ANOMALY AND MISUSE BASED NETWORK INTRUSION DETECTION SYSTEM

A PROJECT REPORT

Submitted by

KAUMUDI GUPTA [Reg No: RA1411003010474]
SUMEDHA KHURANA [Reg No: RA1411003010478]

Under the guidance of

Mr. G. Manoj Kumar

(Assistant Professor, Department of Computer Sciene & Engineering)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM UNIVERSITY
(Under section 3 of UGC Act 1956)

S.R.M. Nagar, Kattankulathur, Kancheepuram District

MAY 2018

SRM UNIVERSITY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled "**ANOMALY AND MISUSE BASED NETWORK INTRUSION DETECTION SYSTEM**" is the bonafide work of " **KAUMUDI GUPTA [Reg No: RA1411003010474], SUMEDHA KHURANA [Reg No: RA1411003010478], , ,** ", who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Mr. G. Manoj Kumar **GUIDE** Assistant Professor Dept. of Computer Sciene & Engineering	Dr. B. Amutha **HEAD OF THE DEPARTMENT** Dept. of Computer Science and Engineering
Signature of the Internal Examiner	Signature of the External Examiner

ABSTRACT

The use of information and technology by different types of devices generates a large quantity of data packets that contains confidential and personal information. The packets are used to send data over an network. Hackers try to manipulate our data or gain illegal access to the files.Hence, it is imperative to detect such attacks using Intrusion Detection Systems. Our work puts forward an IDS that can analyze the packet-based analysis. The flows are organized to be processed by machine learning methods. A network intrusion detection system is built that alerts the admin in case of any kind of intrusion or mishandling of data. We are implementing this system using Support vector machine(SVM) algorithm, Weka and Java in various stages to overcome the challenges. Various shortcomings like Misuse of data, port scanning, denial of services will be dealt with in this system. Training data will be used to train the network and perform the testing of the system.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my guide, Mr. G Manoj Kumar. His valuable guidance, consistent encouragement, personal caring, timely help and providing me with an excellent atmosphere for doing research. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support to me for completing this research work.

Kaumudi Gupta and Sumedha Khurana

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

SVM Support vector Machine algorithm

IDS Intrusion Detection System

NIDS Network based Intrusion Detection System

HIDS Host based Intrusion Detection System

Caret Classification and Regression Training

FTP File Transfer Protocol

KNN K-nearest neighbors algorithm

OPF Optimal Power Flow algorithm

PCA Principal Component Analysis

FDA Fischer Discriminants Analysis

RBF Radial basis function kernal

CRAN Comprehensive R Archive Network

GPL GNU General Public License

CHAPTER 1

INTRODUCTION

In today's world, we all are dependent on technology. Living without technology is living without air. Internet has become an essential part of our life but with all the perks, there are some major concerns like security. There are people who would breach the security for their personal gain. They do it by using bugs or exploits to break into the computer systems of the victim and are known as Hackers. In order to detect such attacks, Intrusion detection System is required. By definition, intruder is someone who enters a place where they are not allowed to go, especially to commit a crime. Thus, intrusion detection system (IDS) is an application that keeps an eye on a network or systems for unhealthy practices or breach of code.

An Intrusion Detection System (IDS) is a software application that monitors network or system activities for malicious activities and unauthorized access to devices. IDS come in a variety of means and approach the goal of detecting suspicious traffic in different ways. There are Network based Intrusion Detection System (NIDS) and Host based Intrusion Detection System (HIDS). There are IDS that detect based on comparing traffic patterns against a baseline and looking for anomalies. There are IDS that simply monitor and alert and there are IDS that perform an action or actions in response to a detected threat. We will cover each of these briefly.The goal of designing a NIDS is to protect data's confidentiality and integrity. Our project focuses on these issues with the help of Machine Learning.

CHAPTER 2

TYPES OF INTRUSION DETECTION SYSTEM

The Intrusion Detection System can be classified into further two categories based on the how detection is done. They are Signature based and Anomaly based Intrusion Detection System.

2.0.1 Signature based Intrusion Detection System

The Signature based Intrusion Detection System depends on the knowledge we have of the previous attacks and checks for the specific patterns, such as byte sequences in network traffic, or known malicious instruction sequences used by malware. Each intrusion leaves a footprint behind (for example - nature of data packets, failed attempt to run an application, failed logins , file and folder access etc.). These footprints are known as signatures and can be used to identify and prevent the same attacks in the future. For example, you might use a signature that looks for particular strings within an exploit payload to detect attacks that are attempting to exploit a particular buffer-overflow vulnerability. The events generated by a signature-based IDS can communicate what caused the alert. Also, pattern matching can be performed very quickly on modern systems so the amount of power needed to perform these checks is minimal for a confined rule set. For instance, if the systems you are protecting only communicate via DNS, ICMP and SMTP, all other signatures can be removed.

Signature engines also have their disadvantages. Because they only detect known attacks, a signature must be created for every attack, and novel attacks cannot be detected. Signature engines are also prone to false positives since they are commonly based on regular expressions and string matching. Both of these mechanisms merely look for strings within packets transmitting over the wire.

While signatures work well against attacks with a fixed behavioral pattern, they do not work well against the multitude of attack patterns created by a human or a worm with self-modifying behavioral characteristics. Detection is further complicated by advancing exploit technology that permits malicious users to conceal their attacks behind nop generators, payload encoders and encrypted data channels. The overall ability of a signature engine to scale against these changes is hamstrung by the fact that a new signature must be created for each variation, and as the rule set grows, the engine performance inevitably slows down.

2.0.2 Anomaly based Intrusion Detection System

The Anomaly based Intrusion Detection System is a behaviour based Intrusion Detection System which takes into account a pattern of normal system activity as a baseline to identify active intrusion attempts. IDS references it with the current pattern and check if the current pattern deviates from the usual pattern or there is anything unusual about the current pattern to detect the attack. Network-based anomalous intrusion detection systems often provide a second line of defense to detect anomalous traffic at the physical and network layers after it has passed through a firewall or other security appliance on the border of a network. Host-based anomalous intrusion detection systems are one of the last layers of defense and reside on computer end points. They allow for fine-tuned, granular protection of end points at the application level.

Anomaly-based Intrusion Detection at both the network and host levels have a few shortcomings; namely a high false-positive rate and the ability to be fooled by a correctly delivered attack. A disadvantage of anomaly-detection engines is the difficultly of defining rules. Each protocol being analyzed must be defined, implemented and tested for accuracy. The rule development process is also compounded by differences in vendor implementations of the various protocols. Custom protocols traversing the network cannot be analyzed without great effort. Moreover, detailed knowledge of normal network behavior must be constructed and transferred into the engine memory for detection to occur correctly. On the other hand, once a protocol has been built and a behavior defined, the engine can scale more quickly and easily than the signature-based model because a new signature does not have to be created for every attack and potential variant.

Intrusion Detection System can also be classified into two categories based on where the detection is done. The types are Hosed based and Network based Intrusion Detection System.

2.0.3 Host based Intrusion Detection System (HIDS)

Host Based Intrusion Detection System (HIDS) runs on a specific device or system. A HIDS checks the incoming and outgoing packets from a single device and reports any suspicious activity to the user of the system. It takes snapshot of the current system and compares it with previous snapshot and alerts the user in case of any modification. In HIDS, anti-threat applications such as firewalls, antivirus software and spyware-detection programs are installed on every network computer that has two-way access to the outside environment such as the Internet.

A host-based IDS is capable of monitoring all or parts of the dynamic behavior and the state of a computer system, based on how it is configured. Besides such activities as dynamically inspecting network packets targeted at this specific host (optional component with most software solutions commercially available), a HIDS might detect which program accesses what resources and discover that, for example, a word-processor has suddenly and inexplicably started modifying the system password database. Similarly a HIDS might look at the state of a system, its stored information, whether in RAM, in the file system, log files or elsewhere; and check that the contents of these appear as expected, e.g. have not been changed by intruders.

2.0.4 Network Based Intrusion Detection System (NIDS)

Network intrusion detection systems (NIDS) analyse a network for day to day transactions and classify them as genuine or fraud based the performed analysis. It checks traffic to and from all devices attached to the network by placing the NIDS at a convenient point or points within the network. It performs an analysis of passing traffic on the entire subnet, and matches the traffic that is passed on the subnets to the library of known attacks. Once an attack is identified, or abnormal behavior is sensed, the alert can be sent to the administrator.

In NIDS, anti-threat software is installed only at specific points such as servers that interface between the outside environment and the network segment to be protected. An example of an NIDS would be installing it on the subnet where firewalls are located in order to see if someone is trying to break into the firewall. Ideally one would scan all inbound and outbound traffic, however doing so might create a bottleneck that would impair the overall speed of the network.

When we classify the design of the NIDS according to the system interactivity property, there are two types: on-line and off-line NIDS, often referred to as inline and tap mode, respectively. On-line NIDS deals with the network in real time. It analyses the Ethernet packets and applies some rules, to decide if it is an attack or not. Off-line NIDS deals with stored data and passes it through some processes to decide if it is an attack or not.

CHAPTER 3

TYPES OF ATTACKS

This section tells you about the types of attacks detected in a network-based system. These attacks can be highly dangerous and make the organization's sensitive information available to the hackers or competing organizations. We worked on detection of the following attacks in our intrusion detection system.

3.0.1 Neptune Attack

A Neptune attack is a denial of service attack. Each half-open TCP connection made to a machine causes the 'tcpd' server to add a record to the data structure that stores information describing all pending connections. Each machine which is using TCP/IP is prone to this attack. A neptune attack sends SYN packets constantly sourced from an unreachable host. This helps in blocking any incoming connections to the ports for almost an hour. It lets the user to specify a victim host, the number of packets to send,the source address , and the ports to hit on the victim machine.

3.0.2 PortSweep

In this, the attackers scans multiple hosts listening on a designated port. The attacker to get common open ports may sweep an address space and then identify those systems and do the port scanning of a system from them to see what services are being offered by the victim and how they can benefit from them. These attackers send ICMP echo requests to multiple destination address hoping that the target user would respond which will reveal the IP address of that user. The attacker sends TCP SYN packets to the target device as part of the TCP handshake. If the device responds to those packets, the attacker gets an indication that a port in the target device is open, which makes the port vulnerable to attack.

3.0.3 Nmap

Nmap is a scanner used for host discovery , port scanning and OS detection. Specially crafted packets are send by the Nmap to the hosts and their response is analyzed. The attacker can use the IP address of the host or information from port scanning to exploit the system. Nmap command can cause a UDP based DDOS attack as it lists out all the systems with open UDP services. Nmap features include:

- Host discovery - Identifying hosts on a network. For example, listing the hosts that respond to TCP and/or ICMP requests or have a particular port open.
- Port scanning- Enumerating the open ports on target hosts.
- Version detection - Interrogating network services on remote devices to determine application name and version number.
- OS detection - Determining the operating system and hardware characteristics of network devices.
- Scriptable interaction with the target - using Nmap Scripting Engine (NSE) and Lua programming language.

3.0.4 Satan

Satan attack is a ransomware which uses a tool called Satan. Satan is a scanning tool which scans the UNIX hosts on a network and reveals their system's vulnerabilities. After knowing the vulnerability of a system, the attacker exploits the system and encrypts the files and demands ransom from the victim. The decryption of the data and files is only done after the amount of money asked by the attacker is paid by the victim. The SATAN scanning tool was written by Dan Farmer and Wietse Venema. It performs scans which will leave "fingerprints" if your system is correctly instrumented.

3.0.5 Smurf

It is a kind of distributed denial-of-services attack in which ICMP packets with victim's spoofed source IP are broadcast to a computer network using an IP broadcast address. They generate a large amount of ICMP replies to the victim at the spoofed address. An ICMP host echo request is sent from host A to host B. Each host sends ICMP response to the spoofed source address.In smurf attack perpetrators take advantage of this function to amplify their attack traffic. Hence, the target server is brought down with enough responses. If the number of machines on the network that receive and respond to these packets is very large, the victim's computer will be flooded with traffic. This can slow down the victim's computer to the point where it becomes impossible to work on. In case of smurf attack, the service of the intermediate network is comparatively degraded.

3.0.6 BufferOverflow

A buffer is a temporary area for data storage. When more data (than was originally allocated to be stored) gets placed by a program or system process, the extra data overflows. It causes some of that data to leak out into other buffers, which can corrupt or overwrite whatever data they were holding.In a buffer-overflow attack, the extra data sometimes holds specific instructions for actions intended by a hacker or malicious user. For example, the data could trigger a response that damages files, changes data or unveils private information. Attacker would use a buffer-overflow exploit to take advantage of a program that is waiting on a user's input. There are two types of buffer overflows: stack-based and heap-based. Heap-based, which are difficult to execute and the least common of the two, attack an application by flooding the memory space reserved for a program. Stack-based buffer overflows, which are more common among attackers, exploit applications and programs by using what is known as a stack: memory space used to store user input.

3.0.7 FTPWrite

In this attack, the attacker tries to use the PORT command by exploiting the File Transfer Protocol (FTP) protocol. This exploit is used for port scanning of the victim's machine and for accessing the ingress and egress ports which is difficult to access through a direct connection. Nowadays, many FTP server programs are configured to refuse any request for the connection except from the host as a precaution but this doesn't prevent the attack as there could be a passive listener.

3.0.8 GuessPassword

A common denominator of most operating system and network security plans is password-based access control. This means your access rights to a computer and network resources are determined by who you are, that is, your user name and your password. Thus, guess password is a trial-and-error method used to guess passwords to hack into a system or someone's private account. An automated software is used to generate large number of passwords which are then used as input to crack the value of the desired data. To defend against this kind of attack, the admin can block a user after repeated number of attempts. It is considered to be an infallible, although time-consuming, approach.

The time to crack a password is related to bit strength (see password strength), which is a measure of the password's entropy, and the details of how the password is stored. Most methods of password cracking require the computer to produce many candidate passwords, each of which is checked. One example is brute-force cracking, in which a computer tries every possible key or password until it succeeds. More common methods of password cracking, such as dictionary attacks, pattern checking, word list substitution, etc. attempt to reduce the number of trials required and will usually be attempted before brute force. Higher password bit strength exponentially increases the number of candidate passwords that must be checked, on average, to recover the password and reduces the likelihood that the password will be found in any cracking dictionary. The ability to crack passwords using computer programs is also a function of the number of possible passwords per second which can be checked. If a hash of the target password is available to the attacker, this number can be in the billions or trillions per second, since an offline attack is possible.

3.0.9 Rootkit

Rootkit is a set of programs which helps the user to gain administration-level access. It consists off a spy-ware that monitors traffic and keystrokes and creates an alternative path for the hacker's use. Moreover, they manipulate the existing tools to escape detection. These rootkits can be used by malware and hackers to get into a system's confidential information. The only way to get rid of rootkits is to completely delete the hard drive and re-install the operating system.

Rootkit installation can be automated, or an attacker can install it after having obtained root or Administrator access. Obtaining this access is a result of direct attack on a system, i.e. exploiting a known vulnerability (such as privilege escalation) or a password (obtained by cracking or social engineering tactics like "phishing"). Once installed, it becomes possible to hide the intrusion as well as to maintain privileged access. The key is the root or administrator access. Full control over a system means that existing software can be modified, including software that might otherwise be used to detect or circumvent it. Rootkit detection is difficult because a rootkit may be able to subvert the software that is intended to find it. Detection methods include using an alternative and trusted operating system, behavioral-based methods, signature scanning, difference scanning, and memory dump analysis. Removal can be complicated or practically impossible, especially in cases where the rootkit resides in the kernel; reinstallation of the operating system may be the only available solution to the problem.[2] When dealing with firmware rootkits, removal may require hardware replacement, or specialized equipment.

CHAPTER 4

RELATED WORK

This section deals with related works performed on Intrusion Detection System. The work verifies the feasibility of machine learning algorithms in detection of malicious behaviour in network computers using network flows. Algorithms include Optimal Power Flow algorithm (OPF) ,Support vector Machine algorithm (SVM),K-nearest neighbors algorithm (KNN) and Bayes'. The work of E. M. Kakihata and Silva (2017) combines the methods of machine learning algorithms like OPF, SVM, KNN and Bayes' to check the feasibility of each algorithm in context of detection of malicious behaviour in networks computers using network flow. Network flow includes the information like source and destination IP addresses, packet any byte counts, type of service, protocol type, source and destination ports, flow start and end timestamps between two hosts on a network. They focused mainly on three attacks that are Denial of Services, Brute force and Port Scanning attacks. IDS tool detected all three types of malicious acts generated in a controlled manner, detecting brute force attacks originated by third parties from the FIPP network. The other two Malicious behaviors were not detected from the FIPP network due to firewall settings blocking some packages and while using the machine learning algorithms, OPF and SVM showed better results.

Heba F. Eid (2010) used Principal Component Analysis (PCA) with Support Vector Machines for intrusion detection system. Their approach was to select the optimum feature subset. The feasibility and the effectiveness of the proposed IDS system was verified by doing several experiments on NSL-KDD dataset by them. J.F Joseph (2011) in their paper proposed an autonomous host-based ID for detecting sinking behaviour in an ad hoc network. A cross-layer approach is used by the author to maximize detection accuracy. SVM is used for training the detection model to increase the accuracy. However, SVM is computationally expensive for resource-limited ad hoc network nodes. Hence, the proposed IDS preprocess the training data. Predefined association functions are used to reduce the number of features in the training data. The linear classification algorithm, namely Fischer Discriminants Analysis (FDA) is used in the proposed system to remove data with entropy. The above data reduction measures have made SVM feasible in ad hoc network nodes. Abhaya (2017) in their paper worked on multiple algorithms and showed that SVM with Fuzzy C- mean gives the best results. S. Latha (2017)talks about the need of an Intrusion Detection System which can detect a larger range of attacks instead of attack specific Intrusion detection system as it will more beneficial. Jayshree Jha (2013) in their paper worked with SVM based Intrusion Detection System and they showed that SVM is one of the best algorithms to classify abnormal behaviour even though it is computationally very expensive. If the number of features are reduced in the training set, then SVM would work better. Preeti Aggarwal (2015) in their paper analyzed KDD data set with respect to four classes I) Basic 2) Traffic 3) Content 4) Host. Analysis was done in weka tool on random tree algorithm. R. Vijayanand (2017) proposed system was a multi-SVM based intrusion detection system developed to detect the cyber attacks occurring in AMI communication network of smart grid.The informative features selected by mutual information technique given as input to multi-SVM classifier had a higher detection ratio than artificial neural network according to their paper. Dr.M.Amutha Prabakar (2013) has used Aho Corasick pattern matching algorithm for the detection and prevention of SQL Injection Attack. It is evaluated by using sample of attack patterns. Initial stage shows that the proposed solution is product of not false positive and false negative. We chose SVM as it seemed to be an appropriate algorithm for an intrusion detection system.

CHAPTER 5

PAST AND PRESENT SCENARIO

5.1 Present Intrusion Detection System

- The present system proposes a host based IDS
- The host operating system logs in the audit
- This audit information includes events like the use of identification and authentication mechanisms
- This audit is then analyzed to detect trails of intrusion

5.1.1 Misuse Detection

- The known patterns have to be hand coded.
- Unable to detect unknown intrusions

5.1.2 Anomaly Detection

- It relies upon in the selecting the system features.
- The sequential interrelation between transactions has to be studied.

5.2 Drawbacks of Present System

- The kind of information needed to be logged is matter of experience
- Unselective logging of messages may greatly increase the audit and analysis burdens
- Selective logging runs the risks that attack manifestations could be missed
- It protects only the host or the end points

5.3 Proposed System

- Implementing and learning how a SVM would perform when detecting an intrusion in the system.
- Testing and learning how ur system using SVM would perform in differentiating the different kinds of attacks from normal behavior.
- Testing and learning the ability of the SVM to distinguish between different types of attacks.
- We design and implement a misuse detection system capable to detect a particular attack where we train our our system using SVM on our training set with two output states: Attack of a specified type/ Other Case. After training we test our system.
- We design and implement a misuse detection system that can identify between few different attack types where we train our system using SVM on our training set with outcomes identifying each of the five attacks and then test our Neural Network on the testing set
- We design an anomaly detection system where we train our ur system using SVM to detect a normal case against any other cases and after training our system, we test our ur system using SVM on a testing set.
- We also record time taken and memory consumed in both training and testing of the system.

CHAPTER 6

MACHINE LEARNING

Machine learning, a branch of artificial intelligence, concerns the construction and study of systems that can learn from data. For example, a machine learning system could be trained on email messages to learn to distinguish between spam and non-spam messages. After learning, it can then be used to classify new email messages into spam and non-spam folders. The core of machine learning deals with representation and generalization. Representation of data instances and functions evaluated on these instances are part of all machine learning systems. Generalization is the property that the system will perform well on unseen data instances; the conditions under which this can be guaranteed are a key object of study in the subfield of computational learning theory.

There are a wide variety of machine learning tasks and successful applications. Optical character recognition, in which printed characters are recognized automatically based on previous examples, is a classic example of machine learning.

A core objective of a learner is to generalize from its experience. Generalization in this context is the ability of a learning machine to perform accurately on new, unseen examples/tasks after having experienced a learning data set. The training examples come from some generally unknown probability distribution (considered representative of the space of occurrences) and the learner has to build a general model about this space that enables it to produce sufficiently accurate predictions in new cases.

The computational analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory. Because training sets are finite and the future is uncertain, learning theory usually does not yield guarantees of the performance of algorithms. Instead, probabilistic bounds on the performance are quite common.

In addition to performance bounds, computational learning theorists study the time complexity and feasibility of learning. In computational learning theory, a computation is considered feasible if it can be done in polynomial time. There are two kinds of time complexity results. Positive results show that a certain class of functions can be learned in polynomial time. Negative results show that certain classes cannot be learned in polynomial time. There are many similarities between machine learning theory and statistical inference, although they use different terms. We used SVM algorithm in our project.

CHAPTER 7

ALGORITHMS, PACKAGES AND TOOLS

These are the packages, algorithm and tool that we used in our proposed system. All of the mentioned tool,algorithm and pack-ages helped us in training our model. These helped our proposed Intrusion Detection System to differentiate between an attack and normal traffic.

7.1 SVM

Support Vector Machines is supervised machine learning algorithm. It is an excellent algorithm if we used it for a classification or regression problem. It is non-probabilistic binary linear classifier as it assigns each training example into one category or the other. It is useful in high dimensional spaces and when the samples are lesser than dimensions. It is also memory efficient.

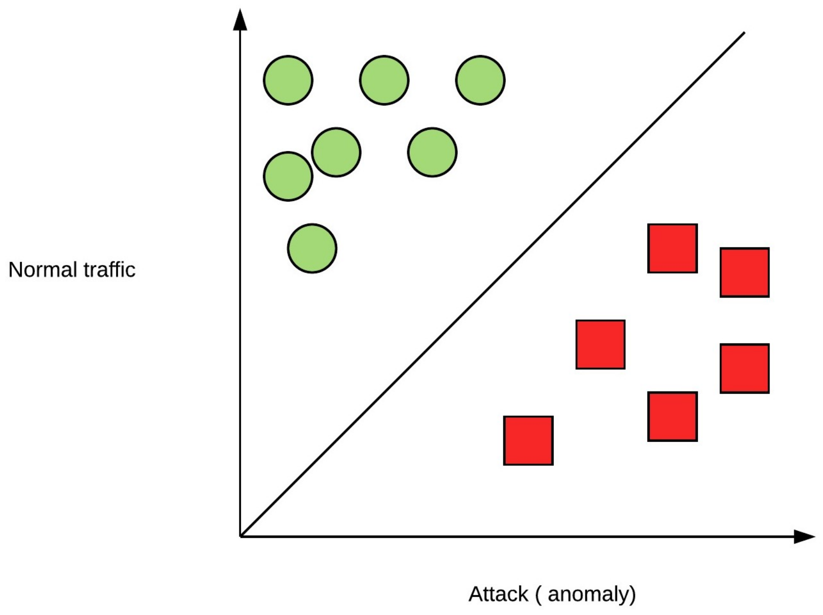


Figure 7.1: Support Vector Machine

The above is a classic example of a linear classifier, i.e., a classifier that separates a set of objects into their respective groups (GREEN and RED in this case) with a line. Most classification tasks, are often more complex structures are used in order to correctly classify new objects (test examples) on the basis of the examples that are available (training examples). Hyperplane classifiers do classification based on drawing separating lines to differentiate between different class objects. Thus, support vector machine algorithm is apt for such a task.We are using the function ksvm in R which will help us in classifying the traffic i.e., to check whether it is an attack or not. In order to achieve this, we will use the type C-svc which is used for classification problems. We chose KSVM because our dataset have non-linear decision boundary. So, in order to make dataset linearly separable in new feature space , we chose a suitable kernal function.The The most regularly utilized kernal functions are radial basis and polynomial function and we have used Radial basis function in our system.

7.2 Caret

Caret means classification and regression training. It is a package in R. It is used for the model training process for classification problems. Caret helps in running several different algorithms for a problem. It also help in knowing the optimal parameters for an algorithm. The grid search method finds the parameters by consolidating different strategies It is utilized to evaluate the execution that is the performance of the model. The grid search method searches the combination that provides the best results after looking at all the possible trial combinations. It also has tools for pre-processing and variable importance estimation. It has the train function which evaluates important features on performance using re-sampling.

7.3 Kernlab

Kernlab is a R package available from Comprehensive R Archive Network (CRAN) under the GNU General Public License (GPL) license. It provides basic kernal functionality like computing a kernel matrix using a particular kernel and modern kernel-based algorithms.Due to moduralty of kernel-based methods,the user is able to switch between kernels on an existing algorithm and also, it helps in creating and using our own kernel functions for the various kernel methods provided in the package.The package contains dot product primitives (kernels), implementations of support vector machines and the relevance vector machine, Gaussian processes, a ranking algorithm, kernel PCA, kernel CCA, and a spectral clustering algorithm. Moreover it provides a general purpose quadratic programming solver, and an incomplete Cholesky decomposition method.There are many computationally efficiently implemented kernels like Gaussian Radial basis function kernal (RBF), polynomial, linear, sigmoid, Laplace, Bessel RBF, spline, and ANOVA RBF. In our proposed system, we are using kernal âĂŹrbfdotâĂŹ.

7.4 Weka Tool

Weka is a suite of machine learning algorithms. It has a lot of algorithms which can be directly used on a dataset or can be used by writing a Java code. It is an open source tool. It is a GUI based tool used for simplification of machine learning algorithms which helps in data preprocessing, classification clustering etc. There are various algorithms to choose from depending on the results you want. The Explorer interface has components like Cluster, Associate, Classify, Preprocess , select attribute and Visualize panel. It can process the result returned by a database query and also provides access to SQL databases. In our system, we chose to use the filter RemoveUseless() which removes the attributes which are not useful for the analysis. All attributes having a missing value are replaced with a constant value. Variance of a sample of attributes are calculated to check the quality of the other attributes. Attributes that amounted to less than 1 percent of total variation were removed.

CHAPTER 8

ARCHITECTURE DIAGRAM

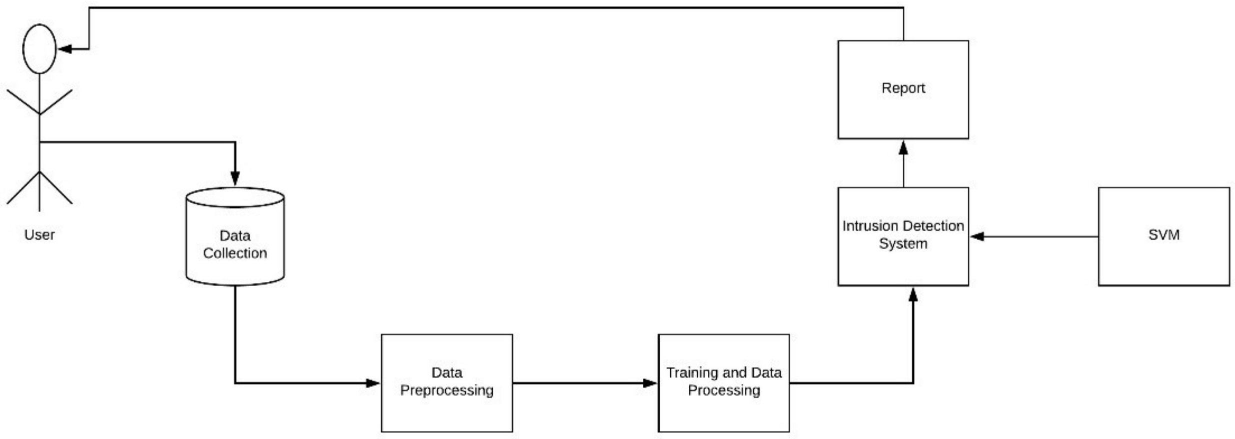


Figure 8.1: Proposed Intrusion Detection System

Here, the user represents the whole network. The data is first collected,then it is cleaned and prepared in the pre-processing step. After that, the data is processed and the system is trained by using SVM. Finally, our IDS is able to differentiate between the ordinary activity and if there is an attack.

CHAPTER 9

METHODOLOGY

9.1 Data Collection

We used KDD CUP99 dataset which is the only available dataset forIntrusion detection. This data was isolated into two sections. 75% of this data set was for training purpose whereas 25% was used as test-ing data. Two data sets are available, one for the anomaly detectionand another for misuse detection. The misuse detection data set has a variable which gives out the name of a specific type of attack suchas Neptune, Portsweep, Satan etc or else classify it as normal.

9.2 Data Preprocessing

The first and foremost approach is to simplify the data that we are using. The dataset contains nominal values also and to train a model we need all numerical values. It is also important to remove the unwanted and useless attributes in order to simplify the procedure and achieve better accuracy. This is done using the RemoveUseless() feature of the weka tool. This step reduces the size of the data and also checks for any missing attributes. The missing attributes are replaced by constant values. New set of data is generated hereafter which is used for training and testing.

9.3 Training and Processing Data

The data generated from weka is used for training using the R script. This is the most crucial step as it determines the desired results. The R script uses the caret and kern-lab helps in classification of the data i.e., in deciding whether there is an attack or not.

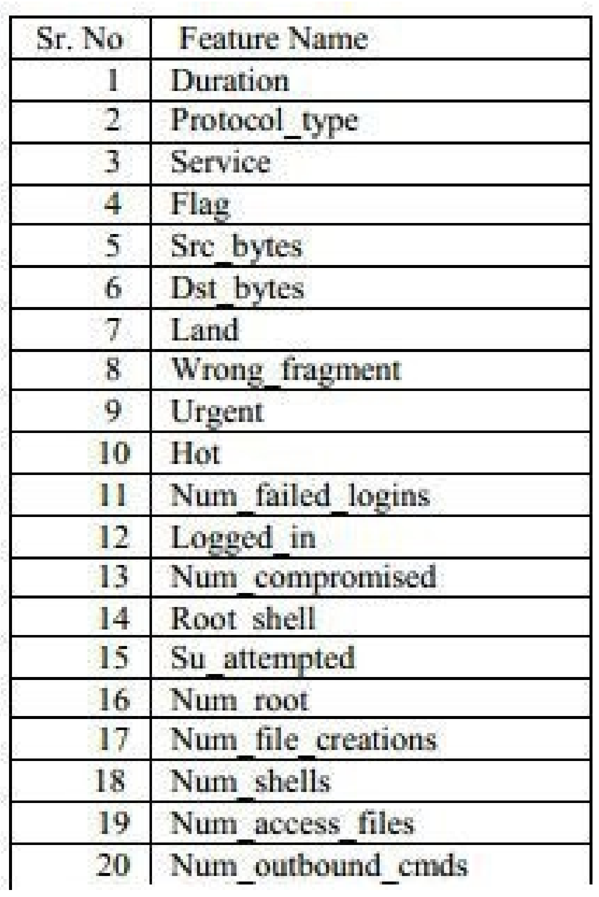
These packages are also used to train using 75%data and to test using 25% the program. Crucial points such as false positives and false negatives are identified by doing a thorough study of the confusion matrix.

CHAPTER 10

DATA FOR INTRUSION DETECTION

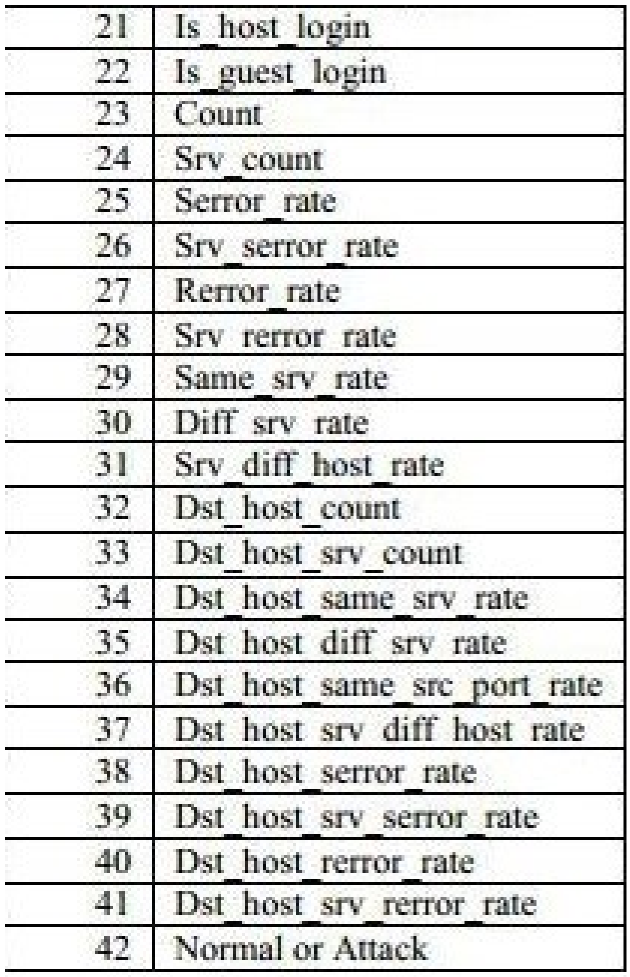
We are using NSL-KDD dataset divided into two parts, one for training and other for testing. There are total 42 features in the dataset including Protocol,_type, Service, Flag, Source bytes, Destination bytes, Num_failed_logins, Root shells etc. The last column contains the output denoting a normal activity or an attack. Different types of attacks are included like DoS, U2R, R2L, Probing.

10.1 Features of Data Set



Sr. No	Feature Name
1	Duration
2	Protocol_type
3	Service
4	Flag
5	Src_bytes
6	Dst_bytes
7	Land
8	Wrong_fragment
9	Urgent
10	Hot
11	Num_failed_logins
12	Logged_in
13	Num_compromised
14	Root_shell
15	Su_attempted
16	Num_root
17	Num_file_creations
18	Num_shells
19	Num_access_files
20	Num_outbound_cmds

Figure 10.1: Dataset



21	Is_host_login
22	Is_guest_login
23	Count
24	Srv_count
25	Serror_rate
26	Srv_serror_rate
27	Rerror_rate
28	Srv_rerror_rate
29	Same_srv_rate
30	Diff_srv_rate
31	Srv_diff_host_rate
32	Dst_host_count
33	Dst_host_srv_count
34	Dst_host_same_srv_rate
35	Dst_host_diff_srv_rate
36	Dst_host_same_src_port_rate
37	Dst_host_srv_diff_host_rate
38	Dst_host_serror_rate
39	Dst_host_srv_serror_rate
40	Dst_host_rerror_rate
41	Dst_host_srv_rerror_rate
42	Normal or Attack

Figure 10.2: More on dataset

Feature Name	Description	Type
duration	number of “hot” indicators	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 10.1: Basic features of individual TCP connections

Feature Name	**Description**	**Type**
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_guestlogin	71 if the login is a “guest”login; 0 otherwise	discrete

Table 10.2: Content features within a connection suggested by domain knowledge.

Feature Name	**Description**	**Type**
count	number of connections to the same host as the current connection in the past two seconds	continuous
	Note: The following features refer to these same-host connections.	
serror_rate	% of connections that have “SYN” errors	continuous
error_rate	% of connections that have “REJ” errors	continuous
same_srv_rate	% of connections to the same service	continuous
diff_srv_rate	% of connections to different services	continuous
srv_count	number of connections to the same service as the current conection in the past two seconds	continuous
	Note: The following features refer to these same-service	connections.
srv_serror_rate	% of connections that have “SYN” errors	continuous
srv_rerror_rate	% of connections that have “REJ” errors	continuous
srv_diff_host_rate	% of connections to different hosts	continuous

Table 10.3: Traffic features computed using a two-second time window.

CHAPTER 11

CODING, TESTING

11.1 Implementation

For classification we took into consideration 4500+ instances of normal cases and attack cases. We chose 10 types of attacks including Neptune, NMap, PortSweep, Satan, Smurf, BufferOverflow, FTPWrite, GuessPassword, Back and Rootkit attacks. Input to our "R" scripts were as follows: Dataset_Anomaly_Attribute Selection.csv to Anomaly.R : This script implements Anomaly based Intrusion Detection to provide Confusion Matrix, Classification Accuracy, Time taken for implementation and Resource Consumption. It classifies if there is Attack/Normal Case Dataset_Misuse_Attribute Selection.csv to Misuse.R: This script implements Anomaly based Intrusion Detection to provide Confusion Matrix, Classification Accuracy, Time taken for implementation and Resource Consumption. It classifies between 10 Atacks/Normal Case Dataset_Misuse_Attribute Selection.csv to Individual.R: This script implementes Misuse Detection System Capable Of Detecting A Particular Attack to provide Confusion Matrix, Classification Accuracy, Time taken for implementation and Resource Consumption for 10 individual attacks. It classifies between a Particular Attack/ Other Cases Procedure

- Open WEKA and select the Explorer Mode
- Open Dataset_Anomaly.csv and Dataset_Misuse.csv which are our datasets from Phase1
- Under Preprocess data tab select : Filter>Unsupervised>Attribute>RemoveUseless()
- We generate new datasets after saving from WEKA as Dataset_Anomaly_Attribute Selection.csv and Dataset_Misuse_Attribute Selection.csv
- Download R from http://www.r-project.org/
- Open R and select Open Script to select new datasets
- Select all the code and press "Run" to execute script commands

- The script uses kernlab and caret package which will train using 75% Training Set
- Tests are carried out on a 25% Test set to generate results such as summary, confusion matrix and accuracy and we can determine the number of false positives and false negatives

This is the code we used for the preparation of the dataset. It is written in Java.

```
import java.io.*;
import java.util.ArrayList;

// This class represents Data Preparation
public class DataPreparaton {
  // List that stores all the data from files
  static ArrayList<String> list = new ArrayList<String>();

  public static void main(String args[]) throws
     FileNotFoundException,
       IOException {
    String FILE_LOCATION = "C:\\Users\\Kaumudi
       Gupta\\Desktop\\final year
       project\\Network-Intrusion-Detection-System-master\\data";
    // Number of maximum instances of each attack
    int noOfAttack = 250;
    int noOfNormal = 3000;
    // Filepath for built datasets
    String anomaly = FILE_LOCATION+"/Dataset_Anomaly.csv";
    String misuse = FILE_LOCATION+"/Dataset_Misuse.csv";
    // Filepath for Input files
    String attackPath = FILE_LOCATION+"/Optimized_";
    // An array that stores the types of attacks
    String[] attacks = { "Neptune", "NMap", "PortSweep",
       "Satan", "Smurf",
         "BufferOverflow", "FTPWrite", "GuessPassword", "Back",
         "Rootkit" };
    for (int i = 0; i < attacks.length; i++) {
      generateFile(attackPath + attacks[i] + ".csv", misuse,
          ", "
           + attacks[i] + "\n", noOfAttack);
    }
    generateFile(attackPath + "Normal.csv", misuse, ",
       Normal\n",
         noOfNormal);
    printShuffle(list.toArray(new String[0]), misuse);

    list = new ArrayList<String>();
    for (int i = 0; i < attacks.length; i++) {
      generateFile(attackPath + attacks[i] + ".csv", anomaly,
           ", Attack\n", noOfAttack);
    }
    generateFile(attackPath + "Normal.csv", anomaly, ",
       Normal\n",
         noOfNormal);
    printShuffle(list.toArray(new String[0]), anomaly);
  }

  // Reads inputs and creates a list that represents data in
      the file
  public static void generateFile(String input, String output,
     String column,
      int size) throws FileNotFoundException, IOException {
    BufferedReader br = new BufferedReader(new
       FileReader(input));
    String line;
    int i = 0;
    while ((line = br.readLine()) != null && i < size) {
      list.add(line + column);
      i++;
    }
    br.close();
  }

  // swaps array elements i and j
  public static void swap(String[] a, int i, int j) {
    String swap = a[i];
    a[i] = a[j];
    a[j] = swap;
  }

  // takes as input an array of strings and rearranges them in
     random order
  public static void printShuffle(String[] a, String output)
      throws FileNotFoundException,
         UnsupportedEncodingException,
      IOException {
    int N = a.length;
    for (int i = 0; i < N; i++) {
      int r = i + (int) (Math.random() * (N - i)); // between
         i and N-1
      swap(a, i, r);
    }
    File file = new File(output);
    Writer writer = null;
    if (!file.exists()) {
      writer = new BufferedWriter(new OutputStreamWriter(
          new FileOutputStream(output), "utf-8"));
    } else
      writer = new PrintWriter(new BufferedWriter(new
         FileWriter(output,
          true)));
    // Attributes list
    writer.write("duration, protocol_type, service, flag,
       src_bytes, dst_bytes, land, wrong_fragment, urgent,
          hot, num_failed_logins, logged_in, num_compromised,
          root_shell, su_attempted, num_root,
          num_file_creations, num_shells, num_access_files,
          num_outbound_cmds, is_host_login, is_guest_login,
          count, srv_count, serror_rate, srv_error_rate,
          rerror_rate, srv_rerror_rate, same_srv_rate,
          diff_srv_rate, srv_diff_host_rate, dst_host_count,
          dst_host_srv_count, dst_host_same_srv_rate,
          dst_host_diff_srv_rate, dst_host_same_src_port_rate,
          dst_host_srv_diff_host_rate, dst_host_serror_rate,
          dst_host_srv_serror_rate, dst_host_rerror_rate,
          dst_host_srv_rerror_rate, AttackType\n");
      for (int i = 0; i < N; i++) {
      writer.write(a[i]);
      }
      writer.close();
    }
  }
```

This code is used for training the system. It is written in R language. This is code for Anomaly based NIDS.

```
library(kernlab)
library(caret)
anomaly<-read.csv("C:\Users\Kaumudi Gupta\Desktop\final year
   project\Network-Intrusion-Detection-System-masterdata/Dataset_Anomaly.cs
   na.strings=c(".", "NA", "", "?"), strip.white=TRUE,
   encoding="UTF-8")
aRow<-nrow(anomaly)
aCol<-ncol(anomaly)

sub<-sample(1:aRow,floor(0.66*aRow))
anomalyTrainingSet<- anomaly[sub,]
anomalyTestSet<- anomaly[-sub,]
anomalyClassifier<-
   ksvm(AttackType~.,data=anomalyTrainingSet,type = 'C-svc',
   kernel = 'rbfdot')
anomalyPrediction<-predict(anomalyClassifier,
   anomalyTestSet[,-aCol])
confusionMatrix(anomalyPrediction,anomalyTestSet[,aCol] )
```

This is also written in R language. This is code for Misuse based NIDS.

```
         library(kernlab)
library(caret)
misuse<-read.csv("C:\Users\Kaumudi Gupta\Desktop\final year
   project\Network-Intrusion-Detection-System-master/data/Dataset_Misuse.cs
   na.strings=c(".", "NA", "", "?"), strip.white=TRUE,
   encoding="UTF-8")
mRow<-nrow(misuse)
mCol<-ncol(misuse)

sub<-sample(1:mRow,floor(0.66*mRow))
misuseTrainingSet<- misuse[sub,]
misuseTestSet<- misuse[-sub,]
misuseClassifier<-
   ksvm(AttackType~.,data=misuseTrainingSet,type = 'C-svc',
   kernel = 'rbfdot')
misusePrediction<-predict(misuseClassifier,
   misuseTestSet[,-mCol])
confusionMatrix(misusePrediction,misuseTestSet[,mCol] )
```

CHAPTER 12

RESULT

12.1 Result

The data set consisted of 41 attributes originally which were later narrowed down to 36 for simplification. This was done using the feature of weka tool named, RemoveUseless(). As a result, we have divided the attacks into anomaly ad misuse types and accordingly the results are obtained. We have 9 attack types, the output of each takes a account of several factors such as sensitivity, specificity, detection rate, detection prevalence and balanced accuracy. The following tables list out the results obtained after each step.

PREDICTION	Bufferoverflow	FTPWrite	GuessPassword	Neptune	Nmap	PortSweep	Rootkit	Smurf	Satan
BufferOverflow	85	0	0	0	0	0	0	0	0
FTPWrite	0	7	1	0	0	0	0	0	0
GuessPassword	0	0	18	0	0	0	0	0	0
Neptune	0	0	0	100	0	0	0	0	0
Nmap	0	0	0	0	88	0	0	0	0
PortSweep	0	0	0	0	0	72	0	0	0
Rootkit	0	0	0	0	0	0	0	0	0
Smurf	0	0	0	0	0	0	0	0	73
Satan	0	0	0	0	0	0	3	1	85

Figure 12.1: Confusion Matrix

PREDICTION	Bufferoverflow	FTPWrite	GuessPassword	Neptune	Nmap	PortSweep	Rootkit	Smurf	Satan
Sensitivity	0.4545	0.0000	0.9200	1.0000	0.9662	1.0000	0.0000	0.9594	0.9875
Specificity	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9993	1.0000	1.0000
Detection Rate	0.003195	0.000000	0.01470	0.06006	0.05495	0.05304	0.000000	0.05048	0.04537
Detection Prevalence	0.003195	0.000000	0.01470	0.06006	0.05495	0.05367	0.000000	0.05367	0.04537
Balanced Accuracy	0.727273	0.500000	0.96000	1.00000	0.98315	0.99966	0.500000	0.99207	0.97973

Figure 12.2: Statistics by Class

Prediction	Attack	Normal
Attack	546	4
Normal	0	1015

Accuracy: 0.9974

No Information Rate:0.6511

Kappa:0.9944

Mcnemar’s Test P-value:0.1336

Sensitivity : 1.0000

Specificity : 0.9961

Pos Pred Value : 0.9927

Neg Pred Value : 1.0000

Prevalence : 0.3489

Detection Rate : 0.3489

Detection Prevalence : 0.3514

Balanced Accuracy : 0.9980

’Positive’ Class : Attack

Figure 12.3: Confusion Matrix and Statistics

Classification Approach	Accuracy for Anomaly based detection	Accuracy for Misuse based detection
Using SVM	98.79	98.59

Figure 12.4: Overall Statistics

CHAPTER 13

CONCLUSION AND FUTURE ENHANCEMENT

The first part involves categorization of different threats into 2 main categories namely anomaly based and misuse based. Various algorithms were tried and tested for leading to the result that SVM works successfully in case of both the types of attacks. The machine is trained using different data sets for anomaly and misuse based detection. After looking at the above results we can say that SVM is an effective algorithm to detect anomalies or malicious behaviour in the network as SVM detects any change in pattern. The algorithm shows 98.79% accuracy in anomaly based detection and 98.59% in misuse based detection, which is commendable result. As future work it is suggested to carry out the detection of other types of threats, such as arbitrary code execution in Internet pages (cross-site scripting) and poisoning the DNS memory (obfuscation of DNS). We also suggest to apply multiple algorithms with SVM to improve the accuracy. Network Intrusion Detection can be improved using latest machine learning techniques like deep neural network, dbscan etc. Also, the dataset is very old and there are some bugs which can be tackled using a well derived dataset. Dimensionality Reduction using principal component analysis can also be used to improve the time and visualization.

REFERENCES

1. Abhaya, K. K. (2017). “An efficient network intrusion detection system based on fuzzy c-means and support vector machine.” *IEEE.*

2. Dr.M.Amutha Prabakar, M. (2013). “An efficient technique for preventing sql injection attack using pattern matching algorithm.” *IEEE International Conference on Emerging Trends in Computing, Communication and Nanotechnology (ICECCN).*

3. E. M. Kakihata, H. M. Sapia, R. T. O. D. R. P. J. P. P. V. H. C. A. and Silva, F. A. (September 2017). “Intrusion detection system based on flows using machine learning algorithms.” *IEEE TRANSACTIONS*, 15(10).

4. Heba F. Eid, Ashraf Darwish, A. E. H. a. A. (2010). “Principle components analysis and support vector machine based intrusion detection system.” *IEEE*, 8.

5. Jayshree Jha, L. R. (2013). “Intrusion detection system using support vector machine.” *International Journal of Applied Information Systems.*

6. J.F Joseph, A. Das, B. S. (2011). “Cross-layer detection of sinking behavior in wireless ad hoc networks using svm and fda.” *IEEE Transaction on dependable and securecomputing*, 8(2).

7. Preeti Aggarwal, S. K. S. (2015). “Analysis of kdd dataset attributes : Class wise for intrusion detection.” *ICRTC.*

8. R. Vijayanand, D. Devaraj, B. K. (2017). “Support vector machine based intrusion detection system with reduced input features for advanced metering infrastructure of smart grid.” *International Conference on Advanced Computing and Communication Systems (ICACCS).*

9. S. Latha, S. J. P. (2017). “A survey on network attacks and intrusion detection systems.” *International Conference on Advanced Computing and Communication Systems.*