PROJECT REPORT

On

HYBRID INTRUSION DETECTION SYSTEM

Submitted in partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENIGNEERING

By

SAHIL SHARMA	**1031210138**
GAUTAMI EDARA	**1031210152**

Under the guidance of

Mrs. P. SARANYA
(Assistant Professor, Department of Computer Science and Engineering)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING AND TECHNOLOGY
SRM UNIVERSITY
(Under section 3 of UGC Act, 1956)
SRM Nagar, Kattankulathur- 603203
Kancheepuram District

APRIL 2016

PROJECT REPORT

On

HYBRID INTRUSION DETECTION SYSTEM

Submitted in partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENIGNEERING

By

SAHIL SHARMA	**1031210138**
GAUTAMI EDARA	**1031210152**

Under the guidance of

Mrs. P. SARANYA

(Assistant Professor, Department of Computer Science and Engineering)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING AND TECHNOLOGY
SRM UNIVERSITY
(Under section 3 of UGC Act, 1956)
SRM Nagar, Kattankulathur- 603203
Kancheepuram District

APRIL 2016

BONAFIDE CERTIFICATE

Certified that this project report titled **"HYBRID INTRUSION DETECTION SYSTEM"** is the bonafide work of SAHIL SHARMA (Reg. No: 1031210138) and GAUTAMI EDARA (Reg. NO: 1031210152), who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion or any other candidate.

Signature of the Guide

Mrs. P. SARANYA
Assistant Professor
Department of Computer Science and Engineering
SRM University
Kattankulathur- 603203

Signature of the HOD

Dr. B. AMUTHA
Professor& Head
Department of Computer Science and Engineering
SRM University
Kattankulathur- 603203

Signature of Internal Examiner

Signature of External Examiner

DATE

ABSTRACT

This project proposes a Hybrid Intrusion Detection System for network which contains a two layer protection scheme. The first layer consists of rule based Intrusion Detection System which uses a database of signatures to validate the network traffic and the second layer consists of a system which uses a supervised machine learning approach to check for any new malicious signature which has never attacked before, based on supervised learning model. The new malicious signature detected using the second layer of the Intrusion Detection System is then placed in database of the first layer; hence the database gets updated dynamically. This Intrusion Detection System is used in a wireless sensor environment. The wireless sensor nodes present in this environment are grouped into clusters and each cluster is assigned with a cluster-head. Further, the optimal position for installing the Intrusion Detection System in this environment is discussed. Several performance metrics such as cost, detection rate, scalability, etc. are analyzed.

ACKNOWLEDGEMENT

I am duly bound to place on record my gratitude to Dr. B. AMUTHA, Professor and Head of Computer Science and Engineering, SRM University for her valuable support throughout to bring out this project in time. I would also like to thank all the panel members for their constant support I extend my gratitude and heartfelt thanks to Asst. Professor P. SARANYA, Project Guide, Department of Computer Science and Engineering for having provided me opportunity to work under her guidance and motivation for shaping the project work. Her constant encouragement, support and vision enabled me to take this new endeavor to the success path. I am very grateful to my guide, who has guided me with inspiring dedication, untiring efforts and tremendous enthusiasm in making this project successful and presentable. At last, I am thankful to SRM University for encouraging me to carry out my project work. This has widened my awareness on the subject. Due to the study conducted in the process of working with this project, I came across many facts, which I never knew existed. My knowledge of area under discussion has developed owning to this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
IDS	Intrusion Detection System.
NIDS	Network Intrusion Detection System.
HIDS	Host Based Intrusion Detection System.
IPS	Intrusion Prevention System
IDPS	Intrusion Detection and Prevention System
SVM	Support Vector Machine
KDD	Knowledge Discovery Database
UDP	User Datagram Protocol
ICMP	Internet Control Message Protocol
U2R	User to Root
R2L	Remote to User.
POP3	Post Office Protocol version 3
IP	Internet Protocol.
DNS	Domain Name Server.
TCP	*Transmission Control Protocol.*
DoS	Denial of Service.
URL	Uniform Resource Locator.
FTP	File Transfer Protocol

LIST OF SYMBOLS

SYMBOL	DESCRIPTION
$J(\Theta)$	Cost function related to the model used Output of the given input Sigmoid function
O	Big-oh Representation
G_x	Output Function

CHAPTER - 1

INTRODUCTION

With the advent of mobile technology, the use of technological devices has increased exponentially and this has led to more network traffic being generated than in the past few decades. A large amount of user data is generated based on this traffic which has to be protected. Firewalls are used to filter the network traffic based on host policies and they provide the first line of defence against any attack on the network. Intrusions in the network are attempts to destabilise it by compromising with the availability, confidentiality or integrity of the system. Intrusions in the network cannot be stopped only by a firewall and hence Intrusion Detection System is required. Intrusion Detection System is divided into two types Network Based Intrusion Detection System (NIDS) and Host Based Intrusion Detection System (HIDS).

There are different types of attacks for which we need IDS.

1. Denial of service attack – denial-of-service attacks are one of the easiest attacks. They can easily target any system and get useful information from them or make the system fail. This can lead to malfunction of services provided by the system. Systems don't have any embedded filters to filter out the attacks. Common denial of service attacks is *flooding*.
2. Confidentiality Threat – Few attackers make programmes which attach themselves to the existing files, they corrupt them or make a copy which can be leaked or distributed via email. This leads to confidentiality threat for the author as this action is being taken without his permission.
3. Modify the contents – Intruders can change the content or they can even make hoax broadcasts from authors system all this can have an economic impact.
4. Masquerade – Masquerade is nothing but masking where one element pretends to another element. Like anyone can access the system if username identification and password is known, but the original person is not authorising. So the only author has the privilege to authorise anyone else so that they have access to the data

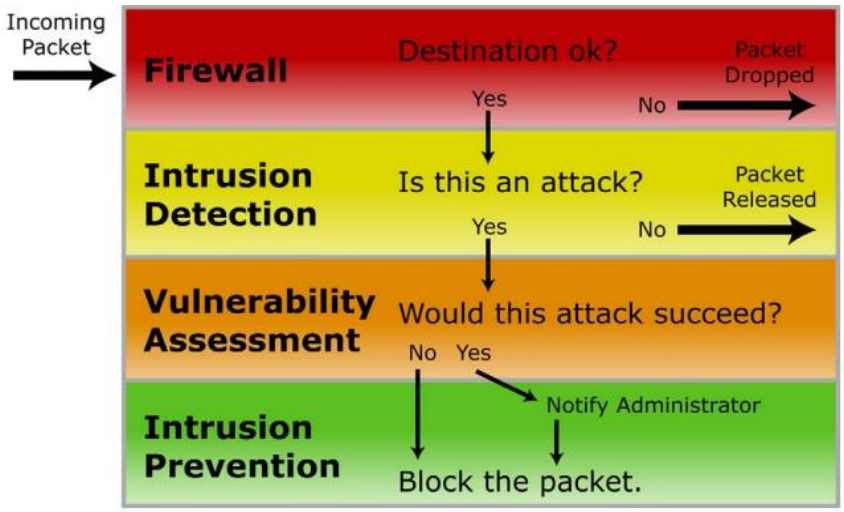


Figure 1.1: Steps of Intrusion Detection.

1.1 NETWORK INTRUSION DETECTION SYSTEM

Network Intrusion Detection System are placed at specific strategic points within the network. It inspects all the devices present in the network; they even monitor user activities by looking into the data portions of packets for malicious command sequences. In NIDS, anti-threat software is only installed at specific points such as servers that interface between the outside environment and the network segment to be protected [6]. NIDS can scan system files for compromised sectors and to check file and system integrity. It also goes through log files to look for some specific signatures which can attempt an attack on the system. Since data portions and header information can be encrypted these changes render NIDS useless. Examples of NIDS implementation are Snort ISS, Cisco Secure IDS and Dragon Enterasys.

Pros:

1. It can run on the cross-platform environment.
2. NIDS is centralised.

Cons:

1. Rigorous training is required.
2. More LAN bandwidth should be present for usage.

1.2 HOST BASED INTRUSION DETECTION SYSTEM

HIDS are installed on individual hosts or devices in the network. They monitor all traffic going in and out of the device and signal users or administrators of the device when some suspicious traffic is found. Firewall, antivirus software and spyware-detection are detection n programmes are installed in every network in the environment. Some of the examples are analysing shell commands analysing any user to root attack (U2R) or remote to user attack (R2L). HIDS provides protection to the local database, log files and its reports from any tampering from any intruder since any modification to these files can lead to changes to HIDS itself.

Pros:

1. It protects off the LAN.
2. It is versatile in nature.
3. HIDS provides local machine registry scan.
4. Requires less training in comparison to NIDS.
5. Less bandwidth is required than NIDS.

Cons:

1. It cannot identify the attack and it cannot prevent it immediately.
2. Data collection is done per-host basis.
3. Log entry and reporting generates extra load for a network.
4. Hackers can attack and they can disable HIDS.
5. While attacking HIDS consumes processing time, storage, memory and other system resources.

1.3 FUNCTIONALITY

Intrusion Detection System (IDS) –

Intrusion detection is the process of identifying malicious activities. They target computer systems and network resources. Types of Intrusion detection systems are 1) HIDS 2) NIDS. IDS detects if there is any intrusion and it alarms about it to the administrator. There are two types intrusion detection techniques 1) Anomaly detection 2) Rule Based detection. Anomaly detection analyses the information gathered from

different resources and compared it to the standard format which is seen as normal service behaviour. Whereas Rule-based detection is based on the signature for known attacks. It only detects the attacks. Algorithms like Adaptive Resonance Theory, Self-organizing m p, and genetic algorithm are used by IDS.

Intrusion Prevention System (IPS)–

Initially, IDS was only capable of detection of intrusion without any prevention action. The proactive technique of IDS prevents the attack from entering the network. It examines the packets and their patterns and then blocks them. It is proactive and smart; this system provides early detection of attacks. IPS works on Data Link Layer, Network Layer, Transport Layer and Application Layer of OSI. When the attack is identified, then it blocks the offending data.

Intrusion Detection and Prevention System (IDPS) –

When there are open ports in the network Firewall does not provide any security against this. Hence, Intrusion Detection Prevention System is used to protect the network services with the firewall. IDPS has three parts 1) preprocessing of packets 2) classification 3) protection. In preprocessing the packet sniffer is used to capture the information from the incoming packets. Then this data is further classified into two types a) attack packet and b) normal packet. Later this information is passed to protection part where appropriate actions are taken according to the type of packet for prevention of attack. Examples are Snort, Cisco IPS Sensor Software.

Intrusion Detection System device monitors all the incoming network traffic. It checks whether an external party is sending any attacks or malicious data on the network. It is also applicable on IP headers, application content, etc., unlike firewall.

An Intrusion Prevention System is a module which is added to a traditional Intrusion Detection System. This module helps in performing specific functions by themselves. An administrator of the system can describe the actions which should be taken by the Intrusion Prevention System. This allows the administrator to specify whether the attack was an event at the denial of service (DoS) level or greater. Certain filter duration is set.

The advantages of Intrusion Prevention Systems –

1. IP should not be blocked permanently for which filter duration is specified.
2. The ability of an attacker to attack the target network can be automatically blocked any time of the day.
3. Email notification can be sent to the administrator of the system.
4. Notification should be notified immediately when the attacker comes into the network|.

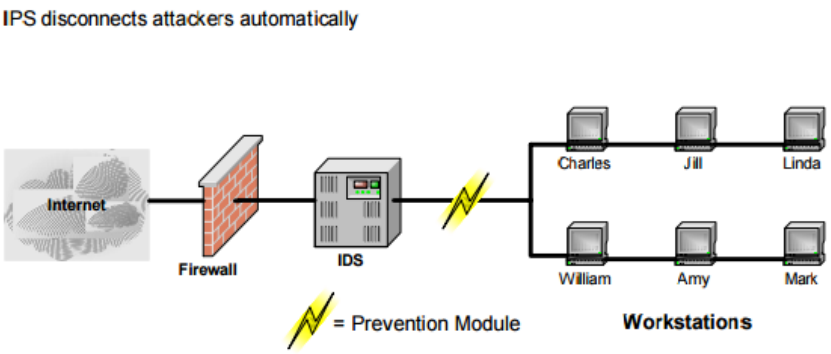


Figure 1.2: Intrusion prevention system

1.4 DEPLOYMENT APPROACH

Single Host – In single-host deployment, the system is installed on a single host. In this, a network may be a server, a router or network switch. The traffic enters from and leaves the network via that node; here it is checked for attacks and non-malicious packets by the NIDS. Examples: GrIDS, NetRanger and Bro.

Pros:

1. It can monitor a wide subnet.
2. The impact on the system is very little; it is a passive device.

Cons:

1. Difficult to process all the packets in a busy network.
2. Multiple Hosts: The distributed deployment of NIDS, here the system is installed on all the nodes it can be called as NIDS agents. These agents then

monitor the traffic which is routed through that node and it generates appropriate results. Results are then sent to central NIDS controller where the management system is coordinated with the agents and generates alarms for appropriate packets and broadcasts it over the network. Examples: AAFID and Micael.

Pros:

1. It over comes the problem of processing all packets by single NIDS.

Cons:

1. Harder to manage and must be configured for each different host.
2. Coordination between NIDS is difficult.

1.5 RULE BASED INTRUSION DETECTION SYSTEM

Rule Based IDS detects malicious signatures using a database set consisting of all the malicious signatures that have attacked previously. All the participants in the network have a signature associated with it. A signature can be an IP header, Packet with an illegal TCP flag combination, anonymous FTP attack, DNS buffer overflow attempt, Subject of an Email, which contains a virus or a Denial of Service attack on a POP3 server. This is a simple system which works efficiently to suppress known attacks but it is not able to detect any new attack due to unavailability of signature in the database. It has low false positives, false negatives rate. Examples: Suricata.

Pros:

1. Low false positive rate if attack signatures are clearly defined.
2. Easy to use.

Cons:

1. Requires specific knowledge of intrusion behaviour and collect and update the data before the intrusion could be out of date.
2. Unknown attacks are difficult to be identified.
3. It alarms regardless of the outcome. Example: the IDS send many alerts if a Windows worm tries to attack a Linux system.
4. The knowledge of the attacks is dependent on the environment of the system.

1.6 ANOMALY BASED INTRUSION DETECTION SYSTEM

An Anomaly Based IDS detects anomalies in the network traffic. It detects the anomaly based on Pattern Matching, Resource utilisation, etc. It also detects well-known as well as new attacks. Depending on the which machine learning techniques used it can be classified such as Supervised Learning, Fuzzy Logic, Support Vector Machine, Bayesian Network, Genetic Algorithm or Neural networks. The problem that may occur with these systems is that they have a high false positive rate and false negative rate. Signature based IDS depends on human supervision to create test and deploy the signatures, but it is not possible for humans themselves to add all the signatures in real time since it can take hours or even days to create a new signature. Hence, an independent human solution is required. Machine learning techniques can be used to implement such a system which can learn from the data it encountered or the present set of data and it can use this to avoid any of the new attacks. Example: Snort, Bro-IDS are anomaly based intrusion detection system.

Pros:

1. It can detect unknown attacks.

Cons:

1. Defining the rule set is difficult.
2. The efficiency of a system depends on the fitness of the rules or test sets.

1.7 HYBRID INTRUSION DETECTION SYSTEM

There is also a third type which combines rule-based IDS and anomaly based IDS this system is called as ***Hybrid Intrusion Detection System***. This system works as a two-way pass in which the traffic is passed through rule-based IDS, which checks for known malicious signatures and then through anomaly based IDS which detects new acts by using heuristic methods. This system is more efficient in finding known as well as new attacks. The false positive and false negative rate are lower as compared to anomaly based IDS.

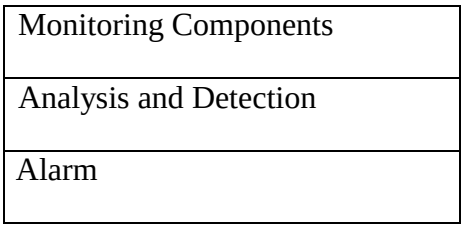


Figure 1.3: The components of IDS

The monitoring component monitors local events as well as neighbour events. It monitors traffic patterns, internal events and resource utilisation. Analysis and Detection are done by the modelling algorithm which analyses network operation, behaviour and activities. Then it makes decisions to declare whether the given signature is malicious or not. The alarm is a response generation component. It alerts the user about an intrusion when it has been successfully detected.

CHAPTER – 2

LITERATURE REVIEW

There are some techniques available today for protecting and securing your network. Intrusion Detection System provides such a technique. It is used to analyse the network for any malicious activity and then ping the user regarding it such that necessary actions can be taken. There is noticeable work done on this field and vast literature is available. The topic is still current since new techniques are in the process of discovery and there are new issues that crop now and then. Akyildiz[2], C. Perera[4] and Rodrigo Roman[15] provided surveys regarding different techniques to perform Intrusion Detection and to build IDS using the same. J.Antony Jeyanna, E.Indumathi, Dr.D.Shalini Punithavathani [7] proposed IDS using clustering and outlier detection. N. Wattanapongsakorn [12] provided a practical approach for Intrusion Detection in the IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications. Signature based implementations using snort are greatly used. Vinod Kumar [21] has provided a Signature based IDS using snort. Hybrid IDS are the latest type of IDS that has come to provide the functionality of both Rule-based IDS and Anomaly based IDS. Manish Somani [11], Kai Hwang [9] and Kayvan Atefi1 [10] have provided relevant work in this direction. Pan, S., T. Morris, And U. Adhikari[13] have provided a Hybrid IDS, which uses data mining for power systems. Many machine learning techniques have been used for Anomaly Based Intrusion Detection out of which Neural Network implementations, Clustering based implementations and regression based implementations are noticeable. Sanjay Kumar Sharma, Pankaj Pande, Susheel Kumar Tiwari and Mahendra Singh Sisodia [19] and Taskin Kocak[20] have discussed Hybrid IDS with Neural network implementations.

CHAPTER – 3

PROJECT DESCRIPTION

3.1 PROPOSED MODEL

The proposed model is a Hybrid IDS, which has two layers protection scheme. The first layer is rule-based detection and the second layer contains a supervised learning IDS. The following figure is an architecture model of the proposed Hybrid IDS.

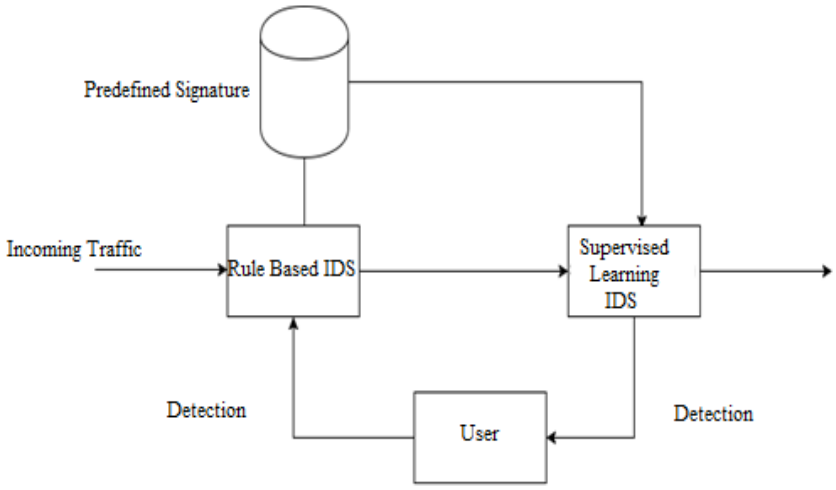


Figure 3.1: Architecture of Hybrid IDS

The Rule Based IDS will have a local repository of malicious signatures which can be a block list of suspected malicious IPs and URLs such as ATLAS from Arbor Networks, DGA list, DShield Block List. There can be other types of signatures such as junk, spam; phishing, etc. in databases such as junk.DB, foxhole_all.Cdb, push.Ndb, lot.ndb, etc. All the packets coming from the traffic are broken into IP headers and content. The IP headers can be thought of as signatures and can be looked for matching malicious IPs in the database. Then the content can be analysed to provide additional signatures such as a subject regarding spam mails and other types of signatures already discussed. After coming across a malicious signature, the user can be notified about the intruder and subsequent actions can be taken. The valid traffic is sent to the next layer.

The second layer contains the IDS using a supervised learning algorithm to classify the traffic into malicious or valid. The supervised learning model used will be a logistic regression model. The logistic regression is defined by a logistic regression cost function **J(Θ)** which can be depicted as

$$-\frac{1}{m}\sum_{i=1}^{m} v^i \log(\mathrm{k}_\theta(x^i)) + (1 - v^i) \log(1 - \mathrm{k}_\theta(x^i))$$

We need to minimise the cost function so as to get optimal Θ. This can be done by using gradient descent. In gradient descent, we perform the following operation

Repeat {

$\Theta_{j:=} \Theta_j - \alpha \frac{d}{d\theta j} J(\Theta)$

// simultaneously update all Θ_j.

}

Gradient descent is performed instead of analytical methods because it is much faster. Analytical methods carry a time complexity of $O(n^3)$ which is impractical for implementing in practical applications.

In logistic regression model, the output is predicted depending upon whether the input passes a given threshold. If the input is above the threshold, it is detected as an anomaly and vice versa. The output is defined by a sigmoid function which is defined as

$$= \frac{1}{1+e^{-z}}$$

Where $z = .\Theta^T x$

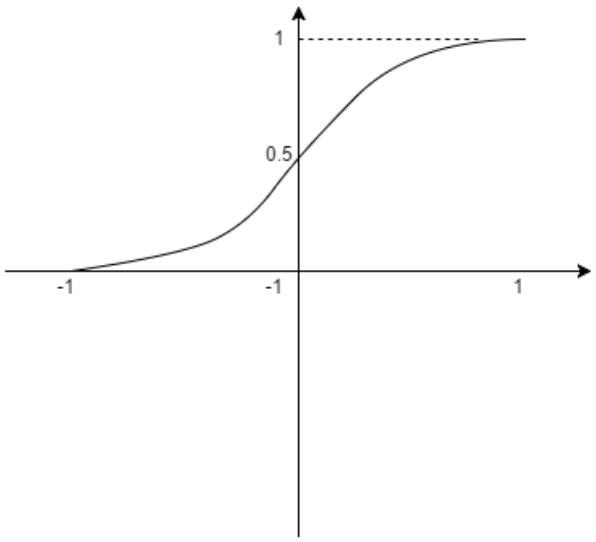


Figure 3.2: Graph Depicting Sigmoid Function

A threshold value is decided to depend on different constraints. Suppose the threshold is 0.5. So, if

$k_\Theta(x) \geq 0.5$; $y = 1$

$k_\Theta(x) < 0.5$; $y = 0$

Hence like this we can predict the output depending on the threshold. If we take the value of threshold around 0.3, then it will give less number of false positives but it may not be able to detect some of the attacks. If we set the threshold value around 0.7, then the false positive rate will be high but it will be able to cover most of the attacks.

3.2 HYBRID INTRUSION DETECTION ALGORITHM

As stated earlier, the algorithm works by capturing the packets and then performing a hybrid algorithm to check for intrusions. Hence, the algorithm can be divided into the following phases:-

3.2.1 Packet Capturing;

3.2.2 Signature-Based Intrusion Detection &

3.2.3 Anomaly Based Intrusion Detection

3.2.1 Packet Capturing

The first phase to detect intrusion is to capture packets. This can be done using jNetPcap, a Java variant of Pcap. It reads all the network interfaces, then selects one and read all the packets coming to that interface. The packet can then be analysed to check the source address, destination address, the protocol used and the content of the packet. The packet capturing will be done in PROMISCUOUS_MODE. It means that all the packet will be captured, even those who are not intended to be for the host machine. A PcapPacketHandler will be created which will provide functions for handling each packet. The packets can then be used to get the packet header and content. A loop will be set with the PcapPacketHandler for defining the number of packets to be captured.

MODULE 1: PACKET FILTERING

```
1: Interface interface[] = check_interfaces();
        // check          for available interfaces
2: if(size_of(interfaces)==0)
         // size_of returns number of elements in
array
3:       return error;
4: else
5:   interface = choose_inerface(interfaces);
        // Recursive method to choose an
interface
6:   if(!Interface.availability(interface))
7:       return error;
8:   else
9:                                          Packet
```

```
packet[]=interface.capture_packet(interface);
        /*Captures the packets and returns an
array   with 2 values*/
10:  Header header= packets[0];
        // Header is in packets[0]
11:  Content content= packets[1];
        //Content is on packets[1]
```

3.2.2 Signature-Based Intrusion Detection

The packet headers will be checked for predefined signatures. If there is a match between the packet header and the malicious signatures, then an alarm will be issued to the user describing the type of intrusion. The Signatures can be defined separately and the packet headers can be compared for each type of signature.

MODULE 2: SIGNATURE BASED INTRUSION DETECTION

```
1: do
2:      Boolean          val          =
Header.match_Signature(header);    /*Boolean
method to check if the
        signature matches or not*/
3:      if(val)
4:              Intrusion.alarm(USER);
        //Send an alarm to the user
5:      System.send_info(USER,
                        Intrusion.type(header));
```

```
        //Send the type of intrusion to the user
6:      else
7:              continue;
7:  while(!checked_all_signatures(header));
```

3.2.3 Anomaly Based Intrusion Detection

This algorithm will work by implementing the proposed machine learning model. The model will be trained using a data set of packets. The training data can be divided into three sets.

a. Training Set
b. Cross Validation set
c. Test Set

And the training error can be computed with the cost functions derived from each set.

MODULE 3: ANOMALY BASED INTRUSION DETECTION

```
1: cur_Model = System,train_Model(data_set);
2:              threshold = System.define_threshold(cur_Model);
3: output = System.predict_output(cur_Model);
4: if(output>threshold)
5:      return "anomaly";
6: else
7:      return "normal";
8: System.Model.add(cur_Model);
```

CHAPTER – 4

SYSTEM DESIGN

4.1 USE CASE DIAGRAM

Use case diagram depicts the different modules with which the user interacts. The user can start the programme whenever he wants to deploy the IDS to check for packets that contain malicious signatures. The IDS will perform Rule-based as well Anomaly based detection to check whether there is something wrong with the incoming packets. Next, the user can update the Signature database for Rule-Based Detection after a new signature has been found for faster processing. The user can also train the classifier with new signatures to create a better model as compared to the previous model. Finally, the output file can be accessed by the user to check for any alarms indicated by the IDS while going through the network traffic.

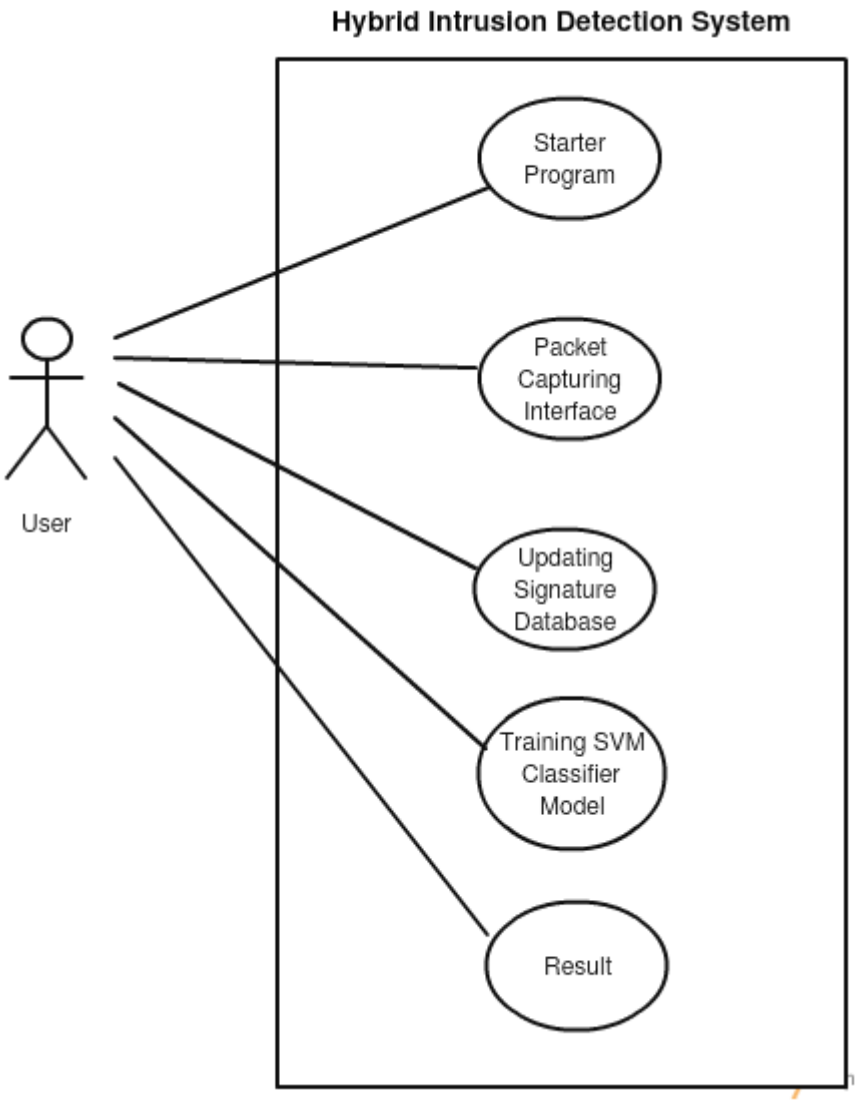


Figure 4.1: Use Case Diagram.

a. ACTIVITY DIGRAM

An activity diagram is a flow chart that depicts the data flow. Whatever algorithms and techniques the IDS uses to check for any malicious content in the packets is given step by step, until the final result is obtained.

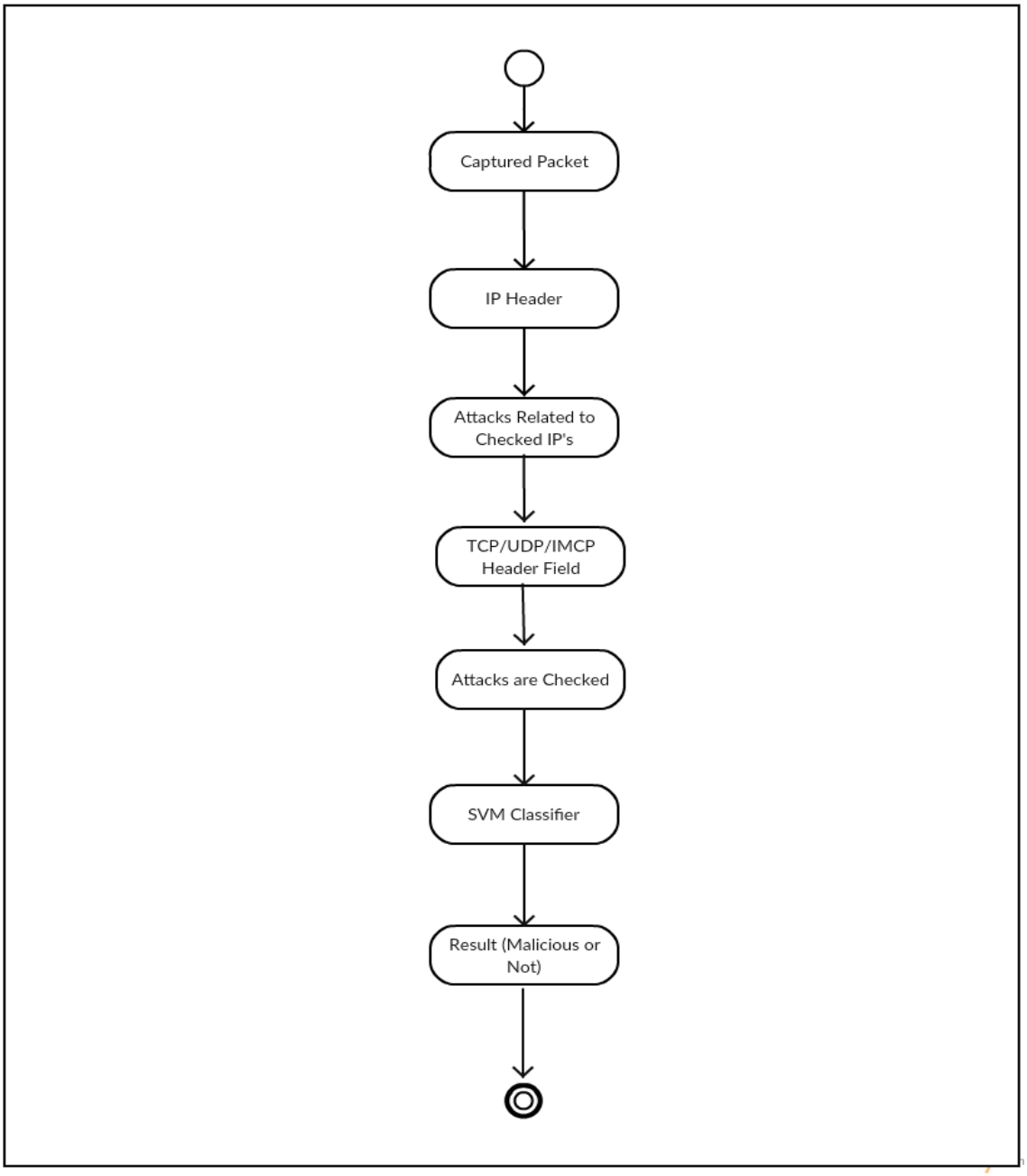


Figure 4.2: Activity Diagram.

4.3 SEQUENCE DIAGRAM

A sequence diagram is a UML artefact that is used for depicting dynamic modelling. It models the collaboration of objects used based on the structural organisation and based on a time sequence. It describes the interaction of each module along with the user focusing on the sequence in which messages are generated and exchanged. Here the doctor starts the programme. This starts the packet capturing interface of the programme which starts capturing all the packets coming to the system. These packets are then checked for any malicious signatures by comparing IP and TCP headers of the packets with the predefined rules. If a match is found, then the user is alarmed regarding the malicious nature of the packets. After the packets pass the Rule-based detection, they are checked for any anomalies by sending them through an SVM classifier. This classifier classifies the packet as malicious or normal and sends the output to the user.

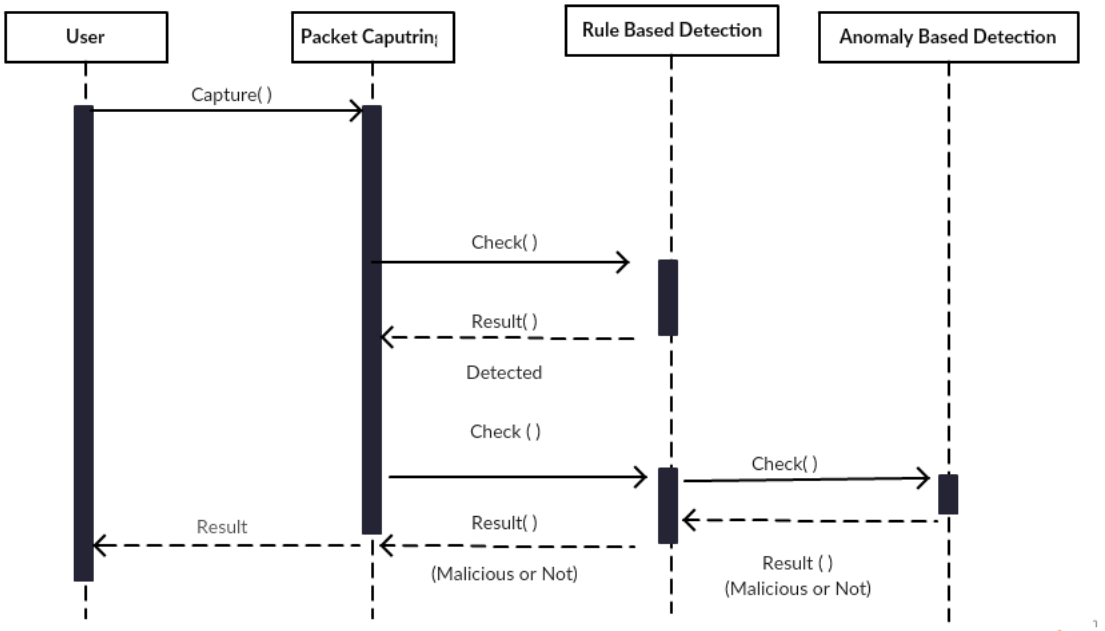


Figure 4.3: Sequence Diagram.

CHAPTER - 5

IMPLEMENTATION

5.1 ANALYSIS OF NSL-KDD DATA SET

The KDD training dataset has 10% of original dataset that is approximately 494,020 single connection vectors. Each of the single connection vectors contains 41 features and it is labelled with one of the specific attack type, i.e., either normal or attack. Anything which has even a little deviation from the ‘normal behaviour' it is considered as ‘abnormal' and hence considered as attacks. [18] Attacks defined as normal are the ones with normal behaviour. A small set of data says, 10% of training dataset is also provided for memory constrained machine learning methods. The training dataset contains 19.69% of normal connections and 80.31% of attack connections. KDD CUP 99 has been most widely used in attacks on the network.

The NSL-KDD data set is the improvised version of the KDD cup99 data set [5]. Many types of analysis have been carried out by researchers on the NSL-KDD dataset to obtain better intrusion detection system.

The protocols in KDD dataset are UDP, TCP and ICMP that are explained below:

1. TCP: TCP is "Transmission Control Protocol". TCP is a crucial protocol at the Transport Layer in internet protocol suite of OSI model. It is a reliable connection-oriented protocol where data is sent from one end to another in the same order. It takes the header part of the data and splits it into differently labelled packets and then sends them across the network. Protocols such as Email Transfer and HTTP employ TCP.

2. UDP: UDP stands for "User Datagram Protocol". It has similar behaviour to TCP except that it is unreliable and connectionless protocol. The data travels over unreliable media; the data may not reach in the same order, few packets may be missing and redundancy of packets in the network is possible. It is a transaction-oriented protocol. It is useful in situations where error checking and correction is possible in application level.

3. ICMP: ICMP stands for "Internet Control Message Protocol". It is used for establishing the connection between two connected computers. The purpose of ICMP is

to send messages to networked computers. It redirects the messages and it is also used by routers to provide the up-to-date routeing information to hosts. After receiving ICMP redirect message, host modifies its routeing table accordingly

TABLE 5.1: Type of Attacks Grouped By Protocol

Protocol_Type	**Attack_Name**
UDP	Teardrop, rootkit normal, satan, nmap.
TCP	ftp_write, normal, Neptune, guess_passwd, Perl, land, portsweep, buffer_overflow, phf, warezmaster, multihop, warezclient, back, loadmodule, satan, spy, imap, rootkit, ipsweep.
ICMP	normal, nmap, portsweep, smurf, satan, pod, ipsweep.

The attack classes present are grouped into following categories:

1. DOS: Denial of service is an attack category, where the victim's resources get depleted thereby making it unable to handle legitimate requests. Example: syn flooding. Features: "percentage of packets with errors" and "source bytes".

2. Probing: Probing attack's objective is to gain information about the remote victim. Example: port scanning. Features: "source bytes" and "duration of connection".

3. U2R: Access to root privileges by a remote user without proper authorization leads to this type of attack. Example: buffer overflow attacks. Features: "number of shell prompts invoked," and "number of file creations".

4. R2L: Unauthorised access from a remote machine, the attacker intrudes into a remote machine and gains local access of the victim machine. Example: password guessing. Features – "service requested", "duration of connection" and host level.

For every record, there are 41 attributes, which can be used to label them as an attack or a normal type. The 42nd attribute is used to classify the type of attack that the vector contains. The attack classes are further classified into Probe, DoS, U2R and R2L.

TABLE 5.2: Basic Features Of Each Network Connection Vector

Attribute No.	Attribute Name	Description	Sample Data
1	Duration_connection	Time for which the connection was set	0
2	Type_Protocol	Type of protocol used	Udp
3	Service_used_destination	Service used in destination network	udp_data
4	Flag	Flag stating connection of the system	SF
5	Source_bytes	Bytes transferred from source to destination	500
6	Destination_bytes	Bytes transferred from destination to source	10
7	Land_attack	Describing land attack- 1 for true, 0 for false	0
8	Fragments_wrng	Irrelevant or wrong bytes in the connection	0
9	Urgent_packets	Urgent bit	0

TABLE 5.3: Content Related Features Of Each Network Connection Vector

Attribute No.	**Attribute Name**	**Description**	**Sample Data**
10	Hot_indicators	Hot indicators	0
11	Number_of_failed_logins	Failed login attempts	1
12	Logging_status	1: successfully logged in 0: otherwise	1
13	Compromised_conditions	Compromised conditions	0
14	Shell_root	1: root shell 0: otherwise	0
15	Attempted_su	1: so root command attempted 0: otherwise	0
16	Root_access_num	Operations performed as root	3
17	File_creations_num	File creation operations	0
18	Shell_prompts_num	Number of shell prompts	0
19	Files_access_num	Operations for access of files	1
20	outbound_cmds_num	Outbound commands in a session	0
21	Hot_login	1: if belongs to hot list 0 : otherwise	1
22	Guest_login	1: guest login 0: otherwise	0
23	Count	Number of connections to the same destination host	5
24	Count_service	Connections in the service port as present connection	2
25	Rate_of_serror	Number of connections which have activate the four flags(23)	0

26	Serror_server_rate	Number of connections which have activate the 4 flags(24)	0
27	Rate _rerror	Connections activating REJ(4) flag among the connections(23)	0
28	Rerror_server_rate	Connections activating REJ(4) flag among the connections(24)	1
29	Serv_rate_same	Connections with the same service	1
30	Serv_rate_diff	Connections with different service	0
31	Serv_rate_diff_host	Connections with different service(24)	0

TABLE 5.4: Time Related Traffic Features Of Each Network Connection Vector

Attribute No.	**Attribute Name**	**Description**	**Sample Data**
32	host_count_destination	Connections having same IP address	200
33	Dst_host_port_count	Connections having same port number	32
34	Dst_host_same _service	Connection related to same service in host_count_destination (32)	0.22
35	Dst_host_diff_service	Connections associated to different services. Connections are aggregated into dst_host_count (32)	0.03
36	Destination_host_same _src_port	Connections are associated to the same source port and aggregated in Dst_host_port_count (33)	0.31
37	Destination_host_srv_ diff_host	Connection sent to different machines in Dst_host_port_count(33)	1
38	Destination_host_srv_s	connections that have	0

	error	activated the four flags (32)	
39	Destination_host_srv_s error	connections that have activated the four flags (33)	3
40	Destination_host_rerro r	REJ flag activated connections host_count_destination. (32)	1.05
41	Destination_host_srv_r error	connections that have flag (4) REJ bit set.(33)	2

5.2 WIRELESS SENSOR NETWORK MODEL

Wireless sensor network has many constraints levied upon it due to which the intrusion detection system developed has to be specialised to work in this model. Some of the constraints are as follows:-

- The sensor nodes have limited battery life due to which higher computations can't be carried out since they lead to battery usage.
- The computational resources available are limited since sensor nodes are independent devices which work on limited batteries, hence providing it with high-end computational resources is not possible.
- Due to the energy constraints of the nodes, they remain dormant if they are not carrying out any computation. Hence, an all-time active system cannot be installed on these nodes.

For instaling an Intrusion Detection System on this network, it has to be crafted according to the above-mentioned constraints. It has to be offline/online depending on the status of the node; it has to perform computations using the limited computational resource constraint and it has to be aware of the limited battery component of each node.

The detection agents have to be deployed to take care of packets going in and out of the local nodes as well as to check the status of their neighbours so as to get an insight of the network. Any intrusion found in network traffic has to be reported back to the base station. The base station then forwards the alert to the user. The alert can be perpetrated to the base station using the underlying architecture of the network. The nodes should provide some knowledge to the IDS. This knowledge can be local information about the

suspicious nodes present in the network and a local repository of alerts, the other type of knowledge can be about the neighbours which can be generated beforehand. The internal alert database should contain the following fields: time of creation, classification and source of alert.

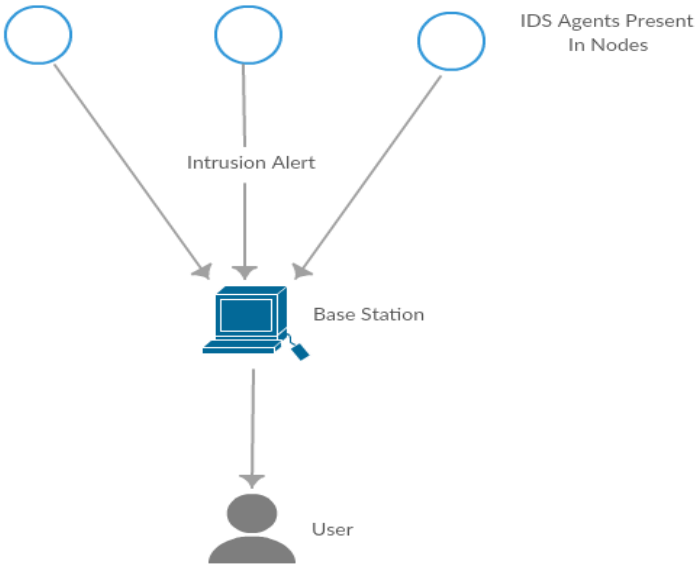


Figure 5.1: Communication between IDS agents and user

5.3 PERFORMANCE METRICS IN INTRUSION DETECTION SYSTEM

1. Cost

Cost is an important metric in determining the feasibility of the IDS. Costs related to the IDS can further be classified into the following three categories

a. Damage Cost

b. Response Cost

c. Operational Cost

Damage Cost is the amount of damage to the target resources by an attacker when intrusion detection is unavailable or ineffective. The value of the target is measured by the resources and its role in the organisation. Example the cost of root is greater than the cost of local user access.

Response Cost is the effective cost to alarm the system or to make the log whenever an intrusion is caught in action. Cost is measured regarding the system reboot and additional packet filters if applicable; it also includes the cost of recovery from

failure and the human resource required to carry out the response task.

Operation Cost is the set of events being monitored by the IDS and the analysing activities made using IDS. The amount of time for computing and the resources required to extract and test the features. It should act as a *real-time IDS*. Other additional costs can be illegal access to the root and Denial of Service.

2. Detection Rate

Detection rate gives the measure of the effectiveness of IDS. Detection rate can be viewed regarding *false positives* and *false negatives.* False positive refers to giving an intrusion alarm when there was no intrusion involved. False negative refers to not sounding an intrusion alarm when there was an intruder involved. A high number of false positives and false negatives are undesirable in IDS. A developer has to balance the two since if the false positive rate is lowered then false negative rate increases and vice versa.

3. Scalable and Resilient to Attacks

The institution using the IDS might be continuously growing. Hence the IDS must be able to adapt to this increasing traffic. It must be able to handle increased number of attacks on this increased traffic. It should be resilient to new types of attacks and the database used should be dynamic to incorporate the increasing signatures. It should be able to identify new signatures to detect intrusions.

CONCLUSIONS AND FUTURE WORK

The current configuration uses a high-level language such as Java. To make it lightweight, we can implement it using Bash Script. Bash Scripts can be run easily across Linux systems and provide faster and reliable computations. Bash scripts can be used to trigger other type of scripts such as a python script resulting in high-end computations being performed easily. It can be configured to run when the system comes online and to pause when the system goes to sleep. Furthermore, the intrusion detection system should be configured in such a way that minimal battery consumption takes place and minimal hardware resources are required.

To conclude the paper, the whole idea can be summarized as follows. Wireless sensors are used to provide continuous reading and mapping of some specialised data ranging from thermal readings to seismic readings. These nodes are susceptible to a wide range of attacks which can be carried out by notorious users. To counter these attacks, an Intrusion Detection System provides the second line of defense. It is of two types – Rule based IDS and Anomaly based IDS. A hybrid intrusion detection system is a combination of both. The proposed hybrid approach provides protection from wide range of attacks. It has the lower false negative rate.

Hence, the proposed model provides:

1. Lower false negative rates
2. Detection of a wide range of attacks
3. Scalability options; and
4. Network protection.

APPENDICES

Package Capture and Rule Based Detection Class

```
package final_project;
import java.io.IOException;
import java.util.ArrayList;

public class PacketCapture {
   public static void main(String[] args) throws IOException {
        List<PcapIf> devs = new ArrayList<>(); // Will be filled with NICs
        StringBuilder errbuf = new StringBuilder(); // For any error msgs
        int r = Pcap.findAllDevs(devs, errbuf);
        if (r != Pcap.OK || devs.isEmpty()) {
                System.err.printf("Can't read list of devices, error is %s",
                                errbuf.toString());
                return;
        }
        System.out.println("Network devices found:");
        int i = 0;
        for (PcapIf device : devs) {
                String description = (device.getDescription() != null) ? device
                                .getDescription() : "No description available";
                System.out.printf("#%d: %s [%s]\n", i++, device.getName(),
                                description);
        }
        PcapIf device = devs.get(2); // Get the interface on which you think traffic is
located
        System.out.printf("\nChoosing '%s' on your behalf:\n",
                        (device.getDescription() != null) ? device.getDescription()
                                        : device.getName());
```

```
        final byte[] maca = device.getHardwareAddress();
        String macI = asString(maca);
        ArrayList<String> reservedIp = new ArrayList<String>();
        reservedIp.add("192.168.1.4");
        reservedIp.add("192.168.1.1");
        reservedIp.add("192.168.1.7");
        reservedIp.add("172.16.0.3");
        System.out.printf("%s=%s\n", device.getName(), macI);
    int snaplen = 64 * 1024;
    int flags = Pcap.MODE_PROMISCUOUS;
    int timeout = 10 * 1000;
    Pcap pcap = Pcap.openLive(device.getName(), snaplen, flags, timeout, errbuf);
    if (pcap == null) {
            System.err.printf("Error while opening device for capture: "
                            + errbuf.toString());
            return;
    }
    PcapPacketHandler<String> jpacketHandler;
jpacketHandler = new PcapPacketHandler<String>() {
    public void nextPacket(PcapPacket packet, String user) {
        byte[] data = packet.getByteArray(0, packet.size()); // the package data
        byte[] sIP = new byte[4];
        byte[] dIP = new byte[4];
        Ip4 ip = new Ip4();
        int payload;
        Ethernet ethernet = new Ethernet();
        if (packet.hasHeader(ip) == true) {
            sIP = packet.getHeader(ip).source();
            dIP = packet.getHeader(ip).destination();
            byte[] eth = packet.getHeader(ethernet).source();
            String sourceIP = org.jnetpcap.packet.format.FormatUtils.ip(sIP);
```

```
                String destinationIP = org.jnetpcap.packet.format.FormatUtils.ip(dIP);
                String mac = FormatUtils.mac(eth);
                System.out.println(mac);
                System.out.println("srcIP=" + sourceIP +
                        " dstIP=" + destinationIP +
                        " caplen=" + packet.getCaptureHeader().caplen());
                if (sourceIP.equals(destinationIP)) {
                   System.err.println("Source and Destination address same : Possible IP spoofing : Land Attack");
                }
                ipSpoofing(mac, macI, sourceIP, reservedIp);
            }
            Tcp tcp =new Tcp();
            if (packet.hasHeader(tcp) == true) {
                payload = ip.getOffset() + ip.size();
                JBuffer buffer = new JBuffer(payload);
                buffer.peer(packet, payload, packet.size() - payload);
                String dataS = buffer.toHexdump();
                //System.out.println(dataS);
                tcp = packet.getHeader(tcp);
                synFin(tcp, packet);
                fin(tcp, packet);
                nullPacket(tcp, packet);
                reserved(tcp, packet);
                portCheck(tcp, packet);
                ackCheck(tcp, packet);
                synCheck(tcp, packet, dataS, reservedIp);
                destinationCheck(packet);
            }
            Udp udp = new Udp();
            if (packet.hasHeader(udp) == true) {
```

```
            portCheckUdp(udp, packet);
        }
        Icmp icmp =new Icmp();
        if (packet.hasHeader(icmp) == true) {
            payload = ip.getOffset() + ip.size();
            echoCheck(payload, packet);
        }
        Ip6 ip6 = new Ip6();
        if (packet.hasHeader(ip6) == true) {
          // System.err.println("IPv6 packet detected : suspicious");
        }

        AnomalyCheck ac = new AnomalyCheck();
        ac.anomalyCheck(pk);*/
      }
    };
      // capture first 10 packages
      pcap.loop(-1, jpacketHandler, "jNetPcap");
      pcap.close();
     }
private static String asString(final byte[] mac) {
    final StringBuilder buf = new StringBuilder();
    for (byte b : mac) {
      if (buf.length() != 0) {
        buf.append(':');
      }
      if (b >= 0 && b < 16) {
        buf.append('0');
      }
      buf.append(Integer.toHexString((b < 0) ? b + 256 : b).toUpperCase());
    }
```

```
        return buf.toString();
    }
    private static void ipSpoofing(String mac, String macI, String sourceIP,
ArrayList<String> reservedIp) {
        if (! mac.equals(macI) && !reservedIp.contains(sourceIP)) {
            if (sourceIP.startsWith("10.") || sourceIP.startsWith("192.168.") ||
                sourceIP.startsWith("169.254.")) {
                System.err.println("Possible IP spoofing using private networks detected");
            }
            else if (sourceIP.startsWith("172.")) {
                for(int i=16; i<=31; i++) {
                    if (sourceIP.startsWith("172."+i+".")) {
                        System.err.println("Possible IP spoofing using private networks
detected");
                        break;
                    }
                }
            }
        }
    }
   private static void synFin(Tcp tcp, PcapPacket packet) {
        if (tcp.flags_FIN() == true && tcp.flags_SYN() == true) {
            System.err.println("Malicious packet containing illegal syn-fin combination");
            Ethernet ethernet = new Ethernet();
            System.err.println("MAC address of the malicious agent : "
+FormatUtils.mac(packet.getHeader(ethernet).source()));
        }
    }
    private static void fin(Tcp tcp, PcapPacket packet) {
        if (tcp.flags_FIN() == true && tcp.flags_ACK() == false && tcp.flags_CWR() ==
false &&
```

```
            tcp.flags_ECE() == false && tcp.flags_PSH() == false && tcp.flags_RST() ==
false &&
             tcp.flags_SYN() == false && tcp.flags_URG() == false) {
               System.err.println("Malicious packet containing only fin bit found : "
                     + "Possible attack - port scan, network mapping, stealth activities");
               Ethernet ethernet = new Ethernet();
         System.err.println("MAC address of the malicious agent : "
+FormatUtils.mac(packet.getHeader(ethernet).source()));
      }
   }
   private static void fin(Tcp tcp, PcapPacket packet) {
      if (tcp.flags_FIN() == true && tcp.flags_ACK() == false && tcp.flags_CWR() ==
false &&
            tcp.flags_ECE() == false && tcp.flags_PSH() == false && tcp.flags_RST() ==
false &&
             tcp.flags_SYN() == false && tcp.flags_URG() == false) {
               System.err.Println("Malicious packet containing only fin bit found: "
                     + "Possible attack - port scan, network mapping, stealth activities");
               Ethernet ethernet = new Ethernet();
         System.err.println("MAC address of the malicious agent : "
+FormatUtils.mac(packet.getHeader(ethernet).source()));
      }
   }
   private static void reserved(Tcp tcp, PcapPacket packet) {
      int res = tcp.reserved();
      if (res > 0) {
         System.err.println("reserved bit set : possibly crafted packet");
         Ethernet ethernet = new Ethernet();
         System.err.println("MAC address of the malicious agent : "
+FormatUtils.mac(packet.getHeader(ethernet).source()));
      }
```

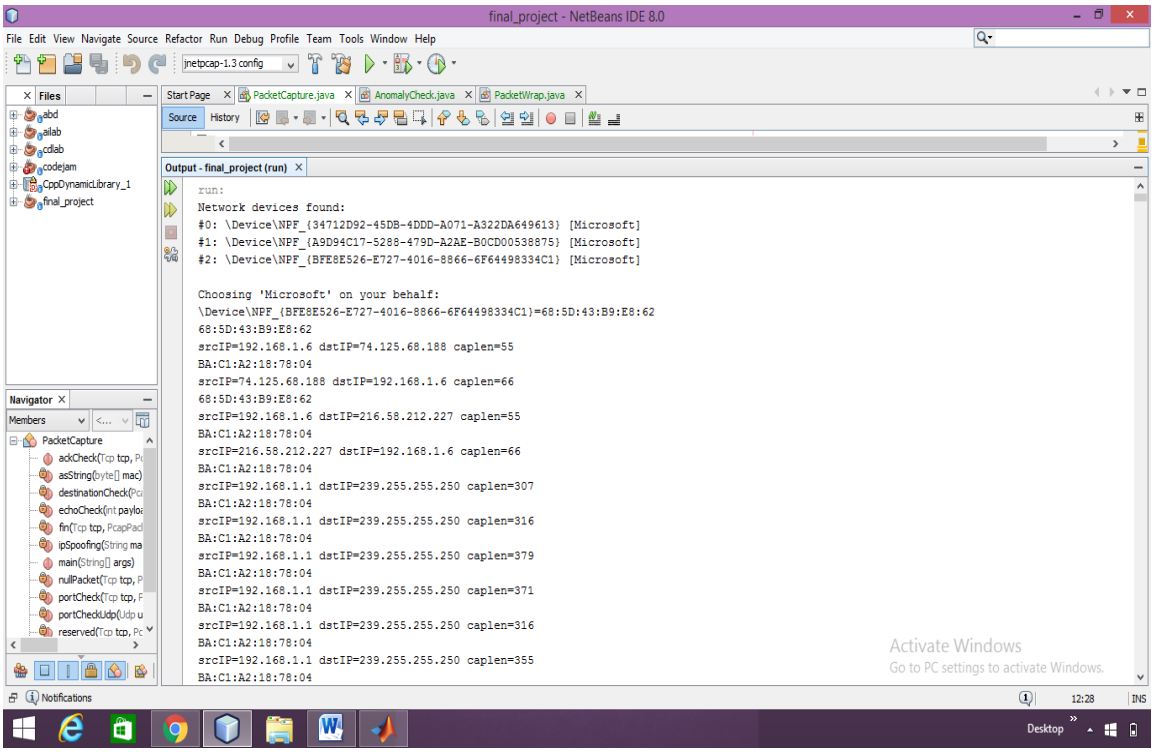
```
    }
private static void nullPacket(Tcp tcp, PcapPacket packet){
        if (tcp.flags_FIN() == false && tcp.flags_ACK() == false && tcp.flags_CWR() ==
false &&
            tcp.flags_ECE() == false && tcp.flags_PSH() == false && tcp.flags_RST() ==
false &&
            tcp.flags_SYN() == false && tcp.flags_URG() == false) {
                System.err.println("Malicious null packet found : illegal");
                Ethernet ethernet = new Ethernet();
                System.err.println("MAC address of the malicious agent : "
+FormatUtils.mac(packet.getHeader(ethernet).source()));
        }
    }
private static void synCheck(Tcp tcp, PcapPacket packet, String data, ArrayList<String>
reservedIp) {
        if (tcp.flags_FIN() == false && tcp.flags_ACK() == false && tcp.flags_CWR() ==
false &&
            tcp.flags_ECE() == false && tcp.flags_PSH() == false && tcp.flags_RST() ==
false &&
            tcp.flags_SYN() == true && tcp.flags_URG() == false &&
!reservedIp.contains(FormatUtils.ip(packet.getHeader(new Ip4()).source()))) {
            if (!data.equals(null)) {
                System.err.println("Illegal : SYN only packet should not contain any data");
                Ethernet ethernet = new Ethernet();
                System.err.println("MAC address of the suspicious agent : "
+FormatUtils.mac(packet.getHeader(ethernet).source()));
            }
        }
    }
    private static void destinationCheck(PcapPacket packet) {
        String dest = FormatUtils.ip(packet.getHeader(new Ip4()).destination());
```

```
        if (dest.endsWith(".0") || dest.endsWith(".255")) {
            System.err.println("Packets should not use destination address that is a broadcast address");
            Ethernet ethernet = new Ethernet();
            System.err.println("MAC address of the malicious agent : " +FormatUtils.mac(packet.getHeader(ethernet).source()));
        }
    }
    private static void portCheck(Tcp tcp, PcapPacket packet) {
        if (tcp.source() == 0) {
            System.err.println("Source Port cannot be zero : illegal");
            Ethernet ethernet = new Ethernet();
            System.err.println("MAC address of the malicious agent : " +FormatUtils.mac(packet.getHeader(ethernet).source()));
        }
        else if (tcp.destination() == 0) {
            System.err.println("Destination Port cannot be zero : illegal");
            Ethernet ethernet = new Ethernet();
            System.err.println("MAC address of the malicious agent : " +FormatUtils.mac(packet.getHeader(ethernet).source()));
        }
    }
    public static void ackCheck(Tcp tcp, PcapPacket packet) {
        long ack = tcp.ack();
        if (tcp.flags_ACK() == true && ack == 0 || tcp.flags_ACK() == false && ack != 0) {
            System.err.println("Illegal since acknowledgement number cannot be zero when ack flag is set");
            Ethernet ethernet = new Ethernet();
            System.err.println("MAC address of the malicious agent : " +FormatUtils.mac(packet.getHeader(ethernet).source()));
```

```
    }
  }
  private static void synCheck(Tcp tcp, PcapPacket packet, String data,
ArrayList<String> reservedIp) {
    if (tcp.flags_FIN() == false && tcp.flags_ACK() == false && tcp.flags_CWR() ==
false &&
      tcp.flags_ECE() == false && tcp.flags_PSH() == false && tcp.flags_RST() ==
false &&
      tcp.flags_SYN() == true && tcp.flags_URG() == false &&
!reservedIp.contains(FormatUtils.ip(packet.getHeader(new Ip4()).source()))) {
      if (!data.equals(null)) {
        System.err.println("Illegal : SYN only packet should not contain any data");
        Ethernet ethernet = new Ethernet();
        System.err.println("MAC address of the suspicious agent : "
+FormatUtils.mac(packet.getHeader(ethernet).source()));
      }
    }
  }
  private static void destinationCheck(PcapPacket packet) {
    String dest = FormatUtils.ip(packet.getHeader(new Ip4()).destination());
    if (dest.endsWith(".0") || dest.endsWith(".255")) {
      System.err.println("Packets should not use destination address that is a broadcast
address");
      Ethernet ethernet = new Ethernet();
      System.err.println("MAC address of the malicious agent : "
+FormatUtils.mac(packet.getHeader(ethernet).source()));
    }
  }
  private static void portCheckUdp(Udp udp, PcapPacket packet) {
    if (udp.source() == 0) {
      System.err.println("Source Port cannot be zero : illegal");
```

```
            Ethernet ethernet = new Ethernet();
            System.err.println("MAC address of the malicious agent : "
+FormatUtils.mac(packet.getHeader(ethernet).source()));
        }
        else if (udp.destination() == 0) {
            System.err.println("Destination Port cannot be zero : illegal");
            Ethernet ethernet = new Ethernet();
            System.err.println("MAC address of the malicious agent : "
+FormatUtils.mac(packet.getHeader(ethernet).source()));
        }
    }
    private static void echoCheck(int payload, PcapPacket packet) {
        if (payload > (65507/2)+1 ) { /* maximum allowable ip packet size is 65535 bytes
                                    including ip header of 20 bytes, icmp contains a pseudo header
of
                                    8 bytes along with this ip header*/
            System.err.println("Unusual large size of echo request : suspicious");
            Ethernet ethernet = new Ethernet();
            System.err.println("MAC address of the malicious agent : "
+FormatUtils.mac(packet.getHeader(ethernet).source()));
        }
    }
```

Output



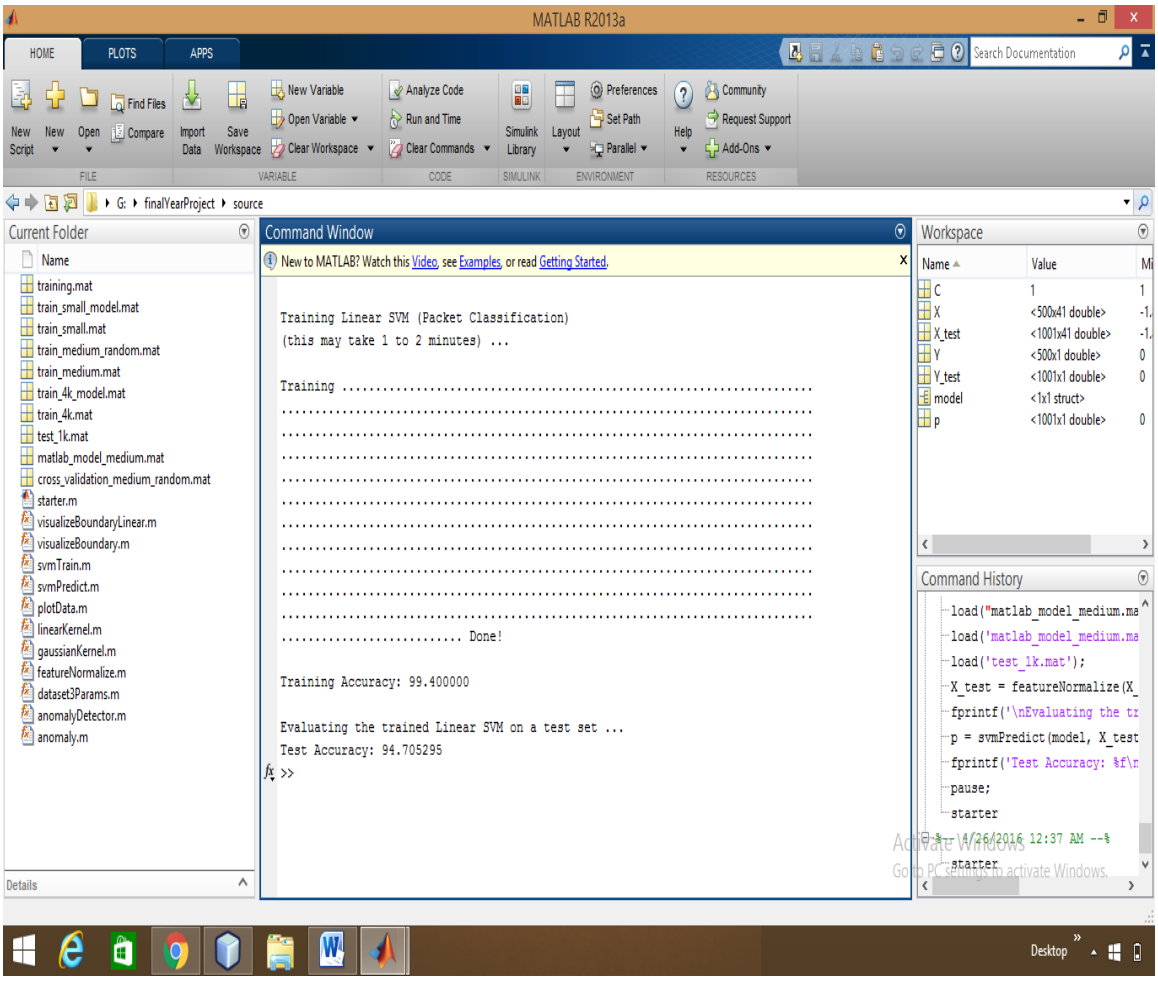
Module for defining Linear Kernel

```
function sim = linearKernel(x1, x2)
%LINEARKERNEL returns a linear kernel between x1 and x2
%   sim = linearKernel(x1, x2) returns a linear kernel between x1 and x2
%   and returns the value in sim
 % Ensure that x1 and x2 are column vectors
x1 = x1(:); x2 = x2(:);

% Compute the kernel
sim = x1' * x2;  % dot product
```

```
end

Module containing starter programme

%% Initialization
Clear; close all; ccc

%% ==================== Part 1: Packets processing ====================

load('train_small.mat');
X = featureNormalize(X);
pause;
%load('cross_validation_medium_random.mat');
%[C, sigma] = dataset3Params(X, Y, Xval, yval);
%pause;
%plotData (X,Y);
pause;
%% ========== Part 2: Train Linear SVM for Packet Classification ========
%  In this section, you will train a linear classifier to determine if a
%  packet is malicious or not

fprintf('\nTraining Linear SVM (Packet Classification)\n')
fprintf('(this may take 1 to 2 minutes) ...\n')
C = 1;
model = svmTrain(X, Y, C, @linearKernel);

p = svmPredict(model, X);

fprintf('Training Accuracy: %f\n', mean(double(p == Y)) * 100);
```

```
%% ================== Part 3: Test anomaly Classification ================
%  After training the classifier, we can evaluate it on a test set. We have
%  included a test set in

% Load the test dataset
% You will have Xtest, test in your environment

load('test_1k.mat');
X_test = featureNormalize(X_test);
fprintf('\nEvaluating the trained Linear SVM on a test set ...\n')

p = svmPredict(model, X_test);

fprintf('Test Accuracy: %f\n', mean(double(p == Y_test)) * 100);
pause;
```

Output


MATLAB R2013a
HOME
PLOTS
APPS
Search Documentation
New Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Analyze Code
Run and Time
Clear Commands
Simulink Library
Layout
Preferences
Set Path
Parallel
Help
Community
Request Support
Add-Ons
FILE
VARIABLE
CODE
SIMULINK
ENVIRONMENT
RESOURCES
G: ▸ finalYearProject ▸ source
Current Folder
Name
training.mat
train_small_model.mat
train_small.mat
train_medium_random.mat
train_medium.mat
train_4k_model.mat
train_4k.mat
test_1k.mat
matlab_model_medium.mat
cross_validation_medium_random.mat
starter.m
visualizeBoundaryLinear.m
visualizeBoundary.m
svmTrain.m
svmPredict.m
plotData.m
linearKernel.m
gaussianKernel.m
featureNormalize.m
dataset3Params.m
anomalyDetector.m
anomaly.m
Details
Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.
Training Linear SVM (Packet Classification)
(this may take 1 to 2 minutes) ...
Training .....
Done!
Training Accuracy: 99.400000
Evaluating the trained Linear SVM on a test set ...
Test Accuracy: 94.705295
>>
Workspace
Name
Value
C 1 1
X <500x41 double> -1.
X_test <1001x41 double> -1.
Y <500x1 double> 0
Y_test <1001x1 double> 0
model <1x1 struct>
p <1001x1 double> 0
Command History
load("matlab_model_medium.ma
load('matlab_model_medium.ma
load('test_1k.mat');
X_test = featureNormalize(X_
fprintf('\nEvaluating the tr
p = svmPredict(model, X_test
fprintf('Test Accuracy: %f\n
pause;
starter
%% 4/26/2016 12:37 AM --%
starter
Activate Windows
Go to PC settings to activate Windows.
Desktop

REFERENCES

[1] Ali, Q. I., and S. Lazim. *"Design and Implementation of An Embedded Intrusion Detection System For Wireless Applications."* IET Information Security 6, No. 3 (September 2012): 171–82. Doi:10.1049/Iet-Ifs.2010.0245.

[2] Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, *"Wireless sensor networks: A survey,"* Comput. Netw., vol. 38, no. 4, 2002.

[3] Chapke Prajkta P., Raut A. B.*"Hybrid Model for Intrusion Detection System"* International Journal Of Engineering And Computer Science ISSN: 2319-7242 Volume1 Issue 3 Dec 2012.

[4] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos*,"Context aware computing for the internet of things: A survey,"* IEEE Commun. Surv. Tutorials, vol. 16, no. 1, 2014.

[5] Gaffney, J. E., and J. W. Ulvila. *"Evaluation of Intrusion Detectors: A Decision Theory Approach."* In 2001 IEEE Symposium on Security and Privacy, 2001. S P 2001. Proceedings, 50–61, 2001. doi:10.1109/SECPRI.2001.924287.

[6] *http://searchsecurity.techtarget.com/definition/HIDS-NIDS accessed on 26.03.2016.*

[7] J.Antony Jeyanna, E.Indumathi, Dr.D.Shalini Punithavathani, *"A Network Intrusion Detection System Using Clustering And Outlier Detection",* International Journal Of Innovative Research In Computer And Communication Engineering, Vol. 3 Issue 2, February 2015.

[8] Jeff Howbert,*"Introduction To Machine Learning".*

[9] Kai Hwang, Fellow, IEEE, Min Cai, Member, IEEE, Ying Chen, Student Member, IEEE, And Min Qin *"Hybrid Intrusion Detection With Weighted Signature Generation Over Anomalous Internet Episodes"* IEEE Transactions On Dependable And Secure Computing, Vol. 4, No. 1, January – March 2007.

[10] Kayvan Atefi1, Saadiah Yahya2, Ahmad Yusri Dak3, And Arash Atefi4, *"A Hybrid Intrusion Detection System Based On Different Machine Learning Algorithms"*, 4th International Conference on Computing And Informatics, ICOCI- August, 2013.

[11] Manish Somani, Roshni Dubey, *"Hybrid Intrusion Detection Model Based On Clustering And Association"*, International Journal Of Advanced Research In Electrical Electronics And Instrumentation, Vol. 3, Issue 3, March 2014.

[12] N. Wattanapongsakorn, S. Srakaew, E. Wonghirunsombat, C. Sribavonmongkol, T. Junhom, P. Jongsubsook, C. Charnsripinyo, *"A Practical Network based Intrusion Detection and Prevention System"* 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications.

[13] Pan, S., T. Morris, And U. Adhikari. *"Developing A Hybrid Intrusion Detection System Using Data Mining For Power Systems." IEEE Transactions On Smart Grid* 6, No. 6 (November 2015): 3104–13. Doi:10.1109/TSG.2015.2409775.

[14] Pharate, Abhishek, Harsha Bhat, Vaibhav Shilimkar, and Nalini Mhetre. *"Classification of Intrusion Detection System." International Journal of Computer Applications* 118, no. 7 (May 20, 2015): 23–26. Doi: 10.5120/20758-3163.

[15] Rodrigo Roman, Jianying Zhou, Javier Lopez, *"Applying Intrusion Detection Systems To Wireless Sensor Networks".*

[16] Shaik Akbar, Dr. K. Nageswara Rao, Dr. J. A. Chandulal. *"Implementing rule based genetic algorithm as a solution for intrusion detection system"* IJCSNS International Journal of Computer Science and Network Security, VOL. 11 No. 8, August 2011.

[17] Sandip Ashok Shivarkar Mininath Raosaheb Bendre *"Hybrid approach for Intrusion detection using conditional random fields"* International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume 1, Issue 3.

[18] Sanjay Kumar Sharma, Pankaj Pande, Susheel Kumar Tiwari and Mahendra Singh Sisodia ,*"An Improved Network Intrusion Detection technique based on k-means clustering via naïve Byes Classification"* IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012) March 30, 31, 2012.

[19] Taskin Kocak, *"Sigmoid functions and Their usage in Artificial Neural Networks"* Applications of Calculus II: Inverse Functions.

[20] Vinod Kumar, Dr. Om Prakash Sangwan, *"Signature based intrusion detection system using Snort"* International Journal of Computer Applications & Information Technology Vol. I, Issue III, November 2012 (ISSN: 2278-7720).

[21] V. Jyothsna, V. V. Rama Prasad, K. Munivara Prasad, *"A review of anomaly based intrusion detection systems"* International Journal of Computer Applications (0975 – 8887) Volume 28– No. 7, August 2011.