with the Blaster and Witty worms and data from a

Swiss backbone internet operator and reported favorable results. While this system works for fast, large-scale scanning worms, it would work far less effectively for slow and stealthy worms. Other applications to network anomaly detection include [32] [29].

2.4.6. Genetic Algorithms/Immune-base Analysis

Immune-based detection algorithms mimic the biological immune system in detecting anomalies and intrusions [62]. Artificial immune systems (AISs) break data into packets, audit files, etc. into strings of a set length of (usually binary) characters. To detect intrusions, an artificial immune system trains detectors through positive or negative selection. In positive selection, detectors are trained to only complement normal occurrences and everything that does not complement a detector is assumed by the AIS to be an intrusion. In negative selection, much like antibodies, detectors are trained to not complement normal occurrences and anything that does complement a detector is assumed by the AIS to be an intrusion. Detector training requires a dataset of purely normal occurrences. The detectors are compared to the data set and any that do not fulfill the requirements are removed. Then detectors are tested against a data set of mixed known normal and anomalous traffic. The best detectors are preserved and mutated and retested until an optimal set of detectors is reached for the data set. Then detectors are deployed to detect intrusions in unlabeled data sets, labeling intrusions based upon their complementarity to detectors of a certain type (positive or negative selection). In immune-based detection, complementarity is defined as anything that complements within a set margin (or else detectors would have to perfectly complement intrusions or self, which reduces this to a signature-based detection technique).

The major limitation of immune-based analysis and of many machine learning methods is that it requires a clean training data set and a known labeled malicious data set. The only known paired data set of this type is the MIT/DARPA IDEVAL data set and that data set is rather outdated and known to contain corrupted traces and unrealistic traffic conditions. Extensive work in this area can be further found in [38] [53].

2.4.7. Data Mining Techniques

Data mining is an area of research dealing with the extraction of features from a data set, that can ultimately be used to define and classify that set and similar sets. This is a useful technique in intrusion detection and can be used in both signature-based and anomaly-based detection techniques [7]. The data mining techniques mentioned here are association rules, frequent episode rules, and classification rules.

2.4.7.1. Association Rules

Association rule mining finds interesting associations and/or correlation relationships among large set of data items [7]. Association rules show attribute value conditions that occur frequently together in a given dataset. Association rules are the simplest of the data mining techniques and uses Bayesian Probability formulas. Association rules provide information in the form of "if-then" statements. These rules are computed from the data and, unlike the if-then rules of logic, association rules are probabilistic in nature. The statistical measures related to this rule are “support” which is the fraction of the data set that fulfills both sides of the rule (both the first and the second events have occurred) and “confidence” which is the probability that the second event will happen if the first event has already happened.

In a more general sense, all “preventative" intrusion detection is based upon application of the association rule. In preventative intrusion detection, the detection system examines data for evidence that an intrusion is about to or likely to happen based upon past actions of a user or computer that the host is interfacing with. If such evidence is found, the detection system decides that an attack is likely and takes appropriate measures.

Wang, et al. [59] describe the development of a worm detection system based on detecting anomalous payloads attached to internet packets. The software scans ingress and egress packets for anomalies and correlates anomalous ingress packets with anomalous egress packets of similar purpose (for example ingress packets that target port 80 and egress packets that target port 80). Packet matching is accomplished through a threshold-based measure of longest common substring or longest common subsequence. In testing, the system seems to perform very well (100% detection no false positives), but this was only on four different worm samples. Further testing is necessary with a greater variety of worms and other anomalous (but benign) packets.

Yung [62] describes a method of detecting hackers through examination of query and response times on computers. Typically, hackers use a chain of machines (Stepping Stones such as anonymous proxies) to hide their identity while they work. This creates a delay between query and response of the attacker's machine and the eventual target machine. The author proposes that such possible attacks can be detected by monitoring query and response times for unusual delays. While this method is inexact and prone to

error, it's simple, fast and may be useful in highlighting a possible malicious connection for further examination.

2.4.7.2. Frequent Episode Rules

Frequent episode rules are an extension of association rules to apply to three events rather than two. They state that if two events A and B have already occurred, it is likely that an event C will also occur within a set period of time. Once again the statistics of support and confidence apply, but also the statistic "window width" which is the maximum period of time in which event C is expected to occur after events A and B have occurred [7]. Being able to connect sequences of events to each other brings us closer to a primitive form of behavior understanding.

Once again, the major application of frequent episode rules is in statistical analysis methods. The use of frequent episode rules would allow a network to more accurately predict the next node in a network. For example, copying text and opening a text editor would imply that the next action would be to paste the text into the editor. However, simply opening a text editor could lead to a general range of actions rather than the specific one of pasting text from the clipboard.

2.4.7.3. Classification Rules

Clustering is used to sort data into groups based on shared attributes and groups of attributes. Then, classification rules describing these clusters, in relation to the inputs, are generated. For example, a decision tree can be further developed from the classification rules.

Kanlayasiri, et al. [26] describe a methodology for detecting whether or not a scanning attack is occurring. It checks for a sequence of TCP packets from different ports on the same IP address that are more numerous than a given value and occur within a given time period. Upon testing, the authors found that this technique works perfectly on light, moderate, and heavy load on a given bandwidth.

Duffield, P. Haffner, et al. [9] constructed rules at the flow level that accurately reproduce the action of packet-level rules. An exhaustive system for translating packet rules into flow rules must leverage the correlations between the packet payload and flow header in order to mitigate the impact of losing payload information. The model and classification for packet rules is as follows: A packet rule is specified by a set of predicates that are combined through logical AND and OR operations; there are three types of predicate: flow-header (FH), packet payload (PP), and meta-information (MI) predicates. To train the machine learning algorithms, it is desired that concurrent flow and packet traces alert the same ones that Snort raises on these packets.

Adaboost is the ML algorithm this paper settles upon, it uses an L1 linear measure of simplicity that encourages sparsity, a property that is well matched to the aim of finding a small number of predicates that are closely related to the packet level rules. The data was gathered at a gateway serving hundreds of users during August–September 2005. All traffic traversing an OC-3 link attached to a border router was gathered via an optical splitter. A standard Linux box performed the role of a monitor reading packets via a DAG card. Results indicate Adaboost is able to correctly interpret (as opposed to merely mimic) many header rules by prioritizing the proper fields: the destination port, which encodes the

ICMP code and type fields, is essential to each of the ICMP rules. Predicates that require access to packet payload information, on the other hand, cannot be reproduced in a flow setting whatsoever. For payload rules to be learned in a flow setting, therefore, the corresponding flow classifier must rely on some combination of other predicates of the original Snort rule, and entirely new predicates constructed by the machine learning algorithm to describe the packets/flows matching these rules. The experiment also discovered that data drift was largely absent at a timescale of two weeks, far longer than the few minutes required for learning. Measuring how performance drifts over time is critical, as it determines how often retraining should be applied.

The advantage of the approach is that the computation complexity remains feasible at the network scale. However, the limitation here is that packet headers can be easily spoofed. If this approach indeed becomes an industry standard, it would not be hard for intruders to insert intentionally erroneous header data in order to mask more nefarious payload.

2.5. Thesis Methodology

This thesis approaches the problem of malicious activity identification by first breaking the state of the host into three categories: Normal, Scanning, and Infected. Normal is when the host is surfing the world wide net and though it may encounter bad packets, it is not being specifically targeted and all defenses are up, meaning the antivirus is operational. Scanning is data collected from the CDX exercise when the NSA was actively scanning for vulnerabilities and trying exploits that could reveal additional vulnerabilities; however, targeted servers are still up and operational. Infected is the state

of the host when a Trojan has been downloaded and confirmed by the antivirus program to have replicated itself into the file system. The host features were extracted and put into a CSV file using text mining tools available in Weka from text files derived of the following SysInternals tools polling in 15 second intervals: DLLs, Handles, LogonSessions, Netstat, Processes, PsInfo and Services. The CSV file about the network data, captured by Wireshark as PCAP files, was derived using an in-house updated version of Andrew Moore's Fullstats packet feature generator on those PCAPs, more details of which are in Chapter 4. The two files were then manually integrated by MS Excel. The final CSV file was tested and trained using a randomly proportioned 20% for training and 80% for testing purposes. The algorithm used was Weka SMO, which is a version of Support Vector Machine, and the reason it was chosen is briefly discussed below and further expounded upon in Chapter 4.

The intermediate step in this endeavor was to identify which machine learning algorithm would be used on the collected host and network data to classify them into the prescribed categories above. The three that were ultimately chosen to test were: Self Organizing Maps, Learning Vector Quantization and Support Vector Machines, which via experimentation was determined to be the best at classification. The data set used to test the performance of these algorithms were those by Dr Andrew Moore of Cambridge which can be found at the url: http://www.cl.cam.ac.uk/research/srg/netos/nprobe/data/papers/sigmetrics/

More details about the Moore set is available in Chapter 4. This was done because those sets were much smaller than the collected combination of host/network data

collected from this project and most appropriate for the aim of finding the best performing algorithm for the real data.

2.6. Summary

This chapter has presented a compilation of the background information relating to the problem of malicious activity identification either using host or network data. It is meant to reduce the possibility of replicating work that has already been done. Chapter 3 builds on Erskine's use of only 6 network features, replacing it with Moore's 248; it also takes the idea of data mining for intrusion detection from Julisch [25] by using text mining techniques on host data, and expands upon Moore's use of network data for application identification for use in identification of malicious attacks. Finally, Chapter 3 also provides further details on the host/network combination in regards to this particular research project.

III. Methodology

This chapter describes the research methodology used to evaluate of the first ever attempt at hybridizing the network and host feature data, applying a machine learning algorithm to detect network attack stages. The processed data obtained by Windows SysInternals tools [51] and Fullstats Packet Feature Generator [41] was run through the machine learning Weka suite, taking advantage of its support vector machine algorithm to classify and test effectiveness. First, the data collection process is detailed, which is followed by a description of how the collected data was formatted into CSV format and in the case of SysInternals text files, filtered for the most frequent string n-grams. Finally, Weka's role in analyzing the preprocessed data is explained. Results of the analysis are presented in Chapter 4.

3.1. Data Acquisition

The following subsections describe the process of data collection, the types of data and their significance.

3.1.1. Host Settings

Due to the difficulties in acquiring good data as touched upon in Chapters 1 and 2, the data collected from the Cyber Defense Exercise was deemed insufficient to represent the three classes being discriminated between: Normal (normal user activity), Scanning (malicious targeted scanning), and Infected (Trojan/Worm infestation). The purpose of CDX was unfortunately not aligned with the purpose of requisite data for this research. The majority of teams involved in CDX achieved their goal of keeping their services up,

playing active defense against a continuous cyber assault throughout the three and a half day exercise. This however, ran counter to the goal of collecting a good representative sample that consisted of normal activity, scanning activity, attempted exploit activity, and successful infection. In the end, the data from CDX was determined to be representative of scanning activity and attempted exploit only. Furthermore, since NSA did not release their exact activities, these two activities were combined into one category that should describe activity representative of a targeted network that hackers are purposefully scanning, trying to find vulnerabilities. The CDX data for the scanning category used in the final analysis was from the network DNS server running Windows 2003 server. A more detailed explanation of the requirements and difficulties in CDX data collection and recommendations is available in Appendix B.

Because of the lack of data gathered for the categories of Normal and Infected were separately obtained using a Dell Latitude D630 with Intel Centrino Duo running an up to date version of Windows Vista connected via a modem to the internet, a simple scenario that is universal for nearly all internet users at some point. The table below summarizes the applications running and tools for data collection. The final virus scan using AVG confirmed that a Trojan had planted itself in two file locations.

- Activity/ Category: Normal
  - Host Operating System: Windows Vista Service Pack 2
  - Applications: Windows Media Player in continuous MP3 loop, Microsoft Internet Explorer 8.0 running automatic page refresh in 2 minute intervals using www.lazywebtools.co.uk/cycler.html, AVG antivirus patched and fully running

- Data Collection Tools (sensors): Always on: Wireshark, Snort IDS with standard rule set; 15 minute intervals: Win32dd memory capture; 1 minute intervals: C:\WINDOWS\System32\drivers\etc\services; SysInternals: Listdlls, LogonSessions, Handle, Pslist, NetStat; Once per logon: PsInfo

- Activity/ Category: Infected
  - Host Operating System: Windows Vista Service Pack 2
  - Applications: Windows Media Player in continuous MP3 loop, Microsoft Internet Explorer 8.0 running automatic page refresh in 2 minute intervals using www.lazywebtools.co.uk/cycler.html, AVG anti-virus Resident Shield off
  - Data Collection Tools (sensors): Always on: Wireshark, Snort IDS with standard rule set; 15 minute intervals: Win32dd memory capture; 1 minute intervals: C:\WINDOWS\System32\drivers\etc\services; SysInternals: Listdlls, LogonSessions, Handle, Pslist, NetStat; Once per logon: PsInfo

Infection Mechanism Went down the list of IPs on http://www.malwaredomainlist.com/mdl.php until an active malicious website was located. When located, the browser safety page was triggered and AVG confirmed block of attempted download of malicious script. Then, confirmed proceed anyway in browser, AVG blocking was turned off and web page was refreshed to successfully download malicious code into host.

3.1.2. Sensors Selection

The following subsections describe the sensors used to collect raw data.

3.1.2.1. Classes of Data Sets

To capture forensic evidence to give the bigger picture, two types of sensors are used: transactional sensors and snapshot sensors. Transactional sensors capture each

transaction within the sensor's monitoring realm. Transactional sensors continuously capture data until they are halted, or until some user-defined rule is met (such as time limit or file size thresholds). An example of this type of sensor is Wireshark [44], a popular sniffer also used by Erskine to monitor a network device for inbound and outbound network packets and save the entire session into a PCAP file for later analysis. Another transactional sensor is Snort, which logs suspicious network activity in real time. Snapshot sensors capture, and provide raw or summary data about a subset of features that can be used to describe the current system state or configuration. An example of a snapshot sensor is SysInternal's pslist.exe [51], which displays a snapshot of process in execution and processor utilization. System events which start and stop between snapshots, in this case, within 15 seconds, may make no imprint on a snapshot sensor.

The purpose of the normal collection is to generate a set of data which consists only of typical user activity in a real environment. Since I am surfing the real Internet to collect this data, it's highly probable that exploits and scanning are being attempted but that the antivirus is preventing any form of compromise and the user is not noticing any anomalous behaviors. It should be pointed out that the important difference is that in the CDX environment, the scanning is much more aggressive and purposeful because the network was specifically being targeted, with the IP ranges known to the perpetrator. During this collection, the settings summarized in the two tables above, the host runs Windows media player, Internet Explorer and AVG antivirus with Resident Shield on, which protects it from downloading and becoming infected by malicious scripts. The data

set generated is used to provide a baseline forensic data set to compare with the scanning and infected collections.

The purpose of the scanning collection is to generate a set of data which consists of scanning and service enumeration events, searching for vulnerabilities in different versions of the services like telnet, ssh, http, etc. The scanning collection consists of ping, Nessus, Nmap and other similar activities NSA may have attempted on the various IP addresses and ports on the CDX network. The data set generated is used to provide information to contribute to the model that would eventually recognize early stage malicious activity and provide more accurate warnings than Snort, which reports all pings as alerts whether or not it may have only been an innocent check for connectivity.

The purpose of the infected collection is to generate a set of data which consists of data collected from a host infected with a trojan which should theoretically contain features that distinguish it from either normal or scanning, such as if the trojan is running extraneous processes or changing the volume of network traffic. http://www.malwaredomainlist.com/mdl.php contains a list of known malicious IP addresses, some of which are active, and some of which have been disabled by the webhosts. The host computer visited these IP's until an active malicious website was located. When located, the browser safety page was triggered and AVG confirmed the block of an attempted download of malicious script. Then, the host confirmed to proceed in the browser, AVG blocking was turned off and web page was refreshed to successfully download malicious code into host. Data was collected and a scan was performed at the

end to confirm the computer was indeed infected. The data set generated is used to provide a representative sample of forensic evidence generated by a trojan infection.

Each dataset consists of outputs from the seven sensors: services.exe; SysInternals: Listdlls, LogonSessions, Handle, Pslist, NetStat; PsInfo described in detail in Section 3.1.2, which generated thousands of files to parse. Weka was used to create a text ARFF format file using the command line "java -classpath ./weka.jar weka.core.converters.TextDirectoryLoader -dir ./testing/ > testing1.arff" This was immediately re-saved as a CSV file. Then the explorer application was used to open the newly generated CSV file and a filter was used to convert the text CSV file into word vector CSV files. This filter is called StringToWordVector and is located under the unsupervised, attribute folder. StringToWordVector is necessary because the Weka Support Vector Machine classifier only works with numeric values. StringToWordVector converts String attributes into a set of attributes representing word occurrence information from the text contained in the strings. The set of words (attributes) is determined by the first batch filtered (typically training data). The option for TF/IDF, Term Frequency/Inverse Document Frequency, was selected and stemming was not done. This converts the documents into vectors of numbers for later classification.

A separate feature CSV file was generated from the packet capture by Wireshark using a perl script originated by [41] Andrew Moore of Cambridge University, fullstat.v1.0.tgz (http://www.cl.cam.ac.uk/research/srg/netos/brasil/downloads/index.html). This script runs under Ubuntu 5.10 "Breezy Badger" and requires, due to deprecated functions, the installation of several additional older packages such as gcc

4.0.1-3, perl 5.8.7-5, tcpdump 3.9.1-1, the following are listed separately in the Fullstats readme but are part of tcpdump: tcpdemux, tcptrace, , tcpslice, etc. Breezy Badger's update repositories have all been closed due to the age of the OS version but should still be available for manual install using SourceForge. Appendix A summarizes the extracted features. Some of the features are correlated, in other words, not all are independent.

Finally, these two CSV files are manually concatenated; data that could not be matched was discarded so that each set of host data was matched to one set of network data. The combined file was then trained and tested using a random 20% for training and the remaining 80% for testing to determine the separability of the data for the three classes Normal, Scanning and Infected.

3.1.2.2. Details of the Snapshot Sensors

Snapshot Sensors give information about the state at the moment the data is gathered. The snapshot sensors used in data collection include Pslist, ListDLLs, logonsessions, Handle, PsInfo, NetStat and looks at the services file under C:\WINDOWS\system32\drivers\etc\services; this file contains port numbers for well known services defined by IANA, Internet Assigned Numbers Authority.

Process Overview Sensor

In order to capture summary process data, such as memory utilization, user and system time, number of threads and handles, SysInternals' pslist.exe program is used. Running this tool from the command line provides process ID, process name, user time and kernel time, sizes of Virtual Memory, Working Set, Private Virtual Memory, Private

Virtual Memory Peak and Page Faults. This tool is categorized as an example of a snapshot sensor. A snapshot is obtained every minute; the data is output to a text file from a .bat script. While pslist.exe is used in digital forensics research, little to no literature has been presented which analyzes the output using textual data mining methods in order to farm for features to use in classification. The attributes identified are thus strings or n-grams that are identified by frequency and as a downside, also subject to variation based on how the tool is formatting the output.

Process DLLs Sensor

To help with the mapping of processes, and to which dynamic-linked libraries (DLL) a process is associated with. SysInternals' ListDLLs.exe was scripted to take a "reading" every minute. This tool is another snapshot sensor. ListDLLs shows the full path names of a processes loaded modules - not just their base names. The output of this tool was sent to a text file which was later categorized into one of the three classes and textually mined along with the other SysInternals' tools' data collection.

Process to Session Sensor

To monitor the users who are logged on at a point in time and what processes were started by those users, SysInternals' logonsessions.exe program is used. It lists the currently active logon sessions and, if you specify the -p option, it reveals the processes running in each session. This is primarily useful in discovering unauthorized users that have gained access via a backdoor and are performing suspicious system calls.

Process to Handle Sensor

Handle.Exe is a utility that displays information about open handles for any process in the system. You can use it to see the programs that have a file open, or to see the object types and names of all the handles of a program. There is also a GUI-based version of this program, called Process Explorer at Sysinternals. If you do not specify any command-line parameters it will list the values of all the handles in the system that refers to open files and the names of the files. This is the information that was outputted to the text file to be analyzed in this research.

Windows OS and Hardware Sensor

PsInfo.exe is a command-line tool that gathers key information about the local or remote Windows NT/2000 system, including the type of installation, kernel build, registered organization and owner, number of processors and their type, amount of physical memory, the install date of the system, and if it's a trial version, the expiration date.

TCP / UDP connections Sensor

NetStat.exe displays active TCP connections, ports on which the computer is listening and their associated PID, Ethernet statistics, the IP routing table, IPv4 statistics (for the IP, ICMP, TCP, and UDP protocols), and IPv6 statistics (for the IPv6, ICMPv6, TCP over IPv6, and UDP over IPv6 protocols). Used without parameters, netstat displays active TCP connections. The output is saved as a text file and categorized into one of the three classes and mined for textual attributes.

CMAT Memory Dump Sensor

CMAT is a memory dump analyzer that works in nearly all versions of Windows [46]. It can output much of the information obtained via the SysInternals snapshot tools after the fact by parsing memory dumps, which were collected using Mathieu Suiche's win32dd/win64dd [56]. The information outputted includes Process Identification numbers (PIDs), applications, users, Dynamic Linked Libraries (DLL)s, mapped address spaces and registry keys information. For this research, this tool was tested but the time needed to analyze all the memory dumps was prohibitive, so due to the limitation of getting a large enough sample size, this source was omitted. In practical terms, volatile memory parsing and data gather is a large field of cyber forensics in and of itself.

3.1.2.3. Network Packet Sensor

To capture network packets being sent to the host system, WireShark, a popular sniffer, is used. All traffic captured over a three to four hour window was run thru the packet Moore's Fullstats feature generator. The protocols evaluated include all TCP, UDP, ICMP, and ARP traffic. Relevant fields captured are listed in Appendix A. There are a number of digital forensics research efforts which utilize packet capture files as reviewed in Chapter 2. The features and methodologies vary from signature detection to anomaly detection, but Haag [21] and Gonzalez [15] both focus on the elements found within packet header-level metrics. This research is novel in that it combines the textual features generated by the host sensors with the features generated by the packet flow generator to determine whether performance can be improved by the added information gain each gets from the other. The "Network packet sensor" features generated by the Full Stat ARFF creator containing the complete set of features is in Appendix A.

3.2. Data Preprocessing and CSV file generation

3.2.1. Text files

To ready the data for Weka analysis, it must first be converted to a CSV or ARFF file format. CSV was ultimately chosen because it is easily edited in MS Excel. To create the preliminary text ARFF, three empty folders are made and named with their classification name, Normal, Scanning, and Infected. Then those folders are populated with all the text files generated by the snapshot tools, each in their respective folder.

Then, the Weka Text Directory Loader command line application is used which transforms a directory of files into an ARFF file. The resultant ARFF file is saved as a CSV after conversion into a string vector. Since most classifiers in Weka cannot handle String attributes, for these learning schemes one has to process the data with appropriate filters, e.g., the StringToWordVector filter which performs a Term Frequency/Inverse Document Frequency (TF/IDF) transformation. [61] TF/IDF is a weight used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the TF/IDF weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. One of the simplest ranking functions is computed by summing the TF/IDF for each query term; many more sophisticated ranking functions are variants of this simple model.

3.2.2. Network IP Packets (PCAP) ARFF Generation

The Fullstats Attributes Generator [26] is used to create the full 266 features as for Traffic Classification. Its output is reduced to 248 features due to the discarding of confidential information from the flows, such as IP addresses in both directions, Port Information etc.

1) Firstly you need to create a list which contains the PCAP dump files. This is then passed to the flowCreator to reassemble the flows and outputs a filelist and also outputs a directory containing the reassembled flows.

```
perl flowCreator.pl dumplist
```

2) Then the filelist created by flowCreator is passed to the attributeGenerator to calculate the features. This outputs several files that contain a prefix.

```
perl attributeGenerator.pl filelist
```

3) Finally arffCreator is used to create the ARFF file by appending all the relevant features from each file created by the attributeGenerator. This outputs two ARFF and two CSV files: awmreduced and allclass.

```
bash arffCreator2.sh output_11....._
```

The arffCreator requires you to provide the prefix of the files. It also uses two additional header files to append the Weka headers to the results.

3.2.3. CSV concatenation

The two CSV files, one from the word vector CSV consisting of the host snapshot features and one from Fullstats, consisting of network features, which are further detailed in Chapter 4, are merged in MS Excel. The Fullstats generated data is unlabeled so it is up to the user to label these before merging In this situation, data was easily labeled since they were collected at different times, in scenarios specifically setup up for each class. Normal was when the Vista was online with AVG running. Infected was with a trojan downloaded and AVG off and Scanning was collected during CDX. The user must also try to closely match the time stamps of the host data and network data during the merged CSV generation process. A host dataset with a timestamp (which is the name of the text file) is matched to a set of network data estimated to be around the same time. There is a large margin of error on the order of three hours given the time asynchronization of CDX. Also the timestamp is largely lost once processed by Fullstats so the window of selection is done to the packets prior to this processing. Once the same number of samples of network data is approximately matched to samples of host data, the basic merge consists of expanding the number of features for the class label such that the features now include both host and network derived features.

Any of the various numeric Weka classifiers can be used on this CSV file to determine if the fused host-network set of generated features improves separability over host or network alone.

3.3. Weka SMO classifier Training and Testing

Weka's Support Vector Machine (SVM) implementation is name SMO. SVMs are one of the more advanced and accurate methods of data classification, however like many computational challenges, it is a trade-off of accuracy for speed [37]. SVMs are not yet primed for real time applications, especially for the high volume task of network and host data analysis. A general explanation of the theory is presented in this section.

The general idea behind SVMs [18] is that the original feature input space which is difficult to separate can be mapped to a higher-dimensional feature space where the training set becomes more easily separable. With this mapping, the discriminant function is now:

$$g(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b = \sum_{i \in \mathrm{SV}} \alpha_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b \tag{1}$$

There is really no need to know this mapping explicitly, because we only use the dot product of feature vectors in both the training and test. Thus, a kernel function is defined that corresponds to a dot product of two feature vectors which maps the samples into an expanded feature space. For example, a linear kernel function is:

$$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \tag{2}$$

Some other commonly used kernels are polynomial, Gaussian and sigmoid [18]. Unfortunately the selection of the best kernel is a trial and error process [18].

To solve for the optimal hyperplane in the linearly separable case, Lagrangian multipliers are introduced: (Lagrangian Dual Problem)

$$\text{maximize} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{3}$$

Such that $0 \le \alpha_i \le C$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$

The solution of the discriminant function is

$$g(\mathbf{x}) = \sum_{i \in \text{SV}} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b \tag{4}$$

The optimization technique then is the same as for the large margin classifier. The solution has the form:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \sum_{i \in \text{SV}} \alpha_i y_i \mathbf{x}_i \tag{5}$$

get b from $y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 = 0,$ where $\mathbf{x}_i$ is support vector.

The basic SVM algorithm is as follows:

1. Choose a kernel function
2. Choose a value for C
3. Solve the quadratic programming problem (many software packages available)
4. Construct the discriminant function from the support vectors

Multiclass SVM aims to assign labels to instances by using support vector machines, where the labels are drawn from a finite set of several elements. The dominating approach for doing so is to reduce the single multiclass problem into multiple binary classification problems. Each of the problems yields a binary classifier, which is

assumed to produce an output function that gives relatively large values for examples from the positive class and relatively small values for examples belonging to the negative class. Two common methods to build such binary classifiers are where each classifier distinguishes between (i) one of the labels to the rest (one-versus-all) or (ii) between every pair of classes (one-versus-one). Classification of new instances for one-versus-all case is done by a winner-takes-all strategy, in which the classifier with the highest output function assigns the class (it is important that the output functions be calibrated to produce comparable scores). For the one-versus-one approach, classification is done by a max-wins voting strategy, in which every classifier assigns the instance to one of the two classes, then the vote for the assigned class is increased by one vote, and finally the class with most votes determines the instance classification.

3.4. Disadvantages of the Selected Research Methodology

There is a great disadvantage in the idea behind using text mining techniques to analyze the snapshot output from a host computer. Specifically, the text processing masks the need to truly understand the behavior of the processes running at the different stages of attack. Another major disadvantage is that it is superficial and highly dependent on the sensor tools and the format in which they output the requested data. The authors of the utilities may come out with a new version that provides an additional layer or column, or takes one away. This will change the frequencies of text and word vectors dramatically even though nothing really has changed “under the hood.” This requires retraining whereas a model that is based on behavior [19] is not as easily affected by such a trivial difference. Finally, it adds a level of complexity by having what some may deem as too

many attributes/features, numbering in the thousands, because it must count each word string as an attribute . That diminishes the optimal information gain obtained from having essentially an internal and external monitor to verify what each is “seeing” on their end.

The one clear advantage of this method is that it closely mimics how people and human systems administrators deal with this data, and that is they mostly read the lists of processes, files and timestamps and then combine that information in broad generalization. They draw conclusions based on a general correlations of this information in their heads, unlike more common research proposals in the field of attack identification which is to try to only pick the fewest most information rich features and try to make a determination from those. This approach is often blindsided by the creativity of human hackers, who find new ways of infiltration rendering the selection of only those few features that worked in the past history to be easily thwarted when faced with fresh challenges.

The other topic of disadvantage is that being the hardships of obtaining good representative data. Because the data from the scanning was procured from a system on a simulated internal network with different OSs and running different applications, it should be no surprise that it easily classifies accurately into its own class. Data obtained while connected to the actual internet such as those collected for the normal and infected classes will have much greater overlap and should provide more useful insight into realistic network and host behavior.

3.5. Summary

The host and network data fusion and SVM classification approach is meant to provide results that validate a proof of concept. That using fused network/host data for a holistic collection of features performs better than the host or network features alone. All this is moving towards the goal of more accurate classification of different stages of network attacks as to be compared with the performances of Snort IDS, pure host side data and pure network side data. An understanding of research goals and hypothesis, a description of the data collection and test environments, and a description of the experiments along with assumptions and limitations, has been provided for anyone intending to duplicate the results of this study or to re-use certain parts for future work. The results of the Weka analysis can be found in Chapter 4.

IV. Experimental Results and Analysis

This chapter discusses the experimental design, and results of from testing the hybrid intrusion detection system. The experimental design covers the data sets and parameters for implementing the experiments. The Moore dataset labeled entry 02, entry 04, entry 08 and entry 09 were used to test the Machine Learning algorithms in Weka, entry 04 and 08 were used as the training/model building sets and entry 02 and 09 were used as their respective testing sets. Since the results showed similar trends, only the entry 02/04 experimental pair results are presented. The Moore dataset is used to perform a comparison test of the Self Organizing Map (SOM), Learning Vector Quantization (LVQ), and Support Vector Machine (SMO) classifiers which results in the Weka SMO as the chosen classifier for final testing. Testing the Weka SMO classifier on the CDX/Vista provided a comparison between pure HIDS (Host data set prior to merging features) and NIDS (Network data set prior to merge) with that of one form of the hybrid IDS and discusses the observed performance metrics. Lastly, the summary presents the results in context and offers a fuller perspective of where things stand.

4.1. The Moore Network Traffic Data Set and IP Feature Extractor

Moore's data set contains one day of authentic network traffic that was classified by the type of traffic, day1.TCP.arff.gz (12 classes, shown in Table 4.1) [41] [42]. The flows labeled as cyber attacks are identified by known signatures, and consist predominantly of worms and viruses. A table of the features contained in the data set from [43] appears in Appendix A. The data set provides a look into a real world application of

the feature selection problem since it is extracted from a day of network traffic and has already been processed for metrics and statistics.

The Moore data set contains 377,526 samples of network flows, 248 features, and 12 classes, whose features include nominal, discrete, continuous, missing and noisy values. The samples of the data are restricted to bidirectional Transmission Control Protocol (TCP) flows. A portion of the data set is employed for this work since the original Moore data set consists of too many network flows to handle in a reasonable amount of time and the researcher encountered unrecoverable heap space issues with the full Moore data set. The reduced data set consists of 40,858 flows out of the 377,526 flows. A majority of the flows consist of email and World Wide Web traffic and so these classes have been reduced to preserve a more equivalent ratio with respect to the other classes. The games class was removed because there were only 8 instances, and due to restrictions should not appear in enterprise network traffic. The composition by class of the original and reduced data set is shown in Table 4.1.

The features consist of protocol parameters, network performance, transmission volume, and size. The features describe many flow characteristics extracted from the network traffic. Protocol parameters include information taken directly from packet-level headers. Performance pertains to a combination of features that are affected by flow and network dynamics (e.g., throughput). Volume includes the quantity of certain distinguishing packet traits. Size encompasses features that describe the flows in terms of bytes. Table 4.2 provides several examples of features by category. The features describe host to host sub flows and aggregate bidirectional statistics and metrics. Specifically, the

features include quartile, min, max, average, and median statistics. The data set feature values are binary, whole and real numbers. Additionally, nearly a third of the values for some features are missing data.

Table 4.1: Moore dataset - Number of Instances in each Class [41].

Class	Original Data Set	Reduced Data Set
Games	8	0
Interactive	110	110
Multimedia	576	576
Attack	1,793	1,793
Peer-to-peer	2,094	2,094
Services	2,099	2,099
Database	2,648	2,648
File Transfer Protocol-passive	2,688	2,688
File Transfer Protocol-control		
File Transfer Protocol-data	3,054	3,054
Mail		
World Wide Web	5,797	5,797
	28,567	9,999
	328,092	10,000
Total	377,526	40,858

Table 4.2: Fullstats Feature Generation Example Features by Category.

Protocol Parameters	Performance	Volume	Size
stream length	inter-arrival time	number of out-of-order packets	average packet size
average window size	throughput	number of acknowledgment	total bytes sent
request for max segment size	round trip time	packets number of retransmissions	amount of control bytes set

The data set presents a complex domain with high dimensionality, varied correlation, multiple feature types and missing values. Analysis of the data sets shows that some features are redundant and/or uninformative for the classification task. Pearson's correlation coefficient is widely used in the sciences as a measure of the strength of linear dependence between two variables. Utilizing Pearson's correlation coefficient as a measure of the correlation (linear dependence) between two variables X and Y, giving a value between +1 and −1 inclusive [23] on the feature pairs as indicated in Table 4.3 show a linear association indicated by $|r| = 1$. Of the pairings, only a single member of a pair would need to be assessed for feature selection. Additionally, the data set contains features with no utility since all their values are zero or missing.

Table 4.3: Pairs of Features with Perfect Correlation.

Index A	Feature A	Index B	Feature B
6	mean IAT	198	mean IAT a b
6	mean IAT	205	mean IAT b a
198	mean IAT a b	205	mean IAT b a
7	q3 IAT	199	q3 IAT a b
7	q3 IAT	206	q3 IAT b a
199	q3 IAT a b	206	q3 IAT b a
217	Effective Bandwidth a b	218	Effective Bandwidth b a

Table 4.4: Uninformative Features.

Index	Feature
76	urgent data pkts b a
78	urgent data bytes b a
103	truncated data a b
104	truncated data b a
106	truncated packets b a
219	FFT all Frequency # 1
229	FFT a b Frequency # 1
239	FFT b a Frequency # 1

Uninformative features are noted in Table 4.4 and may also be removed but were left for completeness. Many other pairs of features contain extremely high correlations in excess of 0.99. For the 248 features, there are 30,628 possible combinations of pairings, of which there are 74 pairings with correlations greater than 0.99, and 326 with correlations greater than 0.90.

4.2. Determining the Machine Learning Algorithm

This section gives the investigation that was done to determine the best classifier to be used on the CDX and Vista network data using the Moore network dataset [42]. There are 11 sets total, entry 04 and entry 08 was used as the basis for building the SOM, LVQ 2.1/3 and SVM classifier models in Weka. Entries 02 and 09 were used as test sets to produce the results to follow. These entries were selected because they represented a smaller dataset (02 and 04) and a larger dataset (08 and 09) since one of the findings by

Kim [27] was that data set size played a significant role in classification accuracy. It compares the performance of Self Organizing Maps (SOM), Learning Vector Quantization (LVQ) version 2.1/3 and Support Vector Machines (SVM) on Andrew Moore's data set, an overview of how these algorithms work is generalized here.

4.2.1. Self Organizing Maps

The SOM parameters were the default settings in Weka using the version downloaded [31] from http://wekaclassalgos.sourceforge.net/, and the reduced data set was used as described in Table 4.1.

Table 4.5: Graphical User Interface Parameters.

Parameter	Value
Debug	FALSE
Initialization Mode	Random Training Data Proportional
Learning Function	Linear Decay
Learning Rate	0.3
Map Height	6
Map Width	8
Neighborhood Function	Gaussian
Neighborhood Size	8
Seed	1
Supervised	FALSE
Topology	Hexagonal
Training Iterations	1440
Use Voting	FALSE

Of importance in the Self Organizing Map experiment was that the false positive rate for the largest class WWW (web browsing traffic) was very high, at 73-93% testing the model on various data sets, a typical example confusion matrix is displayed in Table 4.6. Also, the size of the data set used to create the model that corresponded with the test

set of similar size generally led to better classification accuracy and this was also true of the other algorithms. Table 4.6 illustrates the confusion matrix test result of classifying entry 09 network packet flows, using the SOM model generated by training on entry 04 of the Moore data set.

Table 4.6: entry 09 confusion matrix of SOM model generated by entry 04.

WWW	MAIL	FTP-CONTROL	FTP-PASV	ATTACK	P2P	DATABASE	FTP-DATA	MULTIMEDIA	SERVICES	INTERACTIVE	GAMES	
59936	3658	65	1412	361	199	15	90	0	337	29	3	WWW
0	0	0	0	0	0	0	0	0	0	0	0	MAIL
0	0	0	0	0	0	0	0	0	0	0	0	FTP-CONTROL
0	0	0	0	0	0	0	0	0	0	0	0	FTP-PASV
0	0	0	0	0	0	0	0	0	0	0	0	ATTACK
0	0	0	0	0	0	0	0	0	0	0	0	P2P
0	0	0	0	0	0	0	0	0	0	0	0	DATABASE
0	0	0	0	0	0	0	0	0	0	0	0	FTP-DATA
57	20	10	0	6	50	0	0	0	0	0	0	MULTIMEDIA
0	0	0	0	0	0	0	0	0	0	0	0	SERVICES
0	0	0	0	0	0	0	0	0	0	0	0	INTERACTIVE
0	0	0	0	0	0	0	0	0	0	0	0	GAMES

4.2.2. Learning Vector Quantization

The LVQ version 2.1 and 3 parameters were the default settings in Weka using the reduced data set as described in Table 4.1.

Table 4.7: Graphical User Interface Parameters LVQ 2.1.

Parameter	Value
Debug	False
Initialization Mode	Random Training Data Proportional
Learning Function	Linear Decay
Learning Rate	0.3

Seed	1
Total Codebook Vectors	20
Total Training Iterations	1000
Use Voting	False
Window Size	0.3

Table 4.8: Graphical User Interface Parameters LVQ 3.

Parameter	Value
Debug	False
Epsilon	0.1
Initialization Mode	Random Training Data Proportional
Learning Function	Linear Decay
Learning Rate	0.3
Seed	1
Total Codebook Vectors	20
Total Training Iterations	1000
Use Voting	False
Window Size	0.3

Of importance about the LVQ experimental results was that LVQ 3 performed much better than LVQ 2.1 90% versus 78% respectively, their confusion matrices are displayed in Table 4.9 and 4.10 ; LVQ 3 had a higher true positive rate and lower false positive rate. For version 2.1, two best match units are selected and only updated if one belongs to the desired class and one does not, and the distance ratio is within a defined window [4]. The difference in LVQ 3 is that even if both best match units are of the correct class, they are updated but adjusted using an epsilon value (adjusted learning rate instead of the global learning rate). Another note in using Weka is to turn voting off, or else it would basically put everything into the class WWW because of its overwhelming data proportion relative to the other classes.

Table 4.9: entry 02 data set test results from LVQ 2.1 model 04

WWW	MAIL	FTP-CONTROL	FTP-PASV	ATTACK	P2P	DATABASE	FTP-DATA	MULTIMEDIA	SERVICES	INTERACTIVE	GAMES	
18487	2093	100	344	19	94	329	1226	141	220	2	0	WWW
0	0	0	0	0	0	0	0	0	0	0	0	MAIL
0	0	0	0	0	0	0	0	0	0	0	0	FTP-CONTROL
0	0	0	0	0	0	0	0	0	0	0	0	FTP-PASV
72	633	0	0	0	0	0	31	9	0	0	1	ATTACK
0	0	0	0	0	0	0	0	0	0	0	0	P2P
0	0	0	0	0	0	0	0	0	0	0	0	DATABASE
0	0	0	0	0	0	0	0	0	0	0	0	FTP-DATA
0	0	0	0	0	0	0	0	0	0	0	0	MULTIMEDIA
0	0	0	0	0	0	0	0	0	0	0	0	SERVICES
0	0	0	0	0	0	0	0	0	0	0	0	INTERACTIVE
0	0	0	0	0	0	0	0	0	0	0	0	GAMES

Table 4.10: entry 02 data set test results from LVQ 3 model 04.

WWW	MAIL	FTP-CONTROL	FTP-PASV	ATTACK	P2P	DATABASE	FTP-DATA	MULTIMEDIA	SERVICES	INTERACTIVE	GAMES		
19534	1057	94	22	217	112	8	339	50	113	2	0	WWW	
105	372	0	0	107	1	0	33	1	0	0	0	MAIL	
0	0	0	0	0	0	0	0	0	0	0	0	FTP-CONTROL	
0	0	0	0	0	0	0	0	0	0	0	0	FTP-PASV	
0	0	0	0	0	0	0	0	0	0	0	0	ATTACK	
0	0	0	0	0	0	0	0	0	0	0	0	P2P	
0	0	0	0	0	0	0	0	0	0	0	0	DATABASE	
2	0	0	0	0	1	0	112	3	0	0	0	FTP-DATA	
0	0	0	0	0	0	0	0	0	0	0	0	MULTIMEDIA	
0	0	0	0	0	0	0	0	0	0	0	0	SERVICES	
0	0	0	0	0	0	0	0	0	0	0	0	INTERACTIVE	
0	0	0	0	0	0	0	0	0	0	0	0	GAMES	

4.2.3. Support Vector Machines – Weka SMO

The SMO parameters were the default settings in Weka using the reduced data set as described in Table 4.1.

Table 4.11: SMO GUI Parameters.

Parameter	Value
Build Logistic Models	False
C	1.0
Cache Size	250007
Debug	False
Epsilon	1.0E-12
Exponent	1.0
Feature Space Normalization	False
Filter Type	Normalize Training Data
Gamma	0.01

Lower Order Terms	False
NumFolds	-1
Random Seed	1
Tolerance Parameter	0.001
Use RBF	False

SVM was clearly the superior classifier in the experimental results but processing time was a concern as was noted by Kim[37]; the accuracy was nearly 98%.

Table 4.12: entry 02 data set test results from SMO model 04.

WWW	MAIL	FTP-CONTROL	FTP-PASV	ATTACK	P2P	DATABASE	FTP-DATA	MULTIMEDIA	SERVICES	INTERACTIVE	GAMES	
18527	17	0	9	15	86	40	4	1	3	0	0	WWW
4	2697	11	0	0	0	35	0	1	3	0	0	MAIL
0	1	89	0	0	0	254	0	0	0	0	0	FTP-CONTROL
2	0	0	330	0	0	0	0	0	0	0	0	FTP-PASV
0	0	0	0	0	0	0	0	4	0	0	0	ATTACK
9	1	0	5	0	7	0	0	12	0	0	0	P2P
0	0	0	0	0	0	0	0	0	0	0	0	DATABASE
2	2	0	0	0	0	0	1253	0	1	0	0	FTP-DATA
14	7	0	0	4	1	0	0	132	0	0	1	MULTIMEDIA
1	1	0	0	0	0	0	0	0	213	2	0	SERVICES
0	0	0	0	0	0	0	0	0	0	0	0	INTERACTIVE
0	0	0	0	0	0	0	0	0	0	0	0	GAMES

4.3. Comparison of Machine Learning Methods

These results indicate that for this application, LVQ 2.1 is the worst performer, with a low true positive rate and a high false positive rate. The next would be the SOM which in Weka does not provide a means of doing forbidden magnification; this was an issue due to the nature of the data as not all the classes were even close to being

equally represented in the data. LVQ 3 did a bit better than SOM but still had very high false positive rate for the largest class. The best but most time consuming of all methods investigated was the SVM. Weka has a binary implementation called SMO, which means additional coupling and pair-wise classification and comparison steps, on the order of n choose 2 were required. SVM had the highest accuracy (96-99% depending on the data set used for test), and lowest false positive rate of all the methods investigated. WWW class still had the highest FP rate, but it was only 3.3% for the 4 model tested on data set 2 and 8.6% for the 8 model on data set 9.

This investigation is to use SVMs whenever the application doesn't require real time results. SVMs may still be feasible in the application of network security if one could reduce the number of classes which would result in a speed boost. This may be possible if one were only looking at malicious flows such that there would be a few classes for different attack types and the rest of the traffic is lumped into one "other-miscellaneous" class. But as things are, SVMs lag the rest in terms of computational and time resources. However, since real time is not the point of this thesis, it was decided that SVM aka Weka SMO would be used to classify processed data from CDX and the World Wide Web.

4.4. Hybrid Comparison of Performance to Network or Host Alone

The SMO parameters were the default settings in Weka using the combined data set containing all the 500 and 248 host and network features as to be described in this section. SMO was trained on 20% of the data set and the remainder 80% was used for testing. The parameter settings displayed are from the 3.6.2 version of Weka which is a

more recent version than the one used to determine the best machine learning algorithm used in section 4.2.

Table 4.13: SMO Parameters.

Parameter	Value
Build Logistic Models	False
C	1.0
Checks Turned Off	False
Epsilon	1.0E-12
Filter Type	Normalize Training Data
Kernel	PolyKernel –C 250007 –E 1.0
NumFolds	-1
Random Seed	1
Tolerance Parameter	0.001

It can be immediately noted that the scanning results are highly distinguishable from the normal and infected classes. Their confusion matrices are listed in Table 4.13, 4.14 and 4.15. The rows in the confusion matrix are the labels of the samples, and the columns are the classification results. Host only achieved an accuracy of 76%, Network only achieved 87% and the Hybrid achieved an accuracy of 99%. Distinguish-ability between the Scanning class and the other two classes is high through these three scenarios; but, this is not a testament to the quality and effectiveness of the classification algorithm or of the feature extractor but rather of the fact that this data was collected in a separate environment, namely the CDX network. The other two remaining classes were gathered later from the real World Wide Web and thus share many more similar features that cause the Weka SMO classifier greater confusion and thus increased misclassifications.

Table 4.14: Host Only Classification Results Confusion Matrix.

	Infected	Normal	Scanning

Infected	83	78	0
Normal	26	117	0
Scanning	0	1	138

It is safe to assume here and also in the results that follow that the percentage of correctly classified instances is actually inflated due to the artificially high accuracy of the scanning class detection. Just by averaging the Infected and Normal detection rate would yield a more representative accuracy metric of 66.7%.

Table 4.15: Network Only Classification Results

	Normal	Scanning	Infected
Normal	2672	4	876
Scanning	1	10373	0
Infected	1593	7	402

Averaging the Infected and Normal detection rate would yield a more representative accuracy metric of 73.4%. On the surface, this is clearly better than the results of the host data; however, one should consider the possibility that the type of infection, by trojan malware in this case, could leave a larger footprint or effect more statistically relevant change in network activity when compared to the host monitored activity.

Table 4.16: Hybrid Host and Network Classification Results

	Infected	Normal	Scanning
Infected	159	0	2
Normal	0	140	3
Scanning	0	1	138

Averaging the Infected and Normal detection rate would yield a more representative accuracy metric of 98.4%. This result outperformed host only classification by 31.7% and network only classification by 25%. Text mining is typically used to look at frequencies of word strings and is often used to try to identify natural language features like authors' writing style or language. Because effective HIDS depends heavily on event correlation, the text mining approach did not factor in cause and effect and looked only at the string structure. But this result is a positive indicator that something that seems as un-intuitive as textual frequencies of host data from snapshot sensors contributes to greater accuracy in malicious activity detection. It also lends credence to the hypothesis that if numerical metrics of behavioral information rather than text frequencies could be garnered, it may significantly improve detection while vastly decreasing the number of attributes; and, ergo save on processing capacities. Also, since attacks tend to originate more on the network side, greater accuracy possibly would've been achieved by placing greater bias towards the network features. The final set of features trained tested in Weka contained 500 attributes from the host data and only 248 attributes from the network data. It's conceivable that the number of attributes can be lowered on the host side and still preserve this level of performance. This is an avenue that can be considered in future work.

4.5. Hybrid Host-Network Comparison of Performance to SNORT IDS

The analysis of the host data covered approximately 15 minutes of operational time of data gathered from seven of the Syslnternals tools. The size of the data was 17.2mb. Taking a look at the SNORT alerts, there was 119 SNORT alerts that contained any reference to 10.1.30.5, which was the IP address of the DNS server contributing the CDX scanning data when the DNS was mostly likely discovered by NSA and being actively scanned. This is something that will get lost to system administrators in the sea of alerts to all the other nodes of the network. Most of the alerts are purely repetitious and thus redundant. An example is:

> [**] [1:1000001:0] Test https web activity [**]
>
> [Priority: 0]
>
> 03/02-21:16:10.393891 10.1.30.5:1313 -> 65.55.25.59:443
>
> TCP TTL:128 TOS:0x0 ID:11481 IpLen:20 DgmLen:40 DF
>
> ***A**** Seq: 0xB9C36DA0 Ack: 0x566EC7C0 Win: 0xFAF0 TcpLen: 20

The alert log contains over a hundred of this exact same alert, yet this alert reveals little information as to the true nature of what's going on between the client(s) and the DNS host.

There were 5 SNORT alerts for each of the data sets Normal and Infected and these were regarding SHELLCODE EXECUTION, attributable to the .bat scripts used to start the SysInternals sensors to collect snapshot data. Effectively, SNORT got 0% TP and FP, and fails to capture any relevant information in the VISTA experiment surfing the real World Wide Web.

Comparing SNORT performance to the performance as tested in section 4.4 is not a fair comparison. SNORT is fine grained and intentionally designed to perform on each packet or sequence of packets it sniffs; it is not meant to interpret all the alerts together as a whole to give a classification decision. However, just based on this rough description of its output on the network data, it is clearly performing a dismal job, either missing alerts or overwhelming the user. The goal is to move towards a system where a novice administrator should be able to identify a security breach as it unfolds, yet SNORT is still a system that requires high level training and experience to use effectively.

4.6. Summary

The results of this experiment confirm the hypothesis that statistical analysis using text based data mining in combination with network traffic flows is a more effective method for intrusion detection than host or network detection alone. At present, integrating host data to network data may achieve the highest effectiveness by augmenting existing event based NIDS systems like SNORT or BRO; for example, adding an interface that allows it access to relevant host data to reduce alerts from the age old rules set checking method. Trying to integrate these two vantages in a completely new platform using machine learning techniques is still a ways off from everyday practicality. Machine

learning based applications continue to be resource intensive ones. The new McAfee products called Entercept Host IPS and IntruShield Network IPS are currently the top tier available commercial products making a preliminary attempt at a more holistic approach to the enterprise network.

If one considers the roles of HIDS and NIDS in their own capacities, or of examples of attacks that only Host IDS can detect and block:

- Local Privilege Escalation Attacks
- Client Side Attacks

And, examples of attacks that only Network IDS can detect and block:

- ARP Poisoning
- Protocol Flooding
- Routing Protocol Attacks

It looks as though there will always be cases that augmentation by host data would not contribute to better detection and may in fact hamper the efficiency of malicious activity discovery. Even if future commercial products do integrate host and network data, some degree of separation may actually prove advantageous.

V. Conclusion and Future Work

The results of this thesis demonstrates that using text mining methods to extract word vector attributes from raw host data is one effective though admittedly inefficient (resource wise) method of hybridizing host and network data features. Determining the best approach to use to combine the feature attributes of host and network data proved problematic, largely due to the quality of the raw data collection environment. Specifically, the time offset of the host and network packet information was not constant throughout the CDX exercise and made it difficult to positively identify whether a certain series of packets could be linked to the timestamps on the host data collection. The associations here often had to resort to a best estimate of when one network event could be linked to another on the host side. It does dramatically outperform from what the author had anticipated once the two perspectives came together, despite some imperfect matching. Although it seems that in this one case, text mining was not intuitive or obvious in the beginning, the investigation of the methodology was ultimately worth the effort resulting in impressively tantalizing findings that merit additional research. In many existing signature based intrusion detection applications, strings and word vectors already play an important role in making the determination of malicious infection attempts, McAfee and Norton are just a few popular commercial examples. It would thus seem to make sense that adding statistical analysis to look for correlations in frequency and length of such strings might contribute to better determinations. Although not originally a goal, this thesis also uncovered a very good but obsolete and forgotten tool for extracting features from PCAP files. The vast majority of the credit goes to Dr Gilbert Peterson of the Air Force Institute of Technology for making the 2005 perl source code work for

Andrew Moore's original Fullstats attribute generator on the data gathered in 2010. The Weka analysis of the pure packet CSV was remarkably accurate in terms of having high true positives and low false negatives.

The data classifier admittedly did not have a hard time distinguishing data gathered from the CDX network versus the data gathered from the real internet. This is a definite blemish that detracts from the potential of this research and the results would have been much more trustworthy if all three classes of data could be collected from one environment all at the same time. Now, it becomes questionable as to how much of the classification was based on the differences in the host OS, IP address ranges, timestamps, names of running processes etc. when it is known that accurate classification should mostly be based on the presence of non standard protocol [27], port numbers, packet size info, and TCP header flags on the network side and changes in file i/o events or processes and their resource utilization trends on the host end.

Another negative is the infection mechanism. The best scenario would've have been to have a hacker actively targeting the host, having both successful and failed intrusions, all the while logging actions. Instead, the CDX network failed to be infiltrated and the performance of the HIDS/NIDS model is pitted against the alerts logged by Snort for the CDX data. In the case of the "Infected" data gathered later, a generic trojan was intentionally downloaded but the environment of the internet is not controlled enough as to determine if or when command and control was successfully established to a botnet.

These problems in data collection are not easy ones to mitigate. Lots of research dollars are expended in network security research and we have yet to establish a

thoroughly unbiased method of collection that accurately reflects the environment that the typical user operates in.

5.1. Limitations and Assumptions

5.1.1. Inconsistent Controlled and Uncontrolled Environment

The CDX network defense was chosen over a real network to obtain the raw data set. For CDX competition purposes, the network is isolated and activity is simulated due to the potential damaging consequences of performing network attacks on a real network with real consequences. Another hurdle preventing data collection on a real network is the need to address invasion of privacy fears, an issue that both Google and Facebook have often been portrayed in the media for being in violation of. In addition, any data collected from an operational enterprise network must be scrubbed of personally identifiable information.

Data collected from and the CDX's closed network was later found to be lacking representative samples of normal and infected activity. This necessitated a second data collection on an even more limited network. This limited network consisted of a Windows Vista host surfing the real internet. This gave the opportunity to collect a set of data with normal activity features and a set of data from a host that had a trojan infection acquired by downloading malicious scripts.

The inability to collect realistic enterprise data resulted in failures in trying to perform feature generation and selection in order to separate attack versus normal activity. Even in the controlled environment of the CDX, there were several uncontrolled elements:

it is unknown what were the actual attack exploit attempts used by the NSA and at what time or sequence they occurred. The NSA failed to infiltrate our servers and any abnormal behaviors we noticed like the inability to reach a certain web address, was not linked to any alerts generated by our host event logs nor attributable to any SNORT alerts.

Categories of exploit activity are limited to the accuracy of alerts generated by SNORT IDS, and do not represent the totality of activities which can be executed against a host machine. Collections were made from multiple servers in the AFIT1 network. Due to our unique setup, resulting feature selection and classification results is considered to hold true only for the combination of operating system (Windows Server 2003), selected sensors, and server configuration used here. However, since this thesis applies a methodology which can be extended, it serves as a proof of concept versus a deliverable IDS platform. Additional collections across many systems will include varied events which will likely produce results of lower accuracy but should be similar in trend.

5.1.2. Sensor Impact to System

The selected methodology involves the collection of a live system's internal forensic data, which means that sensor tools must minimally interfere with server functions. While consideration was given to select more lightweight sensors over resource intensive ones, sensor activity still impacts collections. First, the system being monitored is less responsive while running these additional programs to sense the host environment. Second, collected data reflects information that includes evidence of sensor activity, and modify the host systems. These facts should be taken into account during later stages of

forensic analysis so as not to mistake the impact of these tools for that of a successful intrusion where none exists.

5.1.3. Partial Observable Environment

Due to the size of the search space, the limitations of processing power and the sensors selected, the host environment is only partially observable. Given this constraint, care is taken to choose sensors which each provide an uncorrelated subset of the search space in order maximize what can be gained from the collected forensic evidence. Optimality conditions for sensor selection include consistency, completeness, speed, and resource overhead. While no formal method is used to select sensors for observing the target environment, these conditions were taken into account, as was standardization; data commonly used by other researchers in this field was collected here as well.

5.2. Contributions

This work provides several contributions to the field of intrusion detection and Machine Learning. First, it shows that the current focus on just network data for malicious activity classification can be given a huge boost by including some form of host performance and state information. The results compare the novel approach of integrating host and network features and applying support vector machine for the detection and classification of three attack stages. Indications are that the best way to improve is to focus on where the highest accuracy is already being achieved, at the network side.

Future research should be on finding and augmenting the network features with the most relevant host features so that one can move away from the 500 string attributes used

here and toward something more compact that can feasibly be used for real time analysis. Second, deficiencies for data collection are specifically identified and suggestions are made for the need to standardize or develop a standard process akin to the "scientific method," for data collection that can uniquely describe the host state and associated network activity.

Lastly, an updated Fullstats attribute generator is provided to aid future work in network packet analysis. A meticulous background search revealed no other existing tool that can pull as many features from a PCAP file. Further development of this tool to add a GUI interface or to integrate into Weka or MATLAB could greatly aid this field of research.

5.3. Recommendations for Future Work

The highest priority task for future work is to produce a labeled data set that contains a broad continuum of attack stages would be ideal and both host and network data need to be gathered together in the same environment. A methodology for this collection should be developed so that data that must be collected at different times and in different environments can still be compared.

Also of importance, is to identify an auxiliary method for associating host and network data. Timestamps are not one-to-one associations and better "triggers" are needed.

An application that can deal with the high volume of data in "real time" and generate features "on the fly" that could compress the amount of analytical data would be

highly sought after. At the start of this research, the initial time hog was thought to be running the machine learning algorithm. This proved true but additionally, both feature generation and large file transfers took many more hours than anticipated. Research that can bring tools for post mortem forensics into live action would greatly complement the existing means that System Administrators have to identify network breaches.

Further, as mentioned in the Section 5.2, a periodic maintenance update to the Fullstats attribute generator is needed. Since there really is no other tool that can pull as many features from a PCAP file, such an application is valuable in the search for the most important features or combinations of features which could significantly lessen the processing time or burden for classification algorithms. Developing this tool to add a GUI interface or to integrate into Weka or MATLAB could open this field up to other seasoned researchers or novice investigators.

Lastly, due to the explosion in bandwidth of cell phone networks and large area WI-MAX, focus should be shifting to making intrusion detection tools more ubiquitous and able to function on a variety of mobile devices. Thus, it needs to be determined what features are most important for the host if it is a wireless media device that may contain other channels of communication such as 3G/4G, GPS or satellite radio. As communication starts pushing the barriers beyond IP packets, security becomes an even greater challenge to obtain.

Appendix A. Discriminators and Definitions

Number	Short	Long
1	Server Port	Port Number at server; we can establish server and client ports as we limit ourselves to flows for which we see the initial connection set-up.
2	Client Port	Port Number at client
3	min IAT	Minimum packet inter-arrival time for all packets of the flow (considering both directions).
4	q1 IAT	First quartile inter-arrival time
5	med IAT	Median inter-arrival time
6	mean IAT	Mean inter-arrival time
7	q3 IAT	Third quartile packet inter-arrival
8	max IAT	time Maximum packet inter-arrival time
9	var IAT	Variance in packet inter-arrival time
10	min data wire	Minimum of bytes in (Ethernet) packet, using the size of the packet on the wire.
11	q1 data wire	First quartile of bytes in (Ethernet) packet
12	med data wire	Median of bytes in (Ethernet) packet
13	mean data wire	Mean of bytes in (Ethernet) packet
14	q3 data wire	Third quartile of bytes in (Ethernet)
15	max data wire	packet Maximum of bytes in (Ethernet) packet
16	var data wire	Variance of bytes in (Ethernet) packet
17	min data ip	Minimum of total bytes in IP packet, using the size of payload declared by the IP packet
18	q1 data ip	First quartile of total bytes in IP packet
19	med data ip	Median of total bytes in IP packet
20	mean data ip	Mean of total bytes in IP packet
21	q3 data ip	Third quartile of total bytes in IP
22	max data ip	packet Maximum of total bytes in IP packet
23	var data ip	Variance of total bytes in IP packet
24	min data control	Minimum of control bytes in packet, size of the (IP/TCP) packet header
25	q1 data control	First quartile of control bytes in packet
26	med data control	Median of control bytes in packet
27	mean data control	Mean of control bytes in packet
28	q3 data control	Third quartile of control bytes in
29	max data control	packet Maximum of control bytes in packet
30	var data control	Variance of control bytes packet
31	total packets a b	The total number of packets seen (client→server).
32	total packets b a	” (server→client)

Continued on next page

Number	Short	Long
33	ack pkts sent a b	The total number of ack packets seen (TCP segments seen with the ACK bit set) (client→server).
34	ack pkts sent b a	” (server→client)
35	pure acks sent a b	The total number of ack packets seen that were not piggy-backed with data (just the TCP header and no TCP data payload) and did not have any of the SYN/FIN/RST flags set (client→server)
36	pure acks sent b a	” (server→client)
37	sack pkts sent a b	The total number of ack packets seen carrying TCP SACK [6] blocks (client→server)
38	sack pkts sent b a	” (server→client)
39	dsack pkts sent a b	The total number of sack packets seen that carried duplicate SACK (D-SACK) [7] blocks. (client→server)
40	dsack pkts sent b a	” (server→client)
41	max sack blks/ack a b	The maximum number of sack blocks seen in any sack packet. (client→server)
42	max sack blks/ack b a	” (server→client)
43	unique bytes sent a b	The number of unique bytes sent, i.e., the total bytes of data sent excluding retransmitted bytes and any bytes sent doing window probing. (client→server)
44	unique bytes sent b a	” (server→client)
45	actual data pkts a b	The count of all the packets with at least a byte of TCP data payload. (client→server)
46	actual data pkts b a	” (server→client)
47	actual data bytes a b	The total bytes of data seen. Note that this includes bytes from retransmissions / window probe packets if any. (client→server)
48	actual data bytes b a	” (server→client)
49	rexmt data pkts a b	The count of all the packets found to be retransmissions. (client→server)
50	rexmt data pkts b a	” (server→client)
51	rexmt data bytes a b	The total bytes of data found in the retransmitted packets. (client→server)
52	rexmt data bytes b a	” (server→client)
53	zwnd probe pkts a b	The count of all the window probe packets seen. (Window probe packets are typically sent by a sender when the receiver last advertised a zero receive window, to see if the window has opened up now). (client→server)
54	zwnd probe pkts b a	” (server→client)
55	zwnd probe bytes a b	The total bytes of data sent in the window probe packets. (client→server)
56	zwnd probe bytes b a	” (server→client)
57	outoforder pkts a b	The count of all the packets that were seen to arrive out of order. (client→server)

Continued on next page

Number	Short	Long
58	outoforder pkts b a	” (server→client)
59	pushed data pkts a b	The count of all the packets seen with the PUSH bit set in the TCP header. (client→server)
60	pushed data pkts b a	” (server→client)
61	SYN pkts sent a b	The count of all the packets seen with the SYN bits set in the TCP header respectively (client→server) The
62	FIN pkts sent a b	count of all the packets seen with the FIN bits set in the TCP header respectively (client→server)
63	SYN pkts sent b a	The count of all the packets seen with the SYN bits set in the TCP header respectively (server→client) The
64	FIN pkts sent b a	count of all the packets seen with the FIN bits set in the TCP header respectively (server→client)
65	req 1323 ws a b	If the endpoint requested Window Scaling/Time Stamp options as specified in RFC 1323[8] a ‘Y’ is printed on the respective field. If the option was not requested, an ‘N’ is printed. For example, an “N/Y” in this field means that the window-scaling option was not specified, while the Time-stamp option was specified in the SYN segment. (client→server)
66	req 1323 ts a b	. . .
67	req 1323 ws b a	If the endpoint requested Window Scaling/Time Stamp options as specified in RFC 1323[8] a ‘Y’ is printed on the respective field. If the option was not requested, an ‘N’ is printed. For example, an “N/Y” in this field means that the window-scaling option was not specified, while the Time-stamp option was specified in the SYN segment. (client→server)
68	req 1323 ts b a	. . .
69	adv wind scale a b	The window scaling factor used. Again, this field is valid only if the connection was captured fully to include the SYN packets. Since the connection would use window scaling if and only if both sides requested window scaling [8], this field is reset to 0 (even if a window scale was requested in the SYN packet for this direction), if the SYN packet in the reverse direction did not carry the window scale option. (client→server)
70	adv wind scale b a	” (server→client)
71	req sack a b	If the end-point sent a SACK permitted option in the SYN packet opening the connection, a ‘Y’ is printed; otherwise ‘N’ is printed. (client→server)
72	req sack b a	” (server→client)
73	sacks sent a b	The total number of ACK packets seen carrying SACK information. (client→server)
74	sacks sent b a	” (server→client)

Continued on next page

Number	Short	Long
75	urgent data pkts a b	The total number of packets with the URG bit turned on in the TCP header. (client→server)
76	urgent data pkts b a	” (server→client)
77	urgent data bytes a b	The total bytes of urgent data sent. This field is calculated by summing the urgent pointer offset values found in packets having the URG bit set in the TCP header. (client→server)
78	urgent data bytes b a	” (server→client)
79	mss requested a b	The Maximum Segment Size (MSS) requested as a TCP option in the SYN packet opening the connection. (client→server)
80	mss requested b a	” (server→client)
81	max segm size a b	The maximum segment size observed during the lifetime of the connection. (client→server)
82	max segm size b a	” (server→client)
83	min segm size a b	The minimum segment size observed during the lifetime of the connection. (client→server)
84	min segm size b a	” (server→client)
85	avg segm size a b	The average segment size observed during the lifetime of the connection calculated as the value reported in the actual data bytes field divided by the actual data pkts reported. (client→server)
86	avg segm size b a	” (server→client)
87	max win adv a b	The maximum window advertisement seen. If the connection is using window scaling (both sides negotiated window scaling during the opening of the connection), this is the maximum window-scaled advertisement seen in the connection. For a connection using window scaling, both the SYN segments opening the connection have to be captured in the dumpfile for this and the following window statistics to be accurate. (client→server)
88	max win adv b a	” (server→client)
89	min win adv a b	The minimum window advertisement seen. This is the minimum window-scaled advertisement seen if both sides negotiated window scaling. (client→server)
90	min win adv b a	” (server→client)
91	zero win adv a b	The number of times a zero receive window was advertised. (client→server)
92	zero win adv b a	” (server→client)
93	avg win adv a b	The average window advertisement seen, calculated as the sum of all window advertisements divided by the total number of packets seen. If the connection endpoints negotiated window scaling, this average is calculated as the sum of all window-scaled advertisements divided by the number of window-scaled packets seen. Note that in the window-scaled case, the window advertisements in the SYN packets are excluded since the SYN packets themselves cannot have their window advertisements scaled, as per RFC 1323 [8]. (client→server)

Number	Short	Long
94	avg win adv b a	” (server→client)
95	initial window-bytes a b	The total number of bytes sent in the initial window i.e., the number of bytes seen in the initial flight of data before receiving the first ack packet from the other endpoint. Note that the ack packet from the other endpoint is the first ack acknowledging some data (the ACKs part of the 3-way handshake do not count), and any retransmitted packets in this stage are excluded. (client→server)
96	initial window-bytes b a	” (server→client)
97	initial window-packets a b	The total number of segments (packets) sent in the initial window as explained above. (client→server)
98	initial window-packets b a	” (server→client)
99	ttl stream length a b	The Theoretical Stream Length. This is calculated as the difference between the sequence numbers of the SYN and FIN packets, giving the length of the data stream seen. Note that this calculation is aware of sequence space wrap-arounds, and is printed only if the connection was complete (both the SYN and FIN packets were seen). (client→server)
100	ttl stream length b a	” (server→client)
101	missed data a b	The missed data, calculated as the difference between the ttl stream length and unique bytes sent. If the connection was not complete, this calculation is invalid and an “NA” (Not Available) is printed. (client→server)
102	missed data b a	” (server→client)
103	truncated data a b	The truncated data, calculated as the total bytes of data truncated during packet capture. For example, with tcpdump, the snaplen option can be set to 64 (with -s option) so that just the headers of the packet (assuming there are no options) are captured, truncating most of the packet data. In an Ethernet with maximum segment size of 1500 bytes, this would amount to truncated data of 1500 64 = 1436bytes for a packet. (client→server)
104	truncated data b a	” (server→client)
105	truncated packets a b	The total number of packets truncated as explained above. (client→server)
106	truncated packets b a	” (server→client)
107	data xmit time a b	Total data transmit time, calculated as the difference between the times of capture of the first and last packets carrying non-zero TCP data payload. (client→server)

Continued on next page

Number	Short	Long
108	data xmit time b a	” (server→client)
109	idletime max a b	Maximum idle time, calculated as the maximum time between consecutive packets seen in the direction. (client→server)
110	idletime max b a	” (server→client)
111	throughput a b	The average throughput calculated as the unique bytes sent divided by the elapsed time i.e., the value reported in the unique bytes sent field divided by the elapsed time (the time difference between the capture of the first and last packets in the direction). (client→server)
112	throughput b a	” (server→client)
113	RTT samples a b	The total number of Round-Trip Time (RTT) samples found. tcptrace is pretty smart about choosing only valid RTT samples. An RTT sample is found only if an ack packet is received from the other endpoint for a previously transmitted packet such that the acknowledgment value is 1 greater than the last sequence number of the packet. Further, it is required that the packet being acknowledged was not retransmitted, and that no packets that came before it in the sequence space were retransmitted after the packet was transmitted. Note : The former condition invalidates RTT samples due to the retransmission ambiguity problem, and the latter condition invalidates RTT samples since it could be the case that the ack packet could be cumulatively acknowledging the retransmitted packet, and not necessarily ack-ing the packet in question. (client→server)
114	RTT samples b a	” (server→client)
115	RTT min a b	The minimum RTT sample seen. (client→server)
116	RTT min b a	” (server→client)
117	RTT max a b	The maximum RTT sample seen. (client→server)
118	RTT max b a	” (server→client)
119	RTT avg a b	The average value of RTT found, calculated straightforward-ly as the sum of all the RTT values found divided by the total number of RTT samples. (client→server)
120	RTT avg b a	” (server→client)
121	RTT stdv a b	The standard deviation of the RTT samples. (client→server)
122	RTT stdv b a	” (server→client)
123	RTT from 3WHS a b	The RTT value calculated from the TCP 3-Way Hand-Shake (connection opening) [9], assuming that the SYN packets of the connection were captured. (client→server)

Continued on next page

Number	Short	Long
124	RTT from 3WHS b a	” (server→client)
125	RTT full sz smpls a b	The total number of full-size RTT samples, calculated from the RTT samples of full-size segments. Full-size segments are defined to be the segments of the largest size seen in the connection. (client→server)
126	RTT full sz smpls b a	” (server→client)
127	RTT full sz min a b	The minimum full-size RTT sample. (client→server)
128	RTT full sz min b a	” (server→client)
129	RTT full sz max a b	The maximum full-size RTT sample. (client→server)
130	RTT full sz max b a	” (server→client)
131	RTT full sz avg a b	The average full-size RTT sample. (client→server)
132	RTT full sz avg b a	” (server→client)
133	RTT full sz stdev a b	The standard deviation of full-size RTT samples. (client→server)
134	RTT full sz stdev b a	” (server→client)
135	post-loss acks a b	The total number of ack packets received after losses were detected and a retransmission occurred. More precisely, a post-loss ack is found to occur when an ack packet acknowledges a packet sent (acknowledgment value in the ack pkt is 1 greater than the packet’s last sequence number), and at least one packet occurring before the packet acknowledged, was retransmitted later. In other words, the ack packet is received after we observed a (perceived) loss event and are recovering from it. (client→server)
136	post-loss acks b a	” (server→client)
137	segs cum acked a b	The count of the number of segments that were cumulatively acknowledged and not directly acknowledged. (client→server)
138	segs cum acked b a	” (server→client)
139	duplicate acks a b	The total number of duplicate acknowledgments received. (client→server)
140	duplicate acks b a	” (server→client)
141	triple dupacks a b	The total number of triple duplicate acknowledgments received (three duplicate acknowledgments acknowledging the same segment), a condition commonly used to trigger the fast-retransmit/fast-recovery phase of TCP. (client→server)
142	triple dupacks b a	” (server→client)
143	max # retrans a b	The maximum number of retransmissions seen for any segment during the lifetime of the connection. (client→server)

Continued on next page

Number	Short	Long
144	max # retrans b a	” (server→client)
145	min retr time a b	The minimum time seen between any two (re)transmissions of a segment amongst all the retransmissions seen. (client→server)
146	min retr time b a	” (server→client)
147	max retr time a b	The maximum time seen between any two (re)transmissions of a segment. (client→server)
148	max retr time b a	” (server→client)
149	avg retr time a b	The average time seen between any two (re)transmissions of a segment calculated from all the retransmissions. (client→server)
150	avg retr time b a	” (server→client)
151	sdv retr time a b	The standard deviation of the retransmission-time samples obtained from all the retransmissions. (client→server)
152	sdv retr time b a	” (server→client)
153	min data wire a b	Minimum number of bytes in (Ethernet) packet (client→server)
154	q1 data wire a b	First quartile of bytes in (Ethernet) packet
155	med data wire a b	Median of bytes in (Ethernet) packet
156	mean data wire a b	Mean of bytes in (Ethernet) packet
157	q3 data wire a b	Third quartile of bytes in (Ethernet) packet
158	max data wire a b	Maximum of bytes in (Ethernet) packet
159	var data wire a b	Variance of bytes in (Ethernet) packet
160	min data ip a b	Minimum number of total bytes in IP packet
161	q1 data ip a b	First quartile of total bytes in IP packet
162	med data ip a b	Median of total bytes in IP packet
163	mean data ip a b	Mean of total bytes in IP packet
164	q3 data ip a b	Third quartile of total bytes in IP packet
165	max data ip a b	Maximum of total bytes in IP packet
166	var data ip a b	Variance of total bytes in IP packet
167	min data control a b	Minimum of control bytes in packet
168	q1 data control a b	First quartile of control bytes in packet
169	med data control a b	Median of control bytes in packet
170	mean data control a b	Mean of control bytes in packet
171	q3 data control a b	Third quartile of control bytes in packet
172	max data control a b	Maximum of control bytes in packet
173	var data control a b	Variance of control bytes packet
174	min data wire b a	Minimum number of bytes in (Ethernet) packet (server→client)
175	q1 data wire b a	First quartile of bytes in (Ethernet) packet
176	med data wire b a	Median of bytes in (Ethernet) packet
177	mean data wire b a	Mean of bytes in (Ethernet) packet

Continued on next page

Number	Short	Long
178	q3 data wire b a	Third quartile of bytes in (Ethernet) packet
179	max data wire b a	Maximum of bytes in (Ethernet) packet
180	var data wire b a	Variance of bytes in (Ethernet) packet
181	min data ip b a	Minimum number of total bytes in IP packet
182	q1 data ip b a	First quartile of total bytes in IP packet
183	med data ip b a	Median of total bytes in IP packet
184	mean data ip b a	Mean of total bytes in IP packet
185	q3 data ip b a	Third quartile of total bytes in IP packet
186	max data ip b a	Maximum of total bytes in IP packet
187	var data ip b a	Variance of total bytes in IP packet
188	min data control b a	Minimum of control bytes in packet
189	q1 data control b a	First quartile of control bytes in packet
190	med data control b a	Median of control bytes in packet
191	mean data control b a	Mean of control bytes in packet
192	q3 data control b a	Third quartile of control bytes in packet
193	max data control b a	Maximum of control bytes in packet
194	var data control b a	Variance of control bytes packet
195	min IAT a b	Minimum of packet inter-arrival time (client→server)
196	q1 IAT a b	First quartile of packet inter-arrival time
197	med IAT a b	Median of packet inter-arrival time
198	mean IAT a b	Mean of packet inter-arrival time
199	q3 IAT a b	Third quartile of packet inter-arrival time
200	max IAT a b	Maximum of packet inter-arrival time
201	var IAT a b	Variance of packet inter-arrival time
202	min IAT b a	Minimum of packet inter-arrival time (server→client)
203	q1 IAT b a	First quartile of packet inter-arrival time
204	med IAT b a	Median of packet inter-arrival time
205	mean IAT b a	Mean of packet inter-arrival time
206	q3 IAT b a	Third quartile of packet inter-arrival time
207	max IAT b a	Maximum of packet inter-arrival time
208	var IAT b a	Variance of packet inter-arrival time
209	Time since last connection	Time since the last connection between these hosts The
210	No. transitions bulk/trans	number of transitions between transaction mode and bulk transfer mode, where bulk transfer mode is defined as the time when there are more than three successive packets in the same direction without any packets carrying data in the other direction
211	Time spent in bulk	Amount of time spent in bulk transfer mode
212	Duration	Connection duration
213	% bulk	Percent of time spent in bulk transfer
214	Time spent idle	The time spent idle (where idle time is the accumulation of all periods of 2 seconds or greater when no packet was seen in either direction)

Continued on next page

Number	Short	Long
215	% idle	Percent of time spent idle
216	Effective Bandwidth	Effective Bandwidth based upon entropy [10] (both directions)
217	Effective Bandwidth a b	” (client→server)
218	Effective Bandwidth b a	” (server→client)
219	FFT all	FFT of packet IAT (arctan of the top-ten frequencies ranked by the magnitude of their contribution) (all traffic) (Frequency #1)
220	FFT all	” (Frequency #2)
221	FFT all	” . . .
222	FFT all	” . . .
223	FFT all	” . . .
224	FFT all	” . . .
225	FFT all	” . . .
226	FFT all	” . . .
227	FFT all	” . . .
228	FFT all	” (Frequency #10)
229	FFT a b	FFT of packet IAT (arctan of the top-ten frequencies ranked by the magnitude of their contribution) (client→server) (Frequency #1)
230	FFT a b	” (Frequency #2)
231	FFT a b	” . . .
232	FFT a b	” . . .
233	FFT a b	” . . .
234	FFT a b	” . . .
235	FFT a b	” . . .
236	FFT a b	” . . .
237	FFT a b	” . . .
238	FFT b a	” (Frequency #10)
239	FFT b a	FFT of packet IAT (arctan of the top-ten frequencies ranked by the magnitude of their contribution) (server→client) (Frequency #1)
240	FFT b a	” (Frequency #2)
241	FFT b a	” . . .
242	FFT b a	” . . .
243	FFT b a	” . . .
244	FFT b a	” . . .
245	FFT b a	” . . .
246	FFT b a	” . . .
247	FFT b a	” . . .
248	FFT b a	” (Frequency #10)
249	Classes	Application class, as assigned in [1]

Appendix B. Notes from CDX data collection

1. All servers and workstations were on Virtual Machines, AFIT2 had them in ESXi and had no access to portable USB Drives to get extra storage or transport data.

2. Virtual machines also had trouble with recognizing CD drives sometimes.

3. All tools had to be run as Admin. These were from the Windows SysInternals suite: services, processes, handle, netstat, listDLLs, logonsessions and psInfo.

4. The memory tool windd requires a .sys file to be run in batch mode and that file must be stored in the directory indicated by typing "sc qc win32dd" or "sc qc win64dd"

5. windd must be run and memory dump saved in the same directory as where the executable is stored.

6. The memory dump took 4-6 minutes to complete running on low priority, size of dumps ranged from 700mb-2gb.

7. The memory dump process cannot be killed once started except by a hard shutdown.

8. The Task Scheduler is disabled on user workstations by group policy or else it would've been possible to run tools without a user logged on. In Vista and higher, it can even hide the cmd prompt to keep it from interfering with server screen space.

9. The naming of files according to time did not always work for the memory dump. Whenever the hour was in the single digits, the file would not record the time correctly. This error made the files overwrite each other from 12am to 10am (0100 to 1000 hours).

10. Firewall was implemented by OpenBSD with PF, SNORT was also used on this machine.

11. IPSEC was used to limit ports for communication between servers and workstations, this severely limited the actions that could cause servers to be compromised.

12. KIWI Syslog was used as the software for the event logging server. Snare was used as the agent for collecting and directing logs from networked windows machines.

13. Time was not synchronized to the network in a consistent manner and affect the differential between the network packet timestamps and host data timestamps.

User friendly considerations:

If Users are only allowed by policy to log on one at a time, not switch, memory dump interferes with the ability to log off.

If allotted virtual RAM < 1GB, windd will freeze or blue-screen the machine OS.

Memory dump interferes with reboot and/or log off. If the situation is to defend against network attack, data collection can interfere with responsive actions.

Exchange and Domain Controller most sensitive to environmental changes and these tools should only be added if no other changes will be made to the settings on these machines.

Admins can feel insecure about creating other admin accounts, I could not convince server admins to give me another account to be able to frequently monitor these tools. They had to log me in and out, made it difficult when they had higher priorities.

After the exercise started, the passwords were changed on a daily basis and there were not enough copies made of the new list to easily access. They were so complicated that even the admins were unable to memorize them.

Appendix C. List of Software Tools

These tools will be provided as a copy on a supplied external Hard Drive.

A batch script to automate host data collection:

```
@echo off

:loop


echo Starting Dlls to
"F:\F_data\Dlls\dlls_%date:~12,2%_%date:~4,2%_%date:~7,2%_%time:~0,2%_%time:
~3,2%_%time:~6,2%.txt"

echo.

echo Press [CTRL]-C to exit

F:\ForensicScan\Toolkit\Listdlls.exe  >>
"F:\F_data\Dlls\dlls_%date:~12,2%_%date:~4,2%_%date:~7,2%_%time:~0,2%_%time:
~3,2%_%time:~6,2%.txt"


echo Starting LoggedOn to
"C:\F_data\LogonSessions\LoggedOn_%date:~12,2%_%date:~4,2%_%date:~7,2%_%ti
me:~0,2%_%time:~3,2%_%time:~6,2%.txt"

echo.

echo Press [CTRL]-C to exit

F:\ForensicScan\Toolkit\logonsessions.exe  >>
"F:\F_data\LogonSessions\LoggedOn_%date:~12,2%_%date:~4,2%_%date:~7,2%_%tim
e:~0,2%_%time:~3,2%_%time:~6,2%.txt"


echo Starting Handles to
"F:\F_data\Handles\handles_%date:~12,2%_%date:~4,2%_%date:~7,2%_%time:~0,2%_
%time:~3,2%_%time:~6,2%.txt"

echo.
```

```
echo Press [CTRL]-C to exit

F:\ForensicScan\Toolkit\handle.exe  >>
"F:\F_data\Handles\handles_%date:~12,2%_%date:~4,2%_%date:~7,2%_%time:~0,2%_
%time:~3,2%_%time:~6,2%.txt"


netstat -o -n -a >
"F:\F_data\Netstat\netstat_%date:~12,2%_%date:~4,2%_%date:~7,2%_%time:~0,2%_%
time:~3,2%_%time:~6,2%.txt"


echo Starting process snapshots to
"F:\F_data\Processes\processes_%date:~12,2%_%date:~4,2%_%date:~7,2%_%time:~0,2
%_%time:~3,2%_%time:~6,2%.pslistx"

echo.

echo Press [CTRL]-C to exit

F:\ForensicScan\Toolkit\pslist.exe -x >>
"F:\F_data\Processes\processes_%date:~12,2%_%date:~4,2%_%date:~7,2%_%time:~0,2
%_%time:~3,2%_%time:~6,2%.pslistx"


type C:\WINDOWS\System32\drivers\etc\services >
"F:\F_data\Services\services_%date:~12,2%_%date:~4,2%_%date:~7,2%_%time:~0,2%
_%time:~3,2%_%time:~6,2%.services"


ping localhost -n 61 > nul


Goto loop
```

Host Data Collection:

15 minute intervals:

Win32dd memory capture;

1 minute intervals: (as text files)

C:\WINDOWS\System32\drivers\etc\services; SysInternals: Listdlls, LogonSessions, Handle, Pslist, NetStat; Once per logon: PsInfo

Network Collection:

Always on:

Wireshark to collect PCAP, Snort IDS with standard rule set to log alerts

Data Preprocessing:

Network feature extraction- Fullstats by Andrew Moore of Cambridge University, configured to Ubuntu OS supplied on hard drive.

Host text files feature extraction - Weka version 3.6 http://www.cs.waikato.ac.nz/ml/weka/

Determine best Machine learning algorithm - Weka Class Algs [31] from http://wekaclassalgos.sourceforge.net/

MS Excel to merge CSV features files.

Data Features Classification:

Weka version 3.6 http://www.cs.waikato.ac.nz/ml/weka/

Bibliography

1. AntiSource, "Malware Threats Triangle," 2 Feb 2011. http://www.antisource.com/staticpages/index.php/malware-triangle
2. Bai, Yuebin, and Hidetsune Kobayashi. "Intrusion Detection Systems: Technology and Development," 17th International Conference on Advanced Information Networking and Applications, 710 (2003).
3. Biles, Simon. "Detecting the unknown with snort and statistical packet anomaly detection engine (SPADE)," Computer Security Online Ltd. 5 January 2011. http://www.computersecurityonline.com/spade/SPADE.pdf
4. Bullinaria, J.A. "Learning Vector Quantization (LVQ): Introduction to Neural Computation: Guest Lecture 2," 07 Sept 2010. http://www.cs.bham.ac.uk/~pxt/NC/lvq_jb.pdf
5. Clay, Wilson. "Botnets, Cybercrime, and Cyberterrorism: Vulnerabilities and Policy Issues for Congress," Washington, D.C.: Congressional Research Service, 25, (January 29, 2008).
6. CSO Magazine. "OVER CONFIDENCE IS PERVASIVE AMONGST SECURITY PROFESSIONALS," 2007 E-Crime Watch Survey, 26 Feb 2011. www.cert.org/archive/pdf/ecrimesummary07.pdf
7. Davis, Joshua, "Hackers Take Down the Most Wired Country in Europe" WIRED MAGAZINE: ISSUE 15.09, 10 Feb. 2011 http://www.wired.com/politics/security/magazine/15-09/ff_estonia?currentPage=all
8. Depren, O., M. Topallar, E. Anarim, and M.K. Ciliz. "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks," Expert Systems with Applications, 29(4): 713-722, (2005).
9. Duffield, N., P. Haffner, B. Krishnamurthy, and H. Ringberg. "Rule based anomaly detection on IP flows," Proc. IEEE INFOCOM, Rio de Janeiro, Brazil, (April 2009).
10. Epic.Org "Google Violated New Zealand Privacy Law," 10 Feb 2011. http://epic.org/2010/12/google-violated-new-zealand-pr.html
11. Erskine, J.R. "Developing Cyberspace Data Understanding Using CRISP-DM for Host-based IDS Feature Mining," Master's Thesis, Air Force Institute of Technology, WPAFB, (2010).
12. Florez, G. "Analyzing system call sequences with Adaboost," Proceedings of the 2002 International Conference on Artificial Intelligence and Applications (AIA), Malaga, Spain. (2002).
13. Germano, T. "Self-Organizing Maps," 07 Sept 2010. http://davis.wpi.edu/~matt/courses/soms/

14. Gonzalez, F. and Dasgupta, D. "Neuro-immune and self-organizing map approaches to anomaly detection: A comparison," Proceedings of the 1st International Conference on Artificial Immune Systems, 203-211, (2002).
15. Gonzalez, J.A. "Numerical Analysis for Relevant Features in Intrusion Detection (NARFid)", Master's Thesis, Air Force Institute of Technology, WPAFB, (2009).
16. Gregg, M., Certified Ethical Hacker, Indianapolis, IN: Que Certification, (2006).
17. Grimes, Roger, "Stuxnet Marks the Start of the Next Security Arms Race" InfoWorld, 10 Feb 2011. http://www.pcworld.com/article/217725/stuxnet_marks_the_start_of_the_next_security_arms_race.html
18. Gu J. "An Introduction of Support Vector Machine" 16 Oct 2008, 5 Nov 2010. http://www1.cs.columbia.edu/~belhumeur/courses/biometrics/2009/svm.ppt
19. Gu, Guofei, Phillip Porras, Vinod Yegneswaran, Martin Fong, and Wenke Lee. "BotHunter: Detecting malware infection through ids-driven dialog correlation," Proceedings of the 16th USENIX Security Symposium, (2007).
20. Guan, Yu, Ali A. Ghorbani, and Nabil Belacel. "Y-means: a clustering method for intrusion detection," Canadian Conference on Electrical and Computer Engineering, 1-4, Montreal, Quebec, Canada, (May 2003).
21. Haag, Charles R., Gary B. Lamont, Paul D. Williams, and Gilbert L. Peterson. "An artificial immune system-inspired multiobjective evolutionary algorithm with application to the detection of distributed computer network intrusions", GECCO '07: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation. (2007).
22. Hendry, Gilbert, and Shanchieh Yang. "Intrusion Signature Creation via Clustering Anomalies," Proceedings of SPIE, (69730): 1-12 (2008).
23. Hunt, R.J. "Percent agreement, Pearson's correlation, and kappa as measures of inter-examiner reliability." J. Dent. Res. 65: 128-130, (1986).
24. Internet Security Systems White Paper, Network- vs. Host-based Intrusion Detection, A Guide to Intrusion Detection Technology. 5 January 2011. http://www.documents.iss.net/whitepapers/nvh_ids.pdf
25. Julisch, Klause, and Marc Dacier, "Mining Intrusion Detection Alarms for Actionable Knowledge," 8th ACM International Conference on Knowledge Discovery and Data Mining, 366-375 (2002).
26. Kanlayasiri, Urupoj, Surasak Sanguanpong, and Wipa Jaratmanachot. "A Rule-based Approach for Port Scanning Detection," Proceedings of the 23rd Electrical Engineering Conference, Chiang Mai, Thailand, (2000).
27. Kim, H. "Internet Traffic Classification Demystified: Myths, Caveats, and Best Practices." KNOM Tutorial POSTECH (2007), 2 Feb 2011. http://www.caida.org/publications/papers/2008/classification_demystified/

28. Kolbitsch, Clemens, Paolo Milani Comparetti, Christopher Kruegel, Engin Kirda, Xiaoyong Zhou, and Xiaofeng Wang. "Effective and efficient malware detection at the end host," USENIX Security Symposium, (August 2009).
29. Lakhina, Anukool, Mark Crovella, and
30. Larochelle, H. and Y. Bengio. "Classification using discriminative restricted Boltzmann machines", Proceedings of the 25th international conference on Machine learning, 307:536-543, (2008).
31. Lee, J.B. "WEKA Classification Algorithms," 07 Sept 2010. http://wekaclassalgos.sourceforge.net/
32. Lee, Wenke, and Dong Xiang. "Information-Theoretic Measures for Anomaly Detection," IEEE Symposium on Security and Privacy, Oakland, CA, (May 2001).
33. Mahoney, Matthew V, and Phillip K. Chan. "Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks," Proc. SIGKDD, 376-385, (2002).
34. Makanju, A.A.O., A.N. Zincir-Heywood, and E.E. Milios. "Clustering event logs using iterative partitioning," Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, 1255-1264. ACM New York, NY, USA, (2009).
35. Markoff, John, "Before the Gunfire, Cyberattacks" NYTimes, 13 Aug 2008, 10 Feb 2011. http://www.nytimes.com/2008/08/13/technology/13cyber.html
36. Maxion, R.A., and K.M.C. Tan. "Benchmarking Anomaly-Based Detection Systems," 1st International Conference on Dependable Systems & Networks, 25(28): 623-630 (June 2000).
37. McClure, S., J. Scambray, and G. Kurtz. Hacking Exposed 6: Network Security Secrets & Solutions, McGraw-Hill Osborne Media, (2009).
38. McHugh, J. "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA IDS evaluations as performed by Lincoln Laboratory," ACM Transactions on Information and System Security, 3(4) (November 2000).
39. MedCalc. "ROC curve analysis in MedCalc," 20 Feb 2011. http://www.medcalc.org/manual/roc-curves.php
40. Meyer, T.A., and B. Whateley, "Spambayes: Effective open-source, bayesian based, email classification system," Proceedings of the First Conference on Email and Anti-Spam (CEAS), (2004), 1 May 2010. http://www.ceas.cc/papers-2004/136.pdf
41. Moore, Andrew W. and Denis Zuev, "Internet Traffic Classification Using Bayesian Analysis Techniques," Proceedings of the ACM SIGMETRICS, Banff, Canada, (June 2005) 3 Mar 2011. http://www.cl.cam.ac.uk/research/srg/netos/brasil/data/index.html

42. Moore, Andrew W. and Konstantina Papagiannaki. "Toward the Accurate Identification of Network Applications," Proceedings of the Sixth Passive and Active Network Measurement Workshop, (PAM 2005), Lecture Notes in Computer Science, 3431: 41-54. Springer, (2005).
43. Moore, Andrew W., Denis Zuev, and Michael Crogan. "Discriminators for Use in Flow-based Classification" Technical Report RR-05-13, Queen Mary, University of London, August 2005.
44. Newman, Daniel, Kristina M. Manalo, and Ed Tittel. "Intrusion Detection Overview," InformIT, (June 2004). 20 Feb 2011. http://www.informit.com/articles/article.aspx?p=174342
45. Ning, P., Y. Cui, and D. Reeves. "Constructing attack scenarios through correlation of intrusion alerts." Proceedings of Computer and Communications Security, (2002).
46. Okolica, James and Gilbert L. Peterson. "A Compiled Memory Analysis Tool," Advances in Digital Forensics VI, IFIP Advances in Information and Communication Technology, 337: 195-204, (2010).
47. Paxson, Vern. "BRO: A System for Detecting Network Intruders in Real Time," Computer Networks, 31 (23): 2435-2463 (1999).
48. Pieprzyk, Josef, Thomas Hardjono, and Jennifer Seberry. Fundamentals of Computer Security, New York: Springer Books, (2003).
49. Pietraszek, T. "Alert Classification to Reduce False Positives in Intrusion Detection," PhD Thesis. Germany: Albert-Ludwigs-Universitat Freiburg im Breisgau, (2006).
50. Plattner, B., and A. Wagner. "Entropy Based Worm and Anomaly Detection in Fast IP Networks" Proceedings of 14th IEEE WET ICE / STCA security workshop, 172-177 (2005).
51. Russinovich, Mark. "Microsoft SysInternals Suite" 2 Feb 2011, http://technet.microsoft.com/enus/sysinternals/bb842062.aspx
52. Sarasamma, S.T., Q. A. Zhu, J. Huff, "Hierarchical Kohonen Net for Anomaly Detection in Network Security," IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, 35(2): 302-312, (2005).
53. Shapiro, Joseph M., Gary B. Lamont, and Gilbert L. Peterson. "An Evolutionary Algorithm to Generate Hyper-Ellipsoid Detectors for Negative Selection," GECCO 2005: Genetic and Evolutionary Computation Conference, 337-344, Washington, DC, (June 2005).
54. Shyu, M.L., S.C. Chen, K. Sarinnapakorn, and L.W. Chang. "A novel anomaly detection scheme based on principal component classifier," Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, (2003).

55. Simache, C., M. Kaaniche, and A. Saidane. "Event log based dependability analysis of Windows NT and 2K systems," Proc. of the 2002 Pacific Rim International Symposium on Dependable Computing (PRDC02). (2002).
56. Suiche, Mathieu. Mathieu Suiche's Blog, 3 June 2010. http://www.msuiche.net/2009/10/11/windd-1-3-final-x86-and-x64/
57. System Solutions Group. "IDS Thoughts" 2 Feb 2011. http://www.ssg-inc.net/cyber_crime/ids_thoughts.html
58. Valdes, A., and K. Skinner. "Probabilistic alert correlation," Proceedings of Recent Advances in Intrusion Detection (RAID), 54-68, (2001).
59. Wang, K., G. Cretu, and S.J. Solto. "Anomalous Payload-based Worm Detection and Signature Generation," Symposium on Recent Advances in Intrusion Detection, 227-246 (2005).
60. Wei, Yan. "Network Attack Scenarios Extraction and Categorization by Mining IDS Alert Streams," Journal of Universal Computer Science, 11(8): 1367-1382 (2005).
61. Wu, H.C., R.W.P. Luk, K.F. Wong, and K.L. Kwok. "Interpreting tf-idf term weights as making relevance decisions." ACM Trans. Inf. Syst., 26(3):1-37, (2008).
62. Yung, K.H. "Detecting Long Connection Chains of Interactive Terminal Sessions," Proceedings of the 5th Annual Symposium on Recent Advances in Intrusion Detection, 1-16 (2002).
63. Zhang, Z., J. Li, C.N. Manikopoulos, J. Jorgenson, J. Ucles. "HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification", Proc. IEEE Workshop on Information Assurance and Security, 85-90, (2001).

Vita

1st Lt Jenny W. Ji graduated from John P. Stevens High School in Edison, NJ in 2003. She entered undergraduate studies at the University of California – Los Angeles, Los Angeles, California. In the summer of 2006; she was hired as an intern for the Information Services department of Amgen, a large biotechnology corporation. That same year, she finished her Bachelor's of Science degree in Electrical Engineering. She was commissioned into the United States Air Force in September of 2007 though Officer Training School at Maxwell AFB, Alabama.

Lt Ji was first assigned in October of 2007 to the Air Force Research Laboratory, Information Directorate in Rome, New York where she worked as a junior engineer in support of research to develop secure processing chips. In the beginning of 2009, she was admitted into the online MBA program offered by the University of Massachusetts–Amherst and expects to finish her curriculum there in 2012. Also in 2009, Lt Ji was selected to participate in the AFRL Commander's Challenge with the team based in Kirtland AFB, New Mexico to find solutions using lightweight, low power, covert sensor devices for Intelligence, Surveillance and Reconnaissance in low accessibility and mountainous terrain. In the autumn of that same year, she began her MS studies at the Air Force Institute of Technology, Wright-Patterson AFB, Ohio. Upon her AFIT graduation in March of 2011, she will return to AFRL, but will remain at Wright Patterson AFB in the lab's Sensors Directorate.

Permanent address: 2950 Hobson Way
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
24-03-2011	Master's Thesis	Sept 2009 - Mar 2011

4. TITLE AND SUBTITLE
Holistic Network Defense : Fusing Host and Network Features for Attack Classification

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Jenny Weiyue Ji, 1st Lt, USAF

5d. PROJECT NUMBER
10G328

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GE/ENG/11-18

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Joe Carozzoni
Air Force Research Labs (AFRL)
Information Grid Division/ Cyber Science Branch
26 Electronic Parkway
Rome, NY 13441-4514
DSN 587-7796; joe.carozzoni@rl.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RIGG

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approval for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

This work presents a hybrid network-host monitoring strategy,which fuses data from both the network and the host to recognize malware infections. This work focuses on three categories: Normal, Scanning, and Infected. The network-host sensor fusion is accomplished by extracting features from network traffic using the Fullstats Network Feature generator and from the host using text mining, looking at the frequency of the 500 most common strings and analyzing them as word vectors. Testing on data collected at AFIT from the 2010 Cyber Defense eXercise and a controlled data collection of a Normal Windows Vista host and an Infected Vista host. The new approach reduces the number of alerts while remaining accurate compared with the commercial IDS SNORT. These results make it such that even the most typical users could understand alert classification messages.

15. SUBJECT TERMS

Artificial Intelligence; machine learning; intrusion detection, host network fusion, attack classification

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr Gilbert Peterson ENG
U	U	U	UU	122	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-3636, x4281; gilbert.peterson@afit.edu