2017

Enhanced Prediction of Network Attacks Using Incomplete Data

Jacob D. Arthur
Nova Southeastern University, ja1333@mynsu.nova.edu

Enhanced Prediction of Network Attacks Using Incomplete Data

by

Jacob D. Arthur

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

Graduate School of Computer and Information Sciences
Nova Southeastern University

July 19, 2017

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

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

09/28/2017
Date

Wei Li

9/28/2017
Date

Wei Li, Ph.D.
Dissertation Committee Member

Glyn Thomas Gowing, PhD

9/28/2017
Date

Glyn Thomas Gowing, Ph.D.
Dissertation Committee Member

Approved:

9/28/2017
Date

Yong X. Tao, Ph.D., P.E., FASME
Dean, College of Engineering and Computing

College of Engineering and Computing
Nova Southeastern University

2017

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

Enhanced Prediction of Network Attacks Using Incomplete Data

By
Jacob D. Arthur
July 19, 2017

For years, intrusion detection has been considered a key component of many organizations' network defense capabilities. Although a number of approaches to intrusion detection have been tried, few have been capable of providing security personnel responsible for the protection of a network with sufficient information to make adjustments and respond to attacks in real-time. Because intrusion detection systems rarely have complete information, false negatives and false positives are extremely common, and thus valuable resources are wasted responding to irrelevant events. In order to provide better actionable information for security personnel, a mechanism for quantifying the confidence level in predictions is needed. This work presents an approach which seeks to combine a primary prediction model with a novel secondary confidence level model which provides a measurement of the confidence in a given attack prediction being made. The ability to accurately identify an attack and quantify the confidence level in the prediction could serve as the basis for a new generation of intrusion detection devices, devices that provide earlier and better alerts for administrators and allow more proactive response to events as they are occurring.

Acknowledgements

As I reflect on the effort that has gone into this work, it is clear to me that this effort was not completed on accident or without substantial sacrifice from those around me. To my beautiful wife Katie, who supported my decision to pursue this effort and picked me up the many times I became frustrated, I could not have done this without your encouragement and patience. To my two favorite little girls, thank you for invading my office frequently and reminding me of what is really important in life.

I would like to thank Dr. Cannady for his guidance, assistance, and patience as I worked to navigate the dissertation process. Also, thanks to my committee members, Dr. Li and Dr. Gowing, both of you provided excellent suggestions that helped clarify my direction and ensure that I had articulated both my goals and a pathway to achieve them.

I would also like to thank my numerous colleagues at Lipscomb University for supporting my efforts to pursue this degree and for providing encouragement for me on the road. Also, to my war-room compadres at Formos Consulting, thank you for letting me monopolize the whiteboards on far too many occasions and for letting me talk through ideas though I know you had very little interest in them.

Finally, thanks to my extended family and friends for your kind words of encouragement, and for helping pick up the slack in caring for my wife and kids when coursework or library time took me far from home. It does indeed take a village, and we have a great one.

And finally, to my Lord, Jesus Christ, through whom all things are possible.

Table of Contents

List of Tables

Tables

List of Figures

Figures

Chapter 1

Introduction

Introduction

Interconnection between networks around the globe has increased exponentially in volume and variance. A parallel increase has occurred in the number and nature of potential threats against those networks (S. H. Kim, Wang, & Ullrich, 2012). Modern network attacks typically involve a series of discrete stages focused on collecting data, identifying vulnerabilities, and executing exploits (Katipally, Yang, & Liu, 2011). For example, in order to breach a database system, an attacker must discover its existence and address, determine available access methods, and select a method (or methods) to gain access to the system. In the event the selected approach is ineffective in compromising the system, the attacker will either select a different approach or select a new target (Alserhani, Akhlaq, Awan, Cullen, & Mirchandani, 2010).

Though the phases of network intrusion (explore, expose, and exploit) are logical, they are neither transparent nor simplistic (Katipally, Gasior, Cui, & Yang, 2010). Modern intrusion detection systems (IDSs) lack full visibility into all of the activities that comprise an attack. Partial visibility of relevant attack data leads to both false negatives and false positives in the alert streams generated by modern IDSs (Soleimani & Ghorbani, 2012). The perceived value of such alerts is reduced, as administrators respond to events ultimately determined to be irrelevant while failing to respond in a timely

manner to those which potentially pose widespread threat of harm (Sharma & Mukherjee, 2012).

Solutions to limits in IDSs have ranged from increasing the number and coordination among sensors (Abdullah, Xu, & Geva, 2011; Zargar, Takabi, & Joshi, 2011), to distributed individual agent processing (Carvalho & Perez, 2011). Both approaches present challenges. In the case of the former, each sensor contributes an additive resource cost as well as an exponential increase to the search space that must be processed by control nodes. The latter approach results in each sensor making decisions with a lack of overall visibility. Although machine learning techniques have been shown to provide some good results in terms of attack prediction within a network (D. Ariu, R. Tronci, & G. Giacinto, 2011; Fries, 2008), the occurrence of false positives and false negatives in those applications remains high. Ensuring organizational resources are invested in preventing genuine attacks requires improvement in the ratio of valid to invalid alerts. In order to improve that ratio, this work presents experiments on the results of augmenting predictions made by an IDS with actionable confidence levels.

Problem Statement

Modern network-based intrusion detection systems (IDSs) often fail to accurately detect attacks for which only fragments of the attack are visible to the sensors in place. As a result, true attacks occur without raising an alert and normal activity might precipitate an alert that wastes the resources of analysts.

Dissertation Goal

The goal of the proposed research was to develop an improved means of identifying an ongoing attack based on small fragments of attack activity with a lower rate of false positives and false negatives than has been previously achieved by utilizing existing methods. The improvement in prediction rate was primarily captured via the augmentation of the system output with a confidence level based on prior network traffic and alerts that have been raised. In order to deal with the common issue of false positives and false negatives, the primary focus of this research was on the development of an approach for providing both a meaningful confidence level in predictions that the system makes.

Relevance and Significance

Interconnection between networks around the globe has increased, becoming an integral part of daily functioning within the arenas of business, education, governmental operations, and economics. Accompanying the increase in network activity and connection is a parallel increase in security risk (S. H. Kim et al., 2012). The importance of networks and the potential impacts of their compromise are most often evident to the "general public" in the fields of retail or education, as consumers learn through public media of a breach which may threaten their personal finances, credit, or identity. Even more critical, the growing prevalence of network connectivity in fields such as flight control and power grid management contributes to growing urgency in the field. As sensitive information and critical infrastructure increasingly rely on computer systems and those systems become more complex, the intrusion prevention technologies must also

become more sophisticated (Mitchell & Chen, 2014; Shameli-Sendi, Cheriet, & Hamou-Lhadj, 2014).

Because of the great risks associated with both internal and external threats, organizations, both public and private, have deployed various technologies to both limit the attack surface area as well as to gain insight into the potentially massive amounts of data traversing the network. Denning (1987) conceived of one such technology, the intrusion detection system, to give administrators specific intelligence about events happening on the network in real-time or near real-time. Although a vast array of information would serve as inputs, the goal was to distill the information down into a set of alerts that could be outputted in the event that patterns of concern were observed in the network stream. These intrusion detection systems have served well for many years and have provided significant insight into what was occurring moment by moment in networks they observed (Weir, 2012). However, the significant increase in network traffic as bandwidth has increased has resulted in intrusion detection systems that output either too much information to be useful (alert flooding) or systems which filter out so much information that they fail to produce alerts when events in the network represent meaningful threats (S. H. Kim et al., 2012).

As the number of alerts produced by intrusion detection systems have increased, the ability of system and security administrators to respond to each individual alert has decreased (Zargar et al., 2011). A common method of limiting the volume of alerts is to raise an alert only after a specific number of login failures have been detected over a rolling period of time. However, this approach has its own deficiency, in that an attacker who is able to determine the time period through experimentation or insider knowledge

may be able to remain under the alert threshold constantly. Clearly, better technology is needed; this need led to the introduction of machine learning and other more advanced methods of intrusion detection.

Barriers and Issues

The first barrier to this research is related to feature selection (Zaman, El-Abed, & Karray, 2013). Though much work has been done on feature selection in intrusion detection, this research began with a very unclear picture on the features or trends that would be relevant for assigning a confidence level to a prediction being made. This was further complicated by the fact that confidence levels have always been considered *inherently*, that is, a system would not predict that a packet represented an attack if it was not more confident than not. The challenge for this research was to apply existing knowledge to draw out the confidence level explicitly. Although some of the classic features will certainly be applicable, one of the most significant factors in determining confidence level is based on how similar the current traffic is to traffic that has been seen previously. If very similar traffic was previously classified correctly, then this research concluded it would be reasonable to place more confidence in the prediction than if similar traffic had not been seen or had been classified incorrectly.

Another significant barrier to the completion of this study was access to an appropriate data set for development and testing. In research related to intrusion detection, the Defense Advanced Research Projects Agency (DARPA) 1998 and 1999 data has been a common data set used for testing (Laboratory; Lippmann et al., 2000), as has the derivative KDD Cup 1999 data (University of California, 1999). However, all of these data sets have issues, with the most significant being the age and relevance of the

data set for representing modern attacks. This research used a testing approach that aligned with previous research in similar types of attacks, and made use of the attack data that had previously been used for those works for comparison purposes. Additionally, this research also created an additional new dataset that may represent the largest single corpus of modern HTTP-protocol attacks for the purposes of testing the flexibility of the proposed approach. The researcher plans to release this dataset concurrently with this work for future researchers to benefit from.

In addition to challenges related to feature selection and data set accessibility, the iterative and multi-layer nature of the system described in this research created substantial problems related to available resources on the system being used for training. To address that issue, this research designed a framework called a packet data repository which represents common storage location for packet data. This framework explicitly allows the implemented modules to specify whether the repository is used for a given data point, so while points like frame number do not repeat and would not be a good candidate, items like source and destination MAC and IP addresses can be substantially de-duplicated and replaced with a reference to the repository. This represents an explicit time-memory tradeoff decision for each module involved, and it allowed tuning of the system to run at good speeds across a range of data input volumes. Additionally, the repository framework allows the easy presentation of a subset of data from within the repository, which allowed the researcher to readily cross-train and validate using different subsets of data.

Summary

This dissertation presents an approach taken to produce a system that is capable of not only making classifications about network traffic seen, but also provides a confidence level in the prediction made to assist those tasked with reacting to alerts that are created. This chapter provided an overview of the research problem as well as a description of the proposed approach to addressing the problem. Additionally, the case was made for the usefulness of the proposed approach in many real-world intrusion detection scenarios. In the following chapters, the current related research landscape will be explored, and further details on the specific implementation and the results will be discussed. The goal was to develop a method for producing meaningful confidence levels, and to propose metrics which are relevant in the comparison of the developed system that are useful as benchmarks for comparison in other research studies.

Chapter 2

Review of the Literature

The Roots of Intrusion Detection

The seminal paper on intrusion detection was published by Denning (1987). This paper outlined a model for real-time intrusion detection built on a rule-based pattern matching system. Key to the model was the assumption system compromise would be evident in abnormal usage patterns once standard operations were known. Denning's assumptions regarding, and use of, rule-based pattern matching are evident in current literature's references to misuse/signature and anomaly detection mechanisms (Karamcheti, Geiger, Kedem, & Muthukrishnan, 2005; Papadogiannakis, Polychronakis, & Markatos, 2010 ; Sekar, Guang, Verma, & Shanbhag, 1999).

In addition to outlining the above-referenced model, Denning (1987) defined several key terms still present in a core common vocabulary and idea set for modern intrusion detection systems. For example, as defined by Denning, subjects are the initiators of actions on the system in question, while objects are the resources on the system with which the subjects are interacting. Audit records represent a log entry generated by the system in response to a particular subject's action with regards to an object. Profiles are definitions of potential expected behaviors of subjects within the system, and anomaly records are generated when there is a deviation from a profile.

Finally, activity rules are actions taken when a specified condition is met, to perform an action of some sort in the system independent of the subject's action.

Although all of these terms are certainly relevant to the problem at hand, of particular interest is the profile, the potential behaviors of a subject with regard to one or more objects. It is this profile that essentially determines the behavior that will be classified as either normal or anomalous in an anomaly-based system, and the same profile which is matched to user behavior to locate threats in a signature-based system. The profile is particularly relevant to this work because matching subsets of it will be required for achieving the goal of locating an attack based on visible fragments only.

Host-Based Intrusion Detection

One of the earliest implementations of an IDS was the "Haystack" system created for use by the U.S. Air Force (Smaha, 1988). Smaha noted three driving forces behind the need for increased security at the time: establishment of government standards and criteria for security, creation of multi-vendor operating systems, and requirements for utilization of standards for computer procurement. Although Smaha distinguished between insider and external threats, he concluded this distinction was irrelevant once outsiders had successfully penetrated a system.

The Haystack system was created with the goal of reducing large amounts of audit trail data to manageable and meaningful summaries (Smaha, 1988). The system endeavored to accomplish this through the use of two distinct but complimentary methods. First, the system would search for behavior that was out of the ordinary for the user in question and flag such behavior for further review. Second, the system would watch for activity which had been previously defined as "bad" behavior in the context of

the system log being reviewed. Although the Haystack system would recognize behavior defined as normal, the definition of "bad" behavior required input from a human expert.

One of the most notable contributions from the Haystack system arose from the fact the monitoring system was required to be much less powerful than the mainframe it was designed to monitor (Smaha, 1988). Instead of processing all traffic in real-time, Haystack was designed to operate against a tape produced by the mainframe that contained the audit trail file. In processing the file, the system would select specific data points deemed relevant for retention and analysis rather than retaining the entirety of the audit log. This process of feature selection served as the basis for later intrusion detection models, as it offered significantly better performance than full log retention and processing.

Expansion to Local Area Networks

Heberlein et al. (1990) expanded on the aforementioned host-based intrusion detection technologies by applying the concepts to the Local Area Network (LAN) environment. Their efforts were specifically relevant to the proliferation of LANs in academics, business, and research. Acknowledging LANs were initially designed to be shared by trusted users, the authors highlighted risks to network security, both intentional and unintentional, malicious and benign.

In addition to describing a standard attack model, Heberlein et al. (1990) created the Network Security Monitor (NSM). Consistent with Denning's (1987) rule-based model, the NSM analyzed traffic that was traversing a LAN and attempted to identify traffic that either met certain pre-defined "bad behavior" rules, or fell outside what was considered "normal" traffic patterns. This dual search allowed the NSM to identify both

well-known attacks as well as identifying statistical outliers for follow-up that could represent previously un-seen attacks. Also of note, the NSM applied a Monte Carlo-based statistical model for determining the most likely subsets of network traffic to search. By using this approach, the system was able to perform at real-time speeds by adjusting the depth of search to the available resources. The contributions of Heberlein et al., in terms of both attack model and application of intrusion detection concepts to network traffic streams, laid the ground work for research to follow.

Challenges in Classical Misuse and Anomaly Detection

The classical methods of IDS introduced and developed by Denning (1987), Smaha (1988), and Heberlein et al. (1990) were proposed and implemented in response to environmental pressures for accountability and increased security. However, though a necessary response to the pressures prompting their development, classical methods were (and remain) insufficient. Of particular concern is the evolving nature of the work setting and the human resource investment required to ensure a system's rules and definitions regarding both "bad" and acceptable behavior remain current, in addition to the ever changing flow of internet traffic (Papadogiannakis et al., 2010). Changes in staff assignment, agency policy/procedure, and related regulations all impact parameters for appropriate and inappropriate use of network functionality. At the very least, continual updating of the specification to reflect new and changed network conditions and hosts would create a significant management problem in ongoing use (Smaha, 1988).

For example, a change in a user's job responsibilities might require the use of new or different tools, and the change associated with that use would be flagged erroneously as a system security breach. A manual review would ultimately resolve the situation.

However, the additional work required to complete that review could result in a delay in addressing other true breach attempts (Heberlein et al., 1990).

Inherent in a system's ability to accurately identify "bad" behavior are two issues. First, the definition of "bad" behavior requires a human expert for classification. Secondly, "bad" is defined based upon that which has previously been known, without a mechanism for recognizing novel threats -- those not previously encountered by a given system (Heberlein et al., 1990; Katipally et al., 2011).

Relevant to both signature and anomaly-based IDSs is the challenge of system patterns which are either incomplete or only partially visible. Signature-based IDSs typically rely on multiple conditions which must all be met to raise an alert. In the event that a prerequisite condition existed but was not visible to the IDS, no alert would occur, resulting in a false negative. Likewise, anomaly-based IDSs rely on the appearance of sufficient deviation from "normal" behavior to conclude an attack is in progress. However, if a deviation is subtle enough to avoid detection by sensors in place, the attack would not raise an alert, resulting in a false positive (S. H. Kim et al., 2012; Ning & Xu, 2004).

Evolution of the IDS field has included a response to the recognition of the challenges posed by classical methods (Chen & Kim, 2013; Cuong & Giang, 2012). Efforts to address the challenges and maximize performance have included those which minimize the role of on-going human expertise and input. Additionally, attempts have focused on enhancing the ability of systems to respond to information which is incomplete, or not fully visible.

Use of Machine Learning

Overview of Machine Learning

Machine learning is an area of computer science deriving from both computational learning and pattern recognition. Within the field of machine learning, the field is further segmented into supervised learning, unsupervised learning, and reinforcement learning. Generally though, all three approaches share a common core of classifiers which are relevant to the problem domain in question, objective functions that quantify the quality of the classifiers with respect to addressing the problem, and an optimization process which handles iteration of the various inputs to the classifiers in search of a sufficiently acceptable solution. Classifiers also are generally further subcategorized into either multi-class classifiers that attempt to classify an observation into one of *n* known class types or one-class classifiers that attempt to answer the question of whether a particular observation belongs to a particular known class (Sommer & Paxson, 2010).

Both classifiers types are used for intrusion detection purposes in different contexts. One-class classifiers are typically applied to classify traffic as either "normal" or "malicious," with many systems opting to document a model of normal and consider anomalous traffic to be malicious (Hai, Franke, & Petrovic, 2010). The primary reason for this choice is in many systems it is easier to obtain a representative set of the traffic which should be seen for a particular application than it is to obtain a representative set of all the possible attacks which might be directed at that application. By definition, "zero day" type attacks have never been see before, thus it will always be difficult to define a complete model of malicious behavior that includes all possible malicious patterns when

some of those patterns are by definition not yet known (Mahoney, 2003). Multi-class classifiers generally tend to be applied in an intrusion detection context with a goal of identifying particular types of attacks with the reasoning that a particular type of attack will generally follow a particular structure for which a representative set can be created. In one such system, the researchers opted to create four separate classes to represent denial of service (DoS), probe, user-to-root (U2R), and remote to local (R2L) attacks respectively.

Unsupervised machine learning focuses on learning a structure from an unlabeled dataset, and is generally well suited to low-dimensionality, discrete data, even in high volumes. However, as the dimensionality of data increases or more data points become continuous, unsupervised approaches do not generally scale well in terms of the resource requirements and ability to create meaningful boundaries. Indeed, even as dimensionality *n* approaches very moderate values, the performance can suffer seriously in a highly heterogeneous dataset. Because network data flows contain many potential attributes for consideration and are highly variable in their contents, unsupervised approaches to intrusion detection generally do not perform well except in niche areas such as identifying DoS attacks (Jeong, Yoo, Yi, & Choi, 2014).

Supervised and reinforcement learning approaches are fundamentally different in that they both aim to provide the system with real-time feedback in one way or another regarding the "correctness" of classifications being made. Supervised learning systems aim to do this by actually training the system starting with a set of training data pairs that are generally an input vector of some sort coupled with an output value representing the classification or desired output from the system based on the input vector. Reinforcement

learning is different in that specific input/output pairs are not presented to the algorithm (Wilson, Fern, & Tadepalli, 2010). Instead, the system (typically termed an "agent") has the ability to observe the environment in certain ways, knows that available set of actions, and can determine based on actions taken the resulting state of the system and the immediate reward received for the particular action selected. Over a period of time, the agent is designed to attempt to maximize the total reward received, and as such, many such systems are built to deal with realities where sacrifices must be made in the short term in the form of lower immediate rewards in exchange for achieving a higher longer term total reward.

Supervised learning methods also generally require that data be segmented into training and testing data sets, with optimally no overlap between them (R. Smith, Japkowicz, Dondo, & Mason, 2008). These methods are very sensitive to the diversity and volume of data available for training as well, with much better results generally being achieved by training with data sets that provide more complete and thorough coverage of the problem domain. The goal of minimizing overlap between the training and testing datasets is important due to the idea of "overfit," or the idea that a model learns a particular dataset too precisely and becomes unable to generalize its capabilities on to other relevant datasets from the problem domain (Reiser, 2017). Use of the appropriate material for training is a particularly important issue with respect to the research described in this work, as the depth and breadth of network traffic means that it is highly likely that intrusion detection systems will encounter never-before-seen traffic given the nature of computer networks (N. Sen, Sen, & Chattopadhyay, 2014).

Application of Machine Learning to Intrusion Detection

To address the challenges in classical signature and anomaly-based approaches, a number of machine learning techniques have been introduced. Among the earlier machine learning techniques were hidden Markov models (Ourston, Matzner, Stump, & Hopkins, 2003) and genetic algorithms (Pillai, Eloff, & Venter, 2004). More recent machine learning techniques applied to intrusion detection have included neural networks (Choudhary & Swarup, 2009), Bayesian classifiers (Luo, 2010), support vector machines (Cuong & Giang, 2012), artificial immune systems (Akyaz & Uyar, 2010), and fuzzy logic-based data mining (Chapke & Deshmukh, 2015). Of note, Bayesian classifiers (Luo, 2010) and support vector machines (Cuong & Giang, 2012) share a common core with HMMs in the idea of states, one of the key building blocks of Markov models, and good performance has been seen with several of these (Milenkoski, Vieira, Kounev, Avritzer, & Payne, 2015; Xiang, Westerlund, Sovilj, & Pulkkis, 2014). A discussion of the application of Markov models will further illuminate their relevance to present-day IDSs and this research in particular.

Background on Hidden Markov Models

The seminal work on Hidden Markov Models (HMMs) was produced by Baum and Petrie (1966). Their research documented the mathematical theory behind discrete Markov chains. It was Baum and Petrie who documented for the first time an extension where some states were considered "hidden," that is, states with relationships or probabilities not directly apparent to the creator of the chain. Rather, relationships were defined by the use of a probabilistic function which allows the system to model and infer the value of unknowns given a set of known inputs and outputs. Four years later, Baum et al. (1970) expanded on the 1966 work by providing additional documentation and

proof for a maximization technique intended to make sure that the probabilistic functions underlying the Markov chain relationship prediction were reaching the most accurate possible output value (Baum, Petrie, Soules, & Weiss, 1970).

Key Components of HMMs for Implementation

Subsequent papers on the application of HMMs documented the key components of HMMs from an implementation perspective (Rabiner, 1989; Rabiner & Juang, 1986). In a given HMM, the system must contain an $\boldsymbol{N}$ number of states, an $\boldsymbol{M}$ number of distinct possible observations per state, the $\boldsymbol{A}$ state transition probability distribution, the $\boldsymbol{B}$ observation symbol probability distribution, and the initial state distribution, represented by the $\boldsymbol{\pi}$ symbol. These elements must be generally present in any system that is considered an HMM, but in some specific cases, the system may be used to solve for one particular input given the others, as the relationship between them is fixed.

Additionally, Rabiner (1989) documented several known issues with the general application of HMMs that implementations must take into account. These issues include appropriate scaling of values during state transition coefficient re-estimation, proper initial estimation of $\boldsymbol{A}$ state transition parameters, and the impact of insufficient training data on the results of the algorithm. All of these factors must be considered if the resulting model is going to be viable; but, perhaps the most important contribution was the proposal of the Viterbi algorithm (Rabiner, 1989; Viterbi, 1967) as a solution to dealing with one of the three core problems with HMMs, the question of how to determine the likelihood that a given model could give rise to a particular observation sequence. The Viterbi algorithm performs a recursive backtracking maximization of the state transitions until the maximum probability series of transition has been located. This

probability can be correctly thought of as the “best case” likelihood that the series of events was generated from the model in question.

HMMs and Pattern Recognition

HMMs have been shown to be a valuable tool in associating data points and recognizing the presence of a pattern in several fields. Success in fields with similar types of problems and complicating factors is relevant to the likely performance of HMMs within the field of intrusion detection. Fields in which HMMs have been useful include visual recognition (Stoll & Jun, 1995) and speech processing (Scherer, Glodek, Schwenker, Campbell, & Palm, 2012). Additionally, HMMs have been deemed valuable in the field of intrusion detection (Corona, Ariu, & Giacinto, 2009; Xie, Zhang, Seifert, & Zhu, 2010).

Visual Recognition

An early application of HMMs to a pattern recognition problem was documented by Stoll and Jun (1995) in the area of visual recognition. The researchers used HMMs to model human movement with the goal of classifying a particular movement as one of six defined sport activities. The complexity of the particular problem is increased because of the fluid nature of human movements, variations with which different individuals perform similar motions, and the frequent self-occlusion inherent in visibility from a small number of fixed cameras. The HMMs performed well, with accuracy between seventy five and one hundred percent depending on the level of noise in the input video data, substantially higher than other methods that were current at that time.

Speech Processing

Speech processing is another pattern recognition problem to which HMMs have been frequently applied. Scherer et al. (2012) used HMMs as a recognition tool for

laughter in conversations. The researchers compared HMMs to other models including support vector machines (SVMs), which are also frequently applied in the intrusion detection space. Although SVMs performed well in offline testing, the HMM-based model far outperformed the SVM in online testing, and the researchers posit two potential reasons for that. First, they highlight the fact that HMMs perform particularly well on un-segmented and unlabeled data. This characteristic was also particularly useful in the application proposed in this work, as network traffic flows are also un-segmented and may be unstructured. Second, the researchers noted that the HMMs were able to be trained more effectively than SVMs because the SVMs were far more sensitive to the ratio of laughter to speech in the training set. Because the ratio of the target state (laughter) was particularly low, the SVMs did not perform well on the identification. This is also applicable to the proposed research, as malicious packets were generally far less common than benign packets in a general network environment.

Intrusion Detection

Of particular interest to this work, HMMs have also been frequently applied for pattern recognition in intrusion detection with substantial success. Corona et al. (2009) applied HMMs for the identification of attacks against web applications. The researchers used an ensemble of HMMs (between three and seven) trained independently on the same dataset but with varying initial parameters to classify web requests as either normal or malicious. The dataset used was labeled using a semi-automated process, with just under one percent of the dataset representing malicious requests. Even with relatively low occurrence of malicious requests, the ensemble HMM used was able to detect ninety-six percent of attacks with less than one percent false alarms, demonstrating that it was a viable choice for pattern recognition.

Another example of HMMs being applied for pattern recognition in intrusion detection focused on device-level malware detection for cellphones (Xie et al., 2010). The researchers used HMMs to model normal user behavior, including both keyboard and touch interactions as well as system calls being made to key functions in the operating system. The HMM was built with states representing the various stages of user interaction and the resulting system call from each interaction. At each stage, the current action was modeled against the "known good" interaction from the user's normal behavior, and the system could block access to critical function calls if the percentage match with the "known good" interaction dropped below a defined threshold. This work further demonstrated the ability of HMMs to model a particular behavior (in this case, valid behavior) and identify behavior that fell outside of the expected model. This capability was foundational for the research proposed in this work, as the ability of HMMs to model a process and determine the likelihood a particular set of actions match the modeled process is precisely the application intended for use.

Environmental Incentives for HMM Usage

Research has also focused on the use of HMMs in chaotic, noisy environments similar to conditions frequently seen in large-scale networks (Myers, Singer, Shin, & Church, 1992). This work proposed the use of a discrete space model with continuous observations, a design in which the states in the model did not directly relate to actual states seen in the modeled system. Instead, only the final output states of the model were associated to actual output states in the modeled system, leaving the remainder of the system to adjust itself as needed to achieve noise reduction. The researchers' goal was to be able to implicitly isolate and remove noise without the need for explicit noise removal. This concept is particularly applicable in the context of the work proposed in this paper

as any attempts to remove noise in a network environment could result in the removal of important data that could be useful for prediction purposes.

Another major contribution to the application of HMMs focused on the ability to remove the state independence assumption present in early iterations of HMMs (F. J. Smith, Ming, O'Boyle, & Irvine, 1995). The independence assumption is particularly troublesome for applications to network intrusion detection, as it causes the loss of information about the temporal relationship between successive events. Although a temporal relationship between packets does not necessarily mean the packet contents are related, research on attacker behavior has shown that they generally follow a defined flow of activities (Ning & Xu, 2004), and time relationships between packets proved to be important in the systems developed for this research.

Suitability Issues with Other Machine Learning Approaches

As has been previously mentioned, a variety of machine learning approaches have been applied in Intrusion Detection. Some of the other methods have included genetic algorithms (Pillai et al., 2004), neural networks (Choudhary & Swarup, 2009), Bayesian classifiers (Luo, 2010), and support vector machines (Cuong & Giang, 2012). Although some of these methods have been shown to be successful in certain approaches (R. Sen, Chattopadhyay, & Sen, 2015), there are challenges with each for particular aspects of this research.

Genetic algorithms, while well suited for feature selection, have proven to be relatively ineffective in actually classifying events and identifying intrusions. Indeed, the more successful attempts at applying genetic algorithms have generally done so in the context of another method (Ahmad, Hussain, Alghamdi, & Alelaiwi, 2014; Sujatha,

Priya, & Kannan, 2012). Additionally, the structural composition of genetic algorithms does not lend itself to the probability estimation that this research is focused on.

Even recent application of advanced neural network structures such as Back Propagation Neural Networks (R. Sen et al., 2015) have shown issues with application in real-world environments. Although the researchers were able to achieve very high accuracy rates with respect to the synthetic KDD data set, their system was suitable only for batch processing, and they admitted that substantial additional effort and modification would have been required to make the system ready for online operation and learning. Computational capability of their model was also limited by both the dimensionality of the feature space as well as the more advanced algorithm in use, and they were quick to admit that their method was not yet suitable for online use from a performance perspective in a production-scale network.

Bayesian classifiers (Luo, 2010) and support vector machines (Cuong & Giang, 2012) both share some characteristics with HMMs in terms of function and operation. Most notably, all three approaches use the idea of discrete states with data mapping into a particular state based on its characteristics. However, in terms of practical application, both have some design differences which make them less applicable to this research. Naïve Bayesian classifiers are, by their nature, unable to learn interactions between features, a key shortcoming in intrusion detection as varying combinations of features will often indicate different results, and a fact noted in changing datasets seen by Swarnkar and Hubballi (2016). Additionally, both Bayesian classifiers and support vector machines lack the concept of probabilistic transitions between states, instead, opting to represent states as chains of conditional probability or in N-dimensional

hyperplanes, respectively. While both approaches are viable and useful in particular concepts, the probabilistic transition matrix inherent in an HMM more directly lends itself to feeding a confidence-level calculation, which is critical to the success of this work.

HMMs and Multi-Stage Attack Detection

One of the early attempts to apply machine learning to detect multi-stage attacks using HMMs was documented by Ourston et al. (2003). They proposed the design and application of an HMM to make determinations about potential attacks in progress, defending the selection of an HMM over various other available models. Specifically, they suggested design of an HMM implies that each step's probability is independent of the previous step that led to it. This independence of each step allows a rather accurate modeling of a scenario common in the intrusion detection space whereby a particular resultant state could be reached by several different paths, and the path which led to the current state may not be directly relevant to the prediction being made.

Another approach for correlating multi-stage attacks was the attack graph approach addressed in research by Wang, Liu, and Jajodia (2006). This research attempted to address the problem that many correlation methods use which relies on an in-memory index of known recent alerts. Because of the problem of finite memory, the index was typically based on a sliding window, resulting in particular difficulty performing a match when an attacker either by chance or intentionally spread their attack phases over a period of time. The method proposed used an attack queue graph approach to selectively search for and retrieve only the most recent relevant alert when attempting correlation, an optimization that the researchers noted provided significant improvements

in ability to process alerts in real time as well as the ability to match alerts in a live system. However, the limitation of the processing to those alerts that are both "recent" and "relevant" could mean that alerts which were relevant for correlation are not included.

An extension of the attack graph applications called causal event graphs was the focus of research by Pan, Morris, Adhikari, and Madani (2013). In their work, they applied the concept of attack graphs linked via Bayesian network-styled relationships. This extension gave them the capability of modeling cause-and-effect probabilities on top of standard attack graphs, enabling the system to learn more complex relationships and expose that expertise in its attack classification. However, the attack graph and causal event graph methods were both generally used for producing appropriate signatures for diagnosis of attacks, and neither approach provides the confidence level capabilities sought by the current research.

HMMs and "Silent" Attacks

Khanna and Liu (2006) proposed a system for dealing with attacks which are considered "silent," that is, that they operate under the configured thresholds for detection of deviant behavior either because of extremely conservative attack configuration or insider knowledge of the values that would raise an alarm. These silent attacks are a frequent cause of false negative error in intrusion detection systems. The system proposed by the researchers included a Gaussian model used to describe observations on likely attacks which then fed an HMM engine with a hidden state used to predict whether an attack is in progress.

Although the specific application proposed was in mobile ad hoc networks, the work was significant because it described an intrusion engine which includes both a standard HMM engine as well as a feedback component. The inclusion of a feedback mechanism allowed the system to adjust in real-time to behavior changes which are part of normal network operation. Nonetheless, the researchers also pointed out that the proposed use of the HMM model contains a serious limitation in its design: the nature of the model means intrusion state cannot be inferred from reviewing specific inputs or with regards to certain targets, thus an attack in progress is quantified through only a statistical probability with no supporting information that would be useful to responders.

HMMs and Attack Prediction

Research by D. Ariu et al. (2011) also investigated the use of Hidden Markov Models in attack prediction, specifically as applied to HTTP traffic. They found the HMM algorithm was able to demonstrate better performance than previous attempts by specifically including packet payload in the modeling rather than just trying to make a determination based on packet metadata. Further, the system also demonstrated an enhanced level of accuracy obtained using multiple classifier functions within the system. Finally, the researchers proposed an algorithm for composition of results from multiple classifiers. This approach led to a net decrease in the computational cost of the algorithm, because efficiencies were gained by selectively excluding certain classifiers based on values obtained for others that had already been executed.

HMMs and Attacker Behavior Analysis

Katipally et al. (2011) sought to build on successes with previous models of attacker behavior by outlining a generic system using Hidden Markov Models to analyze and predict attacker behavior based on alerts generated by a Snort IDS system. One of

the inputs for their HMM was a human expert opinion of the attacker type and intention based on analysis of the full attack stream. Based on the results described in their research, this input value appears to be very meaningful in predicting the behavior of an attacker while an attack is in progress. Specifically, attacker types were broken into one of eight groups. The groups generally also fell into two categories. The first was attackers who were generally in pursuit of money or other resources, which included criminal groups seeking to attack a system for direct monetary gain, phishers seeking to steal identities or information, spyware/malware authors intending to produce and distribute harmful code, and bot-net operators compromising and controlling networks of systems for the purposes of executing attacks that require a critical mass of resources to be effective (phishing, denial of service, etc.). The second category was those who were focused on power or notoriety, including insiders representing disgruntled employees, terrorists aiming to incapacitate or manipulate critical infrastructure in order to cause casualties or damage public confidence, hackers breaking into networks for the challenge, and nations acting on information-gathering or espionage grounds.

In the work by Katipally et al. (2011), each attack stream was reviewed by the human expert and associated with the actions likely to be taken by each type of attacker in an attack situation. The researchers acknowledged that the use of a human expert represented a weak point in their approach, as the model relied on human input for classification in a key parameter. Although this is certainly feasible in back-testing scenarios, it would be impossible to do accurately and completely in real-time, thus depriving the HMM of a key input variable.

HMMs and Botnet Traffic

Another use of HMMs in intrusion detection was attempted by Lu and Brooks (2011). Their work focused on the application of HMMs to detect and distinguish botnet traffic in a network. Specifically, the HMM produced as part of their research focused on the relationships between packet timings in botnet command and control traffic to identify traffic that was related to a common botnet, Zeus. Packet timings were selected for the focus of the work because they are readily identified and quantified without reverse engineering malware or decrypting traffic. A typical HMM relies on a priori knowledge of the structure of the model in order to make predictions, that is, the relationships between the states must be fully known, and the probabilities of transition between states is then inferred. The work by Lu and Brooks (2011) used a modified version of an HMM which created both the states and transition structure from a set of sample data with no prior knowledge of the states and relationships (Shalizi & Crutchfield, 2001; Shalizi & Shalizi, 2004). The ability to build an HMM based on the dataset with no prior knowledge was critical for the system described in this report, as the necessary states and relationships between the visibility of one or more fragments and the presence of a larger attack are not well understood and may even be different for different classes of attacks. Another key contribution made by Lu and Brooks (2011) was the application of confidence intervals based on receiver operating characteristic (ROC) curves such as those described by Brooks, Schwier, and Griffin (2009) to determine which of a series of generated HMM models represents the best trade-off between true positives and false positives. A version of this model was used for comparison in the research described in this work, as it helped address a number of the classical HMM problems documented by Rabiner (1989) and referenced above.

Exploration of Enhancements to HMMs

In addition to the above-referenced applications of HMM technology to intrusion detection, some non-HMM work related to multi-stage attack detection has offered key insight into potential improvements to standard HMM models. One such significant contribution in the area of multi-stage attack detection was made by Alserhani et al. (2010), who identified a number of issues facing modern approaches to detecting multi-stage attacks. Consistent with Ning and Xu (2004), they observed most existing methods relied on either rule-based methods, likely to miss new attacks because of the lack of defined rules, or statistical methods that related stages based on probability without a cause and effect component. Their research described a statistical model which added an explicit cause and effect relationship model, allowing for the viewing of attack stages as a "waterfall" of sorts where the visibility of early stages contribute to the expectation of and correlation with later stages of the attack.

Ultimately, the proposed model of Alserhani et al. (2010) achieved detection rates marginally higher than existing methods, but it substantially reduced the number of false positives that were present in the alert stream. Although their work still relied on visibility of a substantial portion of the attack for alerting, the addition of cause-and-effect relationships represents an important idea for the research documented in this work. The idea of causality would likely be a major factor in predicting the likelihood that a given fragment of traffic is part of a larger attack, and the approach they documented appears to be viable for use with HMMs.

Other attempts have been made to leverage the research on multi-stage attacks and the relationships between the various stages to provide general predictions about

attacks which may be seen in the environment. One such approach used general history to build a graph of related attacks and leveraged that graph to make forward predictions on the upcoming attacks based directly on previous attacks (Cipriano, Zand, Houmansadr, Kruegel, & Vigna, 2011). Their approach used a two phase approach. Phase one focused on building a table based on a series of Snort (Roesch, 1999) alerts based on four attributes within the alert, the attack type, the source address, the destination address, and the time the alert was received. The system would then traverse the alert list with a sliding time window to match alerts into a set of attack sessions, defined as attacks that met one of three criteria: same destination address, same source address, or source address for the alert is the same as the destination address for an alert already received within the window.

After performing the data extraction process, the Nexat system created a hash table which was then fed to an HMM for training. The algorithm had the advantage of pre-associating the alerts before they were provided to the HMM, which substantially reduced the training time needed since the HMM's only focus was to learn the relationship between a small subset of alerts rather than the entirety of the traffic set. The HMM is effectively being taught in this case not to recognize an attack, that task was left to Snort, but just to learn the pattern of attacks that have actually occurred so that they can be recognized in the future. This approach was able to achieve a high prediction accuracy for an attack coming, but had a relatively low rate of accuracy on predicting the specific attack which was to occur, which limits its utility for security personnel responding to an attack. Additionally, because it was built on Snort as an alert provider, it

was not able to locate or predict never-before-seen attacks in a stream (Cipriano et al., 2011).

Neural Networks and Confidence Levels

Neural networks have been used with some success to perform attack identification and classification (Elfeshawy & Faragallah, 2013; Staudemeyer & Omlin, 2013). While they have been very successful with certain classes of attacks, Staudemeyer and Omlin (2013) do point out specific shortcomings with accurately identifying remote-to-local and user-to-root attacks. Specifically, neural networks are well suited to events which exist in a single view of the data, in this case, a specific packet. Attacks which occur over a range of packets and which cannot be reassembled directly based on the protocol in use are much harder to locate as neural networks do not maintain state effectively. However, this research did apply neural networks in byte occurrences within a particular HTTP stream very effectively, with the end result being very high relative classification rates as part of the overall meta-analyzer in certain types of attacks. Even more important for this research, neural networks also excel at consolidating different input sources and determining proper weighting, and a neural network was ultimately selected for the meta-analyzer layer described below due to both its effectiveness and its portability.

Feature Selection

A discussion of IDS and machine learning technologies would be incomplete without addressing feature selection algorithms, as the features selected for system learning are integral to a system's ability to accurately predict an outcome (Azzouzi & Kadiri, 2015). Specifically, Sommer and Paxson (2010) observed that the features

selected for a given type of event would even likely differ based on an understanding of the machine learning algorithm in use. Below, this work outlines specific research on the evaluation of the suitability of feature selection methods for IDS, as well as specific feature selection methods which have been used in the context of IDS and observations from their use.

Suitability of Feature Selection Methods for IDS Application

There are several well-documented ideas on the suitability of a given feature for use in intrusion detection technology. Wressnegger, Schwenk, Arp, and Rieck (2013) presented three such ideas in the context of the use of n-grams in anomaly-based intrusion detection, but the suitability criteria they developed are relevant to other models as well. The first idea put forth by Wressnegger et al. (2013) was that of perturbation, or the expected percentage of benign packets which were not part of the data available for training. A high perturbation value implies that many new values are being seen which should be identified as benign but were not seen or classified during training. The presence of new values is virtually a given in a network environment, and so the feature method must be capable of adapting the feature space in real time and adjusting the system training appropriately.

The second idea described by the researchers was density, or the ratio of unique packet contents to the total number of possible contents available in the underlying alphabet which defines the packet structure. Higher density implies that the feature space is more crowded, and can lead to difficulty correctly classifying items (Torrano-Gimenez, Nguyen, Alvarez, & Franke, 2015). In some cases, it may be necessary to arbitrarily stretch the feature space in order to reduce density. In these cases, logarithmic functions

may be useful in the comparison process as they can allow scaling of values over a stretched space. In general, a discretization function should take into account density, and optimally, features which represent benign behavior should be distanced from those representing malicious behavior as far as possible to optimize training time.

The third criterion for suitability as a defining feature is variability, or the overall entropy of the data set compared to the maximum possible entropy of the underlying alphabet. Higher variability implies higher randomness in packet contents, and portends difficulty in classification due to the inherent variability of the feature. Additionally, it has been observed in several cases that the growing prevalence of "dev ops," where developers are able to make key operational decisions around selecting and deploying the application stack, has led to substantial growth in the variability of common protocols (Ahmadi, Biggio, Arzt, Ariu, & Giacinto, 2016; Mehetrey, Shahriari, & Moh, 2016). Optimally for the purposes of the research described in this work, all three criteria should be minimized by the discretization methods in use for each feature.

Clustering Approaches to Feature Selection

One approach to feature selection layered fuzzy C-means clustering with neural networks and support vector machines to select optimal features at each layer for classification of network traffic (Chandrashekar & Raghuveer, 2012). Fuzzy C-means clustering was used because the method generally achieves better results than other clustering methods because of its inherent ability to allow data to be assigned to more than one cluster. Specific research on clustering techniques and their use in feature selection for intrusion detection has evaluated fuzzy c-means, k-means, and hierarchical clustering. The ability of the fuzzy c-means approach to allow a single feature to be

selected for more than one cluster enables much more flexibility and better modeling of complex relationships between different network packets (Nadiammai & Hemalatha, 2012). Specifically, clustering algorithms are generally considered a good fit for feature selection as they are specifically designed for learning and quantifying the relationships between different data values. Once the clusters were formed, neural networks are trained based on each of the clusters, with each cluster being assigned its own network (Chandrashekar & Raghuveer, 2012). This process allows each network to learn the specific subset of relevant data, and the results are then fed into a support vector machine for ultimate classification of packets. Although the research used a support vector machine rather than an HMM, the two approaches are similar in their requirements, and thus, the approach to feature selection and pruning is likely viable for HMM based approaches as well as long as the HMM model is capable of self-generation of states and transitions (Shalizi & Shalizi, 2004).

Bayesian Probability in Feature Selection

Another, rather different, approach to feature selection was documented by Sharma and Mukherjee (2012) in their work on naïve Bayes classifiers used for intrusion detection. The researchers' total population of possible features included 41 different data points, but they created a separate classifier for each of the types of attacks their system was intended to detect. The classifiers' categories included denial-of-service, probe attacks, user-to-root, and remote-to-local. For each of the distinct category classifiers, the researchers applied domain knowledge coupled with a sequential search of the feature set, removing one feature at a time until the accuracy dropped below a given required minimum threshold.

Once the subset of features was determined for each category, the classifier for that category was trained to specifically detect attacks of that kind using the features that were determined to be relevant (Sharma & Mukherjee, 2012). This split-model approach where the classifiers were separated by category allowed each category's model to be tuned more specifically than a classical single model approach would typically allow. Ultimately, this led to better recall than non-split layered models achieved using the same dataset and criteria.

Two particular contributions of note were made in the work by Sharma and Mukherjee (2012). First, the researchers documented the observation that there are drastic differences in the frequency of different classes of attacks and their hypothesis that those differences contribute to the difficulty with precision and recall typically faced by intrusion detection systems. Second, the approach of using different classifiers for different types of attacks appears to offer some viable improvements, and since the naïve Bayes approach is closely related to HMMs in structure and functionality, a similar approach may prove to be useful for the research proposed in this work.

Evolutionary Algorithms and Feature Selection

Evolutionary algorithms have also been evaluated for their use in optimizing feature selection. Zaman et al. (2013) tested several evolutionary algorithms including binary particle swarm optimization and different evolution to determine which algorithms yielded the best trade-off between number of features (which directly drives state space) and accuracy of the results. Each of the evolutionary algorithms were then applied using both a neural network and a support vector machine to see which algorithm consistently produced the best and most concise set of features. Using different evolution, the total

number of features selected was cut by 68% on average across their experiments, providing a substantial reduction in the overall state space which must be maintained by the machine learning algorithm used for classification.

Another proposed approach to feature selection for intrusion detection applied genetic algorithms to select a subset of a population of 41 features in the KDD Cup ’99 dataset as relevant for training a support vector machine (SVM) to perform intrusion detection (Saha, Sairam, Yadav, & Ekbal, 2012). For each generation created, the fitness of each combination of classifiers was calculated based on the error rate of the SVM run against the test data set. The best performers were more likely to be maintained in each generation, crossovers and mutations were performed, and the next generation was formed and evaluated. The process iterates for a defined number of generations, at which point the best found combination is used going forward to train the SVM for continued operation. Use of the genetic algorithm iteration for feature selection reduced the error rate from 22% to 13%, once again showing that iterative machine learning models were capable of improving the feature selection process and achieving a good combination of features for performing intrusion detection.

Y. Zhang and Rockett (2009) demonstrated an approach to feature selection based on multi-objective genetic programming (MMOGP) which included a mechanism for evolving both a feature extraction and feature selection that represents a good fit for the problem at hand. The work on MMOGP compared it with a variety of other classifier algorithms including several that are frequently used in intrusion detection. In many cases, MMOGP demonstrated an improvement over the other algorithms on common classification problems, and in no case was it worse than any of the other options. The

algorithm performed at least as well as every other algorithm it was compared to primarily because it essentially evolves a good method specifically for the problem to which it is being applied. This makes it extremely powerful, as it can be used to re-evolve the relevant feature set as the threat landscape changes. Additionally, the multi-objective design allows the introduction of arbitrary objectives beyond simple fitness relevant to the specific application such as reducing complexity in systems where computational resources are a primary limitation, which is often the case in IDS applications.

Approaches to Prioritizing Intrusion Detection Output

On the Accuracy of Intrusion Detection Output

Before one can focus on prioritizing output of an IDS, one must first be concerned about the overall accuracy of the alerts the system is producing. One of the most significant problems in intrusion detection is determining an objective method of measuring the ability of a given system to correctly classify events as either normal traffic or attack traffic. Although various methods have been proposed and used, it is difficult for any one approach to fully capture all relevant information with respect to the accuracy of an alert stream. Largely this problem is structural within the field of intrusion detection – the volume of traffic in most production systems and the ever changing nature of network traffic both contribute to the complexity of accurately classifying traffic. To combat this issue, various specialized metrics have been introduced with the goal of quantifying the accuracy of alerts generated by a given system.

One such specialized metric was described by Gu, Fogla, Dagon, Lee, and Skoric (2006) in their work on the Intrusion Detection Capability ("CID") metric. The CID metric is described as an application of information theory that applies a joint probability mass function to quantify mutual information and compare that to overall entropy in the dataset. The result is a metric which is capable of capturing all of the primary metrics generally used in the measurement of an IDS: true positive rate, false positive rate, positive predictive value, negative predictive value, and base intrusion rate. However, the CID metric is useful only at the level of measuring the overall quality of IDS output.

As such, it can answer the question of whether a particular IDS performs better *generally* (though not necessarily for a given environment) than another, but it offers an analyst tasked with reviewing output from the IDS no additional information in terms of prioritization of alerts for response. Although such a metric may be suitable for IDS comparison and tuning, the aggregate approach necessarily falls short in addressing the goals of the research proposed in this work. The major limitation is that the results of the metric are only available in retrospect, that is, the results of the classification can be compared against the known classifications to determine to what extent the algorithm produced the correct classifications, but the metric offers no indication of how accurate a forthcoming prediction is expected to be.

Recent work on summarizing the common methods of evaluating intrusion detection output has indeed underscored the myopic nature of reviewing intrusion detection performance. In their summary work on the evaluation of intrusion detection systems, Milenkoski et al. (2015) are able to summarize the aspects of the metrics that are reported in to two categories, those related to security and those related to performance.

According to the researchers, the security metrics used to evaluate intrusion detection systems can essentially be summarized in to three categories: attack detection accuracy, attack coverage, and resistance to evasion techniques.

Amongst that group, research from Sommer and Paxson (2010) shows that the general focus is on detection accuracy and coverage, as practical applications in most environments are not concerned with specific efforts to exploit IDS edge cases for evasion. Although all three of those metrics are important to some extent, there is a clear lack of metrics related to the ability of an IDS to predict the accuracy of its own output. Review of the metrics they classified as performance-related metrics shows that nothing in that space would indicate the general use of predicted-accuracy metrics either (Milenkoski et al., 2015).

Rather than a system or metric that is only calculable and relevant in retrospect, it seems desirable to be able to include as an explicit output a value which represents the level of confidence that should be place in the prediction being made. The ability to make a skill estimate with respect to a prediction would be useful for analysts who are tasked with following up on alerts which are the result of a prediction. A similar situation regarding retrospective accuracy calculations existed in the field of weather modeling and forecasting approximately thirty years ago, and the inability to make skill predictions in that field led to significant work in the field of uncertainty modeling.

Uncertainty Modeling

The specific exercise of quantifying and characterizing uncertainties arising from the imprecise modeling of systems is known as uncertainty modeling. One of the primary use cases of uncertainty modeling is seen in weather forecasting. Although

forecast uncertainty output is not directly reported, the presence and degree of uncertainty is certainly represented through the percentages generally associated with particular events (i.e. a 70% chance of rain on a given day or within a given time window), and thus the creation of systems which can quantify the uncertainty effectively is important within the domain. Uncertainty related to predictability of forecasts primarily arises out of two specific components of the forecast: the initial conditions and the model formulation (Palmer, 2000).

Uncertainty with respect to the initial conditions generally does not arise out of an *inability* to know absolutely what the conditions are, but instead, it generally arises out of *scarcity* of resources necessary to fully determine the initial conditions (Z. Zhang & Krishnamurti, 1997). Take for example, a forecast with respect to the continental United States. Although the initial conditions are generally known from a variety of weather reporting stations and technologies, knowledge of the specific conditions at a given location are generally a function of the location's distance from the nearest active reporting station and may also be affected by the quality of the equipment at the reporting station (Palmer, 2000; Rabier, Klinker, Courtier, & Hollingsworth, 1996).

Uncertainty with respect to the model formulation is another significant issue. Since models are generally built based on historical data (Z. Zhang & Krishnamurti, 1997), they are, by definition, limited in design only to the best available information with respect to prior events. In recent years, the quality of measurement has been generally high, allowing models to be very tightly fit to actual events. However, weather patterns generally follow a much longer horizon than a few years, and the ability to obtain accurate data about weather system movement and activity for time periods is

generally lower as a model attempts to work backwards in time (Palmer, 2000). Additionally, weather models are also limited to basis in previously seen events, and may not be able to reflect weather patterns which are new to an area or which are the result of new or expanded outside forces impacting the climate. An extreme example of such an impact is discussed in work by Lelieveld, Kunkel, and Lawrence (2012) where there is an attempt to quantify some of the potential impacts of a nuclear reactor accident on weather patterns and the overall climate. Ultimately, they reach the conclusion that while some impact may be determinable, largely the full impact is beyond the scope of any reasonable ability to model.

The issues with precise determination of initial conditions are also present in the field of intrusion detection. As an example, consider a network which has existed prior to the introduction of an intrusion detection system. It is certainly possible that even at the outset of the IDS operation, the network already contains one or more systems infected by a malicious agent. Even if the resources were committed to perform an exhaustive scan of the environment, there exists the possibility that an undetectable piece of malware exists that is operating and is not located by a scan. Additionally, it is common for production networks to have remote portions which are not covered by the same volume or quality of traffic sensors as the "core" network. In these areas, like in outlying rural areas in the weather analogy, it is difficult to determine precise details of the nature of the traffic.

Similarly, the issues with model formulation also have striking similarities with challenges within the field of intrusion detection. Although intrusion detection benefits from being able to capture a much higher percentage of the history of internet traffic in its

models (due primarily to the relative youth of the Internet), there are still significant resource limitations to being able to process that volume of traffic. Indeed, the amount of concern around performance of IDS systems in research (Ahmad et al., 2014; Sekar et al., 1999; Staudemeyer & Omlin, 2013) implies that it is unlikely that a given IDS could benefit from a complete history of internet traffic even if it were available, as the resources required to process that traffic meaningfully would be the limiting factor. Additionally, there are significant additional challenges within intrusion detection around never before seen traffic. While weather patterns generally repeat and tend to change over longer periods of time, the introduction of a new service or protocol can change the landscape of a particular network rapidly. Indeed G. Kim, Lee, and Kim (2014) observed that in active environments such changes might occur several times per year. While significant weather-impacting events such as nuclear reactor accidents are a rarity, significant network impacting events are a rather common occurrence in the information age. Furthermore, as is the case in weather modeling, the dimensionality of the inputs tends to be very high, further complicating the creation of meaningful models that are capable of outputting accurate and precise predictions.

In the field of weather forecasting, it is generally agreed that the ability to generate predictions is of very little use without the ability to supply some information with respect to the accuracy of those forecasts (Tennekes, 1992). Indeed, this agreement was formalized into the slogan “No forecast is complete without a forecast of forecast skill” by Tennekes, Baede, and Opsteegh (1986). To that end, Tennekes et al. (1986) proposed that “predictability”, in this case referring to the ability to quantify the likely accuracy of the prediction being made, should be treated as a variable that is calculated in

similar fashion and with the same level of rigor and support as the forecast itself. For the purposes of this work, the term used to capture this same meaning will be "defined predictability." Prior to agreement on this point within the field of forecasting, forecast error was tracked in terms of global error and the error statistics were compiled only after the fact to determine how accurate a given model had been. The researchers contended that, though the historical error data on accuracy of a model was useful, calculating error only in retrospect failed to provide useful and important data for those responsible with using the forecast. Indeed, a similar situation exists in intrusion detection, where the vast majority of systems have generally focused on the grading of performance merely on statistics regarding the determined accuracy of alerts that were previously made.

Forecasting and Defined Predictability

In order to better understand the case for the importance of predictability in intrusion detection, it is helpful to explore the precursors to the case for defined predictability within the field of weather forecasting. First, and perhaps most significant, Tennekes et al. (1986) posited that the lack of defined predictability from weather forecasts was a question of credibility. With such significant investments in technology and expertise for the time, the frequency at which a predicted weather forecast was incorrect presented a challenge to forecasters' professional reputations and credibility. The spirit of concern over this was captured in the question: "Do we really want the public to become as cynical about weather forecasts as many of us are about economic forecasts?" (Tennekes et al., 1986) Indeed, a similar situation could be said to exist with respect to intrusion detection capabilities today. Existing intrusion detection technologies do not typically quantify confidence levels for their alerts, leaving practitioners to decide based on their own experience and expertise whether a particular alert deserves to be

pursued (Milenkoski et al., 2015). Furthermore, because so many commercial systems operate in this way, it has essentially become the de-facto normal course of operation for intrusion detection technology, and little more is typically expected of intrusion detection systems. This focus has also resulted in a "limit approaching zero" situation where fractional increases in the percentage of accuracy of predictions is considered a success, regardless of whether those changes practically improve the ability of an operator to respond to the alerts in question.

In addition to the general problem of credibility and confidence, Tennekes et al. (1986) also made the case that, though defined predictability metrics were not commonly reported, many forecasters were willing to privately share their assessment of the accuracy of a given forecast based on the specific weather conditions and their own experiences. In a number of situations where organizations needed to make decisions based on the forecasts, assessments on the accuracy of the forecasts would be solicited, and at least in terms of common experiences shared by the researchers, the estimates made by knowledgeable individuals seemed to generally be correct. Although the researchers admitted this was certainly not scientific evidence that skill predictions could be made accurately, the subjective assessments and the reasoning behind those assessments pointed to the existence of a theoretical foundation on which defined predictability could be reasonably based and studied. The estimated produced by defined predictability, they asserted, were intrinsically valuable to the users of the forecasts, and thus further research and study was both justified and needed to codify and hold accountable the process by which those statistics were created (Tennekes, 1992).

Their work went on to present the first of a number of methodologies designed to specifically quantify the "skill" of a given forecast, essentially a measure of how accurate the forecast was expected to be. The model presented was built on a rather simple core idea which they believed to be the basis of most of the informal assessments that researchers gave, namely, that the accuracy of the output of a model produced on a given day, was a function of how significant that model differed from the one produced by the same system in the previous time period (in their case, the one from the previous day). Essentially, a model which made a prediction for a day which was largely consistent with the previous day's prediction was believed to be more accurate, while one that diverged from the previous prediction was suspect to some extent, with the extent of suspicion directly related to the extent of the divergence (Tennekes et al., 1986).

Of particular importance to this method is the need for the models to be forecasting in the same or a comparable output space. Comparing the differences in the output is only relevant if the output can be made to be comparable. As an example, a model which primarily predicted the presence of precipitation would not be directly comparable to a model that primarily predicted temperature, though the two are most certainly related. This comparability was guaranteed in the initial implementation because the same model was being compared across two different time periods, but this must be taken into account in any comparisons where different models are being used to accomplish a similar prediction (Tennekes et al., 1986).

One particular caveat related to predictability came in the form of the assertion that predictability is not a global concern, but rather should be considered a local variable, where the locality is defined as the area of interest. The primary reason for this

is that the accuracy of a forecast for an area is the function of a number of variables that are themselves local in nature such as specific blocking situations such as high pressure air masses. This caveat has proven important to the field of intrusion detection as well, since it is analogous to conditions which may be specific to a given network but may not exist on the internet as a whole (Mitchell & Chen, 2014). Indeed, in virtually all cases a specific network may have contributing or blocking factors with respect to a particular attack vector. Although it may be difficult to enumerate all of those factors individually, a recognition of the existence of those factors must be considered if an accurate skill prediction is going to be made. In the context of intrusion detection, this means that the level of difference considered between the models must be tunable by the system to ensure an appropriate level of sensitivity given the number of hosts and services as well as other network conditions.

Although the application of this idea is somewhat relevant in the field of intrusion detection, two primary issues must be dealt with to apply the idea directly. First, network environments generally change very quickly as compared to the changes seen in weather patterns, and thus any comparisons between one time period and the next must necessarily be much shorter than the twenty-four hour period which is commonly used in weather forecasting. In addition to the challenge presented by the speed of change, the scale of change is also a significant problem. While individual instantaneous events that impact weather models significantly by introducing new facts or information are generally very few and far between (Lelieveld et al., 2012; Z. Zhang & Krishnamurti, 1997), the network environment has ports, protocols, and systems coming and going from a particular environment almost constantly. It could be said that though the scope of the

entire universe of networking changes may not shift quickly, the scope that is relevant to a particular network may shift multiple times per day (S. Sen, 2014). As such, while a twenty-four hour window was a good initial value with respect to the ensemble comparisons that (Tennekes et al., 1986) attempted, a much lower value is likely needed in intrusion detection. In networks with a high volume of changes, it may be better to exclude time based comparisons completely and instead focus on comparisons between various models.

Another aspect of defined predictability that should be visited is the problem of infinite regression with respect to the predictability model. Popper (1988) claims that if an event is predictable, it must, by definition, be predictable to any arbitrary degree of precision provided that the predictive model is sufficiently detailed and the initial conditions are precisely known. Any failure to make a correct prediction, therefore, should be quantifiable in advance based on known and demonstrable gaps in either the quality of the model or the precision of the initial condition data. However, out of this principle, and issue arises. If a researcher must be able to quantify the extent to which the prediction is suspect, he or she must have a model for that quantification. This model must take into account not only the quality of the model itself, but the uncertainty contributed by imprecision in the initial conditions. Thus the model that determines the skill prediction is unlikely to be any better than the original predicting model, and in all but the best case, will likely be worse (Tennekes, 1992). Popper (1988) posits that this requirement for accountability within the model will then continually call for levels of regression which quickly become unreasonable when faced with a complex system. The resulting conclusion of Tennekes (1992) is that our expectations of the precision of the

skill forecast (that is, the second-order prediction made regarding the accuracy of the first-order prediction) should be tempered in light of the complexity of the system being modeled.

It is also worth noting that one need look no further than any modern forecast to see the ideas around the relevance and importance of defined predictability presented by Tennekes et al. (1986) are now a foregone conclusion in the field of weather forecasting. Indeed, there is significant recent work which is built upon the supposition that the ability to forecast the skill of a given forecast is important (Fortnow & Vohra, 2008; Grover, Kapoor, & Horvitz, 2015; Leutbecher & Palmer, 2008 ; Zarnani & Musilek, 2013). Later work has extended on the comparison between forecasts over a time period primarily through the introduction of same-period ensemble forecasts. In the ensemble forecast approach, a number of models are used to generate predictions regarding the forecasts, and the predictability metric is then calculated based on the level of divergence seen amongst the different models in both the current time period and the previous time periods, a value known as the ensemble mean (Roulston, 2005). The use of ensemble models has not been limited to weather forecasting, and the specific application of ensemble models to machine learning has been used within intrusion detection in a number of cases that are discussed below.

Ensemble Modeling in Intrusion Detection

The concept of "ensemble" methods is not new in applications within the field of intrusion detection. The earliest application of ensemble methods was demonstrated by Mukkamala, Janoski, and Sung (2002) in their work on the use of an ensemble IDS using both neural networks and support vector machines for classification. Their work applied

their proposed ensemble method to the 1998 DARPA dataset. They combined the output from three different types of neural networks (backpropagation, scale conjugate gradient, and one-step secant) with the output from a support vector machine using a formula which calculated the lowest absolute difference in the outputs with respect to any given classification. They focused on a 5-class approach using the major categories within the DARPA dataset that had been previously tagged. The ensemble method achieved higher classification accuracy than any of the individual classification methods because it was able to leverage the strengths of each method in performing the classification. Of note, the improvement in some classes appeared significant, while the improvement seen in areas which were already strong in terms of prediction ability was substantially lower. The range in improvement is shown below in Table 1.

Class	**Best Performing Independent Method**	**Ensemble Method Accuracy**	**Increase (Decline) in Accuracy**
Normal	Neural - OSS - 99.64%	99.71%	0.07%
Probe	SVM - 98.57%	99.86%	1.29%
DoS	SVM - 99.11%	99.95%	0.84%
User-to-Root (U2R)	SVM - 64%	76%	12.0%
Remote-to-Local (R2L)	Neural – SCG - 98.22%	99.64%	1.42%

Table 1 – Comparison of detection rates in early ensemble approach

The researchers did note that it was not clear if the improvement in accuracy was statistically significant in all cases, but they maintained that the improvements showed the ensemble method held promise within the intrusion detection field (Mukkamala et al., 2002).

Additional use of ensemble methods were demonstrated within the sub-area of feature selection and reduction (Chebrolu, Abraham, & Thomas, 2004). In their work, they applied both Bayesian networks as well as classification and regression trees to a reduced set of features within the KDD Cup ’99 data set and were able to achieve better results with the reduced data set than either method operating independently given a particular set of features. The same group of researchers also demonstrated an end-to-end ensemble IDS a year later which used an ensemble approach for all aspects of the IDS functionality. The more mature version of their system actually selected a specific classifier or the ensemble meta classifier on a per-class basis, allowing the system to optimize the results based on the classifiers which performed best when applied to a given data set (Chebrolu, Abraham, & Thomas, 2005). Similarly, (Folino & Sabatino, 2016) also allowed flexibility for their classification mechanism to prefer the more accurate classifiers on a per-protocol basis.

The use of ensemble methods has continually grown, and with that growth, IDS systems have generally become more and more accurate as different machine learning methodologies may be better suited to addressing different classes of attacks. Another variation on the concept of ensemble models has appeared in the use of distributed IDS systems such as the one proposed by Abraham, Jain, Thomas, and Han (2007). Their work used fuzzy rule-based classifiers placed on agents across the network. Each agent independently developed its own rule set in collaboration with the others, and the results were aggregated on a central server which assigned relevant weights to the agent output, provided feedback to the agent, and ultimately made classification decisions based on the various output values.

Similar concepts have been applied as well across logical layers in the creation of IDS systems which have visibility at the network, host, platform, and software levels and have support systems which reconcile and react from the data being received from all of the layers (Zargar et al., 2011). Although their work was originally oriented to usage in a cloud computing environment, similar concepts are certainly applicable in a locally hosted network environment. Of particular interest is the explicit creation of both local and global data sets within their proposed system. Rather than allowing each sensor to collect and correlate data and then sending the output to a central server for the master correlation to occur only there, the central coordinator also sends back to each agent a current copy of the global data set to use in its local processing. These are events which have been seen by other agents but which the coordinator believes are likely to be relevant for other agents as they make decisions about events in progress. The controller-agent approach was also used by Mehetrey et al. (2016), but they noted issues with visibility around certain attacks in the system where the data could not be consolidated in time to make an appropriate decision.

However, in all of the applications of ensemble approaches for various components of IDS functionality, there have not been attempts to apply the ensemble approach explicitly to making an estimation of the skill of a given prediction related to a network event. Indeed, the picture within the intrusion detection space seems to be much more similar to the one that existed in weather forecasting that was discussed by Tennekes et al. (1986). Like weather prediction of that time, predictions in intrusion detection are made constantly, but the accuracy of the predictions are only generally reviewed and quantified in retrospect.

This has resulted in a similar credibility problem that modern IDS systems are facing today, with the result being that analysts are frequently unable to follow up on all the alerts being generated and have difficulty prioritizing response to the alerts being generated appropriately (Mitchell & Chen, 2014). Perhaps even more problematic is that the credibility problem has become "baked in" to some extent in terms of the way that operators handle alerts being raised. Rather than explicitly considering the likely accuracy of each alert based on the rules or track record of the system that generated it, analysts in practice tend to focus first on the alerts that would represent the most significant risk were they actually realized. While this is likely to be a good approach in the current environment, it certainly represents an opportunity in terms the ability to provide more accurate estimations of how likely an alert is to be correct. Not only would this help the analyst better respond, but in the long term, it would likely drive better intrusion detection technologies since the presence of alerts which are frequently low likelihood but high risk will help drive advances that will allow the likelihood to be better and more accurately quantified. In the interest of addressing these challenges, this work below explores some approaches which have been applied to making skill predictions in various areas. Although some of these are not directly relevant to intrusion detection, they form the foundation of a set of ideas that this work built upon for development of the system.

Confidence Levels and Bayesian Probability

Bayesian approaches to probability prediction have proven to be valuable in the context of intrusion detection and prioritization in a number of instances (Chen & Kim, 2013; Sharma & Mukherjee, 2012). Similar methods have also been used to address the general problem of estimating probability of a given event using a number of inputs.

Classic Bayesian networks are defined by a series of nodes and edges which together comprise a directed acyclic graphical model of a particular system (Geiger & Heckerman, 2002). While these models can be excellent at predicting probability in the optimal cases, they rely heavily on a priori knowledge of the possible states and the relationships between those states. If the system lacks the appropriate knowledge or if the relationships change over time, Bayesian approaches can suffer significantly in their accuracy. However, in the context of an ensemble system design, Bayesian approaches have substantial advantages as support comparisons for the defined probability since they are generally very efficient to calculate in real time and can scale well to high throughput systems (Gao, Mei, Chen, & Chen, 2014). Additionally, Bayesian networks could be used as a simplifying structure for models with high output complexity to allow the model comparison algorithm to focus in on specific aspects of the output that are judged to be the most advantageous during training while still incorporating the entire output in the decision making process.

Evolutions of the Bayesian network have also added the concept of dynamic Bayesian models, with the major advantage being that the relationships can be redefined over time to better model the problem (Darwiche, 2003; Wiggers, Mertens, & Rothkrantz, 2011). However, these still suffer from substantial learning time and the network structures may not change quickly enough to produce accurate results. Additionally, use of these dynamic models in IDS scenarios must take care in application to ensure that a malicious attacker cannot intentionally create “drift” within the system by introducing changes over a sufficiently long time window to allow the model to adapt and avoid triggering an alert.

Another improvement on the Bayesian network is the concept of the naïve Bayes model. Naïve Bayes classifiers are a special case of the Bayesian network, one that includes the assumption that all features are independent of each other. Although this is a simplifying assumption that can reduce accuracy, a naïve Bayes classifier may even outperform other approaches if the available training data is sufficient. Lowd and Domingos (2005) demonstrated that the naïve Bayes model was capable of achieving accuracy comparable with the classic Bayesian network, but that those results were achieved in a fraction of the time, making the naïve Bayes model a better fit for real-time uses such as intrusion detection. Naïve Bayes classifiers have also been improved in recent years by semi-naïve models, models that remove the assumption of independence and allow features to be conditionally related in some cases. These relationships are typically limited to one or two per feature to limit complexity to less than that of the full Bayesian network, but they still allow some improvement to prediction accuracy while maintaining a significant time advantage over traditional models (Taheri, Mammadov, & Bagirov, 2011).

Another relevant proposed improvement to naïve Bayes was presented by S. Sen (2014). His work focused on the idea of "concept drift," or the natural change in expected behavior over time. He proposes the use of instance weighted naïve Bayes, an approach which has been demonstrated to significantly improve performance in instances where the available unlabeled data far exceeds the available labeled data (Jiang, 2012). In the instance weighted approach, the originally labeled data is used to classify and the unlabeled data, and the classifications are then used to label and weight the originally

unlabeled data based on the relative confidence that the new label is correct based on the naïve Bayes model.

S. Sen (2014) applied this idea to the challenge of concept drift by applying lesser weights to newly seen behaviors but increasing those weights over time as the behavior becomes more frequent. This idea holds much promise in the area of intrusion detection, as network traffic and applications are constantly changing and the ability to adapt to these changes in real time significantly increased the ability of the developed system to remain relevant in different traffic patterns. Furthermore, the existence of concept drift underscores the need for the ability of a model to output a confidence value that reflects how well it believes it reflects the reality of the situation occurring within the network. At the very least, a network which has drifted from the assumptions made in the development and training of a given model should result in the model being able to estimate the extent of the drift and the likely impact on the reliability of the prediction. In the ensemble approach, the concept of drift estimation would also be useful to baseline the expected amount of change versus the change actually seen and further improve the accuracy of the skill prediction made regarding the alert output.

Fuzzy Set Theory and Applications in Intrusion Detection

Fuzzy set theory has also been applied to the issue of determining the relevance and quality of alerts being generated from an IDS in several cases. Fuzzy sets were first introduced by Zadeh (1965), and they were defined in that work as a "class of objects without a precisely defined criterion of membership." Rather than using precise criteria, fuzzy sets instead define a membership function which then assigns a value within a specified range, generally between zero and one. The resulting output values are often

useful in domains in which the input information is incomplete or imprecise, as the output allows the determination of a degree to which an input matches rather than simply a binary output in which all additional data about the degree to which the match occurred is lost.

In the original work, Zadeh (1965) specified that the range between zero and one, inclusive, with the defined function bearing responsibility for the distribution values as expected within the range by the particular use case. Although there are an unlimited number of membership functions that could possibly be applied, generally membership functions are specified in terms of triangular, step, trapezoidal, or Gaussian functions. Use of these function types generally allow the function to be defined more concisely and computed more directly than more complex definitions, and they have been shown to be effective in handling most relevant cases that fuzzy sets are a good fit for (Zimmermann, 2001).

Zadeh (1965) also defined common set operations with respect to fuzzy sets, with the definitions of union and intersection being particularly relevant for this work. With respect to union, the union *U* of two fuzzy sets, *A* and *B*, can be concisely defined as the smallest fuzzy set that contains both *A* and *B*. The membership function of the union fuzzy set *U* can thus be defined as:

$$f_U(x) = \text{Max}(f_A, f_B)$$

The intersection *I* of fuzzy sets *A* and *B* is defined in similar terms as the largest fuzzy set which is contained within both *A* and *B*. The membership function of the intersecting set *I* is:

$f_I(x) = \text{Min}(f_A, f_B)$

Many anomaly-based IDS applications implicitly apply fuzzy set theory towards the classification of incoming events as either normal or intrusive. Rarely would a given event definitively fall into a class with no uncertainty, instead, generally an event is determined to be intrusive or not based on how close it is to one classification or the other. As an example, in a classic DNS amplification UDP flood DoS attack, DNS responses coming to the targeted system are the result of spoofed requests and generally represent traffic that should be classified as intrusive. However, a user actively using a system being attacked (or behind a network address translating-device which is the target of the attack) might be making DNS requests as part of the normal course of using the system, and the responses to those requests may be classified as intrusive because the anomaly system is flagging all DNS response traffic once the volume crossed a certain reasonable number of responses per second.

In this case, the valid DNS response should be classified as normal, but it could be classified as intrusive by an over-aggressive IDS because it would match the criteria of the fuzzy set which defined the intrusive behavior. However, most modern IDS systems would still ultimately reduce the output to a binary output, the request is either "normal" or "intrusive." The research proposed in this work would not seek to do away with that classification, but to more fully embrace the opportunity provided by the concept of fuzzy sets to output the degree to which the event is either normal or intrusive, providing the operator of the IDS with additional information upon which a decision can be made.

Another such example is detection of brute force attempts amidst failed logins to a system (Javed & Paxson, 2013). A single such failed login would likely not be

classified as intrusive, but systems generally define a fuzzy set that increases the likelihood of raising a brute force alert as the number of failed logins increase. In this case, the use of a fuzzy membership function allowed the researchers to identify distributed brute force attempts by measuring the number of failed attempts against a specific system (or related group of systems) over a sliding window of time and defining the membership function in such a way that it was able to correctly classify outliers. In their specific case, the researchers also parameterized the function to allow the detection sensitivity to be tuned to generate a volume of alerts that takes into account the amount of available analyst time to follow up on the events. Although this particular capability is not included in the scope of the research defined here, further development of the ability to allow a system to flex its output to match the available time and resources of operators to follow up on the alerts being produced should be considered a potential area of improvement for future work.

Another application of fuzzy set theory was demonstrated in work by Yu, Tsai, and Weigert (2008). They proposed an adaptive and automatically tuning intrusion system that they termed "ADAT." ADAT applied the SLIPPER rule learning algorithm (Cohen & Singer, 1999) to the KDD Cup 1999 dataset with a modification to utilize fuzzy set theory rather than simple rule classification. The SLIPPER algorithm generates a set of rules which are then combined into a smaller set of binary classifiers, with each rule contributing a score to the classifier based on whether the rule matches the input record.

In the case of the ADAT system, there were five binary classifiers created based on each of the five classes of traffic present in the KDD Cup 1999 dataset. Each of the

binary classifiers were designed to output not just a binary response, but to output a magnitude that reflected the number of rules within the classifier that were considered a "match" by the event in question. Additionally, the ADAT system was designed to be able to determine if an event raising the same sequence of alerts had been previously seen, and if it had, the resulting classification also took into account feedback on the accuracy of the previous predictions that had been made. Fuzzy set theory was applied to determine when to keep an alert based on the current backlog of alerts and the relevance of the incoming alert as compared to the existing ones. As the backlog became longer or the analyst processing alerts grew slower at responding, a low priority alert was less likely to be retained. This ability of the alert queue to flex in real time enabled better response capabilities for the individuals who were tasked with following up on the alerts (Yu et al., 2008). However, certain risks must be addressed in that any system built with such a design should be careful not to remove high-risk alerts from the stream just because a backlog of alerts exists.

Chapter 3

Methodology

Overview

For the purposes of this research, the problem of creating an approach to provide a confidence value for a prediction was handled as a combination of both a pattern classification problem as well as a statistical analysis problem. This project leveraged several well-known constructs, including hidden Markov models and neural networks, to create a system capable of producing a prediction about whether a given packet is part of an attack or not as well as a confidence level related to the prediction. The system proposed in this research used existing state-of-the-art intrusion detection systems as a point of comparison where relevant, but it also proposed the confidence level extension to current state of the art work that presented challenges in terms of making direct comparison with other works in the field. The goal of this work was not to maximize solely the *accuracy or precision* of the predictions being made; in addition to being correct, the system had a second goal of maximization of the accuracy of the skill prediction (that is, the prediction regarding the likelihood of the alert being correct).

The researcher created a system called the Confidence Layer Introduction Prototype, or CLIP, as the implementation of the ideas developed in this research, then proceeded to test it against network traffic, including both normal traffic as well as attack traffic. This approach of testing is typical in the intrusion detection space, and allows

ready comparison between the system produced for the current project and other state-of-the-art systems. For the purpose of testing accuracy, multiple datasets were utilized. Among those used were a dataset that has been widely used in the literature for comparability, along with an additional dataset used in prior research specifically in the area of web application intrusion detection. One additional dataset developed by the author specifically for the purposes of this work was also included in the testing.

System Architecture

The design of the CLIP system are outlined more fully below, including the following:

1. Data Input and Normalization
2. Feature Selection
3. Packet Processing Modules
4. Meta-Analysis Output Module

The system developed for this work utilizes an ensemble approach that couples various state-of-the-art IDSs with a purpose-built HMM-based IDS created specifically for this work. The core of the system used several state-of-the-art IDS systems (termed "modules" for this work) described in prior research or developed for this work to evaluate packets as they entered the system (either in real-time or via replay). The modules used by the system fell into two categories. The first category, a wrapped intrusion module ("WIM"), represented a state-of-the-art IDS system with classifications that had been converted into results that the CLIP system could use. The second category of modules, a native intrusion module ("NIM"), produced pipelined results in flow for

each packet presented. Each module's output was then combined in a neural network layer, producing both a final classification and a confidence level reflecting the predicted accuracy of the classification being made. The predicted accuracy values were then compared with actual classifications and the system was graded on how closely the predicted accuracy matched the actual classification. The system was also designed with feed-back improvement capability using the back-propagation capabilities of a neural network to tune the weightings and relevance of the input data based on the ongoing performance. The basic architecture of the system is shown below in Figure 1.

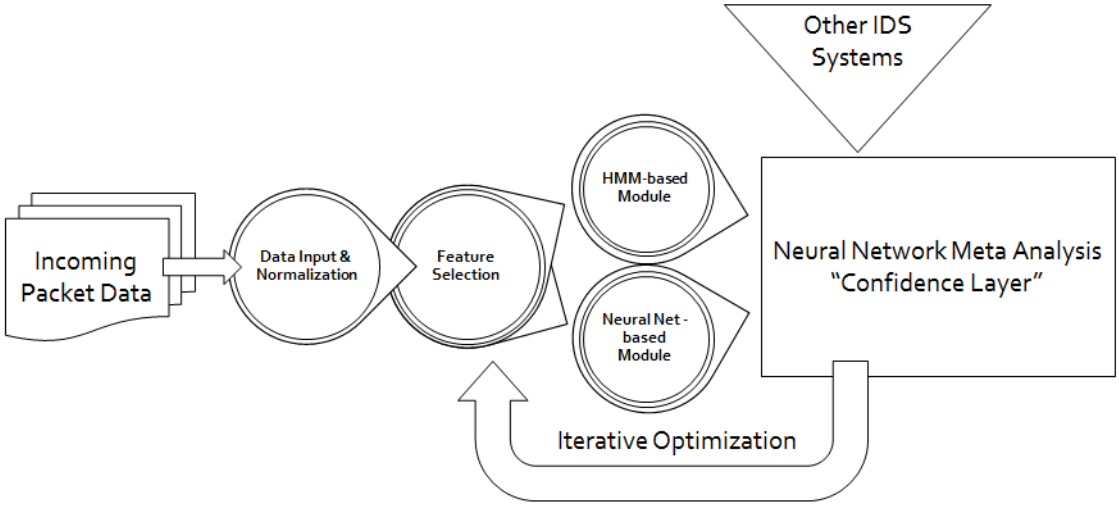

Figure 1 - Basic System Design

Data Input and Normalization

The first phase of the CLIP system encompassed data input and normalization. In this phase, the system read incoming packet data either from a PCap or PCapNG formatted capture file or directly from a network interface using the LibPcap library (Group, 2009). LibPcap was selected because it was available across all relevant platforms and provided low level access to incoming packets on the network interface while maintaining good performance and binding support from higher level languages.

The incoming packets were then parsed and relevant features were ingested into the system. As designed, the system included a modular repository-based feature storage mechanism which enabled the system to reduce duplicate data storage. This mechanism checked the value of a given feature of an incoming packet (such as the source IP address) and then stored only a reference to the repository-stored full IP address rather than storing multiple copies of all such data. Additionally, the repository was implemented as a generic interface, and thus it could be backed against a high-performance in-memory database such as Redis if required. However, for the purposes of the experiments documented in this dissertation, no remote data storage was used; only local memory on the system processing the packets was utilized.

Feature Selection

In the second phase of operation, packets are parsed and a feature set is presented to each module of the CLIP system. Wrapped modules always received full packet data for every packet that entered the system, while native modules were free to select relevant features from among the set available within the incoming packets. This functionality was closely coupled with the previously-described input and repository capabilities as the system could be configured to maintain in memory only features relevant to the registered modules, while writing the full packet data to a file for later further review and/or long term storage. The relevant features, as had been determined via the training process for each module, were then passed into the loaded modules, which were responsible for further processing.

The current features that are available for subscription and use by native modules included the raw binary representation of the packet, as well as the following pre-parsed elements:

Packet / Data-Link Layer	**Network Layer (IP)**	**Transport Layer (TCP & UDP)**	**Application Layer (HTTP)**
Packet Time	IP Version	TCP Source Port	HTTP Host
Packet Length	IP Header	TCP Destination Port	HTTP Request
	IP Flags	UDP Source Port	HTTP User Agent
	IP TTL	UDP Destination Port	HTTP Referer
	IP Protocol	Sequence #	HTTP Cookies
	IP Checksum	Acknowledgement #	HTTP All Other Headers
	IP Source	TCP Flags	HTTP Payload
	IP Destination	UDP Flags	

Table 2 - Available Packet Elements

The NIMs produced for use in this research leveraged a large percentage of the listed elements, but the system is designed such that if elements are available which are not used, the system operator can opt for them not to be stored to reduce necessary storage and memory overhead. Specific details on the features used by each module are included below in each module's respective descriptions.

Packet Processing Modules

The third phase of the CLIP system's operation included the execution of various packet processing modules responsible for producing an analysis result ("AR") for each received packet or stream, depending on the configuration. An AR was made up minimally of a packet number or other identifier, an analysis source defining the module that produced it, and a simple binary classification of either normal or attack. Additionally, the AR could contain a confidence value as well as a detailed classification of the packet in question (i.e. "injection attack" or "buffer overflow").

Wrapped Intrusion Modules

Wrapped Intrusion Modules made up a significant part of the input for the CLIP system. Part of the novel contribution of this work was the creation of a wrapper capable of ingesting the results produced by other state-of-the-art IDS systems. For the purposes of this work, the researcher used both HMMPayl (Davide Ariu, Roberto Tronci, & Giorgio Giacinto, 2011) and the Layered HMM Payload Anomaly Detection System (LHMMP) (Taub, 2013) as input systems, designing ingestion wrappers for each.

Because neither of the aforementioned systems were designed to output a confidence measure, the ingestion wrapper assumed a base confidence for each prediction made of 100%, or 1.0. This assumption was based on the belief that the designers of each of the utilized systems designed their systems to output alerts only for packets/streams which they believed were sufficiently likely to be attacks that a security analyst should act on them. Ultimately, though, the choice of 1.0 was arbitrary, and it did not make a difference in the final results produced by the system as long as it was held constant for a given system (i.e. all alerts produced by a given system were considered to have the same base confidence value from the reporting system's perspective). Additionally, neither of the wrapped modules have been equipped with conversion logic necessary to produce detailed classification on specific attack type; thus, the ARs produced by the wrappers only populated the binary normal/attack classification.

Native Intrusion Modules

In addition to the wrapped modules, the CLIP system was designed with the ability to host native intrusion modules which conformed to the interface defined by the Meta Analysis System. The interface contract required the module register to receive particular data related to the packet (up to and including full packet or reassembled

stream data), accept callbacks with the requested details, and return an analysis result for each function call. Although the CLIP system was run on a single host, it could be readily decomposed and split across multiple systems if required for throughput or fault tolerance reasons. For the purposes of the experiments described in this work, the researcher produced two NIMs. The first was a NIM called "StreamSeq" that applied Hidden Markov Models as anomaly detectors to patterns within an HTTP Stream to locate attacks, while the second was a NIM that used Neural Networks to detect previously unknown HTTP-based attacks within an HTTP stream, known as "StreamFreq". Both NIMs provided good exercise of the data processing layer as they registered for fully reassembled HTTP streams rather than individual packets, and thus they were more able to identify fragmented attacks than most of the systems against which either was compared.

StreamFreq NIM was specifically designed to address shortcomings noted in the pre-existing WIMs; it focused on the byte frequency and weighted location within the payload of a given packet. Generally speaking, a number of the most critical attacks such as buffer overflows rely on the use (and often repetition) of special, often non-ASCII printable bytes for functions such as NOP sleds and handling of control flow in a successful attack. Because the StreamFreq NIM was concerned about byte frequency, it leveraged the raw unparsed HTTP stream to directly produce its results. Using the unparsed results had the added advantage of allowing the system to "see" attacks that were not part of a section of the packet that was being formally processed and named. The StreamFreq module generally operated by noting and flagging those instances as attacks with a high degree of confidence. The confidence level of the prediction was

based on the similarity of the packets to previously-seen and correctly-flagged packets. Thus, lower confidence predictions (sub-50%) were produced for packets similar to others previously predicted incorrectly.

After collecting the total counts for each character, the character counts are then fed into a vector containing 256 values (one for each possible character), and the vector is supplied to the neural network to use for execution. Neural networks were selected for the core of the StreamFreq NIM for several reasons. First, they have been well documented to perform better than most machine learning approaches in the presence of very high numbers of inputs, which was a requirement given the potential size of the input space (maximum of 256 possible inputs in ISO-8859-1 mode, or 65,536 possible Unicode characters per plane in Unicode mode). Second, the multi-layer nature of neural networks allow the network to form conditional-like relationships between neurons allowing high counts of certain characters to trigger greater interest in the counts of other characters, a capability more linear models such as SVMs and Bayes point machines lack. Furthermore, the StreamFreq implementation and the offline training capability of the overall system meant that the relatively longer training times of a neural network were not a problem for the performance of the system.

The StreamSeq NIM was designed to identify specific patterns or sequences within an HTTP request that were indicative of attack traffic. The module operated based on sliding windows similar to those used by HMMPayl and LHMMP. However, unlike prior systems which simply used the base, StreamSeq's expanded approach included the entirety of the HTTP stream, consisting of both parameter and post data, and focused on the unique sections of known-bad packets rather than attempting to model an entire

packet. More specifically, the StreamSeq NIM operated on all the available named HTTP fields at the application layer, as well as iterating through the unknown header collection and including the full payload in the analysis. The StreamSeq NIM works by reassembling the TCP stream if it is not already complete, and iterating through the characters of the stream in sequences of length N. During the training process, the complete list of sequences is built and then de-duplicated in streams unique to the attack sequence. For this reason, the StreamSeq module benefits from the largest corpus of unique captures in terms of requests for both the attack data and the known clean data. Thus, it could reasonably be expected to suffer significantly in a case where data is mislabeled since it would eliminate potentially valid attack patterns. The deduplication process is significant because it substantially reduces the total number of sequences that must be included in training, making the process both faster and more accurate.

After de-duplication, HMMs representing each attack sequence are built with length N matching the selected sequence length. The Hidden Markov model was selected as the modeling and recognition engine because the HMM approach is well suited to arbitrary pattern recognition within a given range as has been demonstrated previously (Yolacan, Dy, & Kaeli, 2014). Because StreamSeq's focus is specifically on identifying known attack characteristics based on specific known bad sequences of input, HMMs are particularly well suited for the recognition process.

The possibilities of NIMs are particularly interesting in practice because it is possible to leverage the confidence level to produce a tightly scoped module aimed at detecting a specific class of attack. As long as the NIM reports an appropriately low confidence level for packets it is not designed to handle well, the meta-analysis layer will

adjust in order to not place any weight on those predictions. Just as in other ensemble-based method fields such as forecasting, the design of a particular system can be readily aimed at addressing a recognized shortcoming of existing models. Indeed, the researcher designed the StreamFreq module after reviewing the most prevalent false positives and false negatives produced by the systems that were already incorporated into the CLIP system and noting their areas of greatest weakness.

Meta-Analysis System

The fourth and final phase of the CLIP system was the meta-analyzer layer. The meta-analyzer accepted as input a series of analysis results from any number of systems, with the goal of distilling those into a single AR which could form the basis of a final determination. Additionally, the meta-analyzer had access to the packet repository, allowing it to factor into its decision-making process the specific type of packet being received. The current version of the meta-analyzer was built using a neural network that took up to three inputs per AR and used those to produce a final analysis result. A neural network was selected for this portion of the system because of the high potential number of inputs and the ability of the system to utilize multiple layers to model relationships between the analysis results that are useful for improving the prediction.

The neural network was designed to with an input layer, two hidden layers, and an output layer. The first hidden layer was built with four neurons per input module, plus an additional four. For each source reporting on a given packet, there were three input nodes: binary classification, confidence level, and the detailed classification if present, leaving one node per input module for future use. For the purposes of the experiments detailed below, four input systems were used, meaning the first layer had a total of twenty

neurons. The second hidden layer was built with two neurons per input system plus an additional two extra, for a total of ten neurons in the training system. The output layer provided two outputs, the first being the classification and the second is the confidence level in the prediction in the range [0...1]. Each layer used a bipolar sigmoid activation function, and the overall network was trained using resilient backpropagation learning (Naoum, Abid, & Al-sultani, 2013). The meta-analysis module did not require a given module to produce output for every packet, as any sources that were part of training but missing in operation were input as zeroes.

Anatomy of a Detection

In order to better understand the detection process, an example of a potential detection follows. Consider an HTTP request flowing through the system that looks like the following:

```
GET /WebGoat/start.mvc HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:53.0)
Gecko/20100101 Firefox/53.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: http://www.google.com/search?hl=en&q=12918193' or
3382=3382--
Cookie: JSESSIONID=76B2A14515F0632AF86EDFC986722976
DNT: 1
Connection: close
```

This request is normal in all respects, with the exception of the text "'or 3382=3382--" included in the referrer field, which is actually a test for a SQL injection vulnerability. When the request is captured by the CLIP system, the full packet data for the request is forwarded to the wrapped intrusion modules in PCAP format for processing and determination. Each module is allowed to return a result which is then translated into a per-module AR for input into the meta-analysis layer. The loaded native intrusion

modules are also forwarded the packet details, but in their case, they receive only the details that are relevant for the information they are reviewing. In the case of StreamFreq only a full raw binary representation of the HTTP portion of the packet is provided, and in the case of StreamSeq the various parsed HTTP data fields of the packet is provided.

StreamFreq reviews the occurrence of the characters within the packet, and ultimately determines that this packet doesn't appear abnormal, so it returns a "normal" classification. However, since StreamFreq is designed to recognize a specific kind of abnormality (high counts of unusual, non-printable characters that frequently occur in byte-code), and that abnormality does not exist, it also returns a low confidence value since it is aware that this is not a type of packet it has been built to accurately classify. This low confidence value is ultimately useful to the meta-analysis system when all the results are available and final classification is made.

StreamSeq receives the parsed packet information, and analyzes the various fields within the packet. Based on previously provided training, it has learned that the sequence of a single quote, followed by a SQL Boolean operator, followed by a SQL comment operator generally is indicative of SQL injection, and so it classifies this packet as an attack with a relatively high confidence level, in this case, approximately 90%. If the attack would have also included SQL keywords such as select, update, or delete or other SQL operators or comments, the confidence level could have increased even more, leading to even more weight being placed on the classification.

Once the distinct analysis results have been received, the meta-analysis later inputs each of the values into its neural network for a final classification and confidence. Through prior training in this case, the network has learned that low confidence

observations from StreamFreq are generally not reliable since StreamFreq is “good” at reporting on packets it believes it can classify well according to its intended purpose. The attack classification received from the StreamSeq NIM is given significantly more weight since it was produced with high confidence, and one of the wrapped intrusion modules made a similar attack classification as well with respect to this packet. In this case too, the system has also learned that these particular two modules agreeing results in very high classification accuracy, so a final classification of “attack” is produced with approximately 98% confidence.

In the event no modules would have produced high-confidence classifications for the packet, the meta-analysis system would have still produced a classification, but would have done so at much lower confidence than it did in this case. The meta-analysis layer can also learn to ignore the confidence levels produced by a given module, and does in the case of the wrapped modules where all confidence levels are produced at a 1, but the production of good confidence levels improves the overall output of the system significantly. This is the basis of a recommendation for future work below.

As an example of the specific improvement shown in this work, consider the example of a similar packet that is not an attack.

```
GET /search?hl=en&q=union+select+syntax HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:53.0)
Gecko/20100101 Firefox/53.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Cookie: JSESSIONID=76B2A14515F0632AF86EDFC986722976
DNT: 1
Connection: close
```

In this case, the StreamFreq analyzes the packet contents with similar results as in the first, essentially it doesn't know how to classify the packet, and so the module makes its best guess but with low confidence. When StreamSeq receives the packet, it recognizes via its models the presence of several operators that are keywords for SQL, but presumably due to the lack of various significant punctuation (single quotes, double-dashes), it classifies as an attack but with only a 40% confidence level. Once again, the classification matches the result from another module, and the system ultimately outputs a 48% confidence level for the packet. Ultimately, the system produced the wrong result, but the confidence level provided explicit insight into the fact that the prediction was suspect. Were both of these alerts received simultaneously, an analyst would have clear indication of the one which should be prioritized, the primary goal of this research.

Data Sets

The CLIP system described above was employed with data sets similar to those used in previous research on intrusion detection systems, with experiments conducted using three distinct data sets. First, the publicly available DARPA 1999 dataset provided a baseline for initial system design and testing. Although the complaints against this dataset are both many and generally valid, the researcher's inclusion of it was intentional and dual-purposed. First, a large majority of intrusion detection research still utilizes it, which results in it representing a common set to compare against for reference. Second, though the dataset is problematic, the problems and shortcomings of it are well documented (Elfeshawy & Faragallah, 2013; Lippmann et al., 2000) and thus the bias introduced is a known quantity.

In addition to the DARPA dataset, the researcher also utilized the McPAD dataset, used previously for comparison purposes by several proposed anomaly-based IDSs. The McPAD data set is composed of a set of generic attacks originally made available by Ingham and Inoue (2007) that contained 66 generic HTTP attacks and 11 shellcode-based HTTP attacks. Additionally, the McPAD researchers used the shellcode attacks in the base data set to produce 96 additional polymorphic versions of the original attacks, for a total of 173 unique attacks.

The third dataset used for comparison was a custom dataset produced by the researcher, using a combination of open source, purpose-built vulnerable web applications including WebGoat 7.1 and DVWA 1.9. Both of these applications are intentionally vulnerable applications designed for teaching web application penetration testing. As such, they made possible the efficient simulation of a wide variety of attacks in a known-vulnerable environment. For the purposes of the testing, the researcher produced three sets of data, a normal set, an attack training set, and an attack testing set. The normal set of data contained normal traffic to both applications as well as several other applications and public websites. The normal data set was produced by browsing the applications using standard web browsers as well as spider tools designed to crawl the available functions of the site.

The attack sets were then produced by following a similar process, but with attack inputs rather than normal inputs. Additionally, several open source and commercial web application penetration testing tools were also directed at the applications to provide a varied baseline of what represents an actual web application attack. The attack training set was prepared by targeting both applications, but with a subset of the attacks that were

included in the attack testing set, a total of 5,830 attacks. For attack types that were going to be present in both data sets, care was taken to ensure that each recorded attack would accomplish the same goal as the attack but using slightly different variations where possible. This created additional diversity of attacks, and ensured that the system was not simply able to memorize known bad attack sequences and recognize them directly.

The total attack dataset was created by targeting the WebGoat application only, and it contained 7,017 discrete attacks representing a variety at attack types and encodings. The data was captured using the CLIP system's PCAP functionality limited to web application traffic only. This dataset aimed to capture a variety of modern web-based attacks such as OS command/SQL injection, cross-site scripting, insecure object references, and issues with authentication and session management. The specific attack types that are part of the custom dataset are outlined below in Table 3.

Attack Type	**In Training Set?**	**In Testing Set?**
SQL Injection – MS SQL	Yes	Yes
SQL Injection – MySql	Yes	Yes
SQL Injection – Oracle	**No**	Yes
Reflected Cross-Site Scripting	Yes	Yes
Stored Cross-Site Scripting	**No**	Yes
File Path Traversal – Linux/Unix	Yes	Yes
File Path Traversal – Windows	**No**	Yes
OS Injection – Linux/Unix	Yes	Yes
OS Injection – Windows	Yes	Yes
Buffer Overflow – Linux/Unix	Yes	Yes
Buffer Overflow – Windows	Yes	Yes
Insecure direct object references	Yes	Yes
Session Management Attacks – IIS	**No**	Yes
Session Management Attacks – Apache	Yes	Yes
Authentication Manipulation	**No**	Yes
Reflected DOM Issues	Yes	Yes
Stored DOM Issues	**No**	Yes
XML Injection	Yes	Yes
XXE Attack	**No**	Yes
Cross-Site Request Forgery	**No**	Yes

Server-side Template Injection	**No**	Yes

Table 3 – Attack Types

Experiment

The test approach for this research intentionally mimicked previous research in intrusion detection, with a specific focus on HTTP-based detection since the modules implemented for the purposes of this research were both HTTP-focused. The researcher used several HTTP-capable intrusion detection systems for comparison including HMMPayl and LHMMP. Both systems were executed on the datasets described above, and the results were then prepared for comparison against the system described in this research.

The results for HMMPayl, and LHMMP were both originally evaluated in their respective research using the receiver operating characteristic (ROC), a metric primarily focused on the sensitivity of a receiver and most useful when the goal is focused minimization of false positives. However, the use of ROC for comparison was not meaningful for this research, as the inclusion of multiple classifiers make reduction to a ROC very difficult in practice. Additionally, ROC curves are sensitive to the relative size of the data sets being used for “normal” versus “abnormal” comparison, so a high ratio of normal to attack packets will generally skew the calculation. Because of the sensitivity to the characteristics of the dataset, ROC curves may not fully capture the efficacy of one system compared to another. Specifically, the ratio of normal to attack traffic in the DARPA/McPAD combined data set is very high, leading the ROC comparison to be somewhat misleading.

Instead, this research utilized the Matthews correlation coefficient (MCC) (Matthews, 1975), a measure that generally performs well across a wide range of scenarios including very unbalanced data class sizes, as well as the Bookmaker Informedness (BM) and the Markedness (MK) measures (Powers, 2011). An MCC of 0 reflects random prediction, while +1 represents a perfect classifier and -1 represents total disagreement between the expected results and the actual results. The formula for the MCC is shown below in Figure 2.

$$\mathrm{MCC} = \frac{\mathrm{TP} \cdot \mathrm{TN} - \mathrm{FP} \cdot \mathrm{FN}}{\sqrt{(\mathrm{TP} + \mathrm{FP}) \cdot (\mathrm{TP} + \mathrm{FN}) \cdot (\mathrm{TN} + \mathrm{FP}) \cdot (\mathrm{TN} + \mathrm{FN})}}$$

Figure 2 – Matthews correlation coefficient formula

In addition to its ability to handle a range of input sizes, MCC also proved to be an excellent choice for a comparison metric because it allowed correct representation and adjustment for the confidence level idea within the calculation. For a classic IDS that does not report confidence levels, each instance of a particular type of input (true positive, true negative, false positive, or false negative) increments the count by 1. However, the MCC was readily adapted to the concept of confidence levels by incrementing by the percentage confidence in a given result rather than by the value 1. In the simplest case where a system reports 100% confidence, the value would increment by 1, but in the case where the system reports 90% confidence instead in a prediction, the relevant input would increase by only 0.9. A system which outputted all equal confidence levels (for instance a confidence of 50%) would find that its MCC was completely unaffected by the confidence level calculation, but if the confidence levels prove to be meaningful, they should raise the MCC by weighting the results for towards the correct

high-confidence predictions and losing less credit for incorrect low-confidence predictions.

Similarly, Bookmaker Informedness and Markedness represent complimentary measures of the usefulness of a given prediction (Powers, 2011). The two calculations are inverses, that is, BM can be driven towards 1 while MK moves towards 0, and vice-versa. Bookmaker Informedness is the expected return per bet made with a fair and reasonable Bookmaker, thus the return of the bet increases as the chance of winning drops. Markedness is calculated as precision less the inverse precision minus one, and it quantifies how marked a condition is for a specific predictor, also represented as the probability that a condition in terms of the predictor exceeds chance. Both work in concert to additionally quantify the extent to which a prediction is valuable.

Because none of the comparison systems reported a BM, MK, or MCC statistics as part of their original experiments, the researcher first had to obtain working models of each system and prepare them for execution for comparison purposes. HMMPayl is freely available online for download, and Lawrence Taub graciously provided a copy of his LHMMP system for use in the comparisons. Some modifications were required for each of the systems to produce interim detailed output sufficient for calculating the MCC, but the effort was ultimately completed for both systems without substantial issues. The confidence level assigned to the outputs of any system that was unaware of the concept of confidence were assumed to be 100%, or 1.0 for the purposes of calculating the BM, MK, and MCC. After creation of the needed outputs from the comparison systems, the outputs were also prepared for ingestion into the meta-analysis system as wrapped intrusion modules using the procedures described previously.

After WIM preparation was completed, the system was then executed with the WIM module output, the NIM modules running actively against the various input sets, and the meta-analyzer ingesting all of the classification output. The DARPA dataset was used as a pure "clean" data set, while the McPAD data was used as pure "attack" data. The custom vulnerable data produced by the researcher was subdivided into both a normal and an attack data set. The data was then split into training and test portions, with 80% being used for training and 20% being used for testing. Additionally, a portion of the vulnerable attack data was reserved and was used purely for testing with no prior specific subset included in training to better approximate conditions with which systems frequently deal in operation around new and unseen attack vectors.

Resources

A number of resources were necessary to complete this research. In addition to the data sets described above, the system described in this research required a computer system with sufficient resources to process the data in near real-time. The researcher used a custom-built PC running Windows 10 with an Intel Core I7-6700K 4GHz CPU and 32GB of RAM. The system described was implemented using .Net 4.5 and .Net Core in the Visual Studio development environment along with the LibPCap network protocol capture and processing library. Additionally, some of the systems used for comparison would only operate on Linux, and a VMWare virtual machine running on the same hardware as described above was used for executing those.

Chapter 4

Results

Introduction

This dissertation introduces a novel concept in intrusion detection, the reporting of explicit confidence levels in predictions made by a system, as well as the use of those levels to provide both an overall meta-prediction and a meaningful confidence level in the meta-prediction (that is, the prediction about the predictions). The meta-analysis from the CLIP system created by the researcher for this project includes the output of both WIMs and NIMs. However, for the purposes of this chapter, the outlined comparisons focus specifically on the relative performance of the comparison WIM systems as they contributed to the meta-analysis produced by CLIP as well as the overall performance of CLIP. Of particular note is the proposed variation on the Matthews Correlation Coefficient described previously, and its impact on the overall reported performance.

System Operation Observations

NIM – StreamSeq

The design of the StreamSeq NIM focused on certain patterns within a packet representing attacks of various kinds. Within a given web request, there are a relatively finite number of keywords and related syntax required to trigger certain types of attacks. For example, an SQL injection attack must include certain characters or keywords in

order to be successful, and the SQL language is relatively limited in terms of the keywords that are sufficient to cause actual damage through data leakage or changes. Similarly, OS command injection attacks include a relatively limited subset of applications which are frequently leveraged by attackers to either test the efficacy of a suspected injection vector or actually execute code on the host.

Once trained, the module evaluates an incoming TCP streams to determine how closely they resemble known attack fragments. Hidden Markov models generally use a cutoff value to determine the point at which the prediction changes over from one classification to the other. The more positive a hidden Markov model reads, the higher the likelihood the pattern matches training data. The StreamSeq also repurposes the cutoff value and defines thresholds based on percentage of false positives at certain intervals above the cutoff within the training data; those percentages are used to approximate the confidence level output of the model in addition to the actual prediction being made. The second tier processing provided by the confidence level is significant in that it allows the model to self-advise the meta-analyzer on predictions in which it does not have high confidence, allowing the overall system to react accordingly.

Although this module is not particularly robust in its ability to detect a wide range of different attacks, it is very good at detecting the particular types of attacks on which it is trained. In cases of command injection attacks where the corpus of relevant effective commands for an attacker is rather small, this module actually proved very effective even in cases with commands structured differently from those seen in the training data. While the module was significantly less effective at locating and identifying attacks not previously seen by the system, it properly considered its predictions regarding those

attacks to be very low confidence, prompting the meta-analysis layer to discount its feedback in those cases.

NIM – StreamFreq

The second NIM that was part of the confidence-reporting portion of the CLIP system was StreamFreq, a module aimed at identifying unusual patterns and frequencies of characters within a given TCP stream. StreamFreq operates on the HTTP request and works by collecting a total count of each of the characters within the request. The original HTTP 1.1 specification RFC (Fielding, 1999) calls for a character set of ISO-8859-1 unless some other character set is specified, although HTML5 has generally switched to Unicode/UTF-8 as the default specified in the request. Although StreamFreq is capable of operating on any either ISO-8859-1 or Unicode, all of the training data was encoded using ISO-8859-1. This simplified the system for the purpose of execution in this research to assume the ISO-8859-1 character set which requires one byte per character.

Early experiments also tested the use of a weighting factor that attempted to capture the relative location of the characters with respect to other characters, but the weighting factor added significant computational complexity without any meaningful increase in performance, so it was disabled in the testing process used to produce the data in this report. Generally, the design of StreamFreq makes it excellent at recognizing attacks that leverage non-printable characters as those do not typically show up in significant quantities in normal HTTP traffic. In fact, the module was near perfect in its identification of shellcode and buffer-overflow based attacks.

The confidence value for this module was based on an amplified delta between the "normal" and "attack" outputs to the network. Thus, output values situated very close together generally lead to rather low confidence, while values that were further apart slowly increased the confidence value up to a maximum of 1.0 in the case where one output was maximized to 1 while the other was minimized to 0. Although perhaps less accurate than the explicit confidence value supplied by the previously-described StreamSeq NIM, this implicit confidence value still proved effective at providing meaningful feedback to the meta-analysis layer that allowed the meta-analyzer to improve performance with the metric as compared to performance without it. Comparisons both with and without the confidence output are shown below in the results.

Meta-Analysis Layer

The meta-analysis layer is responsible for consolidating the results produced by the various modules and reporting the final classification back to the controller. The meta-analysis functionality is provided by a neural network configured with three inputs per module. The first input for a given module inputs the module's prediction with respect to the TCP packet or stream. The second input is the confidence interval value from the module in the range of [0…1], and a one is used if no value is supplied. As previously explained, the purpose of the confidence value is to allow an analyst working with the system to prioritize alerts based on not only the severity of the alleged event, but the confidence the system has that the event actually occurred. The design is such that a module which frequently reports false positives with a high confidence level will, given sufficient data, have its weighting and relevance dropped, resulting in a lower overall confidence level. The classic confusion matrix which consists of true positives ("TP"),

true negatives (“TN”), false positives (“FP”), and false negatives (“FN”) is shown below in Figure 3.

	p′ (Predicted)	n′ (Predicted)
p (Actual)	TP	FN
n (Actual)	FP	TN

Figure 3 – Classic confusion matrix

In contrast, the continuous layered confusion matrix (“CLCM”) proposed and produced by this system described in this work, is more faithfully represented as a three-dimensional object which would be more accurately represented by Figure 4. Although the CLCM is shown below as a two layer object, it could be thought of better as a cube, with the two layers shown representing the front and rear sides, and the actual values being continuous all the way through.

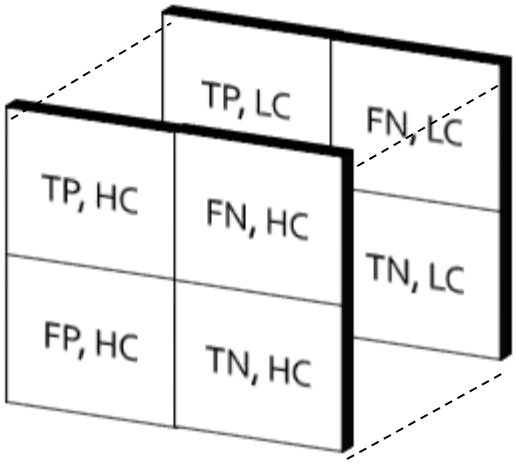

Figure 4 – Visualized continuous layered confusion matrix

The results of the meta-analysis layer are scored using a combination of the accuracy of the prediction made multiplied by the confidence level in the prediction itself. Thus, a prediction with 100% confidence would be a 1.0, while a prediction with an 80% confidence level would be scored a 0.80. The weighted values are then used to total to the TP, TN, FP, and FN values, which are then used as inputs to calculate Bookmaker Informedness, Markedness, and the Matthews Correlation Coefficient. The output of the MCC in some simplistic cases are shown below in Table 4.

Description	TP	TN	FP	FN	BM	MK	MCC
Random guessing, No weighting	1	1	1	1	0.00	0.00	0.00
Perfect prediction, No weighting	2	2	0	0	1.00	1.00	1.00
Random guessing, but incorrect predictions are supplied at only 50% confidence	1	1	0.50	0.50	0.33	0.33	0.33
One incorrect guess, incorrect supplied at 50% confidence	2	1	0	0.50	0.80	0.67	0.73
Random guessing, All predictions at 50% confidence	0.50	0.50	0.50	0.50	0.00	0.00	0.00

Table 4 – Matthews correlation coefficient examples

Analysis of the table above shows that in the case where two systems are equally good at making predictions, the one able to supply better confidence data about the predictions it is making will ultimately have improved BM, MK, and MCC values. Demonstrating the ability to separate good predictions from poor predictions in a quantifiable way is one of the stated goals of this dissertation, and it aligns intellectually with the expected value and usability of reliable versus unreliable predictions.

Comparison of Results – DARPA & McPAD Dataset

Research for the current project began the comparison process by consolidating the DARPA and McPAD datasets into a single runnable test set. The testing dataset was composed of the DARPA test set (604,055 packets, all clean) as well as the McPAD dataset (1,035 packets, all representing attacks). In the case of the comparison systems HMMPayl, LHMMP, and SnortIDS, the datasets were run separately through the processor because the systems were not capable of processing more than a single PCap at a time. In the case of the CLIP system, both PCaps were fed into the test harness simultaneously which consolidated and processed them in a single pass. SnortIDS was configured to ignore all non-HTTP traffic and attacks for comparison purposes, and all of the other systems in question were designed to extract the relevant HTTP data as part of their execution processes. In total, there were 84,448 HTTP packets in the consolidated input dataset. Each of the comparison systems was executed individually, and then their outputs were made available to CLIP for inclusion in its meta-analysis in addition to the NIMs described previously. The results of the execution are shown below in Table 5.

System	TP	TN	FP	FN	BM	MK	MCC
HMMPayl	62	84342	0	44	0.5849	0.9995	.7645
LHMMP	70	83617	725	36	0.6518	0.0876	.2390
CLIP, no confidence values used	71	84342	0	35	0.6698	0.9996	.8183
CLIP, weighted with confidence values	64.83	84324.88	0	24.83	0.7231	0.9997	.8502

Table 5 – Performance Comparison on DARPA/McPAD data set pair

As the data contained in Table 5 above shows, the CLIP system gains a substantial improvement over the MCC values achieved in the comparison systems. Of particular note is the improvement with the inclusion of confidence values, which raises performance of CLIP by approximately 4% over the naïve approach. This difference is attributed to the fact that the average confidence across all correct predictions was 99.9%, while the confidence across the incorrect predictions averaged only 70.9%. A subset of specific false negatives from the CLIP system with confidence values included are shown below in Table 6.

Packet Number	Type	Confidence Level
484	FN	0.430
416	FN	0.487
…		
657	FN	0.961
21	FN	0.9998

Table 6 – Sample Confidence Levels reported by CLIP

The difference in confidence between the correct and incorrect predictions accounts for the positive change in the MCC between the systems with and without the confidence values included in the calculation. The gap in confidence levels also shows a clear "tier-ing" of predictions that would be very useful for an analyst tasked with prioritizing and responding to alert. Also of note, all of the WIM (non-native) input systems were treated as having a confidence level of 100%, and the design of CLIP was

clearly resilient to the meaningless input values in those cases as it is outperformed those systems by a substantial amount.

Comparison of Results – Custom Web Application Intrusion Dataset

Since the DARPA and McPAD datasets are both rather dated, this research also produced a custom web application intrusion dataset for the purposes of testing the proposed system performance against other state-of-the-art systems using current tools and techniques. The HMMPayl, LHMMP, and CLIP systems were all run with the custom web application intrusion dataset, first with a known clean dataset then with a second set that contained attacks injected by the researcher. The results of the execution are shown below in Table 7. Of note, the older HMMPayl system had several issues with connecting some of the HTTP streams and, thus, reported them as separate streams, but the CLIP system's results were extrapolated by the researcher from per-stream to per-packet level results to allow direct comparison. Additionally, the LHMMP system's output was not reported because the design of its layers (specifically, the request/response layer) make it unable to operate meaningfully if the training data does not include traffic from the site that the attack data is from.

System	TP	TN	FP	FN	BM	MK	MCC
HMMPayl	6834	6509	3595	183	0.6181	0.6279	.6230
CLIP, no confidence values included	6390	9681	423	627	0.8688	0.8771	.8729
CLIP, with confidence values included	6212.65	9310.25	295.23	308.32	0.9220	0.9226	.9223

Table 7 – Custom Dataset Results

Once again, the CLIP system gains meaningful improvement over the MCC values achieved in the comparison system, especially with the consideration of the

confidence values in the output of the system. Additionally, both the naïve and the confidence-value aware versions of CLIP far outperformed the comparison HMMPayl system.

The performance of CLIP was slightly worse than HMMPayl in terms of true positives, largely due to the fact that HMMPayl excels at matching very specific patterns and thus was better able to identify some attacks that CLIP did not. However, the trade off in terms of additional true positives for HMMPayl came at a cost of more than eight times the number of false positives, an extremely high count. Specific investigation of the packets where CLIP performed worse than HMMPayl indicates that many of them are cases where one of the subclassifiers provided a very high confidence, but incorrect, "not attack" result. The majority of them were attacks that were very small changes from other non-attack data, such as the insertion of single additional characters attempting to test delimiter processing at the server. In the scope of HTTP vulnerabilities, these likely represent a lower risk, because they are designed to be focused more on detecting weakness than achieving compromise. In the event one of these caused an error condition on the server, a would-be attacker would then follow it up with a request that included some sort of command to inject or other indicator that the system would be more likely to catch. This also demonstrates the value that factoring in risk in a calculation regarding response priority would have to the overall process, and is the basis of a recommendation on future research below.

Limitations

The focus of this research, production of useful confidence measurements for predictions being made, is a new area within intrusion detection research. During the

course of the experiments outlined in this research, a number of limitations arose that need to be considered for understanding the current effort. Addressing these limitations is also used below as the basis for recommendations on further research.

As is generally the case in anomaly-based research, the performance of the system is dependent on the quality of the data used for training and the similarity of the training data and the data that will ultimately be used for testing. The CLIP system is uniquely positioned to deal with this challenge though because of its ability to signal the reliability of its prediction output. Optimally, CLIP would recognize that a particular set of input (whether attack or normal packet) is unlike anything which had been previously seen, and the system should output a lower confidence value because of that. This capability has been demonstrated in the current research by the improvement of the statistics when the predictions are weighted based on the confidence values, but there are likely more explicit ways to handle this would could yield even better results than those demonstrated here.

The current implementation of CLIP is also designed such that the neural network is trained with a maximum number of possible input modules in mind. This was sufficient for demonstrating the viability of the approach described in this research, but it also makes the implementation rather limited in terms of production use. A production-ready version of this system would need to be able to accept output from an arbitrary number of modules and produce both a classification as well as a confidence level prediction. The ability to include an arbitrary number of inputs would increase the usefulness of the system by allowing good predictions to be made about a range of possible attacks.

Summary of Results

This chapter provided details and analysis of the research using several data sources to test the ability of the CLIP system to output a meaningful and relevant confidence level value for predictions being made. The work performed included the design of two native intrusion modules capable of outputting individual confidence level values for their predictions along with the creation of the CLIP system, the meta-analyzer that determines how to translate the intrusion module-produced confidence values into meaningful output.

The research documented results using a total of three datasets, two chosen specifically for comparability with prior systems and one designed to test the viability of the approach on modern web-based attacks. The system employed neural networks and hidden Markov models for the inner native intrusion modules. The meta-analyzer was built using a neural network capable of ingesting output from external sources with either explicit confidence levels or no confidence level specified. Most importantly, this chapter provided evidence that clearly demonstrates the ability to improve the overall prediction ability of the system by including the confidence level values within the system, which provides evidence that the improvement is not simply due to the improved functionality of the NIMs built for this work. Additionally, the ability to ingest and utilize systems that have no concept of confidence level as part of the process is meaningful, because it opens the door to significant expansion of capabilities by expanded collaboration with tools that specialize in particular attack types.

Chapter 5

Conclusions, Implications, Recommendations, and Summary

Conclusions

Intrusion detection continues to face a number of significant problems in real world environments including the volume of data being processed, increased ability and knowledge of attackers, and the finite resources available to respond to alerts. The work conducted in this research demonstrates that confidence levels, when applied correctly, can increase the relative value of predictions being made by the system by providing additional metadata that is useful for prioritization of alerts being generated by the system. Both neural networks and hidden Markov models have been applied effectively for generating meaningful confidence levels.

This research was divided into three major phases. First, two previously proposed intrusion detection systems were obtained and modified to allow the necessary intermediate outputs for inclusion into the CLIP system. The original versions of these systems were then used as points of comparison for the current research, with performance measured using Bookmaker Informedness, Markedness, and Matthews Correlation Coefficient (Powers, 2011). Second, this research created two modules that explicitly output confidence values in the predictions to demonstrate the value of the confidence level data. Third, the meta-analyzer was built to ingest the output produced

by the other four systems and translate it to a final determination of both type of packet and confidence level in the prediction.

Implications

This research expands on the state-of-the-art in intrusion detection with the advent of an explicit confidence level that provides useful information about the predictions being made. While most research in intrusion detection to date has focused on optimizing for a single metric such as false positive rate, precision, accuracy, etc., this work presents a case for a more holistic approach to intrusion detection, one that includes explicitly in its goals the need to provide useful information to a responder beyond a binary “attack”/”no attack” classification. The additional data provided by the confidence level could be the difference between an analyst wasting resources on a false positive and successfully stopping an attack in progress.

Also of note, while some recent work in intrusion detection has been done on ensemble systems, prior efforts have used either naïve combinations of predictions or a second layer of machine learning informed by nothing other than the core predictions. This research demonstrated that the use of confidence levels also improves the results over basic methods of ensemble prediction consolidation, as it can take into account the input confidence levels from the systems that supply them. Although this is not helpful in the case of wrapped (non-native) modules, any module that supplies even a semi-accurate confidence level is providing meaningful additional data that may be used in the combination process. Also of importance, the meta-analyzer layer is able to determine via its own training process which classifiers produce useful confidence levels and the

extent to which the supplied confidence values should be included in the decision making process.

Recommendations

There are a number of opportunities to improve on the results being produced by the CLIP system in future research. This list of recommendations is not intended to be comprehensive, but is meant to provide guidance based on information gleaned during the creation of testing of the system.

Although the confidence level prediction is the core value of the CLIP system, the system as it currently exists makes no explicit recommendation to the user based on a given confidence level, it merely provides the confidence level information. Although some of the confidence levels produced by the system are so low that it is clear the related alerts are suspect, more work could be done around determining the relative meaning of different confidence levels and assigning a text-based tag to them based on a scale. As an example of this, one of the confidence levels produced on a false positive in testing was 0.24, while the other false positive in that test had a confidence level of 0.91. Clearly, there is a significant difference in that case in how strongly the system believed in its prediction, but in the case of a narrower gap, 0.85 to 0.91 for instance, there would be value in providing the analyst with more guidance on what the relative difference represents. It is also feasible that there would be value to discarding all alerts with a confidence level lower than *X*, but again, further work would be needed to understand the thresholds where data is meaningful.

Another significant opportunity for improvement in the system would be the inclusion of relative risk as a component of the calculation process. For instance, if the

CLIP system was 99% confident that a ping sweep is in progress and only 60% confident that a SQL injection attack was in progress, naïve ordering by confidence level would indicate that the ping sweep should be reviewed first, while in reality, the SQL injection attack is far more likely to yield near-term negative results. The inclusion of risk would require substantial additional effort to define and rank attacks in terms of severity, but would be a meaningful and useful input into the process.

Although the native intrusion modules built for this study and the wrapped intrusion modules selected for inclusion in this work proved relatively effective for the comparison done in this research, there are numerous other approaches that could prove useful and valuable additions to the process. Ultimately, any system for which one can devise a method of calculating a useful confidence level would be worth considering for addition. Since the core of the system, the meta-analyzer, is a controller designed to operate and aggregate any number of available models, the system could be readily expanded to include models tuned specifically to different types of attacks. The addition of extremely targeted native modules that self-report low confidence on all packets they are not designed to identify but report high confidence in the very small subset of events that they are built to target could also prove to be a valuable addition to the system's capabilities. In a production implementation, it would also be desirable for the system to be able to bring new models online as needed to address and train on new attack types and train these using online training approaches as the system continues operation (Chis & Harrison, 2015), and the choice of the neural network for the meta analyzer would lend itself to that readily. That capability would also operate most efficiently if the system

were able to load and unload modules in real-time, unloading ones that prove to be consistently ineffective and loading alternatives in their place.

Another significant challenge for this research was the gathering and creation of appropriate data for testing. Although live network data clearly represents the best picture of "actual" real world data, it also is the most difficult to use in practice because of issues with privacy, confidentiality, and even the ability to label the dataset with surety. While artificially generated data is generally more available, virtually every significant simulated data set produced to date has had problems identified that make it questionable for use in comparison purposes. The production of robust data sets would be a very useful area for future research that would have positive impacts on the research that is being done in the field of intrusion detection.

Summary

Intrusion detection continues to improve both in the research and in commercial applications, but there are still significant shortcomings to current approaches. The focus on driving improvement in one dimensional metrics continues to yield small incremental improvements, but real-world experiences have shown that it does not scale well. A 0.00001 false positive rate would still yield 10,000 false positives per day in a 1 billion packets per day hyper-scale environment such as Amazon's AWS (Mueller, 2016), and the amount of data flowing across networks is only expected to increase as more and more devices and people join the internet.

This research was conceived based on the thought process of an analyst who is tasked with responding to alerts being produced by intrusion detection systems. In most environments, the system produces more alerts than can be reasonably triaged by the

available staff. To address that, this research has endeavored to propose a way to provide additional data to an analyst that allows them more context in prioritizing their response procedures when resources become scarce. The confidence level measurements proposed and implemented in this research are a step in that direction, a successful attempt to provide additional meta-data that is useful in the response process.

The CLIP system was tested against multiple datasets including both classic comparison sets as well as a custom dataset designed to highlight a number of modern HTTP attacks including command injection and cross-site scripting. In all cases, the CLIP system was shown to outperform the comparison systems both with and without the confidence level being included in the process. Just as important, CLIP with the confidence level improved over the naïve version without access to the confidence level, demonstrating that the confidence values being produced were meaningful in terms of distinguishing a good prediction from a suspect one. The CLIP system has expanded on the state-of-the-art, and has demonstrated the potential value of explicit confidence levels in intrusion detection.

References

Abdullah, N., Xu, Y., & Geva, S. (2011). *Integrating Fusion Techniques into the Collaborative Filtering Search-Based Recommender Systems.* Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Volume 03.

Abraham, A., Jain, R., Thomas, J., & Han, S. Y. (2007). D-SCIDS: Distributed soft computing intrusion detection system. *Journal of Network and Computer Applications, 30*(1), 81-98.

Ahmad, I., Hussain, M., Alghamdi, A., & Alelaiwi, A. (2014). Enhancing SVM performance in intrusion detection using optimal feature subset selection based on genetic principal components. *Neural Computing and Applications, 24*(7-8), 1671-1682.

Ahmadi, M., Biggio, B., Arzt, S., Ariu, D., & Giacinto, G. (2016). *Detecting Misuse of Google Cloud Messaging in Android Badware*. Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices, Vienna, Austria.

Akyaz, U., & Uyar, A. S. (2010). *Detection of DDoS attacks via an artificial immune system-inspired multiobjective evolutionary algorithm.* Proceedings of the 2010 international conference on Applications of Evolutionary Computation - Volume Part II, Istanbul, Turkey.

Alserhani, F., Akhlaq, M., Awan, I. U., Cullen, A. J., & Mirchandani, P. (2010, 20-23 April 2010). *MARS: Multi-stage Attack Recognition System.* Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications.

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

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

Azzouzi, A. E., & Kadiri, K. E. E. (2015). Semantic System for Attacks and Intrusions Detection. *International Journal of Digital Crime and Forensics, 7*(4), 19-32.

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.

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

Brooks, R. R., Schwier, J. M., & Griffin, C. (2009). Behavior detection using confidence intervals of hidden Markov models. *IEEE Transactions on Systems, Man, and Cybernetics, 39*(6), 1484-1492.

Carvalho, M., & Perez, C. (2011). *An evolutionary multi-agent approach to anomaly detection and cyber defense*. Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research, Oak Ridge, Tennessee.

Chandrashekar, A. M., & Raghuveer, K. (2012). *Fusion of multiple data mining techniques for effective network intrusion detection: a contemporary approach*. Proceedings of the Fifth International Conference on Security of Information and Networks, Jaipur, India.

Chapke, P. P., & Deshmukh, R. R. (2015). *Intrusion Detection System using Fuzzy Logic and Data Mining Technique*. Proceedings of the 2015 International Conference on Advanced Research in Computer Science Engineering & Technology, Unnao, India.

Chebrolu, S., Abraham, A., & Thomas, J. P. (2004). *Hybrid feature selection for modeling intrusion detection systems.* Neural Information Processing.

Chebrolu, S., Abraham, A., & Thomas, J. P. (2005). Feature deduction and ensemble design of intrusion detection systems. *Computers & Security, 24*(4), 295-307.

Chen, Z.-G., & Kim, S.-R. (2013). *Combining principal component analysis, decision tree and naive Bayesian algorithm for adaptive intrusion detection*. Proceedings of the 2013 Research in Adaptive and Convergent Systems, Montreal, Quebec, Canada.

Chis, T., & Harrison, P. G. (2015). Adapting Hidden Markov Models for Online Learning. *Electronic Notes in Theoretical Computer Science, 318*, 109-127.

Choudhary, A. K., & Swarup, A. (2009). *Neural network approach for intrusion detection*. Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, Seoul, Korea.

Cipriano, C., Zand, A., Houmansadr, A., Kruegel, C., & Vigna, G. (2011). *Nexat: a history-based approach to predict attacker actions*. Proceedings of the 27th Annual Computer Security Applications Conference, Orlando, Florida.

Cohen, W. W., & Singer, Y. (1999). *A simple, fast, and effective rule learner*. Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence, Orlando, Florida, USA.

Corona, I., Ariu, D., & Giacinto, G. (2009, 14-18 June 2009). *HMM-Web: A Framework for the Detection of Attacks Against Web Applications.* Proceedings of the IEEE International Conference on Communications, Dresden, Germany.

Cuong, T. D., & Giang, N. L. (2012). *Intrusion detection under covariate shift using modified support vector machine and modified backpropagation*. Proceedings of the Third Symposium on Information and Communication Technology, Ha Long, Vietnam.

Darwiche, A. (2003). A differential approach to inference in Bayesian networks. *Journal of the ACM, 50*(3), 280-305.

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

Elfeshawy, N. A., & Faragallah, O. S. (2013). Divided two-part adaptive intrusion detection system. *Wireless Networks, 19*(3), 301-321.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee. (1999). Hypertext Transfer Protocol -- HTTP/1.1 *RFC 2616*.

Folino, G., & Sabatino, P. (2016). Ensemble based collaborative and distributed intrusion detection systems. *Journal of Network and Computer Applications, 66*(C), 1-16.

Fortnow, L., & Vohra, R. V. (2008). The complexity of forecast testing. *SIGecom Exchange, 7*(3), 1-5.

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

Gao, X.-G., Mei, J.-F., Chen, H.-Y., & Chen, D.-Q. (2014). Approximate inference for dynamic Bayesian networks: sliding window approach. *Applied Intelligence, 40*(4), 575-591.

Geiger, D., & Heckerman, D. (2002). Parameter priors for directed acyclic graphical models and the characterization of several probability distributions. *The Annals of Statistics*, 1412-1440.

Group, L. N. R. (2009). LibPcap. Retrieved from https://ee.lbl.gov/

Grover, A., Kapoor, A., & Horvitz, E. (2015). *A Deep Hybrid Model for Weather Forecasting*. Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia.

Gu, G., Fogla, P., Dagon, D., Lee, W., & Skoric, B. (2006). *Measuring intrusion detection capability: an information-theoretic approach*. Proceedings of the 2006 ACM Symposium on Information, computer and communications security, Taipei, Taiwan.

Hai, N., Franke, K., & Petrovic, S. (2010, 15-18 Feb. 2010). *Improving Effectiveness of Intrusion Detection by Correlation Feature Selection.* Availability, Reliability, and Security, 2010. ARES '10 International Conference on.

Heberlein, L. T., Dias, G. V., Levitt, K. N., Mukherjee, B., Wood, J., & Wolber, D. (1990, 7-9 May 1990). *A network security monitor.* Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, California, USA.

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

Javed, M., & Paxson, V. (2013). *Detecting stealthy, distributed SSH brute-forcing*. Proceedings of the 2013 ACM SIGSAC conference on Computer & Communications Security, Berlin, Germany.

Jeong, H., Yoo, Y., Yi, K. M., & Choi, J. Y. (2014). Two-stage online inference model for traffic pattern analysis and anomaly detection. *Machine Vision and Applications, 25*(6), 1501-1517.

Jiang, L. (2012). Learning Instance Weighted Naive Bayes from labeled and unlabeled data. *Journal of Intelligent Information Systems, 38*(1), 257-268.

Karamcheti, V., Geiger, D., Kedem, Z., & Muthukrishnan, S. (2005). *Detecting malicious network traffic using inverse distributions of packet contents*. Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data, Philadelphia, Pennsylvania, USA.

Katipally, R., Gasior, W., Cui, X., & Yang, L. (2010). *Multistage attack detection system for network administrators using data mining*. Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research, Oak Ridge, Tennessee.

Katipally, R., Yang, L., & Liu, A. (2011). *Attacker behavior analysis in multi-stage attack detection system*. Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research, Oak Ridge, Tennessee.

Khanna, R., & Liu, H. (2006). *System approach to intrusion detection using hidden Markov model*. Proceedings of the 2006 international conference on Wireless communications and mobile computing, Vancouver, British Columbia, Canada.

Kim, G., Lee, S., & Kim, S. (2014). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Syst. Appl., 41*(4), 1690-1700.

Kim, S. H., Wang, Q.-H., & Ullrich, J. B. (2012). A comparative study of cyberattacks. *Communications of the ACM, 55*(3), 66-73.

Laboratory, M. L. (2013). MIT Lincoln Laboratory: Communication Systems and Cyber Security: Cyber Systems and Technology: DARPA Intrusion Detection Evaluation. Retrieved October 16th 2013, from

http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/1999data.html

Lelieveld, J., Kunkel, D., & Lawrence, M. G. (2012). Global risk of radioactive fallout after major nuclear reactor accidents. *Atmospheric Chemistry and Physics, 12*(9), 4245-4258.

Leutbecher, M., & Palmer, T. N. (2008). Ensemble forecasting. *Journal of Computational Physics, 227*(7), 3515-3539.

Lippmann, R. P., Fried, D. J., Graf, I., Haines, J. W., Kendall, K. R., McClung, D., . . . Zissman, M. A. (2000, 2000). *Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation.* Proceedings of the DARPA Information Survivability Conference and Exposition, Hilton Head, South Carolina.

Lowd, D., & Domingos, P. (2005). *Naive Bayes models for probability estimation*. Proceedings of the 22nd international conference on Machine learning, Bonn, Germany.

Lu, C., & Brooks, R. (2011). *Botnet traffic detection using hidden Markov models*. Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research, Oak Ridge, Tennessee.

Luo, Y.-X. (2010). *The Research of Bayesian Classifier Algorithms in Intrusion Detection System*. Proceedings of the 2010 International Conference on E-Business and E-Government, Guangzhou, China.

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

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta, 405*(2), 442-451.

Mehetrey, P., Shahriari, B., & Moh, M. (2016, Oct. 31 2016-Nov. 4 2016). *Collaborative Ensemble-Learning Based Intrusion Detection Systems for Clouds.* 2016 International Conference on Collaboration Technologies and Systems (CTS).

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, 48*(1), 1-41.

Mitchell, R., & Chen, I.-R. (2014). A survey of intrusion detection techniques for cyber-physical systems. *ACM Computing Surveys, 46*(4), 1-29.

Mueller, S. (2016). *Another Day, Another Billion Packets*. Presentation presented at AWS re:Invent 2016, Las Vegas, NV.

Mukkamala, S., Janoski, G., & Sung, A. (2002). *Intrusion detection using neural networks and support vector machines.* Proceedings of the 2002 International Joint Conference on Neural Networks, Honolulu, Hawaii, USA.

Myers, C., Singer, A., Shin, F., & Church, E. (1992, 23-26 Mar 1992). *Modeling chaotic systems with hidden Markov models.* Proceedings of the 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, San Francisco, California, USA.

Nadiammai, G. V., & Hemalatha, M. (2012). *An evaluation of clustering technique over intrusion detection system*. Proceedings of the International Conference on Advances in Computing, Communications and Informatics, Chennai, India.

Naoum, R. S., Abid, N. A., & Al-sultani, Z. N. (2013). An Enhanced Resilient Backpropagation Artificial Neural Network for Intrusion Detection System. *International Journal of Computer Science and Network Security, 13*(3), 98-104.

Ning, P., & Xu, D. (2004). Hypothesizing and reasoning about attacks missed by intrusion detection systems. *ACM Transactions on Information and System Security, 7*(4), 591-627.

Ourston, D., Matzner, S., Stump, W., & Hopkins, B. (2003). *Applications of Hidden Markov Models to Detecting Multi-Stage Network Attacks*. Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9 - Volume 9.

Palmer, T. N. (2000). Predicting uncertainty in forecasts of weather and climate. *Reports on Progress in Physics, 63*(2), 71.

Pan, S., Morris, T. H., Adhikari, U., & Madani, V. (2013). *Causal event graphs cyber-physical system intrusion detection system*. Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop, Oak Ridge, Tennessee, USA.

Papadogiannakis, A., Polychronakis, M., & Markatos, E. P. (2010). *Improving the accuracy of network intrusion detection systems under load using selective packet discarding*. Proceedings of the Third European Workshop on System Security, Paris, France.

Pillai, M. M., Eloff, J. H. P., & Venter, H. S. (2004). *An approach to implement a network intrusion detection system using genetic algorithms*. Proceedings of the 2004 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries, Stellenbosch, Western Cape, South Africa.

Popper, K. R. (1988). *The open universe: An argument for indeterminism* (Vol. 2): Psychology Press.

Powers, D. M. W. (2011). Evaluation: From precision, recall and f-measure to ROC, informedness, markedness & correlation. *Journal of Machine Learning Technologies, 2*(1), 37-63.

Rabier, F., Klinker, E., Courtier, P., & Hollingsworth, A. (1996). Sensitivity of forecast errors to initial conditions. *Quarterly Journal of the Royal Meteorological Society, 122*(529), 121-150.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE, 77*(2), 257-286.

Rabiner, L., & Juang, B. H. (1986). An introduction to hidden Markov models. *IEEE ASSP Magazine, 3*(1), 4-16.

Reiser, H. P. (2017). *Towards intrusion-resilient security monitoring in multi-cloud infrastructures*. Proceedings of the 1st International Workshop on Security and Dependability of Multi-Domain Infrastructures, Belgrade, Serbia.

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

Roulston, M. S. (2005). A comparison of predictors of the error of weather forecasts. *Nonlinear Processes in Geophysics, 12*(6), 1021-1032.

Saha, S., Sairam, A. S., Yadav, A., & Ekbal, A. (2012). *Genetic algorithm combined with support vector machine for building an intrusion detection system*. Proceedings of the International Conference on Advances in Computing, Communications and Informatics, Chennai, India.

Scherer, S., Glodek, M., Schwenker, F., Campbell, N., & Palm, G. (2012). Spotting laughter in natural multiparty conversations: A comparison of automatic online and offline approaches using audiovisual data. *ACM Transactions on Interactive Intelligent Systems, 2*(1), 1-31.

Sekar, R., Guang, Y., Verma, S., & Shanbhag, T. (1999). *A high-performance network intrusion detection system*. Proceedings of the 6th ACM conference on Computer and communications security, Kent Ridge Digital Labs, Singapore.

Sen, N., Sen, R., & Chattopadhyay, M. (2014, 14-16 Nov. 2014). *An Effective Back Propagation Neural Network Architecture for the Development of an Efficient Anomaly Based Intrusion Detection System.* 2014 International Conference on Computational Intelligence and Communication Networks (CICN), Bhopal, India.

Sen, R., Chattopadhyay, M., & Sen, N. (2015). *An Efficient Approach to Develop an Intrusion Detection System Based on Multi Layer Backpropagation Neural Network Algorithm: IDS using BPNN Algorithm*. Proceedings of the 2015 ACM SIGMIS Conference on Computers and People Research, Newport Beach, California, USA.

Sen, S. (2014). Using instance-weighted naive Bayes for adapting concept drift in masquerade detection. *International Journal of Information Security, 13*(6), 583-590.

Shalizi, C. R., & Crutchfield, J. P. (2001). Computational Mechanics: Pattern and Prediction, Structure and Simplicity. *Journal of Statistical Physics, 104*(3-4), 817-879.

Shalizi, C. R., & Shalizi, K. L. (2004). *Blind construction of optimal nonlinear recursive predictors for discrete sequences*. Proceedings of the 20th conference on Uncertainty in artificial intelligence, Banff, Canada.

Shameli-Sendi, A., Cheriet, M., & Hamou-Lhadj, A. (2014). Taxonomy of intrusion risk assessment and response system. *Computers & Security, 45*, 1-16.

Sharma, N., & Mukherjee, S. (2012). *Layered approach for intrusion detection using naive Bayes classifier*. Proceedings of the International Conference on Advances in Computing, Communications and Informatics, Chennai, India.

Smaha, S. E. (1988, 12-16 Dec 1988). *Haystack: an intrusion detection system.* Proceedings of the Fourth Aerospace Computer Security Applications Conference, Orlando, Florida, USA.

Smith, F. J., Ming, J., O'Boyle, P., & Irvine, A. D. (1995, 9-12 May 1995). *A hidden Markov model with optimized inter-frame dependence.* Proceedings of the 1995 International Conference on Acoustics, Speech, and Signal Processing, Detroit, Michigan, USA.

Smith, R., Japkowicz, N., Dondo, M., & Mason, P. (2008). *Using unsupervised learning for network alert correlation*. Proceedings of the Canadian Society for computational studies of intelligence, 21st conference on Advances in artificial intelligence, Windsor, Canada.

Soleimani, M., & Ghorbani, A. A. (2012). Multi-layer episode filtering for the multi-step attack detection. *Computer Communincations, 35*(11), 1368-1379.

Sommer, R., & Paxson, V. (2010, 16-19 May 2010). *Outside the Closed World: On Using Machine Learning for Network Intrusion Detection.* Proceedings of the 2010 IEEE Symposium on Security and Privacy (SP), Oakland, California, USA.

Staudemeyer, R. C., & Omlin, C. W. (2013). *Evaluating performance of long short-term memory recurrent neural networks on intrusion detection data*. Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference, East London, South Africa.

Stoll, P. A., & Jun, O. (1995, 5-7 Jul 1995). *Applications of HMM modeling to recognizing human gestures in image sequences for a man-machine interface.* Proceedings of the 4th IEEE International Workshop on Robot and Human Communication, Tokyo, Japan.

Sujatha, P. K., Priya, C. S., & Kannan, A. (2012). *Network intrusion detection system using genetic network programming with support vector machine*. Proceedings of

the International Conference on Advances in Computing, Communications and Informatics, Chennai, India.

Swarnkar, M., & Hubballi, N. (2016). OCPAD. *Expert Systems with Applications, 64*(C), 330-339.

Taheri, S., Mammadov, M., & Bagirov, A. M. (2011). *Improving naive Bayes classifier using conditional probabilities*. Proceedings of the Ninth Australasian Data Mining Conference - Volume 121, Ballarat, Australia.

Taub, L. (2013). *Application of a Layered Hidden Markov Model in the Detection of Network Attacks.* (Doctoral dissertation), Nova Southeastern University. ProQuest database.

Tennekes, H. (1992). Karl Popper and the accountability of numerical weather forecasting. *Weather, 47*(9), 343-346.

Tennekes, H., Baede, A. P. M., & Opsteegh, J. D. (1986). *Forecasting forecast skill*. Workshop on Predictability in the Medium and Extended Range, Shinfield Park, Reading.

Torrano-Gimenez, C., Nguyen, H. T., Alvarez, G., & Franke, K. (2015). Combining expert knowledge with automatic feature extraction for reliable web attack detection. *Security and Communication Networks, 8*(16), 2750-2767.

University of California, I., Information and Computer Science. (1999). KDD Cup 1999 Data. Retrieved February 23, 2014, from https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

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

Wang, L., Liu, A., & Jajodia, S. (2006). Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. *Computer Communications, 29*(15), 2917-2933.

Weir, J. (2012). Building a Debian\Snort based IDS. Retrieved December 1, 2013, from http://www.snort.org/docs

Wiggers, P., Mertens, B., & Rothkrantz, L. (2011). *Dynamic Bayesian networks for situational awareness in the presence of noisy data*. Proceedings of the 12th International Conference on Computer Systems and Technologies, Vienna, Austria.

Wilson, A., Fern, A., & Tadepalli, P. (2010). *Bayesian role discovery for multi-agent reinforcement learning*. Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1, Toronto, Canada.

Wressnegger, C., Schwenk, G., Arp, D., & Rieck, K. (2013). *A close look on n-grams in intrusion detection: anomaly detection vs. classification*. Proceedings of the 2013 ACM workshop on Artificial intelligence and security, Berlin, Germany.

Xiang, J., Westerlund, M., Sovilj, D., & Pulkkis, G. (2014). *Using extreme learning machine for intrusion detection in a big data environment*. Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop, Scottsdale, Arizona, USA.

Xie, L., Zhang, X., Seifert, J.-P., & Zhu, S. (2010). *pBMDS: a behavior-based malware detection system for cellphone devices*. Proceedings of the third ACM conference on Wireless network security, Hoboken, New Jersey, USA.

Yolacan, E. N., Dy, J. G., & Kaeli, D. R. (2014). *System Call Anomaly Detection Using Multi-HMMs*. Proceedings of the 2014 IEEE Eighth International Conference on Software Security and Reliability-Companion.

Yu, Z., Tsai, J. J. P., & Weigert, T. (2008). An adaptive automatically tuning intrusion detection system. *ACM Trans. Auton. Adapt. Syst., 3*(3), 1-25.

Zadeh, L. A. (1965). Fuzzy sets. *Information and control, 8*(3), 338-353.

Zaman, S., El-Abed, M., & Karray, F. (2013). *Features selection approaches for intrusion detection systems based on evolution algorithms*. Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication, Kota Kinabalu, Malaysia.

Zargar, S. T., Takabi, H., & Joshi, J. B. D. (2011, 15-18 Oct. 2011). *DCDIDP: A distributed, collaborative, and data-driven intrusion detection and prevention framework for cloud computing environments.* Proceedings of the 7th

International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), Orlando, Florida, USA.

Zarnani, A., & Musilek, P. (2013). *Learning uncertainty models from weather forecast performance databases using quantile regression*. Proceedings of the 25th International Conference on Scientific and Statistical Database Management, Baltimore, Maryland, USA.

Zhang, Y., & Rockett, P. I. (2009). A generic multi-dimensional feature extraction method using multiobjective genetic programming. *Evolutionary Computation, 17*(1), 89-115.

Zhang, Z., & Krishnamurti, T. N. (1997). Ensemble forecasting of hurricane tracks. *Bulletin of the American Meteorological Society, 78*(12), 2785-2795.

Zimmermann, H.-J. (2001). *Fuzzy set theory—and its applications*: Springer Science & Business Media.