2019

Reduction of False Positives in Intrusion Detection Based on Extreme Learning Machine with Situation Awareness

Donald A. Burgio

Nova Southeastern University, dburgio@yahoo.com

NSUWorks Citation

Donald A. Burgio. 2019. *Reduction of False Positives in Intrusion Detection Based on Extreme Learning Machine with Situation Awareness.* Doctoral dissertation. Nova Southeastern University. Retrieved from NSUWorks, College of Engineering and Computing. (1093) https://nsuworks.nova.edu/gscis_etd/1093.

Reduction of False Positives in Intrusion Detection Based on Extreme Learning Machine with Situation Awareness

by

Donald A. Burgio

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

College of Engineering and Computing
Nova Southeastern University

2019

We hereby certify that this dissertation, submitted by Donald Burgio conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

12/13/2019

James D. Cannady, Ph.D.
Chairperson of Dissertation Committee

Date

Paul S Cerk

12/13/2019

Paul S. Cerkez, Ph.D.
Dissertation Committee Member

Date

Wei Li

12/13/2019

Wei Li, Ph.D.
Dissertation Committee Member

Date

Approved:

Meline Kevorkian

12/13/2019

Meline Kevorkian, Ed.D.
Interim Dean, College of Computing and Engineering

Date

College of Computing and Engineering
Nova Southeastern University

2019

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

Reduction of False Positives in Intrusion Detection Based on Extreme Learning Machine with Situation Awareness

by
Donald A. Burgio
2019

Protecting computer networks from intrusions is more important than ever for our privacy, economy, and national security. Seemingly a month does not pass without news of a major data breach involving sensitive personal identity, financial, medical, trade secret, or national security data. Democratic processes can now be potentially compromised through breaches of electronic voting systems. As ever more devices, including medical machines, automobiles, and control systems for critical infrastructure are increasingly networked, human life is also more at risk from cyber-attacks. Research into Intrusion Detection Systems (IDSs) began several decades ago and IDSs are still a mainstay of computer and network protection and continue to evolve. However, detecting previously unseen, or zero-day, threats is still an elusive goal. Many commercial IDS deployments still use misuse detection based on known threat signatures. Systems utilizing anomaly detection have shown great promise to detect previously unseen threats in academic research. But their success has been limited in large part due to the excessive number of false positives that they produce.

This research demonstrates that false positives can be better minimized, while maintaining detection accuracy, by combining Extreme Learning Machine (ELM) and Hidden Markov Models (HMM) as classifiers within the context of a situation awareness framework. This research was performed using the University of New South Wales - Network Based 2015 (UNSW-NB15) data set which is more representative of contemporary cyber-attack and normal network traffic than older data sets typically used in IDS research. It is shown that this approach provides better results than either HMM or ELM alone and with a lower False Positive Rate (FPR) than other comparable approaches that also used the UNSW-NB15 data set.

Acknowledgements

First, my appreciation and thanks go out to my dissertation committee of Drs. Cannady, Cerkez, and Li for agreeing to support my research and for giving their time, talents, and feedback. And thanks especially to Dr. Cannady for his willingness to be my advisor, along with his encouragement, concise guidance, and patience. Dr. Cannady’s description of his research efforts as the “intersection between artificial intelligence and information security” had resonated with me while exploring program options prior to embarking on this journey as it still does today.

I’d also like to thank my family, friends, and colleagues, especially Mark, that provided their support, understanding, and encouragement along the way. Finally, this work would not have been possible without drawing upon the prior research of the many that are cited within this dissertation report.

Table of Contents

Chapters

List of Tables

Tables

List of Figures

Figures

List of Equations

Equations

Chapter 1

Introduction

Background

Intrusion Detection Systems (IDSs) are a collection of hardware and software resources that can detect, analyze, and report indications of intrusions in computer systems and networks. Extending from IDS research, there are Intrusion Prevention Systems and Intrusion Response Systems focused on the prevention and response aspects of intrusions respectively (Inayat, Gani, Anuar, Khan, & Anwar, 2016). Some IDSs can be used in-line to both detect indications of and prevent intrusions in near real-time and are sometimes referred to as Intrusion Detection and Prevention Systems. In other cases, these systems can communicate with other security devices such as firewalls, which monitor and control network traffic into and out of a protected network, to automatically implement blocking rules in response to detection.

Research into IDSs began several decades ago with a key paper on computer threat monitoring and surveillance, based on mainframe audit logs, by Anderson (1980). The concept of intrusion detection analysis existed prior to Anderson’s report but typically just consisted of system administrators manually scanning audit logs for anomalies. Yost (2015) reported that Clyde began work in 1977 on a limited scope commercial IDS, named Control, but that effort was not considered as influential or comprehensive as Anderson’s analysis.

Initially, intrusion detection was focused on after-the-fact batch analysis of audit records until Denning and Neumann (1985) proposed requirements and a model for the real-time Intrusion Detection Expert System with the goal of being able to detect most intrusions while making it extremely difficult to avoid detection. In a follow-on seminal paper, Denning (1987) went on to further the research into this field.

IDSs are characterized as either focused on misuse detection, based on known attack patterns or signatures, or anomaly detection, based on deviations in behavior from normal. Misuse detection has been preferred in commercial environments due to a higher level of accuracy since it is grounded in known attacks. Academic research has favored anomaly detection based on its higher potential to recognize novel attacks (Tavallaee, Stakhanova, & Ghorbani, 2010; Mitchell & Chen, 2014).

Substantial research has been done in anomaly detection across many domains including intrusion detection. Azad and Jha (2013), in a survey covering 75 research papers, list over a dozen different data mining techniques that have been applied to intrusion detection.

Other recent research using a variation of neural networks known as Extreme Learning Machine (ELM), first introduced by Huang, Zhu, and Siew (2004), has been applied to intrusion detection with promising results for reducing false positives while providing good generalized performance with extremely fast learning speeds (Creech & Jiankun, 2014; Fossaceca, Mazzuchi, & Sarkani, 2015). Baum and Petrie (1966) originated Hidden Markov Models (HMMs), a type of Markov chain, which model sequences of potential events. HMMs have been applied to several domains including speech and handwriting recognition and more recently to modeling cyber-attacks.

Endsley (1988) first introduced the concept of situation awareness in the context of human factors research. She defined situation awareness as consisting of perception of elements in time and space, comprehension of their meaning, and their projection into the future. Hutchins, Cloppert, and Amin (2011) subsequently took a military-inspired approach to cyber situation awareness, using a "kill chain," citing the need to better detect indications of multistage attacks including Advanced Persistent Threats (APTs).

Problem Statement

Existing anomaly detection techniques for IDSs have a high False Positive Rate (FPR) (Zuech, Khoshgoftaar, & Wald, 2015; Fernandes, Rodrigues, Carvalho, Al-Muhtadi, & Proença, 2019). Anomaly based methods, which show promise of detecting indications of novel cyber-attacks, still typically generate more false positives than signature-based methods which tend to have more false negatives (Pao, Lee, & Huang, 2015). While substantial research has been conducted on IDSs in general, detection uncertainty still exists (Inayat et al., 2016). Contributing to this uncertainty is the growth in network sizes and the increasing complexity and variability of attacks. This, in turn, has significantly complicated the ability of IDSs to produce accurate alerts (Spathoulas & Katsikas, 2013a).

Axelsson (2000) in his widely cited paper on the base rate fallacy called false alerts the biggest issue in IDS effectiveness. Based on Bayesian statistics, he argues that the false alarm rate should be measured in relation to how many intrusions one would expect to detect rather than the maximum number of possible false alarms. And that an

IDS, to be considered effective, would need to have a very high standard of 1/100,000 false alarms per event.

Horne (2015), stressing the continued relevance of Axelsson's research, illustrates the base rate fallacy by noting that due to Bayes' rule that if an IDS has a 99.9% accuracy rate on a network where 1 in 100,000 inputs, the base rate, comes from a malicious source, that means that for every 1 true positive alert, the system will generate 99 false alerts (its positive predictive value).

Perdisci, Ariu, Fogla, Giacinto, and Lee (2009) conclude that the FPRs for IDSs must be very low. Sommer and Paxson (2010), examining the imbalance between research into IDSs based on anomaly detection and their operational deployments, concur that reducing false positives in anomaly detection for IDS must be a top priority given that IDS error rates have a very high operational cost, as compared to other domains, which impedes the adoption of anomaly detection.

Goals

The primary goal of this research was to develop a new approach for an anomaly-based IDS that can better minimize false positives with the ability to detect indications of contemporary cyber-attacks while not sacrificing accuracy. This goal has been demonstrated using a recent and relevant set of comprehensive benchmark data, a secondary goal, containing both cyber-attack and normal traffic.

However, while virtually all agree that FPRs for IDSs must be low, there is not a widely used benchmark rate for FPR cited in recent IDS research. Additionally, some researchers use terms such as false alerts and false alarms interchangeably with false positives but others have different definitions which can create confusion and invalid comparisons.

While Axelsson (2000) put forth a high standard for false alarms and effectiveness, McHugh (2000), in his critique of the evaluation of IDS systems conducted by Lincoln Labs for the Defense Advanced Research Projects Agency (DARPA) in 1998 and 1999, cited that DARPA had a criterion of 0.1% for false alarm rate. However, many researchers since then tend to compare their results among different algorithms and with other studies using the same data sets versus setting an absolute target for FPR. Though, more recently, Swarnkar and Hubballi (2016) cited 0.6% as an acceptable FPR for their anomaly-based detection method for Hypertext Transport Protocol (HTTP) cyber-attacks using a data set of more than one million events.

The primary evaluation measurement for this research was to demonstrate that the proposed two-stage approach produced a reduction of false positives by at least 10% compared to just using the first stage. A secondary measurement was to demonstrate an overall FPR of 0.6% or less. Both evaluation goals were achieved. In addition to showing improved performance using two stages, the results were also compared to other algorithms using the same training and testing data sets and to published research.

Relevance and Significance

Despite substantial academic and commercial research and increased spending on computer and network security, major computer data breaches involving sensitive personal identity, financial, medical, trade secrets, or national security data continue to occur. According to Ponemon and Trunkey (2016), the average cost of a data breach had increased by 29% over the prior two years to $4 Million. Verizon (2016) reported that out of 2,260 confirmed breaches analyzed from 2015 that attackers take minutes or less to compromise a system but 80% of the victims did not realize they had been breached for weeks or longer.

Furthermore, Cyber-Physical Systems, which fuse network and computer components with physical components such as actuators and sensors, have become more commonplace. With the related emergence of the Internet-of-Things (IoT), large numbers of previously un-networked devices, including mundane consumer devices such as voice-activated speakers, thermostats, televisions, and even refrigerators, are being placed on the Internet and can pose a threat. An example is where a refrigerator was used to send spam email with hypertext links that could install malware if the links were clicked (Starr, 2014).

Another IoT related incident occurred in 2016 when the Mirai botnet slowed or stopped major portions of the Internet in the eastern United States. It accomplished that by merely taking advantage of poor security controls in low-cost IoT devices, such as surveillance cameras and wireless routers, that were connected to the Internet via high-speed broadband connections. Mirai, a self-replicating worm, doubled in size every 76 minutes and at its peak controlled around 600,000 devices around the world. It was used to launch a then record setting Distributed Denial of Service (DDoS) attack using the bandwidth harnessed (Graff, 2017).

Other trends such as cloud computing, where computing resources are being increasingly distributed, virtualized, and outsourced, and similar large-scale disruptive networking technology shifts including Network Function Virtualization and Software

Defined Networks, create new cyber-attack surfaces and challenges (Modi, et al., 2013; Alsmadi & Xu, 2015). Cyber-Physical Systems include controls for critical systems such as smart electrical grids hosting nuclear and other power generation plants, oil and gas pipelines, unmanned aircraft and drones, self-driving automobiles, chemical and other industrial plants, and healthcare devices. While many confirmed data breaches often result in many millions of dollars in economic losses, such as those related to credit card and financial account fraud, Cyber-Physical Systems raise the stakes with potential large-scale catastrophic consequences to life. Thus, it is increasingly important to secure them from intrusions, both known and novel (Mitchell & Chen, 2014).

Given the increased stakes, the need for faster and more automated responses to indications of potential cyber-attacks is critical. Many commercial IDSs and related systems are capable of blocking traffic in real-time, by source Internet Protocol (IP) address for example, based on intrusion alerts. Such an automated response is categorized as an active response vs. a passive one. However, an action based on a false positive with an automated response could deny resource access to legitimate users or tasks that could be unacceptable based on the circumstance. An example of such would be preventing access to an Electronic Medical Records system by an emergency room doctor. Or more apocalyptically, the scenario from the 1983 movie *War Games*, where a computer program attempts to automatically launch nuclear missiles, based on the mistaken conclusion that the country is under attack, after humans are taken out of the decision loop. Thus, many organizations prefer a passive response that requires personnel to investigate each alert. But such an approach could lead to delays resulting in a data breach or other undesired consequences (Marchetti, Pierazzi, Colajanni, & Guido, 2016).

Many researchers agree that the key to detecting novel threats is through anomaly-based detection (Tavallaee et al., 2010; Mitchell & Chen, 2014). IDSs are a mainstay of any defense-in-depth security strategy (Zuech et al., 2015) but many commercial security infrastructure deployments are still predominately based on known threat signatures (Bhatt, Manadhata, & Zomlot, 2014). Intrusions evolve continuously and signature-based detection alone will often fail when presented with indications of intrusions that are not part of a known signature base (Wu & Banzhaf, 2010).

However, despite a large body of research on anomaly-based IDSs and the great promise they have shown, operational deployments have been impeded since error rates have very high operational costs (Wang & Lee, 2001; Sommer & Paxson, 2010). And, validating false positives can also distract human operators from real attacks (Ho, Lai, Chen, Wang, & Tai, 2012).

To put the significance of false positive percentages in the context of a real-world operational network, S. Bhatt et al. (2014) estimated that Hewlett Packard's corporate network, which then spanned 166 countries with more than 300,000 employees, generated between 100 billion to 1 trillion security events daily of which approximately 3 billion were processed by the security infrastructure. The infrastructure included IDSs feeding into a Security Information and Event Management system along with audit logs from other network devices. Thus, even a small percentage of false positives can translate into a large number of events to review. IBM estimated that organizations spend $1.3 million a year dealing with false positives, wasting 21,000 hours on average, and proposed more research on advancing Artificial Intelligence as a solution (Barlow, 2017).

According to research from the Information System Security Certification Consortium (2018), a non-profit organization which specializes in training and certifications for cybersecurity professionals, the shortage of cybersecurity professionals is close to three million people globally. They also cite that nearly 60% of roughly 1,500 security professionals surveyed say their companies are at a moderate or extreme risk of cybersecurity attacks due to this shortage. Cisco (2016) had previously estimated that there would be a shortage of 2 million cybersecurity professionals globally by 2019, particularly amongst those monitoring and responding to alerts from IDSs and related systems in Security Operations Centers. Such staffing shortages further justify the need for more accurate and automated approaches to intrusion detection and response. While no credible research purports to completely eliminate false positives in IDS anomaly detection techniques, improvements to further reduce the rate and need for human intervention are well-justified.

Barriers and Issues

An anticipated issue prior to conducting this research was to identify a publicly available and suitable data set that reflects real-world conditions with ample benchmarks for comparison. Newer data sets exist for various purposes, such as cyber competitions, but given their recent nature there will generally be a lack of ample published research in the academic literature for comparison. However, more than a dozen research papers have been published between 2017 and 2019 for the chosen University of New South

Wales Network Based 2015 (UNSW-NB15) data set (Moustafa & Slay, 2015) which provided some basis for comparisons as this research evolved.

Assumptions, Limitations, and Delimitations

An assumption for this research is that the UNSW-NB15 data set is correctly labeled and reflective of both contemporary attack and normal traffic. Given the goal of using a contemporary data set, there will have been less academic scrutiny given the amount of time since the release of that data set as compared to the more traditional, and dated, ones typically used for intrusion detection research.

Another anticipated limitation of using a newer data set was that false positives may be much higher than would have been achieved using the same exact approach as on one of the more traditional intrusion research data sets. Moustafa and Slay (2016) illustrated this in repeating five experiments using Naïve Bayes (NB), Decision Tree (DT), Artificial Neural Network (ANN), Logistic Regression, and Expectation-Maximization clustering on both the Knowledge Discovery in Databases Cup 1999 (KDD99) (University of California, Irvine, 1999), a classic and widely used intrusion detection research data set, and the UNSW-NB15 data set. The False Alarm Rate (FAR) obtained on the UNSW-NB15 data set was higher than when using KDD99 in all five cases. In the ANN example, KDD99 yielded a FAR of 1.48% while that same experiment using UNSW-NB15 produced a 21.13% FAR.

Separately, Khammassi and Krichen (2017) explored feature selection algorithms and applied the same proposed method to both KDD99 and UNSW-NB15. For KDD99, they used a subset of 18 features with a 0.105% FAR while the same algorithm for

UNSW-NB15, with a subset of 20 features, provided a FAR of 6.390%. They concluded that the difference in FAR was due to UNSW-NB15 being more complex than KDD99.

This limitation was mitigated by setting goals as a marked improvement compared to other experiments using the same data set as opposed to using benchmarks from experiments performed with outdated data sets. As for delimitations, the scope of this research focused on network-based intrusion detection as opposed to host-based.

Definition of Terms

Definitions without citations are the local definitions of this research only.

Analysis: attack methods which breach Internet-based applications such as via ports (e.g., port scans), emails (e.g., spam), or web scripts (Moustafa, Turnbull, & Choo, 2018)

Attack Traffic: network packets containing an attempt to compromise the confidentiality, integrity, or availability of a computer network or host such as through the manipulation of network packets, protocols, or payloads

Backdoor: an attack technique to bypass normal authentication to secure unauthorized resource access (Moustafa, Turnbull, & Choo, 2018)

Classifier: a model designed to predict which target class, or category, a data element belongs, such as attack or normal traffic

Denial of Service (DoS): an attempt to make host or network resources, such as memory, processing, and bandwidth, unavailable to authorized users by overwhelming those resources (Moustafa, Turnbull, & Choo, 2018)

IDS: a collection of hardware and software resources that can detect, analyze, and report indications of intrusions in computer systems and networks (Inayat et al., 2016)

ELM: a type of classifier derived from neural network models which does not require its hidden layer to be tuned (Huang, Zhu, & Siew, 2004)

Exploit: a sequence of instructions that takes advantage of a vulnerability causing unintentional behavior on a host or network (Moustafa, Turnbull, & Choo, 2018)

Fuzzer: a program designed to discover weak points in an application, an operating system, or a network by feeding it with massive inputting of random data (Moustafa, Turnbull, & Choo, 2018)

Generic: a technique that works against a block-cipher to cause a collision without respect to the configuration of the block-cipher (Moustafa, Turnbull, & Choo, 2018)

HMM: a type of Markov Chain that can be used to model and classify series of events (Baum & Petrie, 1966)

Normal Traffic: computer network packets which are not malicious in intent

False Positive (FP): the incorrect classification of a data element as belonging to a given class, such as attack traffic for the purposes of this research, when it actually belongs to another class, such as normal traffic

Reconnaissance: attacks that are designed to gather information about a network or hosts to evade security controls; also called probing (Moustafa, Turnbull, & Choo, 2018)

Shellcode: a small piece of code, often written in machine language, that is used as a payload to exploit a software vulnerability (Moustafa, Turnbull, & Choo, 2018)

Situation awareness: perception of elements in time and space, comprehension of their meaning, and their projection into the future (Endsley, 1988)

Worm: a malware program that replicates itself in order to spread to other hosts via a network by exploiting a vulnerability (Moustafa, Turnbull, & Choo, 2018)

List of Acronyms

ANN:	Artificial Neural Network
APT:	Advanced Persistent Threat
C2:	Command and Control
CFS:	Correlation Feature Selection
CSV:	Comma Separated Values

DARPA:	Defense Advanced Research Projects Agency
DDoS:	Distributed Denial of Service
DoS:	Denial of Service
DNS:	Domain Name System
DT:	Decision Tree
DR:	Detection Rate
ELM:	Extreme Learning Machine
FAR:	False Alarm Rate
FN:	False Negative
FNR:	False-Negative Rate
FP:	False Positive
FPR:	False-Positive Rate
HIDS:	Host-based Intrusion Detection System
HMM:	Hidden Markov Model
HTTP:	Hypertext Transport Protocol
IDS:	Intrusion Detection System
IoT:	Internet of Things
IP:	Internet Protocol
KDD99:	Knowledge Discovery in Databases Cup 1999
KNN:	K-Nearest Neighbor
MATLAB:	Matrix Laboratory
ML:	Machine Learning
MLP:	Multi-Layer Perceptron

NB:	Naïve Bayes
NIDS:	Network-based Intrusion Detection System
OS:	Operating System
PCA:	Principal Component Analysis
SVM:	Support Vector Machine
TCP:	Transmission Control Protocol
TN:	True Negative
TP:	True Positive
TPR:	True-Positive Rate
UNSW-NB15:	University of New South Wales - Network Based 2015

Summary

This chapter provided an overview of the problem of high rates of false positives and how they are impeding the adoption of anomaly detection techniques that are needed to better detect indications of compromise for today's continuously evolving cyber-attacks. Several examples are given of the changing cyber landscape to illustrate that the relevance and significance of this research reaches far beyond merely preventing financial losses. Life and liberty are now more at risk than ever before. The primary goal of this research, which was achieved, was to create a new technique to minimize false positives using promising emerging research into ELM as a classifier, in conjunction with HMM, in a situation awareness framework. The anticipated primary challenge, which was overcome, was finding a data set for training and testing that is indicative of both modern normal and attack traffic patterns, along with a basis for comparison, versus using one of the more commonly used, and dated, IDS research data sets prevalent in the literature.

Chapter 2

Review of the Literature

Overview

To appreciate and comprehend the problem of reducing false positives in intrusion detection, it is necessary to cover several relevant areas of the literature starting with IDSs. The history of IDSs, going back to Anderson (1980), and earlier, was briefly covered in the introduction. This section will focus on their evolution post-Denning (1987) along with the different types of IDSs that have been researched and deployed. The review will then focus on how IDSs are evaluated including by measuring FPs and other metrics. Popular data sets for training and testing that have been used for research in the literature and the associated data set challenges will also be covered. This will be followed by a review of the literature of the various techniques that have been used in machine learning with a focus on the reduction of FPs along with the importance of feature selection. The review will then cover situation awareness. Different types of cyber-attacks, often as illustrated by the UNSW-NB15 data set chosen for training and testing, will also be discussed throughout the section.

Types of IDS

Many of the most widely deployed IDSs, such as Snort (Roesch,1999) and Bro (Paxson, 1999), are signature-based. These are often also referred to as misuse-based, rules-based, knowledge-based, ontology-based, or expert systems. With this type of IDS, events are compared to predefined rules or patterns that are generalized knowledge of attacks. Each event observed is matched against fields such as source and destination IP addresses, ports, transport headers, and payload. In the Snort example, each rule often has documentation about the potential for false positives and negatives and often the corrective action that should be taken. Snort users can contribute rules and there are over 20,000 that were developed between 1999 and 2014 (Bhuyan, Bhattacharyya, & Kalita, 2014). However, signature rule writing is highly dependent on the expertise of the writer. There is latency between the time a new vulnerability or type of cyber-attack is discovered and the time a new rule is written and implemented which renders signature-based systems vulnerable to novel attacks.

According to Inayat et al. (2016), at least 25 other IDSs of note, in addition to Snort and Bro, had been proposed between 1996 and 2015. Five were characterized as signature-based, four as anomaly-based, eight as both anomaly and signature-based (hybrid), and the rest as other.

In addition to classifying IDSs as based on their detection approach by either signatures or anomalies, IDSs are further classified as either Host-based IDS (HIDS) or Network-based IDS (NIDS), respectively, depending on the whether they analyze system logs generated by a host Operating System (OS) or network traffic data based on communications to and from hosts (Tavallaee et al., 2010; Zuech et al., 2015). Hosts are sometimes referred to as end-points. An end-point could be a personal computer, server, mobile phone, Internet connected automobile, printer, voice-activated speaker, or other networked device.

NIDS can be based on either packet inspection or flow detection (Luh, Marschalek, Kaiser, Janicke, & Schrittwieser, 2016). Packet detectors analyze communication packet payloads. Deep Packet Inspection requires significant computing resources so packet detectors often only analyze a subset of the packet such as HTTP headers for the case of web traffic. Encryption, in some cases, can thwart the analysis of certain payload data beyond the packet headers needed for routing or switching. Some flow-based detection systems analyze communications patterns using attributes such as source and destination IP addresses, port numbers, state flags, and the number of packets and amount of data transmitted. Many routers and switches can export flow data using common formats such as NetFlow.

NIDS generally free up resources by not having processing on each host, may be able to detect issues that could be obfuscated by log tampering on a compromised host, and are more OS independent. But they are considered more difficult to configure to have full coverage within a network. HIDS, on the other hand, benefit by distributing resources across hosts and use OS-specific rules that might improve detection performance (Mitchell & Chen, 2014). Often, HIDS and NIDS are both used on the same network being monitored to provide better defense-in-depth.

IDS Performance Measurements

IDS performance for a binary classifier is typically defined using measurements based on four variables, which make up a confusion matrix, comparing the classifier's predicted output with known labeled actual values (Fawcett, 2006; Mitchell & Chen, 2014). A True Positive (TP) occurs when the predicted output is positive, an indication of an attack for this research, and the actual instance is positive. A True Negative (TN) occurs when the predicted output is negative, normal traffic for this research, and the actual instance is negative. An FP occurs when the predicted output is positive, an attack for this research, but the actual instance is negative, or normal traffic. A False Negative (FN) occurs when the predicted output is negative, or normal for this research, but the actual instance is positive. Various measurements can be calculated from these four variables, including:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{1}$$

$$\text{False Positive Rate (FPR)} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2}$$

$$\text{False Negative Rate (FNR)} = \frac{\text{FN}}{\text{FN} + \text{TP}} \tag{3}$$

$$\text{True Positive Rate (TPR) or Detection Rate (DR)} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4}$$

$$\text{False Alarm Rate (FAR)} = \frac{\text{FPR} + \text{FNR}}{2} \tag{5}$$

For the purposes of this research, the Moustafa and Slay (2015) definition of False Alarm Rate (FAR) is used unless otherwise noted. Other researchers often use FAR as interchangeable with FPR. Thus, it is important to verify the formulas used in different research papers, that may have the same or similar name or acronym, before comparing results.

There is often a trade-off between desirable characteristics such as TPR with undesirable ones such as FPR. A Receiver Operating Characteristic graph (Fawcett, 2006; Bhuyan et al., 2014) can be used to depict how well a given algorithm is performing. The TPR is graphed on the Y-axis and the FPR on the X-axis. Different IDS algorithms can be compared on the same graph to show their respective lift from average detection based on a known test data set. Other related measures, including Area Under the Receiver Operating Characteristic Curve, have been proposed as a measure of classifier robustness. Another proposed measure is the geographic mean, G-Mean, which is the square root of the positive class accuracy times the negative class accuracy (Zong, Huang, & Chen, 2013).

Data Sets for Training and Testing

A significant and recurring issue for research in anomaly detection for IDSs is the availability of data sets for training and testing that reflect current real-world conditions and continuously evolving intrusion threats (Brown, Cowperthwaite, Hijazi, & Somayaji, 2009; Shiravi, Shiravi, Tavallaee, & Ghorbani, 2012). Some of the earliest IDS datasets commonly used for research are those produced for IDS evaluations for DARPA by Lincoln Labs in 1998 (Lippmann et al., 2000) and 1999 (Lippmann, Haines, Fried, Korba, & Das, 2000). Many more recent studies in IDS use the KDD99 data set (University of California, Irvine, 1999) which was derived from data originally created for the DARPA evaluations (Tavallaee et al., 2010). Weller-Fahy, Borghetti, and

Sodemann (2015), argue that the KDD99 intrusion data set remains the only comprehensive and fully labeled benchmark data set that is widely used by researchers. Hu, Gao, Wang, Wu, and Maybank (2014) also suggest that the KDD99 is still the most credible data set for benchmarking IDS results.

However, KDD99 has also been criticized by many, including Tavallaee, Bagheri, Lu, and Ghorbani (2009), as a poor evaluator of anomaly detection techniques. The earlier DARPA evaluations, upon which KDD99 was based, were also criticized by McHugh (2000) for several issues associated with their design and execution. A refined version of KDD99, Network Security Lab-KDD, or NSL-KDD, which removes some of the duplicate records from both the training and testing data sets, was created to address some of the original criticisms of KDD99. Other researchers point out that the cybersecurity landscape has changed significantly since the creation of the KDD99 data and that many don't consider experiments that use older data sets as relevant today (Sommer & Paxson, 2010).

To address this, Song et al. (2011) developed their own data set since they could not find what they considered a viable alternative to the KDD99 data set. They used data collected from a diverse network of honeypots over three years to create a new data set called Kyoto 2006+. It includes 93 million sessions with 50 million being normal and the rest being attack. And, it was based on 14 features derived from KDD99 plus 10 additional features such as anonymized IP source address.

Other research has used commercial data sets from large security service providers. For example, Sundaramurthy, Bhatt, and Eisenbarth (2012), whom were affiliated with Hewlett Packard, analyzed Intrusion Prevention System data from Hewlett

Packard's Tipping Point platform which had included over 35 billion alerts collected over a five-year period from devices located in over 1,000 customer networks worldwide. However, Zuech et al. (2015) point out that organizations can be reluctant or legally constrained from sharing such data. The results published from the above data sets were aggregate statistics and presumably the authors would not be able to share the underlying data sets from their employer even if anonymized. A study by Coull, Wright, Monrose, Collins, and Reiter (2007) discusses how similar network trace data can be de-anonymized which provides a further disincentive for sharing potentially sensitive data.

Dainotti, Pescape, and Claffy (2012), while researching related Internet traffic classification problems, cite a lack of sharable network packet traces which they believe is partially due to privacy concerns. Spathoulas and Katsikas (2013b) concur that such a lack of data sets to test intrusion detection methods are a problem. They also believe that the problem is further compounded when researchers create new data sets for studies but do not make them available to others.

Yet, despite the criticism of KDD99 in particular, Azad and Jha (2013) analyzed 75 research papers in intrusion detection from 2000 through 2012 and found that 42% used the KDD99 data set. Hubballi and Suryanarayanan (2014) cite data sets as an issue in truly understanding the impact of false positive reduction research. Milenkoski, Vieira, Kounev, Avritzer, and Payne (2015) also list errors as an issue in publicly available data traces along with their limited shelf-life due to new attacks. In addition to DARPA and KDD99, other data sets are available from the Cooperative Association for Internet Data Analysis, capture the flag and cyber defense competitions such as DefCon, the Internet Traffic Archive, the Lawrence Berkeley National Lab/International Computer

Science Institute, MAWILab, and the Internet Storm Center (Milenkoski et al., 2015; Zuech et al., 2015).

More recently, Moustafa and Slay (2015), also concluded that there was a lack of a suitable comprehensive data set for evaluation of NIDS research efforts since data sets such as KDD99 do not reflect modern network traffic. In response, they created a new data set called UNSW-NB15. It consists of a hybrid of modern real normal activities and synthesized attack behaviors. The attack traffic was created using the Ixia PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security.

They have made UNSW-NB15 freely available to researchers for academic purposes. It contains nine categories of attack traffic (fuzzers, analysis, backdoor, DoS, exploit, generic, reconnaissance, shellcode, & worm) and 49 features. The 49 features are further divided as: flow, basic, content, time, connection, additional generated, and labeled features. Two of the 49 features are labels: one for normal or attack traffic and a second one for the attack traffic type. A full list of these features in included in Appendix A.

Moustafa and Slay (2016) provided a separate justification of the complexity of the data set. Moustafa (2017) also gave details of how the derived features were constructed. A ground truth table giving additional information on each attack including the category and corresponding Common Vulnerability and Exposures data cross-reference identifier and description is also available.

Khammassi and Krichen (2017) state that the NIDS research community considers UNSW-NB15 a new benchmark data set to be used for evaluations of IDSs. They explored feature selection algorithms and applied the same proposed method to both KDD99 and UNSW-NB15. They concluded that UNSW-NB15 is more complex than KDD99 and that further research was needed to improve classification for this new benchmark data set.

IDS and Machine Learning Techniques

IDS research that is based on anomaly detection often uses Machine Learning (ML) techniques. Russell and Norvig (2003) define ML as part of the field of Artificial Intelligence and as the ability to adapt to new circumstances to detect and extrapolate patterns. Two of the main approaches to ML are supervised and unsupervised learning. In supervised learning, labeled training data is used to infer a function to predict future values. In unsupervised learning, the data instances are not labeled. Unsupervised techniques have more potential applicability, given a lack of sufficiently labeled data, but assume that normal instances are more prevalent in the test data than abnormal instances. If that is not true, then those algorithms tend to have higher false positive rates (Bhuyan et al., 2014). In addition to supervised and unsupervised learning, there is semi-supervised learning where labeled training instances are only available for the normal class. Two primary types of supervised and unsupervised learning are classification and clustering respectively. Numerous ML algorithms have been applied in IDS research including: ANN, Bayesian statistics such as NB, Gaussian Regression, Support Vector Machine (SVM), HMM, DTs, and K-Nearest Neighbor (KNN).

Each method has its strengths and weaknesses. For many of the IDS methods above, algorithms are needed at some stage to optimize – either minimize or maximize – an objective function (Rai & Tyagi, 2013).

Biologically Inspired Models

Biologically inspired models, in addition to ANN, include the Genetic Algorithm, Particle Swarm Optimization, Ant Colony Optimization, and Artificial Immune Systems (Tsai, Hsu, Lin, & Lin, 2009; Bhuyan et al., 2014). Benmessahel, Xie, Chellal, and Semong (2019) created an IDS based on Locust Swarm Optimization in conjunction with a Feed-forward Neural Network and achieved a lower FPR than the same approach using Particle Swarm Optimization & Genetic Algorithm. Igbe (2019) proposed an IDS using an ensemble of Artificial Immune System techniques including Negative Selection Algorithm and Dendritic Cell Algorithm that achieved an FPR of 1.34% with a 98.11% accuracy using a subset of UNSW-NB15 consisting of DoS & normal traffic.

Clustering

Clustering is an unsupervised method that seeks to categorize items, such as alerts, based on similarity (Weller-Fahy et al., 2015). An example of clustering is k-means clustering where the data (observations) are split into k categories, or clusters. Each cluster is identified by its center. The algorithm is highly dependent on the initial states and can converge quickly to a local versus global optimum, which can lead to erroneous results (Liu & Yu, 2005). To overcome the local optima problem, researchers have proposed several techniques such as biologically inspired methods including ANN, evolutionary algorithms, and swarm intelligence such as Ant Colony and Particle Swarm Optimization (Rai & Tyagi, 2013).

Similarity and Distance Measures

Weller-Fahy et al. (2015) cite the importance of similarity and distance measures in network intrusion anomaly detection. They specifically call out measures used during feature selection, classification, and clustering. They broadly define measures as power based, such as Euclidean, Manhattan, and Mahalanobis or distribution based, including Kullback-Leibler distance and entropy.

Entropy is a measure of randomness and has been applied to detecting activities such as port scanning, DoS, and worms. Bereziński, Jasiul, and Szpyrka (2015) used entropy as a method of detecting behaviors related to botnets such as Command and Control (C2). They found that Tsallis and Renyi entropy performed better than Shannon entropy as measured using FPR and accuracy.

Decision Trees

DTs are implemented using “if then else” rules. To classify a data sample, they start with a root node and end with a leaf node. The attributes used to create the tree are important. Sindhu, Geetha, and Kannan (2012) evaluated several types of DTs including Decision Stump, C4.5, NB Tree, Random Forest, Random Tree, and Representative Tree for use as a light-weight IDS. They identified that a suitable subset of features and removing redundant instances are important to avoid bias in the learning algorithm and are key factors for achieving better detection accuracy. Shah, Khiyal, and Awan (2015) categorize DTs as simple but very powerful.

Valero León (2017) used the test and training set referenced in Moustafa and Slay (2016) to create his models but used the full UNSW-NB15 data set for testing. He provided confusion matrices for two models. One model was based on a DT and the other one, for contrast, used Snort (Roesch, 1999), a signature-based system that was loaded with the latest rules that were available at the time of the experiment. The DT-based model outperformed Snort by a significant margin.

Boosting

Boosting is a technique that improves a weak classifier, or learner, to achieve a higher accuracy. Adaptive Boosting (AdaBoost) is the most commonly used form and was introduced by Freund and Schapire (1997). Another form is LogitBoost, introduced by Friedman, Hastie, and Tibshirani (2000) to better address the issue of noise and outliers than AdaBoost. AdaBoost uses an exponential loss function while LogitBoost uses a linear loss function hence making LogitBoost less sensitive to noise and outliers. Kamarudin, Maple, Watson, and Safa (2017) created a LogitBoost based IDS in conjunction with a Random Forest DT. As a motivation, they cited achieving low false alarm rates with high attack recognition for novel attacks as a major challenge. They were able to achieve a slightly better performance in terms of DR and accuracy using LogitBoost as compared to AdaBoost. They concluded that their algorithm provided a comparable detection accuracy rate with a low false alarm rate. They called a low false alarm rate, which they defined the same as FPR, as the most crucial property of IDSs.

Tama and Rhee (2019) created an IDS using Gradient Boosted Machine, also known as a Gradient Boosted Regression Tree. Gradient Boosted Machine was created to improve the performance of Classification and Regression Tree. Using UNSW-NB15 data, they demonstrated that their approach achieved a lower FPR and a higher accuracy rate than compared to another type of DT.

Mixture Models

Mixture models are a type of unsupervised learning technique used to identify subpopulations. Moustafa, Slay, and Creech (2017) created a method they called Geometric Area Analysis based on Trapezoidal Area Estimation computed from the parameters of a Beta Mixture Model. They used Principal Component Analysis (PCA) for reducing the high dimensionality of the data. Citing the inability of conventional IDSs to detect new intrusive events due to a high FPR, they demonstrated that their method provided a lower FPR than several other techniques using the NSL-KDD and UNSW-NB15 data sets.

Moustafa, Creech, and Slay (2017) proposed an Anomaly Detection System based on Finite Dirichlet Mixture Model with a PCA-based feature reduction. Their approach yielded a lower FPR than other comparable approaches, such as Multivariate Correlation Analysis and Triangle Area Nearest Neighbor, using the NSL-KDD and UNSW-NB15 data sets. The other approaches that they used for comparison were based on correlation and distance measures which usually cannot detect modern attacks that mimic normal traffic. However, they imply that their approach can detect such attacks due to the precise boundaries that can detect small differences between normal traffic and attack vectors.

Moustafa, Adi, Turnbull, and Hu (2018) put forth a new scheme based on a Beta Mixture Model and HMM which they call Beta Mixture HMM. Using UNSW-NB15 data, they demonstrated that their scheme outperforms Classification and Regression Tree, KNN, SVM, Random Forest, and Outlier Gaussian Mixture in terms of FPR, DR, and Accuracy.

Bayesian Approaches

Bayesian networks are a common classifier based on a directed acyclic graph with nodes representing attributes and arcs representing dependencies. A simplified form is the NB classifier where all of the attributes are assumed to be independent. Barbara, Couto, Jajodia, Popyack, and Wu (2001) used NB for their Audit Data Analysis and Mining platform for classifying cyber-attacks and non-attacks without prior knowledge of new cyber-attacks. Koc, Mazzuchi, and Sarkani (2012) cite several prior uses of Bayesian statistics for IDS and present a Hidden NB model that relaxes the NB conditionally independence assumption for better accuracy than traditional NB models.

Genetic Algorithm

A genetic algorithm is based on Darwin’s evolutionary principle of survival of the fittest and seeks to optimize a population of candidate solutions based on a fitness function. Genetic algorithms simulate natural reproduction using cross over and other techniques similar to gene selection, mutation, and recombination. Crosbie and Spafford (1995) were one of the first to propose using a genetic algorithm for intrusion detection. Hoque, Mukit, and Bikas (2012) and others subsequently used genetic algorithms for IDS research but the results in relation to false positives were not promising compared to other cited research. A drawback of the genetic algorithm is the amount of time and computing resources it takes to reach an optimal solution.

Neural Networks

The concept of an ANN, inspired by the function of the human brain, particularly the interconnection between neurons, has been around since McCulloch and Pitts (1943) devised a mathematical model for a neuron. ANNs typically consist of layers of nodes, which contain activation functions, connected by weighted directed links. ANNs use supervised learning. Labeled data are used to train the ANNs to learn classification patterns such as intrusion or no intrusion in the case of a binary classifier. Data are presented to an input layer that often links with one or more hidden layers that then link to an output layer that provides a result.

Cannady (1998) presented an ANN for misuse detection with network traffic and concluded that the advantages of that approach included the ability to learn.   He listed disadvantages as the difficulty of obtaining sufficiently large amounts of training data and the black box nature of neural networks. In addition to the ability to learn, Russell and Norvig (2003) cite another advantage of ANNs is being able to tolerate noisy inputs.

Wang, Hao, Ma, and Huang (2010) argued that ANNs can improve the performance of IDSs when compared with other methods but that enhancement is required. They proposed an approach based on ANN and fuzzy clustering to achieve a higher detection rate, lower false positive rate, and better stability.

There has been much research into ANNs around optimal network structures, optimization of weights, training methods such as Back Propagation, and methods to avoid overfitting and other challenges. There are several different network structures but Multi-Layer Perceptron (MLP) is the one that is most used in IDS research (Shah et al., 2015).

Self-Organizing Map

Self-Organizing Map, as described by Kohonen and Somervuo (1998), is a type of neural network that can map highly dimensional data into a two-dimensional array. Rhodes, Mahaffey, and Cannady (2000) cite that the first use of a Self-Organizing Map in misuse detection was described by Cannady in 1998 which involved the output of a Kohonen map as input to a feed-forward ANN to detect temporally dispersed attacks (over a period of time by one attacker) and possibly collaborative attacks (multiple attackers). Self-Organizing Map does not need to learn intrusive behavior but rather it learns it through characterizing normal activities. Shah et al. (2015), concur that Self-Organizing Map provides a simple and efficient method to self-categorize inputs for clustering that offers a higher speed compared with other learning methods.

Kernel Machines

SVM, sometime referred to more generally as kernel machines, research was begun by Boser, Guyon, and Vapnik (1992). SVM requires labeled training data and operates as a classifier by creating a hyper-plane to group data into normal or abnormal classes.

Perdisci et al. (2009), citing SVMs tendency to have low FPRs, created an IDS using a multiple SVM voting scheme to detect malicious payloads including polymorphic shell code. Li et al. (2012) cited good accuracy results when using SVM for IDS with clustering and Ant Colony Optimization.

Advantages of SVMs include low FPRs and the ability to work with data that are not linearly separable by other techniques. However, they tend to be complex to implement due to the required mapping of the feature space into a higher dimension which often requires optimization techniques that lead to long training times (Gilmore & Haydaman, 2016).

Bamakan, Wang, and Shi (2017) created an IDS using a variant of SVM called Ramp Loss K-Support Vector Classification-Regression which is a multi-class SVM approach that is well-adapted for skewed and imbalanced data sets, such as attack data sets used in IDSs. To combat the presence of noise in training data, they introduced a ramp loss function instead of the hinge loss function usually used in SVMs. The ramp function helps to depresses outliers. They compared their method to regular K-Support Vector Classification-Regression and demonstrated, using UNSW-NB15 data, that their algorithm substantially improved accuracy and reduced FPR.

ELM

ELM, introduced by Huang, Zhu, and Siew (2004) is an emerging technology for ML. ELM was originally developed from single-hidden-layer feed forward ANNs. ELM overcomes issues including slow learning speed and poor scalability faced with other techniques, such as ANN and SVM. The crux of ELM is that the hidden layer need not be tuned (Huang, Wang, & Lan, 2011). The input biases and weights that connect the inputs to the hidden layer are chosen randomly while the output weights are calculated using a Moore-Penrose inverse which results in faster training times.

Numerous variations of ELMs have been proposed. Ding, Xu, and Nie (2014) outlined the evolution of ELM and discussed eleven different variations that had been proposed to improve its performance. Huang, Zhou, Ding, and Zhang (2012) proposed the equality constrained-optimization-based ELM, which integrates with the learning rules of a variant of SVMs, called least squares SVM, to find a global solution for the weights of the output layer. They purport ELM tends to have much better generalized performance for multi-class classification problems at up to a thousand times faster learning speeds than traditional SVM.

For the basic version of ELM, the number of hidden nodes (neurons) needs to be determined by the user typically by trial-and-error. Ding et al. (2014) stated that determining the number of hidden neurons to use for different data sets is an open research problem. However, Huang, Zhu, and Siew (2006) demonstrated that ELM is very stable across a wide range of hidden nodes but performance can degrade with too few or too many neurons.

Huang, Chen, and Siew (2006) proposed a solution for this called Incremental ELM where hidden neurons are added incremental until a given criteria is met. Huang and Chen (2008) offered an enhanced version of this called Enhanced Incremental Extreme Learning Machine. And Wang, Xu, Lee, and Lee (2018) proposed an enhanced version of Equality Constrained-optimization-based Extreme Learning Machine, called Construction with Adaptive Increments, where the number of hidden neurons are determined in an adaptively incremental way and the output weights are derived without having to be recomputed.

Cheng, Tay, and Huang (2012) applied ELM to the intrusion detection domain and concluded that the basic ELM model outperforms SVM in training and testing speed but had slightly lower accuracy using KDD99 data.   But, a kernel-based ELM achieved higher accuracy than an SVM.

Castaño, Fernández-Navarro, and Hervás-Martínez (2013) introduced PCA-ELM. Their method eliminates the random initialization of the ELM weights and determines them based on a PCA of the training data. This method transforms the data into a number of principal components which often better highlights anomalies. They trialed their method on various non-IDS related data sets and reported positive results.

Zong et al. (2013) introduced a weighted ELM to compensate for complex data classes, such as in IDSs, where the majority class tends to be favored based on training data. In many IDS implementations, this class would typically be normal events. They assigned a weight to each sample to heighten the impact of the minority class while dampening the majority class. Other techniques were discussed such as over sampling or under sampling of the testing data but were dismissed. The authors demonstrated their approach using a variety of non-IDS data sets and were able to influence the balance of FPR and TPR as shown on a Receiver Operating Characteristic graph. Xia and Hoi (2013) used Multiple Kernel Boosting with the Multiple Classification Reduced Kernel ELM (Deng, Zheng, & Zhang, 2013).

Creech and Jiankun (2014) used an ELM approach for a HIDS in conjunction with a full semantic analysis of system calls.   That resulted in higher accuracy and reduced false positives compared to other techniques including SVM, HMM, MLP, and KNN. They highlighted the decision engine training speed with ELM but noted it has slightly higher processing needs.

According to Fossaceca et al. (2015), ELMs are straightforward to implement, computationally efficient, and have excellent learning performance characteristics. They implemented an ELM based IDS solution, dubbed MARK-ELM, that they purport achieves superior detection rates and much lower false alarm rates than other approaches to intrusion detection. The authors compare ELM to SVM and note that ELM significantly outperforms SVM in computational speed while being on par with SVM for accuracy.

Wang et al. (2018) also applied their ELM-based Construction with Adaptive Increments solution to different IDS data sets and compared the results to those of MLP and SVM solutions. They concluded that ELM had faster training times but had mixed results for other performance measures such as FPR.

Deep Learning Approaches

Another more recent area of research in ML is on deep networks, which are multi-layer ANNs. Types of deep networks based on Restricted Boltzmann Machine include Deep Belief Network and Deep Restricted Boltzmann Machine. Types based on Auto-Encoders include Stacked Auto-Encoders and Stacked De-noising Auto-Encoders. Kasun, Zhou, Huang, and Vong (2013) concluded deep networks outperform traditional multi-layer ANNs, SLFNs, and SVMs for big data sets but exhibit slow learning speeds. Yu, Zhuang, He, and Shi (2015) citing the success of deep learning models and ELM, created a deep learning ELM named DrELM. They compared both ELM and deep learning methods including linear ELM, kernel-based ELM using both Gaussian and Sigmoid kernels, Optimally-Pruned ELM, Deep Restricted Boltzmann Machine, and Stacked Auto-Encoder. They concluded that their DrELM outperforms linear and kernel-based ELM and is comparable to other deep learning methods using non-IDS related data sets.

Tchakoucht and Ezziyyani (2018), stated that Recurrent Neural Networks, a type of ANN, are one of the most widely used deep learning techniques due to their predictive ability with sequential (temporal) data. They developed an IDS using a Multilayered Echo-State Machine based on Reservoir Computing which in turn is based on both ML and computational neuroscience. Reservoir Computing was introduced as a solution for Recurrent Neural Network training bottlenecks, from accurate but slow training methods such as Long Short-Term Memory, that can involve exploding gradient, vanishing gradient, and slow convergence. Recurrent Neural Networks as a reservoir have been described as Echo-State Networks. Using an Echo-State Network involves randomly generating an individual Recurrent Neural Network layer (reservoir) and then only training the reservoir to output connections. Tchakoucht and Ezziyyani (2018) used multiple layers of reservoirs based on promising results for other applications. They were able to show improved performance over select other IDS techniques and suggested additional research to further improve DR and FPR. They provided comparisons to prior results using three data sets including UNSW-NB15.

Blanco, Cilla, Malagón, Penas, and Moya (2018) proposed another deep learning technique, Convolutional Neural Networks, for a multi-class IDS in conjunction with a genetic algorithm. Convolutional Neural Networks are a type of MLP with a convolutional step to generate intermediate features to preserve spatial relationships between inputs before input into an MLP. Convolutional Neural Networks were inspired by the biology of animal visual cortexes and are typically used for image processing tasks. They arranged the features of UNSW-NB15 to create a 5x5 pixel image and optimized the classifier using a genetic algorithm to find a better layout for the input features. They obtained an accuracy of 98.14% but did not provide an FPR.

Muna, Moustafa, and Sitnikova (2018) proposed an IDS approach using Deep Auto Encoder with a Deep Feed Forward Neural Network architecture. They tested their approach against both the NSL-KDD and UNSW-NB15 data sets and concluded that their proposed approach provided a lower FPR and higher DR than other techniques used for evaluation.

Vinayakumar et al. (2019) modeled an IDS using a Deep Neural Network. They compared results to several other algorithms and with varying numbers of layers. While the results beat some classifiers in terms of FPR, their four-layer Deep Neural Network had an FPR of 26.4% compared to NB with 2.5% for normal traffic using the UNSW-NB15 data set.

Markov Models

A Markov model attempts to calculate the likelihood of a system in a given state based on a sequence of observations. An HMM (Baum & Petrie, 1966) is a Markov model with unobserved (hidden) states. Hu, Yu, Qiu, and Chen (2009) described HMMs as a double stochastic process with an upper layer Markov process whose states are not observable and a lower layer one where outputs are emitted and can be observed. An HMM requires a five-tuple for input and is written as $\lambda = (A, B, \pi)$ in compact notation where N is the number of states, M is the number of symbols per state, A is the state transition probability, B is the observation symbol emission probability distribution, and π is the initial state distribution (Rabiner, 1989; Gilmore & Haydaman, 2016).

Warrender, Forrest, and Pearlmutter (1999), calling HMMs one of the most powerful data modeling methods in existence, compared an HMM analyzing system calls as part of a HIDS, with three other methods. They found their HMM gave the best accuracy on average but at a high computational cost. Other researchers such as Hu et al. (2009) continued the focus on using system calls with HMM for a HIDS and proposed improved training methods which reduce training time but at the expense of slightly more false positives.

Ourston, Matzner, Stump, and Hopkins (2003) used an HMM to model multi-stage attacks using system calls. Their states consisted of probe, consolidate, exploit, and compromise. They used a C4.5 DT, an ANN, and an HMM on the same data and concluded the HMM had the most promise to detect multi-stage attacks. Bhatt, Yano, and Gustavsson (2014) proposed a framework using the kill chain and concluded that using an HMM for event correlation for detecting APTs looks promising.

Wright, Monrose, and Masson (2004) used HMMs for determining what network protocol, such as File Transfer Protocol was being used in a given packet, even if the payload was encrypted, based solely on packet size and inter-arrival time. They used network traces provided by George Mason University and DARPA and reported promising results.

Liang, Wang, Cai, and He (2008) proposed using an HMM for network security situation awareness based on using network services as states: Domain Name Service (DNS), World Wide Web, File Transfer Protocol, Network File System, and Mail. They used normal, attack, compromised, and hacked as observations. While they described the training and initial set-up of the proposed model, they did not provide any results to evaluate or any further information on their proposed data set for training and testing.

Sendi, Dagenais, Jabbarifar, and Couture (2012) used an HMM with other algorihms to model a specific type of DDoS attack relying on the Sadmind Remote Administration Tool using the DARPA 2000 data set for training and testing. They used four states for the HMM: normal, attempt, progress, and compromise. They were able to perfectly predict this specific attack and reduce false positives. However, while detecting a very specific attack sequence seems trivial they did highlight the ability of HMMs to detect multi-stage attacks that might be otherwise missed.

Zhou et al. (2015) used an HMM to develop a classifier to differentiate real attacks from normal traffic and non-attack related faults in an industrial control system environment. They used three states: normal, fault, and attack and focused on three types of attacks: spoofing, tampering, and DoS. They used a proprietary data set and were able to minimize false positives while providing fast detection. They highlighted the ability of HMMs to detect indications of attacks from both a spatial and temporal view and the importance of infusing domain specific knowledge into the model.

Chen, Guan, Huang, and Ou (2016) proposed an HMM using three states: reconnaissance, attack, and stepping-stone. Using IDS records, they reported success with their HMM to detect on-going multi-state attacks with a precision rate of 93.2%.

They believe HMMs can be used to further reduce the number of suspicious alerts generated by IDSs through classification; however, they pointed out the importance of considering different attack strategies in each stage in developing classifiers. They did not specify a data set that could be used by other researchers.

Liang, Chen, Yan, Zheng, and Zhuo (2017) also proposed using HMMs to model network security situation awareness with a scaled training method using entropy. They chose four states: good, probed, attacked, and compromised and used the DARPA 2000 traces as a data set. However, in addition to using an older data set, they did not provide any measures for false positives. Their primary focus was on the training aspect of the models.

HMMs have been used successfully as classifiers within intrusion detection. However, many researchers have used HMMs as part of a HIDS with a focus on using system calls as observations. While NIDS and related network classification research using HMMs has also been promising, many of the studies use older or unspecified data sets and often focus on a narrow range of specific attack types such as DDoS. But many researchers agree that HMMs provide a temporal aspect to classification problems in addition to spatial which is important in detecting indications of multi-stage attacks.

Hybrid Methods

There are combined learners that include ensemble, fusion-based, and hybrid. These approaches generally seek to combine several different methods so that the end result outperforms the individual results. An example of an IDS hybrid that combines both signature and anomaly-based methods is called EMERALD (Porras & Neumann, 1997). This type of approach has been shown to reduce false alarms and increase the capability of detecting unknown attacks. Mukkamala, Sung, and Abraham (2005), demonstrated that an ensemble of classifiers performs better than each algorithm individually but there is often difficulty in scaling such an approach to large data sets.

Depren, Topallar, Anarim, and Ciliz (2005) proposed a hybrid anomaly and misuse IDS using Self-Organizing Map for modeling normal behavior. The misuse portion used a J48 DT to classify various cyber-attacks. They concluded that their hybrid approach performed better than the individual ones.

Karthick, Hattiwale, and Ravindran (2012) used a hybrid approach with network traffic to identify malware. They used a NB classifier as a first stage followed by an HMM as a second stage to identify malicious source IP addresses. The NB passed traffic in real-time while the HMM was off-line. TCP state flags were used for the HMM states. The authors looked at several sets of attributes and concluded that for each server they needed both IP and port addresses to achieve the lowest FPR and highest detection accuracy. They further concluded that having more states available in the data improved the accuracy.

Akusok, Miche, Hegedus, Nian, and Lendasse (2014) used a two-stage approach to identify malware, using data from F-Secure, with the goal of minimizing both false positives and false negatives. Their first stage was based on KNN clustering but still yielded a high FPR. They then fed the KNN results into two separate ELMs. The data flagged as malware was fed into a FP minimized ELM and the traffic flagged as clean was fed into a FN minimizing ELM. The end result using 18,437 data points was a lower

FPR (2 FPs) with the two-stage approach compared to just using the KNN layer (183 FPs).

Moustafa, Turnbull, and Choo (2018), seeking a solution to better mitigate malicious events, particularly botnet attacks against protocols used in IoT networks such as DNS, HTTP, and Message Queue Telemetry Transport, created an ensemble using DT, NB, ANN, and AdaBoost. Using UNSW-NB15 data, they demonstrated that their ensemble method had a lower FPR and higher DR compared with each classification method separately.

Voting Schemes

Voting schemes, in which the output of several classifiers is combined to reach a conclusion, are popular in pattern recognition. Woods, Kegelmeyer, and Bowyer (1997) categorize techniques to combine multiple classifiers as either classifier fusion or dynamic classifier selection. For fusion, the individual classifiers are run in parallel and then a scheme to reach a group consensus is applied. For dynamic selection, the goal is to determine which classifier is likely to be correct. They cite several fusion algorithms including: majority voting, unanimous consensus, thresholding, heuristic polling, weighted ranking, and others. They proposed an improved approach to dynamic selection by using a local accuracy scheme.

Lin, Yacoub, Burns, and Simske (2003) demonstrated that a combination of classifiers can result in significant accuracy improvement and that voting methods are simple and effective. They also point out that for many applications that there is often only a marginal, if any, difference in performance between simple voting schemes and more advanced combination techniques.

Lin, Lai, Ho, and Tai (2013) believe that to overcome the limitations of a single IDS, multiple IDSs can be used to more accurately recognize threats. However, the results of multiple IDSs can often be in conflict. They cite majority voting as a technique to resolve such conflicts but note that it often leads to inaccurate decisions given that technique disregards the different domain knowledge of the IDSs in the minority. They proposed a creditability-based weighted voting scheme to overcome that limitation.

False Alarm Reduction Techniques

Hubballi and Suryanarayanan (2014) cite several reasons for false alarm generation in IDS including: intrusion activities are sometimes very similar to normal activities and thus difficult to differentiate, a lack of context data related to the alarm, and cases where circumstance determine whether an activity is malicious such as a network scan done by a security administrator versus a hacker. Many IDSs also alarm on attempts which do not necessarily lead to a compromise. Additionally, an alarm may represent a stage in a multi-stage attack that may fail in subsequent stages.

One way to throttle false positives is to adjust the alert threshold. But in existing systems there is usually a trade-off. Less false positives will also cause the system to miss attacks. Usually, administrators adjust the level to achieve a balance between the level of security and risk and the resources available to respond to alerts but that is a time consuming and knowledge intensive task (Wang & Lee, 2001).

Julisch and Dacier (2002) applied data mining techniques to reduce false positives. They discovered the cost of finding relevant episode rules outweighed the benefit but did discover interesting patterns such as that IDS alarms are very homogenous and repetitive. They concluded that a source host triggering a heterogeneous stream of alarms is likely an attacker. And that clustering using attribute-oriented induction produced results that were too general. The authors further suggested that custom made filtering rules, to automatically discard alarms, or correlation rules, to group alarms, can help reduce alert volume but that humans should be involved in the analysis to avoid destroying valuable data.

In a related paper, Julisch (2003) found that a few dozen root causes generally account for over 90% of alarms and suggested if these were removed, they would allow operators to concentrate on the remainder. He focused on nine cases including SYNchronize Flood, Suspicious GET, Host Scan, Fragmented IP, DNS Zone Transfer, TCP Hijacking, and Code Red. Some of these are still suspicious traffic today but this list would need to be updated for current threats since Code Red was an artifact of the time the paper was written.

One of the challenges with IDSs, according to Spathoulas and Katsikas (2010), is that intrusion methodologies and attack strategies evolve over time with technology. Large numbers of alerts and false positives are a common problem among almost all categories of IDSs. While many methods have been used to reduce false positives in IDSs, research has been evolving to develop effective technologies to classify activity at an acceptable DR. Spathoulas and Katsikas (2013a) outlined a solution to reduce false positives by up to 75%. The authors indicate that most false alert reduction research is focused on post-processing of alerts. A three-part post-processing filter based on their study of the distribution of false positives in alert sets from Snort was proposed. The first component, Neighboring Related Alerts, was based on the observation that attacks produce bursts of alerts at first as attackers scan for victims and vulnerabilities. The second component, High Alert Frequency, was based on their observation that attacks produce anomalies in the distribution of alerts. The third component, Usual False Positive, was based on false positives usually resulting from the same specific causes.

In a subsequent survey of false positive reduction, alert correlations, and visualization research, Spathoulas and Katsikas (2013b), put forth that while much research has been done, a complete solution is still missing and that there are many open issues and ideas to be explored with post-processing. They believe that any solution must be adaptable to future attacks, be able to work with other methods, and be efficient regardless of the IDS being used or the system being protected.

Hubballi and Suryanarayanan (2014) conducted a survey of research on false alarm minimization techniques for signature-based IDSs. They created a taxonomy of nine different techniques: signature enhancement, stateful signature-based, vulnerability signature-based, alarm (data) mining, alert correlation, alert verification, flow analysis, alert prioritization, and hybrid methods. For alarm mining, the techniques were further classified as clustering, classification, ANN, and frequent pattern mining. For alert correlations, their categories included: multi-step, knowledge-based, complimentary evidence, casual relation, fusion-based, attack graph, and rule-based. They reviewed commercial Security Information and Event Management systems and noticed that the majority of those systems use rule-based techniques for event correlation. Despite all of the research to date, Hubballi and Suryanarayanan (2014) concluded that more research is needed to address the usability of the proposed techniques in real-world scenarios.

Zuech et al. (2015) believe that a more comprehensive approach in monitoring and correlating security events from many different heterogeneous sources can give a more holistic view and greater situation awareness of cyber threats. Shittu, Healing, Ghanea-Hercock, Bloomfield, and Rajarajan (2015) used post-correlation prioritization based on anomaly detection and clustering to reduce false positives. They grouped related IDS alerts into meta-alerts. Those with higher outlier values indicating a larger anomaly were given a higher priority. They showed a false positive reduction of 97% in one scenario and 16% in another using industry data derived from a Snort system from 2012. The large difference between the two scenarios was attributed to lesser distinct outliers in the second scenario. The authors concluded that additional research is needed on real-time incremental outlier detection and clustering given their results were based on a batch analysis.

Situation Awareness

Endsley (1988) first introduced the concept of situation awareness in the context of human factors research in the aerospace field. Endsley defined situation awareness as having three components: the first was perception of elements in time and space, the second was the comprehension of their meaning, and the third their projection into the future.

Bass (1999) applied situation awareness to the field of intrusion detection, calling it cybersecurity situational awareness, and proposed that the fusion of multiple sensor data along with context-dependent threat and vulnerability information could form a model for the next generations of IDS. The field of network security situation awareness evolved from the work of Bass.

In applying situation awareness to intrusion detection, Bass (1999) borrowed the military concept of Observe, Orient, Decide, and Act and multisensory data fusion where a diverse array of data sources can be employed. He applied this to achieve a higher cyberspace situation awareness using data fusion for intrusion detection where data fusion can provide varying levels of data from just being aware of an intrusion up through analyzing the associated threats and vulnerabilities. His approach included analyzing data across multiple different device types and data sources concurrently. Wang, Liu, Lai, and Liang (2007) offered that network security situation awareness seeks to provide a solution to the high ratio of false positives and associated expensive response to IDS alerts.

Hutchins et al. (2011) took a military-inspired kill chain model approach to cyber situation awareness. They asserted that traditional computer defenses alone such as anti-virus and IDS are not effective against APTs where the threat actors are often well resourced, trained, and patient. They proposed an intrusion cyber kill chain that recognized different phases of an attack, which they defined as: reconnaissance, weaponization, delivery, exploitation, installation, C2, and actions on objectives. They determined that host-based IDSs were the best to detect indications of exploitation and installation while network-based IDSs were the best to detect weaponization and C2. Indications of reconnaissance may include host and port scanning activities that could be detected by IDSs. And indications of actions on objectives could include exfiltration that could likewise use IDSs for detection.

While substantial research has been done in intrusion detection for alert correlation, relatively little has been done for data fusion as compared to other domains such as military applications (Zuech et al., 2015). Other researchers such as Mees and Debatty (2015) are attempting to further define situation awareness frameworks for cyber defense. They argue situation awareness needs to be largely automated given the speed at which cyber-attacks are executed. They also call for more research on methods for working with data that may be incomplete, uncertain, or erroneous.

The reconnaissance phase of cyber-attacks includes activities such as finding target information through web crawling. Huang, Shen, Doshi, Thomas, and Duong (2015) observed that attackers often attempt to obtain information on a network as a first step in a cyber-attack. Attackers typically do this through a network scan to discover network topology and host information such as IP addresses. This is typically followed by port scans to determine host types and what services are running. This would then inform the weaponization phase using the kill chain model where an exploit vehicle such as a Portable Document Format file is chosen. This, in turn, leads to a delivery method such as via email or a web site download. After the exploit is delivered to the host, exploitation could be in the form of malware which exploits a vulnerability in processing files, for example, to install malicious code. Some examples of malware include viruses, Trojan horses, back doors, worms, root kits, scareware, and spyware (Luh et al., 2016). Once the attacker's code is active on the host, a C2 channel is often established to communicate back to the attacker. The attacker can then take actions such as exfiltration of data.

Marchetti et al. (2016) proposed a NIDS based approach to identify weak signals related to data exfiltration and other Advanced Persistent Threat (APT) activities. They purported that existing security solutions based on pattern matching work well for common attacks but can often not identify APTs. This is since APTs use unknown (zero-day) vulnerabilities and seek to hide within normal network traffic. APTs also typically use only a few internal hosts along with evasion detection techniques such as "low-and-slow." APTs may slowly exfiltrate data over long time periods to avoid detection and use encryption, which often thwarts signature-based IDSs. The authors list five main phases of an Advanced Persistent Threat including: reconnaissance, compromise, maintaining access, lateral movement, and data exfiltration which is similar to the Hutchins et al. (2011) kill chain model. The compromise phase is typically created through a spear phishing email with a zero-day exploit where a Remote Administration Tool is usually installed. The tool then initiates contact with a C2 server since connections initialized by an internal host are often allowed through a firewall and attract less attention. They used three features of network traffic to identify hosts potentially involved in data exfiltration: number of megabytes uploaded by internal hosts to external addresses, number of flows to external hosts, and the number of external IP addresses related to a connection initiated by the internal host. APTs are sometimes known as Advanced Targeted Attacks (Luh et al., 2016).

C2: Botnets

Botnets are defined as networks of machines compromised by malware. They are typically composed of hosts from both institutions and consumers that usually do not know they are infected. They are often used to propagate spam; perform DDoS attacks; distribute malware; facilitate software piracy, information harvesting, identify theft, Bitcoin mining, and extortion; and for manipulating online games, surveys, and web advertising click fraud. Since botnets are made up of many previously non-malicious hosts, those hosts are often not initially on lists of malicious IP addresses. A common defensive strategy is to check incoming IP addresses against a list of known bad actors. These lists are called black lists, watch lists, or Indicator of Compromise lists. They are often based on IP addresses, signatures such as a Uniform Resource Locator, a malware hash code based on Message Digest 5, or other similar algorithms. Botnets also allow attackers to confuse such defenses by launching attacks from a stream of changing IP addresses.

Some botnets have included in excess of one million hosts under control of "bot masters" who often rent out their network to malicious actors on a time-sharing basis for monetary gain. Botnets were initially developed for legitimate activities. The first ones used Internet Relay Chat as a C2 channel. The first bot of that type, Eggdrop, was developed in 1993. Bots were quickly adapted for malicious purposes. Examples of recent malicious botnet related software includes Zeus, often used for hijacking bank account credentials, which has more than 3,000 variants and is estimated to have infected 3.6 million hosts. Another botnet, Conficker used a Peer-to-Peer architecture to infect an estimated 10.5 million hosts in 2009.

Botnets have adapted rapidly and more recent versions, such as Torpig, have evolved to use techniques such as domain fast flux to take advantage of the DNS protocol to evade detection via methods such as blacklists. Countermeasure defenses for fast fluxing include monitoring DNS protocol activity. Once infected, bots or individual hosts on a botnet exchange messages via an established C2 channel. Such traffic is usually not high volume and thus avoids detection from many methods. Botnets often use multiple redundant C2 servers for resiliency in case one is taken out of service. Anomaly-based methods are considered the main research area for botnets. Detection includes network traffic anomalies in areas such as latency, traffic volumes, traffic on unusual ports, and unusual system behavior (Silva, Silva, Pinto, & Salles, 2013). Some researchers refer to the use of a C2 channel as an “ET Phone Home” protocol where the host communicates with a botnet controller in a nod to the 1980’s science fiction movie.

Data Preprocessing and Feature Selection

Data preprocessing is an important step in anomaly detection. Such processing includes data set creation, feature construction, feature reduction or selection, data transformation (such as converting attributes from nominal to binary and scaling/normalization), and labeling in the case of supervised learning. Examples of features for a NIDS could include packet length, destination and source IP and port addresses, time stamps, and TCP flags.

Feature selection is the process of choosing the best subset of features, sometimes referred to as attributes or variables, in a given data set. A data set with a large number of features is termed highly dimensional. Feature selection is often done to alleviate this "curse of dimensionality" by removing redundant or irrelevant features to reduce the computational complexity and cost of an algorithm (Davis & Clark, 2011). Another benefit of feature selection is to better avoid overfitting in supervised training models (Saeys, Inza, & Larrañaga, 2007). Feature selection affects the resulting classifier accuracy, including the FPR (Bahrololum, Salahi, & Khaleghi, 2009), so sometimes a trade-off between computational complexity and accuracy needs to be evaluated.

Feature selection algorithms can be broadly categorized as either filter or wrapper methods. Wrapper methods use the classification algorithm itself to evaluate features while filter methods are independent of the classifier used (Karegowda, Manjunath, & Jayaram, 2010; Hall, 2000). Some refer to a third category of algorithms as embedded where feature selection is built into the algorithm as in the case of DTs (Saeys et al., 2007).

Saeys et al. (2007) cite advantages of filter models as being fast, scalable, and independent of the classifier with the primary disadvantage being they ignore interactions with the classifier. Since the filter methods are independent of the classifier, these methods can be used to perform feature selection once for evaluation of multiple classifiers. They also further categorize filter methods as univariate, meaning that each feature is considered separately, or multivariate. A disadvantage of univariate methods is they ignore dependencies among features which can lead to worse classification performance. However, several multivariate filter techniques have been proposed to overcome this. Examples of univariate methods include Chi-square, Euclidean distance, Gain Ratio, and Information Gain. Multivariate methods include Correlation-based Feature Selection (CFS) and Markov Blanket Filter.

Some filter methods perform feature ranking rather than feature selection. They are often then combined with search methods such as forward selection, backward elimination, and best-first (Karegowda et al., 2010).

Advantages of wrapper methods include that they are simple and interact with the classifier. Disadvantages include classifier dependence, being prone to getting stuck in a local optimum, and the risk of overfitting. Examples of wrapper methods include genetic algorithms, simulated annealing, and beam search (Saeys et al., 2007).

Substantial research has been done on feature selection and numerous other algorithms and variations have been proposed. Alhaidari and Zohdy (2018) proposed a feature pruning model for HMMs and achieved an FPR improvement from 19.16% to 0.38% by reducing the number of features used from UNSW-NB15 to 16. Another example of a technique used on intrusion detection data sets is gradual feature removal (Li et al., 2012).

Summary

There is ample literature to support the use of machine learning techniques in anomaly detection for reducing false positives. However, much research is still being done using older data sets, such as KDD99, that are not indicative of modern normal and attack traffic. But, according to Hu et al. (2014), KDD99 is still the most credible data set to benchmark IDS results. Weller-Fahy et al. (2015) also agree that KDD99 remains the only comprehensive and fully labeled benchmark data set that is widely used by researchers.

Thus, the largest gap in the literature is an absence of widely used contemporary data sets, reflecting recent cyber-attack methods, for training and testing for IDS research. While Khammassi and Krichen (2017) state that the NIDS research community considers UNSW-NB15 a new benchmark data set to be used for evaluations of IDSs, it will likely take several years for researchers to embrace this and for it to be significantly demonstrated via the literature. Another common gap is that many researchers do not provide sufficient details to replicate their research.

Chapter 3

Methodology

Overview

The approach of this research uses the three stages, phases, or levels of situation awareness as defined by Endsley (1988) and as shown in Figure 1 below. The first level, perception, was accomplished with a network-based (NIDS) anomaly-detection engine for intrusion detection using an ELM as the classifier. The ELM classifier monitors the environment by analyzing the attributes of each event record in search of indications of cyber-attacks. The ELM classifier only considers events occurring at a discrete point in time as determined by the duration field in each event record.

The second level, comprehension, was accomplished using HMMs. Comprehension is about synthesis of information and understanding the bigger picture. Since the HMMs are trained to evaluate sequences of events over time as opposed to a single event in time, they provide a more comprehensive view of what is going on in the monitored network. This also results in a higher level of awareness in recognizing multi-stage events. Each HMM looks at a specific behavior of a single attribute over time.

The third level, projection, is the decision informed by the perception and comprehension stages, in space and time, on whether a given alert from the perception stage should be put forth as an intrusion alert to either a human operator or for automated actions. Space, for the purposes of this research, is the network space as represented by

IP addresses, ports, protocols, and services. This phase was implemented as the combined output of the ELM and HMM classifiers using a unanimous voting scheme.

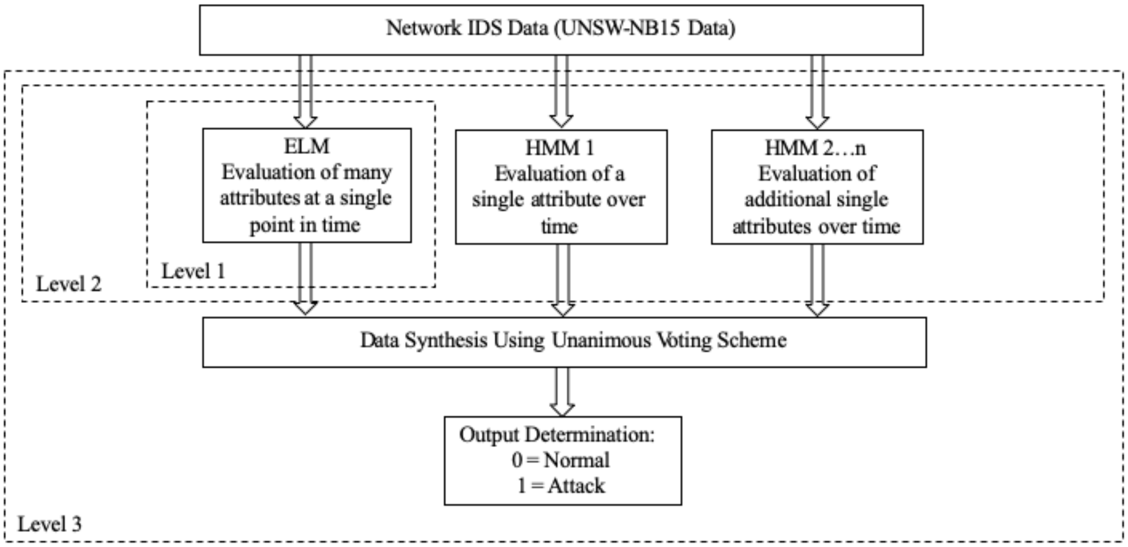

Network IDS Data (UNSW-NB15 Data)
ELM
Evaluation of many attributes at a single point in time
HMM 1
Evaluation of a single attribute over time
HMM 2…n
Evaluation of additional single attributes over time
Level 1
Level 2
Data Synthesis Using Unanimous Voting Scheme
Output Determination:
0 = Normal
1 = Attack
Level 3

Figure 1: IDS Model with Situation Awareness Level Boundaries

An ELM was chosen as the classifier for the perception phase based on the positive results as a classifier for IDS (Fossaceca et al., 2015) compared to other methods such as SVM. While ELM research only goes back to Huang, Zhu, and Siew (2004) with the first IDS-based ELM research presented by Cheng et al. (2012), ELM has shown its versatility to minimize false positives (Zong et al., 2013). Singh, Kumar, and Singla (2015) also implemented an ELM based solution against two different sets of benchmark data sets (KDD99 and Kyoto) and achieved superior false positive rates for both compared to other techniques including ANN, NB, and AdaBoost. Fossaceca et al. (2015), achieved the lowest false positive rates across categories comparing their ELM implementation to 20 other methods in published papers including NB, SVM, Random Forest, MLP, J48 DT, K-Means, and Self-Organizing Map.

HMMs were chosen for the comprehension phase given their ability to model temporal events and based on the promising results from other research. HMMs perform generally better than decision trees and substantially better than neural nets in detecting complex multi-stage Internet attacks (Ourston et al., 2003). And, HMMs proved to be very effective at detecting all types of attacks by acting as an anomaly detector over a set of IDS alarms providing a low rate of false positives and high rate of alarm reduction (Treinen & Thurimella, 2009) and for differentiating attacks (Zhou et al., 2015). Other research has also shown the advantages of using HMMs for modeling systems under attack (P. Bhatt et al., 2014; Gilmore & Haydaman, 2016).

A combined approach was used since prior research has shown positive results in reducing false positives using post processing (Spathoulas & Katsikas, 2013b). Hybrid results have yielded positive results such as using a KNN followed by ELM (Akusok et al., 2014) and a NB to HMM (Karthick et al., 2012) for malware detection.

The framework was further informed by existing models such as the kill chain (Hutchins et al., 2011) and mapped to data elements indicative of relevant cyber-attacks. The attack stages associated with the kill chain are: reconnaissance, delivery, exploit, install, C2, and exfiltration. But these stages were distilled into a simpler model of normal, probe, and attack and implemented in HMMs as three states. The HMMs provide additional context around interactions with a given network in time, for example, compared to a more traditional IDS approach which is often making a prediction based on a single event. This view provides better context for situation awareness since attacks can occur over longer periods of time than considered by some traditional IDSs. APTs in particular often use a low and slow approach to avoid detection.

The Data Set

This research used the UNSW-NB15 data set (Moustafa & Slay, 2015). This met the research goal of using a recent and relevant set of comprehensive benchmark data, containing both cyber-attack and normal traffic. This research used two subsets of data from UNSW-NB15. Their construction along with related UNSW-NB15 data sets is discussed below.

The Full 2.54M Data Set

The full UNSW-NB15 data set created by Moustafa and Slay (2015) includes approximately 2.5 Million records of which just over 14% are representative of cyber-attack traffic with the rest being normal. It contains nine categories of attack traffic (fuzzers, analysis, backdoor, DoS, exploit, generic, reconnaissance, shell code, and worm) and 49 features including two labels with one denoting attack or normal traffic and the other for the specific attack category. The features are further divided into flow, basic, content, time, additional generated, and connection features. A complete list of features is included in Appendix A. This data set is referred to as the Full 2.54M data set for the purposes of this research.

Moustafa and Slay (2016) validated that UNSW-NB15 is more complex than KDD99 and thus is a valid benchmark to be used for NIDS evaluations through three means: statistical analysis, evaluation of feature correlation, and comparison of FAR and accuracy for five classifiers as compared to KDD99. Khammassi and Krichen (2017) also stated that the NIDS research community now considers UNSW-NB15 a new benchmark for evaluations of IDSs.

The Test and Train Data Set

Moustafa and Slay (2016) accomplished this validation through decomposing UNSW-NB15 (University of New South Wales, 2015) into a smaller subset of training and test data sets. That subset consists of 257,673 records which were divided roughly into 60% for training and 40% for testing per the authors (but closer to 68% and 32% based on actual records). It includes 164,673 attack records and 93,000 normal records. Several features of the larger Full 2.54M data set were removed from this subset including source and destination IP addresses and port numbers along with start and end record times. To differentiate among UNSW-NB15 subsets, this data is referred to as the Train and Test dataset for the purposes of this research.

The 440K Data Set

Since this research uses HMMs for temporal situation awareness, it is important that both the training and testing data reflect an ordered sequence of events in time. Given the UNSW-NB15 provided Train and Test data subset removed needed time stamp features, a new set of training and testing data was created from the UNSW-NB15 Full 2.54M data set. A total of 440,044 sequential records, sorted by time stamp, were chosen from the data set and divided into roughly 60% for training (264,026 records) and 40% for testing (176,018 records). This data set is referred to as the 440K data set for the purposes of this research.

The 440K data was split using a simple out-of-sample hold out method. That is the first 60% of the records sequentially were used for training and the last 40% of the records were held out for testing. Cerqueira, Torgo, and Mozetic (2019), in an evaluation of performance estimation methods for time series forecasting tasks, state that there is no settled approach among researchers, but that out-of-sample methods are traditionally used for time-dependent data. Out-of-sample is in contrast to basic cross-validation which shuffles records which is not conducive to maintaining temporal order. A key characteristic of out-of-sample methods is that they always preserve the temporal order which in turn means the resulting model is never tested on past data.

Cerqueira et al. (2019) also reviewed other methods for time-dependent data such as cross-validation in a blocked form. They concluded that for real-world time series data, that approaches which maintain the temporal order are better. In particular, they recommended out-of-sample using repeated holdout.

To further validate this approach, a simplified repeated holdout method was used for the 440K data where several sections of the Full 2.54M dataset were evaluated in relation to maintaining a good mix of normal and attack records. This approach also better approximates a real-world IDS environment where data is ingested sequentially in time order.

The DoS Data Set

To provide for additional analysis and comparison to other research, another data set was created from the UNSW-NB15 Train and Test data set for just normal and DoS traffic. That data set will be referred to as the DoS data set for the purposes of this research. While not time-ordered, it was used for benchmarking the ELM portion, which is not time sensitive, and to evaluate the HMM classifiers and overall methodology in relation to how it performs with data that is not time-ordered. Table 1 below summarizes the UNSW-NB15 data sets relevant to this research.

Table 1: UNSW-NB15 Based Data Sets

Dataset Abbreviation	Description	# of Records	# of Normal Records	# of Attack Records	# of Training Records	# of Testing Records	Categories	Includes Time Stamps
Full 2.54M	Original Moustafa & Slay (2015) data set	2,540,044	2,218,761	321,283	N/A	N/A	Normal + 9 Attack	Yes
Train and Test	Moustafa & Slay (2016) created training and testing data set	258,673	93,000	165,673	176,341	82,332	Normal + 9 Attack	No
440K	Sequentially time ordered subset derived from the Full 2.54M records	440,044	351,150	88,894	264,026	176,018	Normal + 9 Attack	Yes
DoS	DoS and normal traffic extracted from the Train and Test subset	109,353	93,000	16,353	68,264	41,089	Normal + DoS	No

Feature Selection

As discussed in the literature review section, feature selection is an important step in the design of classifiers to remove redundant and irrelevant features. This reduces the computational complexity and the cost of an algorithm (Davis & Clark, 2011) and affects the resulting classifier accuracy, including the FPR (Bahrololum et al., 2009).

Given the selection of UNSW-NB15, two feature selection studies for those data were reviewed. The first (Moustafa & Slay, 2017) was from the data set creators citing that irrelevant features may cause a higher FAR. They used a hybrid feature selection technique based on Central Points and Associate Rule Mining. They were able to reduce the number of features to 11, using the Train and Test data set, which is approximately 25% of the given feature set. The Central Points technique selects the most frequent values reducing processing time and the Associate Rule Mining helps to remove irrelevant or noisy features. They pointed out that reducing the number of features to less than 11 resulted in undesirable results. The second study specific to UNSW-NB15, using the Train and Test data set, is from Janarthanan and Zargari (2017). They used

Information Gain (Kayacik, Zincir-Heywood, & Heywood, 2005) and CFS (Hall, 2000) algorithms to validate the Moustafa and Slay (2017) study and also provided comparisons to KDD99. Other intrusion detection research has also used Information Gain to select relevant features (Kayacik et al., 2005).

For this research, the appropriate features for both the ELM and HMM classifiers were evaluated and chosen using Information Gain and CFS respectively. Information Gain was chosen for the ELM since it is a filter method which is independent from the classifiers used. Given that one of the evaluation criterions is to compare the results against other classifiers, Information Gain allows the same attributes to be used for those other techniques (Saeys et al., 2007). Since Information Gain is a ranking method, a cut-off or threshold point needed to be determined on where to draw the line to select the number of attributes. For the first run, the cut-off point was determined using an informed estimate based on other feature selection studies using UNSW-NB15. An additional experiment was conducted using backwards feature removal to determine a cut-off point. Both results were compared.

For the HMM models, CFS was used. This was done to choose a subset of features in contrast to the ELM model which used Information Gain. One downside of Information Gain is it is a univariate method which ignores dependencies among features (Saeys et al., 2007). CFS, in contrast, is a multivariate method which was created on the basis that good feature subsets are highly correlated with the class yet uncorrelated to each other (Hall, 2000). For each feature selected by CFS, an HMM was created. This was done to create a series of classifiers which were then combined using a voting scheme. As discussed above and shown in Figure 1, the ELM classifier is focused on a point in time while the HMM classifiers are focused on temporal patterns. The selection of CFS provides for classifiers which are highly correlated to the class but uncorrelated to each other. The feature selection process is depicted in Figure 2 below.

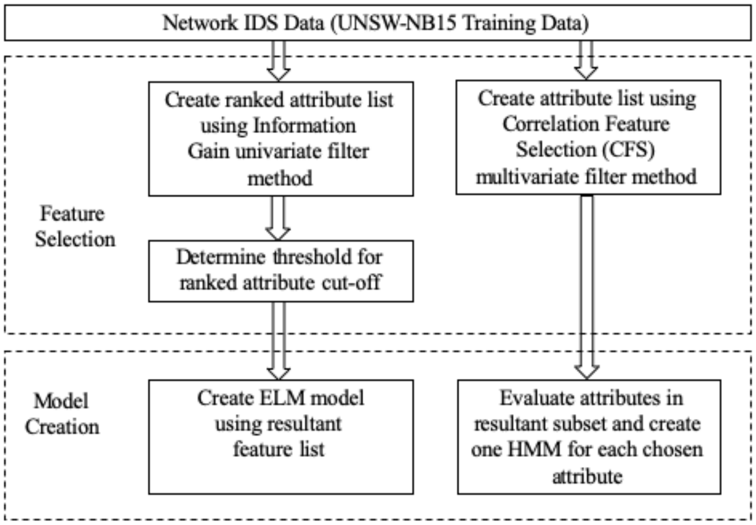

Network IDS Data (UNSW-NB15 Training Data)
Create ranked attribute list using Information Gain univariate filter method
Create attribute list using Correlation Feature Selection (CFS) multivariate filter method
Feature Selection
Determine threshold for ranked attribute cut-off
Model Creation
Create ELM model using resultant feature list
Evaluate attributes in resultant subset and create one HMM for each chosen attribute

Figure 2: Feature Selection Process

ELM Classifier

The ELM classifier was built to classify normal and attack traffic using the UNSW-NB15 data sets shown in Table 1. The basic ELM program (Huang & Zhu, 2004), MATLAB version, was obtained to implement the ELM classifier. MATLAB was chosen since that is what was used to perform the seminal ELM research (Huang, Zhu, & Siew, 2004) and many subsequent variations from various researchers have been made available in MATLAB.

The required inputs for the program are the file names for the training and testing data sets, ELM type (regression or classification), the number of hidden neurons, and the activation function. Supported activation functions include: sigmoidal, sine, hardlim, triangular basis, and radial basis. The training and testing file input to the ELM program is in the format of a tab-delimited text file. The first column must contain an integer label. For binary classification, 0 is used for normal (negative) record and 1 for an attack (positive) record.

The following preprocessing was done for each data set used for an ELM run. First, the binary label for attack and normal traffic was moved to be the first column and the label for attack category was removed. Second, depending on the run, the attributes not selected by the feature selection process were removed. Third, nominal or categorical attributes, such as protocol, state, and services for this data set, were converted to binary attributes as applicable. Fourth, all numeric attributes were normalized to be between -1 and 1 as recommended by Huang and Zhu (2004). Finally, the resulting test and training data sets were saved as tab-delimited text files.

In terms of other program parameters, the sigmoid activation function was chosen based on past experience. For ELM Type, the value was set to 1 for classification. The program also requires the number of hidden neurons as input. Ding et al. (2014) stated that determining the number of hidden neurons to use for different data sets is an open problem for ELM researchers. Others have proposed solutions to this (Huang & Chen, 2008; Wang et al., 2018). However, Huang, Zhu, and Siew (2006) demonstrated that ELM is very stable across a wide range of hidden nodes but performance can degrade with too few or too many nodes. For this research, different values were experimented with and charted.

HMM Architecture

An HMM requires the specification of five parameters including two model parameters: N (the number of states) and M (the number of distinct observation symbols per state) and three probability measures: A (the state transition probability distribution), B (the observation symbol emission probability distribution), and π (the initial state distribution). In compact notation form, an HMM is often written as $\lambda = (A, B, \pi)$ (Rabiner, 1989).

Various values of N (states) have been used for intrusion detection in the literature. Ourston et al. (2003) distilled a multi-stage attack sequence to four states: probe, consolidate, exploit, and compromise. Karthick et al. (2012) used TCP state flags as HMM symbols. Zhou et al. (2015) used three states: normal, fault, and attack. Hurley, Perdomo, and Perez-Pons (2016) used three states for a NIDS for Software Defined Networks: expected, unexpected, and somewhat expected. Liang et al. (2017) chose four states: good, probed, attacked, and compromised. Based on the nine labeled attack categories plus normal from the UNSW-NB15 data set, this research used: normal, probe (corresponding to analysis, fuzzers, and reconnaissance instances), and attack (corresponding to backdoor, DoS, exploit, generic, shell code, and worm instances). The number of distinct symbols per state, M, was based the results of the feature selection for the HMMs. The HMM state transition representation is shown in Figure 3 below.

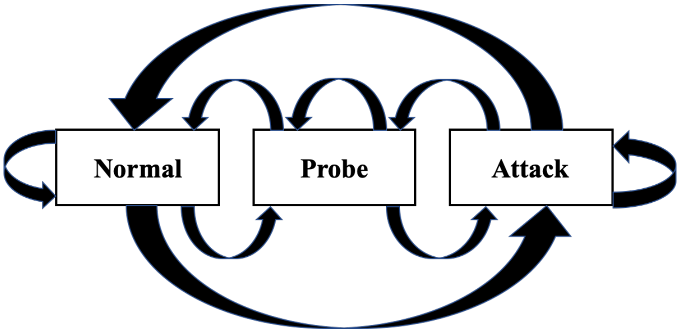


Figure 3: HMM State Transition Representation

For each HMM classifier, which represents one feature, symbols were selected, using the letters of the alphabet, to represent each possible feature value or range of values for the feature. Each symbol was mapped to its state, either normal, probe, or attack, for each record in the training data.

The probabilities were then determined through training using the Baum-Welch algorithm (Rabiner, 1989), which is the most commonly used HMM training algorithm (Holgado, Villagra, & Vazquez, 2017). For execution of the HMMs against the test data, the emission symbols for each model, as represented by one feature, were translated from their respective attributes. These symbol streams were then run through the Viterbi algorithm (Viterbi, 1967; Forney, 1973) to determine the most likely states.

The HMMs were implemented using the machine learning tool kit in MATLAB (Theodoridis, Pikrakis, Koutroumbas, & Cavouras, 2010). Training was performed using the hmmestimate and hmmtrain functions. Testing was performed with the hmmviterbi function.

The Combined Classifier

The third phase, projection, was the combined output of the HMMs and ELM providing both a projection in time and space. A unanimous voting scheme was chosen for the final output. Lin et al. (2003) demonstrated that a combination of classifiers can result in a significant accuracy improvement and that the use of voting methods to combine those classifiers are both simple and effective.

A unanimous voting scheme was chosen with the goal of eliminating false positives. Thus, all of the classifiers must agree on an attack. Other voting schemes such as majority voting can lead to an inaccurate decision given the knowledge of the minority classifiers are effectively ignored (Lin, Lai, Ho, & Tai, 2013). More complicated voting schemes were discounted given prior research that there is only a marginal, if any, difference in performance between simple voting schemes and more advanced combination techniques (Lin et al., 2003).

Experiments

Two groups of experiments were conducted and referenced as Experiments A and B. Experiment A used the 440K Data Set. The 440K Data Set was used to provide the time ordered data needed to demonstrate the temporal awareness capabilities of the HMMs.

Experiment B used the DoS data set. This was done for three reasons. The first was to use the Moustafa & Slay (2016) Test and Train dataset for which they justified the complexity of the data set and its validity for the evaluation of NIDS research. Although, many of their findings that are applicable to the Test and Train data set should also be applicable to the larger 440K Data Set as well. The second was to determine how the ELM and HMM scheme with situation awareness performed with data that were not time-ordered. And the third was to provide a more direct comparison to other research. In particular, Igbe (2019) had shown very good results using an Artificial Immune System scheme to detect DoS attacks using the UNSW-NB15 Test and Train dataset.

For the ELM models, multiple runs were done since the input biases and weights that connect the inputs to the hidden layer are chosen randomly for the basic ELM algorithm. Thus, each run will produce a different result but usually within a range. Wang and Huang (2005) used a similar process for their ELM research using an average of 50 simulation runs. For this research, ten runs were used. Additionally, each experiment also included comparisons to two other classifiers using the same data: an MLP and an J48 DT.

Feature selection for the ELM, J48 DT, and MLP runs was done using Information Gain which provides a ranked list of features in order of merit. The method used to select the features from the ranked list was backwards elimination with a cut-off point based on an estimate informed by other research which used the UNSW-NB15 data sets. An alternate to the informed estimate for a cut-off point was performed through an iterative search for contrast.

The feature selection method for the HMMs was CFS which provided a subset selection. For each feature selected in the CFS subset, a separate HMM was created. The number of hidden neurons for the ELM models was initially chosen based on prior experience. A second evaluation was done to determine an optimal value through iterative search.

Evaluation Criteria

The primary evaluation metric was the FPR given the primary goal of this research was to reduce false positives. This was measured based on the results of running three algorithms using the same data sets. The algorithms were: ELM alone, HMM alone, and the combination of ELM with HMM. The goal was to show that the combination of ELM with HMM produces a lower FPR than ELM alone by more than 10%. Accuracy was also calculated to ensure that was not significantly impacted.

Both experiments show the results of the ELM and HMM classifiers along with the combined output with the confusion matrix variables TP, FP, TN, and FN in a table. FPR and Accuracy were also calculated. Results of the J48 DT and MLP were also shown for the respective runs to provide comparisons.

As discussed in earlier sections, the decision to not use an older data set such as KDD99 limited the ability to compare this proposed research to specific FPRs from prior published studies. However, results from other research using various subsets of the UNSW-NB15 data set is also shown for contrast.

Computing Resources Used

This research was conducted using a MacBook Pro running MacOS with a 2.6 GHz Intel Core i7 Central Processing Unit, 16 Gigabytes of Random Access Memory, and a 750 Gigabyte hard drive. The hard drive provided sufficient swap space needed to support some of the large matrix calculations required of this research.

Summary

In this section, the research methodology using both ELM and HMMs in a situation awareness framework was detailed and justified. Feature selection using information gain and CFS was also chosen and justified. A data set, UNSW-NB15, to meet the goal of using one indicative of modern normal and attack traffic was also chosen and justified. The resources used along with evaluation criteria and metrics, including achieving a lower FPR with the combined classifier compared to the ELM or HMM classifiers individually were provided.

Chapter 4

Results

This chapter provides the results of the experiments designed to gauge the effectiveness of using ELM in conjunction with HMM with a situation awareness framework to reduce false positives. These experiments used subsets of the UNSW-NB15 data set (Moustafa & Slay, 2015). UNSW-NB15 was chosen to meet the stated research goal of using a recent and relevant set of comprehensive benchmark data, containing both cyber-attack and normal traffic.

The first experiment, Experiment A, used the 440K data set which is a time-ordered subset of the UNSW-NB15 data set. A time-ordered data set was needed to test the effectiveness of the HMMs per the situation awareness framework.

The second experiment, Experiment B, used a subset of the separate Train and Test data set, the DoS data set, that included both DoS attack and normal traffic. The Train and Test data set was validated as being statistically complex for evaluating existing and novel techniques for NIDS (Moustafa & Slay, 2016); however, it was not time-ordered.

Experiment A (Time-Ordered Data)

Experiment A was repeated twice. Both runs were conducted against the 440K subset of the UNSW-NB15 data set (Moustafa & Slay, 2015). Both used Information

Gain to select the features for the ELM and CFS to select the features for the HMMs. The difference between the two runs were in the number of features and the number of hidden neurons used by the ELM model. For the first run, the number of features selected for the ELM was determined by a cut-off threshold of 18 that was informed by other researchers and past experience and the number of hidden neurons was selected at 60 based on past experience. For the second run, a mini-experiment was run to more optimally determine a number of features, 24, and hidden neurons, 125, to use. The HMMs remained the same for both runs.

Data Analysis

As discussed in the methodology section, a new data set, named 440K, was created from the full UNSW-NB15 data sets (Full 2.54M). A total of 440,044 sequential records, sorted by time stamp, were chosen from the data set and divided into roughly 60% for training (264,026 records) and 40% for testing (176,018 records). The data set characteristics for normal and attack traffic are shown in Table 2 below and the distribution of the attack categories is given in Table 3.

Table 2: Distribution of Training and Test Data by Traffic Type (440K Data Set)

Traffic Type	Training	Testing	Total
Normal	214,202	136,948	351,150
Attack	49,824	39,070	88,894
Total	264,026	176,018	440,044

Table 3: Distribution of Training and Test Data by Attack Type (440K Data Set)

Attack Type	Training	Testing	Total
Reconnaissance	2,074	1,456	3,530
Exploits	6,901	4,538	11,439
DoS	2,609	2,298	4,907
Generic	34,119	27,759	61,878
Shellcode	227	144	371
Fuzzers	3,168	2,222	5,390
Backdoor	351	315	666
Worms	27	16	43
Analysis	348	322	670
Total	49,824	39,070	88,894

Feature Selection

Two feature selection algorithms, Information Gain (Kayacik et al., 2005) and CFS (Hall, 2000) were run against the training data set to evaluate all 47 non-labeled features. The Information Gain ranking of the 47 features is included in Appendix B.

The CFS evaluation returned a subset of only two features: sttl and ct_state_ttl which are attributes 10 and 37 respectively in Appendix A. Those two features were used to construct two HMMs.

For Information Gain, a cut-off threshold of the first 18 features with the highest ranked merit were chosen for the first run. The lower limit for the number of features was informed by prior feature selection research for UNSW-NB15 and prior experience. Moustafa and Slay (2017) cautioned that reducing the number of features to less than 11 resulted in undesirable results.

To illustrate a more precise cut-off determination, a mini-experiment was done with a backwards feature removal technique to determine the cut-off threshold. A model was run 47 times varying the cut-off threshold from 47 to one. The resulting Accuracy and FPR are shown in figures 4 and 5 respectively below. The highest Accuracy (0.9886) was achieved at a cut-off at 24 features which corresponded to an FPR of 0.0068.

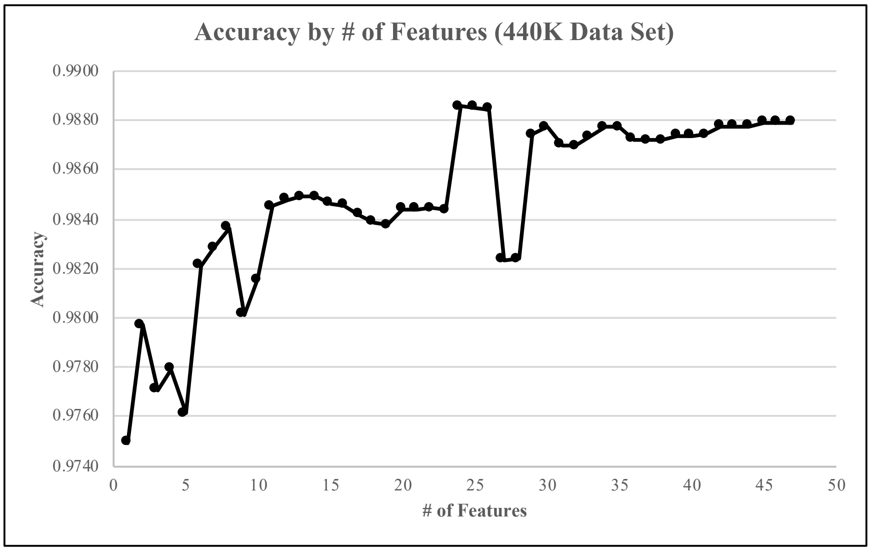


Figure 4: Accuracy by # of Features (440K Data Set)

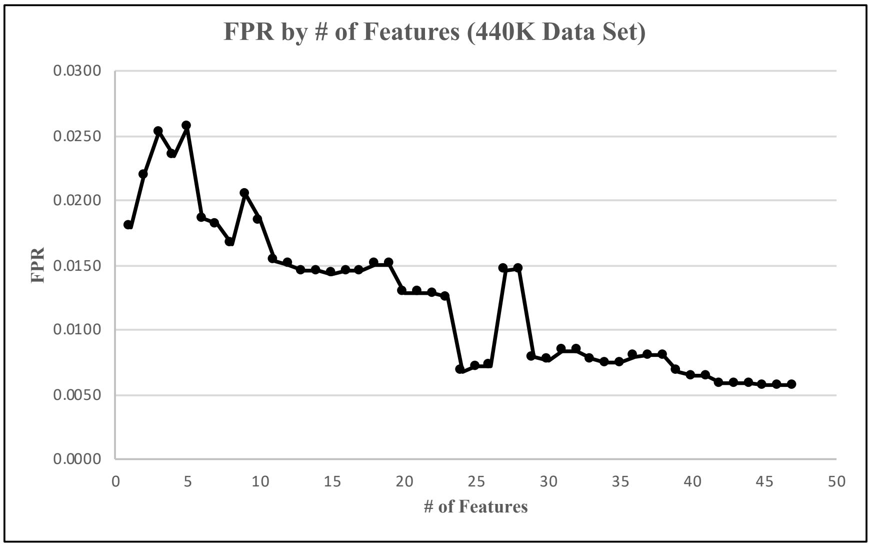


Figure 5: FPR by # of Features (440K Data Set)

Effect of Number of ELM Hidden Neurons on Accuracy and FPR

A separate mini-experiment was conducted to gauge the impact of the number of hidden neurons on Accuracy and FPR. The number of hidden neurons for the ELM model was varied from one to 160. The results for Accuracy and FPR are shown respectively in figures 6 and 7 below. The highest Accuracy (0.9823) occurred at 125 nodes which corresponded to an FPR of 0.0170.

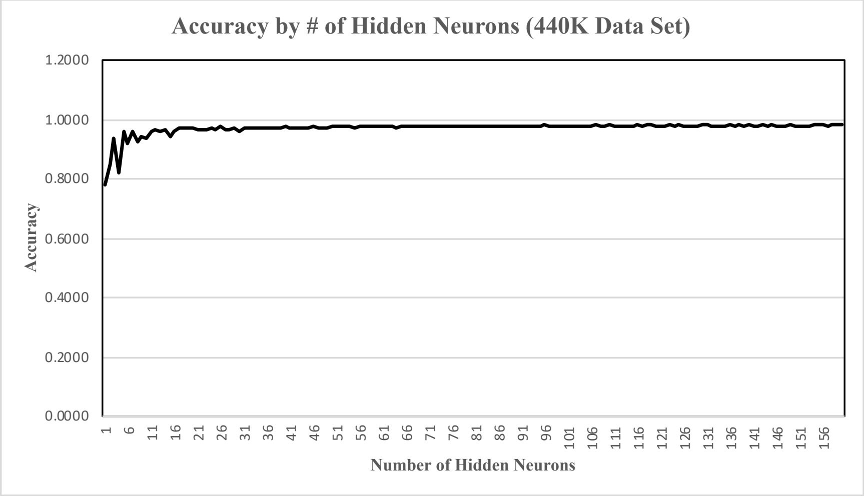


Figure 6: Accuracy by # of ELM Hidden Neurons (440K Data Set)

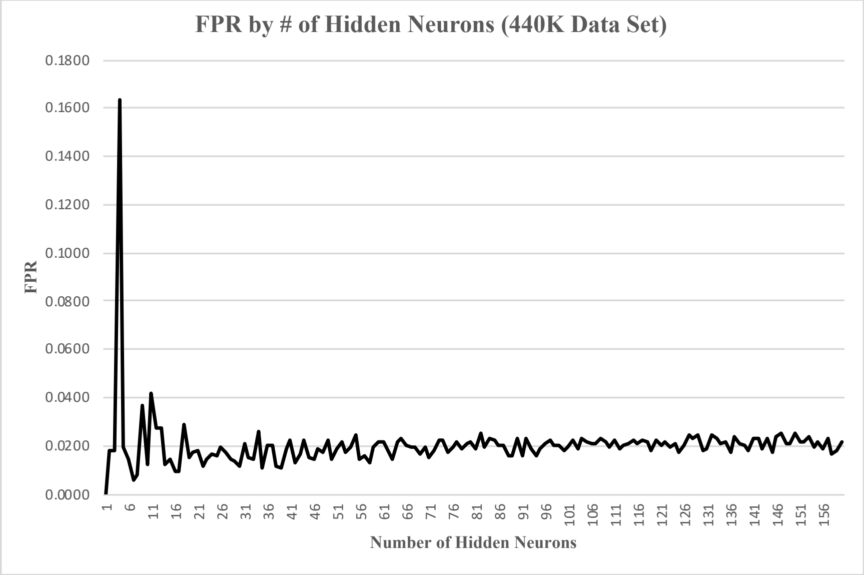


Figure 7: FPR by # of ELM Hidden Neurons (440K Data Set)

HMM Construction

In compact notation, an HMM is notated as λ = (A, B, π) (Rabiner, 1989) where N is the number of states, M is the number of symbols per state, A is the state transition probability, B is the observation symbol emission probability distribution, and π is the initial state distribution.

Both of the created HMMs share the same three states: normal, probe (corresponding to the analysis, fuzzers, and reconnaissance categories), and attack (corresponding to the backdoor, DoS, exploit, generic, shell code, and worm categories). The probe and attack states combined represent all of the attack traffic. The three states are represented by N for Normal, P for Probe, and A for Attack.

For the first HMM (HMM S), the symbols represented all of the possible values for sttl (attribute 10 in Appendix A) which is source to destination Time to Live. This feature is an integer with values between 0 and 255. For the data set, there were 11 distinct values (the M variable for the HMM). These were represented by symbols labeled A through K.

For the second HMM (HMM C) the symbols represent the possible values for ct_state_ttl (attribute 37 in Appendix A) which is a derived value based on state and both destination to source and source to destination Time to Live. This feature is an integer with values between 0 and 6 and has 5 distinct values (the M variable for the HMM) in the data set. These were represented by symbols labeled A through E. Since the two HMMs are separate, there was no conflict with the reuse of the same letters to represent the symbols.

The training of each of the HMMs was performed using similar steps using the training data set. First, the attack category value of each record was translated to the state value (N, A, or P). Then the corresponding feature (either sttl or ct_state_ttl) was mapped to their appropriate symbols. The training file with the known states and given symbols was then used to estimate the initial state distribution (π) and the symbol emission probability distribution (B) using the MATLAB hmmestimate function, which is part of the MATLAB machine learning tool kit (Theodoridis, Pikrakis, Koutroumbas, & Cavouras, 2010). These data were then trained using the Baum-Welch algorithm (Rabiner, 1989), which is the most commonly used HMM training algorithm (Holgado et al., 2017), via the MATLAB hmmtrain function. The resulting parameters for the training of each HMM are included in Appendix C.

For the execution of the HMMs against the test data, the emission symbols for each model were translated from their respective feature. These symbol streams were then run through the Viterbi algorithm (Viterbi, 1967; Forney, 1973) to determine the most likely states using the MATLAB hmmviterbi function. The results were then output to a file for each model for further processing with the ELM model.

The ELM Model

The basic ELM program was modified to provide an output file containing a zero or one predicted value for each row in the test data set for further use in the combination phase. The ELM model was run multiple times following the process outlined in the methodology section.

For experiment A, the ELM model was run once using 18 features and 60 hidden neurons to create a baseline. That model was named ELM1. An additional 10 ELM models were created and labeled ELM2 to ELM11. Models ELM1 to ELM11 all used 18 features with 60 hidden neurons. A new set of ELM models, ELM12 to ELM21, was then created using 24 features with 125 hidden neurons.

The Combined Model

The combined model represents the third situation awareness phase, projection, and is the combined output of the HMMs and ELM providing both a projection in time and space. To eliminate false positives, the outputs of the two HMMs and the ELM classifier were combined using a unanimous voting scheme and resulted in an attack only if all three agreed. Otherwise, the record would be considered normal.

Three ELM models where chosen for combination with the HMMs: ELM1, ELM11, and ELM14. ELM1 was chosen since it was the baseline. ELM11 was chosen since it had the lowest FPR among for the ELMs created using 18 features with 60 hidden neurons. ELM14 was chosen since it had the lowest FPR for the ELMs created with 24 features using 125 hidden neurons.

The combined models were denoted as ELM + HMM followed by the HMMs. So, ELM11+HMM C&S is the combined output of ELM11, HMM C, and HMM S.

Comparison to Other Classifier Models

To provide a comparison to other classifiers, the same training and testing data was used with a J48 DT and an MLP. These are labeled as J48-1 and MLP1 using 18 features and J48-2 and MLP2 using 24 features.

Findings

For each model, the confusion matrix variables TP, TN, FP, and FN were calculated along with FPR and Accuracy. The results sorted by highest FPR to lowest FPR are displayed in Table 4 below.

This shows that the combined model outperformed the individual models and other classifiers for both the 18 feature (ELM11 + HMM C&S) and 24 feature (ELM14 + HMM C&S) models. Table 5 provides a summary view of how the results compared to the primary and secondary goals.

Table 4: Experiment A Results Sorted by Highest to Lowest FPR

Model	Features	TN	FP	FN	TP	Accuracy	FPR
ELM1	18	133168	3780	547	38523	0.9754	0.0276
ELM9	18	133307	3641	573	38497	0.9761	0.0266
ELM10	18	133485	3463	749	38321	0.9761	0.0253
MLP1	18	133529	3419	394	38676	0.9783	0.0250
ELM3	18	133742	3206	762	38308	0.9775	0.0234
ELM18	24	133779	3169	450	38620	0.9794	0.0231
ELM12	24	133790	3158	918	38152	0.9768	0.0231
ELM17	24	133805	3143	833	38237	0.9774	0.0230
ELM5	18	133885	3063	1032	38038	0.9767	0.0224
ELM4	18	133898	3050	979	38091	0.9771	0.0223
ELM20	24	133929	3019	791	38279	0.9784	0.0220
ELM16	24	133934	3014	3300	35770	0.9641	0.0220
ELM6	18	134081	2867	827	38243	0.9790	0.0209
ELM13	24	134127	2821	708	38362	0.9800	0.0206
ELM21	24	134129	2819	2895	36175	0.9675	0.0206
ELM7	18	134148	2800	730	38340	0.9799	0.0204
ELM19	24	134184	2764	749	38321	0.9800	0.0202
MLP2	24	134201	2747	761	38309	0.9801	0.0201
ELM8	18	134201	2747	987	38083	0.9788	0.0201
ELM2	18	134302	2646	992	38078	0.9793	0.0193
ELM15	24	134305	2643	799	38271	0.9804	0.0193
HMM S	1	134597	2351	337	38733	0.9847	0.0172
J48-1	18	134776	2172	775	38295	0.9833	0.0159
ELM11	18	134780	2168	761	38309	0.9834	0.0158
ELM14	24	134915	2033	1212	37858	0.9816	0.0148
ELM1 + HMM S	18	134938	2010	856	38214	0.9837	0.0147
HMM C	1	135643	1305	651	38419	0.9889	0.0095
ELM1 + HMM C	18	135767	1181	1045	38025	0.9874	0.0086
ELM1 + HMM C&S	18	135942	1006	1045	38025	0.9883	0.0073
J48-2	24	136014	934	1078	37992	0.9886	0.0068
ELM14 + HMM S	24	136060	888	1498	37572	0.9864	0.0065
ELM11 + HMM S	18	136074	874	1058	38012	0.9890	0.0064
ELM14 + HMM C	24	136527	421	1716	37354	0.9879	0.0031
ELM14 + HMM C&S	24	136528	420	1716	37354	0.9879	0.0031
ELM11 + HMM C	18	136563	385	1359	37711	0.9901	0.0028
ELM11 + HMM C&S	18	136565	383	1359	37711	0.9901	0.0028

Table 5: Summary of Results Compared to Goals for Experiment A

	Primary Goals				Secondary Goals					
	FPR Reduction of 10% Compared to ELM Alone		FPR less than 0.6%		FPR vs. MLP		FPR v. J48 DT		Accuracy v. ELM Alone	
	% Reduction in FPR	Goal Met	FPR Less than 0.6%	Goal Met	FPR Difference vs. MLP	Goal Met	FPR Difference v. J48 DT	Goal Met	Accuracy Change	Improved/ Reduced
Model										
ELM11 + HMM C&S	-82.33%	Yes	0.28%	Yes	-88.80%	Yes	-82.37%	Yes	0.69%	Improved
ELM14 + HMM C&S	-79.34%	Yes	0.31%	Yes	-84.71%	Yes	-55.03%	Yes	0.46%	Improved

Experiment B (Non-Time Ordered Data)

Experiment B is similar to Experiment A. The primary difference is that the DoS data set was used.

Data Analysis

As discussed in the methodology section, a new data set, named DoS, was created from the UNSW-NB15 Test and Train data set. Table 6 below shows the distribution of traffic by normal and attack. The Training and Testing split follows the original Test and Train data set split of roughly 60% for Training and 40% for Testing. The only attack category included in this data set was DoS.

Table 6: Distribution of DoS Data by Traffic Type (DoS Data Set)

Traffic Type	Training	Testing	Total
Normal	56,000	37,000	93,000
Attack	12,264	4,089	16,353
Total	68,264	41,089	109,353

Feature Selection

The UNSW-NB15 Train and Test data set from which the DoS data set was derived has less features than the Full 2.45M data set. In particular, the authors removed features such as source and destination IP address and ports and record start and end time were removed. The list of available features in the Train and Test data set is included in Appendix D.

The CFS evaluation returned a subset of four features: sttl, proto, dttl, and ct_dst_sport_ltm which are features 10, 2, 11, and 35 respectively in Appendix D. From those, 3 HMMs were built: sttl (HMM DS), dttl (HMM DD), and ct_dst_sport_ltm (HMM DC).

A similar process was followed as for Experiment A using Information Gain and CFS. The result of the Information Gain ranking for the DoS data set is shown in Appendix E. The optimal number of features using a backwards cut-off threshold with Accuracy as a determination was 11 which had an Accuracy of 0.9855 and an FPR of 0.0071. Graphs of Accuracy by features and FPR respectively for the DoS data set is shown in figures 8 and 9 below.

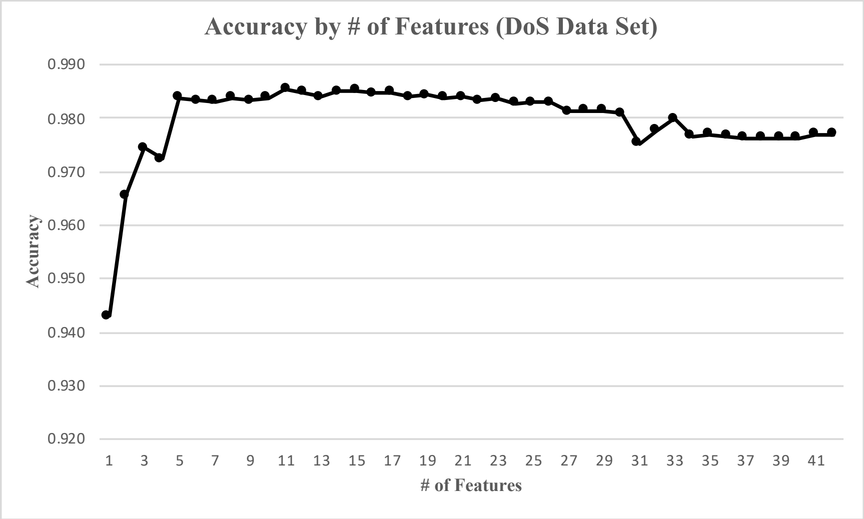


Figure 8: Accuracy by # of Features (DoS Data Set)

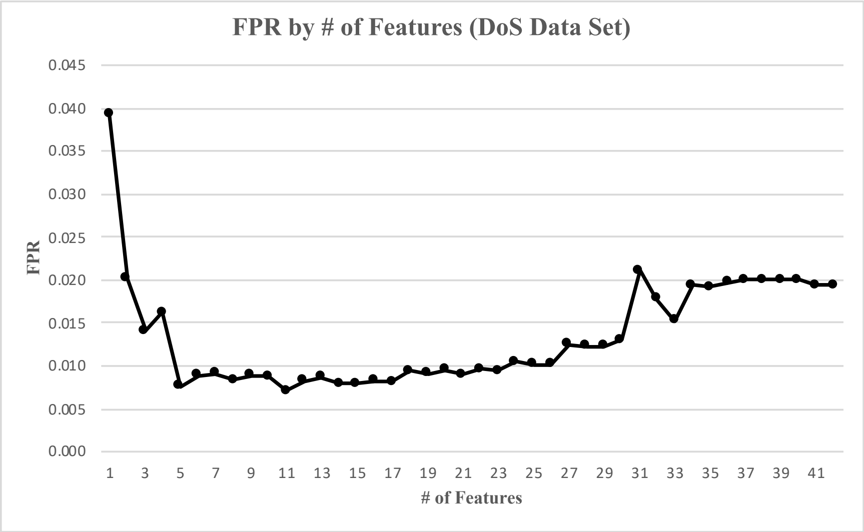


Figure 9: FPR by # of Features (DoS Data Set)

Effect of Number of ELM Hidden Neurons on Accuracy and FPR

As with experiment A, to determine an optimal number of hidden neurons for the ELM, both Accuracy and FPR were evaluated as the number of hidden neurons was increased from 1 to 160. The highest Accuracy occurred at 144 hidden neurons which corresponded to an FPR of 0.0006. This is shown in figures 10 and 11 below.

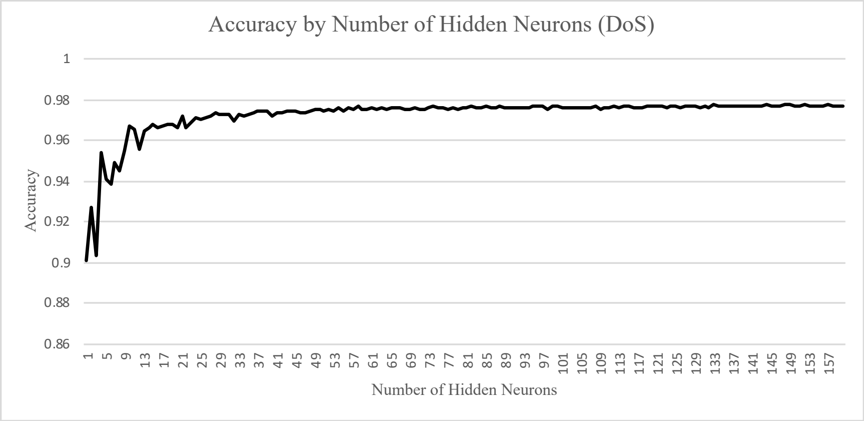


Figure 10: Accuracy by # of ELM Hidden Neurons (DoS Data Set)

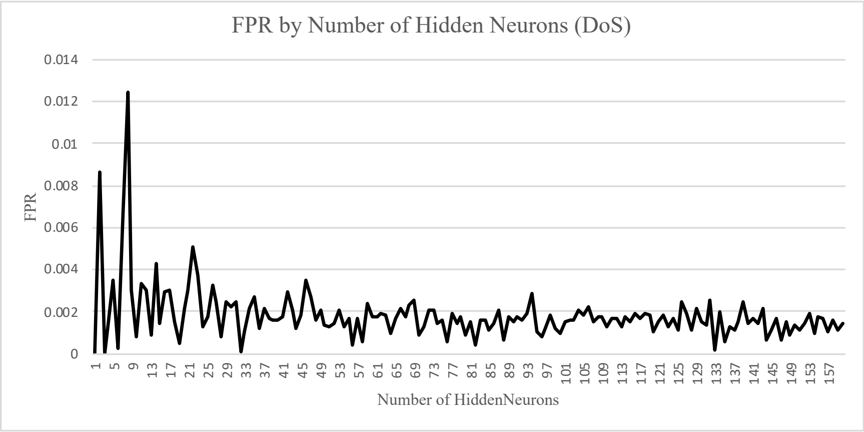


Figure 11: FPR by # of ELM Hidden Neurons (DoS Data Set)

HMM Construction

For this experiment, 3 HMMs were built. The same process as outlined in Experiment A above was used. The three states were represented by N for Normal, P for Probe, and A for Attack. However, since this experiment only used DoS and Normal traffic, the only states expected would be N and A.

For the first HMM (HMM DS), the symbols represented all of the possible values for sttl (attribute 10 in Appendix D) which is source to destination Time to Live. This feature is an integer with values between 0 and 255. For the DoS data set, there were 13 distinct values (the M variable for the HMM). These were represented by symbols labeled A through M.

For the second HMM (HMM DC) the symbols represent the possible values for ct_dst_sport_ltm (attribute 35 in Appendix D) which is a derived feature involving the number of connections between the same destination address and source port. This feature is an integer. There were 16 unique values for this field (the M variable for the HMM). These were represented by symbols labeled A through P.

For the third HMM (HMM DD) the symbols represent the possible values for dttl (attribute 11 in Appendix D) which is the destination to source Time to Live value. This feature is an integer which can vary from 0 to 255. There were 9 unique values for this field (the M variable for the HMM). These were represented by symbols labeled A through I.

As with experiment A, since the HMMs are separate, there was no conflict with the reuse of the same letters to represent the symbols. The training of each of the three HMMs was performed using the same steps as outlined for Experiment A above. The resulting parameters for these HMMs are included in Appendix F.

The ELM Model

For experiment B, the ELM model was run 10 times. The resulting models were named ELMD1 to ELMD10 with the number of hidden neurons set at 144.

The Combined Model

The combined model represents the third situation awareness phase, projection, and is the combined output of the three HMMs and ELM9 providing both a projection in time and space. To eliminate false positives, the outputs of the three HMMs and the ELM classifier were combined using a unanimous voting scheme and resulted in an attack only if all three agreed. Otherwise, the record would be considered normal. The result is labeled ELMD9 + HMM DS, DD, & DC.

Comparison to Other Models

To provide a comparison to other classifiers, the same training and testing data was used with a J48 DT and an MLP. These are labeled as J48 D and MLP D.

Findings

For each model, the confusion matrix variables TP, TN, FP, and FN were calculated along with FPR and Accuracy. The results sorted by highest FPR to lowest FPR are displayed in Table 7 below. Table 8 provides a summary view of how the results compared to the primary and secondary goals.

Table 7: Experiment B Results Sorted by Highest to Lowest FPR

Model	TN	FP	FN	TP	Accuracy	FPR
HMM DD	16932	20068	2	4087	0.5115	0.5424
HMM DS	16932	20068	0	4089	0.5116	0.5424
HMM DC	34677	2323	989	3100	0.9194	0.0628
J48 D	36737	263	332	3757	0.9855	0.0071
ELMD5	36909	91	890	3199	0.9761	0.0025
ELMD8	36916	84	885	3204	0.9764	0.0023
ELMD4	36926	74	887	3202	0.9766	0.0020
ELMD6	36928	72	893	3196	0.9765	0.0019
ELMD2	36936	64	902	3187	0.9765	0.0017
ELMD3	36941	59	900	3189	0.9767	0.0016
ELMD1	36943	57	911	3178	0.9764	0.0015
ELMD7	36943	57	894	3195	0.9769	0.0015
ELMD10	36954	46	890	3199	0.9772	0.0012
ELMD9	36956	44	840	3249	0.9785	0.0012
ELMD9 + HMM DS, DD, & DC	37000	0	1361	2728	0.9669	0.0000
MLP D	37000	0	1235	2854	0.9699	0.0000

Table 8: Summary of Results Compared to Goals for Experiment B

	Primary Goals				Secondary Goals					
	FPR Reduction of 10% Compared to ELM Alone		FPR less than 0.6%		FPR vs. MLP		FPR v. J48 DT		Accuracy v. ELM Alone	
	% Reduction in FPR	Goal Met	FPR less than 0.6%	Goal Met	FPR Difference vs. MLP	Goal Met	FPR Difference v. J48 DT	Goal Met	Accuracy Change	Improved/ Reduced
Model										
ELMD9 + HMM DC & DD & DS	-100.00%	Yes	0.00%	Yes	0.00%	Yes / Tied	-100.00%	Yes	-1.19%	Reduced

Training and Test Time Comparisons

Table 9 below provides a summary of CPU times for training and testing both the 440K and DoS data sets using the same hardware. ELM outperforms J48 DT, HMM, and MLP in terms of training time. MLP takes orders of magnitude longer to train than ELM, J48 DT, or HMM.

Table 9: Training & Testing CPU Time Comparison by Classifier

Data Set	Attributes	Classifier	Training Time (s)	Testing Time (s)
440K	24	ELM (average, n=10)	2.95	0.56
	24	J48 DT	37.50	0.09
	1	HMM (average, n=2)	139.66	0.26
		ELM + HMM	282.27	1.08
	24	MLP	30213.62	11.91
DoS	11	ELM (average, n=10)	0.97	0.19
	11	J48 DT	3.14	0.03
	1	HMM (average, n=3)	6.09	0.05
		ELM + HMM	19.24	0.34
	11	MLP	13058.10	4.13

Comparisons to Other Literature

For additional comparison purposes, Table 10 lists results from 17 research papers published between 2017 and 2019, including a total of 44 algorithms, that were based on the UNSW-NB15 data sets. All included FPR and most included Accuracy results. In some cases, FRP and Accuracy were derived from provided confusion matrices. Care was taken to review the FPR formula used in each paper, to ensure a valid comparison, since some researchers referred to FPR as FAR or Fall Out. Additionally, seven of the key results from Tables 4 and 7 were also replicated in Table 10 for comparison purposes.

Table 10: Comparisons to Other Research Using UNSW-NB15 Data Sorted by FPR

Reference	UNSW-NB15 Data Set	Algorithm	FPR	Accuracy
Yang, Zheng, Wu, Niu, & Yang (2019)	Train & Test Subset	SVM	0.4164	0.7936
Wang, Xu, Lee, & Lee (2018)	Train (50%) & Test Subset	MLP	0.4029	0.8161
Yang, Zheng, Wu, Niu, & Yang (2019)	Train & Test Subset	Multinominal NB	0.3800	0.7675
Wang, Xu, Lee, & Lee (2018)	Train (50%) & Test Subset	C-ELM (CAI)	0.3646	0.8274
Yang, Zheng, Wu, Niu, & Yang (2019)	Train & Test Subset	ANN	0.3611	0.8329
Valero León (2017)	Full 2.54M records	Snort (signature-based)	0.3066	0.6087
Benmessahel, Xie, Chellal, & Semong (2019)	Train & Test Subset	ANN w/ Genetic Algorithm	0.2910	0.8644
Yang, Zheng, Wu, Niu, & Yang (2019)	Train & Test Subset	Deep Belief Network	0.2788	0.8602
Vinayakumar, et al. (2019)	Train & Test Subset (Normal Traffic)	Deep Neural Network	0.2640	0.7970
Wang, Xu, Lee, & Lee (2018)	Train (25%) & Test Subset	SVM	0.2554	0.8587
Yang, Zheng, Wu, Niu, & Yang (2019)	Train & Test Subset	KNN	0.2519	0.8538
Yang, Zheng, Wu, Niu, & Yang (2019)	Train & Test Subset	RF	0.2315	0.8756
Yang, Zheng, Wu, & Yang (2019)	Train & Test Subset	Deep Neural Network	0.1901	0.8908
Yang, Zheng, Wu, Niu, & Yang (2019)	Train & Test Subset	Deep Belief Network	0.1715	0.9021
Benmessahel, Xie, Chellal, & Semong (2019)	Train & Test Subset	ANN w/ Particle Swarm Optimization	0.1478	0.9242
Moustafa, Adi, Turnbull, & Hu (2018)	Inferred Full 2.54M records	KNN	0.1148	0.8664
Benmessahel, Xie, Chellal, & Semong (2019)	Train & Test Subset	ANN w/ Locust Swarm Optimization	0.0940	0.9542
Moustafa, Adi, Turnbull, & Hu (2018)	Inferred Full 2.54M records	SVM	0.0873	0.9260
Tama & Rhee (2019)	Subset 20%	Gradient Boosted Machine (hold out)	0.0860	0.9131
Valero León (2017)	Full 2.54M records	DT	0.0857	0.9228
Moustafa, Adi, Turnbull, & Hu (2018)	Inferred Full 2.54M records	Classification & Regression Tree	0.0851	0.9023
Kamarudin, Maple, Watson, & Safa (2017)	Subset 27K records	Random Forest DT w/ AdaBoost	0.0830	0.9027
Moustafa, Turnbull, & Choo (2018)	DNS Traffic Subset	NB	0.0825	0.9117
Kamarudin, Maple, Watson, & Safa (2017)	Subset 27K records	Random Forest DT w/ LogitBoost	0.0822	0.9033
Muna, Moustafa, & Sitnikova (2018)	Train & Test Subset	Deep Auto Encoder	0.0820	0.9240
Moustafa, Turnbull, & Choo (2018)	DNS Traffic Subset	ANN	0.0787	0.9261
Moustafa, Adi, Turnbull, & Hu (2018)	Inferred Full 2.54M records	Random Forest	0.0656	0.9372
Bamakan, Wang, & Shi (2017)	Train & Test Subset	SVM	0.0612	0.8465
Moustafa, Creech, & Slay (2017)	Train & Test Subset	Finite Dirichlet Mixture Model	0.0588	0.9430
Moustafa, Turnbull, & Choo (2018)	DNS Traffic Subset	DT	0.0522	0.9532
Moustafa, Slay, & Creech (2017)	Inferred Train & Test Subset	Beta Mixture Model	0.0510	0.9280
Tchakoucht & Ezziyyani (2018)	Train & Test Subset	Recurrent Neural Network	0.0510	--
Moustafa, Adi, Turnbull, & Hu (2018)	Inferred Full 2.54M records	Outlier Gaussian Mixture	0.0472	0.9519
Moustafa, Turnbull, & Choo (2018)	HTTP Traffic Subset	ANN	0.0426	0.9627
Moustafa, Turnbull, & Choo (2018)	HTTP Traffic Subset	NB	0.0418	0.9591
Moustafa, Adi, Turnbull, & Hu (2018)	Inferred Full 2.54M records	Beta Mixture HMM	0.0382	0.9632
Moustafa, Turnbull, & Choo (2018)	HTTP Traffic Subset	DT	0.0343	0.9713
Tama & Rhee (2019)	Subset 20%	Gradient Boosted Machine (10 fold)	0.0297	0.9508
Moustafa, Turnbull, & Choo (2018)	HTTP Traffic Subset	Ensemble	0.0258	0.9897
Vinayakumar, et al. (2019)	Train & Test Subset (Normal Traffic)	NB	0.0250	0.8370
This Research (from Table 4)	Subset 440K records	MLP1	0.0250	0.9783
Bamakan, Wang, & Shi (2017)	Train & Test Subset	SVM	0.0246	0.9352
This Research (from Table 4)	Subset 440K records	MLP2	0.0201	0.9801
This Research (from Table 4)	Subset 440K records	DT (J48 - 1)	0.0159	0.9833
This Research (from Table 4)	Subset 440K records	ELM11	0.0158	0.9834
Moustafa, Turnbull, & Choo (2018)	DNS Traffic Subset	Ensemble	0.0138	0.9954
Igbe (2019)	Train & Test (DoS)	Artificial Immune System	0.0134	0.9811
This Research (from Table 7)	Train & Test (DoS)	J48 D	0.0071	0.9855
This Research (from Table 4)	Subset 440K records	DT (J48 - 2)	0.0068	0.9886
Alhaidari & Zohdy (2018)	Training/Testing (80%/20%)	HMM	0.0038	0.9641
This Research (from Table 4)	Subset 440K records	ELM14 + HMM C&S	0.0031	0.9879
This Research (from Table 4)	Subset 440K records	ELM11 + HMM C&S	0.0028	0.9901
This Research (from Table 7)	Train & Test (DoS)	ELM9 + HMM DS, DD, &DC	0.0000	0.9669
This Research (from Table 7)	Train & Test (DoS)	MLP D	0.0000	0.9699

Summary of Results

The primary evaluation metric for this research was FPR. For Experiment A (time-ordered data), Tables 4 and 5 demonstrate that the combined model of ELM with HMMs in a situation awareness framework using a unanimous voting scheme achieved the goals of an FPR reduction of 10%, an FPR under 0.6%, and a better FPR than either a J48 DT or MLP classifier using the same data sets. This was achieved for both models using 18 attributes as well as 24 attributes. However, the difference between 18 and 24 attributes and 60 and 125 hidden neurons was not overly significant as shown in Table 4.

For experiment B, which was using DoS traffic (non-time ordered), the primary goals of an FPR reduction of 10% and an FPR under 0.6% were also met. The secondary goals of a better FPR than either a J48 DT or MLP classifier were achieved for the J48 DT but resulted in a tie for the MLP classifier since both solutions had an FPR of 0%. With the FPR being 0%, that was a 100% reduction from ELM alone but that resulted in a slight degraded Accuracy rate. Also, for experiment B, two of the three HMMs had a perfect and almost perfect TPR but very poor FPR performance. However, due to the unanimous voting scheme, the very high rates of false positives were cancelled out for the overall ensemble.

Chapter 5

Conclusions, Summary, Implications, & Recommendations

This research demonstrates that false positives can be better minimized, while maintaining detection accuracy, by combining ELM and HMMs as classifiers within the context of a situation awareness framework. This research was performed using the UNSW-NB15 data set which is more representative of contemporary cyber-attack and normal network traffic than older data sets typically used in IDS research. It is shown that this approach provides better results than either HMM or ELM alone and with a lower FPR than other comparable approaches that also used the UNSW-NB15 data set.

Conclusions

The overall approach proved effective for both time-ordered and non-time ordered data but for likely different reasons. For the 440K data set, the two HMMs demonstrated very good Accuracy rates even as a single attribute classifier. The Accuracy was 0.9889 and 0.9847 for HMMs C and S respectively. The Information Gain merit for those attributes, from Appendix B, was 0.614 and 0.593 respectively which wouldn't seem to support those type of Accuracy results. Thus, the Accuracy is likely due to a boost in predictive ability due to patterns in the time sequence data associated with each of those attributes.

However, for the DoS data set, two of the HMMs, DS and DD, both had poor Accuracy rates of 0.5424 and very high FPRs. DS was also constructed from the feature sttl which is the same feature used for HMM S for the 440K data set. For the DoS data set sttl showed an Information Gain merit of 0.300 (Appendix E). HMM DD was created from feature dttl, destination Time to Live, and had a merit of 0.316. The overall result would seem to indicate that there was no temporal pattern to boost the individual Accuracy. The third HMM, HMM DC, based on feature ct_dst_sport_ltm had a merit of only 0.165 but performed much better as a single attribute classifier with an Accuracy of 0.9194 as compared to HMMs DS & DD. Overall, the poor HMM individual performance from the DoS data set as compared to the 440K data set supports evidence of a temporal pattern in the time-ordered data. Given most real-world IDSs ingest such time ordered data, this boosts the case for evaluations using time-ordered data sets vs. those that have been shuffled from techniques such as cross-validation.

Of other note for feature selection is that sttl showed up in the CFS selection subsets for both 440K and DoS. This is likely since sttl was noted as being a feature that has relevance in more than one type of attack (Moustafa & Slay, 2017; Janarthanan & Zargari, 2017). So, given the 440K data set has nine categories of attack, that makes the feature more valuable when the task is to detect any attack versus a particular type of attack.

So, despite the poor performance of the DoS HMMs, the overall result still ended up at an FPR of 0%. This highlights one of the benefits of a unanimous voting scheme. All it takes is one classifier to disagree on an attack to negate incorrect decisions by other classifiers. HMM DS had a perfect TPR of 100% while HMM DD was virtually at 100%. But the excessive number of FPs significantly hurt both the FPR and the Accuracy calculations. However, the ELMs had generally low FPs for the DoS data set. Thus, when combined, there was a perfect FPR at only a slightly reduced Accuracy. It should be noted that the MLP also had a perfect FPR of 0%. This is likely since the DoS attacks used to create the data set have more distinctive patterns than other types of attack and are less complex to model.

In terms of comparison to other research papers using UNSW-NB15, the approach in this research compared very favorably to FPR and Accuracy metrics from other research in Table 10. However, many researchers use different parts of UNSW-NB15 and sometimes they are not clear on the parts they are using and the attributes chosen. Thus, the comparisons to other research are indirect but still directional.

The most direct comparison from the table is for Igbe (2019) who used an Artificial Intelligence System scheme on UNSW-NB15 Test and Train data set for just Normal and DoS traffic. Igbe (2019) had achieved an FPR of 0.0134 with an Accuracy of 0.9811. That compares to the FPR of 0.0000 and an Accuracy of 0.9669 for the ELM plus HMM solution proposed by this research. So, while this solution achieved a better FPR, the goal of the research, Igbe (2019) did have a slightly better Accuracy.

In terms of training time, the results of this research confirm the ELM trains very quickly. As shown in Table 9, for the DoS data set, the ELM trained in 0.97 seconds of CPU time. The MLP took 13058.10 CPU seconds on the same hardware. That is one second to train for the ELM versus more than 3 hours for the MLP. The ELM combined with the three HMM models took 19.24 seconds to train. The MLP, in this case, achieved the same 0% FPR as the combined model with a 0.3% increase in Accuracy. The question is whether that is worth an extra three plus hours of CPU time which, of course, would be application dependent. For the 440K data set, the MLP took over eight hours of CPU time compared to just under three seconds for the ELM model. Other research has shown that ELM needs up to four orders of magnitude in less time to train as compared to ANNs using Back Propagation (Wang & Huang, 2005). This research supports that conclusion.

The experiment with the number of hidden neurons to use for ELM confirms that ELM is very stable across a wide range of hidden nodes but performance can degrade with too few or too many neurons (Huang, Zhu, and Siew, 2006). As seen in figures six through nine, the Accuracy of the system rises quickly as neurons are added from zero but eventually levels off with smaller improvements as more neurons are added. The FPR results tend to oscillate in a band once the system approaches a steady state as more neurons are added.

Differences Among Algorithms

To summarize the difference in algorithm performance in terms of Accuracy and FPR, the HMMs performed very well on the time-ordered 440K data set as shown in Table 4. They had better FPRs individually than both MLP models that used 18 and 24 features as well as several of the 18 and 24 feature ELM models. The HMMs performed much better than would be expected from their information gain merit alone as single feature classifiers. This was most likely due to their detection of temporal patterns in the data. This conclusion is further supported by the results of the non-time ordered DoS Train & Test data set.

In the DoS case, two of the three HMMs performed poorly as shown in Table 7 as was anticipated from non-time ordered data. This performance is closer to what would have been expected for a single feature classifier based on their information gain merit. However, HMM DC performed relatively well. This is likely since it was a derived feature involving the number of connections between the same destination address and source port and that was a good indicator for detecting the type of DoS attacks used in the data set.

The J48 DTs had mixed results with generally good performance on the 440K data set where both the 18 and 24 feature versions had a better FPR than many of the ELM models, some of the HMM models, and both of the MLP models. However, both the MLP and all of the ELM models had a lower FPR than the J48 DT for the DoS data set. And the MLP had one of the best performances on the DoS data set with a 0% FPR.

Overall, accounting for training time and results, this research supports prior findings that ELM has shown its versatility to minimize false positives (Zong et al., 2013; Fossaceca et al., 2015). ELM also had the lowest training times of any of the algorithms as shown in Table 9 especially when compared to MLPs.

The one constant among algorithms is that the combined model using a unanimous voting scheme had the best performance (or was tied for best performance) for FPR. Lin et al. (2003) had previously demonstrated that a combination of classifiers can result in a significant accuracy improvement.

Feature Selection Anomalies

For feature selection using Information Gain on the 440K data set, two data points representing features stand out on Figures 4 and 5 which graph the Accuracy and FPR by number of features respectively. Both have a lower Accuracy and higher FPR than would be indicated by a trend line. A similar anomaly is seen with the DoS data set for two feature points as shown in Figures 8 and 9. While inconsequential, since none of the features in question were chosen for the experiments, the most likely explanation is that there was a slight variation in distributions between the respective test and training data that caused some observations to skew which can affect the FPR. This phenomenon was discussed in the statistical analysis of the UNSW-NB15 data set by Moustafa and Slay (2016).

Summary

A primary purpose of this research was to extend promising emerging research into ELM as a classifier, in conjunction with HMMs using a situation awareness framework, to minimize false positives while maintaining accuracy. That goal was achieved given the evaluation criteria.

More broadly, the intent was to further the use of ML based anomaly detection techniques which have shown promise of detecting indications of novel cyber-attacks but still typically generate more false positives than signature-based methods (Pao et al., 2015). Incidentally, Valero León (2018), who created an ML based model using UNSW-NB15 data, used a version of Snort with the then most recent signatures and achieved an FPR of 30.66% compared to 8.57% with an anomaly-based method, DT, using the same training and testing data. Hence, in this one instance, a signature-based approach generated a much higher than expected FPR.

Another research goal was to use a recent and relevant set of comprehensive benchmark data, containing both cyber-attack and normal traffic. This research used UNSW-NB15 (Moustafa & Slay, 2015) which achieved that goal. Researchers such as Sommer and Paxton (2010), pointed out that the cybersecurity landscape has changed significantly since the creation of the KDD99 data and that they do not consider experiments that use older data sets such as KDD99 relevant anymore. And more recently, Khammassi and Krichen (2017) had stated that the NIDS research community now considers UNSW-NB15 a new benchmark data set to be used for evaluations of IDSs. However, there is still a gap in the literature on ample published research for comparison purposes which will likely take some time to close. But given that 17 papers (as shown in Table 10) published between 2017 through mid-year 2019 provided comparable results for FPR using UNSW-NB15, this gap is clearly closing.

However, given the needs of different researchers, more than 10 different portions up to and including the full data set of UNSW-NB15 (as listed in the UNSW-NB15 data set column) were used in the 17 papers cited. Different research also used different feature selection methods. But these still provide a better comparison to the literature than using results from older data sets such as KDD99.

This research also used the three situation awareness components defined by Endsley (1988). The first stage, perception, was accomplished with a network-based (NIDS) anomaly-detection engine for intrusion detection using an ELM as the primary classifier.

The second stage, comprehension, which achieved the goal of further reducing false positives and improving accuracy, was accomplished using a post processing approach and implemented using HMMs. Treinen and Thurimella (2009) stated that HMMs are very effective at detecting all types of attacks by acting as an anomaly detector over a set of IDS alarms thus providing a low rate of false positives and high rate of alarm reduction.   The results in Tables 5 and 8 demonstrate that point with significant reductions in FPR after combining the ELM results with the HMMs used for this research. The temporal aspect from the HMMs provided better context around interactions in time compared to a more traditional IDS approach that is just looking at a single event in space.

The third phase, projection, was a decision informed by the perception and comprehension stages, in space and time, on whether a given alert from the perception stage should be put forth to a human operator or an automated action as an intrusion alert. This was the combined output of the ELM and HMM classifiers.

An ELM was chosen as a classifier for the perception phase based on the positive results as a classifier for IDS (Fossaceca et al., 2015). While ELM research only goes back to Huang, Zhu, and Siew (2004) with the first IDS-based research from Cheng et al. (2012), ELM has shown its versatility to minimize false positives (Zong et al., 2013). But given the random initialization, ELM results can vary quite a bit, within a range, as seen in Tables 4 and 7. To mitigate this, multiple runs can be considered as part of a training and validation step prior to testing.

A combined approach was also used since prior research had shown positive results in reducing false positives in IDS using post processing (Spathoulas & Katsikas, 2013b). Other hybrid results had yielded positive results such as using a KNN followed by ELM (Akusok et al., 2014) and a NB to HMM (Karthick et al., 2012) for malware detection.

The primary evaluation metric for this research was FPR. Table 5 demonstrated that the combined classifier did have the lowest or the same FPR including against an MLP and J48 DT using the same testing and training data. The goal was that the addition of the HMMs would reduce the FPR by at least 10% compared to the ELM classifier alone and have an FPR of less than 0.6%. Those goals were exceeded.

This approach is novel as an extension of drawing upon several separate research results by combining the positive aspects of each one. While there have been various hybrids using either HMM or ELM for IDS, it is believed this approach was the first to use a combination of an ELM with multiple HMMs for IDS. It is further unique in its use of these classifiers in a situation awareness framework for network intrusion detection.

Implications

Given the increased stakes as a result of cyber-attacks, the need for faster and more automated responses to indications of a potential cyber-attack is critical. Many commercial IDSs and related systems are capable of blocking traffic in real-time, by source IP address for example, based on intrusion alerts. However, as discussed, an action based on a false positive with an automated response could deny resource access to legitimate users or tasks that could be unacceptable based on the circumstance. Thus, steps, such as demonstrated by this research, to further reduce false positives will help to achieve better threat detection and response.

Recommendations

While this research has highlighted the versatility of ELM and the ability to make ELM better using situation awareness in conjunction with HMMs, there is ample opportunity to extend this research to both further reduce FRP and improve accuracy. One area of additional research would be to evaluate the feature selection process. This research used both information gain (Kayacik et al., 2005) and CFS (Hall, 2000). However, other studies that were cited, even those using similar feature selection methods, have come up with widely varying results using different subsets of UNSW-NB15. So, further research on feature selection methods would be warranted. Likewise, a further evaluation of which features produce the best temporal results with HMMs, beyond the features chosen, would likely provide other interesting insights.

The choice of testing and training data is another area for study. Different research has used different portions of the UNSW-NB15 data set. Questions such as the optimal training and testing data split, how varying the number of records impacts the results, and how to best choose sequential data for time-series analysis could be further studied.

Other researchers, as discussed, had also modified the ELM algorithm itself. For example, Castaño et al. (2013) introduced PCA-ELM which eliminated the random initialization of the weights and determined them based on a PCA of the training data. So, another avenune of further research would be to evaluate variations of the ELM algorithms and their impact on FPR. Variations of key ELM parameters such as the optimal number of hidden neurons and the best activation function could also be further explored.

For the HMM classifiers, experiments could be done to determine if more than two or three HMM classifiers improve the results. For the third phase of projection, this research used a unanimous voting scheme where a record was marked as an attack only if the ELM and the HMM models agreed. Variations on this could also be explored such as by weighting the results from each model and using discrete thresholds for the ELM and HMMs instead of binary ones for each model.

Another avenue of inquiry would be to determine how ELM, or a similar combination with HMMs, could better reduce the FNR while also maintaining Accuracy. This could help to better ferret out indications of compromise which might not alarm. Overall, this research provided promising results and has hopefully provided insights for additional exploration and improvement opportunities.

Appendix A

UNSW-NB15 Features Description for Experiment A (Full 2.54M) (Moustafa & Slay, 2015)

Table 11: UNSW-NB15 Features Description

No.	Name	Type	Description
			Flow Features
1	srcip	nominal	Source IP address
2	sport	integer	Source port number
3	dstip	nominal	Destination IP address
4	dsport	integer	Destination port number
5	proto	nominal	Transaction protocol
			Basic Features
6	state	nominal	Indicates to the state and its dependent protocol, e.g. ACC, CLO, CON, ECO, ECR, FIN, INT, MAS, PAR, REQ, RST, TST, TXD, URH, URN, and (-) (if not used state)
7	dur	Float	Record total duration
8	sbytes	Integer	Source to destination transaction bytes
9	dbytes	Integer	Destination to source transaction bytes
10	sttl	Integer	Source to destination time to live value
11	dttl	Integer	Destination to source time to live value
12	sloss	Integer	Source packets retransmitted or dropped
13	dloss	Integer	Destination packets retransmitted or dropped
14	service	nominal	http, ftp, smtp, ssh, dns, ftp-data ,irc and (-) if not much used service
15	Sload	Float	Source bits per second
16	Dload	Float	Destination bits per second
17	Spkts	integer	Source to destination packet count
18	Dpkts	integer	Destination to source packet count
			Content Features
19	swin	integer	Source TCP window advertisement value
20	dwin	integer	Destination TCP window advertisement value
21	stcpb	integer	Source TCP base sequence number
22	dtcpb	integer	Destination TCP base sequence number
23	smeansz	integer	Mean of the flow packet size transmitted by the src
24	dmeansz	integer	Mean of the flow packet size transmitted by the dst
25	trans_depth	integer	Represents the pipelined depth into the connection of http request/response transaction
26	res_bdy_len	integer	Actual uncompressed content size of the data transferred from the serverís http service.
			Time Features
27	Sjit	Float	Source jitter (mSec)
28	Djit	Float	Destination jitter (mSec)
29	Stime	Timestamp	record start time
30	Ltime	Timestamp	record last time
31	Sintpkt	Float	Source interpacket arrival time (mSec)
32	Dintpkt	Float	Destination interpacket arrival time (mSec)
33	tcprtt	Float	TCP connection setup round-trip time, the sum of ísynackí and íackdatí.
34	synack	Float	TCP connection setup time, the time between the SYN and the SYN_ACK packets.
35	ackdat	Float	TCP connection setup time, the time between the SYN_ACK and the ACK packets.
			Additional Generated Features
36	is_sm_ips_ports	Binary	If source (1) and destination (3) IP addresses equal and port numbers (2)(4) equal then, this variable takes value 1 else 0
37	ct_state_ttl	Integer	No. for each state (6) according to specific range of values for source/destination time to live (10) (11).
38	ct_flw_http_mthd	Integer	No. of flows that has methods such as Get and Post in http service.
39	is_ftp_login	Binary	If the ftp session is accessed by user and password then 1 else 0.
40	ct_ftp_cmd	integer	No of flows that has a command in ftp session.

No.	Name	Type	Description
			Connection Features
41	ct_srv_src	integer	No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26).
42	ct_srv_dst	integer	No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26).
43	ct_dst_ltm	integer	No. of connections of the same destination address (3) in 100 connections according to the last time (26).
44	ct_src_ ltm	integer	No. of connections of the same source address (1) in 100 connections according to the last time (26).
45	ct_src_dport_ltm	integer	No of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26).
46	ct_dst_sport_ltm	integer	No of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26).
47	ct_dst_src_ltm	integer	No of connections of the same source (1) and the destination (3) address in in 100 connections according to the last time (26).
			Labelled Features
48	attack_cat	nominal	The name of each attack category. In this data set, nine categories: Fuzzers, Analysis, Backdoors, DoS Exploits, Generic, Reconnaissance, Shellcode and Worms
49	Label	binary	0 for normal and 1 for attack records

Appendix B

Information Gain Analysis of UNSW-NB15 Features for Experiment A

Table 12: Experiment A: Information Gain Analysis of UNSW NB-15 Features

Rank	Attribute #	Attribute Name	Merit
1	8	sbytes	0.637
2	3	dstip	0.629
3	1	srcip	0.618
4	37	ct_state_ttl	0.614
5	10	sttl	0.593
6	15	Sload	0.563
7	23	smeansz	0.541
8	11	dttl	0.330
9	9	dbytes	0.329
10	24	dmeansz	0.303
11	7	dur	0.290
12	32	Dintpkt	0.278
13	16	Dload	0.274
14	18	Dpkts	0.261
15	46	ct_dst_sport_ltm	0.258
16	45	ct_src_dport_ltm	0.214
17	33	tcprtt	0.212
18	34	synack	0.209
19	35	ackdat	0.209
20	4	dsport	0.204
21	31	Sintpkt	0.201
22	6	state	0.201
23	43	ct_dst_ltm	0.196
24	2	sport	0.192
25	44	ct_src_ ltm	0.167
26	17	Spkts	0.165
27	27	Sjit	0.161
28	28	Djit	0.160
29	47	ct_dst_src_ltm	0.156
30	42	ct_srv_dst	0.154
31	41	ct_srv_src	0.146
32	13	dloss	0.141
33	12	sloss	0.133
34	5	proto	0.122
35	19	swin	0.083
36	21	stcpb	0.082
37	22	dtcpb	0.082
38	20	dwin	0.082
39	29	Stime	0.081
40	30	Ltime	0.078
41	14	service	0.059
42	26	res_bdy_len	0.031
43	25	trans_depth	0.001
44	36	is_sm_ips_ports	0.000
45	38	ct_flw_http_mthd	0.000
46	40	ct_ftp_cmd	0.000
47	39	is_ftp_login	0.000

Appendix C

HMM Parameters for Experiment A

For HMM S

Initial estimates:

```
estTR =

    0.9776    0.0070    0.0154
    0.2503    0.2610    0.4887
    0.0769    0.0596    0.8635


estE =

  Columns 1 through 8

    0.0027    0.0011    0.0000    0.7722    0.1919    0.0046    0.0004    0.0028
    0.0021    0         0         0         0         0.0134    0         0
    0.0014    0         0         0         0         0.0737    0         0

  Columns 9 through 11

    0.0000    0.0240    0.0000
    0         0.9843    0.0002
    0         0.9249    0
```

After Baum-Welch algorithm:

```
estTR1 =

    0.9703    0.0200    0.0097
    0.4876    0.4560    0.0564
    0.0474    0.0113    0.9413


estE2 =

  Columns 1 through 8

    0.0008    0.0012    0.0000    0.7804    0.1939    0.0022    0.0004    0.0029
    0.0562    0         0         0         0         0.3591    0         0
    0.0000    0         0         0         0         0.0169    0         0
  Columns 9 through 11

    0.0000    0.0181    0.0000
    0         0.5843    0.0004
    0         0.9831    0
```

For HMM C

Initial estimates:

```
estTRC =

    0.9776    0.0070    0.0154
    0.2503    0.2610    0.4887
    0.0769    0.0596    0.8635


estEC =

    0.9723    0.0174    0.0064    0.0021    0.0018
    0.0030    0.5363    0.4453    0         0.0154
    0.0040    0.1179    0.8735    0.0010    0.0036
```

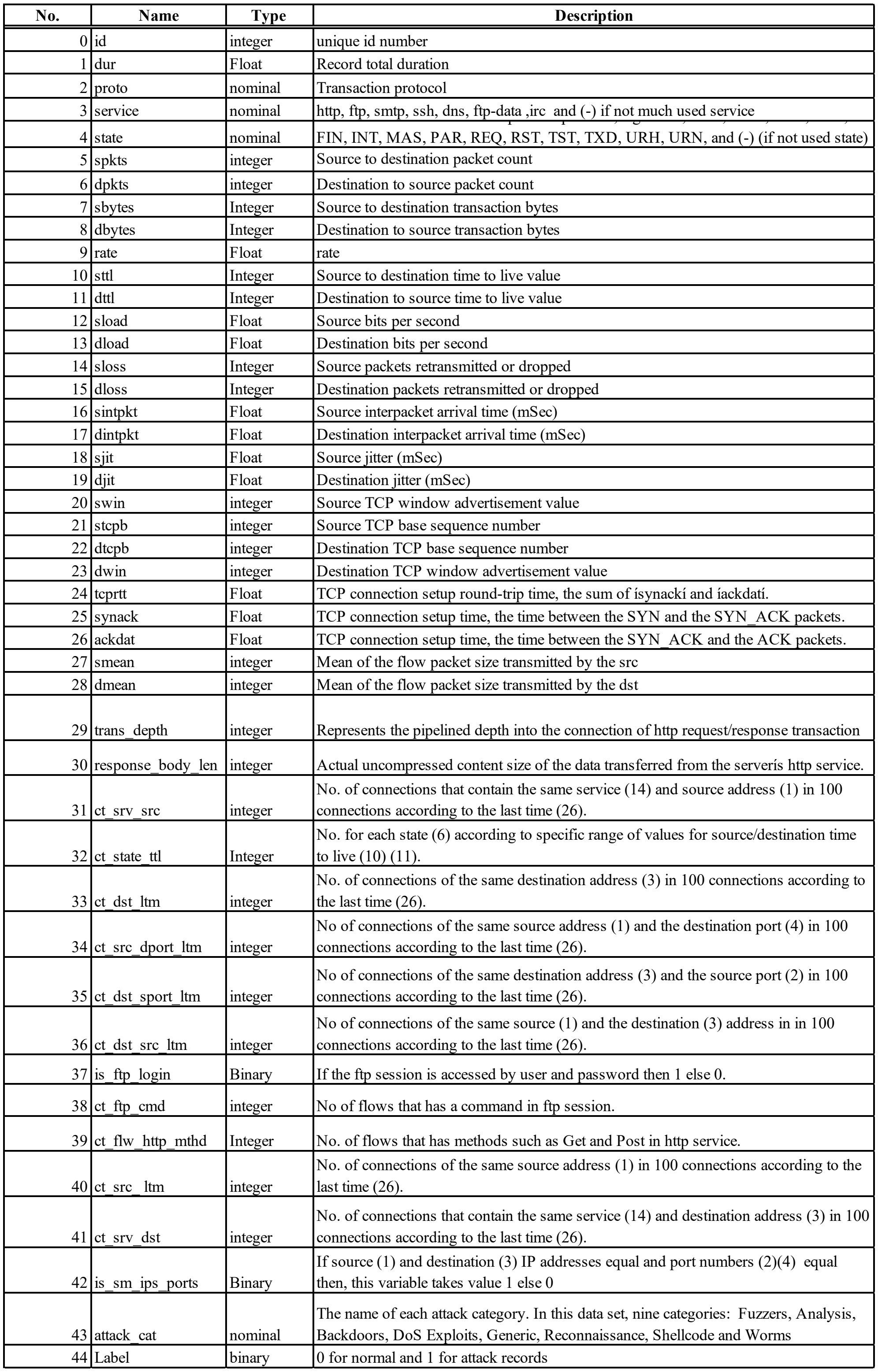
After Baum-Welch algorithm:

```
estTRC1 =

    0.9710    0.0206    0.0084
    0.4577    0.4734    0.0689
    0.0439    0.0138    0.9423


estEC1 =

    0.9812    0.0128    0.0032    0.0023    0.0005
    0.0000    0.8028    0.1434    0         0.0538
    0.0001    0.0399    0.9598    0.0002    0.0001
```

Appendix D

UNSW-NB15 Features Description for Experiment B (Train and Test) (Moustafa & Slay, 2015; Moustafa & Slay, 2016)

Table 13: UNSW-NB15 Features Description for the Train and Test Data Set



No.	Name	Type	Description
0	id	integer	unique id number
1	dur	Float	Record total duration
2	proto	nominal	Transaction protocol
3	service	nominal	http, ftp, smtp, ssh, dns, ftp-data ,irc and (-) if not much used service
4	state	nominal	FIN, INT, MAS, PAR, REQ, RST, TST, TXD, URH, URN, and (-) (if not used state)
5	spkts	integer	Source to destination packet count
6	dpkts	integer	Destination to source packet count
7	sbytes	Integer	Source to destination transaction bytes
8	dbytes	Integer	Destination to source transaction bytes
9	rate	Float	rate
10	sttl	Integer	Source to destination time to live value
11	dttl	Integer	Destination to source time to live value
12	sload	Float	Source bits per second
13	dload	Float	Destination bits per second
14	sloss	Integer	Source packets retransmitted or dropped
15	dloss	Integer	Destination packets retransmitted or dropped
16	sintpkt	Float	Source interpacket arrival time (mSec)
17	dintpkt	Float	Destination interpacket arrival time (mSec)
18	sjit	Float	Source jitter (mSec)
19	djit	Float	Destination jitter (mSec)
20	swin	integer	Source TCP window advertisement value
21	stcpb	integer	Source TCP base sequence number
22	dtcpb	integer	Destination TCP base sequence number
23	dwin	integer	Destination TCP window advertisement value
24	tcprtt	Float	TCP connection setup round-trip time, the sum of ísynackí and íackdatí.
25	synack	Float	TCP connection setup time, the time between the SYN and the SYN_ACK packets.
26	ackdat	Float	TCP connection setup time, the time between the SYN_ACK and the ACK packets.
27	smean	integer	Mean of the flow packet size transmitted by the src
28	dmean	integer	Mean of the flow packet size transmitted by the dst
29	trans_depth	integer	Represents the pipelined depth into the connection of http request/response transaction
30	response_body_len	integer	Actual uncompressed content size of the data transferred from the serverís http service.
31	ct_srv_src	integer	No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26).
32	ct_state_ttl	Integer	No. for each state (6) according to specific range of values for source/destination time to live (10) (11).
33	ct_dst_ltm	integer	No. of connections of the same destination address (3) in 100 connections according to the last time (26).
34	ct_src_dport_ltm	integer	No of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26).
35	ct_dst_sport_ltm	integer	No of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26).
36	ct_dst_src_ltm	integer	No of connections of the same source (1) and the destination (3) address in in 100 connections according to the last time (26).
37	is_ftp_login	Binary	If the ftp session is accessed by user and password then 1 else 0.
38	ct_ftp_cmd	integer	No of flows that has a command in ftp session.
39	ct_flw_http_mthd	Integer	No. of flows that has methods such as Get and Post in http service.
40	ct_src_ ltm	integer	No. of connections of the same source address (1) in 100 connections according to the last time (26).
41	ct_srv_dst	integer	No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26).
42	is_sm_ips_ports	Binary	If source (1) and destination (3) IP addresses equal and port numbers (2)(4) equal then, this variable takes value 1 else 0
43	attack_cat	nominal	The name of each attack category. In this data set, nine categories: Fuzzers, Analysis, Backdoors, DoS Exploits, Generic, Reconnaissance, Shellcode and Worms
44	Label	binary	0 for normal and 1 for attack records

Appendix E

Information Gain Analysis for Experiment B

Table 14: Experiment B: Information Gain Analysis of UNSW NB-15 Features

Rank	Attribute #	Attribute Name	Merit
1	7	sbytes	0.546
2	27	smean	0.479
3	2	proto	0.453
4	12	sload	0.442
5	8	dbytes	0.356
6	9	rate	0.356
7	1	dur	0.346
8	28	dmean	0.323
9	11	dttl	0.316
10	32	ct_state_ttl	0.305
11	10	sttl	0.300
12	17	dinpkt	0.287
13	13	dload	0.285
14	6	dpkts	0.283
15	4	state	0.244
16	16	sinpkt	0.229
17	25	synack	0.188
18	24	tcprtt	0.188
19	5	spkts	0.183
20	26	ackdat	0.179
21	35	ct_dst_sport_ltm	0.165
22	19	djit	0.164
23	14	sloss	0.157
24	18	sjit	0.154
25	15	dloss	0.152
26	20	swin	0.118
27	21	stcpb	0.111
28	22	dtcpb	0.111
29	23	dwin	0.111
30	34	ct_src_dport_ltm	0.061
31	36	ct_dst_src_ltm	0.055
32	3	service	0.051
33	30	response_body_len	0.038
34	41	ct_srv_dst	0.035
35	31	ct_srv_src	0.031
36	33	ct_dst_ltm	0.026
37	40	ct_src_ltm	0.021
38	42	is_sm_ips_ports	0.012
39	38	ct_ftp_cmd	0.003
40	37	is_ftp_login	0.003
41	39	ct_flw_http_mthd	0.001
42	29	trans_depth	0.000

Appendix F

HMM Parameters for Experiment B

HMM S

```
estTRC =

0.9759        0       0.0241
0             0       0
0.1100        0       0.8900


estEC =

  Columns 1 through 9

   0.0511   0.0011   0.0000 0   0.7046   0   0.0001        0.0387   0.0006
   0             0        0        0        0        0        0         0          0
   0.0059        0        0        0        0        0        0         0.1024   0

  Columns 10 through 13

   0.0032   0.0000   0.2005   0.0000
       0        0        0        0
       0        0        0.8909   0.0008
```

After Baum-Welch Algorithm

```
estTRC1 =

   0.9999        0   0.0001
       0   1.0000        0
   0.0001        0   0.9999


estEC1 =

  Columns 1 through 9

   0.0035   0.0016   0.0000   0.0001   0.9663   0.0001   0.0001   0.0000   0.0008
   0        0        0        0        0        0        0        0        0
   0.1017    0         0         0         0         0         0         0.1248    0

  Columns 10 through 13

   0.0044    0.0000    0.0231    0.0000
   0         0         0         0
   0         0         0.7732    0.0004
```

HMM C

```
estTRC =

   0.9759         0    0.0241
   0              0    0
   0.1100         0    0.8900


estEC =

  Columns 1 through 9

   0.9201    0.0721    0.0035    0.0008    0.0003    0         0         0         0
   0         0         0         0         0         0         0         0         0
   0.4603    0.2644    0.1651    0.0646    0.0192    0.0034    0.0040    0.0098    0.0044

  Columns 10 through 16

   0          0.0003    0.0004    0.0004    0.0007    0.0007    0.0008
   0          0         0         0         0         0         0
   0.0049     0         0         0         0         0         0
```

After Baum-Welch Algorithm:

```
estTRC1 =

   0.9924    0         0.0076
   0         1.0000    0
   0.0281    0         0.9719


estEC1 =

  Columns 1 through 9
```

```
   0.9712    0.0241    0.0010    0.0003    0.0002    0         0         0         0
   0         0         0         0         0         0         0         0         0
   0.3421    0.4126    0.1495    0.0568    0.0167    0.0029    0.0034    0.0083    0.0037

  Columns 10 through 16

   0         0.0003    0.0004    0.0004    0.0007    0.0007    0.0009
   0         0         0         0         0         0         0
   0.0041    0         0         0         0         0         0
```

HMM D

```
estTRC =

   0.9759         0    0.0241
   0              0    0
   0.1100         0    0.8900


estEC =

   0.1205    0.7046    0    0.0000         0    0.0013    0.1736         0    0.0000
   0         0         0    0              0    0          0             0    0
   0.7951    0         0    0              0    0.0181    0.1868         0    0
```

After Baum-Welch Algorithm

```
estTRC1 =

   0.9998         0    0.0002
      0    1.0000         0
   0.0002         0    0.9998


estEC1 =

   0.0180    0.9668    0.0001    0.0001    0.0001    0.0017    0.0133    0.0001    0.0001
   0         0         0         0         0         0         0         0         0
   0.5741    0         0         0         0         0.0081    0.4178    0         0
```

Reference List

Akusok, A., Miche, Y., Hegedus, J., Nian, R., & Lendasse, A. (2014). A two-stage methodology using K-NN and false-positive minimizing ELM for nominal data classification. *Cognitive Computation, 6*(3), 432-445.

Alhaidari, S. & Zohdy, M. (2018). Feature pruning method for hidden markov model-based anomaly detection: a comparison of performance. *Jordanian Journal of Computers and Information Technology (JJCIT)*, *4*(03).

Alsmadi, I. & Xu, D. (2015). Security of Software Defined Networks: A survey. *Computers & Security*, 53, 79-108.

Anderson, J.P. (1980). Computer Security Threat Monitoring & Surveillance. Technical Report, James P. Anderson Co., Fort Washington, Pennsylvania.

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

Azad, C. & Jha, V. K. (2013). Data mining in intrusion detection: a comparative study of methods, types and data sets. *International Journal of Information Technology and Computer Science (IJITCS), 5*(8), 75.

Bahrololum, M., Salahi, E., & Khaleghi, M. (2009). Machine learning techniques for feature reduction in intrusion detection systems: a comparison. *IEEE Fourth International Conference on Computer Sciences and Convergence Information Technology (ICCIT),* 1091-1095.

Barbara, D., Couto, J., Jajodia, S., Popyack, L., & Wu, N. (2001). ADAM: Detecting intrusions by data mining. *Proceedings of the IEEE Workshop on Information Assurance and Security*.

Bamakan, S. M. H., Wang, H., & Shi. Y. (2017). Ramp loss K-Support Vector Classification-Regression; a robust and sparse multi-class approach to the intrusion detection problem. *Knowledge-Based Systems*, 126, 113–126.

Barlow, C. (2017, March 17). Artificial intelligence makes cybersecurity the ideal field for 'new collar' jobs. *The Hill.* Retrieved from thehill.com

Bass, T. (1999). Multisensor data fusion for next generation distributed intrusion detection systems. *Proceeding of the IRIS national symposium on sensor and data fusion*, 99-105.

Baum, L. E. & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, *37*(6), 1554-1563.

Benmessahel, I., Xie, K., Chellal, M., & Semong, T. (2019). A new evolutionary neural networks based on intrusion detection systems using locust swarm optimization. *Evolutionary Intelligence*, *12*(2), 131-146.

Bereziński, P., Jasiul, B., & Szpyrka, M. (2015). An entropy-based network anomaly detection method. *Entropy, 17*(4), 2367-2408.

Bhatt, S., Manadhata, P. K., & Zomlot, L. (2014). The operational role of security information and event management systems. *IEEE Security & Privacy, 12*(5), 35-41.

Bhatt, P., Yano, E. T., & Gustavsson, P. (2014). Towards a framework to detect multi-stage advanced persistent threats attacks. Paper presented at the *Service Oriented System Engineering (SOSE), 2014 IEEE 8th International Symposium on*.

Bhuyan, M.H., Bhattacharyya, D.K., & Kalita, J.K. (2014). Network anomaly detection: Methods, systems, and tools. *IEEE Communication Surveys & Tutorials*, *16*(1), 303-336.

Blanco, R., Cilla, J. J., Malagón, P., Penas, I., & Moya, J. M. (2018, June). Tuning CNN Input Layout for IDS with Genetic Algorithms. In *International Conference on Hybrid Artificial Intelligence Systems* (pp. 197-209). Springer, Cham.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory*, 144-152.

Brown, C., Cowperthwaite, A., Hijazi, A., & Somayaji, A. (2009). Analysis of the 1999 DARPA/Lincoln Laboratory IDS evaluation data with NetADHICT, *IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 1-10.

Cannady, J. (1998). Artificial neural networks for misuse detection. Paper presented at the National Information Systems Security conference.

Castaño, A., Fernández-Navarro, F., & Hervás-Martínez, C. (2013). PCA-ELM: A robust and pruned extreme learning machine approach based on principal component analysis. *Neural Processing Letters, 37*(3), 377-392.

Cerqueira, V., Torgo, L., & Mozetic, I. (2019). Evaluating time series forecasting models: An empirical study on performance estimation methods. *arXiv preprint arXiv:1905.11744*.

Chen, C. M., Guan, D. J., Huang, Y. Z., & Ou, Y. H. (2016). Anomaly network intrusion detection using Hidden Markov model. *Int. J. Innov. Comput. Inform. Control*, 12, 569-580.

Cheng, C., Tay, W. P., & Huang, G.-B. (2012). Extreme learning machines for intrusion detection. Paper presented at the Neural Networks (IJCNN), The 2012 International Joint Conference on.

Cisco (2016, June 14). Cisco launches $10 million global cybersecurity scholarship to increase talent pool; Introduces New and Updated Certifications. Retrieved from https://newsroom.cisco.com/pressreleasecontent?type=webcontent&articleId=1772385

Coull, S. E., Wright, C. V., Monrose, F., Collins, M. P., & Reiter, M. K. (2007). Playing devil's advocate: Inferring sensitive information from anonymized network traces. *Network and Distributed System Security (NDSS) Symposium of the Internet Society*, *7*, 35-47.

Creech, G. & Jiankun, H. (2014). A semantic approach to Host-Based Intrusion Detection Systems using contiguous and discontiguous system call patterns. *Computers, IEEE Transactions on, 63*(4), 807-819.

Crosbie, M. & Spafford, E. H. (1995). Defending a computer system using autonomous agents. Retrieved from https://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=2199&context=cstech

Dainotti, A., Pescape, A., & Claffy, K. C. (2012). Issues and future directions in traffic classification. *Network, IEEE, 26*(1), 35-40.

Davis, J. J. & Clark, A. J. (2011). Data preprocessing for anomaly based network intrusion detection: A review. *Computers & Security*, *30*(6-7), 353-375.

Deng, W., Zheng, Q., & Zhang, K. (2013). Reduced kernel extreme learning machine. *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013,* 63-69.

Denning, D. E. & Neumann, P. G. (1985). Requirements and model for IDES—a real-time intrusion detection expert system. *Document A005, SRI International,* 1-70.

Denning, D.E. (1987). An intrusion detection model. *IEEE Transactions on Software Engineering – Special Issue on Computer Security and Privacy*, *13*(2), 222-232.

Depren, O., Topallar, M., Anarim, E., & Ciliz, M. K. (2005). An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert Systems with Applications, 29*(4), 713-722.

Ding, S., Xu, X., & Nie, R. (2014). Extreme learning machine and its applications. *Neural Computing and Applications, 25*(3-4), 549-556.

Endsley, M. R. (1988). Design and evaluation for situation awareness enhancement. Paper presented at the Proceedings of the Human Factors and Ergonomics Society Annual Meeting.

Fawcett, T. (2006). An Introduction to ROC Analysis. *Pattern Recognition Letters*, *27*(8), 861-874.

Fernandes, G., Rodrigues, J. J., Carvalho, L. F., Al-Muhtadi, J. F., & Proença, M. L. (2019). A comprehensive survey on network anomaly detection. *Telecommunication Systems*, *70*(3), 447-489.

Forney, G. D. (1973). The Viterbi Algorithm. *Proceedings of the IEEE*, *61*(3), 268-278.

Fossaceca, J. M., Mazzuchi, T. A., & Sarkani, S. (2015). MARK-ELM: Application of a novel multiple kernel learning framework for improving the robustness of network intrusion detection. *Expert Systems with Applications, 42*(8), 4062-4080.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, *28*(2), 337-407.

Freund, Y. & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, *55*(1), 119-139.

Gilmore, C. & Haydaman, J. (2016). Anomaly detection and machine learning methods for network intrusion detection: An industrially focused literature review. *Proceedings of the International Conference on Security and Management (SAM)*.

Graff, G. M. (2017, December 13). How a dorm room Minecraft scam brought down the Internet. *Wired*. Retrieved from wired.com

Hall, M.A. (2000). Correlation-based feature selection of discrete and numeric class machine learning (Working paper 00/08). Hamilton, New Zealand: University of Waikato, Department of Computer Science.

Ho, C. Y., Lai, Y. C., Chen, I. W., Wang, F. Y., & Tai, W. H. (2012). Statistical analysis of false positives and false negatives from real traffic with intrusion detection/prevention systems. *IEEE Communications Magazine*, *50*(3), 146-154.

Holgado, P., Villagra, V. A., & Vazquez, L. (2017). Real-time multistep attack prediction based on Hidden Markov Models. *IEEE Transactions on Dependable and Secure Computing*.

Hoque, M. S., Mukit, M. A., & Bikas, M. A. N. (2012). An implementation of intrusion detection system using genetic algorithm. *International Journal of Network Security & Its Applications, 4*(2), 109.

Horne, B. (2015). Umbrellas and octopuses. *IEEE Security & Privacy*, *13*(1), 3-5.

Hu, J., Yu, X., Qiu, D., & Chen, H. H. (2009). A simple and efficient Hidden Markov Model scheme for host-based anomaly intrusion detection. *IEEE Network*, *23*(1), 42-47.

Hu, W., Gao, J., Wang, Y., Wu, O., & Maybank, S. (2014). Online AdaBoost-based parameterized methods for dynamic distributed network intrusion detection. *IEEE Transactions on Cybernetics*, *44*(1), 66-82.

Huang, G. B. & Chen, L. (2008). Enhanced random search based incremental extreme learning machine. *Neurocomputing*, *71*(16-18), 3460-3468.

Huang, G. B., Chen, L., & Siew, C. K. (2006). Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Trans. Neural Networks*, *17*(4), 879-892.

Huang, G.-B., Wang, D. H., & Lan, Y. (2011). Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, *2*(2), 107-122.

Huang, G.-B., Zhou, H., Ding, X., & Zhang, R. (2012). Extreme learning machine for regression and multiclass classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 42*(2), 513-529.

Huang, G.-B. & Zhu, Q.-Y. (2004). Basic ELM Algorithm, MATLAB version

[Software]. Available from http://www.ntu.edu.sg/home/egbhuang/elm_codes.html

Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks. *Proceedings of the IEEE International Joint Conference on Neural Networks*.

Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2006). Extreme learning machine: theory and applications. Neurocomputing, 70(1-3), 489-501.

Huang, Z., Shen, C.-C., Doshi, S., Thomas, N., & Duong, H. (2015). Cognitive task analysis based training for cyber situation awareness, *Information Security Education Across the Curriculum,* 27-40, Springer.

Hubballi, N. & Suryanarayanan, V. (2014). False alarm minimization techniques in signature-based intrusion detection systems: A survey. *Computer Communications*, *49*(0), 1-17.

Hurley, T., Perdomo, J. E., & Perez-Pons, A. (2016). HMM-based intrusion detection system for Software Defined Networking. In *Machine Learning and Applications (ICMLA), 2016 15th IEEE International Conference on,* 617-621. IEEE.

Hutchins, E. M., Cloppert, M. J., & Amin, R. M. (2011). Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research, 1*, 80.

Igbe, O. (2019). *Artificial immune system based approach to cyber attack detection* (Doctoral dissertation). Available from ProQuest Dissertations & Theses Global. (2177283169). Retrieved from http://search.proquest.com.ezproxylocal.library.nova.edu/docview/2177283169?accountid=6579

Inayat, Z., Gani, A., Anuar, N. B., Khan, M. K., & Anwar, S. (2016). Intrusion response systems: Foundations, design, and challenges. *Journal of Network and Computer Applications, 62*, 53-74.

Information System Security Certification Consortium (2018, August). Cybersecurity Professionals Focus on Developing New Skills as Workforce Gap Widens, (ISC)2 Cybersecurity Workforce Study, 2018 [PDF document]. Retrieved from https://www.isc2.org/-/media/7CC1598DE430469195F81017658B15D0.ashx

Janarthanan, T. & Zargari, S. (2017). Feature selection in UNSW-NB15 and KDDCUP'99 datasets. In *Industrial Electronics (ISIE), 2017 IEEE 26th International Symposium on,* 1881-1886. IEEE.

Julisch, K. & Dacier, M. (2002). Mining intrusion detection alarms for actionable knowledge*. Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining.*

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

Karthick, R. R., Hattiwale, V. P. & Ravindran, B. (2012). Adaptive network intrusion detection system using a hybrid approach. Paper presented at the 2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012).

Kamarudin, M. H., Maple, C., Watson, T., & Safa, N. S. (2017). A logitboost-based algorithm for detecting known and unknown web attacks. *IEEE Access*, *5*, 26190-26200.

Karegowda, A. G., Manjunath, A. S., & Jayaram, M. A. (2010). Comparative study of attribute selection using gain ratio and correlation based feature selection. *International Journal of Information Technology and Knowledge Management*, *2*(2), 271-277.

Kasun, L. L. C., Zhou, H., Huang, G. B., & Vong, C. M. (2013). Representational learning with ELMs for Big Data. *IEEE Intelligent Systems*, *28*(6), 31-34.

Kayacik, H. G., Zincir-Heywood, A. N., & Heywood, M. I. (2005, October). Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets. In *Proceedings of the third annual conference on* *privacy, security and trust*.

Khammassi, C. & Krichen, S. (2017). A GA-LR wrapper approach for feature selection in network intrusion detection. *Computers & Security*, *70*, 255-277.

Koc, L., Mazzuchi, T. A., & Sarkani, S. (2012). A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier. *Expert Systems with Applications, 39*(18), 13492-13500.

Kohonen, T. & Somervuo, P. (1998). Self-Organizing Maps of symbol strings. *Neurocomputing*, *21*(1), 19-30.

Li, Y., Xia, J., Zhang, S., Yan, J., Ai, X., & Dai, K. (2012). An efficient intrusion detection system based on support vector machines and gradually feature removal method. *Expert Systems with Applications, 39*(1), 424-430.

Liang, W., Chen, Z., Yan, X., Zheng, X., & Zhuo, P. (2017). Multiscale entropy-based weighted Hidden Markov network security situation prediction model. In *Internet of Things (ICIOT), 2017 IEEE International Congress on,* 97-104. IEEE.

Liang, Y., Wang, H. Q., Cai, H. B., & He, Y. J. (2008). A novel stochastic modeling method for network security situational awareness. In *Industrial Electronics and Applications, 2008. ICIEA 2008. 3rd IEEE Conference on,* 2422-2426. IEEE.

Lippmann, R. P., Fried, D. J., Graf, I., Haines, J. W., Kendall, K. R., McClung, D., ... & Zissman, M. A. (2000, January). Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00* (Vol. 2, pp. 12-26). IEEE.

Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., & Das, K. (2000). The 1999 DARPA off-line intrusion detection evaluation. *Computer networks*, *34*(4), 579-595.

Lin, Y. D., Lai, Y. C., Ho, C. Y., & Tai, W. H. (2013). Creditability-based weighted voting for reducing false positives and negatives in intrusion detection. *Computers & security*, *39*, 460-474.

Lin, X., Yacoub, S., Burns, J., & Simske, S. (2003). Performance analysis of pattern classifier combination by plurality voting. *Pattern Recognition Letters*, *24*(12), 1959-1969.

Liu, H. & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering, 17*(4), 491-502.

Luh, R., Marschalek, S., Kaiser, M., Janicke, H., & Schrittwieser, S. (2016). Semantics-aware detection of targeted attacks: a survey. *Journal of Computer Virology and Hacking Techniques*, 1-39.

Marchetti, M., Pierazzi, F., Colajanni, M., & Guido, A. (2016). Analysis of high volumes of network traffic for Advanced Persistent Threat detection. *Computer Networks*.

McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115-133.

McHugh, J. (2000). Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and System Security (TISSEC)*, *3*(4), 262-294.

Mees, W., & Debatty, T. (2015). An attempt at defining cyberdefense situation awareness in the context of command & control. *Military Communications and Information Systems (ICMCIS), 2015 International Conference on*.

Milenkoski, A., Vieira, M., Kounev, S., Avritzer, A., & Payne, B. D. (2015). Evaluating computer intrusion detection systems: A survey of common practices. *ACM Computing Surveys (CSUR)*, *48*(1), 12.

Mitchell, R. & Chen, I. (2014). A survey of intrusion detection techniques for Cyber

Physical Systems. *ACM Computing Surveys (CSUR)*, *46*(4), article 55.

Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., & Rajarajan, M. (2013). A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications, 36*(1), 42-57.

Moustafa, N. (2017). *Designing an online and reliable statistical anomaly detection framework for dealing with large high-speed network traffic* (Doctoral dissertation, University of New South Wales, Canberra, Australia). Retrieved from: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

Moustafa, N., Adi, E., Turnbull, B., & Hu, J. (2018). A new threat intelligence scheme for safeguarding industry 4.0 systems. *IEEE Access*, *6*, 32910-32924.

Moustafa, N., Creech, G., & Slay, J. (2017). Big data analytics for intrusion detection system: Statistical decision-making using finite dirichlet mixture models. In *Data analytics and decision support for cybersecurity*, 127-156. Springer, Cham.

Moustafa, N. & Slay, J. (2015). UNSW-NB15: A Comprehensive data set for Network Intrusion Detection Systems (UNSW-NB15 network data set). In *Military Communications and Information Systems Conference (MilCIS), 2015,* 1-6, IEEE.

Moustafa, N. & Slay, J. (2016). The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*, *25*(1-3), 18-31.

Moustafa, N. & Slay, J. (2017). A hybrid feature selection for network intrusion detection systems: Central points. *arXiv preprint arXiv:1707.05505.*

Moustafa, N., Slay, J., & Creech, G. (2017). Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks. *IEEE Transactions on Big Data*.

Moustafa, N., Turnbull, B., & Choo, K. K. R. (2018). An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. *IEEE Internet of Things Journal*.

Mukkamala, S., Sung, A. H. & Abraham, A. (2005). Intrusion detection using an ensemble of intelligent paradigms. *Journal of Network and Computer Applications*, *28*(2), 167-182.

Muna, A. H., Moustafa, N., & Sitnikova, E. (2018). Identification of malicious activities in industrial internet of things based on deep learning models. *Journal of Information Security and Applications*, *41*, 1-11.

Ourston, D., Matzner, S., Stump, W., & Hopkins, B. (2003). Applications of Hidden Markov Models to detecting multi-stage network attacks. Paper presented at the System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on.

Pao, H. K., Lee, Y. J., & Huang, C. Y. (2015). Statistical learning methods for information security: fundamentals and case studies. *Applied Stochastic Models in Business and Industry, 31*(2), 97-113.

Paxson, V. (1999). Bro: a system for detecting network intruders in real-time. *Computer Networks, 31*(23), 2435-2463.

Perdisci, R., Ariu, D., Fogla, P., Giacinto, G., & Lee, W. (2009). McPAD: A multiple classifier system for accurate payload-based anomaly detection. *Computer Networks*, *53*(6), 864-881.

Ponemon, L. & Trunkey, A. (2016). Key findings from the 2016 cost of data breach study: Global analysis [PowerPoint slides]. Retrieved from https://www.slideshare.net/ibmsecurity/the-2016-ponemon-cost-of-a-data-breach-study

Porras, P. A. & Neumann, P. G. (1997). EMERALD: Event monitoring enabling response to anomalous live disturbances. Paper presented at the Proceedings of the 20th National Information Systems Security conference.

Rabiner, L. R. (1989). A Tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257-286.

Rai, D. & Tyagi, K. (2013). Bio-inspired optimization techniques: a critical comparative study. *ACM SIGSOFT Software Engineering Notes*, *38*(4), 1-7.

Rhodes, B. C., Mahaffey, J. A., & Cannady, J. D. (2000). Multiple self-organizing maps for intrusion detection. *Proceedings of the 23rd National Information Systems Security Conference*.

Roesch, M. (1999). Snort: Lightweight Intrusion Detection for Networks. Paper presented at LISA conference.

Russell, S. J. & Norvig, P. (2003). *Artificial intelligence: A modern approach*. Upper Saddle River, N.J: Prentice Hall/Pearson Education.

Saeys, Y., Inza, I., & Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics*, *23*(19), 2507-2517.

Sendi, A. S., Dagenais, M., Jabbarifar, M., & Couture, M. (2012). Real time intrusion prediction based on optimized alerts with Hidden Markov Model. *Journal of Networks (JNW)*, *7*(2), 311-321.

Shah, A. A., Khiyal, M. S. H., & Awan, M. D. (2015). Analysis of machine learning techniques for Intrusion Detection System: A Review. *International Journal of Computer Applications, 119*(3).

Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, *31*(3), 357-374.

Shittu, R., Healing, A., Ghanea-Hercock, R., Bloomfield, R., & Rajarajan, M. (2015). Intrusion alert prioritisation and attack detection using post-correlation analysis. *Computers & Security, 50*(0), 1-15.

Silva, S. S., Silva, R. M., Pinto, R. C., & Salles, R. M. (2013). Botnets: A survey.

Computer Networks, 57(2), 378-403.

Sindhu, S. S. S., Geetha, S., & Kannan, A. (2012). Decision tree based light weight intrusion detection using a wrapper approach. *Expert Systems with Applications, 39*(1), 129-141.

Singh, R., Kumar, H. & Singla, R. K. (2015). An intrusion detection system using network traffic profiling and online sequential extreme learning machine. *Expert Systems with Applications*, *42*(22), 8609-8624.

Sommer, R. & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. *2010 IEEE Symposium on Security and Privacy (SP)*, 305-316.

Song, J., Takakura, H., Okabe, Y., Eto, M., Inoue, D., & Nakao, K. (2011). Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation. *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*.

Spathoulas, G. P. & Katsikas, S. K. (2010). Reducing false positives in intrusion detection systems. *Computers & Security*, *29*(1), 35-44

Spathoulas, G. & Katsikas, S. (2013a). Enhancing IDS performance through comprehensive alert post-processing. *Computers & Security, 37*(0), 176-196.

Spathoulas, G. & Katsikas, S. (2013b). Methods for post-processing of alerts in intrusion detection: A survey. *International Journal of Information Security Science, 2*(2), 64-80.

Starr, M. (2014, January 19). Fridge caught sending spam emails in botnet attack. Retrieved from https://www.cnet.com/news/fridge-caught-sending-spam-emails-in-botnet-attack/

Sundaramurthy, S. C., Bhatt, S., & Eisenbarth, M. R. (2012). Examining intrusion prevention system events from worldwide networks. Paper presented at the *Proceedings of the 2012 ACM Workshop on Building Analysis Datasets and Gathering Experience Returns for Security,* 5-12.

Swarnkar, M. & Hubballi, N. (2016). OCPAD: One class Naive Bayes classifier for payload based anomaly detection. *Expert Systems with Applications*, *64*, 330-339.

Tama, B. A. & Rhee, K. H. (2019). An in-depth experimental study of anomaly detection using gradient boosted machine. *Neural Computing and Applications*, *31*(4), 955-965.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defense Applications,* 1-6.

Tavallaee, M., Stakhanova, N., & Ghorbani, A.A. (2010). Toward credible evaluation of anomaly-based intrusion-detection methods. *IEEE Transactions on Systems, Man, and Cybernetics*, *40*(5), 516-524.

Tchakoucht, T. A. & Ezziyyani, M. (2018). Multilayered Echo-State Machine: A novel architecture for efficient intrusion detection. *IEEE Access*, *6*, 72458-72468.

Theodoridis, S., Pikrakis, A., Koutroumbas, K., & Cavouras, D. (2010). *Introduction to Pattern Recognition: A MATLAB Approach*. Academic Press.

Treinen, J. J. & Thurimella, R. (2009). Finding the needle: Suppression of false alarms in large intrusion detection data sets. Paper presented at the *Computational Science and Engineering 2009 (CSE'09), International Conference on*.

Tsai, C. F., Hsu, Y. F., Lin, C. Y., & Lin, W. Y. (2009). Intrusion detection by machine learning: A review. *Expert Systems with Applications*, *36*(10), 11994-12000.

University of California Irvine. (1999). KDD Cup 1999 Data [Data file]. Retrieved from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99

University of New South Wales. (2015). UNSW-NB15 Data Set [Data files]. Retrieved from https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

Valero León, A. (2017). INsIDES: A new machine learning-based intrusion detection system (Bachelor thesis). Retrieved from: http://hdl.handle.net/10230/32875

Verizon. (2016). 2016 data breach investigations report, Verizon Enterprise [Webpage]. Retrieved from: http://www.verizonenterprise.com/dbir2016

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access*, *7*, 41525-41550.

Viterbi, A. (1967). Error bounds for Convolutional Codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, *13*(2), 260-269.

Wang, G., Hao, J., Ma, J., & Huang, L. (2010). A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering. *Expert Systems with Applications, 37*(9), 6225-6232.

Wang, D. & Huang, G. B. (2005, July). Protein sequence classification using extreme learning machine. *In Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* (Vol. 3, pp. 1406-1411). IEEE.

Wang, J. & Lee, I. (2001). Measuring false-positive by automated real-time correlated hacking behavior analysis, *Information Security,* 512-535.

Wang, H., Liu, X., Lai, J., & Liang, Y. (2007). Network Security Situation awareness based on heterogeneous multi-sensor data fusion and neural network. *Second International Multi-Symposiums on Computer and Computational Sciences*, 352-359.

Wang, C. R., Xu, R. F., Lee, S. J., & Lee, C. H. (2018). Network intrusion detection using equality constrained-optimization-based extreme learning machines. *Knowledge-Based Systems*, *147*, 68-80.

Warrender, C., Forrest, S., & Pearlmutter, B. (1999). Detecting intrusions using system calls: Alternative data models. *Security and Privacy, Proceedings of the 1999 IEEE Symposium on,* 133-144. IEEE.

Weller-Fahy, D. J., Borghetti, B. J., & Sodemann, A. (2015). A survey of distance and similarity measures used within network intrusion anomaly detection. *Communications Surveys & Tutorials, IEEE, 17*(1), 70-91.

Woods, K., Kegelmeyer, W. P., & Bowyer, K. (1997). Combination of multiple classifiers using local accuracy estimates. *IEEE transactions on pattern analysis and machine intelligence*, *19*(4), 405-410.

Wright, C., Monrose, F., & Masson, G. M. (2004, October). HMM profiles for network traffic classification. *Proceedings of the 2004 ACM workshop on Visualization and Data Mining for Computer Security,* 9-15. ACM.

Wu, S. X. & Banzhaf, W. (2010). The use of computational intelligence in Intrusion Detection Systems: A review. *Applied Soft Computing*, *10*(1), 1-35.

Xia, H. & Hoi, S. C. (2013). MKBoost: A framework of Multiple Kernel Boosting. *Knowledge and Data Engineering, IEEE Transactions on*, *25*(7), 1574-1586.

Yost, J.R. (2015). The March of IDES: Early History of Intrusion-Detection Expert Systems. *IEEE Annals of the History of Computing, 38*(4), 42-54.

Yu, W., Zhuang, F., He, Q., & Shi, Z. (2015). Learning deep representations via Extreme Learning Machines. *Neurocomputing,* 149, Part A, 308-315.

Zhou, C., Huang, S., Xiong, N., Yang, S. H., Li, H., Qin, Y., & Li, X. (2015). Design and analysis of multimodel-based anomaly Intrusion Detection Systems in industrial process automation. *IEEE Transactions on Systems, Man, and Cybernetics, 45*(10), 1345-1360.

Zong, W., Huang, G.-B., & Chen, Y. (2013). Weighted Extreme Learning Machine for imbalance learning. *Neurocomputing, 101*, 229-242.

Zuech, R., Khoshgoftaar, T. M., & Wald, R. (2015). Intrusion detection and big heterogeneous data: A Survey. *Journal of Big Data, 2*(1), 1-41.