

RASPBERRY PI BASED SECURITY INCIDENT AND MANAGEMENT SYSTEM FOR SMALL AND MEDIUM IT SETUPS

(Implementation of Machine Learning on Raspberry Pi to detect Man In The Middle Attacks)



Author

MUHAMMAD GHUFRAN ASHRAF

NUST2015MSIS00000118419

Supervisor

DR. SHAHZAD SALEEM

A thesis submitted in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE IN INFORMATION SECURITY (MS-IS)

DEPARTMENT OF COMPUTING (DoC)
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
(SECS)
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY (NUST)
ISLAMABAD, PAKISTAN
APRIL 2019

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at National University of Sciences & Technology (NUST) School of Electrical Engineering & Computer Science (SEECs) or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECs or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Muhammad Ghufan Ashraf

Signature: _____

Acknowledgements

I am thankful to my Creator Allah Subhana-Watala to have guided me throughout this work at every step and for every new thought which You setup in my mind to improve it. Indeed I could have done nothing without Your priceless help and guidance. Whosoever helped me throughout the course of my thesis, whether my parents or any other individual was Your will, so indeed none be worthy of praise but You.

I am profusely thankful to my beloved parents who raised me when I was not capable of walking and continued to support me throughout in every department of my life.

I would also like to express special thanks to my supervisor **Dr. Shahzad Saleem** for his help throughout my thesis and also for Digital Forensics course which he has taught me.

I would also like to pay special thanks to **Dr. Naveed Ahmed** for his tremendous support and cooperation. Each time I got stuck in something, he guided me towards the solution.

Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.

Dedicated to my exceptional parents and wife whose tremendous support and cooperation led me to this wonderful accomplishment.

Abstract

We are living in the time of data innovation, where every IT setup, office, home and client needs to impart and share data. This requirement for correspondence and sharing of data has resulted in an exponential increase in number and size of computer systems. These systems are utilized to share data both inside and outside the IT setup. With the noteworthy increment in number of cyber assaults, there are various protective applications accessible in the market to identify and deal with these assaults. These incorporate Firewalls, IDS and IPS. These frameworks suits well to financially stable setups but they are pricy and beyond the access for financially constraint enterprise IT setups. The installation, maintenance and power costs of these security systems are beyond the capacity of a small and medium sized organization as they are facing difficulties in detecting and managing ever increasing network attacks.

To address this problem, this research proposes a cost and power efficient security incident and event management system for small and medium organizations using Raspberry pi computers. Raspberry pi is cheap microcomputer and has low cost and power consumption and have no installation, infrastructure and maintenance requirements. Such hardware can easily be afforded by the user sitting at home or in small and medium size IT setups.

Once installed on local area network, solution will be able to capture and analyze network traffic against commonly known man in middle attacks to log any malicious activity for better understanding of network administrator with minimum cost in terms of power and maintenance.

Key Words: *Raspberry Pi, Machine Learning, MITM attacks, Weka, Python, Wireshark & LogisticRegression.*

Table of Contents

Certificate of Originality	ii
Acknowledgements	iii
Abstract	v
Table of Contents	vi
List of Figures.....	viii
List of Tables	ix
List of Acronyms	x
CHAPTER 1: INTRODUCTION	1
1.1. Motivation.....	1
1.2. Background	1
1.3. Objective.....	2
1.3.1 MITM attacks.....	2
1.4. Components	5
1.4.1 Raspberry Pi – Intrusion detection node	6
1.4.2 Intrusion Detection System (IDS)	6
1.4.3 Security incident and event management.....	7
1.5. Thesis organization	7
CHAPTER 2: LITERATURE REVIEW.....	9
2.1. SQL Injection Detection and Prevention System with Raspberry Pi Honeypot Cluster for Trapping Attacker [9].....	9
2.2. Intrusion Detection System Using Raspberry PI Honeypot in Network Security [10].....	10
2.3. Design and Analysis of Real-time Network Intrusion Detection and Prevention System using Open Source Tools [12]	12
2.4. A firewall for SOHO networks using Raspberry Pi and Snort [14]	13
2.5. Pi-IDS: Evaluation of Open-Source Intrusion Detection Systems on Raspberry Pi 2[15]	14
2.6. Conclusion	15
CHAPTER 3: TOOLS AND METHODS	16
3.1. Raspberry Pi.....	16
3.2. Kali Linux	17
3.3. Machine Learning.....	18
3.4. Wireshark.....	19
3.5. Weka.....	20
3.6. Spyder IDE.....	21
CHAPTER 4: RESEARCH METHODOLOGY	23

4.1.	Research Process	23
4.1.1	Dataset Generation	23
4.1.2	Feature Selection	24
4.1.3	Feature Evaluation.....	24
4.1.4	Raspberry Pi Implementation.....	24
4.1.5	Design Evaluation	25
CHAPTER 5: SYSTEM DESIGN AND INTEGRATION		26
5.1.	System Architecture.....	26
5.2.	Data set Generation.....	27
5.2.1	Lab setup for Data Set Generation.....	28
5.2.2	ARP Spoofing	31
5.2.3	Port Stealing Attack.....	32
5.2.4	MAC Flooding Attack	33
5.3.	Feature Selection	37
5.4.	Feature Evaluation.....	43
5.4.1	Logistic Regression	44
5.4.2	Feature Evaluation in Weka.....	45
5.4.3	Feature Evaluation in Spyder IDE.....	49
5.4.4	Implementation on Raspberry Pi	50
CHAPTER 6: RESULTS AND DISCUSSION		56
6.1.	Weka Evaluation and Results	56
6.2.	Python Evaluation and Results	59
6.3.	Raspberry Pi Implementation Results	63
CHAPTER 7: Conclusion.....		67
7.1.	Future Directions	68
REFERENCES.....		69

List of Figures

Figure 1: ARP Spoofing Attack	3
Figure 2: MAC Flooding Attack	5
Figure 3: Proposed Design Architecture.....	5
Figure 4: Suggested Design to identify and avoid SQL injection [9]	10
Figure 5: Raspberry Pi - Honeypot deployment [10].....	11
Figure 6: Network Connectivity Diagram [12]	12
Figure 7: System Architecture for Experiment [16]	14
Figure 8: Raspberry Pi Model 2B.....	16
Figure 9: Kali Linux.....	18
Figure 10: Wireshark GUI	20
Figure 11: Weka user interface	21
Figure 12: Spyder IDE interface	22
Figure 13: Research Methodology	23
Figure 14: System Design.....	27
Figure 15: Lab setup for Data set	29
Figure 16: Sample Wireshark Pcap	30
Figure 17: ARP spoof Attack.....	31
Figure 18: Port Steal Attack using ettercap.....	32
Figure 19: Switch CAM table	33
Figure 20: Macflooding Attack.....	34
Figure 21: Data set CSV file	36
Figure 22: Filter Method for Feature Selection [41].....	37
Figure 23: Wrapper Method for Feature Selection [41].....	37
Figure 24: Weka GUI	38
Figure 25: Information Gain Ranking Filter Output.....	38
Figure 26: Correlation Ranking Filter	39
Figure 27: Gain Ratio Feature Evaluator	40
Figure 28: Best First Feature Selection.....	40
Figure 29: Data set with selected features.....	42
Figure 30: Logistic Regression Graph [46].....	45
Figure 31: Data set preprocessing using Weka	46
Figure 32: Logistic Regression using Weka.....	47
Figure 33: Python Code for Multiclass label Using Logistic regression.....	51
Figure 34: Raspberry Pi in promiscuous mode	52
Figure 35: Intrusion Detection Flow.....	53
Figure 36: Enabling auto capture in Wireshark.....	53
Figure 37: tshark command to extract features	54
Figure 38: Python code on Raspberry Pi	55
Figure 39: Graphical Plot of Weka Results.....	59
Figure 40: ROC curves for each class label	61
Figure 41: Graphical Plot of Python Results.....	62
Figure 42: Graphical Plot of Summary	63
Figure 43: Raspberry Pi Output.....	64
Figure 44: Graphical Plot - Raspberry Pi Results.....	66

List of Tables

Table 1: System Hardware Configurations	29
Table 2: Features with description.....	35
Table 3: Data Packets with Label	36
Table 4: Summary - Feature Selection Methods	41
Table 5: Weka Results	49
Table 6: Spyder IDE Results.....	49
Table 7: Weka Evaluation Metrics	56
Table 8: Confusion Matrix - Weka.....	57
Table 9: Confusion Matrix Attributes - Weka.....	58
Table 10: Python Evaluation Metrics	59
Table 11: Confusion Matrix – Python	61
Table 12: Calculated TP and FP values - Python	62
Table 13: Summary of Results	63
Table 14: Raspberry Pi Implementation Results	65

List of Acronyms

MITM	Man In The Middle
ARP	Address Resolution Protocol
SEIM	Security Incident and Event Management
DHCP	Dynamic Host Configuration Protocol
LAN	Local Area Network
IDS	Intrusion Detection System
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
ROC	Receiver Operating Characteristics
DNS	Domain Name System

CHAPTER 1: INTRODUCTION

1.1. Motivation

Organizations dealing with resource sharing are subjected towards cyber attacks. Data is the most important asset for any organization whether they are handling small or large data networks. Data security is of prime importance for all organizations. Information security insinuates the path toward protecting data from unapproved access and data defilement all through its lifecycle [1]. Variety of cyber attacks are present which results in unauthorized access of user data. Out of these attacks one of their kind are Man in Middle attacks. The Man-In-The-Middle (MITM) attack is a standout amongst the most notable assaults in cyber security, speaking to one of the greatest worries for security experts [2]. Man in Middle attack allows the attacker to get hold of information being shared between users over the network. Different types of these attacks are present including ARP poisoning, port stealing and mac flooding etc.

In order to protect data from these variety of cyber attacks, organizations used to spend lot of money on different protective solutions available in the market in form of firewalls, Intrusion detection (IDS) and prevention systems (IPS). Large organizations having no financial issues can afford these costly solutions but these can act as killing source for small and medium organizations. Such kind of network solution is needed that these financial constraint organizations can easily afford. Raspberry Pi base security solutions suits these small organizations well in terms of standalone or multiple tapping nodes deployment in network if needed due to its modularity, low maintenance and infrastructure as well as low cost.

1.2. Background

We are living in the time of data innovation, where pretty much every association, office, home and client needs to impart and share data. This requirement for correspondence and sharing of data has gotten an exponential development number and size of networks. Systems in these networks are utilized to share data both inside and outside the association. Security, privacy and

trustworthiness of these systems and the data they share has the highest priority for every IT setup.

With the critical increment in number of system assaults, there are various business applications accessible in the market to distinguish and deal with these assaults. These incorporate Firewalls, IDS and IPS. These frameworks suits well to financially stable organizations however they are expensive and pricy for minor and medium IT setups. The installation, maintenance and power costs of these security systems are beyond the capacity of a small and medium sized organization as they are facing difficulties in detecting and managing ever increasing network attacks. Raspberry Pi based security solutions suits well to these organizations. Raspberry Pi is a microcomputer with low cost and power consumption. Due to its modularity, flexibility, no infrastructure and maintenance requirements such computers once configured can proof to be protective source to these small and medium IT setups.

1.3. Objective

Once installed on local area network, solution will be able to capture and analyze network traffic against commonly known MITM attacks to log any malicious activity for better understanding of network administrator with minimum cost in terms of power and maintenance. Brief description of MITM and its types is as follows.

1.3.1 MITM attacks

In the modern world, attacks on local area network (LAN) are numerous. Basic goal of all these attacks by the attacker is to get hold of information being shared over the network between systems. Out of these attacks one of their kind are MITM attacks. In this case, the attacker often decisively captures messages between the client and the server [5]. MITM attacks can be launched on network in different ways but the commonly known infrastructure based LAN attacks are as under.

- **Address Resolution Protocol (ARP) spoofing:-** ARP is protocol used for mapping between system IP and Mac address. ARP spoofing is the process of faking ARP packets to be able to mimic another user on the system [6]. IP address is the logical entity assigned to host machine whereas mac

address is the physical or hardware address of host machine in any network. In order to implement arp spoofing, attacker tends to poison the ARP table on host machine with fake entry by mapping the IP address of gateway router with mac address of her machine. Consequently, all communications between victim machine and gateway router moves through the attacker. End systems have no idea that there communications is being listened by the third unauthorized party.

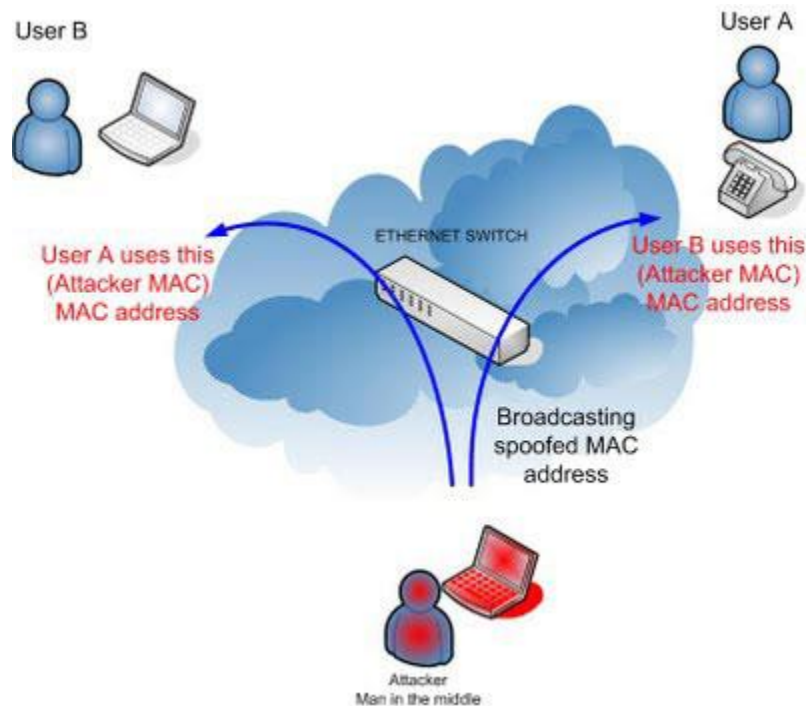


Figure 1: ARP Spoofing Attack

- Port stealing:-** Port stealing is another version of MITM attack. Port steal attacks are focused on switch ports on which host machines are connected in the network. These attacks don't target host machines directly as was in the case of ARP spoofing attack. Every switch maintains Content Address Memory (CAM) table. CAM table consisting of mac addresses of machines along with the switch port numbers on which machines are connected. Each entry indicates that the particular machine is connected on the particular switch port. This allows the switch to forward traffic to the correct machine when a particular MAC address is requested [7]. Goal of

attacker in this type of attack is to flood the CAM table with wrong entries. Port stealing is executed by the attacker by sending storm of fake ARP packets having her machine address as source mac address and mac address of victim machine as destination mac address. As a result, traffic destined for victim machine tends to get routed through attacker machine which in turn sends ARP request requiring original mac address of victim. Consequently, when attacker gets the legitimate mac address of victim machine packets are sent towards their original destination.

- **Mac Flooding:-** Mac flooding is the attack that targets CAM table of switch. It is also known as CAM flooding attack. Entries in the switch CAM table are limited meaning that there is a limit on entries depending upon the hardware of switch that a switch can hold. CAM table stores data which includes MAC addresses accessible on physical ports with their associated VLAN characteristics [8]. Goal of the attack is to eavesdrop the shared information by choking the memory of CAM table. In this attack, attacker floods the CAM table with fake source and mac addresses in each frame to utilize holding capacity of CAM table completely. As a result, CAM table of switch gets filled with the fake frames and consequently cannot handle any further entry. Once the CAM table gets full, all the data frames coming to the switch are then broadcasted over the network to all users as in the case of network hub. Consequently, attacker gets data destined for victim machine.

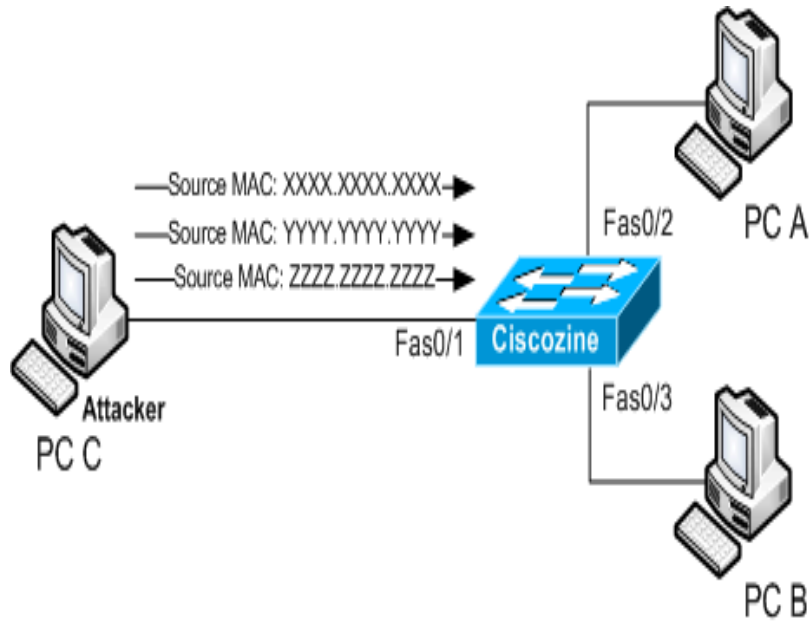


Figure 2: MAC Flooding Attack

1.4. Components

Proposed solution will consists of following components.

- Raspberry Pi - Intrusion detection node
- Security incident and event management

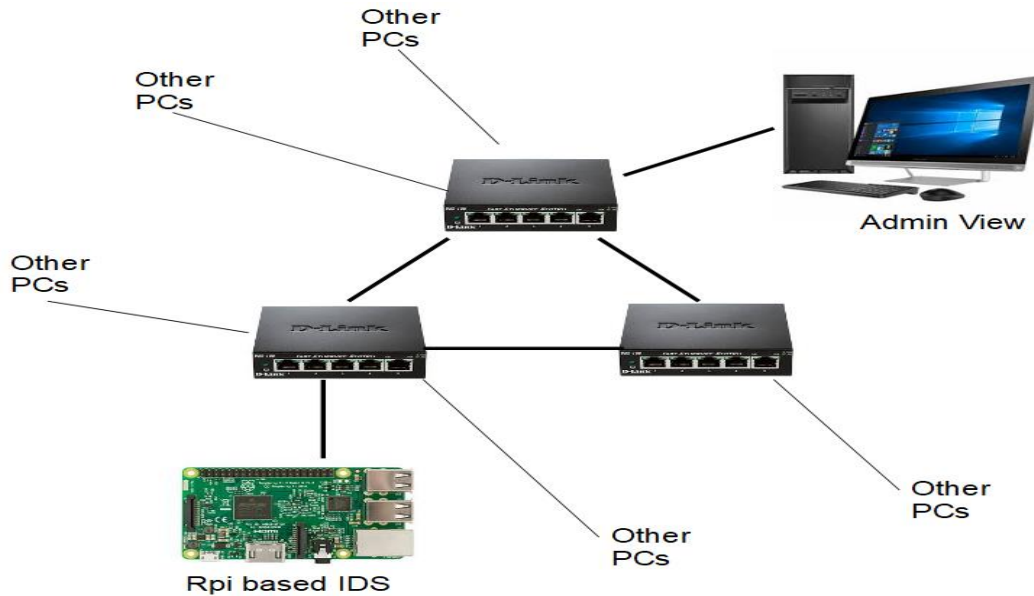


Figure 3: Proposed Design Architecture

1.4.1 Raspberry Pi – Intrusion detection node

Raspberry pi computer will be configured and installed to work as intrusion detection node in the network to sniff and perform traffic analysis to detect any security incident with respect to the security algorithms implemented on it. Basic concepts related to intrusion detection and its types are as follows.

1.4.2 Intrusion Detection System (IDS)

Intrusion detection is a arrangement that screens and checks the network for any intrusion. Intrusion refers to any malicious activity carried out by attacker or unauthorized person. Main functionality of an IDS to alert the network administrator by monitor the network traffic for any malicious pattern. Intrusion activities are logged for troubleshooting of administrator. IDS are of following types depending upon their functions.

- **Active Intrusion Detection System:** - An active IDS is a framework that is designed to naturally block alleged assaults in progress with no intercession required by an administrator [3]. They come with the advantage of providing real time detection as well as prevention of malicious activity detected over the network. Active IDS functions as inline IDS for incoming traffic and further prevents the flow of malicious traffic to systems on network.
- **Passive Intrusion Detection System:-** A passive intrusion detection system is type of method that extracts the portion of incoming traffic, perform analyses on the basis of detection rules defined and then alerts the network administrator regarding any intrusion into the network. Passive IDS doesn't provide any auto protective or corrective capability to the network as provided in the case of active IDS.
- **Network based Intrusion Detection System (NIDS):-** In this type of IDS, the detection is done at the network level. NIDS is designed and positioned at a point where it can screen all the traffic that enters or leaves the system.

It monitors system traffic on network for particular system fragments or devices and examines network and application protocol activity to detect uncertain network activity [4]. It mainly consist of sensor, network interface card (NIC) in promiscuous mode and managing interface.

- **Host based Intrusion Detection System (HIDS):-** In this type of system, IDS is installed at the host end for capturing and analyzing traffic originating as well as destined for respective host machine. HIDS displays the features of a particular host and the events happening within that host for distrustful event [4]. Intrusion detection needs to be installed at each host over the network.

1.4.3 Security incident and event management:-

Centralized logging and monitoring of any malicious pattern detected by raspberry pi will be done on separate server or host machine on same network to prompt network administrator for better understanding and management. Through centralized server, network administrator will be able to monitor the LAN status and configure raspberry pi node. Generally, any security incident and event management technology consists of following.

- **Security Event Management (SEM):-** The main function of SEM is to perform analysis on captured logs and events in real time to provide risk monitoring, event association and response.
- **Security Information Management (SIM):-** This process corresponds the collection of logs from different devices over the network to centralized location. Reports are generated on the basis of collected log files which acts as a key for network administrator to troubleshoot the reported suspicious activity.

1.5. Thesis Layout

This research is prepared and enlightened in seven chapters. First chapter explains the upbringing problem and incentive behind this research. This chapter also debates the

introduction and diverse components of the suggested security scheme which is the end result of this research.

Second chapter describes review of the correlated work already carried out. In this chapter, different cyber security procedures implemented on Raspberry Pi for intrusion detection have been analyzed and compared with our proposed solution for small and medium organizations.

Chapter three illustrates brief introduction to tools and methods incorporated in this research work. Chapter four gives research methodology followed. Chapter five gives the detail procedures and methods carried out in designing of proposed security solution and its integration.

In chapter six, output and results have been discussed in detail in order to validate the solution output. Chapter seven gives conclusion and future direction with respect to its practical implementation.

CHAPTER 2: LITERATURE REVIEW

The process of identifying malicious activity as well as detecting unauthorized access into the local network is called protecting a network or in simple words network security. In the previous chapter, agenda of our research along with the related concepts was explained keeping in view the limited resources of small and medium IT setups. Plenty of work in this regard have already been carried out by researchers. Our goal is to come up with the solution to detect intrusion against more sophisticated MITM attacks using easily affordable raspberry pi computers. In this chapter, we will carry out the review of solutions given by the researchers and engineers using raspberry pi for network security. Literature review is as follows.

2.1. SQL Injection Detection and Prevention System with Raspberry Pi Honeypot Cluster for Trapping Attacker [9]

This paper suggests a solution against sequel (SQL) injection attack using raspberry pi computers in cluster formation. The proposed solution tends to enhance the system proficiency to identify and avoid SQL injection outbreaks against database. SQL injection is kind of occurrence that targets the database of server. Invader executes malicious SQL statements that can control the web based database servers. The paper uses raspberry pi computers in cluster as a honey pot cluster. Honeypots are the systems which are the replica of actual servers with limited services to track the suspicious activity of attacker. Honeypots are accessible to the attackers without any restriction. It functions as electronic bait claiming to be an ordinary framework however sitting tight to follow saltine exercises [9].

The proposed solution uses raspberry pi computers in cluster to hide the identity as well as access of actual web servers present. Proposed design consists of web server for handling web requests from clients, proxy server for deciding whether to forward incoming requests to server or honeypot and load balancer for distributing incoming load of requests between raspberry pi computers in cluster as shown

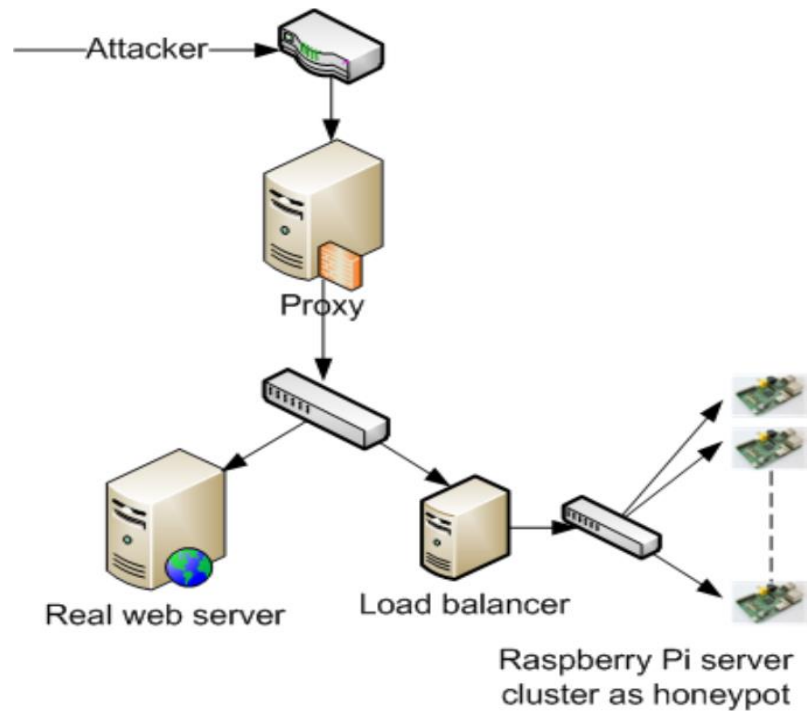


Figure 4: Suggested Design to identify and avoid SQL injection [9]

Proxy server in the design acts as detector for SQL injection attack by checking the incoming requests from client. Proxy server checks the incoming requests from client, if the request method is not POST they are forwarded towards web server. Requests having POST are passed through the SQL query deletion method, if succeeds they are forwarded to server otherwise they are sent towards honeypot cluster for logging the request as attack.

Suggested solution [9] lacks implementation of any rule forming capability as data collected by Raspberry Pi honey pot was not used to generate any prevention rule for blocking any such activity in future. On the other hand, proposed solution only focuses on SQL injection attack leaving network vulnerable against other sophisticated attacks.

2.2. Intrusion Detection System Using Raspberry PI Honeypot in Network Security [10]

The author of the paper suggest a security model for protecting network devices using combination of Raspberry Pi as honeypot and open source IDS available. A distraction based framework, Honeypot alongside Raspberry Pi makes system safety cost effective and simple to appliance [10]. Suggested design uses honeypot for capturing attacker's activity. Honeypot

displays the attack type after data analysis but plays a vital role in improving network security. Suggested architecture uses Snort IDS on raspberry pi for detection of any suspicious activity. Snort is the well known open source IDS with millions of detection rules implemented in its signature database for detecting cyber attacks. Snort IDS is a protective tool of network safety. It has been broadly used for shielding the network of IT based enterprise setups [11].

Detection mechanism works on the basis of client server architecture. It uses central main server interacting with requests from multiple clients on network. Proposed layout is as shown

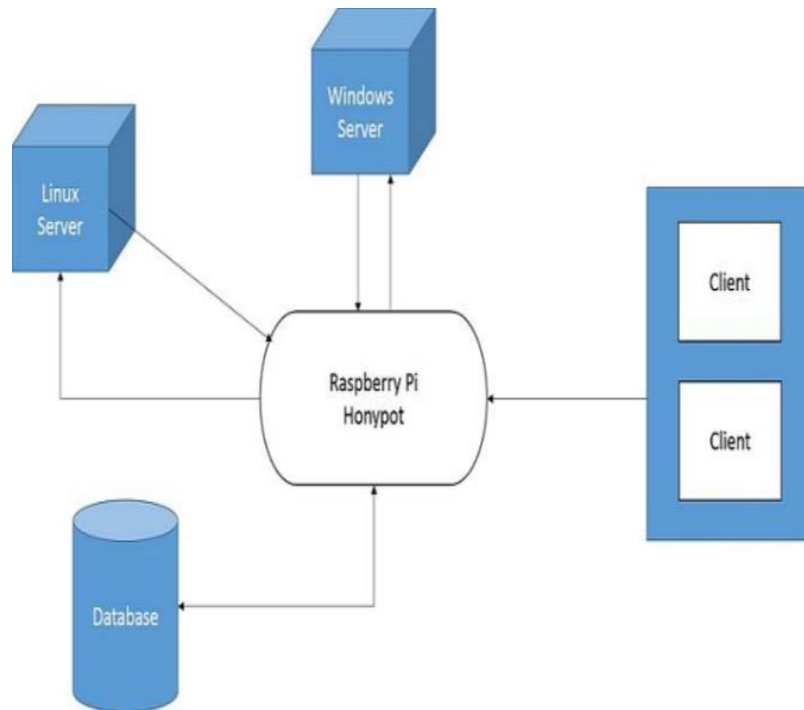


Figure 5: Raspberry Pi - HoneyPot deployment [10]

The author proposes use of raspberry pi honeypot as a part of client architecture for capturing malicious activity. Workstation on client side serves to collect the attacker's activity either records it or simply passes it on to the server side for further analysis. Server side architecture serves to analyze the received data from client end and depending upon the rules defined decides whether to generate or not generate intrusion detection warning and display it to the administrator.

Suggested design [10] only theoretically focuses on implementing Raspberry Pi HoneyPot for improving network security but lacks any practical testing and experimentation.

2.3. Design and Analysis of Real-time Network Intrusion Detection and Prevention System using Open Source Tools [12]

This paper proposes the design for providing network security to small and medium IT organisations. Keeping in view the limitations of small IT setups, paper proposes security design by making use of open source tools. Comparison of different open source tools available for intrusion detection and prevention was carried out and solutions suiting the hardware available to author were selected. The paper focuses on the use of open source tools easily compatible with Juniper devices available to author. Design layout is as shown

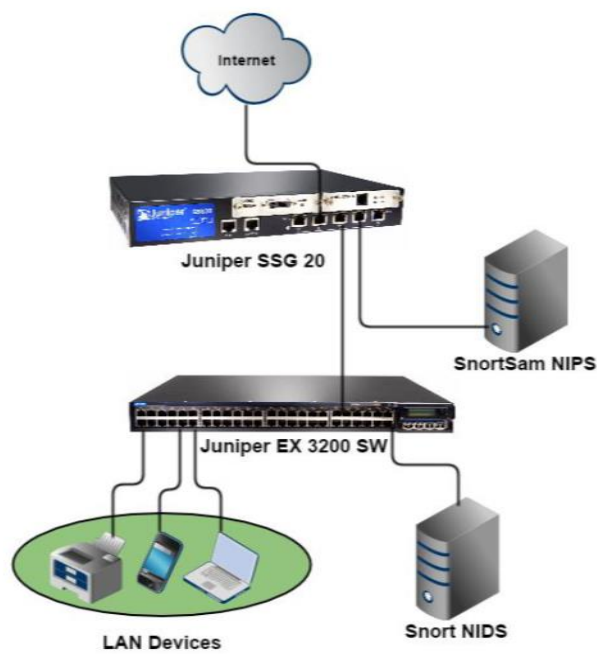


Figure 6: Network Connectivity Diagram [12]

Proposed design consists of detection, prevention and analysis components. The paper uses combination of juniper switch Ex 3200 for hosts connectivity and netscreen firewall SSG 20 at the entry point for perimeter defence. Design describes use of Snort both as IDS for detection as well as IPS for prevention of any unauthorized access. Snort IDS is a protective tool of network safety. It has been broadly used for shielding the network of IT based enterprise setups [11].

The author uses Snort IDS with built in Snort rules as NIDS for detecting intrusion

using one of the switch port in mirror mode. IPS is implemented using SnortSam agent on firewall for further preventing the intrusion into the network. SnortSam entails two parts: an agent that runs on the entry device and takes directions, and an output plugin for Snort that refers instructions created on activated procedures. [13].

Suggested design only focuses on network with static IP configuration and tests the solution against only one kind of attack i.e ICMP ping. Blocking time span for SnortSam deployed on firewall for prevention of particular IP address is set to 5 minutes only.

2.4. A firewall for SOHO networks using Raspberry Pi and Snort [14]

This paper puts forward the idea of using raspberry pi for intrusion detection for small and medium organizations. The suggested solution targets the IT setups which cannot afford expensive IDS and IPS solutions available in the market. In the proposed design, author of the paper uses raspberry pi as network monitoring terminal with open source IDS tool Snort for capturing and checking the traffic for any suspicious activity. Solution uses built in Snort signature rules for detecting intrusion into the network.

Author tests the proposed solution against ICMP ping and TCP SYN flood attacks with the corresponding snort rules installed.

- **Internet Control Message Protocol (ICMP) ping:-** This attack pursues to overcome the server's capability to answer, therefore stalling valid demands [15]. Attacker directs huge amount of ping request to the server such that server cannot handle legitimate requests anymore which results in denial of service (DoS) to client.
- **Transmission Control Protocol (TCN) SYN Flood:-** Attacker flood series of TCP SYN packets for connection to server consuming the server resources completely such that it cannot handle legal SYN requests resulting in DoS.

Suggested design includes testing against few of network attacks on static networks. The paper [14] doesn't describe the testing of proposed solution against MITM attacks.

2.5. Pi-IDS: Evaluation of Open-Source Intrusion Detection Systems on Raspberry Pi 2[16]

This paper presents assessment of open source IDS tools available on raspberry pi 2 model B. The author compares the performance of open source IDS on raspberry pi hardware. Two of the open source IDS tools that are used in this evaluation are Snort and Bro IDS solutions. Snort is the most commonly used IDS with millions of built in detection rules. Snort is supported on a quantity of hardware platforms and compute platforms [17]. Bro IDS is NIDS which passively detects for suspicious signature pattern in the captured traffic. Bro has the aptitude to perform multi-layer investigation, Behavior observing, Strategy implementation, Policy-based invasion discovery and Recording network activity [17]. Layout used for comparison is shown

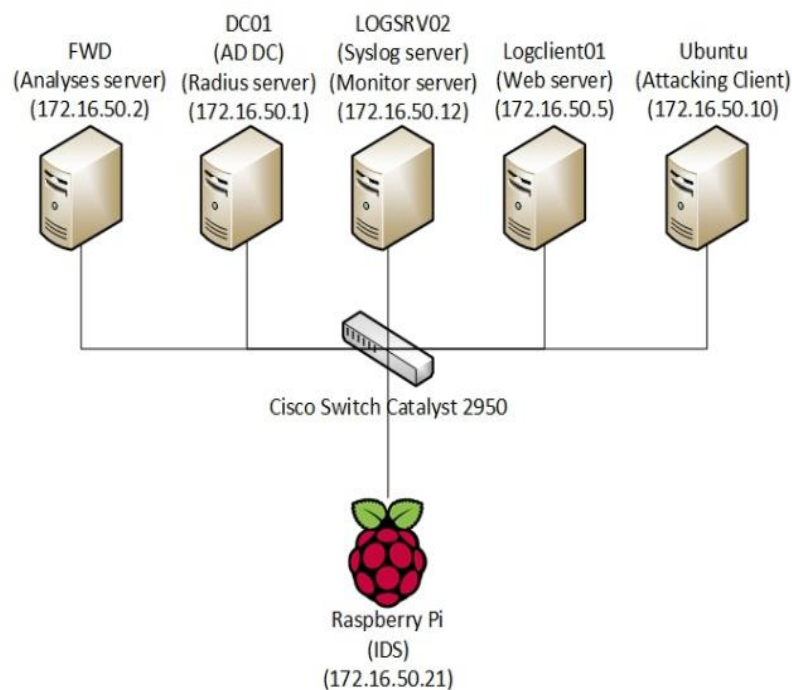


Figure 7: System Architecture for Experiment [16]

The paper compares Pi performance in terms of CPU and memory usage in terms of data collection and processing ability against attacks such as SYN flood and ARP spoofing. Between the two, performance of Snort comes better. The author of paper also recommends use of results for small organizations but lacks practical evaluation.

2.6. Conclusion

In this chapter research in the field of cyber security using Raspberry Pi hardware is reviewed. Raspberry Pi hardware has been utilized for network security for intrusion detection against various attacks. Design have been suggested by the researchers against SQL injection, ICMP and TCP flood attacks considering static LAN environments. Suggested solutions using raspberry pi are not tested against MITM attacks in the dynamically configured LAN environments where our hardware tends to learn the network parameters automatically.

CHAPTER 3: TOOLS AND METHODOLOGY

This chapter mentions hardware and software tools along with methodology used during the course of research. Combinations of different software products and hardware have been used. Brief description of the hardware and software products used is given below.

3.1. Raspberry Pi

Raspberry Pi is a card-sized minicomputer that can either work on mains or battery control [17]. It is a small size computer that was basically design for experimentation purposes. It is a low cost hardware consisting its own RAM and CPU for data processing. Due to its low cost with almost no installation and maintenance requirement, it is an easily affordable hardware for small and medium organizations. Raspberry Pi was intended for the Linux working framework, and numerous Linux circulations currently have an adaptation enhanced for the Raspberry Pi [18]. Separate debian based operating system designed for raspberry pi is Raspbian.



Figure 8: Raspberry Pi Model 2B

There are many generations of raspberry pi that have evolved over a period of time. Different versions of this hardware varies from Raspberry Pi zero to Raspberry Pi model 3B. These different versions of hardware differs it term of their hardware and specifications. Since goal of our research is to design a prototype for intrusion detection using this hardware, version of this hardware used for this purpose is raspberry pi model 2B. The generic specifications of this model [19] are.

- RAM - 1GB
- CPU - 900MHz Quad-core ARM
- HDMI interface - 1
- USB ports - 4
- Ethernet 10/100Mbps`Interface - 1

Raspberry Pi hardware possess many advantages including low power requirements, no installation, infrastructure and maintenance requirements, no noise, modularity and low cost. Due to these major advantages of this small size portable computer hardware, it was used in this research work.

3.2. Kali Linux

Kali Linux is the open source penetration testing and hacking operating system that has been used in this work. Kali Linux is one of the best open-source security bundles of an ethical hacker, comprising a set of tools separated by classifications [21]. Purpose of using this operating system is to launch and study the pattern of MITM attacks on the network. Using this methodology, packets can be captured and analyzed to develop signature pattern or selecting network features of intrusion detection.

The architecture of this ethical hacking operating system has been divided into many categories. Attacks have been divided into separate categories in Kali Linux. Some of these categories are information gathering, social engineering, forensics, stress, sniffing and spoofing tools.



Figure 9: Kali Linux

Kali Linux is developed with almost 600 tools which can be helpful towards developers and security experts in accomplishing goals in the field of information security tasks such as Penetration Testing, Security study, Computer Forensics and Reverse Engineering etc. Kali Linux was used in this research work to simulate attacks on LAN in order to identify the features that were used to detect the respective attacks.

3.3. Machine Learning

Machine learning is the application program that can be used by systems to learn from the data sets given and train them. Machine learning is the concept that makes the hardware or application to learn from past experience in order to predict the future possibility. It has been talk of modern era because in today's world main goal is to let our systems automatically predict the output on the basis of past data given using machine learning concepts. It is the best strategy utilized in the field of information investigation so as to anticipate something by formulating a few models and calculations [22].

Machine learning is the concept that uses built in algorithms to design a training model for our devices so that they can be trained to predict the outcomes. Machine learning algorithms are divided into two main categories which are

- **Supervised Learning:-** In this category of machine learning algorithms, systems are trained and tested on data sets given them as input. Data sets are given in supervised learning with labeled outcomes. On the basis of given datasets with labeled outcomes systems get trained and used for predicting the future. Examples of supervised machine learning algorithms are linear and logistic regression, naïve Bayes, random forests, decision trees and neural networks etc.
- **Unsupervised Learning:-** In this category of machine learning algorithms machines are given datasets without any labeled outputs. Basic purpose of unsupervised learning is to find hidden patterns that can be used for futuristic prediction in unlabeled input dataset. In other words we can say that learning from unlabeled dataset to differentiate the given input data [23] is unsupervised learning. Examples of unsupervised learning are K-mean clustering, hierarchal clustering and hidden markov models etc.

Machine Learning was incorporated in this research in order to identify the features from the captured packets. Machine learning was used to propose a solution that can be implemented on any network so that it would be able to extract the features automatically irrespective of any threshold and infrastructure. Different versions of supervised machine learning algorithms were available. Before choosing a particular algorithm for this research purpose, detailed comparative analysis was carried out among multiple machine learning algorithms i.e *Naïve Bayes, J4 and Logistic regression* on dataset with set of selected independent features as mentioned in *section 5.4*. On the basis of comparative analysis and output results, Logistic Regression was chosen to be used in this research to implement machine learning.

3.4. Wireshark

Wireshark is an open source network analyzer that can be used to capture and perform packet analysis. It is the software application that can be used for network troubleshooting in

case of any unwanted access into secure network. Wireshark provides GUI based interface to user for packet capture and analysis.

Wireshark is a packet capturing software that understands structure of various networking protocols. It gives user layer by layer understanding of all fields in the captured packets. Wireshark uses pcap libraries to capture network packets only supported by pcap format. Once captured user can view different fields and can perform data filtering to get the desired packet data.

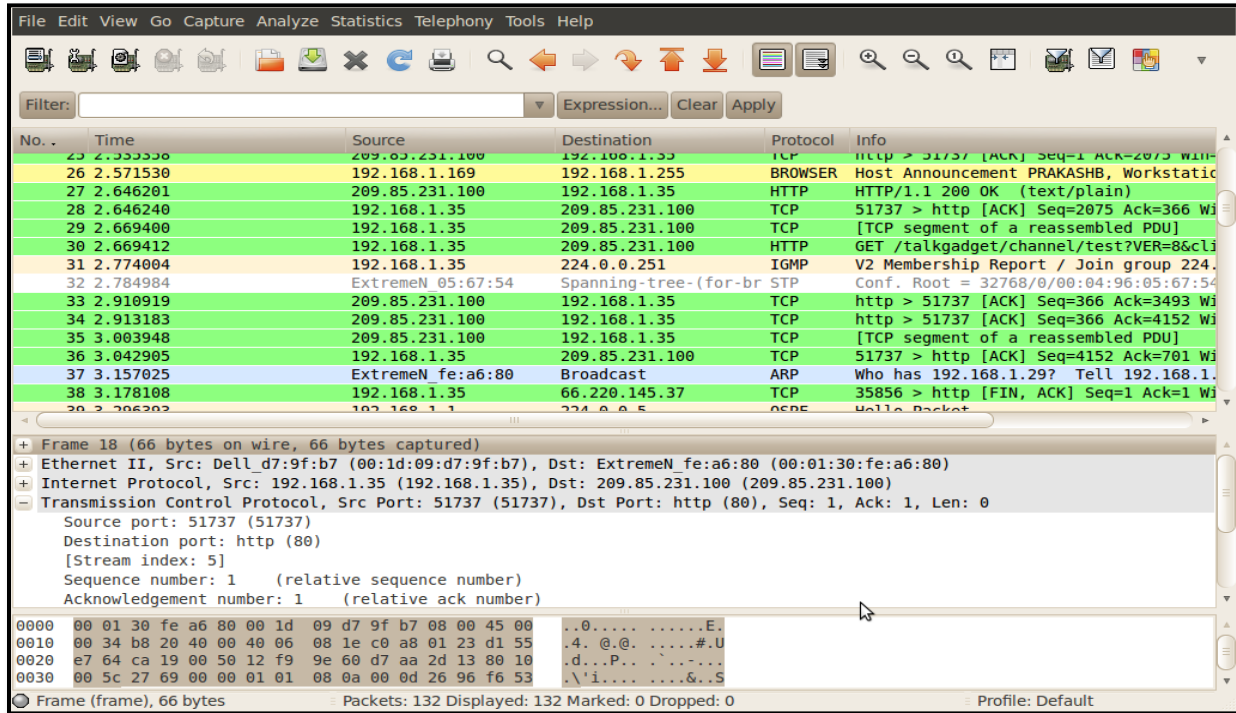


Figure 10: Wireshark GUI

Wireshark uses built in filters and color codes to perform deep packet filtering and understanding. Wireshark was used on raspberry pi operating in mirror mode to capture network packets for in depth analysis to identify an event as intrusion.

3.5. Weka

Weka is the software application that is used for training/testing in our proposed design for detecting intrusions using machine learning. It is not a single package, but somewhat a bunch of system tools grouped together by a joint user interface [24]. Weka is a workbench for machine

learning that is anticipated to help in the use of machine learning techniques to a variety of real world issues.

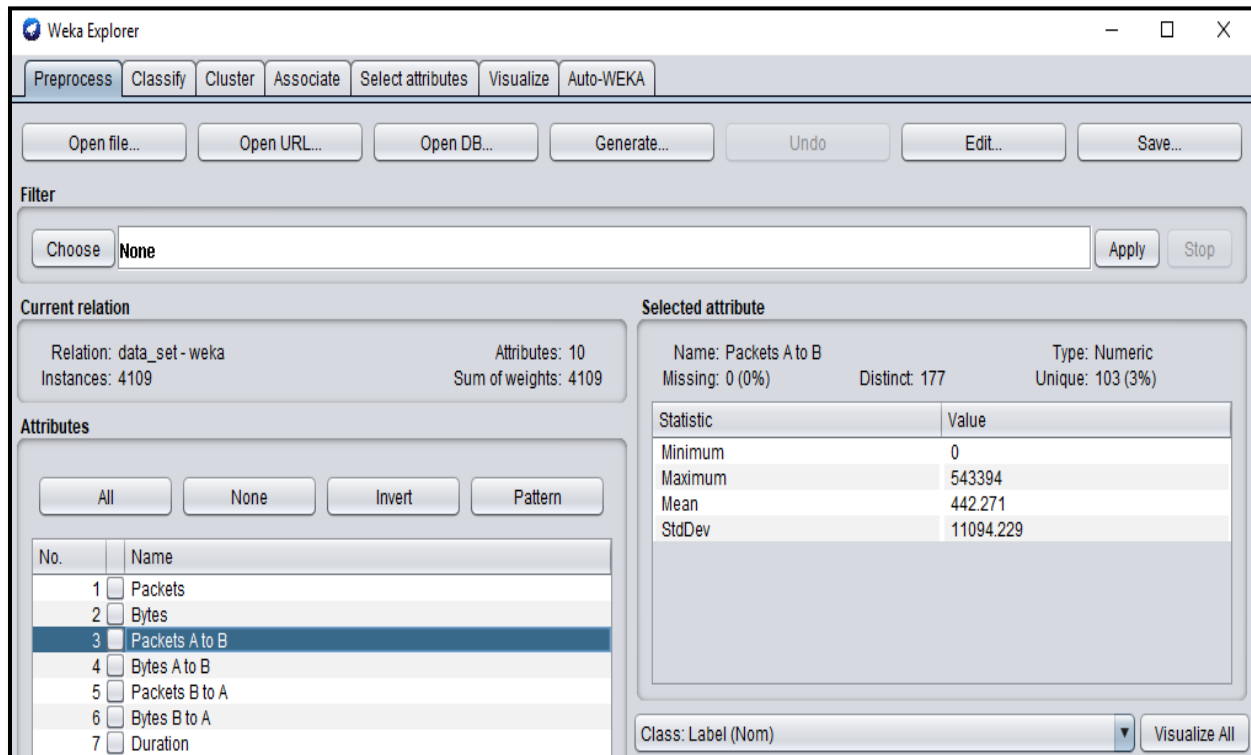


Figure 11: Weka user interface

Weka enables client to assemble applications utilizing both their historical information of data and the learning capacity required to fabricate intellectual information structures utilizing artificial intelligence and deep learning.

Weka consists of almost all types of supervised and unsupervised machine learning algorithms. Weka was used in this work in order to perform data pre-processing using logistic regression to select the combination of features that can be used to get the best possible characteristics of detection system using machine.

3.6. Spyder IDE

Spyder IDE is the python development platform used by the coders to develop code using python. It also provides a code development and testing environment from where coder can check the errors and correct them. Python is simple and easy to acquire programming language.

It has proficient high-level data structures and an easy but effective method to object-oriented programming [25]. Python has many built in library functions that can be used to embed machine learning algorithms. Spyder IDE provides python based developing environment that is compatible with modern day computing platforms such as Windows and Linux etc.

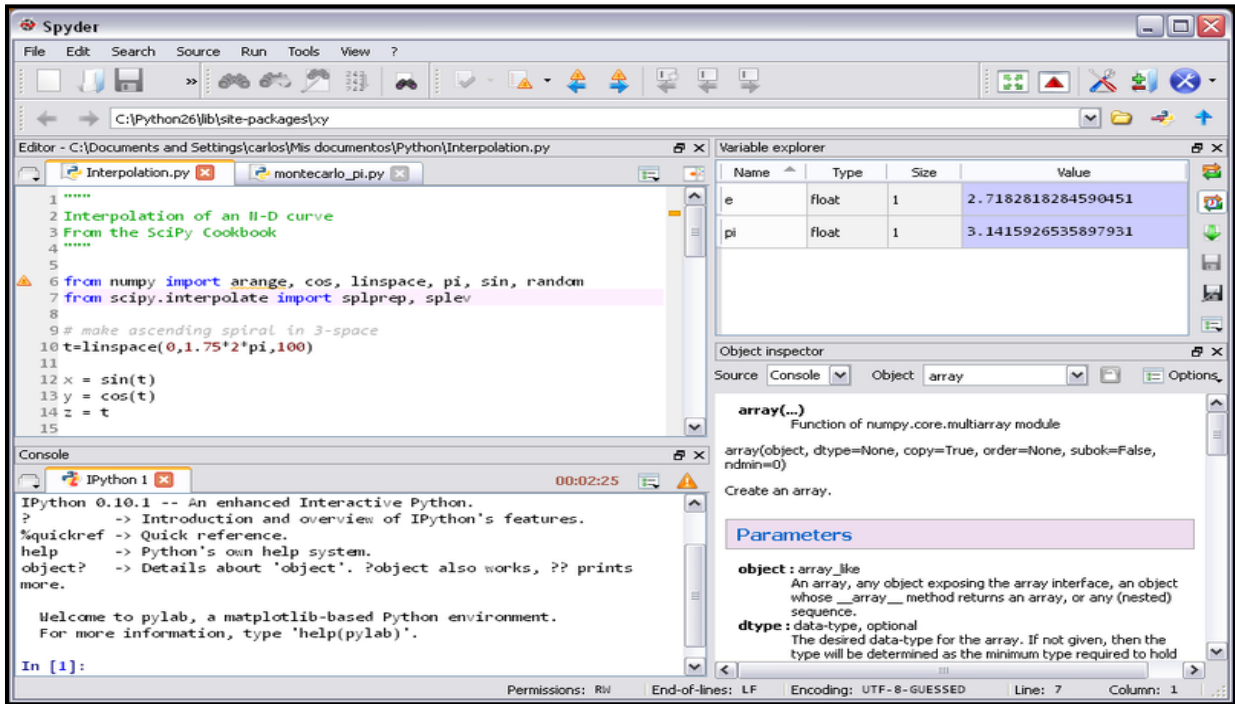


Figure 12: Spyder IDE interface

This research is about developing intrusion detection prototype using raspberry pi computers against MITM attacks. This software has been used by the author in developing python code using logistic regression before implementing it on raspberry pi hardware. Outcome of both weka and spyder IDE against the given dataset has been compared and used as reference its implementation on raspberry pi.

CHAPTER 4: RESEARCH METHODOLOGY

4.1. Research Process

This section contains the mapping of our research process according to our problem statement and proposed solution. Depending upon the objective of this research work, our research methodology consists of (i) Data set generation (ii) Feature Selection (iii) Feature Evaluation (iv) Raspberry Pi Implementation and (v) Design Evaluation as shown

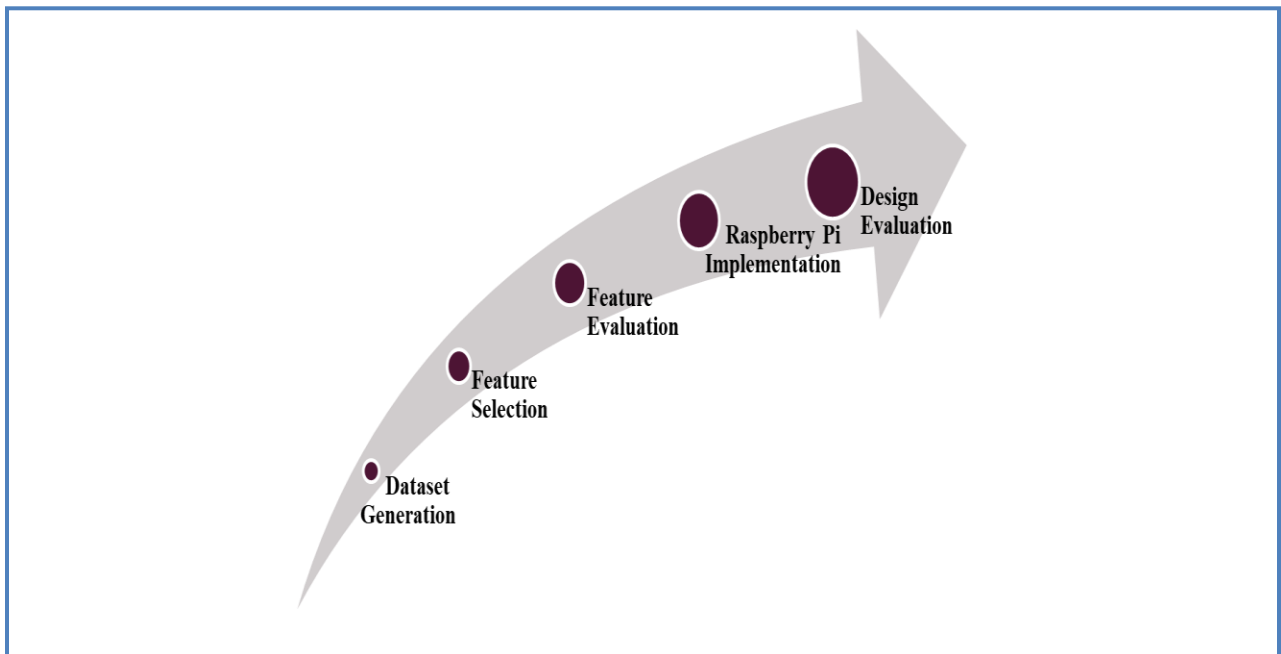


Figure 13: Research Methodology

4.1.1 Dataset Generation

The first step in our research was dataset generation. In order to implement machine learning in any system, the most important thing is the dataset which is required for train and test purposes. Dataset is the collection of data on which machine learning algorithms are trained so that they will be able to predict the outcome on the basis of train data. Many datasets are available for use in research work in the field of cyber security

e.g UGR'16 [26] and NETRESEC [27] etc. No dataset with respect to Arp spoofing, Mac flooding and Port stealing was available, therefore dataset was generated in the first step.

4.1.2 Feature Selection

Selection of features is the important part of machine learning because performance of system depends upon the selected features. Feature collection is an essential practice for data mining [28]. It helps us to identify the parameters or variables that will be helpful in predicting the outcome. In the second step, captured datasets were labeled and analyzed to select the subset of independent features having high impact on predicting the outcome. Different feature selection techniques are available which were used in this step. Feature selection methods provide us a great comprehension of the information in AI or example acknowledgment applications [29].

4.1.3 Feature Evaluation

After the selection of features done in previous step, next step was to assess the impact of features selected on predicting the output. The requirement was to evaluate system performance using selected features before implementing them on raspberry pi. System performance in predicting the output on the basis of train dataset is evaluated using different parameters i.e Confusion Matrix, ROC curve, Precision and Recall. Evaluation of selected features was carried out on the basis of results generated by Weka using logistic regression in terms of machine learning metrics [30]. Similar evaluation in terms of metrics [30] was carried out by means of python based code using spyder.

4.1.4 Raspberry Pi Implementation

Evaluating the outcome of Weka and python using machine learning in terms of evaluation metrics in the previous step resulted in fine tuning of features and training of system to differentiate between benign and suspicious traffic. After performing offline training and testing, next step was to implement it on raspberry pi for online testing and results. Code was written in python for raspberry pi to extract the features as selected in

the last step from incoming network traffic and perform logistic regression on captured values to log the packet as malicious.

4.1.5 Design Evaluation

In the last step of research, analysis and evaluation of proposed security design for small and medium networks was carried out. Design evaluation was carried out by comparing the results of offline and online training and testing in terms of confusion matrix, ROC curve, precision and recall. Performance of raspberry pi hardware was also evaluated in terms of CPU and memory utilization.

CHAPTER 5: SYSTEM DESIGN AND INTEGRATION

This chapter discusses system design, experiments and its integration into small and medium LAN to detect intrusions.

5.1. System Architecture

The computer network in our lab consists of 10 workstations sharing a common LAN. Connectivity between these workstations was provided through cisco switch i.e 3isys 5500. This switch was chosen due to fact that it is managed Ethernet switch and is easy to use and configure. It supports comprehensive QoS, enhanced VLAN functions [31] and can easily be afforded by organizations as compared to other network switches. Switch was configured by making the workstations to obtain IP address automatically via Dynamic Host Configuration Protocol (DHCP). Layer 2 VLAN was configured on switch to segregate the workstations into single shared domain.

Raspberry Pi was configured with the code of detecting the intrusion on the basis of selected features. Raspberry pi was connected to switch port operating in promiscuous mode to monitor all incoming and outgoing traffic from LAN. As a result, raspberry pi was able to view all type of traffic entering and leaving the network. The port set in promiscuous mode acts as gateway and communicates with all VLAN ports [32]. Wireshark was installed on raspberry pi to capture incoming packets. Workstation used for analysis and central logging was also connected to same switch to work as central repository for log collection as generated by raspberry pi computer.

Complete network workstations were provided internet facility through Ethernet cable connected to DSL router. Attacker machine was also part of this network by making use of Kali Linux installed using VM ware on it. Attacker machine was used to simulate man in middle attacks on LAN to generate datasets and to check the capabilities of solution in detecting any suspicious activity or intrusion. System architecture is shown

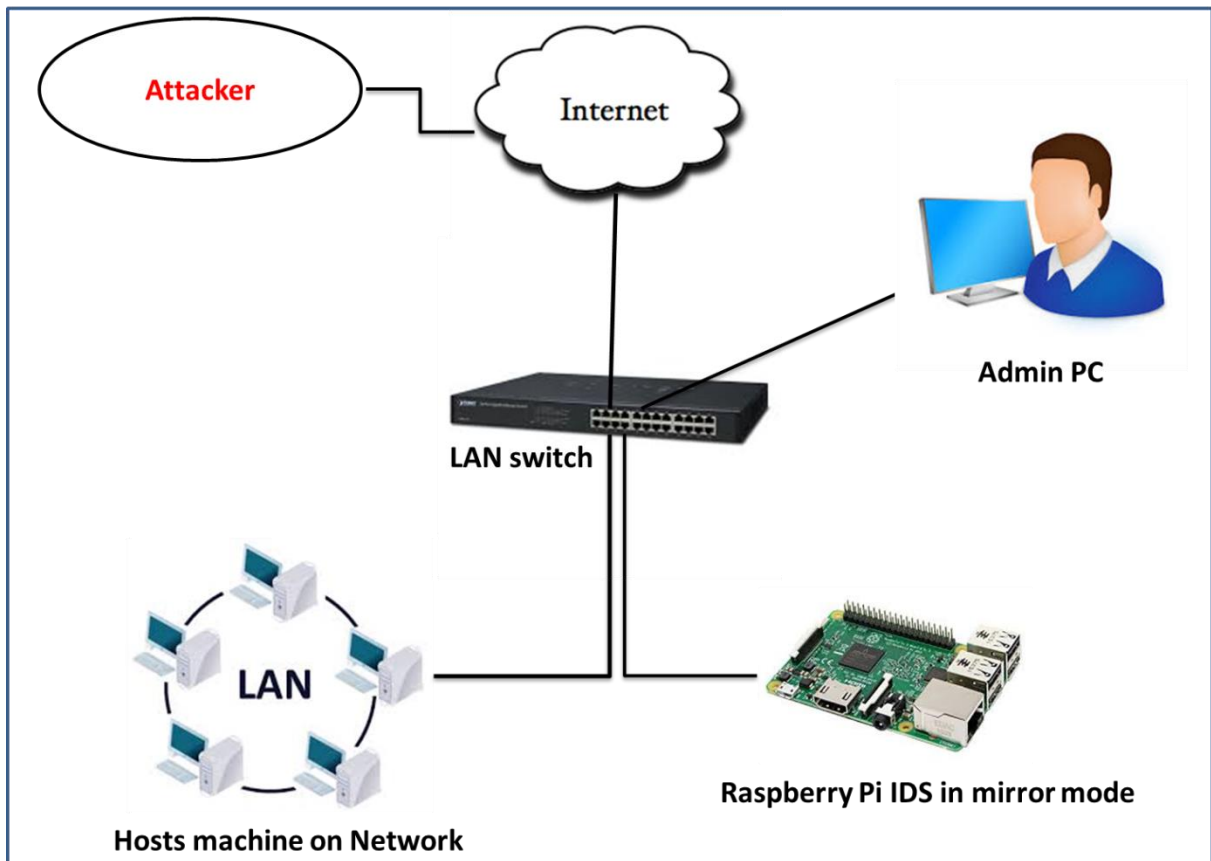


Figure 14: System Design

Our proposed solution is the development of security solution for small and medium enterprise networks using raspberry pi that can ensure the data security. It should be able to not only detect intrusion using machine learning but also ensure logging of such event. Detail of process performed in order to build up required product is given as follows.

5.2. Data set Generation

Since we have used concept of machine learning in this research, most valuable thing for any of machine learning algorithm is input data set. Data set is the collection of historical data examples used to train machine learning algorithms before testing them. Data sets are the

important part in the dynamics of machine learning. Machine learning algorithms use data sets to train themselves to predict the future outcome.

Many datasets related to various fields of information technology are publically available over the internet for use by researchers and developers using machine learning. In the similar way reference repository of public data sets related to cyber security is also available online for usage in information security research [33]. These datasets have been divided into different categories depending upon the data samples they have. Few of these categories are namely malware, systems, threat feeds and third party networks. These datasets were collected in different working scenarios by their developers either in form of some competition or task. These data sets have labeled data samples for both normal and malicious traffic collected over span of hours or weeks during their research. One such example is KDD data set [34] containing standard set of data including extensive variation of intrusions.

In our research, we had focused on infrastructure based MITM attacks such as ARP spoofing, Port stealing and MAC flooding etc. Different repositories having datasets related to cyber security attacks are available for public use. Data sets related to MITM attacks were not found for use. Keeping this in view, a new data set related to LAN based MITM attacks was generated for use during the course of this research.

5.2.1 Lab setup for Data Set Generation

In order to generate data set, lab setup was used having 10 x workstations. These work stations were connected via managed Ethernet switch. One of the workstations was tasked to monitor and collect data samples by connecting it onto the switch port configured in promiscuous mode. Software application used to collect data on this workstation was Wireshark. Wireshark is a magnificent tool with examination and estimation of system traffic observing [35].

In order to capture related to samples of malicious traffic related to MITM attacks, there was need to simulate these attacks onto the network that was used for data set generation. For this purpose Kali Linux [21] used. Kali Linux is ethical hacking and penetration testing operating system containing almost 600 tools which can be helpful towards developers and security experts in accomplishing goals in the field of

information security tasks. Lab network was also provided with the internet connectivity to setup network that is normally used in IT organizations.

Data sets were generated for both benign and malicious traffic by monitoring and capturing the traffic using wireshark over duration of 2 days and time span of 2 hours. First day data contained only normal traffic and second day data samples were the combination of both benign and malicious packets. Lab setup and specifications related to workstations used are as shown in figure and table respectively.

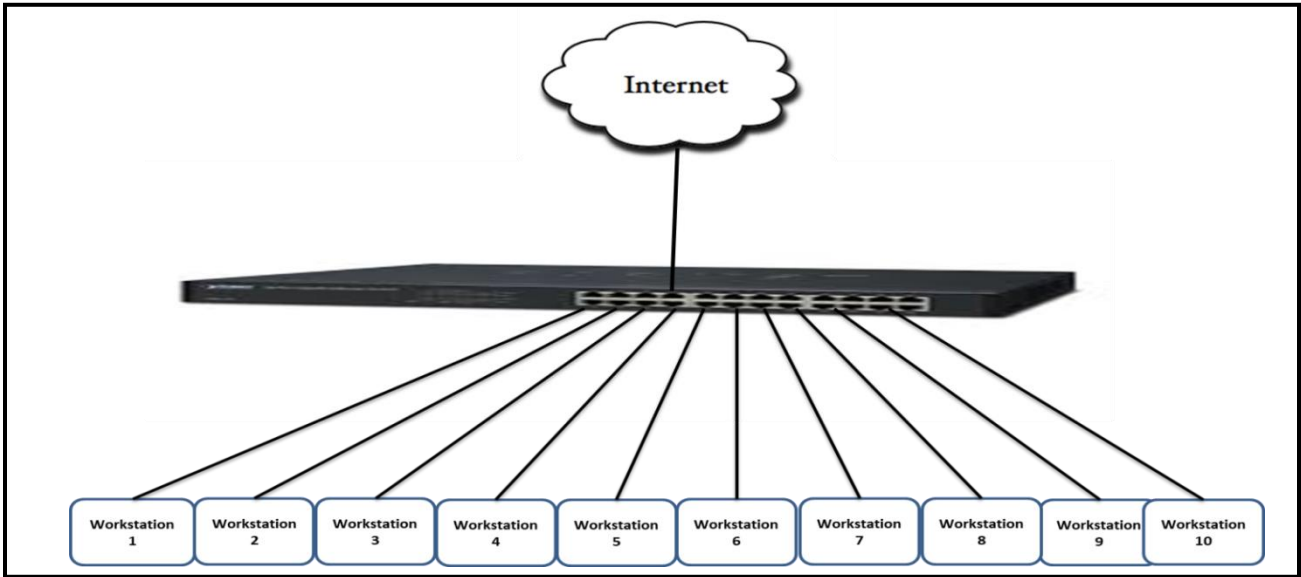


Figure 15: Lab setup for Data set

Table 1: System Hardware Configurations

Ser	Type	OS	RAM	IP
1.	Core i 5	Window 7	4 GB	172.17.77.49
2.	Core i 5	Window 7	4 GB	172.17.77.51
3.	Core i 3	Window 7	4 GB	172.17.77.55
4.	Dual Core	Window 10	2 GB	172.17.77.73
5.	Dual Core	Window 10	2 GB	172.17.77.74
6.	Dual Core	Window 10	2 GB	172.17.77.76
7.	Dual Core	Window 10	2 GB	172.17.77.152
8.	Dual Core	Window 10	2 GB	172.17.77.153
9.	Dual Core	Window 10	2 GB	172.17.77.154
10.	Attacker Machine	Kali Linux	4 GB	172.17.77.56

Table 1 shows workstations along with their processor type, Operating system, RAM and IP address assigned through DHCP running on switch. Raw data files for normal traffic were collected by Wireshark during the course of experiment. Captured pcap files contained variety of browsing and file sharing traffic including TCP, FTP and HTTP protocols. Sample view of captured pcap file is shown.

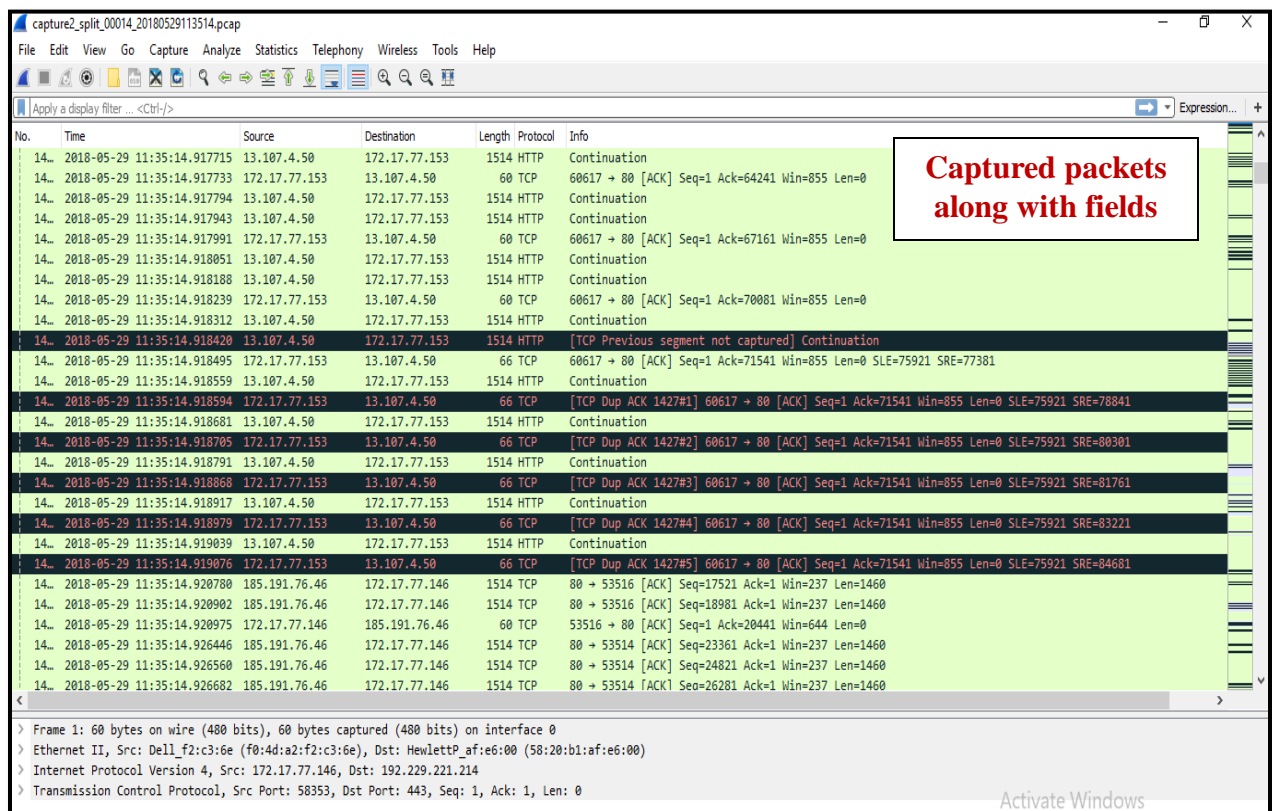


Figure 16: Sample Wireshark Pcap

Raw data collected for normal traffic using wireshark in pcap format as shown in figure 16 indicates source address, destination address, payload length, protocol and time stamp on which packet was collected. 10 GB of data files containing normal traffic was collected over the time span of 2 hours from workstations used in lab.

For the generation of malicious traffic, kali linux was installed as virtual machine on one of the workstation used in lab.

5.2.2 ARP Spoofing

ARP spoofing is the type of MITM attack in which attacker poisons the mac table of victim by sending fake gratuitous ARP replies indicating herself as a gateway. In kali linux it was launched using command line utility.

arp spoof -i [Interface] -t [Router ip address] [Target ip address] [36]

Network interface name is the interface through which attack is launched, router IP is the gateway IP and victim IP indicates IP address of target.

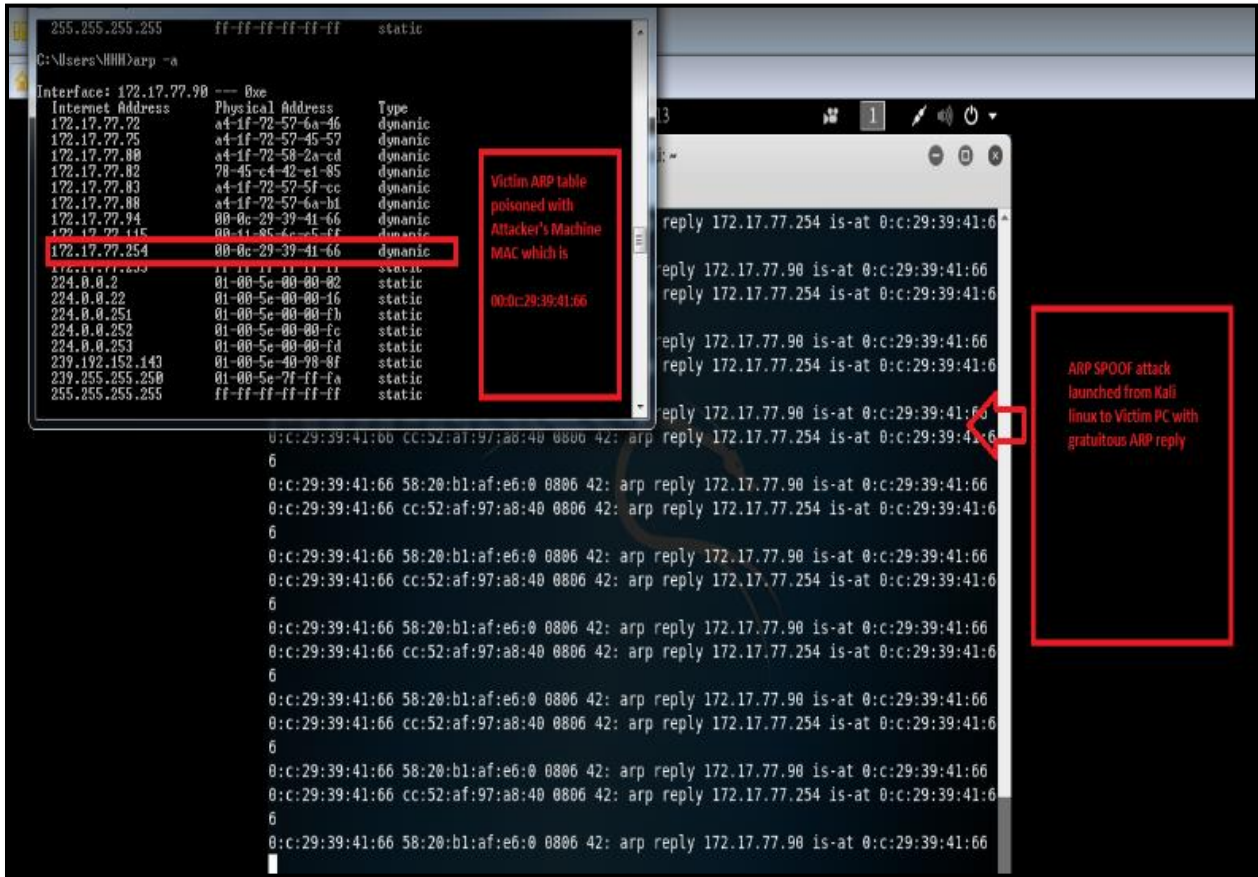


Figure 17: ARP spoof Attack

Figure 17 indicates the fake ARP replies generated to spoof the communications between the gateway and victim machine. ARP spoof attack was launched at different intervals during capture.

5.2.3 Port Stealing Attack

This technique is worthwhile to snort in a layer 2 atmosphere when ARP poisoning is not in effect [36]. It is the form of attack which attacks CAM table of layer 2 switch by targeting the mapping between MAC address and its associated port number. In kali linux it was launched using ettercap facility. It is basically a suite of tools to simplify MITM attacks [37].

Graphical interface of Ettercap was launched in kali linux using “*ettercap -G*” command. After launch of ettercap, host scan was done on network to see the active hosts on network, port steal attack was launched selecting port stealing option from MITM tab as shown

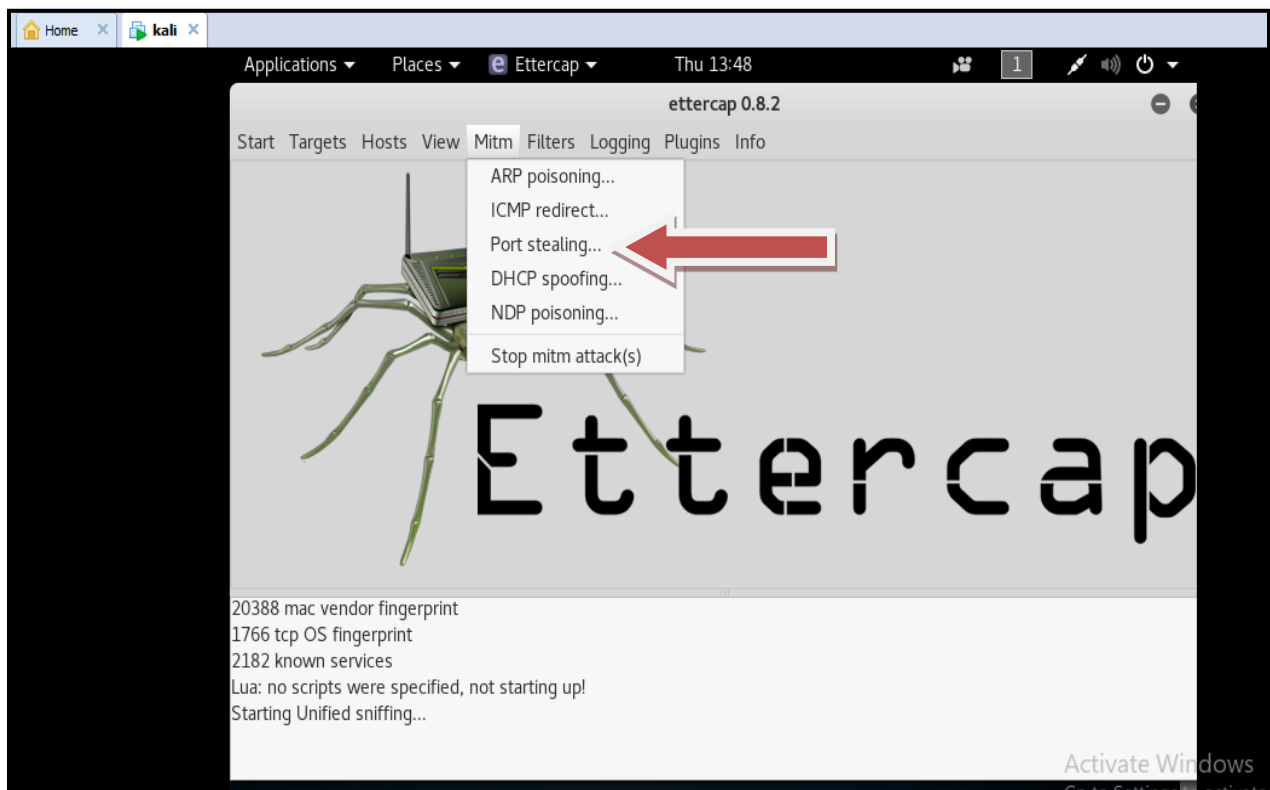


Figure 18: Port Steal Attack using ettercap

Before the launch of attack, CAM table of switch was checked mentioning the MAC addresses with their associated switch port. After the attack was launched, mapping between MAC addresses and their ports was changed with the port of attacker’s machine as shown

```

192.168.10.1 - PuTTY
Vlan Mac Address          Type    Creator  Ports
-----
100 00-0c-29-ec-ef-01       DYNAMIC Hardware Ethernet1/0/2
100 00-25-64-80-3b-f4       DYNAMIC Hardware Ethernet1/0/5
100 2c-27-d7-cd-56-f7       DYNAMIC Hardware Ethernet1/0/2
100 b8-27-eb-aa-71-bf       DYNAMIC Hardware Ethernet1/0/3
100 cc-f5-38-03-13-bf       STATIC  System   CPU
switch(config)#show mac-address-table
Read mac address table....
Vlan Mac Address          Type    Creator  Ports
-----
100 00-0c-29-ec-ef-01       DYNAMIC Hardware Ethernet1/0/2
100 00-25-64-80-3b-f4       DYNAMIC Hardware Ethernet1/0/5
100 2c-27-d7-cd-56-f7       DYNAMIC Hardware Ethernet1/0/2
100 b8-27-eb-aa-71-bf       DYNAMIC Hardware Ethernet1/0/3
100 cc-f5-38-03-13-bf       STATIC  System   CPU
switch(config)#show mac-address-table
Read mac address table....
Vlan Mac Address          Type    Creator  Ports
-----
100 00-0c-29-ec-ef-01       DYNAMIC Hardware Ethernet1/0/2
100 00-25-64-80-3b-f4       DYNAMIC Hardware Ethernet1/0/5
100 2c-27-d7-cd-56-f7       DYNAMIC Hardware Ethernet1/0/2
100 b8-27-eb-aa-71-bf       DYNAMIC Hardware Ethernet1/0/2
100 cc-f5-38-03-13-bf       STATIC  System   CPU
switch(config)#show mac-address-table
Read mac address table....
Vlan Mac Address          Type    Creator  Ports
-----
100 00-0c-29-ec-ef-01       DYNAMIC Hardware Ethernet1/0/2
100 00-25-64-80-3b-f4       DYNAMIC Hardware Ethernet1/0/2
100 2c-27-d7-cd-56-f7       DYNAMIC Hardware Ethernet1/0/2
100 b8-27-eb-aa-71-bf       DYNAMIC Hardware Ethernet1/0/2
100 cc-f5-38-03-13-bf       STATIC  System   CPU
switch(config)#show mac-address-table

```

Switch CAM table before attack

Switch CAM table after attack launch

Figure 19: Switch CAM table

Port Steal attack was launched at different intervals during capture.

5.2.4 MAC Flooding Attack

Every switch has CAM table with limited entries depending upon the hardware. In this type of attack, attacker floods fake ARP packets each with different source and destination mac addresses to fill the CAM table so that switch starts behaving like a hub for any incoming packet afterwards. Attack was launched using Macof tool in kali linux.

```

192.168.10.1 - PuTTY
100 00-6e-9b-20-6c-f4 DYNAMIC Hardware Ethernet1/0/3
100 00-6f-68-6c-d0-be DYNAMIC Hardware Ethernet1/0/3
100 00-72-2c-10-57-54 DYNAMIC Hardware Ethernet1/0/3
100 00-74-bd-76-82-c3 DYNAMIC Hardware Ethernet1/0/3
100 00-78-06-0c-5a-54 DYNAMIC Hardware Ethernet1/0/3
100 00-78-c5-48-01-0d DYNAMIC Hardware Ethernet1/0/3
100 00-79-87-67-e3-d3 DYNAMIC Hardware Ethernet1/0/3
100 00-7b-61-77-e4-5c DYNAMIC Hardware Ethernet1/0/3
100 00-7c-d5-0f-3b-4f DYNAMIC Hardware Ethernet1/0/3
100 00-7d-30-6d-dc-58 DYNAMIC Hardware Ethernet1/0/3
100 00-7e-2f-2b-da-39 DYNAMIC Hardware Ethernet1/0/3
100 00-7f-7e-77-49-65 DYNAMIC Hardware Ethernet1/0/3
100 00-80-69-53-4d-f7 DYNAMIC Hardware Ethernet1/0/3
100 00-81-11-4e-f5-d1 DYNAMIC Hardware Ethernet1/0/3
100 00-87-14-68-55-a4 DYNAMIC Hardware Ethernet1/0/3
100 00-88-66-15-9e-96 DYNAMIC Hardware Ethernet1/0/3
100 00-88-fa-23-24-67 DYNAMIC Hardware Ethernet1/0/3
100 00-8a-ca-03-62-9c DYNAMIC Hardware Ethernet1/0/3
100 00-8d-b2-1c-c2-48 DYNAMIC Hardware Ethernet1/0/3
100 00-8e-5e-0d-23-40 DYNAMIC Hardware Ethernet1/0/3
100 00-90-80-56-af-42 DYNAMIC Hardware Ethernet1/0/3
100 00-90-e2-1e-0a-8a DYNAMIC Hardware Ethernet1/0/3
100 00-94-0b-67-78-6e DYNAMIC Hardware Ethernet1/0/3
100 00-95-60-11-87-54 DYNAMIC Hardware Ethernet1/0/3
100 00-96-c6-54-a5-d2 DYNAMIC Hardware Ethernet1/0/3
100 00-9b-da-3f-90-2f DYNAMIC Hardware Ethernet1/0/3
100 00-9f-1c-08-2c-1d DYNAMIC Hardware Ethernet1/0/3
100 00-a1-5c-79-24-47 DYNAMIC Hardware Ethernet1/0/3
switch(config)#show mac-address-table count
Compute the number of mac address...
Max entries can be created in the largest capacity card:
Total Filter Entry Number is: 16384
Static Filter Entry Number is: 16384
Unicast Filter Entry Number is: 16384
Multicast Filter Entry Number is: 1024

Current entries have been created in the system:
Total Filter Entry Number is: 16384
Individual Filter Entry Number is: 16384
Static Filter Entry Number is: 1
Dynamic Filter Entry Number is: 16383
Multicast(Insert) Filter Entry Number is: 0
Multicast(Wait) Filter Entry Number is: 0
switch(config)#

```

Switch CAM table flooded

Switch CAM table limit exceeds

Figure 20: Macflooding Attack

Macof is a hacker’s utility that can overflow a layer 2 LAN with fake irregular hardware addresses [38]. Thousands of packets with fake mac addresses are sent in few seconds which completely utilizes the handling capacity of table. Mac flooding was launched using kali linux terminal through “*Macof -I [interface]*” command [38] where interface is the port through with attack was launched on switch. Ethernet switch used had capacity to handle only 16384 entries. Once this limit was reached switch started to behave like hub as shown.

Impact of attack on CAM table is shown in figure 20. Attack was launched on different intervals.

5 GB of data files containing malicious traffic at different intervals was collected over the time span of 2 hours from workstations used in lab. Raw pcap files that were collected using wireshark for both normal and malicious traffic had entries indicating source IP, destination IP, protocol type, payload and timestamps. Since these captured fields are network dependent, we needed dataset with network independent features due

to the fact that machine learning algorithms always takes independent features as input to predict the dependent variable as output.

In order to extract independent features, deep packet analysis of pcap files for both normal and malicious files using various wireshark display filters was carried out. Display filters enable you to focus on the packets you are keen on while concealing the uninteresting ones [39]. Packet analysis of packet metadata and use of filters on both types of traffic resulted in generation of data set with independent nine features as shown in table 2.

Table 2: Features with description

Ser	Feature	Description
1.	Packets	Complete number of packets during session
2.	Bytes	Complete number of bytes during session
3.	Packets A to B	Packets sent by source address to destination address
4.	Bytes A to B	Bytes sent by source address to destination address
5.	Packets B to A	Packets received by source Mac
6.	Bytes B to A	Bytes received by source Mac
7.	Bits/s A to B	Bits per second from source Mac to destination Mac
8.	Bits/s B to A	Bits per second received by source Mac
9.	Duration	Communication duration in seconds

Extraction of features as shown in table 2 resulted in labeled data set containing traces both normal and attack traffic. Dataset used for train and test purpose consisted of 4110 instances with 9 features having each type of traffic identified by its respective label. Dataset instances were then saved as comma separated values (CSV) in excel as shown in figure 21.

A	B	C	D	E	F	G	H	I	J
Packets	Bytes	Packets A to B	Bytes A to B	Packets B to A	Bytes B to A	Duration	Bits/s A to B	Bits/s B to A	Label
2228	146352	2228	146352	0	0	3169	369	0	Normal
20	3874	20	3874	0	0	28	1090	0	Normal
14	868	8	480	6	388	1	3820	3087	Normal
42	10100	28	8908	14	1192	0	2269121	303636	Normal
28	5646	14	4454	14	1192	0	1176129	314761	Normal
28	5646	14	4454	14	1192	0	640909	171523	Normal
28	5646	14	4454	14	1192	0	220891	59116	Normal
28	5646	14	4454	14	1192	0	142151	38043	Normal
28	5646	14	4454	14	1192	0	96842	25917	Normal
40	2456	28	1680	12	776	2	6657	3075	Normal
8	580	0	0	8	580	0	0	0	Normal
4	240	4	240	0	0	0	0	0	Normal
552	62260	552	62260	0	0	1416	352	0	Normal
240	51840	240	51840	0	0	1083	383	0	Normal
112	16576	112	16576	0	0	917	145	0	Normal
8	688	8	688	0	0	0	0	0	Normal
32	2848	32	2848	0	0	117	195	0	Normal
32	2208	32	2208	0	0	117	151	0	Normal
4	240	4	240	0	0	0	0	0	Normal
8	688	8	688	0	0	0	0	0	Normal
8	688	8	688	0	0	0	0	0	Normal
8	480	8	480	0	0	0	25982	0	Normal
8	720	8	720	0	0	0	38976	0	Normal
96	13656	96	13656	0	0	2944	37	0	Normal
184	20254	82	9040	102	11214	126	576	715	Normal
548	58118	288	33464	260	24654	1324	202	149	Normal
28	2068	0	0	28	2068	54	0	304	Normal
4	1524	0	0	4	1524	0	0	0	Normal
1214	115252	1214	115252	0	0	3126	295	0	Normal
196	21038	88	9436	108	11602	62	1217	1496	Normal
374	36030	192	19832	182	16198	61	2610	2131	Normal

Figure 21: Data set CSV file

Labels were used in order to differentiate between benign and malicious packets. Since we were using supervised version of machine learning algorithms which requires labeled data as input for training and testing purpose. Multiclass labels were used to label data set which are shown in table 3.

Table 3: Data Packets with Label

Ser	Packet	Label
1.	Normal	0
2.	Arp spoof	1
3.	Port Steal	2
4.	Mac Flood	3

5.3. Feature Selection

After the generation of dataset with extracted features, feature selection was performed to select the subset of features that resulted in best possible performance by proposed IDS. Different techniques for feature selection are available in the field of machine learning. Feature selection techniques help you in your main goal to make a precise prescient model [40]. In machine learning feature selection techniques falls under two types which are

Filter Method: - In this type of method, features are assigned with score and are further ranked with respect to their scores. Scores are given depending upon the impact of each independent feature on predicting the dependent variable. Depending upon their ranking, feature is either kept or removed. Filter techniques for feature selection used in this research coefficient ranking, gain ratio feature evaluator and information gain ranking methods.



Figure 22: Filter Method for Selection of Features [41]

Wrapper Method:- In this method, combination of different features is selected, evaluated and compared with respect to model accuracy in predicting the output. It also uses backward or forward selection method to either remove or add features. Wrapper method used in this research was Best effort method.

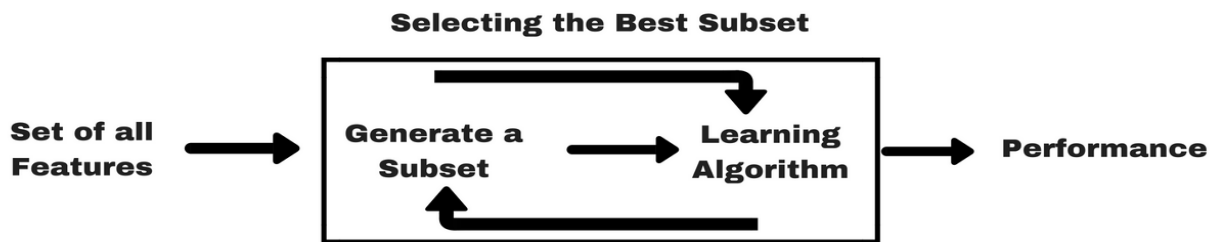


Figure 23: Wrapper Method for Feature Selection [41]

Software application used for performing feature selection was Weka. Weka has built in attribute selection tool that uses feature collection methods to select the finest subset of features.

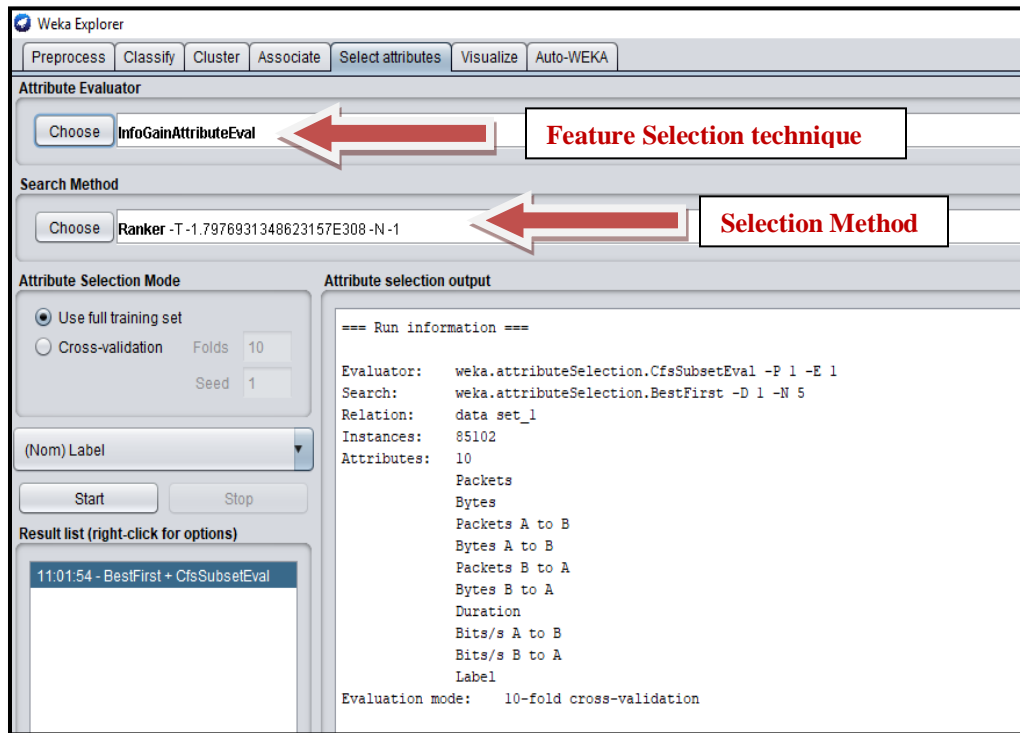


Figure 24: Weka GUI

Four different methods for feature selection were used which were then compared to select the best combination of features as shown in figures 25 – 28

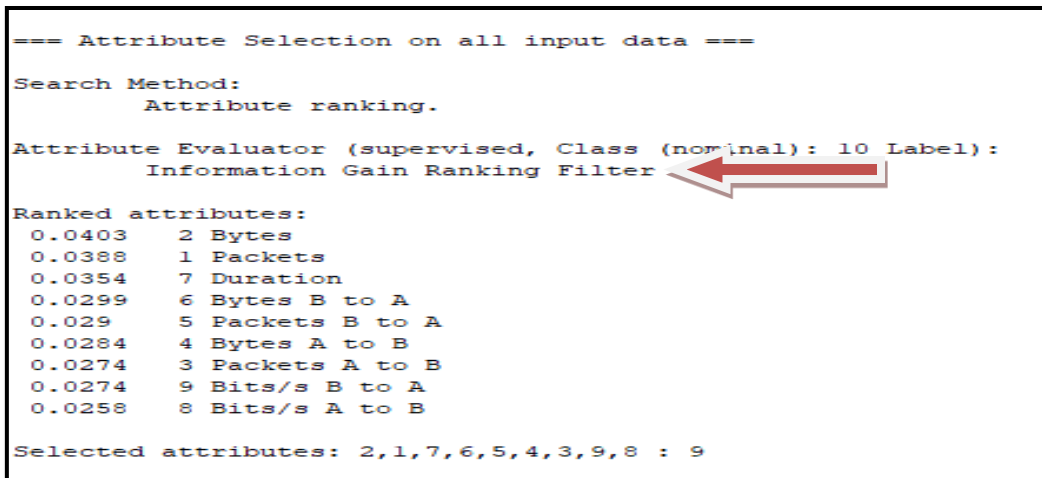


Figure 25: Information Gain Ranking Filter Output

Output of feature selection using information gain filter for supervised learning on the whole dataset is shown in figure 25. Information gain evaluates the worth of the feature by calculating information weight of each feature, considering the class features [42]. Features ranked higher in predicting the labeled output out of nine extracted features were Bytes, Packets and session duration.

```
Search Method:
  Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 10 Label):
  Correlation Ranking Filter
Ranked attributes:
0.5553   7 Duration
0.1044   9 Bits/s B to A
0.0627   8 Bits/s A to B
0.0578   5 Packets B to A
0.0562   1 Packets
0.0553   3 Packets A to B
0.0544   2 Bytes
0.0528   4 Bytes A to B
0.0482   6 Bytes B to A

Selected attributes: 7,9,8,5,1,3,2,4,6 : 9
```

Figure 26: Correlation Ranking Filter

Output of feature selection using correlation ranking filter for supervised learning on the whole dataset is shown in figure 26. This filter estimates the value of an attribute by calculating the connection between it and the class. Features ranked higher in predicting the labeled output out of nine extracted features were session duration, number of bits per second exchanged between source and destination mac addresses.

```
Search Method:
  Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 10 Label):
  Gain Ratio feature evaluator

Ranked attributes:
0.7001  7 Duration
0.6498  2 Bytes
0.6333  9 Bits/s B to A
0.6256  8 Bits/s A to B
0.0369  1 Packets
0.0195  6 Bytes B to A
0.0189  5 Packets B to A
0.0185  4 Bytes A to B
0.0179  3 Packets A to B

Selected attributes: 7,2,9,8,1,6,5,4,3 : 9
```

Figure 27: Gain Ratio Feature Evaluator

Output of feature selection using gain ratio feature evaluator for supervised learning on the whole dataset is shown in figure 27. Gain Ratio increases the information achievement by taking fundamental information from every attribute [43]. Features ranked higher were session duration, bytes and number of bits per second received by source mac address.

```
=== Attribute Selection on all input data ===

Search Method:
  Best first.
  Start set: no attributes
  Search direction: forward
  Stale search after 5 node expansions
  Total number of subsets evaluated: 44
  Merit of best subset found: 0.889

Attribute Subset Evaluator (supervised, Class (nominal): 10 Label):
  CFS Subset Evaluator
  Including locally predictive attributes

Selected attributes: 2,7,8 : 3
  Bytes
  Duration
  Bits/s A to B
```

Figure 28: Best First Feature Selection

Output of feature selection using best first selection method for supervised learning on the complete dataset is shown in figure 28. This method selects combination of different may use backward or forward selection method to either remove or add features during selection process. Subset of features which were selected as result of this method were bytes, session duration and bits per second sent from source to destination mac.

Summary of different feature selection methods used is shown in table 4.

Table 4: Summary - Feature Selection Methods

Ser	Filter Method	Features Ranking
1.	Information Gain Ranking	Bytes, Packets, Duration, Bytes B to A, Packets B to A, Bytes A to B, Packets A to B, Bits/s B to A, Bits/s A to B
2.	Correlation Ranking	Duration, Bits/s B to A, Bits/s A to B, Packets B to A, Packets, Packets A to B, Bytes, Bytes A to B, Bytes B to A
3.	Gain Ratio Feature Evaluator	Duration, Bytes, Bits/s B to A, Bits/s A to B, Packets, Bytes B to A, Packets B to A, Bytes A to B, Packets A to B
4.	Best First	Bytes, Duration, Bits/s A to B

Summary of feature selection method with their output is shown in table 4. Features are shown in descending order with respect to their ranking. After studying and analyzing the results of chosen feature selection methods in detail, features selected for the proposed design were Bytes, Total packets exchanged between two workstations in LAN, Session duration and Packets sent from source to destination mac address.

These selected features can be extracted fast from the traffic and thus simplify detection of malicious packets. Data set after the selection of these four features is shown

Packets	Bytes	Packets A to B	Duration	Label
2228	146352	2228	3169	Normal
20	3874	20	28	Normal
14	868	8	1	Normal
42	10100	28	0	Normal
28	5646	14	0	Normal
28	5646	14	0	Normal
28	5646	14	0	Normal
28	5646	14	0	Normal
28	5646	14	0	Normal
40	2456	28	2	Normal
8	580	0	0	Normal
4	240	4	0	Normal
552	62260	552	1416	Normal
240	51840	240	1083	Normal
112	16576	112	917	Normal
8	688	8	0	Normal
32	2848	32	117	Normal
32	2208	32	117	Normal
4	240	4	0	Normal
8	688	8	0	Normal
8	688	8	0	Normal
8	480	8	0	Normal
8	720	8	0	Normal
96	13656	96	2944	Normal
184	20254	82	126	Normal
548	58118	288	1324	Normal
28	2068	0	54	Normal
4	1524	0	0	Normal
1214	115252	1214	3126	Normal
196	21038	88	62	Normal
374	36030	192	61	Normal

Figure 29: Data set with selected features

Four independent features were selected which are:

- **Total number of packets and bytes** exchanged during session between source and destination mac in order to differentiate between normal and malicious traffic as each packet sent consists of 60 bytes. No of packets and bytes exchanged between two mac addresses differs between normal and malicious traffic as normal exchange can have any number depending upon the size of data sent and received whereas in case of attacks such as mac flooding thousands of new packets were usually sent between different mac addresses each time making it differentiating feature and is as clear from the results produced by feature selection methods.
- **No Packets were sent from source mac to destination mac** in case of port stealing attack as compared to normal communication as thus selected.

- In case of mac flooding attack, thousands of unidirectional packets/ frames were sent from source to destination mac within seconds without establishing any session resulting in session duration of 0 seconds which is not the case in normal communication session between two layer 2 addresses and thus **Session Duration** was selected as differentiating feature.

5.4. Feature Evaluation

After the selection of features, next step was to evaluate them in terms of model evaluation metrics. Metrics in terms of which model performance was evaluated were Confusion matrix, ROC curve, Precision and Recall [44]. Before choosing appropriate machine learning algorithm in this research work, brief comparison was carried out among three different machine learning algorithms i.e J48, Naïve Bayes and Logistic Regression. Comparative study was performed on data set with selected independent features:

- **Logistic Regression:-** Out of test data set of 1027 instances, 1023 instances are correctly identified and 4 are incorrectly identified depending on trained dataset. 177 normal instances were correctly identified giving *True Positive rate* of 0.983 and *False Positive rate* of 0.12 ARP instances were correctly identified giving *True Positive rate* of 1 and *False Positive rate* of 0.48 Port steal instances were correctly identified giving *True Positive rate* of 1 and *False Positive rate* of 0.001. 785 Mac flood instances were correctly identified giving *True Positive rate* of 0.999 and *False Positive rate* of 0.012.
- **Naïve Bayes:-** Out of test data set of 1027 instances, 859 instances were correctly identified and 168 were incorrectly identified depending on trained dataset. 13 normal instances were correctly identified giving *True Positive rate* of 0.072 and *False Positive rate* of 0.001.12 ARP instances were correctly identified giving *True Positive rate* of 1 and *False Positive rate* of 0. 49 Port steal instances were correctly identified giving *True Positive rate* of 1 and *False Positive rate* of 0.009. 785 Mac flood instances were correctly identified giving *True Positive rate* of 0.999 and *False Positive rate* of 0.386.

- **J48:-** Out of test data set of 1027 instances, 1023 instances were correctly identified and 4 were incorrectly identified depending on trained dataset. 177 normal instances were correctly identified giving *True Positive rate* of 0.983 and *False Positive rate* of 0.001. 12 ARP instances were correctly identified giving *True Positive rate* of 1 and *False Positive rate* of 0. 49 Port steal instances were correctly identified giving *True Positive rate* of 1 and *False Positive rate* of 0.003. 785 Mac flood instances were correctly identified giving *True Positive rate* of 0.999 and *False Positive rate* of 0.

Results produced by both logistic regression and J48 algorithms were better depending upon the selected independent features. Supervised version of machine learning algorithm which was used for evaluation in this work was Logistic Regression.

5.4.1 Logistic Regression

Logistic regression in the field of statistics is the method that determines the dependent output on the basis of one or more independent variables. It is the method based on binary outputs i.e 1 or 0. All the output values in logistic regressions are scaled between 0 and 1 depending upon the threshold value (normally taken as 0.5). If the value of variable dependent is greater than threshold, it is classified as 1 and if value of dependent variable is less than threshold value it is classified as 0.

It is also acknowledged as sigmoid function. It's an S-shaped curve that takes any real valued number and maps it between 0 and 1 [45]. Sigmoid function is shown by following expression

$$y = \exp^{(b^0 + bx^1)} / (1 + \exp^{(b^0 + bx^1)}) [45]$$

where y is the output, b^0 is bias, e is base of natural logarithmic function and b^1 is coefficient of independent variable x.

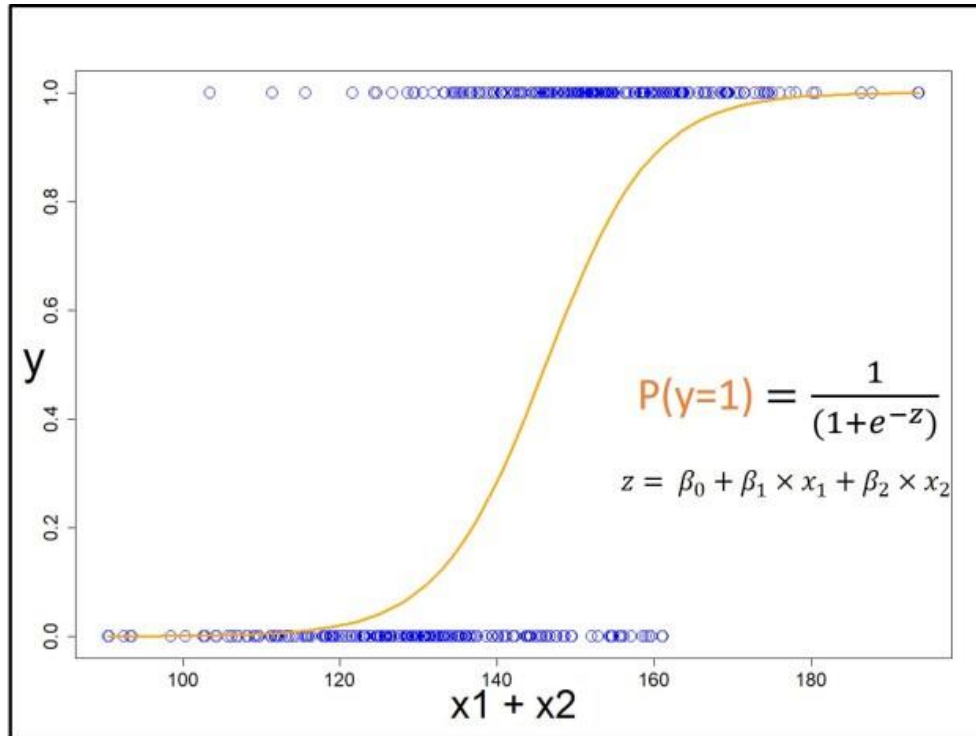


Figure 30: Logistic Regression Graph [46]

Feature evaluation was carried by Weka and Python. Evaluation was compared in terms of evaluation metrics [44].

5.4.2 Feature Evaluation in Weka

Weka is the software application having hundreds of built in machine learning tools that are helpful in data mining projects. Keeping these capabilities in view, it was used in this research work. First, data set finalized on the basis of selected features was uploaded in weka as shown in figure 31.

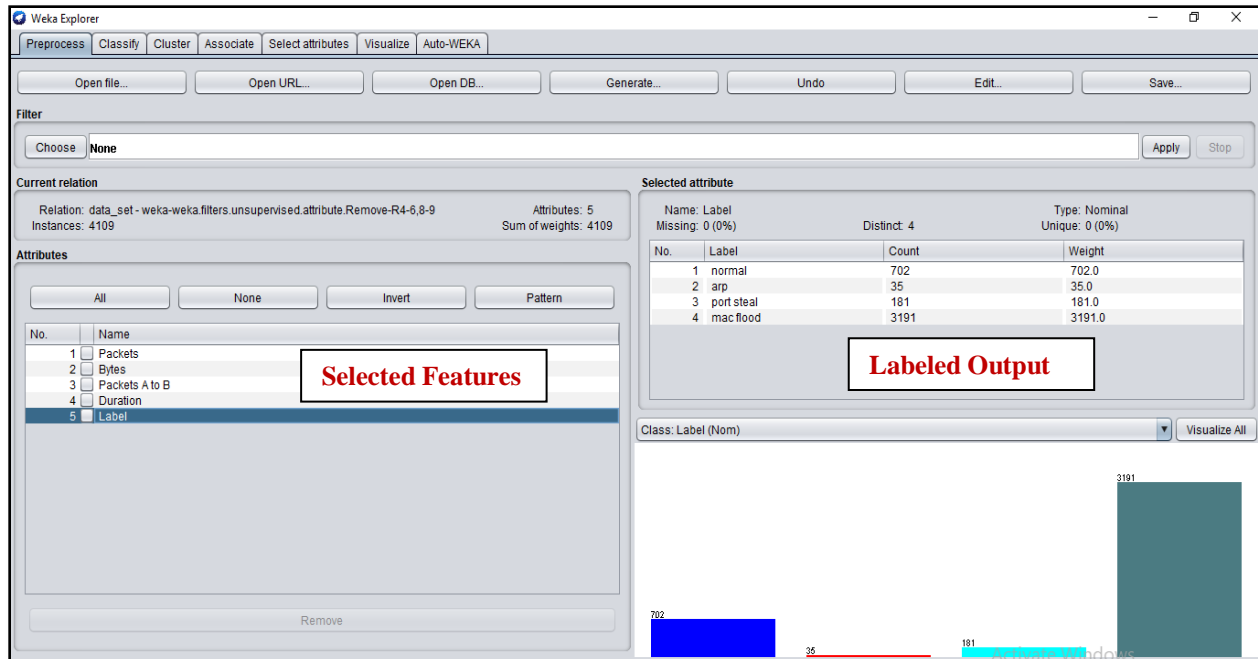


Figure 31: Data set preprocessing using Weka

Since Weka works on output with nominal values, there outcomes were classified as normal, arp, port steal and mac flood as shown in figure 31. After preprocessing of data set, logistic regression was applied under Classify tab in Weka. Classifier was set to the desired machine learning model i.e Logistic Regression in our case and test option was set to Percentage Split. Weka gives different test options from where desired one can be chosen which are

- **Use Training Set:-** This option uses complete given data set for training purpose.
- **Supply Test Set:-** Separate unknown data set can be given for test purpose using this option.
- **Cross Validation:-** This option divides the given data set into given number of folds using first set from each fold for testing and rest for training purpose.

- **Percentage Split:-** This option trains the model on splitting the dataset on the basis of given split ratio and testing the model on rest of the dataset.

In this research, percentage split with split window of 75% was used which works by training model on 75% of train data and testing the model on remaining 25% of supplied dataset as shown in figure 32.

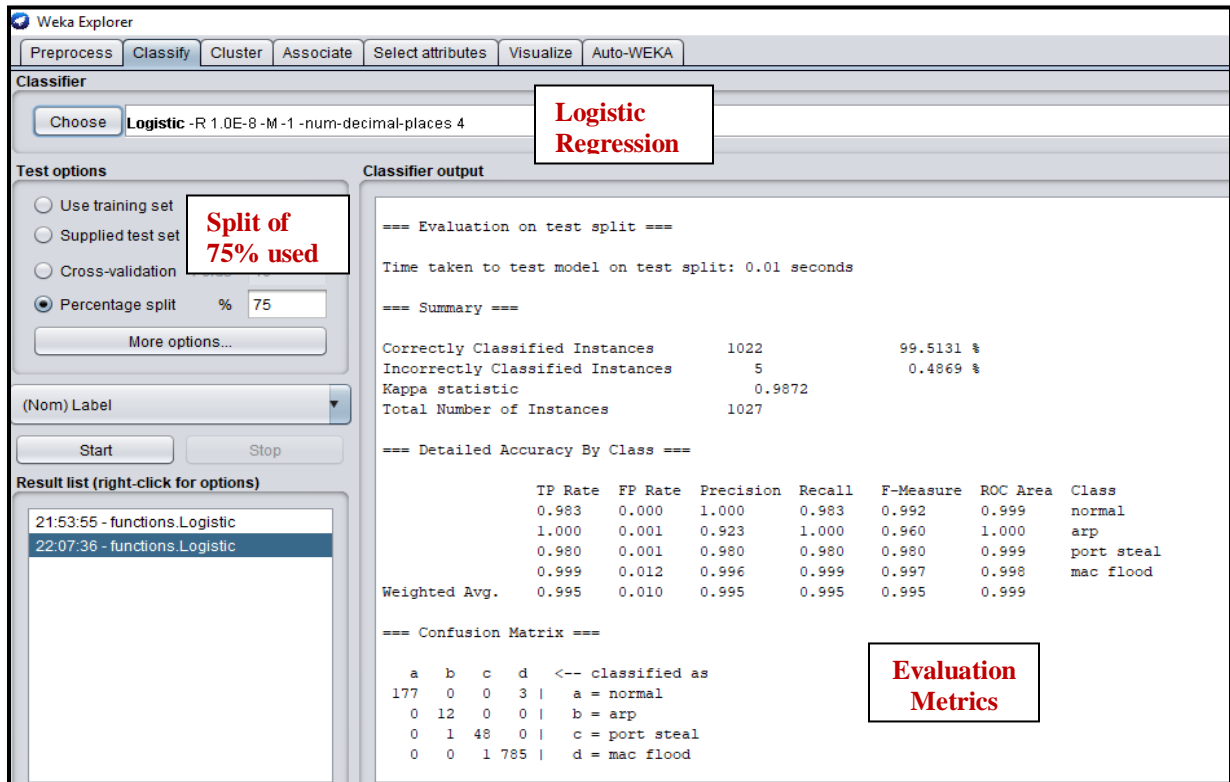


Figure 32: Logistic Regression using Weka

Figure 32 shows weka results on given dataset with logistic regression. Evaluation was done in terms of following

- **Confusion Matrix:-** It defines performance of model in terms of n x n matrix between actual and predicted values. It simply gives us the number of values or outcomes predicted correctly and incorrectly against the actual values or outcomes. Performance of any system model is evaluated using

this matrix between actual and predicted values. It consists of four important attributes

- **True Positive** - Real positive value identified correctly
- **True Negative** - Real negative value identified correctly
- **False Positive** - Real negative value identified incorrectly
- **False Negative** - Real positive value identified incorrectly
- **Precision:-** It is defined in terms of model performance evaluation as number of correctly predicted values i.e TP values out of actual values. It defines the model accuracy in terms that how precisely model predicts the outcome correctly. It may also be defined as ratio between the TP values to combination of TP and FP values or ratio between the TP values to the actual results. Higher precision value means that model has low false positive outcomes.
- **Recall:-** It is defined in terms of model performance evaluation as number of correctly predicted values i.e TP values out of total predicted values. It gives us the number indicating how many values are predicted correctly by the system model configured algorithms. It may also be defined as ratio between TP values to combination of TP and FN values or ratio of TP values to total predicted values. Higher recall value means that model has low false negative outcomes.
- **Receiver Operating Characteristics (ROC) curve:-** This parameter is very important in evaluating system performance. It is graphical plot between TP rate and FP rate with False Positive rate plotted on horizontal axis and True Positive rate plotted on vertical axis. Greater the area under the curve higher will be the system accuracy meaning model will be able to predict values correctly i.e 0s as 0s and 1s as 1s. It demonstrates the tradeoff between sensitivity (TP rate) and FP rate (1 – specificity) [44].

Summary of evaluation metrics calculated by Weka is shown in table 5.

Table 5: Weka Results

Label	Confusion Matrix	Precision	Recall	ROC
Normal	[847 0 3 177]	1.0	0.983	0.999
Arp	[1014 1 0 12]	0.923	1.0	1.0
Port Steal	[977 1 1 48]	0.980	0.980	0.998
Mac Flood	[238 3 1 785]	0.996	0.998	0.999

5.4.3 Feature Evaluation in Spyder IDE

Spyder IDE is python based development environment used by the developers to write python code. Evaluation was also done using python code. Python is simple and easy to use language having built in library functions to perform different tasks. Library function used in this research was scikit learn. Scikit learn is the python collection for machine learning that contain tools for data mining and analysis [47]. Results of python evaluation is shown in table 6.

Table 6: Spyder IDE Results

Label	Confusion Matrix	Precision	Recall	ROC
Normal	[832 0 22 174]	1.0	0.887	0.94
Arp	[1017 2 2 7]	0.778	0.778	0.89
Port Steal	[989 0 0 39]	1.0	1.0	1.0
Mac Flood	[230 14 0 784]	0.982	1.0	0.97

Python code used for implementing logistic regression using extracted features is shown in figure 33.

5.4.4 Implementation on Raspberry Pi

After the selection and evaluation of features using weka and python, next step was to implement our hardware to detect on the basis of learned parameters. Hardware in our design was raspberry pi model 2B having RAM of 1GB as shown in figure 14.

In the figure 5.1, raspberry pi was connected in LAN on the port operating in promiscuous mode. Purpose of enabling promiscuous mode was to mirror incoming traffic towards raspberry pi which would extract the features as selected in section 5.3 as shown in figure 34.

```

Editor - C:\Users\Gufran\Desktop\Kitsune-py-master\Kitsune-py-master\AfterImage\untitled6.py
temp.py x untitled0.py x untitled3.py x example.py x Kitsune.py x FeatureExtractor.py x
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from itertools import cycle
4 import pandas as pd
5 from sklearn.linear_model import LogisticRegression
6 from sklearn.metrics import roc_curve, auc
7 from sklearn.cross_validation import train_test_split
8 from sklearn.preprocessing import label_binarize
9 from sklearn.multiclass import OneVsRestClassifier
10 from sklearn.metrics import recall_score
11 from sklearn.metrics import precision_score
12 from sklearn.metrics import confusion_matrix
13 import random
14 dataset = pd.read_csv('C:\Users\Gufran\Desktop\data_set- spyder.csv')
15 dataset.head()
16 X = dataset.iloc[:, [0,1,2,3]].values
17 y = dataset.iloc[:, 4].values
18 y = label_binarize(y, classes=[0, 1, 2, 3])
19 n_classes = y.shape[1]
20 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.25, random_state=0)
21 classifier = OneVsRestClassifier(LogisticRegression (random_state=0)).fit(X, y)
22 y_score = classifier.fit(X_train, y_train).decision_function(X_test)
23 y_pred = classifier.fit(X_train, y_train).predict(X_test)
24 fpr = dict()
25 tpr = dict()
26 roc_auc = dict()
27 for i in range(n_classes):
28     fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_pred[:, i])
29     roc_auc[i] = auc(fpr[i], tpr[i])
30 plt.figure()
31 lw = 2
32 plt.plot(fpr[3], tpr[3], color='darkorange',
33         lw=lw, label='ROC curve (area = %0.2f)' % roc_auc[3])
34 plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
35 plt.xlim([0.0, 1.0])
36 plt.ylim([0.0, 1.05])
37 plt.xlabel('False Positive Rate')
38 plt.ylabel('True Positive Rate')
39 plt.title('Receiver operating characteristic example')
40 plt.legend(loc="lower right")
41 plt.show()
42 colors = cycle(['blue', 'red', 'green', 'orange'])
43 for i, color in zip(range(n_classes), colors):
44     plt.plot(fpr[i], tpr[i], color=color, lw=lw,
45             label='ROC curve of class {0} (area = {1:0.2f})'
46             .format(i, roc_auc[i]))
47 plt.plot([0, 1], [0, 1], 'k--', lw=lw)
48 plt.xlim([0.0, 1.0])
49 plt.ylim([0.0, 1.05])
50 plt.xlabel('False Positive Rate')
51 plt.ylabel('True Positive Rate')
52 plt.title('Some extension of Receiver operating characteristic to multi-class')
53 plt.legend(loc="lower right")
54 plt.show()
55 precision = dict()
56 recall = dict()
57 for i in range(n_classes):
58     precision[i]= precision_score(y_test[:, i], y_pred[:, i])
59     recall[i]= recall_score(y_test[:, i], y_pred[:, i])
60 conf_matrix = dict()
61 for i in range(n_classes):
62     conf_matrix = confusion_matrix(y_test[:, i], y_pred[:, i])
63     print (conf_matrix)

```

Figure 33: Python Code for Multiclass label Using Logistic regression

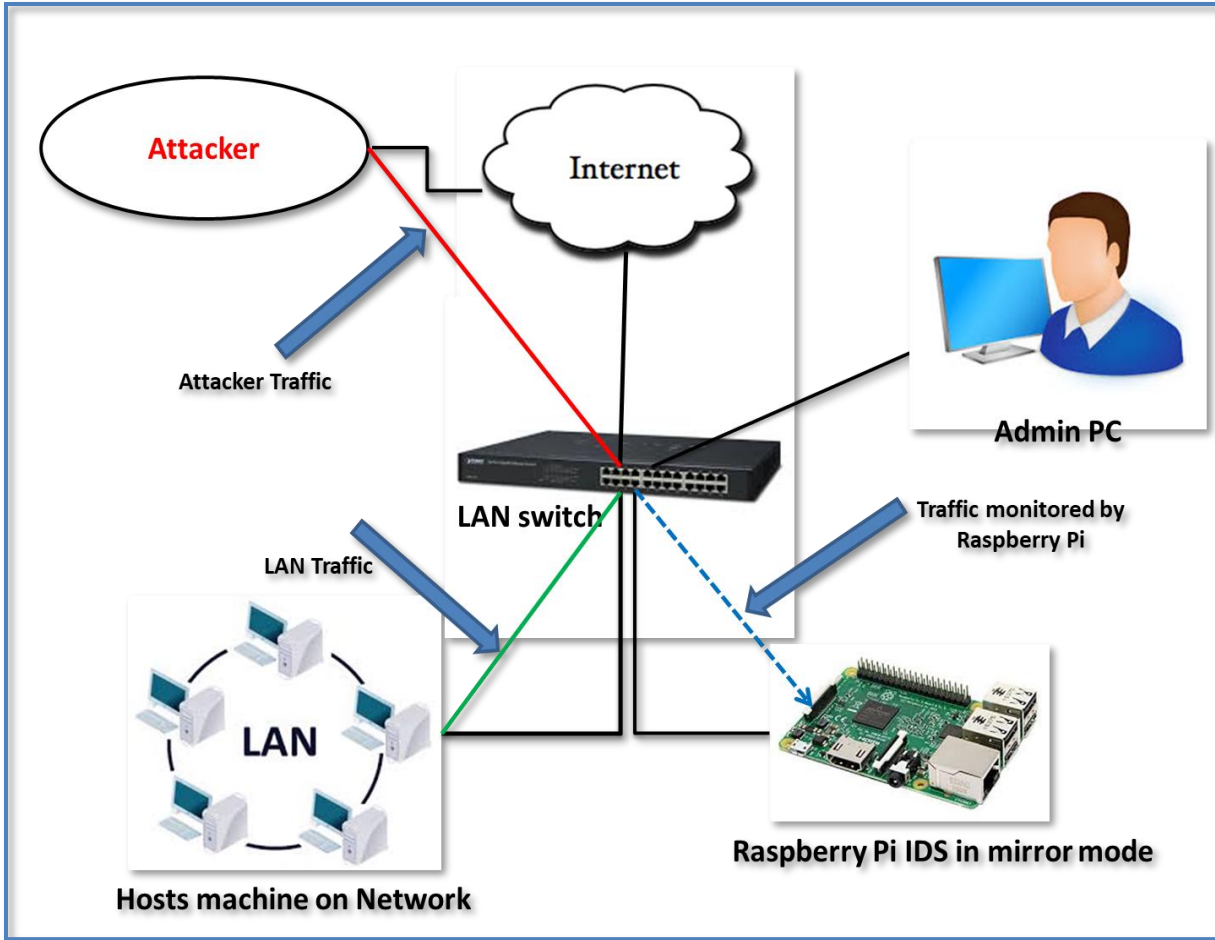


Figure 34: Raspberry Pi in promiscuous mode

Packets capture was done on raspberry pi in form of wireshark pcap files and feature extraction was done using tshark. Tshark is a terminal arranged adaptation of Wireshark intended for capturing and showing packets [48]. It provides user the command line facility to capture and filter network packets using different types of built in flags.

After capturing and extraction of features by raspberry pi, logistic regression model was applied on new captured data to predict and further classify the traffic on the basis of probabilities calculated by regression function as shown in figure 35.

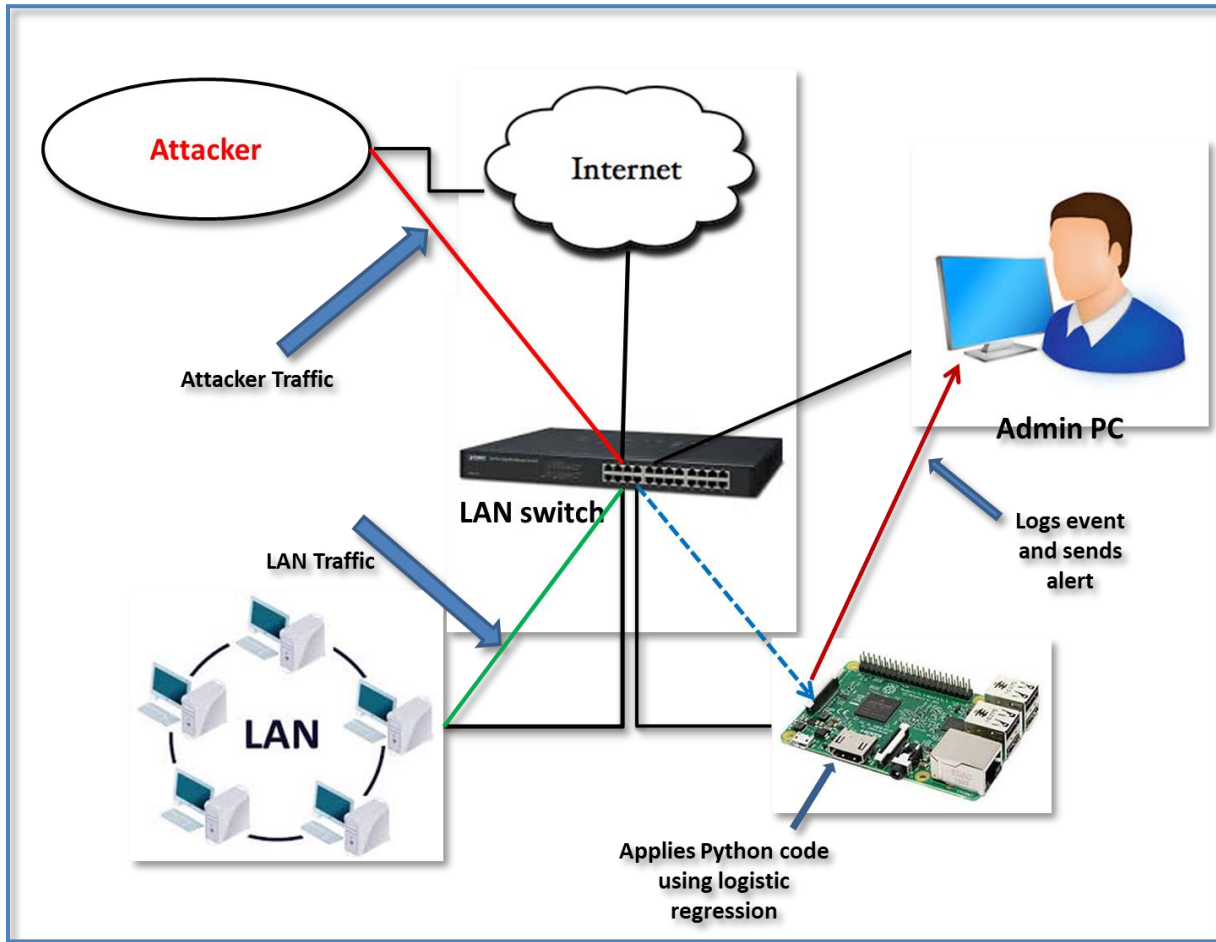


Figure 35: Intrusion Detection Flow

Since model was trained on more than one class, logistic regression was implemented as One vs Rest classifier on raspberry pi.

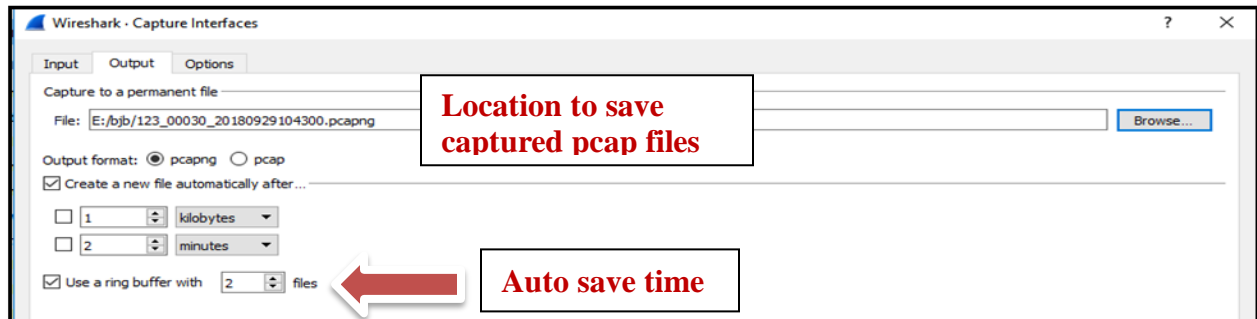


Figure 36: Enabling auto capture in Wireshark

Since we had to implement raspberry pi as passive NIDS, raspberry pi was configured to capture packets in form of pcap files using wireshark as shown in figure 36.

This was done using capture option before starting capturing of packets on raspberry pi. After the capture of pcap files, extraction of selected features was done using following tshark command on command prompt.

tshark -r <input pcap/pcapng file> -q -z conv,eth

Execution of command along with the output in form of extracted features is shown in figure 37.

```
C:\Program Files\Wireshark>tshark -r "E:\normal\capture2_split_00014_20180529113514.pcap" -q -z conv,eth
=====
Ethernet Conversations
Filter:<No Filter>
```

	<-		->		Total		Relative Start	Duration
	Frames	Bytes	Frames	Bytes	Frames	Bytes		
HewlettP_af:e6:00 <-> Dell_f2:c3:7e	4382	388386	13280	18780990	17662	19169376	0.004884000	31.0674
HewlettP_af:e6:00 <-> Dell_f2:c3:6e	6177	399613	10261	15454942	16438	15854555	0.000000000	31.8731
HewlettP_af:e6:00 <-> Dell_f2:c4:a5	734	51682	1227	1808124	1961	1859806	0.214249000	25.5055
HewlettP_af:e6:00 <-> Dell_af:d7:ff	49	3222	78	6862	127	10084	0.251749000	30.5498
HewlettP_af:e6:00 <-> Dell_b0:24:cf	41	2694	35	2944	76	5638	3.290852000	25.6398
Dell_f2:c3:7e <-> Dell_f2:c4:a5	40	21512	28	5752	68	27264	24.610335000	0.0607
HewlettP_cd:56:f7 <-> HewlettP_af:e6:00	29	2718	21	3707	50	6425	7.096930000	16.3402
Dell_f2:c3:c4 <-> Dell_f2:c4:a5	20	10756	28	5752	48	16508	24.599922000	0.0404
Dell_f2:a0:a7 <-> Dell_f2:c4:a5	20	10756	28	5752	48	16508	24.600267000	0.0467
Dell_f2:07:f5 <-> Dell_f2:c4:a5	20	10756	28	5752	48	16508	24.601109000	0.0497
Dell_f2:06:86 <-> Dell_f2:c4:a5	20	10756	28	5752	48	16508	24.601249000	0.0427

Figure 37: tshark command to extract features

Tshark command was used to filter out the layer 2 conversations from captured pcap file in form of session duration, packets number in total along with bytes sent from source mac to destination mac and vice versa as shown in figure 37. Features were further saved in .csv file format which was used as input to logistic regression function on raspberry pi. Since raspberry pi is small size computer with limited RAM and storage, capturing and extraction could also be done on separate machine for preprocessing of incoming data to avoid load on raspberry pi.

Logistic regression algorithm was then applied and tested on new set of unlabeled data to predict the output on the basis of calculated probabilities. Since, training was done

on selected features using multilabel data, predicted of outcome was done by comparing probabilities calculated by logistic regression on each instance and selecting event with high probability. Python code implemented on raspberry pi for this purpose is shown in figure 38.

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split
from sklearn.multiclass import OneVsRestClassifier
dataset = pd.read_csv('C:\Users\Gufraan\Desktop\data_set- spyder.csv')
dataset.head()

dataset_2 = pd.read_csv('E:/desktop/progress/train test datasets/test_set - spyder.csv')
dataset_2.head()

X = dataset.iloc[:, [0,1,2,6]].values
y = dataset.iloc[:, 9].values

X_test1 = dataset_2.iloc[:, [0,1,2,3]].values ← New test set to perform predictions

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.25, random_state=0)
classifier = OneVsRestClassifier(LogisticRegression (random_state=0)).fit(X, y)
y_score = classifier.fit(X_train, y_train).decision_function(X_test)

y_pred = classifier.fit(X_train, y_train).predict(X_test1)

y_prob = classifier.fit(X_train, y_train).predict_proba(X_test1)

print classifier.score(X,y)
print('intercept:', classifier.intercept_)
y_coef= classifier.coef_


```

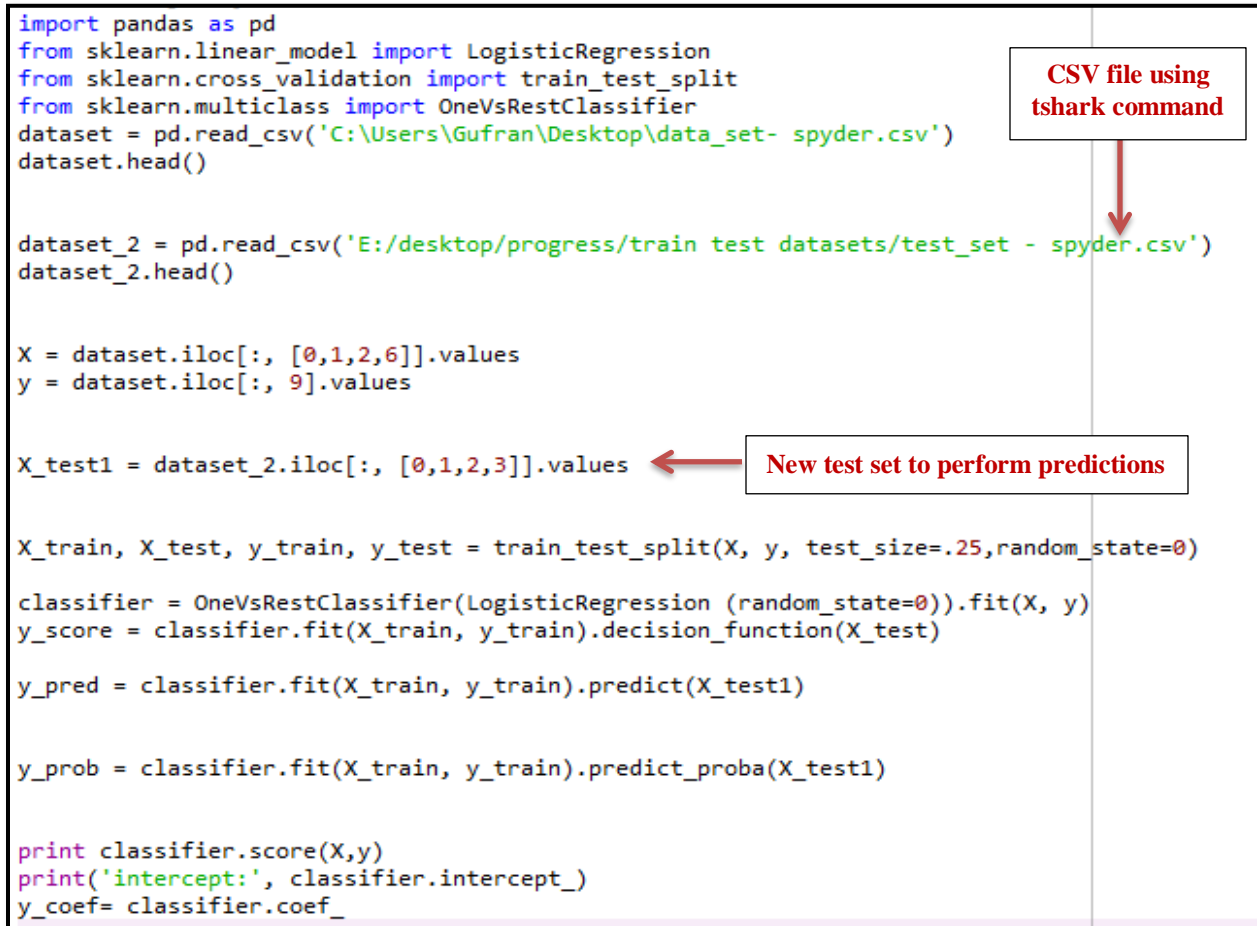


Figure 38: Python code on Raspberry Pi

CHAPTER 6: RESULTS AND DISCUSSION

The implementation of proposed design to detect infrastructure based MITM attacks using machine learning model was carried out in two phases. In the first phase, results were generated on the basis of selected set of features using our own generated dataset on data mining application and java based python development environment i.e Weka and Spyder IDE respectively. In the second phase, trained logistic regression model was implemented on raspberry pi hardware to identify packets on the basis of trained labels. Comparison of results in terms of model metrics and perceptions of both the stages were noted and are elaborated in the consequent sections.

6.1. Weka Evaluation Results

In this section, model evaluation by Weka in terms of metrics are presented and discussed. After the selection of features from the dataset using feature selection methods, evaluation of features was carried out using logistic regression as discussed in section 5.4.2. Evaluation of model was carried out in terms of evaluation metrics i.e confusion matrix, precision, recall and ROC curves as shown in table 5.

Training and testing was performed in weka using percentage split of 75%. Training of model was carried out on **3083** instances and test was carried out on remaining **1027** instances from the given data set of **4110** instances. Weighted average of model evaluation in terms of TP and FP rate were **0.995** and **0.010** respectively. Test results in terms of evaluation metrics with selected features against each type of traffic is shown in table 7. Impact of precision, recall and ROC curve have already been mentioned in section 5.4.2.

Table 7: Weka Evaluation Metrics

Ser	Class	TP rate	FP rate	Precision	Recall	ROC curve
1.	Normal	0.983	0.000	1.000	0.983	0.999
2.	ARP Spoof	1.000	0.001	0.923	1.000	1.000
3.	Port Steal	0.980	0.001	0.980	0.980	0.999
4.	Mac Flood	0.999	0.012	0.996	0.999	0.998

Confusion matrix is defined as the matrix of $n \times n$ order between actual and predicted values. Generated confusion matrix by Weka on test set of **1027** instances is shown in table 8. It shows matrix of 4×4 order between actual against the predicted class labels. Class labels for traffic have been classified into four different categories i.e Normal, Arp Spoof, Port steal and Mac Flood. Confusion matrix evaluates the system performance in terms of attributes which are TP, TN, FP and FN values.

These attributes of confusion matrix gives the model performance in terms of the fact that how accurately model predicts the correct class labels against actual class labels. Values in confusion matrix shows that actual 177 instances of normal class label were predicted correctly with 3 instances predicted incorrectly, actual 12 instances of arp spoof class label were predicted correctly, actual 48 instances with port steal class label were predicted correctly with 1 instance predicted incorrectly and actual 785 instances with mac flood class label were predicted correctly with 1 instance predicted incorrectly.

Table 8: Confusion Matrix - Weka

	Class	Predicted			
		Normal	ARP spoof	Port Steal	Mac Flood
Actual	Normal	177	0	0	3
	ARP Spoof	0	12	0	0
	Port Steal	0	1	48	0
	Mac Flood	0	0	1	785

Since the test set given to machine learning model was labeled consisting of 1027 instances with 180 normal, 12 ARP spoof, 49 port steal and 786 Mac flood instances. On the basis of values shown in table 8, important attributes of confusion matrix inferred against each class label is shown in table 9.

Table 9: Confusion Matrix Attributes - Weka

Ser	Class	TP	FP	TN	FN
1.	Normal	177	0	847	3
2.	ARP Spoof	12	1	1004	0
3.	Port Steal	48	1	967	1
4.	Mac Flood	785	3	238	1

Table 9 shows the attributes of confusion matrix for each class label concluded from the values shown in table 8 indicating the impact of confusion matrix on system performance. From the attribute values shown against different class labels, following results were concluded:

- For normal class label, model gives confusion matrix attributes with TP value of 177, FP value of 0, TN value of 847 and FN value of 3 for test set of 1027 instances. It shows that our model predicted 177 instances correctly out of total 180 actual normal instances in test set with overall TP rate of 0.983.
- For arp spoof class label, model gives confusion matrix attributes with TP value of 12, FP estimation of 1, TN estimation of 1004 and FN estimation of 0 for test set of 1027 instances. It shows that our model predicted 12 instances correctly out of total 12 actual arp spoof instances in test set with overall TP rate of 1.00.
- For port steal class label, model gives confusion matrix attributes with TP value of 48, FP estimation of 1, TN estimation of 967 and FN estimation of 1 for test set of 1027 instances. It shows that our model predicted 48 instances correctly out of total 49 actual port steal instances in test set with overall TP rate of 0.980.
- For mac flood class label, model gives confusion matrix attributes with TP value of 785, FP value of 3, TN value of 238 and FN value of 1 for test set of 1027 instances. It shows that our model predicted 785 instances correctly out of total 786 actual port steal instances in test set with overall TP rate of 0.999.

Graphical plot of these values for each class label is also shown in figure 39.

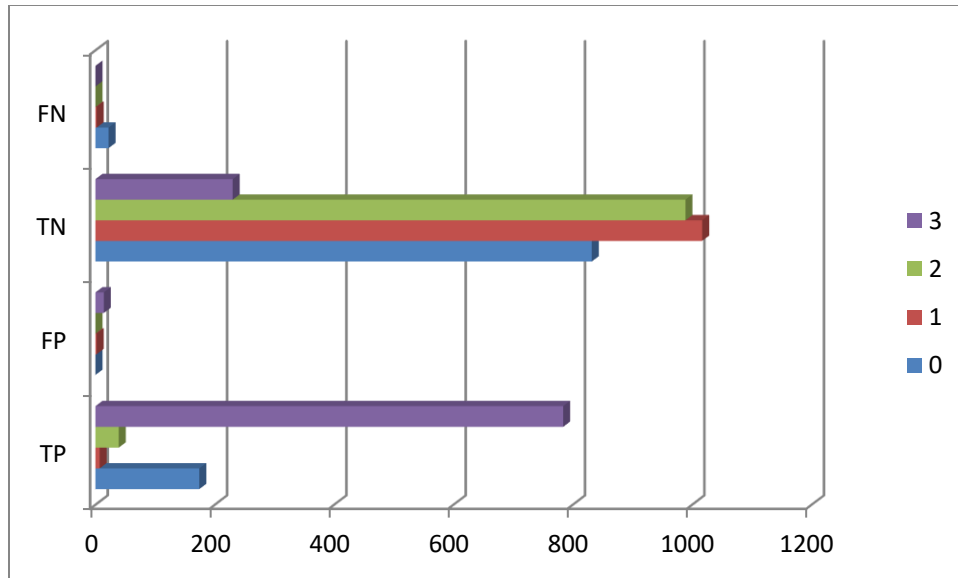


Figure 39: Graphical Plot of Weka Results

6.2. Python Evaluation and Results

In this section, model evaluation using python in terms of metrics are presented and discussed. Evaluation of features was carried out using sklearn library of machine learning as discussed in section 5.4.3. One vs Rest classifier of logistic regression was implemented on data set with the test size of 0.25. Model evaluation was carried out in terms of evaluation metrics i.e confusion matrix, precision, recall and ROC curves as shown in table 6.

Logistic regression was applied on data set with label values to predict the output against the test set. Weights against each column of feature was calculated and used in logistic regression sigmoid function to calculate the probability for each instance.

Model evaluation using python code shown in figure 33 in terms of precision, recall and accuracy against each type of traffic is shown in table 10.

Table 10: Python Evaluation Metrics

Ser	Label	Precision	Recall	Accuracy
1.	Normal (Label 0)	1.000	0.887	0.978
2.	ARP SpooF (Label 1)	0.778	0.778	0.996
3.	Port Steal (Label 2)	1.0	1.0	1.0
4.	Mac Flood (Label 3)	0.982	1.0	0.986

Table 10 shows model performance in terms of accuracy of prediction for label 0, 1, 2 and 3 respectively. Precision evaluates system performance in terms of fact that how much of model positive predictions are correct. Precision values for each class label in table 10 show that our model predicted class label 0 correctly every time, class label 1 correctly 77.8% of the time, class label 2 correctly every time and class label 3 correctly 98.2% of the time. Recall evaluates system performance in terms of fact that what proportion of all actual positive values is predicted by model correctly. Recall values for each class label in table 10 show that our model predicted 88.7% of all class label 0 correctly with 97% accuracy, 77.8% of all class label 1 correctly with 99% accuracy, 100% of all class label 2 correctly with 100% accuracy and 100% of all class label 3 correctly with 98% accuracy.

Another factor used for evaluation was ROC curve. ROC curve as explained in section 5.4.2 evaluates the model performance by calculating the area under the curve by means of graph plotted between True Positive rate and False Positive rate. Greater area covered under graph for each of class label, it tells the model accuracy for the prediction of that respective class label. ROC curves indicating the area under the curve values for each of class label is shown in figure 40.

Figure 40 shows that model predicted ROC curve value of 0.94 for class label 0 indicating 94% accuracy in correctly predicting class label 0, ROC curve value of 0.89 for class label 1 indicating 89% accuracy in correctly predicting class label 1, ROC curve value of 1.0 for class label 2 indicating 100% accuracy in correctly predicting class label 2 and ROC curve value of 0.97 for class label 3 indicating 97% accuracy in correctly predicting class label 3 respectively.

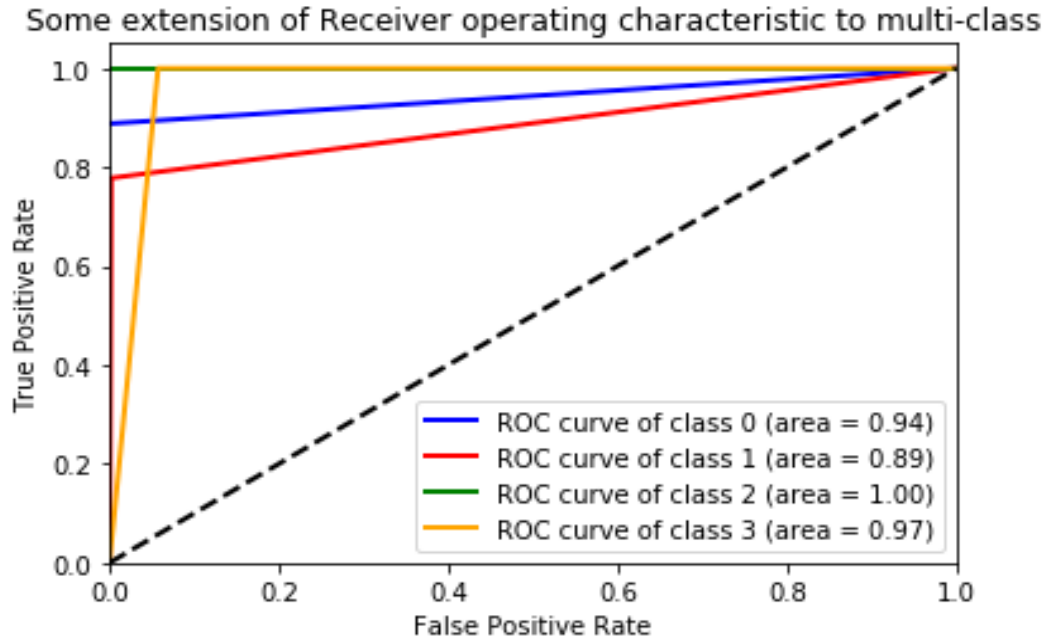


Figure 40: ROC curves for each class label

Performance of logistic regression model in terms of confusion matrix by python on test set having multi label traffic is shown in table 11.

Table 11: Confusion Matrix – Python

	Class Label	Predicted			
		0	1	2	3
Actual	0	174	0	0	6
	1	0	7	0	8
	2	0	1	39	0
	3	0	1	0	784

Confusion matrix indicates the actual values in test set and their corresponding predicted values against each class. On the basis of results shown in table 11, TP and FP were inferred against each class label as shown in table 12.

Table 12: Calculated TP and FP values - Python

Ser	Class Label	TP	FP
1.	0	174	0
2.	1	7	2
3.	2	39	0
4.	3	784	14

Table 12 shows that python code using machine learning on the basis of extracted features correctly identifies 174 instances of class label 0, 7 instances of class label 1, 39 instances of class label 2 and 784 instances of class label 3 respectively. Graphical plot of these values for each class is also shown in figure 41.

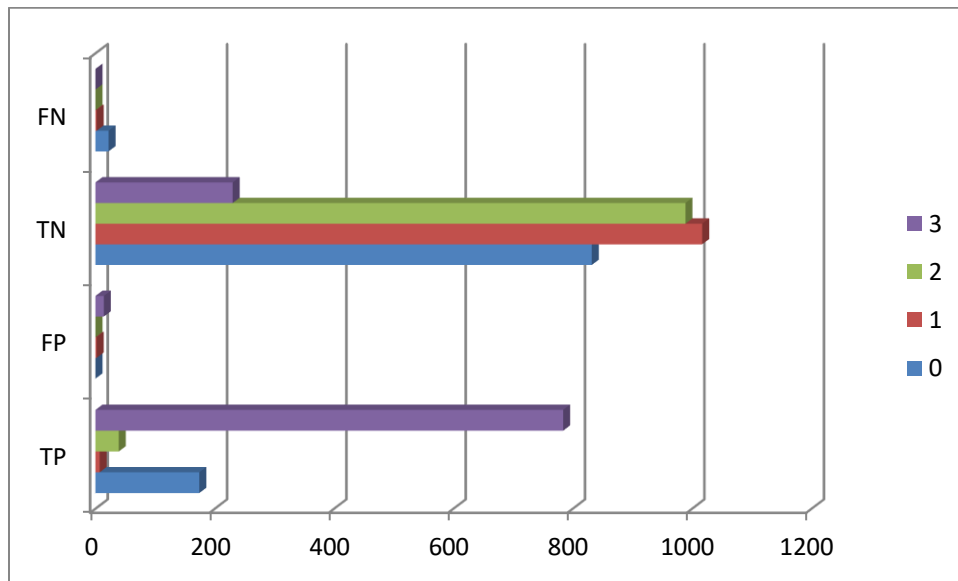


Figure 41: Graphical Plot of Python Results

Comparison in form summary of results generated by Weka and Python is shown in table 13.

Table 13: Summary of Results

Metrics	Weka				Python			
	Normal	ARP	Port Steal	Mac Flood	Label 0 (Normal)	Label 1 (ARP)	Label 2 (Port Steal)	Label 3 (Mac Flood)
TP	177	12	48	785	174	7	39	784
FP	0	1	1	3	0	2	0	14
Precision	1.00	0.923	0.980	0.996	1.00	0.778	1.00	0.982
Recall	0.983	1.00	0.980	0.999	0.887	0.778	1.00	1.00
ROC	0.999	1.00	0.999	0.998	0.94	0.89	1.00	0.97

Graphical plot of results generated by Weka and Python is also shown in figure 42.

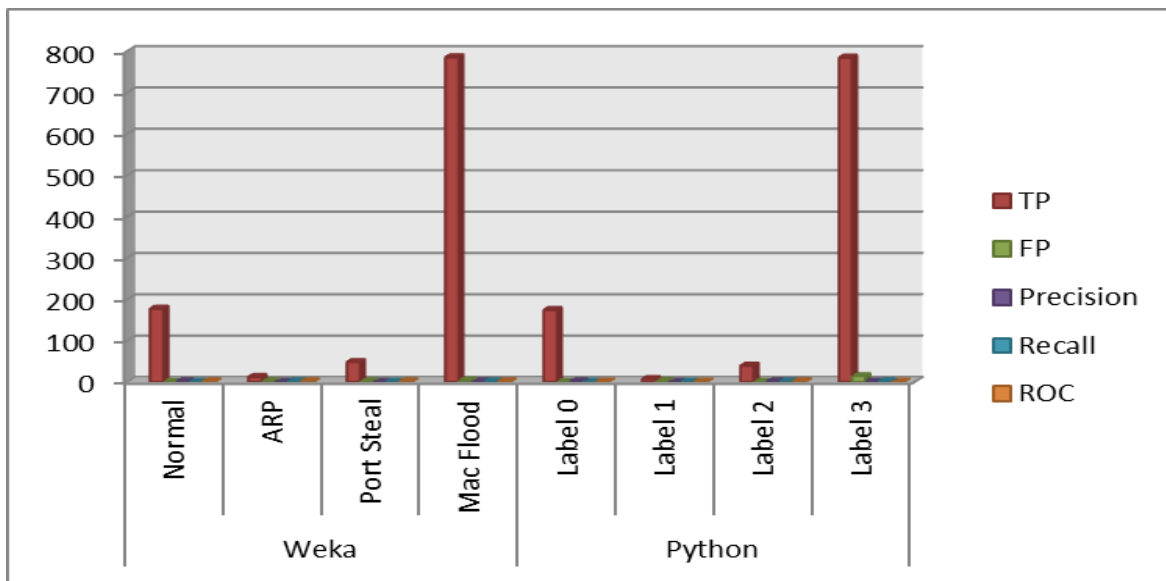


Figure 42: Graphical Plot of Summary

6.3. Raspberry Pi Implementation Results

In this section, results generated by implementing the trained and tested model of machine learning using logistic regression algorithm on raspberry pi computer were discussed. After the training and testing of model using weka and python, model was implemented on

raspberry pi in order to determine whether the trained model predicts class or labels. Raspberry pi was configured to monitor the network traffic in promiscuous mode capturing packets in form of pcap files after every 2 minutes. Selected features as discussed in section 5.4 were then extracted using tools such as tshark and logistic regression model was tested against the new set of features on which model was trained previously as shown in figure 5.28. New set of features given as input to model was unlabeled.

Probability against each instance in new test set using weights for each feature was calculated to predict the corresponding class label. On the basis of packets captured automatically after 2 minute and feature extraction, raspberry pi predicts the labels as shown in figure 43.

	0	1	2	3
0	0	0	0	1
1	0	0	1	0
2	0	0	0	1
3	0	1	0	0
4	1	0	0	0
5	1	0	0	0
6	0	0	0	1
7	1	0	0	0
8	0	1	0	0
9	1	0	0	0
10	0	0	0	1
11	0	0	0	1

Figure 43: Raspberry Pi Output

Figure 43 shows the output of implementation of trained model on raspberry pi. Once trained, raspberry pi predicted labels on the basis of learned features and trained data set. Model evaluation is only possible when we have set of known actual values so that metrics could be inferred by comparing actual and predicted labels.

In order to check performance of hardware on new data set depending upon model training, data set in controlled environment was collected over the same network having traces of both benign and attack packets with total of 300 instances. Testing was performed on raspberry pi and when predicted values of instances depending upon the selected features were compared with actual values, it was seen that 79 out of 91 instances of normal traffic were predicted correctly, 19 out of 23 instances of arp spoof traffic were predicted correctly, 24 out of 26 instances of port steal traffic were predicted correctly and 157 out of 159 instances of mac flood traffic were predicted correctly. Performance of raspberry pi in form of evaluation metrics discussed previously in this research work is shown in table 14.

Table 14: Raspberry Pi Implementation Results

Ser	Class Label	Confusion Matrix Attributes		Precision	Recall
		TP	FP		
1.	Normal	79	1	0.987	0.877
2.	Arp Spoof	19	2	0.904	0.826
3.	Port Steal	24	0	1.00	0.923
4.	Mac Flood	157	12	0.928	0.987

Raspberry Pi implementation results in table 14 indicate hardware performance in a form of precision and recall for multiple class labels of traffic discussed in this research work. Since precision evaluates system performance in terms of proportion of positive predictions done correctly, values in table 14 shows that raspberry pi predicted normal traffic 98.7% of time correctly, arp spoof traffic 90.4% of time correctly, port steal traffic every time correctly and mac flood traffic 92.8% of time correctly. Since recall evaluates system performance in terms of proportion of actual positive predictions done correctly, its values in table 14 shows that raspberry pi predicted 87.7% of normal traffic correctly, 82.6% arp spoof traffic correctly, 92.3% of port steal traffic correctly and 98.7% of mac flood correctly. Overall performance of hardware depending on training set and testing on new set with selected features was 93 %. Graphical plot of this discussion is shown in figure 44.

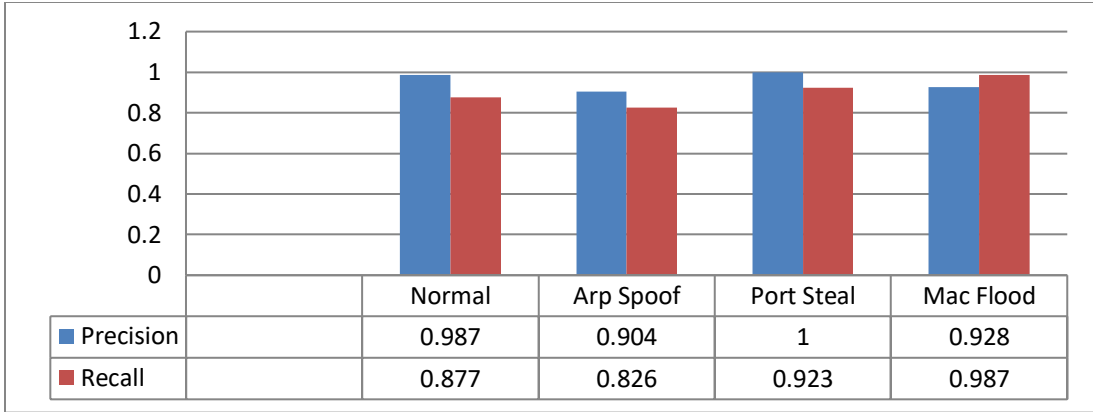


Figure 44: Graphical Plot - Raspberry Pi Results

CHAPTER 7: Conclusion

With the advancements in the arena of information technology, need for sharing and protecting information has increased. Organizations are giving due importance to protect their data against the unauthorized access by spending huge amounts on security solutions available in market. Goal of organizations using expensive security solution is to protect their most important asset which is data from variety of cyber-attacks. Commercially available solutions for intrusion detection and prevention are a sort of financial burden on small and medium IT setups due to their infrastructure constraints and financial limitations as these solutions demand high cost in terms of their installation and maintenance.

Keeping the limitations and requirements of small scale IT setups in view, scalable and inexpensive secure solution was required to protect eavesdropping of data. With the passage of time, many methods, models and procedures using raspberry pi computer have been proposed. Models proposed or developed in this regard either used open source intrusion detection and prevention tools or integrated algorithms against specific threat. Some of these proposed models and solutions have been critically reviewed in previous chapters and their flaws points are identified. Raspberry pi computer as discussed during the course of this research is a low cost plug and play hardware that operates on low power with no requirement of installation infrastructure setup and maintenance. Such hardware can easily be scaled up in size on need of user in form of multiple nodes at different locations when size of network increases. Since in the light of flaws identified in the previously presented security solutions using raspberry pi and its advantages, a solution is proposed that offers better protection with low scalability, maintenance and infrastructure cost.

The proposed solution using raspberry pi computer is effective, feasible and affordable by every organization with constraints and limitations. It provides not only protection against unauthorized data access but also a secure solution that can be scaled up depending upon the size of network due to its scalability. We developed a tool using machine learning for detection extracting network independent features without setting any threshold as a proof of concept with

against some flavors of MITM attacks to evaluate our proposed solution. After the evaluation of the prototype tool and comparison with proposed solutions using raspberry pi computers and techniques used by detection tools such as Xarp for arp spoof, port steal and mac flood, it can be concluded that our approach suits well to small and medium IT setups.

7.1. Future Directions

Since the tool is developed as proof of concept and it is a prototype tool, it has its limitations. Our prototype tool is developed using logistic regression algorithm to detect MITM attacks automatically irrespective of network size. Training and testing of model before implementing it on raspberry pi is carried out using self-generated data set having traces of attacks as no public data set was available having traces of MITM attacks. Our solution provides detection against ARP spoof, Port steal and Mac Flood attacks. An industrial standard intrusion detection tool can be developed based on our proposed solution. It may add following features and improvements to provide

- Support for protection against other MITM attacks i.e DHCP and DNS spoofing.
- Support for more composite and large networks by growing the resources and adding multiple raspberry pi computers as intrusion detection nodes.
- Integrating cameras to monitor motion detection to prevent physical intrusions to critical infrastructure.

REFERENCES

- [1]. Micro Focus. What is Data Security? | Micro Focus. Retrieved September 13, 2018, from <https://www.microfocus.com/en-us/what-is/data-security>
- [2]. Conti, M., Dragoni, N., & Lesyk, V. (2016). A Survey of Man in the Middle Attacks. *IEEE Communications Surveys and Tutorials*, 18(3), 2027–2051. <https://doi.org/10.1109/COMST.2016.2548426>
- [3]. John Wiley & Sons. Examining Different Types of Intrusion Detection Systems - dummies. Retrieved September 13, 2018, from <https://www.dummies.com/computers/operating-systems/windows-xp-vista/examining-different-types-of-intrusion-detection-systems/>
- [4]. Scarfone, K. A., & Mell, P. M. (2007). Guide to Intrusion Detection and Prevention Systems (IDPS). Gaithersburg, MD. <https://doi.org/10.6028/NIST.SP.800-94>
- [5]. Nayak, G. N., & Samaddar, S. G. (2010). Different flavours of Man-In-The-Middle attack, consequences and feasible solutions. *Proceedings - 2010 3rd IEEE International Conference on Computer Science and Information Technology, ICCSIT 2010*, 5, 491–495. <https://doi.org/10.1109/ICCSIT.2010.5563900>
- [6]. Paper, C., & Nandi, S. (2014). Detecting ARP Spoofing : An Active Technique Detecting ARP Spoofing : An Active Technique, (December 2005). <https://doi.org/10.1007/11593980>
- [7]. Calvert, C., Khoshgoftaar, T. M., Najafabadi, M. M., & Kemp, C. (2016). A Procedure for Collecting and Labeling Man-in-the-Middle Attack Traffic. *International Journal of Reliability, Quality and Safety Engineering*, 24(01), 1750002. <https://doi.org/10.1142/s0218539317500024>
- [8]. Convery, S. (2002). Hacking Layer 2: Fun with Ethernet Switches. *BlackHat USA*, 1–84.
- [9]. Djanali, S., Arunanto, F. X., Pratomo, B. A., Studiawan, H., & Nugraha, S. G. (2014). SQL injection detection and prevention system with raspberry Pi honeypot cluster for trapping attacker. *ISTMET 2014 - 1st International Symposium on Technology*

- Management and Emerging Technologies, Proceedings, (Istmet), 163–166.
<https://doi.org/10.1109/ISTMET.2014.6936499>
- [10]. Mahajan, S., Adagale, A. M., & Sahare, C. (2016). Intrusion Detection System Using Raspberry PI Honeypot in Network Security. *International Journal of Scientific and Engineering Research- IJESC*, 6(3), 2792–2795. <https://doi.org/10.4010/2016.651>
- [11]. Khamphakdee, N. (2014). Improving Intrusion Detection System Based on Snort Rules for Network Probe Attack Detection, (May 2015).
<https://doi.org/10.1109/ICoICT.2014.6914042>
- [12]. Uddin, A., & Hasan, L. (2016). Design and Analysis of Real-time Network Intrusion Detection and Prevention System using Open Source Tools. *International Journal of Computer Applications*, 138(7), 975–8887. <https://doi.org/10.5120/ijca2016908921>
- [13]. SnortSam | Engineering360.Retrieved September 19, 2018, from
<https://www.globalspec.com/reference/36966/203279/snortsam>
- [14]. Mustafa Cosar, S. K. (2017). A firewall application on SOHO networks with Raspberry Pi and snort. In *International Conference on Computer Science and Engineering (UBMK)* (Vol. 2, pp. 1000–1003).
- [15]. ICMP Flood, Ping Flood, Smurf Attack.Retrieved from
<https://www.f5.com/services/resources/glossary/icmp-flood-ping-flood-smurf-attack>
- [16]. Ar Kar Kyaw, Yuzhu Chen, J. J. (2015). Pi-IDS: Evaluation of Open-Source Intrusion Detection Systems on Raspberry Pi 2. In *Second International Conference on Information Security and Cyber Forensics (InfoSec)* (pp. 165–170).
<https://doi.org/10.1109/infosec.2015.7435523>
- [17]. Mehra, P. (2012). A brief study and comparison of Snort and Bro Open Source Network Intrusion Detection Systems.
- [18]. Srimathi, P., Venkat, S., Yamuna, M., & Sudha, V. (2017). Design of Low Cost CNC Controller Using. *International Journal of Scientific Engineering and Research (IJSER)*, 5(3), 2015–2017.
- [19]. What is a Raspberry Pi? | Opensource.com.Retrieved from
<https://opensource.com/resources/raspberry-pi>

- [20]. Raspberry Pi 2 arrives with quad-core CPU, 1GB RAM, same \$35 price | Ars Technica. Retrieved from <https://arstechnica.com/information-technology/2015/02/raspberry-pi-2-arrives-with-quad-core-cpu-1gb-ram-same-35-price/>
- [21]. Kali Linux Tutorial. Retrieved from https://www.tutorialspoint.com/kali_linux/index.htm
- [22]. Sheena Angra, S. A. (2017). Machine Learning and its Applications: A Review. International Conference on Big Data Analytics and Computational Intelligence, 57–60.
- [23]. Saimadhu Polamuri. Supervised and Unsupervised learning. Retrieved from <http://dataaspirant.com/2014/09/19/supervised-and-unsupervised-learning/>
- [24]. G. Holmes, A. Donkin, I. h. W. (1994). WEKA: a machine learning workbench. <https://doi.org/10.1109/anzis.1994.396988>
- [25]. The Python Tutorial — Python 3.8.0a2 documentation. Retrieved from <https://docs.python.org/3.8/tutorial/index.html>
- [26]. UGR'16 Dataset. Retrieved from <https://nesg.ugr.es/nesg-ugr16/>
- [27]. Public PCAP files for download. Retrieved from <https://www.netresec.com/index.ashx?page=PcapFiles>
- [28]. Wang, J., Zhao, P., Hoi, S. C. H., & Jin, R. (2014). Online feature selection and its applications. *IEEE Transactions on Knowledge and Data Engineering*, 26(3), 698–710. <https://doi.org/10.1109/TKDE.2013.32>
- [29]. Chandrashekar, G. and Sahin, F. (2014) A Survey on Feature Selection Methods. *Computers and Electrical Engineering*, 16-28. <https://doi.org/10.1016/j.compeleceng.2013.11.024>
- [30]. Aditya Mishra. (2018). Metrics to Evaluate your Machine Learning Algorithm. Retrieved October 16, 2018, from <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234>
- [31]. 3isys CS-5500 Series L3 Gigabit Ethernet Switch CS-5500-28T-G. Retrieved from <https://www.helptechco.com/content/CS-5500-28T-G>
- [32]. Beebe, M. A. (Ed.). (2003). *Africa Dot Edu: It opportunities and higher education in Africa*. Tata McGraw Hill India.

- [33]. Mike Sconzo. (2018). SecRepo - Security Data Samples Repository. Retrieved October 16, 2018, from <http://www.secrepo.com/>
- [34]. The UCI KDD Archive Information and Computer Science, & University of California, I. (1999). KDD Cup 1999 Data. Retrieved January 6, 2019, from <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>
- [35]. Wang, S., Xu, D., & Yan, S. (2010, April). Analysis and application of Wireshark in TCP/IP protocol teaching. In 2010 International Conference on E-Health Networking Digital Ecosystems and Technologies (EDT) (Vol. 2, pp. 269-272). IEEE.
- [36]. How to perform a Man-in-the-middle (MITM) attack with Kali Linux | Our Code World. Retrieved from <https://ourcodeworld.com/articles/read/422/how-to-perform-a-man-in-the-middle-mitm-attack-with-kali-linux>
- [37]. MiTM Attack with Ettercap | hackers-arise. Retrieved from <https://www.hackers-arise.com/single-post/2017/08/28/MiTM-Attack-with-Ettercap>
- [38]. Yeung, K. H., Fung, D., & Wong, K. Y. (2008, March). Tools for attacking layer 2 network infrastructure. In Proceedings of the international multiconference of engineers and computer scientists (Vol. 2, pp. 1-6).
- [39]. Wireshark · Go Deep. Retrieved from <https://www.wireshark.org/>
- [40]. Jason Brownlee. (2014). An Introduction to Feature Selection. Retrieved September 16, 2018, from <https://machinelearningmastery.com/an-introduction-to-feature-selection/>
- [41]. Feature Selection methods with example (Variable selection methods). Retrieved from <https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/>
- [42]. Saeys, Y., Abeel, T., & Van de Peer, Y. (2008, September). Robust feature selection using ensemble feature selection techniques. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 313-325). Springer, Berlin, Heidelberg.
- [43]. Priyadarsini, R. P., Valarmathi, M. L., & Sivakumari, S. (2011). Gain ratio based feature selection method for privacy preservation. ICTACT Journal on soft computing, 1(4), 201-205.

- [44]. Aditya Mishra. (2018). Metrics to Evaluate your Machine Learning Algorithm. Retrieved October 16, 2018, from <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234>
- [45]. Jason Brownlee. (2016). Logistic Regression for Machine Learning. Retrieved September 18, 2018, from <https://machinelearningmastery.com/logistic-regression-for-machine-learning/>
- [46]. YOU CANalytics-Gradient Descent for Logistic Regression Simplified - Step by Step Visual Guide – YOU CANalytics-.Retrieved from <http://ucanalytics.com/blogs/gradient-descent-logistic-regression-simplified-step-step-visual-guide/>
- [47]. scikit-learn: machine learning in Python — scikit-learn 0.20.3 documentation. Retrieved from <https://scikit-learn.org/stable/index.html>
- [48]. D.2. tshark: Terminal-based Wireshark.Retrieved from https://www.wireshark.org/docs/wsug_html_chunked/AppToolstshark.html