The Graduate Institute of Sciences and Engineering

M.S. Thesis in Electrical and Computer Engineering

MANAGING THE HUMAN FACTORS IN INFORMATION SECURITY THROUGH COMPUTATIONAL INTELLIGENCE METHODS

by

Murat OĞUZ

June 2014
Kayseri, Turkey

MANAGING THE HUMAN FACTORS IN INFORMATION SECURITY THROUGH COMPUTATIONAL INTELLIGENCE METHODS

by

Murat OĞUZ

A thesis submitted to

the Graduate Institute of Sciences and Engineering

of

Meliksah University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical and Computer Engineering

June 2014
Kayseri, Turkey

APPROVAL PAGE

This is to certify that I have read the thesis entitled "Managing the Human Factors in Information Security through Computational Intelligence Methods" by Murat OGUZ and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Electrical and Computer Engineering, the Graduate Institute of Science and Engineering, Melikşah University.

_____

June 9, 2014 Assoc. Prof. Dr. İ. Ömür BUCAK

Supervisor

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

June 9, 2014 Prof. Dr. Murat UZAM

Head of Department

Examining Committee Members

Title and Name		Approved
Assoc. Prof. Dr. İ. Ömür BUCAK	June 17, 2014	_____
Assoc. Prof. Dr. Ahmet UYAR	June 17, 2014	_____
Assist. Prof. Dr. Hasan PALTA	June 17, 2014	_____

It is approved that this thesis has been written in compliance with the formatting rules laid down by the Graduate Institute of Science and Engineering.

_____

Prof. Dr. M. Halidun KELEŞTEMUR
Director

June 2014

ABSTRACT

MANAGING THE HUMAN FACTORS IN INFORMATION SECURITY THROUGH COMPUTATIONAL INTELLIGENCE METHODS

Murat OĞUZ

M.S. Thesis - Electrical and Computer Engineering
June 2014

Supervisor: Assoc. Prof. Dr. İhsan Ömür BUCAK

ABSTRACT

Humans are consistently referred to as the weakest link in information security. Human factors play a significant role in information security; factors such as individual differences, cognitive abilities and personality traits can impact on behavior. The purpose of this thesis is to identify, describe and classify the human factors affecting Information Security and develop a model to reduce the risk of insider misuse and assess the use and performance of the best-suited artificial intelligence techniques in detection of misuse. More specifically, this study provides the following: a comprehensive view of the human related information security risks and threats, classification study of the human related threats in information security, a methodology developed to reduce the risk of human related threats by detecting insider misuse by behavior-based intrusion detection systems through the use of artificial intelligence techniques, and finally the comparison of the numerical experiments for analysis of this approach.

Keywords: Human Factors in Information Security, Artificial Intelligence, Intrusion Detection Systems.

ÖZ

BİLGİ GÜVENLİĞİNDE İNSAN FAKTÖRLERİNİ SAYISAL ZEKA YÖNTEMLERİYLE YÖNETME

Murat OĞUZ

Yüksek Lisans Tezi – Elektrik ve Bilgisayar Mühendisliği
Haziran 2014

Tez Yöneticisi: Doç. Dr. İhsan Ömür BUCAK

ÖZ

Bilgi güvenliğinde en zayıf halkanın insan olduğu kabul edilmektedir. Bireysel farklılıklar, kişisel özellikler ve kavrama yeteneği gibi faktörler, insan davranışlarını etkilemektedir ve insan davranışlarını bilgi güvenliğinde önemli bir rol oynamaktadır. Bu tez çalışmasının amacı, bilgi güvenliğini etkileyen insan faktörlerini tanımlamak, belirlemek, sınıflandırmak ve sonrasında kötüye kullanımı azaltmak ve önlemek için yapay zekâ tekniklerini kullanan bir model geliştirmektir. Bu çalışma, bilgi güvenliği riskleri ve insan kaynaklı güvenlik tehditlerini kapsamlı bir bakış, bilgi güvenliğinde insan kaynaklı tehditlerin sınıflandırma çalışması, insan kaynaklı risklerin yapay zeka teknikleri kullanan saldırı tespit sistemleri ile azaltılması için bir yaklaşım ve bu yaklaşımın analizi için deneysel çalışma sonuçlarının karşılaştırılmasını ihtiva etmektedir.

Anahtar Kelimeler: Bigi Güvenliğinde İnsan Faktörü, Yapay Zeka, Saldırı Tespit Sistemleri.

DEDICATION

To my wife and family for their love and support.

ACKNOWLEDGEMENT

I express sincere appreciation to my thesis supervisor Assoc. Prof. Dr. İhsan Ömür BUCAK for his guidance and insight throughout the research.

Thanks go to the other faculty members, Assoc. Prof. Dr. Ahmet UYAR, Assist. Prof. Dr. Aytekin VARGÜN, and Assist. Prof. Dr. Hasan PALTA for their valuable suggestions and comments.

I express my thanks and appreciation to my family for their understanding, motivation and patience. Lastly, but in no sense the least, I am thankful to all colleagues and friends who made my stay at the university a memorable and valuable experience.

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURE

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Recently published annual reports of enterprise security companies consistently show that employees are very often the cause of the most significant and costly security incidents and this has motivated our work towards the master's thesis [1, 2]. Organizations that value their information need to protect it from threat sources that exploit vulnerabilities in information systems. Although attacks originating from outside threat sources, such as hacking attempts or viruses, have gained a lot of publicity, the more risky attacks come from inside [3, 4].

The insiders can be employees, consultants, contractors, employees of business partner firms and visitors. Insiders are trusted and, therefore, have the necessary access to be able to exploit vulnerabilities more easily. Many forms of technology are available to protect information but this is usually applied to identify and restrict outsider access with the-shelf products such as firewalls and intrusion detection and prevention systems.

Deploying security solutions such as firewalls to combat insider threats will result in complete denying of services to the employees. This is not a possible solution as company needs people to perform various tasks. In order to prevent insider threats from occurring, companies should identify their assets and threats that exist from

insiders. Once threats have been identified, controls could be deployed to address these threats.

In this study we identified, described and classified the human factors affecting Information Security in organizations. We introduced a most up-to-date taxonomy about human threats. We developed a model to reduce the risk of insider misuse and analyzed the performance of machine learning algorithms in detection of misuse. We also measured the detection rates of attack types such as Probe, Denial of Service, User to Root, and Remote to Local, which we defined in our Human Threats Taxonomy.

More specifically, this study provides the following: a comprehensive view of the human related information security risks and threats, classification study of the human related threats in information security, a methodology developed to reduce the risk of human related threats by detecting insider misuse by behavior-based intrusion detection systems through the use of artificial intelligence techniques, and finally the comparison of the numerical experiments for analysis of this approach.

1.2 RESEARCH PROBLEM AND QUESTIONS

Humans are consistently referred to as the weakest link in security [5, 6]. Currently there is a lack of research in human factors in Information Security. An exclusive focus on the technical aspects of security, without due consideration of how the human interacts with the system, is clearly inadequate. None of the previous taxonomies are directly related with detection of human threats, therefore there is a need for a detection-related taxonomy which will be more effective for detection. There is also a lack of studies about applying machine learning algorithms to behavior-based intrusion detection systems. Existing studies compared limited types of machine learning algorithms. Attack types of human threats are not defined and detection rate of attack types is not measured. The purpose of this study is to answer the following questions:

- What types of human behaviors can lead to security breaches?
- Which machine learning algorithm has a better detection rate in anomaly detection?
- What types of attacks are more difficult to detect?

1.3 SCOPE OF THE STUDY

Threats can be posed by different factors. Threats can be caused by nature, the environment and by human. The principal objective of this study is to identify what are the most problematic human factors affecting security in organizations, analyze the performance of detection systems using artificial intelligence techniques, and measure the detection rates of attack types.

CHAPTER 2

BACKGROUND

2.1 INFORMATION SECURITY

According to ISO/IEC 17799 standard: Information is an asset that, like other important business assets, is essential to an organization's business and consequently needs to be suitably protected. This is especially important in the increasingly interconnected business environment. As a result of this increasing interconnectivity, information is now exposed to a growing number and a wider variety of threats and vulnerabilities.

Information can exist in several forms. It can be printed or written on paper, stored electronically, transmitted by post or by using electronic means, shown on films, or spoken in conversation. Whatever form information takes, or means by which it is shared or stored, it should always be appropriately protected.

Modern organizations make use of information systems to store, process and distribute valuable information assets. Information can be defined as data that have been converted into a meaningful and useful context for the receiver [7]. What exact information is considered valuable depends on the organization, but examples are strategic information and intellectual property that give the organization a competitive edge over its competitors.

Information security is a system of procedures and policies designed to protect and control information [8]. Information security is the protection of information from a wide range of threats in order to ensure business continuity, minimize business risk, and maximize return on investments and business opportunities [9].

Failure of security could, for example, lead to unauthorized disclosure, modification, or interruption of information. These three examples relate to three properties of information security, commonly called confidentiality, integrity and availability of information; in addition, other properties, such as authenticity, accountability, non-repudiation, and reliability can also be involved [9].

There is a general consensus as to the meanings of the terms: availability, integrity and confidentiality.

i. Availability: The prevention of unauthorized withholding of information or resources. This does not apply just to personnel withholding information. Information should be as freely available as possible to authorized users.
ii. Integrity: The prevention of erroneous modification of information. Authorized users are probably the biggest cause of errors and omissions and the alteration of data. Storing incorrect data within the system can be as bad as losing data. Malicious attackers also can modify, delete, or corrupt information that is vital to the correct operation of business functions.
iii. Confidentiality: The prevention of unauthorized disclosure of information. This can be the result of poor security measures or information leaks by personnel. An example of poor security measures would be to allow anonymous access to sensitive information.

It can be concluded that organizations that value their information need to safeguard it from threat sources that may also place value on their information in a manner that is contrary to the interest of the organization [10].

2.2 HUMAN FACTOR

People are still the weakest security link [11] 'We need to remind ourselves again and again that information security is not a technology issue – it's a people issue [12]. 'Security is both a feeling and a reality' [13]. It is important to recognize that humans have foibles and weaknesses [14]. All of previous authors emphasize the importance of human factor within information security.

Gardner states that a human is not used to thinking that his feelings are a source of his conscious decisions, but many researches prove that human behavior is affected by cognition and affect [15]. But the human brain is designed with blind spots, not only optical, but also psychological [16]. Even those who have knowledge and skills have blind spots and make errors all the time [14].

An employee can contribute with security related actions every day, and his/her view on information security is built on organizational, technological and individual factors [17]. Information security usually has a lot of tradeoffs and mainly it affects functionality – employees have various limitations to perform their duties.

The main problem is that the ones responsible for information security do not take the thoughts, feelings and behavior of employees into account, Kabay even states that it is common for organization management and people responsible for security not to listen to employees but mainly deal with commanding them [18]. McIlwraith notices the important characteristics of information security practice within organization – it arouses emotions, sometimes even significant negative emotions [14].

Unintelligent countermeasures may result in employees behaving in a way that would negatively affect security, because security solutions are developed to attempt to protect information, but the human factor is often left without attention [19].

2.3 ATTACK TYPES

There are a large variety of different attack types [20]. An attacker may attempt to guess a user's password. An attacker may also monitor the network to obtain the information they require to launch an attack. Sometimes attackers try to put unauthorized programs onto computers that they have access to. Sometimes they may steal information or corrupt information. They may also try to perform a denial of service attack.

A good taxonomy makes it possible to classify individual attacks into groups sharing common properties [21]. One widely used taxonomy [22] divides attacks into four classes: Probes, Denial of Service (DoS), User to Root (U2R) and Remote to Local (R2L). The class Normal that will be used later in this work represents network traffic which is considered attack-free.

In the section below, various attacks are explained. These attacks may be referred to in other parts of this thesis to describe the behavior of the human threats.

2.3.1 Probe

Probe attacks are often the first step of all other attacks that we have seen previously. They are used to gather information about the targeted network or a specific machine on a network. Without network probes, an attacker would have a hard time finding the vulnerabilities present on his target. That is the reason why it is crucial to detect this type of attacks. However, since probing or scanning abuses a perfectly legitimate feature used by network administrators to check on machines on a network, it is also difficult to differentiate attacks from regular actions [22]. Many programs have been developed to scan a network. The most famous is probably "nmap" which is a powerful tool that can be used to look for active machines and active ports on a machine. This information is very valuable because knowing that the port 80 is active, for instance, means that a web server with potential vulnerabilities or misconfigurations runs on the machine. If port 80 is open, the attacker can also conclude that the machine serves its content unencrypted. Nmap is not limited to finding the open ports, it is also possible to discover the type and version of the server or the type and version of the operating system [20]. Other attacks such as "saint" and "satan" are specialized in discovering vulnerabilities in the targeted system. These scanning tools allow even unskilled attackers

to find vulnerabilities automatically on a large number of machines. A typical attack scenario would involve a first phase where the attacker tries to scan the network that he intends to compromise. Thanks to the scan or probe, the attacker will have a complete map of the machines and services running on the network. The next phase is to find vulnerabilities in the services available with automated programs. Once one or more vulnerabilities are found, the attacker will be able to launch another attack depending on his goals and on the vulnerability found [20].

2.3.2 User to Root (U2R)

In a User to Root attack, an attacker starts a session on a computer as a normal user with restricted rights and by exploiting some vulnerability on the software installed on the system, the user can elevate his privilege. The goal of this class of exploits is obviously to obtain administrator rights on the attacked computer in order to have full control of it [22].

There are several different types of U2R attacks. Buffer over flow [23] is certainly the major vulnerability used by hackers when trying to obtain privileged rights on a computer. This implementation bug is found mainly in software written in programming languages such as C or C++ which allow the programmer to manually allocate the memory. Memory allocation can be very powerful when used carefully, but is subject to buffer over flow if managed by an inexperienced programmer. The goal of a buffer over flow attack is to corrupt a program running with high privileges (i.e. root) in order to take control of the program. If the program has root privilege, the attacker can immediately execute a command to obtain a root shell. In that case, the attacker has full control of the host computer which runs the vulnerable program. The attack is performed in two steps. In the first step, the hacker must find a way to have the appropriate code to launch a root shell in the memory of the program. To manage that, the attacker uses a buffer with non-existent or poorly performed boundary checking. The second step is to subvert the state of the program. The attacker must corrupt the stack pointer to make it point to his malicious code. Several options are possible but the most common is to overwrite a function return address to point to the first instruction of the code of the attacker. This attack is also called “stack smashing attack” [24]. Other attacks such as “loadmodule” or “perl” take advantage of the way some programs sanitize their environment. Others still (“ps”) exploit poor management of temporary files.

2.3.3 Remote to Local (R2L)

There are some similarities between this class of intrusion and U2R, as similar attacks may be carried out. However, in this case, the intruder does not have an account on the host and attempts to obtain local access across a network connection. To achieve this, the intruder can execute buffer overflow attacks, exploit misconfigurations in security policies or engage in social engineering (i.e., obtaining data by tricking a human operator, rather than targeting software flaws) [22].

Examples of remote to local attacks include “warezmaster” and “warezclient”. Those two attacks exploit weaknesses in the file transfer protocol (FTP). The first one grants any user with writing permission on the FTP server. An attacker could use this bug to create a hidden directory and upload illegal files on the server. The “warezclient” attack can be seen as the second step of the “warezmaster” attack since it involves a user downloading the uploaded files from the hidden directory created during the “warezmaster” attack. Other remote to local attacks called “imap”, “named” and “sendmail” exploit bugs in well-known protocols used on the Internet such as DNS and SMTP. Attacks exploiting misconfigurations in the system include “dictionary”, “ftp-write”, “guest” and “Xsnoop”.

2.3.4 Denial of Service (DoS)

In a denial of service attack, an attacker makes a resource on a network either unavailable to legitimate users or too busy or too full to process their queries. The resource can be network bandwidth, computer memory or computing power. There are many different types of DoS attacks [22].

For example, “ARP poisoning” attack [25, 26] can deny access to a machine on a network. The address resolution protocol (ARP) is a protocol used to convert network layer address (such as the IP address) into link layer address (such as the media access control (MAC) address). Each computer on a network has an ARP table which maps network layer addresses to link layer addresses of the other computers or devices on the network. In an “ARP poisoning” attack, an attacker sends unwanted ARP replies to a user on the same network or replies to an ARP query faster than the destination of the query in order to falsify the information contained in the ARP table of the victim. In this case,

it is possible for an attacker to deny access to a resource to one or more users on a network. For instance, depending on the network structure, it is possible for an attacker to modify the entry corresponding to a gateway in the ARP tables of the victim on the network. In this case, the victim might not be able to access the Internet any more. “ARP poisoning” is not represented in the KDD99 dataset, but the concept of denial of service can be easily understood from this attack. It can be noted that “ARP poisoning” can also be used to perform a man-in-the-middle (MITM) attack. A MITM is a type of sniffing attack where the attacker stands in the middle of a communication between two hosts. By poisoning the ARP table of one of the two hosts taking part in the communication, the attacker can redirect the traffic to his computer first and then forward it to the intended destination after having read the content of the message.

The other major type of DoS focuses on resource exhaustion. The attacker sends a huge amount of queries in a short amount of time to the targeted victim. If the victim is a server, resource exhaustion occurs when the server receives more queries than it can process. In that case, legitimate users will not be able to access this resource during the time of the attack or even afterwards if the server crashes. An example of a DoS aiming at exhausting the resource of a machine on a network or an entire network is the “UDP Port DoS” attack, also called “UDP packet storm” [25, 27]. In an “UDP storm”, an attacker forges a packet with a spoofed source address of a host running an “echo” or “chargen” process and sends it to another hosts running a similar “echo” or “chargen” process. The receiving host replies with an echo packet to the spoofed source which also replies with another echo packet. A loop is created between the two hosts leading to resource exhaustion or at least, performance degradation. When targeted at a switch or router, the performance of the entire network can be affected.

Another very popular variant of DoS that has been used extensively by hackers in the last decade is the distributed denial of service (DDoS) [28, 29]. A “DDoS” is performed in two main steps. In the first step, an attacker, called master, gains control over a number of computers, called slaves or zombies, by exploiting unpatched vulnerabilities found in the target systems. The number of slaves can vary from one “DDoS” to another but is usually huge; hundreds of thousands of computers is a perfectly reasonable number in most cases. Once the attacker has taken control of a sufficient number of slaves, the second step can start. The master orders all of the slaves to query a

designated machine at the same time. The target is flooded with the simultaneous queries. After a short time, the memory of the server is exhausted making it unable to handle all of the queries including the ones from legitimate users. The service proposed by the server is denied.

2.4 INTRUSION DETECTION SYSTEMS

Intrusion Detection Systems (IDS) are monitoring systems which are used to detect intrusions on a computer or a network. Intrusions are unauthorized and anomalous activities which were defined by Christopher Kruegel et al. as "a sequence of related actions performed by a malicious adversary that results in the compromise of a target system" [30]. An intrusion detection system is an indispensable tool for network administrators because, without such a device, it would be impossible to analyze the huge amount of packets traversing current networks every second. After more than thirty years of intensive research on intrusion detection systems, the field is still open to further investigations especially regarding the accuracy of the detection. Moreover, variants of known attacks as well as new attacks can often go through the system without being detected.

According to Bishop [31], a good intrusion detection system detects a wide variety of intrusions, in a timely fashion, and presents analysis results as simply and accurately as possible, using any combination of anomaly detection, misuse modeling, or signature detection to identify threats. Anomaly detection works by assuming that attacks look out of the ordinary. Before we can find an anomaly, we need to map out what is normal, and using thresholds look for traffic that is out of these bounds. Misuse modeling looks for specific commands or actions that lead to a known misuse or abuse of otherwise appropriate system states. Signature detection involves recognizing patterns of known code states that can put the system in an undesirable state when executed.

2.4.1 Behavior Classification Systems

Intrusion Detection Systems (IDS) and behavior classification have come a long way since the inception of digital forensic auditing. Intrusion Detection Systems cannot detect all types of intrusions, as attack permutations are constantly being generated. Attack types can have many permutations, and static signatures do not always work. The alternative to signature detection, namely threshold establishment and monitoring, is used to detect the unknown.

A number of machine learning algorithms can be used to effect the classification of normal and malicious network traffic, enhancing an IDS to be able to generalize network traffic into “good” and “bad”, thereby avoiding the necessity to use exact string matches [32] These algorithms fall into two categories, based on their use of supervised and unsupervised learning. Supervised learning techniques require a human operator to classify a training set for the algorithm to learn from. A testing set with known classifications is then used to verify the accuracy of the learner. Unsupervised learning uses unlabeled data, and groups training samples by distinguishing features [33].

2.5 MACHINE LEARNING

Machine learning is a field of artificial intelligence that makes intensive use of statistics. Machine learning algorithms enable computers to make predictions based on a dataset [33]. In this section will describe various approaches to machine learning. The data sets previously described will sometimes have human involvement to decide whether or not the data falls within specific classifications, or the algorithms are trusted to decide how many classes there are, and which instances will belong to which class [34].

2.5.1 Supervised and Unsupervised Learning

The most common machine learning problems fall into two categories; namely supervised, and unsupervised learning [33]. Supervised learning generally consists of a supplied data sets that includes the correct classification of each instance. Supplying the correct values to the learning function allows the trainer to supervise the algorithm, which learns a model that represents the complete set of observed instances and hopefully provides the ability to classify new, or previously unseen instances correctly.

In contrast to supervised learning, unsupervised learning algorithms are supplied the training datasets with an unknown classification, typically because no known classification exists [35]. Unsupervised algorithms, typically clustering algorithms (such as K-means), are designed to look for feature clusters, or groups and patterns of features that have some sort of correlation, or relationship. These relationships can then be used to determine where a future dataset may fit into the model developed from the previously supplied instances.

2.5.2 Supervised Learning and Anomaly Detection

Anomaly detection is typically used when the features presented in the dataset fit nicely into a Gaussian distribution and the features belonging to the positive classes are statistical outliers. The two main differences between anomaly detection and standard supervised learning are the ratio of positive to negative class examples, and the types of classes found in the data sets [36].

For example, in anomaly detection there is a very low number of positive examples (suspicious traffic) found within thousands more negative examples (normal traffic). Conversely, standard supervised learning would typically be expected to have relatively even number of positive and negative examples. It is worth noting the number of anomalous cases could be artificially increased to balance the training sets for model creation.

Traditional anomaly detection algorithms may not work as desired, since attack behaviors do not change over time; only the payload does. Attack behaviors on the other hand will stay relatively consistent, and enough positive examples should be present for the algorithm to get a sense of what a typical positive example will look like; thus a future positive example is likely to look similar to the training set, and the algorithm can be said to generalize fairly well to the problem domain.

2.5.3 Evaluation Metrics

In machine learning and statistics, there is a variety of methods used to evaluate the performance of a classifier. In this section we will introduce and discuss the most common evaluation metrics; the ideas of true and false positive and negative classifications, how they contribute to precision and recall, the F-Score, and finally the Receiver Operating Characteristic (ROC) curve.

2.5.3.1 Confusion Matrix

Anomaly detection can be thought of as binary classification. The traffic is either normal or abnormal. If a single prediction has two possible outcomes, and each can be correct or incorrect, there are thus four possible combinations of predictions and actual outcomes. Based on this, one way to view the performance of a classification algorithm is through a confusion matrix, also known as a contingency table [37]. Considering the anomaly dataset, there are two classes for any prediction: normal and anomaly. If we consider the classes and the correctness of each, we arrive at a 2x2 matrix, as shown in Table 2.1.

Table 2.1: Confusion Matrix.

a	**b**	**classified as**
TP	FP	a = anomaly
FN	TN	b= normal

In this case, the first row represents the actual anomalous classified instances, the second row represents the actual normal instances, and each column represents how those instances were classified during prediction. Reading into this table, all correctly classified instances are in the top-left and bottom-right corners, whereas the incorrectly classified instances are in the bottom-left or top-right corners, and the total of all entries represents the cardinality of the entire dataset. In the case of intrusion detection, as described above, a traffic anomaly is a positive sample whereas normal traffic is negative.

Considering the confusion matrix in Table 2.1, if the actual class is normal, and the predicted value is normal, then we have a true negative. In practice, we desire this value to be as high as possible. If the actual class is normal, and the predicted class is anomaly, we have a false positive. In practice, we want this value to be as low as possible,

as it represents traffic that is normal, but was mistakenly classified as anomalous. If the actual class is anomaly, and the predicted value is normal, then we have a false negative. Just like false positives, we want this value to be as low as possible, as it represents traffic that is not normal, but was mistakenly classified as normal. If the actual class is anomaly, and the predicted value is anomaly, then we have a true positive. These represent anomalies were correctly flagged as such, and warrant either further investigation or outright blocking, as they do not conform to some preset behavioral policy.

Based on the above description, we can define the False Positive Rate as

$$FP\ Rate = \frac{FP}{N_n} \tag{2.1}$$

where FP is the number of false positives observed, and N_n is the total number of observed instances in the negative class (Equation 2.1).

As well, we can define the True Positive Rate as

$$TP\ Rate = \frac{TP}{N_p} \tag{2.2}$$

where TP is the number of true positives observed, and N_p is the total number of observed instances in the positive class (Equation 2.2).

2.5.3.2 Recall

Recall is a measurement method found in data mining used to demonstrate a model's ability to pick up relevant positive instances in a dataset [37, 38]. The Recall ability of a model can be calculated as

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

where FN is the number of false negatives observed (Equation 2.3). Since this value can be used to demonstrate how good the model is at picking out positives, if this was the only measurement being considered, then the model could cheat by always predicting the positive class. In the case of the Zero-R algorithm, this would happen if the majority of training instances seen were positive.

2.5.3.3 Specificity

Specificity is a measurement method found in data mining used to demonstrate a model's ability to pick up relevant negative instances in a dataset [37, 38]. The Specificity of a model can be calculated as

$$Specifity = \frac{TN}{TN + FP} \tag{2.4}$$

Just as with Recall, if this was the only measurement being considered, then the model could cheat by always predicting the negative class (Equation 2.4). In the case of the Zero-R algorithm, this would happen if the majority of training instances seen were negative.

2.5.3.4 Precision

Precision is a measurement method found in data mining used to demonstrate the tradeoff in a model between the sensitivity of picking up true positives, while balancing the false positives [37, 38]. The Precision ability of a model can be calculated as

$$Precison = \frac{TN}{TN + FP} \tag{2.5}$$

This test is different from the other two as it is dependent on the proportion of positive examples seen in the test set (Equation 2.5).

2.5.3.5 F-Score

The relationship between Precision and Recall, as previously observed, is not mutually exclusive. There is a tradeoff between one and the other. The F-Score is a method used in machine learning to compute the harmonized mean between the previously defined Precision and Recall values of a model [37, 38]. This score can be calculated as

$$F - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.6}$$

and is typically used to determine the average between the two results (Equation 2.6). This metric can be valuable in calculating the overall accuracy of a model, however it has been shown that this method of calculation does not take the true negative rate into account [37].

2.5.3.6 Receiver Operating Characteristic

The Receiver Operating Characteristic (ROC) curve is a method of measuring the accuracy of a binary classifier by using the Recall (sensitivity) and the False Positive rate (1 - Specificity) as defined above [38]. This form of measurement has been shown to be a more complete method than other traditional scoring algorithms, at it takes all of the true/false positive/negative combinations into account [38, 39]. The ROC curve, when considering a binary classifier, can be visualized as a two dimensional graph, depicting the ratio between the true positive rate and the false positive rate.

The Area Under the Curve (AUC) can be calculated in a stepwise fashion, by considering all of the instances seen, and plotting their TPR-to-FPR ratio as further instances are observed during the testing phase. These points can then be connected, and the area under this curve taken to form the AUC value. In practice, all of the classifiers that perform better than random guessing will have an AUC between 0.5 and 1.0.

2.5.4 Machine Learning Algorithms

Having examined the methods by which we evaluate the outcomes of classification, we can now turn to examining machine learning algorithms themselves. The machine learning algorithms described below will form the basis for the experiments performed in my thesis.

2.5.4.1 ZeroR

One of the simplest classifiers is the ZeroR, which is typically used as a baseline classifier against which other, more complicated, classifiers are measured [33]. The ZeroR classifier is unique in that it does not actually take into account the features or attributes of the training data used to build the model. In the case of supervised learning, ZeroR calculates the average value of the supplied class or the mode of the supplied class. This value is then used every time a new test instance is given to the model.

Intuitively, the model prediction will have accuracy roughly the same as the most common class found in the training data, provided the training, holdout, and cross validation random sampling contains the same distribution of classes. In our experimental test set, the attack types have been simply labeled as normal and anomaly.

2.5.4.2 Naive Bayes

The Naive Bayes algorithm uses Bayes' Theorem to predict the posterior probability of a class, given the presence of one or more features [33]. This algorithm is said to be naive because of the strong assumption that features are independent from one another in a probabilistic sense. Despite this limitation, compared to other more sophisticated algorithms, some researchers [40] have found that it is fairly competitive in predictive performance in general.

2.5.4.3 J48 Decision Tree

Decision trees are used to classify data by sorting the attributes and their values in a tree fashion, with the leaves of the tree being the classification. The most basic example of decision tree algorithms are the ID3 and C4.5 which uses entropy and information gain to recursively split the available attributes into decisions that will decrease the entropy across all values associated with that attribute [41]. Decision tree J48 is the implementation of algorithm ID3 developed by the WEKA project team [102]. The result is such that the highest information gain split is at the root node, all the way down to the values with the lowest near to the leaves.

2.5.4.4 Linear and Logistic Regression

Linear regression, as the name specifies, tries to find a function that intersects the average of all supplied data points in order to predict a real-valued output (as opposed to discrete valued output) [33]. The algorithm can be supervised, as the training data in this case also supplies the known class (correct answer).

2.5.4.5 K-Nearest Neighbour

The K-nearest-neighbor algorithm is an instance-based learning algorithm that relies on the distance of each new sample to one or more previously seen instances in a feature space for classification [42].

2.5.4.6 Support Vector Machines

Support Vector Machines involve finding a plane of separation between two groups of graphed instances such that the plane has a maximum margin between the groups. This is done by finding an n-1 dimensional hyper plane in an n-dimensional space to separate the two classes [43].

2.5.4.7 Artificial Neural Networks

Modern neural networks are based on the structure of interconnected networks of nodes, or neuron-like structures. The neurons in an artificial neural network are also referred to as units, or nodes. The simple neuron contains input signals, each signal representing a variable under consideration to be passed through the neuron's function, ultimately producing an output signal [33, 44].

The back propagation learning algorithm used for neural networks determines how the weights are adjusted after each iteration of training. If the network produces a desired output, the weights need not change. However, if the output is not correct, the weights of each node must be modified depending on multiple factors. These connections are strengthened through the process of back propagation [33].

2.5.4.8 Genetic Programming

Genetic programming (GP) is a subclass of evolutionary algorithms [45, 46]. Similarly to other artificial intelligence paradigms such as swarm intelligence, evolutionary computation (EC) is a bio-inspired, population-based technique. However, as the name suggests, EC is inspired by the theory of evolution. The individuals in the populations are often called chromosomes and the pieces of these chromosomes that are modified during the evolutionary process are called genes. Every evolutionary algorithm follows the same basic scheme.

CHAPTER 3

LITERATURE REVIEW

This chapter contains literature review about the human factors most significantly affect information security in organizations. This review focuses on studies of the dimensions of security needs, security threats, classification of human behaviors, and intrusion detection systems. Currently there is a lack of research in analyzing human factors in Information Security, as the majority of studies are focusing on either usability studies or task analyses. We sort literature review results by subject.

Potential environmental risks can come in many different forms, both externally and within organizations. Generally, sources of security threats can be broken up into two categories: natural disasters and human threats [47, 48].

Human threats are threats perpetrated by individuals or groups of individuals that attempt to penetrate systems through computer networks, telephone networks or other sources. These attacks generally target known security vulnerabilities of systems and many of these vulnerabilities are simply due to configuration errors [49]. The major sources of human security threats can take the form of insider and outsider corporate hacking for information or hacking for malice.

It can be concluded that the main distinction between insiders and outsiders is the fact that insiders are trusted [50]. These trusted insiders include employees but also, due to collaboration across companies, contractors and consultants, temporary helpers

and third party business partners [3]. Trusted insiders have legitimate access to an organization's information [51, 52, 53].

Many times, organizations overlook the human factor which is a factor that security depends upon [54]. Technology is often seen as the immediate answer to Information Security problems [55]. However, despite the fact that many organizations make use of a high number of technical security controls, they still show a non-proportional number of security breaches; this happens because Information Security is primarily a human factors problem that remains unaddressed [56]. Since people are the ones who utilize technology, it is just as important to invest in the human factor [55].

A security system, regardless of design and implementation, will have to rely on the human factor; the continuous implementation of technical solutions will fail to handle the people [57]. In addition, Schneier states that technology cannot solve the security problems and believing so shows a lack of understanding of the problems and technology [5]. Mitnik finds technological protection inadequate and argues that users are targeted as the weakest system link [58]. Information Security is a set of measures which should be seen as a system and not a single unit [59]. An Information Security system, except of encapsulating the human factor as a component, is also described as a continuously evolving entity [60]. Panko recognizes the intentional threat from both in and out of the organization premises without analyzing the unintentional exposure of the system to a threat [61]. A security survey from Cisco Systems reveals that users who work remotely, although they claim to have awareness of security risks, would still engage into actions which endanger the system security [62].

The unauthorized use of computer systems is made by either accidental or deliberate causes. Accidental causes are any unexpected natural disasters and the human factor, such as power surges or misconfiguration. The deliberate causes are actions made by conscious choice; for example, using a program flaw to gain access on a computer system [63]. An evaluation of factors which produce security breaches has shown that sixty five percent of the economic loss in Information Security breaches is due to human error, and only three percent from malicious outsiders. Considering the fact that the efforts to evaluate the human factor in Information Security are basically nonexistent, it is questionable why there has been so much focus on technological means [64].

People as part of the system interact by developing, implementing and using both software and hardware; when a user has poor training, an ideal and flawless software or hardware solution will still not be of any use. Therefore, people will always be a weak system component [59]. Users often perceive their computer systems as a black box, without understanding or the functionality or will to know it [59, 60]. A good example is that users want to operate their computers in the same way as any other household electric appliances [59]. Many users are found to treat confidential information in an irresponsible manner by having empty passwords or using their name as one in contradiction to the fact that the same users would never intentionally leave their keys in the outside lock [59]. Regardless of the partial automation that is introduced, people are without doubt involved in technology. Therefore, there is a probability for human error which may result in system exposure [56].

Security breaches are often caused by careless and unaware users [60]. The majority of people want to get their jobs done more than they are interested in protecting themselves; a behavioral tendency that gives surface for attacks. In addition, most people do not understand subtle threats and they engage into actions which might expose the system [59]. One more view that was not mentioned by any of the related sources is the exception handling, or differently how people might react when something unexpected occurs; many times attackers rely on the alternative actions that people might take when they encounter something for the first time [59].

Security is outlined as a continuous process which requires an unending investment in both technology and users' education; technology, as only a part of it, cannot be the only component for having a secure infrastructure [60]. Users should get educated about risks and responsibilities; education and awareness [65] are identified as key factors in addressing the human element of security [62]. Except giving users an understanding of threats existence, it is also proposed to convince users of the need for security; people would then follow the security requirements in a given situation [65, 62].

While Kraemer creates a human factors evaluation method for computer and Information Security, it does have certain constraints. The derived vulnerability evaluation follows a technical vulnerability audit and it takes place on top of vulnerabilities with the human factor components, this limits the possibility of having a

human factors vulnerability evaluation without inspecting the technological component [63]. In addition, the vulnerability evaluation is made only according to the earlier found technical vulnerabilities and therefore there is a large part of the human factor, the non-technical unintentional vulnerabilities, which remain unaddressed [63]. The feedback comes through qualitative interviews with the involved network administrators and not the end users while by doing the opposite the obtained information would be more prosperous and realistic. Furthermore, the evaluation is qualitative and based on results which come from a qualitative analysis software package which raises a risk of inconsistency as the results may vary if an improper categorization is made [63].

Anderson's early work in this domain classifies system abusers into External Penetrators, Internal Penetrators, and Misfeasors [66]. It is very useful at a broad conceptual level, the classification does not provide any significant assistance in terms of incident detection, with all insider misuse related incidents being grouped under the single 'misfeasor heading.

The Neumann-Parker taxonomy is based on incidents reported over 20 years [67]. It classifies intrusions into nine categories, which describe the nature of the attacks.

Cheswick and Bellovin have classified attacks into seven categories which is drawn upon their work on firewalls. Although, this approach gives an overview of the attacks and classifies the main categories of attacks, it is too general and does not give an insight to the characteristics of attacks [68].

Tuglular's taxonomy is the first comprehensive taxonomy of misfeasor incidents [69]. The taxonomy classifies computer misuse incident in three dimensions: incidents, response and consequences. The incidents dimension is further classified into target, subject, method, place, and time sub-dimensions. The response dimension is divided into recognition, trace, indication, and suspect. The consequences dimension includes disruption, loss, effect, violation, misuse type, misuse act, and result.

Magklaras-Furnell's Insider Threat Prediction Model is human centric, and the authors argue that all actions that constitute IT misuse lead back to the human factors.

The fundamental aspect for the taxonomy is classifying people in three basic dimensions: system role, reason of misuse and system consequences [70].

Stanton and colleagues define behavioral information security as the human actions that influence the availability, confidentiality, and integrity of information systems. They use social, organizational, and behavioral theories and approaches, and conduct a series of empirical investigations in developing taxonomy of security behaviors and identifying the motivational predictors of such behaviors [71].

Information security behavior refers to a set of core information security activities that have to be adhered to by end-users to maintain information security as defined by information security policies [72]. Padayachee's taxonomy of the motivational factors associated with compliant security behavior. Aspects such as personality traits and cultural norms are relevant to motivation, however, these aspects were not considered in the taxonomy.

After reviewing the taxonomy studies, we need to review detection studies to manage the insider threats.

The first major work in the area of intrusion detection was discussed by Anderson in 1980 [66]. Anderson introduced the concept that certain types of threats to the security of computer systems could be identified through a review of information contained in the system's Audit Trail. Many types of operating systems, particularly the various "flavors" of UNIX, automatically create a report which details the activities occurring in the system. Anderson indicates that there is a particular class of external attackers, known as clandestine users who escape both system access controls and auditing mechanisms through the manipulation of system privileges or by operating at a level that is lower than what is regularly monitored by the audit trail.

Dr. Dorothy Denning proposed an Intrusion Detection model in 1987 which became a landmark for the research in this area [73]. The model which she proposed forms the fundamental core of most Intrusion Detection methodologies in use today.

There are two general categories of attacks which intrusion detection technologies attempt to identify - anomaly detection and misuse detection [74, 75]. Anomaly detection identifies activities that vary from established patterns for users, or

groups of users. Anomaly detection typically involves the creation of knowledge bases that contain the profiles of the monitored activities.

The second general approach to intrusion detection is misuse detection. This technique involves the comparison of a user's activities with the known behaviors of attackers attempting to penetrate a system [76, 77]. While anomaly detection typically utilizes threshold monitoring to indicate when a certain established metric has been reached, misuse detection techniques frequently utilize a rule-based approach. When applied to misuse detection, the rules become scenarios for network attacks. The intrusion detection mechanism identifies a potential attack if a user's activities are found to be consistent with the established rules. The use of comprehensive rules is critical in the application of expert systems for intrusion detection.

There are a few different groups advocating various approaches to using neural networks for intrusion detection. A couple of groups created keyword count based misuse detection systems with neural networks [78, 79]. The data that they presented to the neural network consisted of attack-specific keyword counts in network traffic. Such a system is close in spirit to a host-based detection system because it looks at the user actions.

In a different approach, researchers created a neural network to analyze program behavior profiles instead of user behavior profiles [80]. This method identifies the normal system behavior of certain programs, and compares it to the current system behavior. Cannady developed a network-based neural network detection system in which packet-level network data was retrieved from a database and then classified according to nine packet characteristics and presented to a neural network [81].

An increasing number of researches have been conducted on intrusion detection based on neural networks. Neural-net-based IDSs can be classified into the following four categories, the first category MLFF neural-net-based IDSs includes the systems built on Multi-Layer Feed-Forward (MLFF) neural nets, such as the Multi-Layer Perceptron (MLP) and Back Propagation (BP). MLFF neural nets have been used in most early research in neural-net-based IDSs [82].

InSeon and Ulrich tried to integrate a smart detection engine into a firewall and detecting unusual structures in data packets uses a classical feed-forward multi-layer perceptron network: a back propagation neural network and time delay neural network to program-based anomaly detection [83]. Also Kang and colleagues built IDS deals well various mutated attacks as well as well-known attacks by using Time Delay Neural Network classifier that discriminates between normal and abnormal packet flows [84].

Other researchers have compared the effectiveness of MLFF neural networks to other neural networks, Siddiqui compared the effective of BP with Fuzzy ARTMAP [85], Grediaga compared the effective of MLFF with Self organization map (SOM) [86], Zhang made comparison between BP and RBF network in IDSs [87], and Vatisekhovich compared effectives between MLFF and recurrent neural network, MLFF neural nets have been shown to have lower detection performance than SOM [88].

CHAPTER 4

CLASSIFICATION STUDY

4.1 OVERVIEW

Taxonomy is an important milestone for this thesis because it will enhance the ability to examine the problem in a more systematic way and will eventually contribute to the establishment of a behavior-based intrusion detection model. The derivation of this model will contribute to further insider misuse detection research and development efforts around the world.

In this context, we have primarily researched the human factors in information security breaches, and the following questions came up at this stage:

i. What types of human factors cause what kind of information security breaches?

The purpose of asking this question is to identify information security breaches and human factors, and highlight the link in between.

ii. Is there a classification, taxonomy or a study already published about this subject?

The purpose of asking this question is to get to know that how the researchers approach the subject historically and where we are standing. The intrusion detection systems research community has developed various approaches for systematically

classifying intrusion incidents. Legitimate user misuse is considered a special case of an intrusive activity.

A recent taxonomy in this subject matter has been published by Padayachee. Padayachee's study surveys the extrinsic and intrinsic motivations that influence the propensity toward a compliant information security behavior [72].

Padayachee indicates that the compliant information security behavior refers to the set of core information security activities that have to be adhered to by end-users to maintain information security as defined by information security policies [72]. In addition, the compliance mindset also subscribes to what might be called a deterrence theory of motivation, which employs mandates, procedural controls and threats of punishment to manage and motivate people.

One of the most referred taxonomy was published by Stanton and colleagues [71]. Stanton and colleagues define behavioral information security as the human actions that influence the availability, confidentiality, and integrity of information systems. They use social, organizational, and behavioral theories and approaches, and conduct a series of empirical investigations in developing a taxonomy of security behaviors and identifying the motivational predictors of such behaviors.

Security behavior can also be described using two-factor taxonomy, where the two factors are intentionality and technical expertise [71]. As shown in Figure 4.1, this creates six categories of security behaviors, where two of those behaviors, Aware Assurance and Basic Hygiene, are positive designed to increase security, and four of the behaviors may result in breaches to security.

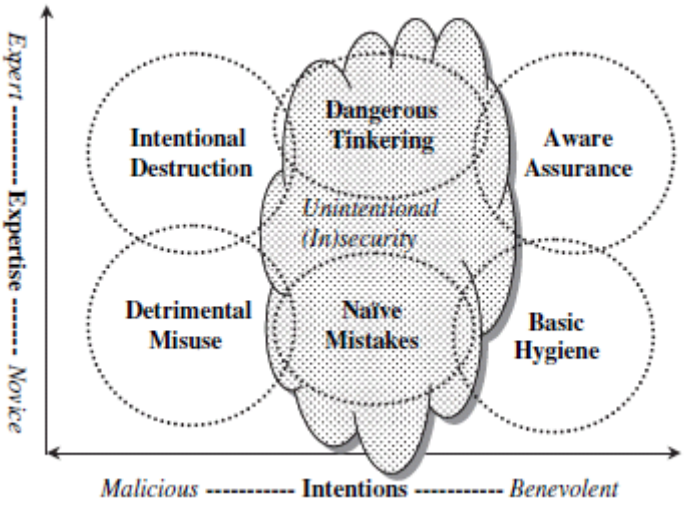

Figure 4.1: Two-factor Taxonomy of End User Security Behaviors [71].

Intentional Destruction covers the actions of malicious insiders, who have technical expertise and the intent to do harm, whereas Detrimental Misuse involves personnel who have malicious intent, but lack technical expertise. Dangerous Tinkering covers behaviors that require technical expertise, but where there is not an intention to do harm. Perhaps the most common behavior, which will be covered in the most detail in this thesis is Naïve Mistakes, in which individuals with low expertise and without malicious intentions perform an action which was not intended to harm the organization, but yet could result in a security breach.

4.2 AN EXTENDED TAXONOMY STUDY WITH ADDITIONAL DIMENSIONS AND FEATURES

This taxonomy study covers a more recent and up-to-date taxonomy effort with increased dimensions and features. We certainly believe that the level of the effort in this taxonomy study brings it to the more advanced and general level that can be used for developing a behavior-based intrusion detection model.

The next objective is to apply some emerging and well-known Artificial Intelligence techniques as based on the extended taxonomy for classification purposes with high accuracy.

4.2.1 Impact Level

After looking through these taxonomies and the others mentioned in Literature Review chapter, we decided to study on Padayachee's, and Stanton and his colleagues' taxonomies but we found the Padayachee's taxonomy very theoretical mostly related with motivation but it was advanced. On the other hand, Stanton and colleagues' taxonomy was very practical but it was basic. At this point a practical and advanced taxonomy study that can combine their main strengths came forward as a result. We also found Stanton and colleagues' taxonomy very useful and improvable. Firstly we thought about adding impact level as the third dimension because the impact level is related with the risk of the behavior. If one can define the risk of a behavior, one can take precautions to reduce or avoid it and if the risk is low or the cost of treatment is not cost-effective for the organization, it can be ignored [9].

The impact level comes from consideration of three potential compromises as follows:

- Confidentiality: the potential impact if the information is seen by those who should not see it,
- Integrity: the potential impact if the accuracy or completeness of the information is compromised,
- Availability: the potential impact if the information becomes inaccessible.

We get the impact levels from National Vulnerability Database (NVD) [110], Common Vulnerability Scoring System (CVSS) [111] and Common Vulnerabilities and Exposures (CVE) [112].

Table 4.1: Three Dimensional Human Threats Taxonomy.

Expertise	Intention	Impact Level	Title	Description	Example
High	Malicious	High	Intentional destruction	Behavior requires technical expertise together with a strong intention to do harm to the organization's IT and resources and has high impact level.	Employee breaks into an employer's protected files in order to steal a trade secret.
High	Malicious	Medium	Man in the middle	Behavior requires technical expertise and includes intention to do harm through annoyance, harassment, rule breaking, etc. and has medium impact level.	Employee stands in the middle of a communication between two hosts. By poisoning the ARP table of one of the two hosts taking part in the communication, the attacker can redirect the traffic to his computer first and then forward it to the intended destination after having read the content of the message.
Low	Malicious	High	Resource exhaustion	Behavior requires minimal technical expertise but nonetheless includes intention to do harm through annoyance, harassment, rule breaking, etc. and has high impact level.	Employee sends large ping packets to a computer or a server, resource exhaustion occurs when the server or the computer receives more queries than it can process. In that case, legitimate users will not be able to access this resource during the time of the attack or even afterwards if the server crashes.
Low	Malicious	Medium	Steeling Privilege	Behavior requires technical expertise and includes intention to do harm through annoyance, harassment, rule breaking, etc. has medium impact level.	Getting an employee's user name and password who has no restricted internet access and using it.
High	Neutral	High	Dangerous tinkering	Behavior requires technical expertise but no clear intention to do harm to the organization's IT and resources, has high impact level.	Employee configures a wireless gateway that inadvertently allows wireless access to the company's network by attackers who scan wireless networks and access them for stealing secrets.
High	Neutral	Medium	Accidentally Allowing	Behavior requires technical expertise but no clear intention to do harm to the organization's IT and resources, has medium impact level.	Employee configures the firewall that inadvertently allows employees use probing services which attackers use for attacking.
Low	Neutral	High	Naive mistakes	Behavior requires minimal technical expertise and no clear intention to do harm to the organization's information technology and resources and has high impact level.	Choosing a bad password such as ''password.'' An employee can use his/her colleague's account for accessing private or protected files.
Low	Neutral	Medium	Personal usage	Behavior requires minimal technical expertise and no clear intention to do harm to the organization's information technology and resources and has medium impact level.	Employee gets root privilege and stores personal large size data on a company server and shares it on the internet.
High	Beneficial	High	Aware assurance	Behavior requires technical expertise together with a strong intention to do good by preserving and protecting the organization's information technology and resources against high level breaches.	Recognizing the presence of a backdoor program through careful observation of own PC.
High	Beneficial	Medium	Paying attention	Behavior requires technical expertise together with a strong intention to do good by preserving and protecting the organization's information technology and resources against medium level breaches.	Employee recognizes an abnormal usage or services on a computer or a server and recognize that somebody obtained administrator rights on the attacked computer in order to have full control of it.
Low	Beneficial	High	Basic hygiene	Behavior requires no technical expertise but includes clear intention to preserve and protect the organization's IT and resources against high level breaches.	A trained and aware employee resists an attempt at social engineering by refusing to reveal her password to a caller claiming to be from computer services.
Low	Beneficial	Medium	Awareness	Behavior requires no technical expertise but includes clear intention to preserve and protect the organization's IT and resources against medium level breaches.	Reporting a suspicious e-mail which wants to click the link and enter the personal info and also not clicking the link or entering personal info.

Upon adding the impact level we obtained a three level taxonomy as shown in Table 4.1 and we know the risk level of the behavior so that we can accept it or ignore it but how can we detect it? Another dimensional need came out with the motivation of

this question. None of the previously mentioned taxonomies are oriented towards detection of insider misuse, in terms of considering how we would approach the task of monitoring activities to determine where problems may be apparent.

4.2.2 Threat Layer

In determining a means to link classification to the method of detection, it is considered appropriate to classify human behavior as based on the level of the system at which they might be detected. The basis for this is that different types of behaviors manifest themselves at varying layers of the system. With this form of classification in mind, the concept can be illustrated using a variety of recognized insider activities, and then considering the different layers at which they may be detected. The classification is presented in Table 4.2, and then examples of the incidents concerned are considered in the sub-sections that follow. These consider what could be monitored, and how this could be used to detect, control and restrict misuse-related behavior.

Table 4.2: Four Dimensional Human Threats Taxonomy.

Expertise	Intention	Impact Level	Threat Layer	Title	Description	Example
High	Malicious	High	OS	Intentional destruction	Behavior requires technical expertise together with a strong intention to do harm to the organization's IT and resources and has high impact level.	Employee breaks into an employer's protected files in order to steal a trade secret.
High	Malicious	Medium	Network	Man in the middle	Behavior requires technical expertise and includes intention to do harm through annoyance, harassment, rule breaking, etc. and has medium impact level.	Employee stands in the middle of a communication between two hosts. By poisoning the ARP table of one of the two hosts taking part in the communication, the attacker can redirect the traffic to his computer first and then forward it to the intended destination after having read the content of the message.
Low	Malicious	High	Network	Resource exhaustion	Behavior requires minimal technical expertise but nonetheless includes intention to do harm through annoyance, harassment, rule breaking, etc. and has high impact level.	Employee sends large ping packets to a computer or a server, resource exhaustion occurs when the server or the computer receives more queries than it can process. In that case, legitimate users will not be able to access this resource during the time of the attack or even afterwards if the server crashes.
Low	Malicious	Medium	OS	Steeling Privilege	Behavior requires technical expertise and includes intention to do harm through annoyance, harassment, rule breaking, etc. has medium impact level.	Getting an employee's user name and password who has no restricted internet access and using it.
High	Neutral	High	Network	Dangerous tinkering	Behavior requires technical expertise but no clear intention to do harm to the organization's IT and resources, has high impact level.	Employee configures a wireless gateway that inadvertently allows wireless access to the company's network by attackers who scan wireless networks and access them for stealing secrets.
High	Neutral	Medium	Network	Accidentally Allowing	Behavior requires technical expertise but no clear intention to do harm to the organization's IT and resources, has medium impact level.	Employee configures the firewall that inadvertently allows employees use probing services which attackers use for attacking.
Low	Neutral	High	OS	Naive mistakes	Behavior requires minimal technical expertise and no clear intention to do harm to the organization's information technology and resources and has high impact level.	Choosing a bad password such as ''password.'' An employee can use his/her colleague's account for accessing private or protected files.
Low	Neutral	Medium	OS	Personal usage	Behavior requires minimal technical expertise and no clear intention to do harm to the organization's information technology and resources and has medium impact level.	Employee gets root privilege and stores personal large size data on a company server and shares it on the internet.
High	Beneficial	High	OS	Aware assurance	Behavior requires technical expertise together with a strong intention to do good by preserving and protecting the organization's information technology and resources against high level breaches.	Recognizing the presence of a backdoor program through careful observation of own PC.
High	Beneficial	Medium	Network	Paying attention	Behavior requires technical expertise together with a strong intention to do good by preserving and protecting the organization's information technology and resources against medium level breaches.	Employee recognizes an abnormal usage or services on a computer or a server and recognize that somebody obtained administrator rights on the attacked computer in order to have full control of it.
Low	Beneficial	High	OS	Basic hygiene	Behavior requires no technical expertise but includes clear intention to preserve and protect the organization's IT and resources against high level breaches.	A trained and aware employee resists an attempt at social engineering by refusing to reveal her password to a caller claiming to be from computer services.
Low	Beneficial	Medium	Network	Awareness	Behavior requires no technical expertise but includes clear intention to preserve and protect the organization's IT and resources against medium level breaches.	Reporting a suspicious e-mail which wants to click the link and enter the personal info and also not clicking the link or entering personal info.

- Network-Layer: Misuser activity may relate to the use of network services, several type of misuse would be detectable by monitoring the activities at the network traffic layer.

- System-Layer: In contrast to detecting network-layer incidents, monitoring at the system layer necessitates that monitoring activity be conducted upon individual host systems.

4.2.3 Threat Type

We identify the monitoring level but how can we detect the actions which endanger the integrity, confidentiality or availability of a resource as an effort to provide a solution to existing security issues? This can be done by intrusion detection systems (IDS).

There are a large variety of different attack types [20]. An attacker may attempt to guess a user's password. Attackers may also monitor the network to obtain the information they require to launch an attack. Sometimes attackers try to put unauthorized programs onto computers that they have access to. Sometimes they may steal information or corrupt information. They may also try to perform a denial of service attack.

A good taxonomy makes it possible to classify individual attacks into groups sharing common properties [21]. One widely used taxonomy [22] divides attacks into four classes: Probes, Denial of Service (DoS), User to Root (U2R) and Remote to Local (R2L).

i. **User to Root Attack (U2R)** is a class of exploit in which the attacker starts out with access to a normal user account on the system (perhaps gained by sniffing passwords, a dictionary attack, or social engineering) and is able to exploit some vulnerability to gain root access to the system [22].
ii. **Remote to Local Attack (R2L)** occurs when an attacker who has the ability to send packets to a machine over a network, but who does not have an account on that machine, exploits some vulnerability to gain local access as a user of that machine [22].
iii. **Probing Attack** is an attempt to gather information about a network of computers for the apparent purpose of circumventing its security controls [22].

iv. **Denial of Service Attack (DoS)** is an attack in which the attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or denies legitimate users access to a machine [22].

Table 4.3: Human Threats Taxonomy.

Expertise	Intention	Impact Level	Threat Layer	Attack Type	Title	Description	Example
High	Malicious	High	OS	User to Root (U2R)	Intentional destruction	Behavior requires technical expertise together with a strong intention to do harm to the organization's IT and resources and has high impact level.	Employee breaks into an employer's protected files in order to steal a trade secret.
High	Malicious	Medium	Network	Denial of Service (DoS)	Man in the middle	Behavior requires technical expertise and includes intention to do harm through annoyance, harassment, rule breaking, etc. and has medium impact level.	Employee stands in the middle of a communication between two hosts. By poisoning the ARP table of one of the two hosts taking part in the communication, the attacker can redirect the traffic to his computer first and then forward it to the intended destination after having read the content of the message.
Low	Malicious	High	Network	Denial of Service (DoS)	Resource exhaustion	Behavior requires minimal technical expertise but nonetheless includes intention to do harm through annoyance, harassment, rule breaking, etc. and has high impact level.	Employee sends large ping packets to a computer or a server, resource exhaustion occurs when the server or the computer receives more queries than it can process. In that case, legitimate users will not be able to access this resource during the time of the attack or even afterwards if the server crashes.
Low	Malicious	Medium	OS	Remote to Local (R2L)	Steeling Privilege	Behavior requires technical expertise and includes intention to do harm through annoyance, harassment, rule breaking, etc. has medium impact level.	Getting an employee's user name and password who has no restricted internet access and using it.
High	Neutral	High	Network	Probe	Dangerous tinkering	Behavior requires technical expertise but no clear intention to do harm to the organization's IT and resources, has high impact level.	Employee configures a wireless gateway that inadvertently allows wireless access to the company's network by attackers who scan wireless networks and access them for stealing secrets.
High	Neutral	Medium	Network	Probe	Accidentally Allowing	Behavior requires technical expertise but no clear intention to do harm to the organization's IT and resources, has medium impact level.	Employee configures the firewall that inadvertently allows employees use probing services which attackers use for attacking.
Low	Neutral	High	OS	User to Root (U2R)	Naive mistakes	Behavior requires minimal technical expertise and no clear intention to do harm to the organization's information technology and resources and has high impact level.	Choosing a bad password such as ''password.'' An employee can use his/her colleague's account for accessing private or protected files.
Low	Neutral	Medium	OS	User to Root (U2R)	Personal usage	Behavior requires minimal technical expertise and no clear intention to do harm to the organization's information technology and resources and has medium impact level.	Employee gets root privilege and stores personal large size data on a company server and shares it on the internet.
High	Beneficial	High	OS	Remote to Local (R2L)	Aware assurance	Behavior requires technical expertise together with a strong intention to do good by preserving and protecting the organization's information technology and resources against high level breaches.	Recognizing the presence of a backdoor program through careful observation of own PC.
High	Beneficial	Medium	Network	User to Root (U2R)	Paying attention	Behavior requires technical expertise together with a strong intention to do good by preserving and protecting the organization's information technology and resources against medium level breaches.	Employee recognizes an abnormal usage or services on a computer or a server and recognize that somebody obtained administrator rights on the attacked computer in order to have full control of it.
Low	Beneficial	High	OS	Remote to Local (R2L)	Basic hygiene	Behavior requires no technical expertise but includes clear intention to preserve and protect the organization's IT and resources against high level breaches.	A trained and aware employee resists an attempt at social engineering by refusing to reveal her password to a caller claiming to be from computer services.
Low	Beneficial	Medium	Network	Remote to Local (R2L)	Awareness	Behavior requires no technical expertise but includes clear intention to preserve and protect the organization's IT and resources against medium level breaches.	Reporting a suspicious e-mail which wants to click the link and enter the personal info and also not clicking the link or entering personal info.

Threat type is important; because if we want to manage the threats we need to be able to detect them, so we add threat type attribute in our taxonomy and we get five dimensional Human Threats Taxonomy as shown in Table 4.3.

4.3 SUMMARY

In this chapter we introduced a suitable taxonomy of the human threat factors, based on network and system-layer events associated to legitimate user actions. We described the impact level of attacks and also described the attack types. We gave examples about the human threats. The taxonomy is tailored to the needs of automated human threat prediction. The establishment of this classification scheme paves the way for the construction of a suitable behavior-based intrusion detection model.

CHAPTER 5

METHODOLOGY

5.1 OVERVIEW

In the last chapter we classified human threats and to prevent those threats, we need to detect them. There are many approaches which use data mining algorithms to detect insider attacks. Network based detection is one of the mechanism to accurately distinguish insider behavior from the normal behavior. Anomaly detection has attracted the attention of many researchers to overcome the weakness of signature-based IDSs in detecting novel attacks [113].

However, when we look at the state of the detection solutions and commercial tools, there is little evidence of using the anomaly detection approach, and people still think that it is an immature technology. We believe that if we productively apply machine learning while narrowing the large variety of algorithms, we can get high detection rate, low false alarm rate and better time cost in anomaly detection. In order to accomplish this goal, a suitable experimental methodology was designed with the following properties in mind:

i. Algorithms should be selected from a variety of statistical models in the machine learning area, so a proper representation of the fundamental options are tested.

ii. The dataset against which the algorithms are tested should be a realistic representation of both normal and abnormal traffic, including zero-day attack instances.

In this thesis, our focus is on supervised learning algorithms using labeled instances to train each representative model, and testing the model against an established test set to evaluate the potential predictive power of each model against attacks that represent both previously seen and novel network behaviors. Machine learning algorithm schemes were evaluated to see which approach to learning provides the best solution to the difficult task of network traffic classification for security purposes.

Finding an appropriate dataset against which to test supervised learning algorithms is a difficult task. DARPA's military network attack simulation dataset from the KDD Challenge Cup was used as a representation of network flows found in a realistic network environment [89]. This dataset, despite having origins dating back a decade, is still being used today as it contains a variety of well-known attack type behavior representative of similar attacks seen today.

In this thesis, we have compared the existing algorithms, and we have spent a great deal of time in discovering robust algorithms that are adequate representative samples of what is currently used by the Machine Learning community. This exploration naturally led to several existing frameworks that already supplied a fair number of robust algorithms. We came to the conclusion that the Weka [90] framework not only included well-developed and widely vetted libraries, but also included a fairly robust data exploration and experimentation framework that would satisfy the consistency and reliability attributes we were after.

Our experimental methodology follows three main phases to ensure a comprehensive evaluation of the KDD dataset against the proposed algorithms. These steps are:

i. pre-processing the dataset,
ii. training and testing algorithms,
iii. systematic evaluation of each training and testing evaluation metrics.

The dataset preparation, model parameter selection, and selected evaluation metrics will be further explained in this chapter. The algorithm evaluations then follow in next chapter.

5.2 KDD DATA SET

KDD is the most widely used data set for the evaluation of anomaly detection methods. This data set is prepared by Stolfo et al. [91] and is built based on the data captured in DARPA IDS evaluation program [78]. KDD is about 4 gigabytes of compressed raw data of 7 weeks of network traffic, which can be processed into about 5 million connection records, each with about 100 bytes. The two weeks of test data have around 2 million connection records. The simulated attacks fall in one of the following four categories:

i. User to Root Attack (U2R) is a class of exploit in which the attacker starts out with access to a normal user account on the system (perhaps gained by sniffing passwords, a dictionary attack, or social engineering) and is able to exploit some vulnerability to gain root access to the system.

ii. Remote to Local Attack (R2L) occurs when an attacker who has the ability to send packets to a machine over a network, but who does not have an account on that machine, exploits some vulnerability to gain local access as a user of that machine.

iii. Probing Attack is an attempt to gather information about a network of computers for the apparent purpose of circumventing its security controls.

iv. Denial of Service Attack (DoS) is an attack in which the attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or denies legitimate users access to a machine.

The KDD training set contains 4,898,430 entries. Each entry is represented by 41 variables such as duration, “src_bytes”, “dst_bytes”, etc., and a label. From these 41 variables, 3 are non-numerical: “protocol_type”, “service” and “flag”. There are 3 protocol types (TCP, UDP and ICMP), 70 services and 11 flags. All the examples are

separated into four different classes of attacks and the class Normal. The distribution of examples over the different classes is shown in Table 5.1.

The analysis of the test set can also reveal interesting facts. The test set is composed of 311,029 entries. The distribution of examples containing previously unseen attacks and examples containing previously seen attacks for each class of attacks is shown in Table 5.1. Table 5.1 shows that the number of unseen attacks added in the test set is huge, especially for the classes. The distribution of the examples in the test set over the different classes is very similar to the distribution of the training set. Another noticeable fact is that the attacks “spy” and “warezclient” belonging to the class R2L are not represented in the test set.

It is important to note that the test data is not from the same probability distribution as the training data, and it includes specific attack types not in the training data which make the task more realistic. Some intrusion experts believe that most novel attacks are variants of known attacks and the signature of known attacks can be sufficient to catch novel variants. The data sets contain a total number of 22 training attack types, with an additional 17 types in the test data only.

Table 5.1: Proportions of Attack Classes in the KDD Data Set.

Category	**Attack Name**	**Training Data set**	**Test Data set**
U2R	buffer_overflow	30	22
U2R	loadmodule	9	2
U2R	perl	3	2
U2R	ps	-	16
U2R	rootkit	10	13
U2R	sqlattack	-	2
U2R	xterm	-	13
R2L	ftp_write	8	3
R2L	guess_passwd	53	4367
R2L	httptunnel	-	158
R2L	imap	12	1
R2L	multihop	7	18
R2L	named	-	17
R2L	phf	4	2
R2L	sendmail	-	17
R2L	snmpgetattack	-	7741
R2L	snmpguess		2406
R2L	spy	2	-
R2L	warezclient	1020	-
R2L	warezmaster	20	1602
R2L	worm	-	2
R2L	xlock	-	9
R2L	xsnoop	-	4
Probing	ipsweep	12481	306
Probing	mscan	-	1053
Probing	nmap	2316	84
Probing	portsweep	10413	354
Probing	saint	-	736
Probing	sendmail	15892	1633
DoS	apache2	-	794
DoS	back	2203	1098
DoS	land	21	9
DoS	mailbomb	-	5000
DoS	neptune	1072017	58001
DoS	pod	264	87
DoS	processtable	-	759
DoS	smurf	2807886	164091
DoS	teardrop	979	12
DoS	udpstorm	-	2
Normal	Normal	972780	60593
SUM		4898430	311029

5.2.1 Features of Data Set

The original TCP dump files were preprocessed for utilization in the Intrusion Detection System benchmark of the International Knowledge Discovery and Data Mining Tools Competition [92]. To do so, packet information in the TCP dump file is summarized into connections. Specifically, a connection is a sequence of TCP packets starting and ending at some well-defined times, between which data flows from a source IP address to a target IP address under some well-defined protocol. This process is completed using the Bro IDS [33], resulting in 41 features for each connection. Features are grouped into three categories:

i. **Basic Features:** This category encapsulates all the attributes that can be extracted from a TCP/IP connection. Most of these features lead to an implicit delay in detection. Basic features can be derived from packet headers without inspecting the payload. (See Table 5.2.)

ii. **Content Features:** unlike most of the DoS and Probing attacks, the R2L and U2R attacks do not have any intrusion frequent sequential patterns. This is because the DoS and Probing attacks involve many connections to the same host(s) in a very short period of time; however the R2L and U2R attacks are embedded in the data portions of the packets and normally involve only a single connection. To detect these kinds of attacks, one needs some features to be able to look for suspicious behavior in the data portion, e.g., number of failed login attempts. Domain knowledge is used to assess the payload of the original TCP packets. These features are called content features. (See Table 5.3)

iii. **Traffic Features:** This category includes features that are computed with respect to a window interval and is divided into two groups: (See Table 5.4)

    a. **Service-based Traffic Features:** These features are designed to capture properties that mature over a 2 second temporal window. This type of features examines only the connections in the past 2 seconds that have the same service as the current connection.

b. **Host-based Traffic Features:** These features utilize a historical window estimated over the number of connections – in this case 100 – instead of time. Host based features examine only the connections in the past 2 seconds that have the same destination host as the current connection and calculate statistics related to protocol behavior, service, etc.

Table 5.2: Basic Features of Individual TCP Connections.

feature name	**description**	**type**
duration	length (number of seconds) of the connection	continuous
protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
service	network service on the destination, e.g., http, telnet, etc.	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong_fragment	number of ``wrong'' fragments	continuous
urgent	number of urgent packets	continuous

Table 5.3: Content Features within a Connection Suggested by Domain Knowledge.

feature name	**description**	**type**
hot	number of ``hot'' indicators	continuous
num_failed_logins	number of failed login attempts	continuous
logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	number of ``compromised'' conditions	continuous
root_shell	1 if root shell is obtained; 0 otherwise	discrete
su_attempted	1 if ``su root'' command attempted; 0 otherwise	discrete
num_root	number of ``root'' accesses	continuous
num_file_creations	number of file creation operations	continuous
num_shells	number of shell prompts	continuous
num_access_files	number of operations on access control files	continuous
num_outbound_cmds	number of outbound commands in an ftp session	continuous
is_hot_login	1 if the login belongs to the ``hot'' list; 0 otherwise	discrete
is_guest_login	1 if the login is a ``guest''login; 0 otherwise	discrete

Table 5.4: Traffic Features Computed Using a Two-second Time Window.

feature name	**description**	**type**
count	number of connections to the same host as the current connection in the past two seconds	continuous
serror_rate	% of connections that have ``SYN'' errors (host based)	continuous
rerror_rate	% of connections that have ``REJ'' errors (host based)	continuous
same_srv_rate	% of connections to the same service (host based)	continuous
diff_srv_rate	% of connections to different services (host based)	continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
srv_serror_rate	% of connections that have ``SYN'' errors (service based)	continuous
srv_rerror_rate	% of connections that have ``REJ'' errors (service based)	continuous
srv_diff_host_rate	% of connections to different hosts (service based)	continuous

In the International Knowledge Discovery and Data Mining Tools Competition, only “10% KDD” dataset is employed for the purpose of training. This dataset contains 22 attack types and out of which DOS attack dataset collected. It contains more examples of attacks than normal connections and the attack types are not represented equally. Because of their nature, denial of service attacks account for the majority of the dataset.

In order to further simulate what may be encountered in the real world, additional attack types were added to the test data that were not found in the training data, in order to help determine which challenging intrusion detection systems were able to generalize to new attacks that may not have the same signature or behavior as previously seen training categories.

5.3 DATA SET CORRECTION AND PREPARATION

Conducting a thorough analysis of the recent research trend in anomaly detection, one will encounter several machine learning methods reported to have a very high detection rate of 98% while keeping the false alarm rate at 1% [93]. However, when we look at the state of the art IDS solutions and commercial tools, there are a few products using anomaly detection approaches. Practitioners still believe that it is not a mature technology yet. To find the reason of this contrast, we studied the details of the research done in anomaly detection and considered various aspects such as learning and detection approaches, training data sets, testing data sets, and evaluation methods

Recent studies shows that there are some inherent problems in the KDD data set [89], which is widely used as one of the few publicly available data sets for network-based anomaly detection systems. Some statistical flaws were found in the original dataset that cause bias in training and testing algorithms [94], causing a skewed distribution of simulated attack types to bias the results. Due to these issues, correction and preparation need came out on KDD dataset before running algorithms.

5.3.1 Duplicate Records

One of the most important deficiencies in the KDD data set is the huge number of redundant records, which causes the learning algorithms to be biased towards the frequent records, and thus prevent them from learning infrequent records, which are usually more harmful to networks such as U2R and R2L attacks [114]. In addition, the existence of these repeated records in the test set will cause the evaluation results to be biased by the methods which have better detection rates on the frequent records.

To solve this issue, we removed all the repeated records in the entire KDD train and test set, and kept only one copy of each record. Tables 5.5 and 5.6 illustrate the statistics of the reduction of repeated records in the KDD train and test sets, respectively.

Table 5.5: Statistics of Redundant Records in the KDD Train Set.

	Original Records	**Distinct Records**
Attacks	3.925.650	262.178
Normal	972.780	812.814
Total	4.898.430	1.074.992

Table 5.6: Statistics of Redundant Records in the KDD Test Set.

	Original Records	**Distinct Records**
Attacks	250.436	29.378
Normal	60.591	47.911
Total	311.027	77.289

5.3.2 Data Size

The corrected KDD data set still has a very large number of records. In KDD Cup due to hardware requirements they used %10 of training set. We also needed to reduce the size of data sets because of the same reasons, therefore we selected 126135 records from corrected KDD training data set as shown in Table 5.7 and named it as Improved KDD Training Data Set. We did this homogeneously by considering the attack type ratio. We used holdout validation in our experiments in Chapter 6 and according to the findings of Kearns [95], for holdout validation, the data ratio was 80% for training and the %20 for testing. We selected 25184 records from corrected KDD test data set homogeneously and created Improved KDD Test Data Set, The proportions of attack classes in Improved KDD data set shown in Table 5.8.

Table 5.7: Statistics of Selected Records from Corrected KDD Data Set.

	Distinct Records	**Selected Records**
KDD train set	1.074.992	126.135
KDD test set	77.289	25.184

Table 5.8: Proportions of Attack Classes in Improved KDD Data Set.

Category	Attack Name	Training Data Set	Corrected Training Set	Improved Training Set	Test Data Set	Corrected Test Set	Improved Test Set
U2R	buffer_overflow	30	30	30	22	22	22
U2R	loadmodule	9	9	9	2	2	2
U2R	perl	3	3	3	2	2	2
U2R	ps	-	-	-	16	16	16
U2R	rootkit	10	10	11	13	13	13
U2R	sqlattack	-	-	-	2	2	2
U2R	xterm	-	-	-	13	13	13
R2L	ftp_write	8	8	8	3	3	3
R2L	guess_passwd	53	53	53	4367	2847	326
R2L	httptunnel	-	-	-	158	158	158
R2L	imap	12	12	12	1	1	1
R2L	multihop	7	7	7	18	18	18
R2L	named	-	-	-	17	17	17
R2L	phf	4	4	5	2	2	2
R2L	sendmail	-	-	-	17	17	17
R2L	snmpgetattack	-	-	-	7741	4378	756
R2L	snmpguess		-	-	2406	1526	452
R2L	spy	2	2	2	-	-	-
R2L	warezclient	1020	894	894	-	-	-
R2L	warezmaster	20	20	20	1602	984	984
R2L	worm	-	-	-	2	2	2
R2L	xlock	-	-	-	9	9	9
R2L	xsnoop	-	-	-	4	4	4
Probing	ipsweep	12481	10345	1834	306	169	169
Probing	mscan	-	-	-	1053	631	631
Probing	nmap	2316	1812	381	84	84	84
Probing	portsweep	10413	7437	1543	354	-	-
Probing	saint	-	-	-	736	448	449
Probing	sendmail	15892	9739	1273	1633	-	-
DoS	apache2	-	-	-	794	328	328
DoS	back	2203	639	639	1098	476	476
DoS	land	21	21	21	9	9	9
DoS	mailbomb	-	-	-	5000	1209	420
DoS	neptune	1072017	52126	8365	58001	4752	1075
DoS	pod	264	264	264	87	87	87
DoS	processtable	-	-	-	759	419	419
DoS	smurf	2807886	316647	42703	164091	11249	2124
DoS	teardrop	979	630	630	12	12	12
DoS	udpstorm	-	-	-	2	2	2
Normal	Normal	972780	674280	67428	60593	47378	16080
SUM		4898430	1074992	126135	311029	77289	25184

5.3.3 Preparation Summary

Our Improved KDD data set has some improvements over the original KDD data set as outlined below:

i. It does not include redundant records in the train set, therefore the classifiers will not be biased towards more frequent records.

ii. There are no duplicate records in the test set; therefore, the performance of the learners is not biased by the methods which have better detection rates on the frequent records.

iii. The number of records in the training and testing sets is reasonable, which makes it reasonable to run the experiments.

The resulting dataset removed redundant records from both the train and test set in order to remove the bias towards frequent and repetitive net flows typical in an artificial network environment, allowing machine learning algorithms to observe traffic more typically found in a spontaneous and ever-changing real world networking environment, and hence improve the utility of the learned models, and the evaluation of potential commercial products.

5.4 EVALUATION METHODOLOGY

The evaluation of machine learning algorithms on the DARPA attack data can be done a number of ways. The following subsections explain how they were configured for use with the datasets we generated, the framework and environment in which they were trained and tested.

5.4.1 Test Platform

The experiments described below were performed on a computer with hardware and software specifications shown in Table 5.9.

Table 5.9: Test Platform Specifications.

Feature	**Value**
Operating System	Windows Server 2012 R2
Processor	Intel Xeon E5504 2.00 GHz
Memory	8 Gigabytes
Software	Weka 3.6.10

5.4.2 Weka

Weka is a practical machine learning tools package [96]. “Weka” stands for the Waikato Environment for Knowledge Analysis, which was developed at the University of Waikato in New Zealand. Weka is extensible and has become a collection of machine learning algorithms for solving real-world data mining problems. It is written in Java and runs on almost every platform. Weka is easy to use and to be applied at several different levels. One can access the Weka class library from his own Java program, and implement new machine learning algorithms.

Weka supports several standard data mining tasks, more specifically, data preprocessing, clustering, classification, regression, visualization, and feature selection. All of Weka's techniques are predicated on the assumption that the data is available as a single flat file or relation, where each data point is described by a fixed number of attributes (normally, numeric or nominal attributes, but some other attribute types are also supported). Weka provides access to SQL databases using Java Database Connectivity and can process the result returned by a database query. It is not capable of multi-relational data mining, but there is separate software for converting a collection of

linked database tables into a single table that is suitable for processing using Weka [97]. Another important area that is currently not covered by the algorithms included in the Weka distribution is sequence modeling.

Weka's main user interface is the Explorer, but essentially the same functionality can be accessed through the component-based Knowledge Flow interface and from the command line. There is also the Experimenter, which allows the systematic comparison of the predictive performance of Weka's machine learning algorithms on a collection of datasets.

Weka permits the input data set to be in numerous file formats like CSV (comma separated values: *.csv), Binary Serialized Instances (*.bsi) etc. However, the most preferred and the most convenient input file format is the attribute relation file format (arff). So the first step in Weka always is taking an input file and making sure that it is in ARFF.

5.4.3 Algorithm Configuration

There are a variety of machine learning and data mining-based classification algorithms that have been used in this problem domain. In order to cover a broad range of learning types, we have sampled algorithms from Logistic Regression [98], Naive Bayes Classifier [99], Support Vector Machines [21], K-nearest-neighbor Algorithm [100, 32], and Artificial Neural Networks [44, 88, 101].

The following sections describe the parameters used in each algorithm:

5.4.3.1 Naive Bayes

The Naive Bayes algorithm in Weka [102], was run against each dataset using the default settings such that it compares the frequency across all attributes to the class without assuming any conditional probabilities between classes, in other words complete independence.

5.4.3.2 J48 Decision Tree

The J48 Decision Tree implemented in Weka [102], is an open source version of C4.5, and was run with the confidence factor set to 0.25, and the minimum number of instances per leaf set to 2.

5.4.3.3 Logistic Regression

Evaluation of logistic regression was done using the Logistic library [102], which is a standard implementation of the sigmoid function training through gradient descent.

5.4.3.4 K-Nearest Neighbour

Evaluation of the instance-based learning algorithm was performed using the iBk library in Weka [102], which is a standard implementation of the K-NN training through calculating nearest points in an n-dimensional Euclidean plane. In this experiment, the independent variable k, being the number of grouped points involved in the voting process was initialized to default value 1 for initial training and testing, with no distance weighting for penalization.

5.4.3.5 Support Vector Machines

The Sequential Minimal Optimization (SMO) algorithm in Weka [103], was used for the support vector machine experiments. Considering the hyperplane used to classify each parameter may be non-linear, we employed a radial basis function kernel machine, as proposed by Aizerman et al. [104], to improve the classification boundary.

5.4.3.6 Artificial Neural Networks

Evaluation of the Multilayer Perceptron (MLP) algorithm was done using the Multilayer Perceptron library in Weka [102], which is a standard implementation of the sigmoid perceptron using Least Mean Squared cost calculation, and gradient descent back-propagation.

The number of input nodes corresponds to the number of features in each training set. Only one output node is required for anomaly detection. The learning rate

used was 0.3, and the learning momentum was 0.2. Back-propagation was cycled for 500 epochs per learning example.

5.4.3.7 Genetic Programming

The Genetic Programming (GP) algorithm in Weka [46], was run against the dataset using the default settings that elite size set to 0.5 population size and new population size set to 100.

5.4.4 Validation Method

A number of validation techniques have been developed over time, such as Resubstitution Validation, Hold-Out Validation, K- Fold Cross-Validation, Leave-One-Out Cross-Validation, and Repeated K-Fold Cross- Validation.

The most widely used form of cross validation for model parameter tuning is k-fold cross validation. Cross Validation is a method used to compare the performance of two or more machine learning algorithms, while relying on the same pool of labeled test set data for both training and testing, yet doing so in a statistically significant manner [105].

The application of cross validation techniques to real world problems can be driven by multiple goals. Some of these goals include algorithm performance estimation, tuning model parameters, and ultimately model selection [105].

Hold-out validation [105] is an alternate method for algorithm testing, typically used on algorithms whose parameters have been previously fixed. The premise behind this testing is that training and preliminary testing will have already occurred (typically through a training and validation set) with a holdout dataset reserved for real world prediction estimation. The advantages of this method include a purely independent training and testing dataset, however this can be an issue when dealing with already small data sets as well as a large variance in estimated performance.

Depren et al. [106] state that cross validation is preferred when there is limited data available. From the range of experiments conducted, the results obtained with

holdout validation appear deceptive in some cases compared with the results obtained with cross validation seeming to be very sensitive to the class balance and duplicates.

According to the empirical findings obtained here, the results with cross validation appear more positive than those obtained with holdout validation. This was found to be due to holdout validation being sensitive to the training/test split, whilst cross validation is more robust since it provides an average from partitions (folds) so cross validation may not be an appropriate validation method for intrusion detection since it is then not possible to explicitly evaluate the detection of new attacks. By considering this holdout validation was used in the KDD competition, in which the data was split into a training and test set. In our study we performed holdout validation and shared the results in Chapter 6.

5.5 EVALUATION METRICS

Considering each classifier, the experiments measure these dependent variables through the use of receiver operative curve (ROC), and precision and recall measurements, in order to determine the overall effectiveness of each algorithm. The evaluation types will be outlined below.

5.5.1 Performance

The experiment observation is described in terms of classification performance, or accuracy. Classification performance in this context is measured in terms of true-positive, false-positive, true-negative, and false-negative. These dependent measurements are typically performed in terms of percentage of data sets identified over all possible sets to be identified correctly, a measurement applicable to classification is the concept of precision and recall, which is used in the ROC calculation as well as additional analysis [107].

5.5.2 Receiver Operating Characteristic

The relative operating curve (ROC) Farid and Rahman has been used to compare the relationship between detection and false positive rates [36]. The deciding metric

used to measure overall accuracy was the mean Area under the ROC. Precision, Recall, and Specificity of each algorithm was dependent on which of the classes was considered for the positive and negative label. The Mean AUC, as used in the performance comparison, considers both scenarios, and takes the average of the two values. The observed AUC deviation between the classes was minimal or non-existent in most cases, and hence the AUC mean was chosen as an all-encompassing measurement such that the positive and negative class designations were not an issue.

5.5.3 Training and Testing Time

Each of the algorithms was tested for overall training and test time. The absolute times are highly dependent on the implementation platform, however the relative times are a strong indicator of relative algorithm performance. These metrics were analyzed according to their contribution for each algorithm. In practice, the training time is not as important as the testing time. Model creation in this case would be performed offline using already labeled data, whereas testing would be on-line and produces an events-per-second metric that is a major consideration in the intrusion detection industry.

CHAPTER 6

EXPERIMENT AND RESULTS

6.1 OVERVIEW

In this chapter we will discuss the algorithm evaluation for a large number of supervised learning algorithms with the KDD dataset in order to establish benchmark measurements for each major type of algorithm found in contemporary Machine Learning. General supervised machine learning algorithms provide a method for building a prediction model using historical data comprised of attribute-classification sets. The evaluation of the various algorithms investigates the performance of each individual learning algorithm, when applied to our data set.

6.2 ZEROR

The ZeroR function in Weka [102], was trained against the training set. Since 50% of the training instances are classified as normal (that is the mode of the nominal classes) ZeroR classifies every test instance as normal. Averaging out over all test folds reveals exactly what one would expect the classifier correctly identifies the same ratio of normal-to-anomaly as the training set, and in this case is correct roughly more than half the time; a truly sub-optimal performing classifier.

This prediction model was trained in 0.45 second, and tested against the test set in 0.72 second, using the platform we described.

ZeroR can be evaluated through a confusion matrix. Considering the anomaly dataset, there are two classes; normal and anomaly. Considering all classes in a row by column matrix, this will produce a 2x2 matrix. The confusion matrix produced is shown in Table 6.1.

Table 6.1: ZeroR Confusion Matrix.

a	**b**	**classified as**
16080	0	a = normal
9104	0	b= anomaly

In this case, the first row represents all of the normal classified test instances and each column represents how those instances were classified. In the case of Zero-R, because the algorithm defaults all instance classifications to the class that is seen the most often in training instances, there are only true positive, and no misclassifications as the “b” class. The training set, as seen earlier, had a larger number of normal class instances, and thus is the class Zero-R will always select. Since the test set has more anomalies than normal, the classifier performs worse than guessing (50%), and is obviously a poor choice. Detailed accuracy by class is shown in Table 6.2, the mean AUC was 0.5 and the Receiver Operating Characteristic (ROC) curve for normal and anomaly classes are shown in Figures 6.1 and 6.2. The x-axis attributes False Positive Rate and the y-axis attributes True Positive Rate.

Table 6.2: ZeroR Detailed Accuracy by Class.

TP Rate	**FP Rate**	**Precision**	**Recall**	**F-Measure**	**ROC Area**	**Class**
1	1	0.639	1	0.779	0.5	normal
0	0	0	0	0	0.5	anomaly
0.639	0.639	0.408	0.639	0.498	0.5	**Weighted Avg.**

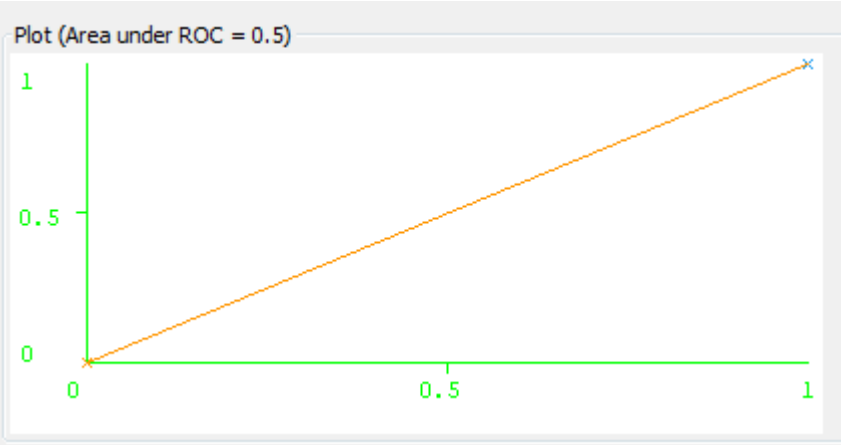

Figure 6.1: ZeroR ROC Curve for Normal Class.

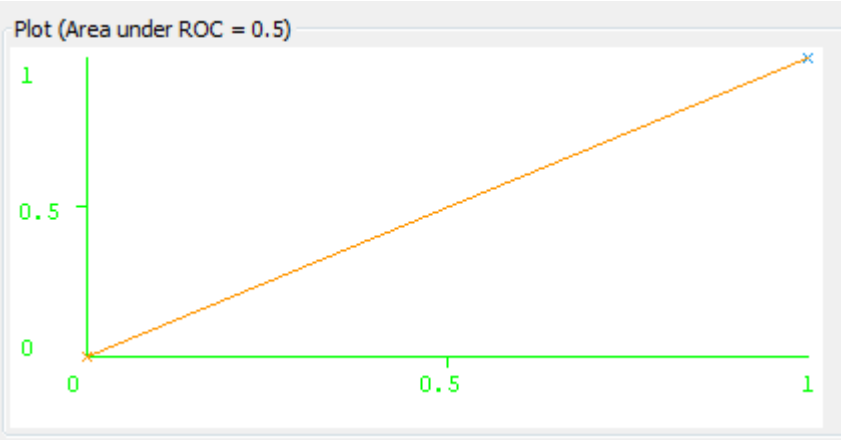

Figure 6.2: ZeroR ROC Curve for Anomaly Class.

The ZeroR algorithm, as simple as it is, does provide us with a worst-case baseline; using the testing set provided demonstrates that blindly selecting the class based on the majority of seen classes in the training set can be worse than guessing, for example when the majority of training classes becomes the minority in the test dataset.

6.3 NAIVE BAYES

The Naive Bayes algorithm in Weka [102], was run against the dataset using the default settings required such that it compares the frequency across all attributes to the class without assuming any conditional probabilities between classes, in other words, complete independence. An overview of runtime performance is explained below.

The Naive Bayes algorithm was run against the dataset, and took 3 seconds to build, and 0.84 seconds to run, using the platform we described.

The confusion matrix produced is shown in Table 6.3 and Table 6.4 illustrates the results of the detailed accuracy using Holdout Validation against the data set.

Table 6.3: Naive Bayes Confusion Matrix.

a	**b**	**classified as**
15184	896	a = normal
1856	7248	b= anomaly

Table 6.4: Naive Bayes Detailed Accuracy by Class.

TP Rate	**FP Rate**	**Precision**	**Recall**	**F-Measure**	**ROC Area**	**Class**
0.944	0.204	0.891	0.944	0.917	0.946	normal
0.796	0.056	0.89	0.796	0.84	0.946	anomaly
0.891	0.15	0.891	0.891	0.889	0.946	**Weighted Avg.**

Naïve Bayes algorithm produced a 0.891 average TP rate, and 0.15 FP rate. The average Precision and Recall measurements were 0.891. The mean AUC was 0.946 and the ROC curve for normal and anomaly classes are shown in Figures 6.3 and 6.4, the x-axis attributes False Positive Rate and the y-axis attributes True Positive Rate.

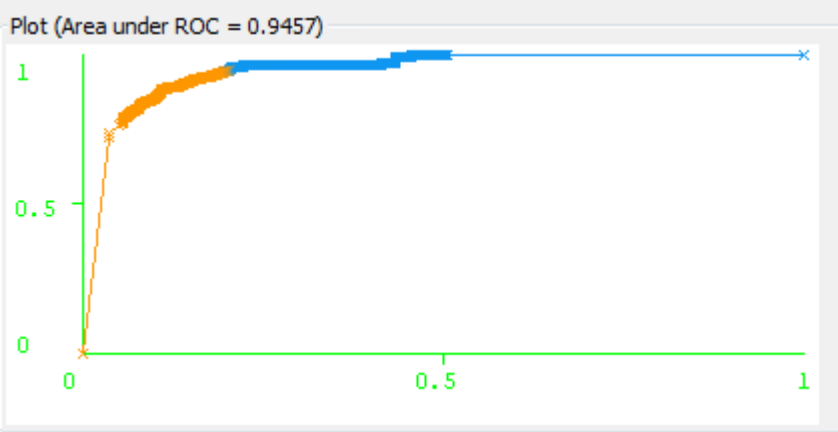

Figure 6.3: Naive Bayes ROC Curve for Normal Class.

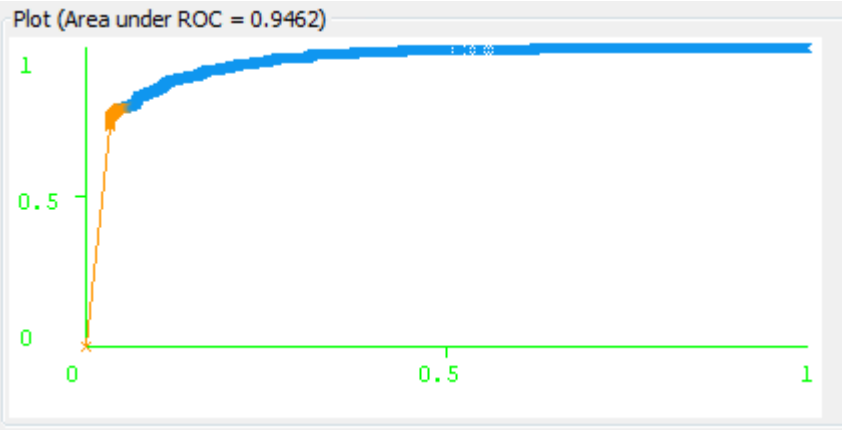

Figure 6.4: Naive Bayes ROC Curve for Anomaly Class.

The overall accuracy and mean AUC values of Naïve Bayes algorithm is much higher than the previously seen ZeroR implementation.

6.4. J48 DECISION TREE

The J48 decision tree implemented in Weka [102], is an open source version of C4.5, and was run with the confidence factor set to 0.25, and the minimum number of instances per leaf set to 2.

The J48 algorithm was run against the dataset, took 32 seconds to build, and 1.07 seconds to run, using the platform we described.

Table 6.5: J48 Confusion Matrix.

a	**b**	**classified as**
15936	144	a = normal
672	8432	b= anomaly

Considering the confusion matrix in Table 6.5 and detailed accuracy in Table 6.6, the following performance metrics were produced: the algorithm using the dataset produced a 0.968 average True Positive rate, and 0.05 average False Positive rate. The average Precision and Recall measurements were 0.968.

Table 6.6: J48 Detailed Accuracy by Class.

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.991	0.074	0.96	0.991	0.975	0.958	normal
0.926	0.009	0.983	0.926	0.954	0.958	anomaly
0.968	0.05	0.968	0.968	0.967	0.958	**Weighted Avg.**

The mean AUC was 0.958 and the ROC curve for normal and anomaly classes are shown in Figures 6.5 and 6.6. The x-axis attributes False Positive Rate and the y-axis attributes True Positive Rate. The values are much higher than the Naive Bayes results.

Figure 6.5: J48 ROC Curve for Normal Class.

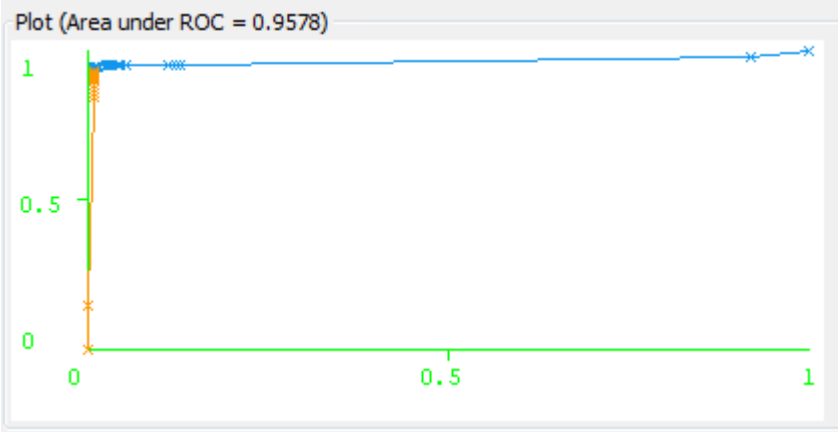

Figure 6.6: J48 ROC Curve for Anomaly Class.

6.5 LOGISTIC REGRESSION

Evaluation of the logistic regression algorithm was done using the Logistic library in Weka [102], which is a standard implementation of the sigmoid function training through gradient descent.

The Logistic Regression algorithm was run against the dataset, and took 19 seconds to build, and 0.67 seconds to test, using the platform we described.

Table 6.7: Logistic Regression Confusion Matrix.

a	**b**	**classified as**
15264	816	a = normal
928	8176	b= anomaly

Considering the confusion matrix in Table 6.7, the following performance metrics were produced as shown in Table 6.8, the algorithm using the dataset produced a 0.931 average True Positive rate, and 0.083 average False Positive rate. The average Precision and Recall measurements were 0.931.

Table 6.8: Logistic Regression Detailed Accuracy by Class.

TP Rate	**FP Rate**	**Precision**	**Recall**	**F-Measure**	**ROC Area**	**Class**
0.949	0.102	0.943	0.949	0.946	0.938	normal
0.898	0.051	0.909	0.898	0.904	0.938	anomaly
0.931	0.083	0.931	0.931	0.931	0.938	**Weighted Avg.**

The mean AUC was 0.938 and the ROC curve for normal and anomaly classes were shown in Figures 6.7 and 6.8. The x-axis attributes False Positive Rate and the y-axis attributes True Positive Rate. The values are higher than the Naive Bayes results but lower than the J48 values.

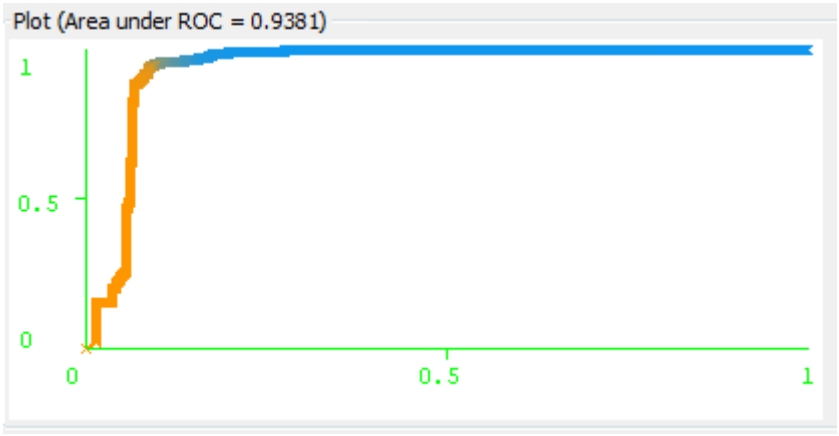

Figure 6.7: Logistic Regression ROC Curve for Normal Class.

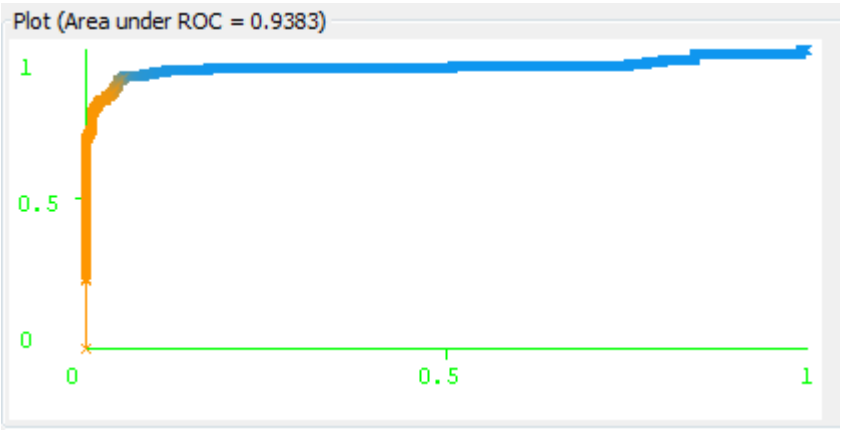

Figure 6.8: Logistic Regression ROC Curve for Anomaly Class.

6.6 K-NN

Evaluation of the K-Nearest Neighbour algorithm was performed using the iBk library in Weka [102], which is a standard implementation of the K-NN training through calculating nearest points in an n-dimensional Euclidean plane. In this experiment, the independent variable k, being the number of grouped points involved in the voting process, was initialized to default value 1 for initial training and testing, with no distance weighting for penalization.

The iBk algorithm was run against the Standard dataset, and took 0.03 seconds to build, and 3 minutes 7 seconds to test using the platform we described.

Table 6.9: iBK Confusion Matrix.

a	b	classified as
15824	256	a = normal
752	8352	b= anomaly

Considering the confusion matrix in Table 6.9, the following performance metrics were produced as shown in Table 6.10, the algorithm using the dataset produced a 0.96 average True Positive rate, and 0.058 average False Positive rate. The average Precision and Recall measurements were 0.96.

Table 6.10: iBK Detailed Accuracy by Class.

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.984	0.083	0.955	0.984	0.969	0.95	normal
0.917	0.016	0.97	0.917	0.943	0.95	anomaly
0.96	0.058	0.96	0.96	0.96	0.95	**Weighted Avg.**

The mean AUC was 0.95 and the ROC curve for normal and anomaly classes are shown in Figures 6.9 and 6.10. The x-axis attributes False Positive Rate and y-axis attributes True Positive Rate. The values are higher than the Naive Bayes and Logistic regression results but lower than the J48 values. The iBk training and testing times were slightly different than the previously seen classifiers.

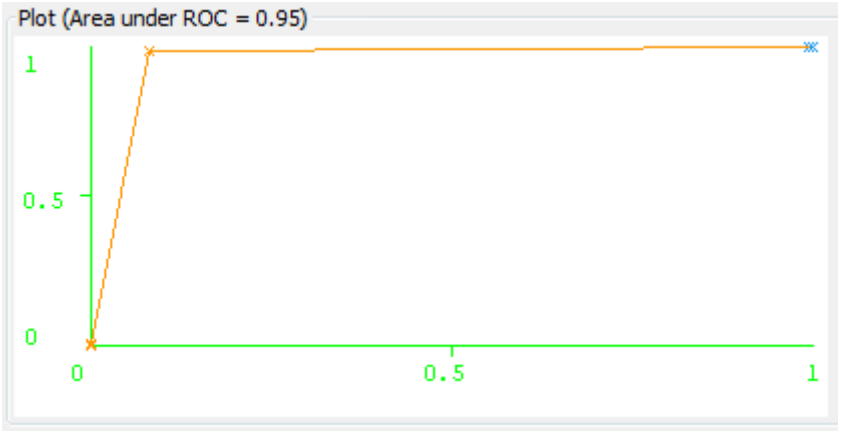

Figure 6.9: iBK ROC Curve for Normal Class.

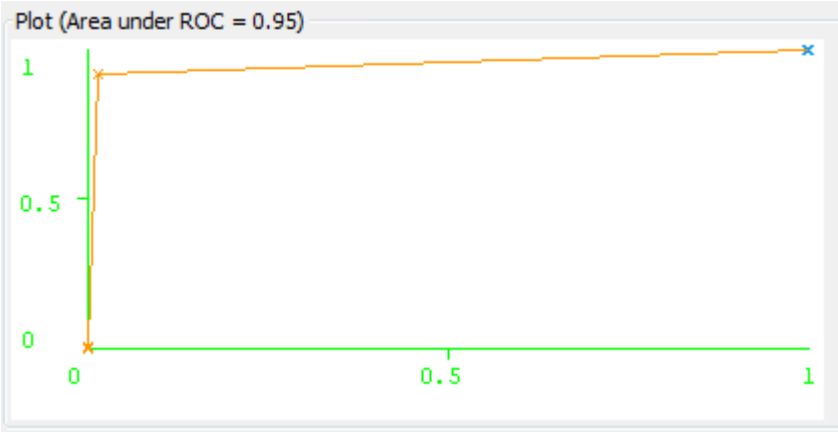

Figure 6.10: iBK ROC Curve for Anomaly Class.

6.7 SUPPORT VECTOR MACHINES

The Sequential Minimal Optimization (SMO) algorithm in Weka [103], was used for the support vector machine experiments. Considering the hyperplane used to classify each parameter may be non-linear, we employed a radial basis function kernel machine as proposed by Aizerman et al. to improve the classification boundary [104].

The Sequential Minimal Optimization (SMO) algorithm was run against the dataset, and took 1 hour, 56 minutes, and 19 seconds to build the model, and 3 minutes, 52 seconds to test the model, using the platform we described.

Table 6.11: SMO Confusion Matrix.

a	**b**	**classified as**
15520	560	a = normal
1200	7904	b= anomaly

Considering the confusion matrix in Table 6.11, the following performance metrics were produced as shown in Table 6.12, the algorithm, using the standard dataset, produced a 0.93 average True Positive rate, and 0.097 average False Positive rate. The average Precision and Recall measurements were 0.93.

Table 6.12: SMO Detailed Accuracy by Class.

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.965	0.132	0.928	0.965	0.946	0.917	normal
0.868	0.035	0.934	0.868	0.9	0.917	anomaly
0.93	0.097	0.93	0.93	0.93	0.917	**Weighted Avg.**

The mean AUC was 0.917 and the ROC curve for normal and anomaly classes are shown in Figures 6.11 and 6.12. The x-axis attributes False Positive Rate and the y-axis attributes True Positive Rate. The values are higher than the Naive Bayes and close to Logistic regression results but lower than the J48 and iBK values. The SMO training and testing times were much higher than the previously seen classifiers.

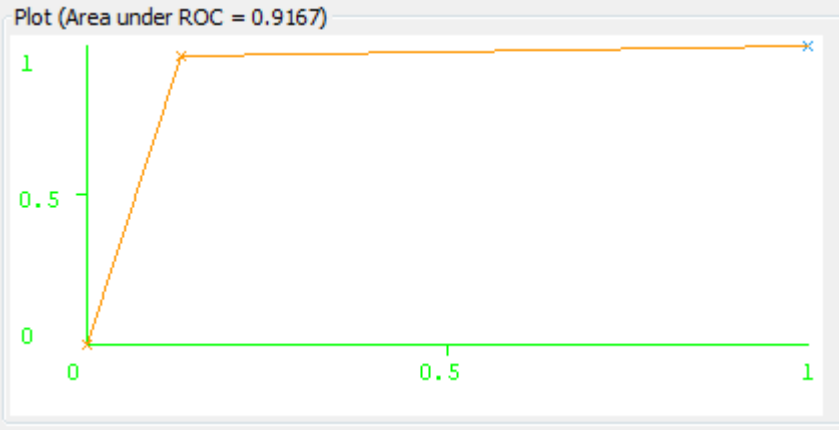

Figure 6.11: SMO ROC Curve for Normal Class.

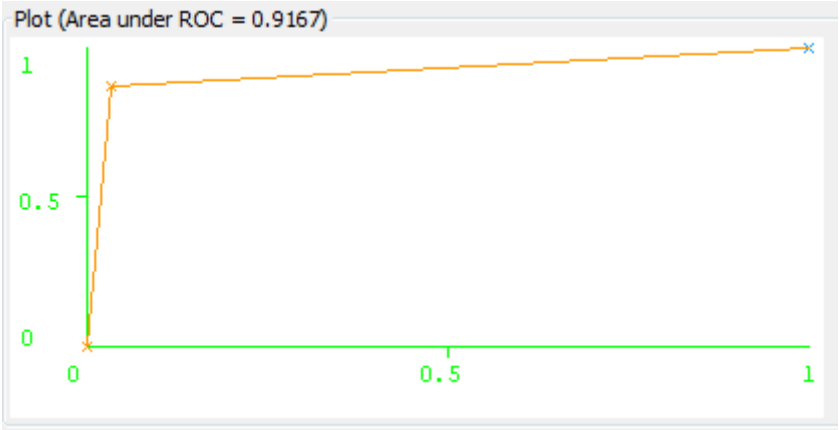

Figure 6.12: SMO ROC Curve for Anomaly Class.

6.8 ARTIFICIAL NEURAL NETWORKS

Evaluation of the Multilayer Perceptron (MLP) algorithm was done using the Multilayer Perceptron library in Weka [102], which is a standard implementation of the sigmoid perceptron using Least Mean Squared cost calculation, and gradient descent back-propagation.

The number of input nodes corresponds to the number of features in each training set. Only one output node is required for anomaly detection. The learning rate used was 0.3, and the learning momentum was 0.2. Back-propagation was cycled for 500 epochs per learning example.

The Multilayer Perceptron algorithm was run against the Standard dataset, and took 1 hour, 58 minutes, and 38 seconds to build the model, and 9 seconds to test the model, using the platform we described.

Table 6.13: MLP Confusion Matrix.

a	**b**	**classified as**
15296	784	a = normal
880	8224	b= anomaly

Considering the confusion matrix in Table 6.13, the following performance metrics were produced as shown in Table 6.14. The multilayer perceptron algorithm, using the dataset, produced a 0.934 average True Positive rate, and 0.079 average False Positive rate. The average Precision and Recall measurements were 0.934.

Table 6.14: MLP Detailed Accuracy by Class.

TP Rate	**FP Rate**	**Precision**	**Recall**	**F-Measure**	**ROC Area**	**Class**
0.951	0.097	0.946	0.951	0.948	0.951	normal
0.903	0.049	0.913	0.903	0.908	0.951	anomaly
0.934	0.079	0.934	0.934	0.934	0.951	**Weighted Avg.**

The mean AUC was 0.951 and the ROC curve for normal and anomaly classes are shown in Figures 6.13 and 6.14. The x-axis attributes False Positive Rate and the y-axis attributes True Positive Rate. The values are higher than the Naive Bayes, Logistic

and SMO but lower than the J48 and iBK values. It can be seen that the Multilayer Perceptron network has a significant degradation in training time.

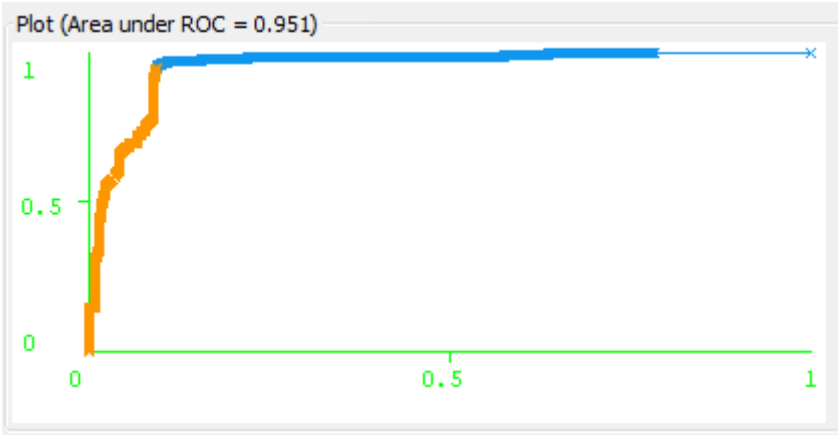

Figure 6.13: MLP ROC Curve for Normal Class.

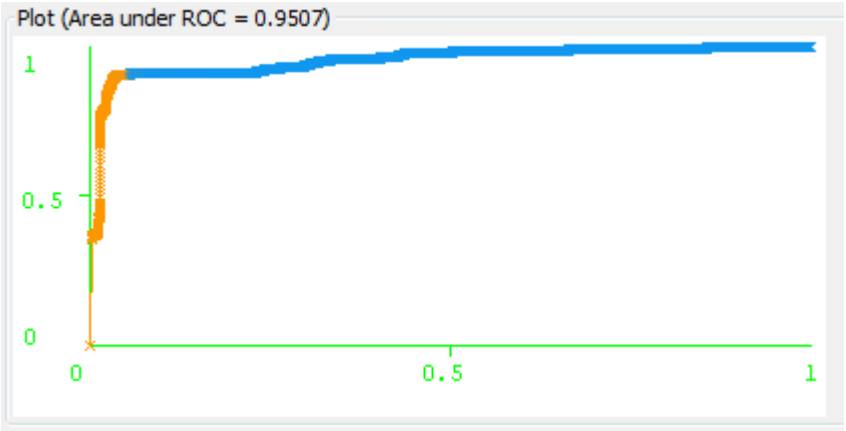

Figure 6.14: MLP ROC Curve for Anomaly Class.

6.9 GENETIC PROGRAMMING

The Genetic Programming (GP) algorithm in Weka [46], was run against the dataset using the default settings including size of the elite population set of 0.5, and population size and new population size set to 100.

The Genetic Programming (GP) algorithm was run against the dataset, and took 2 minutes, 29 seconds to build the model, and 6 seconds to test the model, using the platform we described.

Table 6.15: GP Confusion Matrix.

a	b	classified as
15600	480	a = normal
2032	7072	b= anomaly

Considering the confusion matrix in Table 6.15, the following performance metrics were produced as shown in Table 6.16. The Genetic Programming algorithm, using the dataset, produced a 0.901 average True Positive rate, and 0.154 average False Positive rate. The average Precision and Recall measurements were 0.903 and 0.901 respectively.

Table 6.16: GP Detailed Accuracy by Class.

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.970	0.223	0.885	0.970	0.925	0.873	normal
0.777	0.030	0.936	0.777	0.849	0.873	anomaly
0.901	0.154	0.903	0.901	0.898	0.873	**Weighted Avg.**

The mean AUC was 0.873 and the ROC curve for normal and anomaly classes are shown in Figures 6.15 and 6.16. The x-axis attributes False Positive Rate and the y-axis attributes True Positive Rate. The values are higher than the Naive Bayes, but lower than the other classifier values.

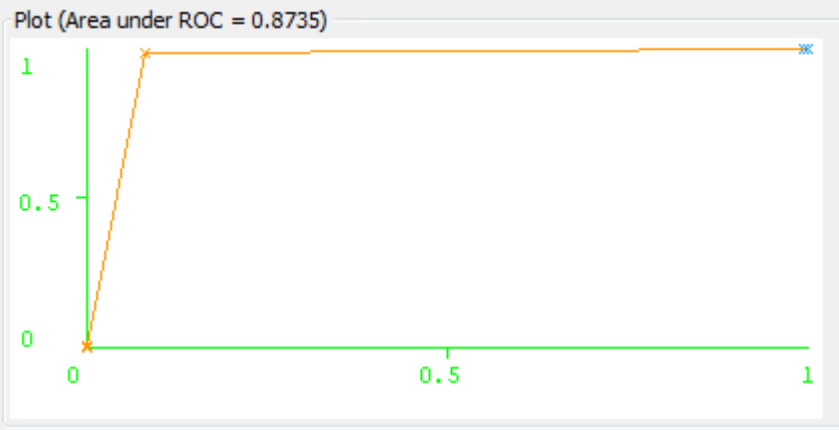

Figure 6.15: GP ROC Curve for Normal Class.

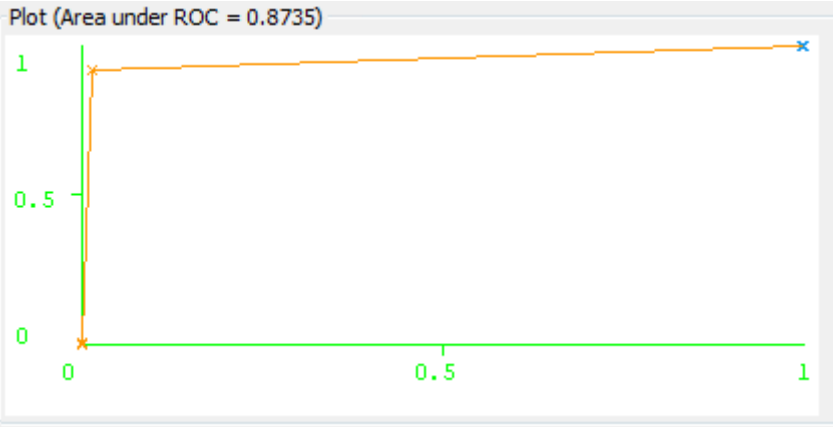

Figure 6.16: GP ROC Curve for Anomaly Class.

6.10 COMPARISON OF RESULTS

The previous sections in this chapter have discussed the performance of each algorithm. This section will take a more holistic view, comparing the performance of the algorithms to one another, in terms of overall testing and training times as well as accuracy.

6.10.1 Training Time Comparison

The training times in seconds for each of the algorithms are given in Table 6.17 and illustrated in Figure 6.17.

Table 6.17: Training Time in seconds.

Algorithm	ZeroR	Naive Bayes	J48	Logistic	iBK	SMO	MLP	GP
Training Time	0.45	3.00	32.00	19.00	0.03	6979.00	7118.00	149.00

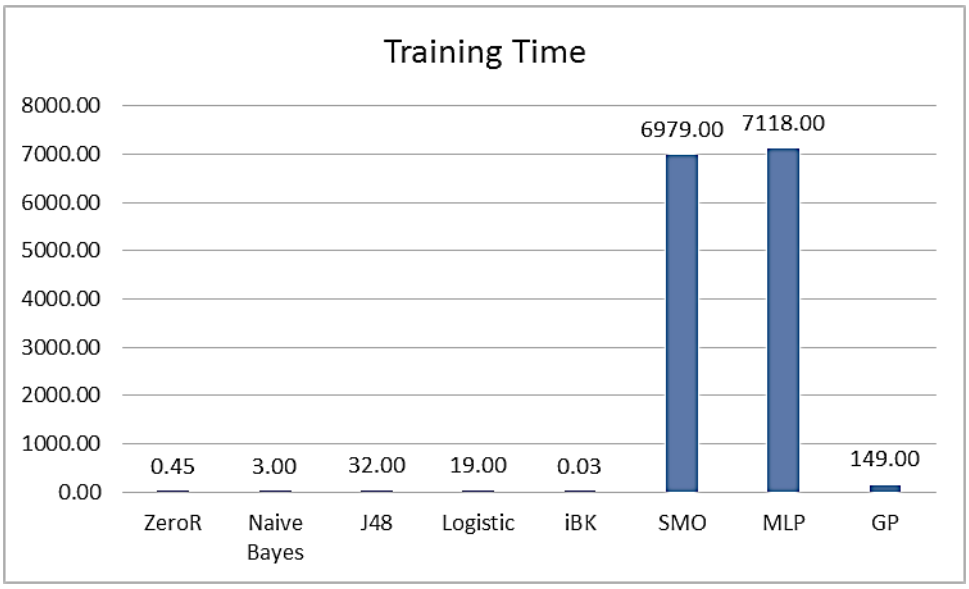

Figure 6.17: Training Time in seconds.

The algorithm training times are widely different. As expected, the more complex algorithms such as artificial neural networks and support vector machines require more time to train than less complex algorithms such as decision trees or logistic regression. Based on the results of our experiments, the iBk (K-Nearest Neighbour) learner had the fastest learner build time (0.03 seconds).

6.10.2 Testing Time

The testing times in seconds for each of the algorithms are given in Table 6.18 and illustrated in Figure 6.18.

Table 6.18: Testing Time in seconds.

Algorithm	ZeroR	Naive Bayes	J48	Logistic	iBK	SMO	MLP	GP
Testing Time	0.72	0.84	1.07	0.67	187.00	232.00	9.00	6.00

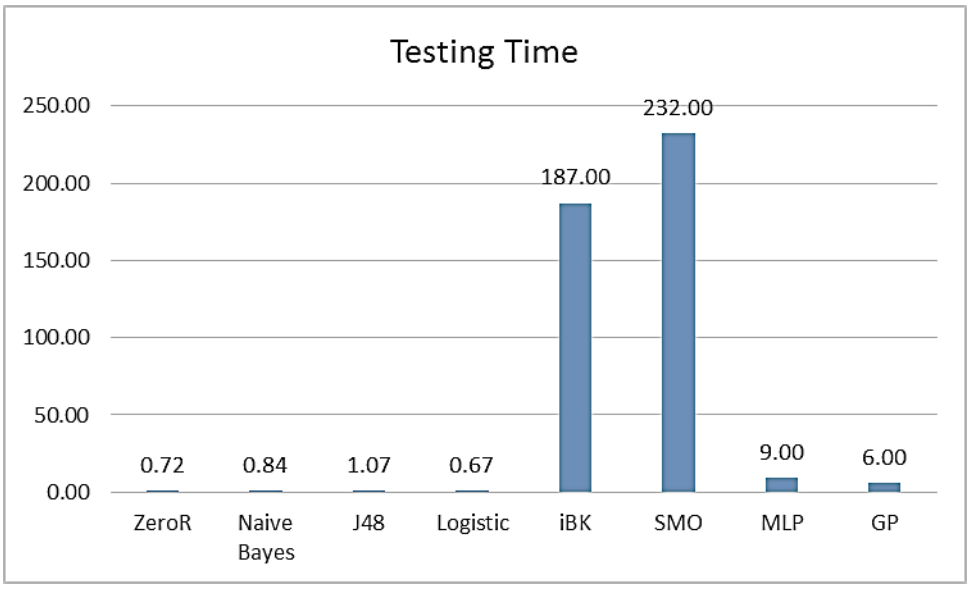

Figure 6.18: Testing Time in seconds.

As with the training times, the algorithm testing times are also widely distributed. The previously more expensive learning algorithms, due to their quick access (decision tree) and propagation (neural network), are somewhat faster. Based on the results of our experiments, the Logistic Regression learner had the fastest learner test time (0.67 seconds).

6.10.3 AUC Comparison

The ROC AUC values for each of the algorithms values are given in Table 6.19 and illustrated Figure 6.19.

Table 6.19: AUC values.

Algorithm	ZeroR	Naive Bayes	J48	Logistic	iBK	SMO	MLP	GP
AUC	0.500	0.946	0.958	0.938	0.950	0.916	0.951	0.873

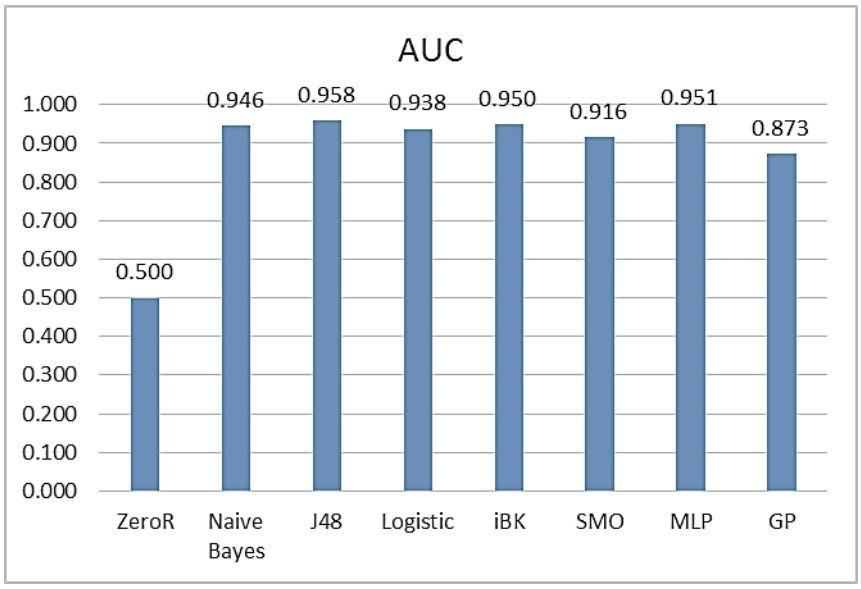

Figure 6.19: AUC values.

The difference in accuracy between algorithms is much less apparent than the differences previously seen in the training and testing times. Based on the results of our experiments, the J48 classifier had the highest AUC (0.958). ZeroR, as expected, is the worst-case lower bound against which all others are compared.

6.11 DETECTION RATE OF ATTACK TYPES

In Chapter 4, we created human threats taxonomy to contribute to the establishment of a behavior based detection model. We classified the human behaviors and defined the attack types of the behaviors in order to define the risk and measure the detection rate of those threats individually. Detection rate is an important factor in determining the risk.

None of the previous taxonomies are directly related with the detection of human threats and additionally, the previous studies did not measure the detection rate of attack types. In our human threats taxonomy, we divided attacks into four classes: Probe, User to Root, Remote to Local, and Denial of Service attacks. According to our experimental results, J48 algorithm had the highest accuracy in anomaly detection therefore we used J48 algorithm for measurement of detection rates. In this section by running the J48 algorithm in Weka [102], we measured the detection performance of attacks by types which we defined in our taxonomy. The mean AUC, True Positive and False Positive Rate values for the each type are shown in Table 6.20.

Table 6.20: Detection Rate of Attack Types.

Attack Type	**AUC**	**TP Rate**	**FP Rate**
Probe	0.968	0.972	0.058
User to Root (U2R)	0.909	0.915	0.107
Remote to Local (R2L)	0.874	0.892	0.141
Denial of Service (DoS)	0.971	0.979	0.049

According to the results in Table 6.20, Denial of Service attacks such as Man in the Middle and Resource Exhaustion threats of our Human Threats Taxonomy ended up with the best detection rate above the average, and Probe type of attacks such as Dangerous Tinkering and Accidentally Allowing threats of the taxonomy ended up with a better detection rate above the average. User to Root types of attacks such as Intentional Destruction, Naïve Mistakes, Personal Usage and Paying Attention threats of the taxonomy had a lower detection rate below the average. Lastly, Remote to Local attacks such as Stealing Privilege, Aware Assurance, Basic Hygiene, and Awareness threats of the taxonomy study had the lowest detection rate.

The results show that detecting User to Root and Remote to Local attacks is more difficult than detecting Denial of Service and Probe types of attacks. The reason behind this lies in the fact that most of the machine learning algorithms offer an acceptable level of classification rate for DoS and Probe attack categories as they exhibit multiple connections over a short period of time while demonstrate a poor performance for the R2L and U2R categories as these attacks are embedded in their data packets itself and do not form a sequential pattern unlike DoS and Probe attacks. This makes their detection by any classifier a difficult task.

6.12 SUMMARY

This chapter reviewed a broad range of machine learning algorithms from the simplistic Zero-R and Naive Bayes to the more sophisticated and expensive Support Vector Machine, Artificial Neural Network and Genetic Programming.

These experiments were performed utilizing a large number of combinations in order to demonstrate a number of factors. The basic confusion matrix was used to derive more intelligent measuring tools such as the True Positive and False Positive rate, the

Precision and Recall, all of which can be combined into one summarized score through the Receiver Operating Curve.

These results produced view of machine learning experiments utilizing the Improved KDD training and testing datasets to predict human threats. The results show that detecting User to Root and Remote to Local attacks is more difficult than detecting Denial of Service and Probe types of attacks.

CHAPTER 7

CONCLUSIONS

7.1 OVERVIEW

In Chapter 1, we described the aim of our study, explained the research problem and scope of our study. In Chapter 2, we presented the problem of Insider Misuse and gave some information about Information Security, Human Factors in Information Security, Attack Types, Intrusion Detection Systems, and Machine Learning Algorithms and discussed detection systems that predicted the occurrence of insider threats by using the Artificial Intelligence techniques. Chapter 3 contains a literature review about the human factors affecting information security in organizations very significantly and human threats prediction techniques. In Chapter 4, we introduced a suitable taxonomy of human threat factors. In Chapter 5, we gave information about KDD data set and explained how we prepared the data set for our experimentation and also explained our experimental methodology including evaluation metrics and algorithm configurations. In Chapter 6, we discussed the algorithm evaluation for all the supervised learning algorithms used with the KDD dataset in order to establish benchmark measurements for each major type of algorithm found in contemporary Machine Learning. These results produced view of machine learning experiments utilizing the KDD training and testing datasets to predict human threats. We measured the detection performance of attacks by types which we defined in our taxonomy.

7.2 CONCLUSIONS

None of the previous taxonomies are directly related with detection of human threats and also the previous studies did not measure the detection rate of attack types. We introduced a most up-to-date taxonomy which aims to encompass today's human threat factors associated to legitimate user actions. We described the impact level of attacks and also described the attack types. We added these attributes because they are significantly related with anomaly detection. We gave examples about the human threats. The taxonomy is tailored to the needs of automated human threat prediction. The establishment of this classification scheme paves the way for the construction of a suitable behavior-based intrusion detection system. Taxonomy is an important milestone for this study because it will enhance the ability to examine the problem in a more systematic way and will eventually contribute to the establishment of behavior-based intrusion detection systems.

We reviewed a broad range of machine learning algorithms from the simplistic Zero-R and Naive Bayes to the more sophisticated and expensive Support Vector Machine, Artificial Neural Network and Genetic Programming.

Based on the results of our experiments, the J48 classifier had the highest AUC (0.958), the iBk (K-Nearest Neighbour) learner had the fastest learner build time (0.03 seconds), and the Logistic Regression learner had the fastest learner test time (0.67 seconds).

The experiment observation is described in terms of classification performance, or accuracy. Classification performance in this context is measured in terms of true-positive, false-positive, true-negative, and false-negative. These dependent measurements are typically performed in terms of percentage of data sets identified over all possible sets to be identified correctly.

The absolute times are highly dependent on the implementation platform, however the relative times are a strong indicator of relative algorithm performance. These metrics were analyzed according to their contribution for each algorithm. In practice, the training time is not as important as the testing time. Model creation in this case would be performed offline using already labeled data whereas testing would be

on-line and produces an events-per-second metric that is a major consideration in the intrusion detection industry.

If we want to choose the best algorithm for intrusion detection we should compare the accuracy and testing time values of the algorithms. In this context J48 is the best algorithm for detection when we compare these values. Because J48 had the highest accuracy and J48 had very fast test time which was very close to the fastest time.

Decision tree algorithm J48 had the best score; to understand why a decision tree algorithm came out with the best score we can look through the following decision tree properties. Decision trees are non-parametric; therefore no specific data distribution is necessary. Decision trees easily handle continuous and categorical variables. Decision trees handle missing values as easily as any normal value of the variable. In decision trees, elegant tweaking is possible. One can choose to set the depth of the trees, the minimum number of observations needed for a split, or for a leave, the number of leaves per split (in case of multilevel target variables). Decision trees run fast even with lots of observations and variables. Decision trees identify subgroups. Each terminal or intermediate leave in a decision tree can be seen as a subgroup/segment of one's population. Decision trees can easily handle unbalanced datasets. If one has 0.1% of positive targets and 99.9% of negative ones, it is not a problem for decision trees [108].

In human threats taxonomy, attacks were divided into four classes as Probe, User to Root, Remote to Local, and Denial of Service attacks. By running the J48 algorithm, which ended up with the best performance in the experiments, we measured the detection performance of attacks by types defined in our taxonomy. According to the experimental results, Denial of Service type of attacks such as Man in the Middle and Resource Exhaustion threats of our Human Threats Taxonomy ended up with the best detection rate. Probe type of attacks such as Dangerous Tinkering and Accidentally Allowing threats of the taxonomy ended up with a better detection rate above the average. User to Root types of attacks such as Intentional Destruction, Naïve Mistakes, Personal Usage and Paying Attention threats of taxonomy had a lower detection rate below the average. Lastly, Remote to Local attacks such as Stealing Privilege, Aware Assurance, Basic Hygiene, and Awareness threats of the extended taxonomy study had the lowest detection rate.

These results show that detecting User to Root and Remote to Local attacks is more difficult than detecting Denial of Service and Probe types of attacks. The reason behind this lies in the fact that most of the machine learning algorithms offer an acceptable level of classification rate for DoS and Probe attack categories as they exhibit multiple connections over a short period of time while demonstrate a poor performance for the R2L and U2R categories as these attacks are embedded in their data packets itself and do not form a sequential pattern unlike DoS and Probe attacks. This makes their detection by any classifier a difficult task.

Detecting User to Root and Remote to Local attack types is more difficult than detecting the other types of attacks due to the low detection rate. Therefore, these types of attacks are more dangerous for organizations. For organizations to prevent these types of attacks from occurring, they should take extra precautions such as keeping the systems up-to-date, avoiding misconfigurations, and applying additional security policies.

7.2.1 Answers of Research Questions

What types of human behaviors can lead to security breaches?

In Chapter 2, we described human behaviors by developing most up-to-date five-factor Human Threats Taxonomy where the factors are intentionality, expertise, impact level, threat layer and attack type.

Which machine learning algorithm has a better detection rate in anomaly detection?

Based on the results of the experiments presented in Chapter 6, J48 had the highest accuracy and J48 had very fast test time which was very close to the fastest time. In this context J48 is the best algorithm for detection.

What types of attacks are more difficult to detect?

According to the experiments, Probe and Denial of Service attacks had better detection rate and User to Root and Remote to Local attacks had lower detection rate. It means that detecting User to Root and Remote to Local attacks is more difficult than detecting the other types of attacks.

7.3 FUTURE WORK

The area of human factor within information security is still not fully explored and needs further investigations although within the past few years there has been a great emphasis on researching social psychology, cognitive science, and neuroscience from the point of information security risks. The interaction between information security and human psychology will be the huge research area within the next years [109], so not only knowledge about how security countermeasures for internal threat are perceived should be researched, but any interaction among security and psychology will help to improve information security.

This study involved the application of a broad range of supervised learning algorithms for the purpose of anomaly detection. These algorithms require an offline training phase, but the testing phase requires much less time and future work could investigate how well it can be adapted to performing online. The main difficulties in adapting these techniques for practical use are the difficulties involved in acquiring labeled training data and in investigating how the training on this dataset can be useful in classifying real datasets.

Our thesis used the default settings for most of the algorithms tested, with a few deviations. Every algorithm is unique, and will perform differently depending on the dataset in question. Some of the algorithms tested had very few options to consider when implementing them but some of the more sophisticated algorithms have more parameters that are tunable. Considering the massive parameter space available to the more sophisticated algorithms, future research could be performed into optimal methods of finding the right parameters for each algorithm combination in order to further increase the performance.

REFERENCES

[1] Verizon, *The 2013 Data Breach Investigations Report*, 2013.

[2] Checkpoint, *Checkpoint 2013 Security Report*, 2013.

[3] E. E. Schultz, *A framework for understanding and predicting insider attacks. Computers and Security*, 2002.

[4] W. H. Baker, C. D. Hylender, and J.A. Valentine, *2008 Data Breach Investigations Report*, 2008.

[5] B. Schneier, *Secrets and Lies*, John Wiley, New York, 2000.

[6] D. Huang, P. P. Rau, and G. Salvendy, *A survey of factors influencing people's perception of information security*, 2007.

[7] R. L. Daft, *Management*, Dryden Press, South-Western, 2000.

[8] SANS Institute, *NSA Glossary of Terms, NSA Glossary of Terms Used in Security and Intrusion Detection*, 2008.

[9] ISO, ISO/IEC 17799 *Information Technology, Security Techniques, Code of practice for information security management*, 2005.

[10] ISO, ISO/IEC 15408-1 *Information Technology, Security Techniques, Evaluation criteria for IT security*, 1999.

[11] Deloitte, *2007 Global financial services security survey*, 2007.

[12] J. Colley, *The information security professional is more than 'a necessary evil'*, 2007.

[13] B. Schneier, "The psychology of security", *Progress in Cryptology – AFRICACRYPT*, Vol. 5023, pp. 50-79, 2008.

[14] A. Mcllwraith, *Information security and employee behavior: how to reduce risk through employee education, training and awareness*, Gower, Hampshire, 2006.

[15] D. Gardner, *The Science of Fear*, Dutton, New York, 2008.

[16] C. Tavris and E. Aronson, *Mistakes were made (but not by me)*, Harcourt, Florida, 2007.

[17] E. Albrechtsen, *A qualitative study of users' view on information security*, 2007.

[18] M. E. Kabay, *Using social psychology to implement security policies*, 2002.

[19] K. L. Thomson, R. Solms, and L. Louw, *Cultivating an organizational information*, 2006.

[20] Fyodor, *Fyodor's Exploit World*, 2000, http://insecure.org/sploits.html, Accessed in December 2013.

[21] S. Mukkamala, A. Sung, and A. Abraham, "Intrusion detection using ensemble of soft computing and hard computing paradigms", *Journal of Network and Computer Applications*, Vol. 28, pp. 167-182, 2005.

[22] K. Kendall, *A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems*, 1999.

[23] C. Cowan, C. Pu, D. Maier, and H. Hintony, "Stackguard: automatic adaptive detection and prevention of buffer-overflow attacks", *SSYM'98 Proceedings of the 7th conference on USENIX Security Symposium*, Vol. 7, pp. 63-78, San Antonio, Texas, 1998.

[24] A. One, "Smashing The Stack For Fun And Profit", *Phrack*, No. 7, pp. 49, November 1996.

[25] D. M. Gregg, W. J. Blackert, D. C. Furnanage, and D. V. Heinbuch, *Denial of service (DOS) attack assessment analysis report Johns Hopkins University*, Technical report, Maryland, 2001.

[26] G. Vign, *Network Security Analysis, 2011*, http://www.cs.ucsb.edu/~vigna/courses/ cs279, Accessed in December 2013.

[27] CERT Coordination Center, *UDP Port Denial-of-Service Attack, 1996*, http://www.cert.org/advisories/CA-1996-01.html, Accessed in November 2013

[28] F. Lau and S. H. Rubin, "Distributed Denial of Service Attacks", *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 1, pp. 2275-2280, Nashville, Tennessee, USA, 2000.

[29] P. Charalampos, M. Masikos, and O. Zouraraki. "Distributed Denial of Service Attacks", *The Internet Protocol Journal*, Vol. 7, pp. 13-35, December 2004.

[30] C. Kruegel, F. Valeur, and G. Vigna, "Intrusion Detection and Correlation: Challenges and Solutions", *Advances in Information Security*, Vol. 14, Springer-Verlag, 2005.

[31] M. Bishop, *Introduction to Computer Security*, Addison Wesley, Boston, 2005.

[32] I. Chairunnisa, Lukas, and H. D. Widiputra, "Clustering based intrusion detection for network profiling using k-means, ecm and k-nearest neighbor algorithms" *Konferensi Nasional Sistem dan Informatika*, Bali, 2009.

[33] T. M. Mitchell, *Machine Learning*, McGraw-Hill, New York, 1997.

[34] A. Balon-Perin, *Ensemble-based methods for intrusion detection*, 2012.

[35] S. Zanero and S. M. Savaresi, "Unsupervised learning techniques for an intrusion detection system", *SAC '04: Proceedings of the 2004 ACM symposium on applied computing*, pp. 412-419, New York, USA, 2004.

[36] D. M. Farid and M. Z. Rahman, *Anomaly network intrusion detection based on improved self-adaptive Bayesian algorithm*, 2010.

[37] D. M. W. Powers, "Evaluation: From Precision, Recall and F-Factor to ROC", *Informedness, Markedness & Correlation. Technical Report*, SIE-07-001, School of Informatics and Engineering, Flinders University, Adelaide, Australia, 2007.

[38] T. Fawcett, *Roc graphs: Notes and practical considerations for researchers*, 2004.

[39] S. D. Walter, "The partial area under the summary roc curve", *Statistics in Medicine*, Vol. 24, pp. 2025-2040, 2005.

[40] American Association for Artificial Intelligence, "An empirical study of the naive Bayes classifier", *International Joint Conference on Artificial Intelligence*, pp. 41-46, 2001.

[41] J. R. Quinlan, *C4.5: programs for machine learning*, Morgan Kaufmann Publishers Inc., San Francisco, 1993.

[42] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms", *Mach. Learn*., Vol. 6, pp. 37-66, January, 1991.

[43] D. Fradkin and I. Muchnik, *Dimacs series in discrete mathematics and theoretical computer science support vector machines for classification*, 2000.

[44] A. Ali, A. Saleh, and T. Ramdan, "Multilayer perceptrons networks for an intelligent adaptive intrusion detection system" *International Journal of Computer Science and Network Security*, Vol. 10, February, 2010.

[45] M. Brameier, *On linear genetic programming*, 2005.

[46] J. R. Koza, *Genetic Programming*, MIT Press, Massachusetts, 1992.

[47] J. D. Howard, *An Analysis of Security Incidents on the Internet*, 1997.

[48] C. Benson, *Security Threats*, Microsoft, 2000.

[49] L. E. Bassham and W. T. Polk, *Threat assessment of malicious code and human threats*, 1994.

[50] J. W. Butts, R. F. Mills, and R. O. Baldwin, *Developing an Insider Threat Model Using Functional Decomposition*, 2005.

[51] R. C. Brackney and R. H. Anderson, "Understanding the Insider Threat", *Proceedings of a March 2004 Workshop*, RAND, 2004.

[52] M. D. Carroll, *Information Security: Examing and Managing the insider Threat*, 2006.

[53] J. Predd, S. L. Pfleeger, J. Hunker, C. Bulford, *Insiders Behaving Badly*, 2008.

[54] E. Kahraman, *Evaluating it security performance with quantifiable metrics*, 2005.

[55] G. Hinson, *Human factors in information security*, 2003.

[56] E. E. Schultz, *The human factor in security*, 2005.

[57] J. J. Gonzalez and A, Sawicka, *A tension between conditioning and cognition*, 2006.

[58] K. D. Mitnick, W. L. Simon, and S. Wozniak, *The Art of Deception: Controlling the Human Element of Security*, Wiley, 2002.

[59] K. Sapronov, *The human factor and information security*, 2005.

[60] D. Danchev, *Reducing "human factor" mistakes*, 2006.

[61] R. R. Panko, *Corporate Computer and Network Security*, 2004.

[62] Cisco, Understanding Remote Worker Security: *A Survey of User Awareness vs. Behavior*, 2006.

[63] S. Kraemer and P. Carayon, *A human factors vulnerability evaluation method for computer and information security*, 2003.

[64] L. L. Crumpton and P. R. McCauley-Bell, *The human factors issues in information security: What are they and do they matter?*, 1998.

[65] S. L. Pfleeger and C. P. Pfleeger, *Security in Computing*, Pearson, Boston, 2003.

[66] J. P. Anderson, "Computer Security Threat Monitoring and Surveillance", *Technical Report, James P Anderson Co*., Fort Washington, April 1980.

[67] P. G. Neumann and D. B. Parker, "A summary of computer misuse techniques", *Proceedings of the 12th National Computer Security Conference*, pp. 396-407, Baltimore, October, 1989.

[68] W. R. Cheswick and S. M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, Addison-Wesley, 1994.

[69] T. Tuglular, "A preliminary Structural Approach to Insider Computer Misuse Incidents", *EICAR 2000 Best Paper Proceedings*, pp. 105-125, 2000.

[70] G. B. Magklaras and S. M. Furnell, "Insider Threat Prediction Tool: Evaluating the probability of IT misuse", *Computers & Security*, Vol. 21, No 1, pp. 62-73, 2002.

[71] J. M. Stanton, K. R. Stam, P. Mastrangelo, and J. Jolton, *Analysis of end user security behaviors*, 2004.

[72] K. Padayachee, *Taxonomy of compliant information security behavior*, 2012.

[73] D. Denning, "An Intrusion-Detection Model", *IEEE Transactions On Software Engineering*, Vol. Se-13, No. 2, pp. 222-223, February, 1987.

[74] D. Anderson, T. Frivold, and A. Valdes, *Next-generation Intrusion Detection Expert System (NIDES)*, 1995.

[75] P. Helman, G. Liepins, and W. Richards, *Foundations of Intrusion Detection*, 1992.

[76] S. Kumar and E. Spafford, *A Pattern Matching Model for Misuse Intrusion Detection*, 1994.

[77] S. Kumar and E. Spafford, *A Software Architecture to Support Misuse Intrusion Detection*, 1995.

[78] R. Lippmann, and R. Cunningham, *Improving Intrusion Detection performance using Keyword selection and Neural Networks*, 1999.

[79] J. Ryan, M. Lin and R. Mikkulainen, *Intrusion Detection with Neural Networks*, 1998.

[80] A. Ghosh, A. Schwartzbard, and M. Shatz, *Learning Program Behavior Profiles for Intrusion Detection*, 1999.

[81] J. Cannady, *Artificial Neural Networks for Misuse Detection*, 1998.

[82] A. Morteza, R. Jalili, and R. S. Hamid, *A practical solution to real-time network-based intrusion detection using unsupervised neural networks*, 2006.

[83] Y. InSeon and U. Ulrich, *An intelligent firewall to detect novel attacks?*, 2002

[84] B. D. Kang, J. W. Lee, J. H. Kim, and O. H. Kwon, *A mutated intrusion detection system using principal component analysis and time delay neural network*, 2006.

[85] M. A. Siddiqui, *High performance data mining techniques for intrusion detection*, 2004.

[86] A. Grediaga, F. Ibarra, F. García, B. Ledesma, and F. Brotons, *Application of neural networks in network control and information security*, 2006.

[87] C. Zhang, J. Jiang, and M. Kamel, *Comparison of BPL and RBF Network in intrusion detection system*, 2004.

[88] L. Vatisekhovich, *Intrusion detection in TCP/IP networks using immune systems paradigm and neural network detectors*, 2009.

[89] KDD Cup. *Kdd cup 99 task description*, 1999, http://kdd.ics.uci.edu/databases/kddcup99/task.html, Accessed in February 2014.

[90] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update", *SIGKDD Explor. Newsl.*, Vol. 11, pp. 10-18, November 2009.

[91] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the jam project", *DARPA Information Survivability Conference and Exposition*, 2000.

[92] H. Debar, M. Dacier, and A. Wespi, *A Revised Taxonomy for Intrusion Detection Systems*, 2000.

[93] M. L. Shyu, S. C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier" *Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with the Third IEEE International Conference on Data Mining*, pp. 172-179, 2003.

[94] J. McHugh, *Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory*, 2000.

[95] M. Kearns, "A Bound on the Error of Cross Validation Using the Approximation and Estimation Rates, with Consequences for the Training-Test Split", *In Advances in Neural Information Processing Systems*, Vol. 8, pp. 183–189, The MIT Press, 1996.

[96] Weka, *Weka 3: Data Mining Software in Java*, 2013, http://www.cs.waikato.ac.nz/ml/weka, Accessed in May 2014

[97] B. Mukherjee, L. Heberlein, and K. Levitt, *Network intrusion detection*, 1994.

[98] K. Xu, Z. L. Zhang, and S. Bhattacharyya, "Profiling internet backbone traffic: behavior models and applications", *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 169-180, New York, 2005.

[99] K. C. Khor, C. Y. Ting, and S. P. Amnuaisuk, "From feature selection to building of Bayesian classifiers: A network intrusion detection perspective", *American Journal of Applied Sciences*, Vol. 6, pp. 1949-1960, 2009.

[100] K. M. Faraoun and A. Boukelif, *Neural networks learning improvement using the k-means clustering algorithm to detect network intrusions*, 2007.

[101] D. A. Zilberbrand, *Efficient Hybrid Algorithms for Plan Recognition and Detection of Suspicious and Anomalous Behavior*, 2009.

[102] H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, Burlington, 2011.

[103] Y. EL-Manzalawy and V. Honavar, *WLSVM: Integrating LibSVM into Weka Environment*, 2005.

[104] M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer, *The probability problem of pattern recognition learning and the method of potential functions*, 1964.

[105] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation", *Encyclopedia of Database Systems*. Springer. US, 2009.

[106] O. Depren, M. Topallar, E. Anarim, and M.K. Ciliz, *An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks*, 2005.

[107] J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel, "Performance measures for information extraction", *Proceedings of DARPA Broadcast News Workshop*, pp. 249-252, 1999.

[108] J. R. Quinlan, "Induction of Decision Trees", *Machine Learning*, Vol. 1, pp. 81-106. 1986.

[109] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems (2nd ed.)*, Wiley, New York, 2008.

[110] NVD, National Vulnerability Database, November 2013, URL http://nvd.nist.gov, Accessed in January 2014.

[111] CVSS, Common Vulnerability Scoring System, November 2013, URL http://www.first.org/cvss, Accessed in January 2014.

[112] CVE, Common Vulnerabilities and Exposures, November 2013, URL http://cve.mitre.org, Accessed in January 2014.

[113] A. A. Ghorbani, W. Lu, and M. Tavallaee, *Network Intrusion Detection and Prevention: Concepts and Techniques*, Springer, New York, 2010.

[114] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set,", *IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2009.