OPEN SOURCE IDS FOR A RESOURE CONSTRAINED SET-UP

BY

ROSE WAMBUI WANJOHI

I132/0873/2013

MICHAEL GITAU MBUGUAH

I132/0889/2013

A PROJECT SUBMITTED TO THE SCHOOL OF INFORMATICS AND INNOVATIVE SYSTEMS IN PARTIAL FULFILLMENT OF THE AWARD OF BACHELORS DEGREE IN COMPUTER SECURITY AND FORENSICS AT JARAMOGI OGINGA ODINGA UNIVERSITY OF SCIENCE AND TECHNOLOGY

DECEMBER 6, 2016

DECLARATION

We declare that this project is our original work and has not been presented before for an award of a diploma or conferment of a degree in any other university or institution.

Signature Date

…… ……

ROSE WAMBUI WANJOHI

REG. No. I132/0873/2013

Signature Date

…… ……

MICHAEL MBUGUA GITAU

REG. No. I132/0889/2013

This research project has been submitted for my approval as the university supervisor

Signature Date

…… ……

MR. CASTRO YOGA

SUPERVISOR

SCHOOL OF INFORMATICS AND INNOVATIVE SYSTEMS

DEDICATION

This project is dedicated to our parents, Mr. Peter Wanjohi & Mrs. Winnie Wanjohi, Mr. Maina Waiganjo & Mrs. Mary Waiganjo for love, sacrifice, commitment and support that made the success of our university education.

ACKNOWLEDGEMENT

First of all, we thank the Almighty God for giving us this opportunity and for His protection throughout the university education period.

Our other acknowledgment goes to the School of Informatics and Innovative Systems department for instilling me with the knowledge and the experience which has endured me with the necessary skills as far as Information security and IT field are concerned. Thanks to out project supervisor Mr. Castro Yoga, for taking the time to guide us through our project which contributed to the success of this project.

Sincere thanks to our parents, Mr. Peter Wanjohi & Mrs. Winnie Wanjohi, Mr. Maina Waiganjo & Mrs. Mary Waiganjo. Thanks, you for your guidance during the period and during the preparation of this project was so helpful to me. Our siblings Barney, Joseph, Becky, Ruth and Elizabeth who have been a great inspiration in pursuit of academic endeavors, thank you so much.

Our friends, with special thanks to Grace Kirugumi and Peter Mwangi, who helped us throughout the university academic life and offering moral support. Thank you so much.

Thank you all and the Almighty God bless you.

ABSTRACT

Intrusion detection system(IDS) is devices or software that monitor the network systems for any malicious activity or policy violation. Firms and organizations which are resource constrained are not able to implement these network security systems since most of them are very expensive. There is always a question of whether the amount they will use to implement these systems is relatively equal or less than the amount used after a security breach has occurred. This research project was conducted in a virtualized environment which simulated a typical network. The two intrusion detection systems, bro and snort were implemented and tests run against them. The main objective of this project was to identify, implement and recommend a suitable open source IDS for a resource constrained environment/set-up. After running the test bro appeared to be the best IDS. It had a higher speed, less time to analyze traffic and detected more exploits than snort. Therefore we recommend that the resource constrained firms to implement the Bro intrusion detection system.

Keywords: open source, IDS, resource constrained

TABLE OF CONTENTS

LIST OF TABLES.

LIST OF FIGURES.

LIST OF ABBREVIATIONS.

IDS- Intrusion Detection System

TCP-Transport Control Protocol

UDP-Unified Dynamic Protocol

DHCP-Dynamic Host Control Protocol

DNS-Domain Name Server

HTTP-Hyper Text Transfer Protocol

SSH-Secure Shell

SSL-Secure Socket Layer

IP-Internet Protocol

SSDP-Simple Service Discovery Protocol

UPnP-Universal Plug and Play

OPEN SOURCE IDS FOR A RESOURCE CONSTRAINED SET-UP

CHAPTER 1: INTRODUCTION.

1.1 Background information

Recently malicious traffic on the internet has increased. This has caused loss of data and harm to computer systems. Companies, government agencies spend billions of dollars on computer security and still computer get attacked and loss of information. Studies show that Kenya for example loss around 4 billion per annum due to cybercrime. (Pfleeger, 2006)

The best security tools available e.g. firewalls, antivirus, intrusion detection system, intrusion prevention systems are expensive. Large companies may afford these tool but what about smaller companies and agencies? Smaller companies do not have the same economy and implementing expensive security tools is difficult. We find that these companies are resource constrained and therefore should use open source tools that are free. (Rodfoss,2011)

Open source community offers a wide range of these tools and have an advantage over the payment solutions in that they provide source codes which are customizable to one's environment and set-up. Bought security tools come with a preconfigured box where the users should set the network. One can interact with other open source users, conduct a research and tests to come up with the best results. But the questions that these smaller companies should ask, are all open source the same? Do they have the same functionality? And if not which is better than the other? (Security in computing 4th edition.,2006)

Intrusion detection system (IDS) has become a common way to ensure network security. They detect any intrusion or hostile traffic in a network to prevent loss and manipulation of information. Sourcefire company provide both open source and payment IDS. It provides open source IDS such as snort, Bro Network Security Monitor and Suricata. Snort and Bro have existed since 1998, while Suricata first stable version was released in July 2010.

1.2 Statement of the problem.

There exists a gap in smaller companies which are resource constrained and therefore cannot afford to buy IDS. Most of these companies end up with obscurity measure or just transfer the risks that arise due to cyber-attacks. This should not be the case since they are free available open source detection systems that these agencies should adopt. The question that arises is which of these open source these companies should adopt. By using certain parameters, we will compare the three available IDS to come up with the best IDS and have an ending solution to these problems.

1.3 Main objective

1. To identify, implement and recommend a suitable open source IDS for a resource constrained environment/set-up.

1.4 Specific objectives

1. To Identify and analyze open source IDS approaches suitable for resource constrained environment
2. To implement and deploy at least two IDS and test their performance.
3. Compare and contrast the deployed IDS systems with one another in relation with the results obtained in second objective so as to propose a suitable IDS for a resource constrained set-up.

1.5 Significance of the project.

Companies with low economy end up being the target of attack since they are not able to employ security mechanisms to prevent these cyber-attacks. This project seeks to solve these and help under resourced agencies also to have a chance to implement these security measures. This will enhance the efficiency of their networks, proper e-commerce channel since VoIP in protected and also it will reduce the cost of having to transfer the risks maybe to an insurance companies. The project will also ensure that the best IDS is implemented and therefore these companies can have to enjoy too.

1.6 Scope of the project.

This project was conducted in a virtualized environment which will simulated a typical network. VMware in this case was used where both the IDS were implemented for the test analysis.

1.7 Limitation

1. Virtual machines are a bit slower than the physical environment.
2. Virtualization will not simulate the exact physical network.

1.8 Assumptions

This project assumes that Virtual Machine will simulate a real-time network of a company, thus the project can work in a physical environment.

CHAPTER 2: LITERATURE REVIEW

2.1: Introduction

Since intrusion detection systems were invented in the middle of the 1980's, a lot of different intrusion detection systems have been developed, different methods to detect malicious traffic, different algorithms and other approaches as well. Intrusion detection systems use two different approaches to detect malicious traffic, which are

a) Signature based detection
b) Anomaly detection.

2.2: Signature based detection is when the intrusion detection system uses a database with signatures of known malicious traffic, and compare these sig-natures against the traffic and see if there are some matches. If these signatures match any traffic on the network, alerts are created.

Signature based rules are based on pattern matching, and with modern day systems pattern matching can be per-formed very quickly. This is very important for multi-gigabit IDS systems.

One can easily tweak signature based rulesets. Since signature based IDS only can detect malicious traffic with known signatures, malicious with not known signatures will not be discovered. (IDS, 2016)

The key advantage of signature detection is that signatures are easy to develop and understand if you know what network behavior you are trying to identify. The events generated by a signature based IDS can give you detailed information about what caused the alert.

Signature based IDS can detect so called 0-day attacks. And the more signatures there are in the database, the slower will the detection engine be. As well, signature based IDS will create many false positives since they are usually based on regular expressions and string matching. Since they are based on pattern matching, they don't work well against different variants of the attacks as well. (IDS,2016)

2.3: Anomaly detection would detect statistical anomalies in the network traffic. The idea behind anomaly detection is to create a" baseline" that defines what kind of traffic that are deemed

normal, while traffic that is outside this baseline are looked as malicious traffic and alerts are created.

Anomaly detection has the ability to detect 0-day attacks, if it falls out of the baseline that is set. It works very good against IRC based botnets and other malicious activity. It creates lower number of false positives than the signature based IDS, and anomaly based IDS is very scalable, due to its architecture and method of operation. There is no need for creating new signatures for every attack and variant. (IDS,2016)

The disadvantage is that the anomaly detection engine is not able to decode and process the network protocols being analyzed in order to understand its goal and the payload. This is computationally expensive. In addition, there is very difficult to defined anomaly based rules, as every protocol analyzed by the system must be defined, base-lined and tested for precise thresholds. Most network protocols are implemented in a different way by different operating systems. As well, custom protocols need to be analyzed, reverse engineered and require a lot of effort. Malicious activity which falls under normal usage pattern won't be detected by the anomaly engine. (Rodfoss,2011)

Anomaly based IDS is the most researched method of these two. The signature based IDS are all about creating signatures of malicious activity, but anomaly detection has the strength of detect 0-day attacks, and all other malicious activity if the baseline is optimal. There is not so much one can improve by the signature based IDS, since is only uses signatures. While anomaly based IDS can be configured to stop all kinds of attacks, especially new malware.

There are different methods within the signature based and anomaly based IDS that can be improved. The high amount of data on the high speed network, demands high packet processing.

2.4: Overview of Bro.

Bro was originally developed in 1994 by Vern Parson and was named in reference to George Orwell's Big Brother from his novel *Nineteen Eighty-Four*. UNIX history buffs and computer science majors may recognize Paxson as the original author of flex, the fast lexical analyzer. (Bro ids. 2016)

The Bro Network Security Monitor (Bro) is a network-based analysis framework. Bro's powerful analysis engine makes it adept at high-performance network monitoring, protocol analysis, and real-time application layer state information. This makes Bro a very good intrusion detection system (IDS) and network analysis framework.

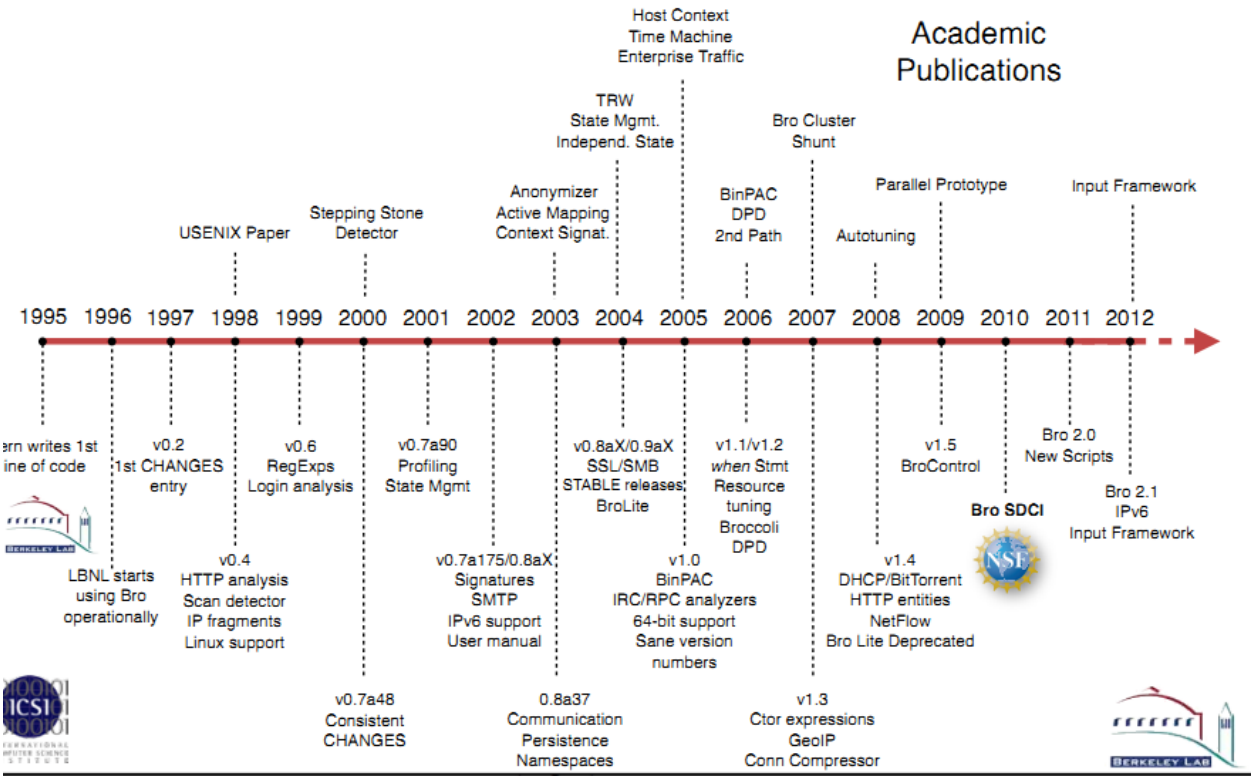

Figure 1. History of Bro

2.4: Overview of Snort

Snort is well-known name in the information security community as it was created in 1998 by Martin Roesch who is the founder of Sourcefire and who is still leading development of Snort. Snort is an open source network intrusion prevention and detection system (IDS/IPS) that combines the benefits of signature, protocol, and anomaly based inspection. It uses set of rules to check for hostile packets in the network and then generate alerts to the network administrator. The main aim of Snort, Suricata and any other IDS system is to effectively analyze all packets passing throw the network without any packet drops. (Snort ids,2016)

The program can also be used to detect probes or attacks, including, but not limited to, operating system fingerprinting attempts, common gateway interface, buffer overflows, server message block probes, and stealth port scans. There are three main modes in which Snort can be configured: sniffer, packet logger, and network intrusion detection. Sniffer modes read the network packets and display them on the console in a continuous stream. Packet logger mode logs the network packets to the disk. Network intrusion detection mode is the most complex mode.

Network Intrusion detection mode monitors network traffic and analyze it against a rule set defined by the user and then perform a specific action based on what has been identified. (snort ids,2016)

Components of Snort

Snort is logically divided into multiple components. These components work together to detect particular attacks and to generate output in a required format from the detection system. A Snort-based IDS consists of the following major

Components:

a. Packet Decoder

b. Preprocessors

c. Detection Engine

d. Logging and Alerting System

e. Output Modules

2.5: Overview of Suricata

Suricata is a rule-based Intrusion Detection/Prevention System (IDS/IPS) that takes advantage of externally developed rule sets to monitor sniffed network traffic and provide alerts when suspicious events take place. Like most IDS it is designed to fit within existing network security components. The initial release of Suricata runs on a Linux 2.6 platform and supports both inline and passive traffic monitoring configuration capable of handling multiple gigabit traffic levels. (IDS,2016) Suricata works as a multithreaded engine. According to its creators, the objective of the Suricata Project Phase 1 was to have a distributable and functional IDS/IPS engine. On January 1st, 2010 Suricata was made available for download. (Security in computing 4th edition, 2006).

2.6: Comparison of Three IDS (bro, snort & Suricata)

Table 1: IDS comparison

Parameter	**Bro**	**Snort**	**Suricata**
Contextual signatures	Yes	No	No
Flexible site customization	High	Medium	Low
High speed network capability	High	Medium	Medium
Large user community	No	Yes	No
Configuration GUI	No	Yes	Yes
Analysis GUI	A Few	A Lot	A Few
Installation /deployment	Hard	Easy	Easy
Operating System compatibility	Unix	Any	Unix

Table 1 IDS Comparison

CHAPTER 3: METHODOLOGY.

3.1: Introduction

When comparing Snort, Bro one have to decide what kind of information there are possible to compare. There are many things that could be compared, such as output logs, alarms, configuration, ruleset, how to set them up, and test environment. Setting up Snort, Bro were a time-consuming process. Each of them required a set of installed packages, which helps them in the process of detecting malicious activity, and logging information about them.

3.2: Requirements

1. Physical machine.

This will host the virtual machine. This machine should have the following requirements; RAM minimum 4 GB, speed 2.6 ghz and minimum of 2 processors. This is because VMware is heavy and require a lot of resources.

2. Virtualization environment. VMware workstation
3. Linux platform to install the IDS
4. Two other machine to form the network. (Debian and windows)

3.3: Test environment.

The test environment representation is shown in the figure below. The diagram represents a private LAN which represent a range of 192.168.20.0/24.

Bro and snort intrusion detection system were installed on PC1 which is supposed to monitor the network traffic within the LAN. The IDS should monitor the workstation and the webserver for any intrusion.

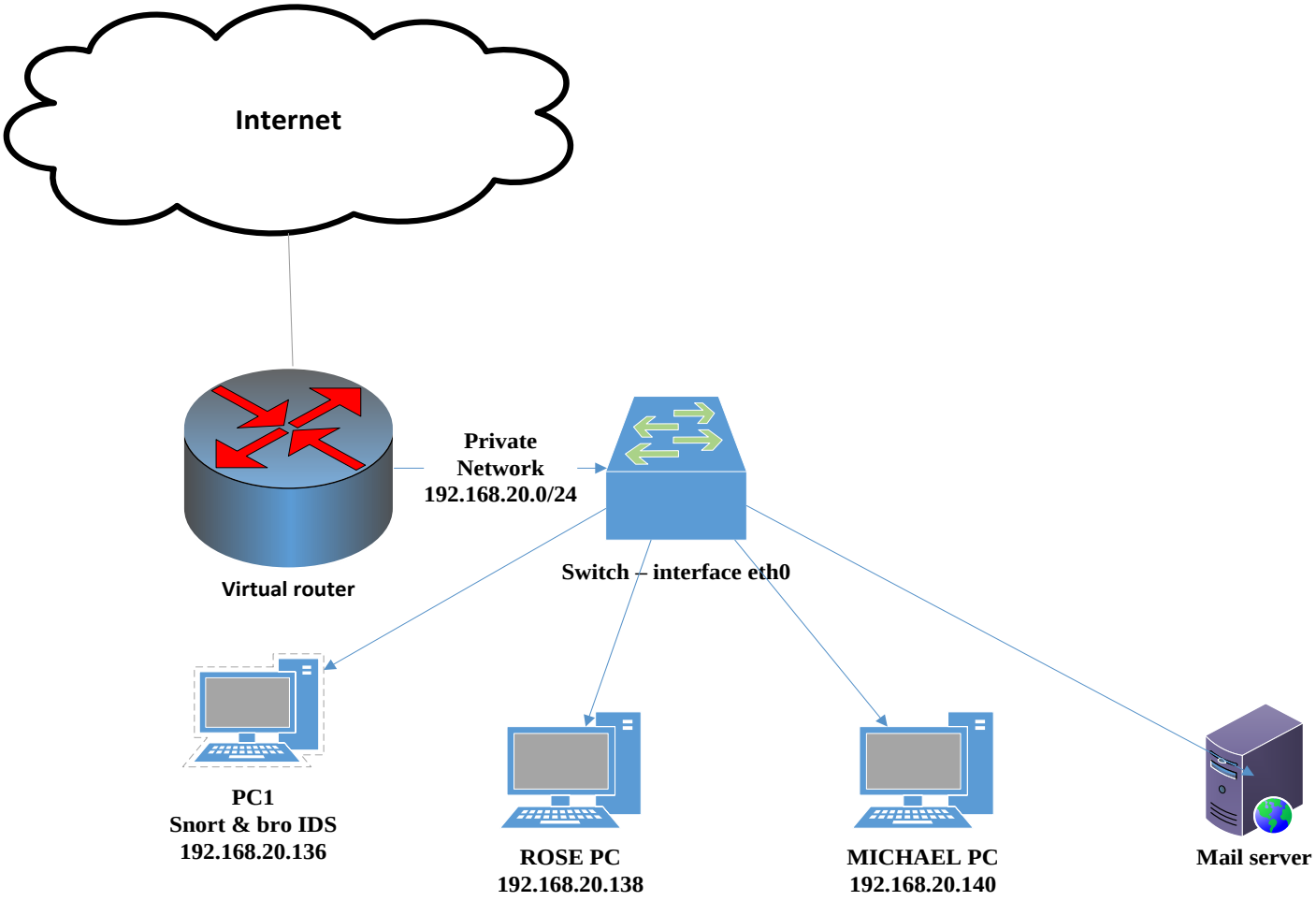

Figure 2. Test Environment

Project design.

This research project was conducted in a virtualized environment which simulated a real-time network. A network was formulated which consisted of 3 workers and one PC which the two IDS were installed. This was a controlled environment where traffic was induced. Two approaches were used, capturing a tcpdump file and using the metasploit framework.

Tcpdump: This a common packet analyzer that runs under the command line. It allows the user to display TCP/IP and other packets being transmitted or received over a network to which the computer is attached.

Metasploit framework. This a platform in Linux operating system which consist of well-known attacks and exploits. It helps attackers to created payloads which are used to attack the target machine.

The two approaches used are explained below.

3.4: APPROACH 1

3.4.1: Tcpdump file.

Tcpdump was used to capture traffic against the interface eth0 within the network. This was used so that the two IDS could be used to analyze the same traffic. Traffic was captured into a file which was run against bro and snort respectively. The time it takes for the two ids to capture the traffic is determined.

When the two-intrusion detection create alarms and logs about the traffic when run against the tcpdump file, one can find out what kind of alarms and logs that were triggered and compare these alarms. Tcpdump file of size of 36MBs was created which was set to capture up to 50000 packets. The file was then run against the two IDS.

3.5 APPROACH 2

3.5.1 Metasploit framework.

Metasploit framework was used to test bro and snort to determine which IDS will detect the exploits run against the 192.168.20.0/24. The Metasploit Framework contains some of the most well-known attacks, by running different exploits against the machine where Snort and bro are installed, it will create alarms based on the exploits that is run. As the alarms are based on the traffic from the exploits.

The computer 192.168.20.138 in the network was used to exploit the machine where bro and snort were installed (192.168.20.136). When the exploits were run in the attacking machine, tcpdump capture file was started in the receiving machine. This is to ensure that snort and bro analyze the same exploit traffic.

Running the metasploit framework.

- Search for open ports in PC1 through nmap.
- Create a database in postgres.
- Run the msfconsole and load the created database.
- Run the exploit against the target machine.

  (shown in the appendixes)

CHAPTER 4: FINDINGS AND ANALYSIS

4.1 APPROACH 1

Same traffic analysis using a tcpdump file

In the first approach the 2-intrusion system were run on a tcpdump file so as to analyze the same traffic. The file was set to capture 50000 packets which led to size of 36MBs

Time was taken of how long each of them took to analyze the tcpdump file and the results obtained are shown below.

- Bro used 1min .7 seconds
- Snort used 6min .6 seconds.

Therefore, calculating the speed of the two IDS:

$$\text{Speed} = \frac{\text{size of file in mbs}}{\text{Time taken in seconds}}$$

Bro speed = 36/ 67 = 0.573 mbs/s

Snort speed = 36/366 = 0.0984 mbs/s

This shows that bro is a faster intrusion detection system to analyze traffic and create alerts.

Analysis of the logs and alerts created.

After running the two IDS against the captured file some logs and alerts were created. These were used for comparing and contrasting the functionality of the two IDS.

Snort:

Snort was configured in full alert mode. When running snort against the tcpdump file created an alert log and snort log file were created. An example of the alert log created by snort is shown below.

```
[**] [1:1917:6] SCAN UPnP service discover attempt [**]
[Classification: Detection of a Network Scan] [Priority: 3]
12/02-18:20:07.739200 192.168.20.1:58343 -> 239.255.255.250:1900
UDP TTL:1 TOS:0x0 ID:950 IpLen:20 DgmLen:161
Len: 133

[**] [1:1917:6] SCAN UPnP service discover attempt [**]
[Classification: Detection of a Network Scan] [Priority: 3]
12/02-18:20:10.739631 192.168.20.1:58343 -> 239.255.255.250:1900
UDP TTL:1 TOS:0x0 ID:951 IpLen:20 DgmLen:161
Len: 133

[**] [1:1917:6] SCAN UPnP service discover attempt [**]
[Classification: Detection of a Network Scan] [Priority: 3]
12/02-18:20:13.741890 192.168.20.1:58343 -> 239.255.255.250:1900
UDP TTL:1 TOS:0x0 ID:952 IpLen:20 DgmLen:161
Len: 133
```

The information derived from the snort alert log file are:

- Snort ID
- Alarm message
- Classification
- Priority
- Timestamp
- Source and destination IP
- Source and destination port
- Protocols

Bro.

When bro was run against the tcpdump captured file it produces different logs depending on the activity involved. The logs produced in this case were: conn log, dhcp log, dns log, http log, packet filter log, ssh log, ssl log, weird log.

Conn log. This show the connection between all the hosts.

```
type   time           source IP      port    destination IP port
protocol type    interval
480682307.696409        192.168.20.148 59662   192.168.20.2    53
udp     dns      88
1480682312.523673       192.168.20.148 49888   117.18.237.29   80
tcp     -        0
1480682312.522916       192.168.20.148 49886   117.18.237.29   80
tcp     -        0
1480682309.121043       192.168.20.148 43798   52.34.245.108   443
tcp     ssl      200
1480682308.933116       192.168.20.148 55012   52.210.89.42    443
tcp     ssl      360
```

From this log the source and destination IP as well as the port can be retrieved. This log also gives the protocol involves during connection.

Dhcp log. This shows the machine connected within the network, their ip addresses and the mac addresses.

```
type     time             ip           count    assigned ip
mac address                      lease time
1480692952.026365       192.168.20.135 68      192.168.20.254 67
00:0c:29:71:6a:1d       192.168.20.135 1800.000000 $
1480693126.618629       192.168.20.148 68      192.168.20.254 67
00:0c:29:be:ca:90       192.168.20.148 1800.000000 $
```

```
1480693129.768879       192.168.20.136 68      192.168.20.254 67
00:0c:29: b4:68: aa     192.168.20.136 1800.000000 $
```

Dns log. This displays the source ip, the default gateway of that machine and the accessed site through that machine.

```
type   time      source IP        port  default g    port    protocol
count   site
480682305.784694      192.168.20.148 34385   192.168.20.2    53
udp     44722   tools.kali.org
1480682305.784683     192.168.20.148 34385   192.168.20.2    53
udp     59096   tools.kali.org
1480682305.784697     192.168.20.148 44324   192.168.20.2    53
udp     58056   www.kali.org
1480682305.784698     192.168.20.148 52477   192.168.20.2    53
udp     34429   www.offensive-security.com
```

.

Ssh log. Bro displays any ssh login within the network, the ip addreses and the version of ssh protocol used.

```
type    time         source ip       lport   dest IP        Rport
       version
480683673.558222       192.168.20.148  43358   192.168.20.136  22
list    SSH-2.0-OpenSSH_7.3p$
1480683787.730975      192.168.20.148  43941   192.168.20.136  22
edit    SSH-2.0-OpenSSH_7.3p$
1480683908.363852      192.168.20.148  33115   192.168.20.136  22
ping 192.168.20.148     SSH-$
```

Ssl log. Base SSL analysis script. This script logs information about the SSL/TLS handshaking and encryption establishment process.

```
1480682723.848786       Ciz3LO3fRqHeXHTouc      192.168.20.148 53706
52.10.239.169   443     TLSv12  TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
secp$
1480682724.101306       CDILXyYLWbqgXVUM8       192.168.20.148  53708
52.10.239.169   443     TLSv12  TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
secp$
1480692957.100052       CXxVrN3P7TXnIUtjL7      192.168.20.135  49610
157.56.77.140   443     TLSv12  TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
secp$
```

X509 log: The record type which contains the fields of the X.509 log.

```
type       time     cert.version    cert serial no.
     descrption
1480682315.408327       3           083BE056904246B1A1756AC95991C74A
CN=DigiCert Global Root CA,OU=www.digicert.com,O=Dig$
1480682316.279475       3           08F7AF7CB34B880721345ED45DA07670
CN=self-repair.mozilla.org,O=Mozilla Foundation,L=Mo$
1480682316.279475       3           0C79A944B08C11952092615FE26B1D83
CN=DigiCert SHA2 Extended Validation Server CA,OU=ww$
1480682316.517745       3           08F7AF7CB34B880721345ED45DA07670
CN=self-repair.mozilla.org,O=Mozilla Foundation,L=Mo$
1480682316.517745       3           0C79A944B08C11952092615FE26B1D83
```

4.2: APPROACH 2

4.2.1Metasploit framework

After exploits were made and a tcpdump file was captured. The file was run against bro and snort. This was to ensure the same exploit traffic was used. A file of 6MB was captured

Bro

Apart from the logs that were created in approach 1 bro had created a weird log which displayed some information about the payload that we run. The contents of the weird log are shown below.

```
type        time       gateway ip          port    attacker ip
port    name
1480787052.371935     192.168.20.1          51795 192.168.20.150   4444
     possible_split_routing      -     F     bro
1480787174.508683     192.168.20.1          51835 192.168.20.150   4444
     possible_split_routing      -     F     bro
1480787415.905890     192.168.20.1          51920 192.168.20.150   4444
     possible_split_routing      -     F     bro
1480787487.775882     192.168.20.1          51933 192.168.20.150   4444
     possible_split_routing      -     F     bro
1480788412.681238     fe80::9c45:d301:b66e:2038 143    ff02::16   0
     bad_ICMP_checksum     -     F     bro
1480788449.702899     fe80::20c:29ff:fe1d:6b55143 ff02::16   0
     bad_ICMP_checksum     -     F     bro
```

The payload created was a reverse tcp connection type and was set to attack the host machine (192.168. 20.1) the listening port was set to 4444 while the lhost was 192.168.20.150(worker3). All that information is displayed by bro.

Snort.

Snort created a snort log after the file was captured. The exploit involved creating a payload that was executed in machines within the network including the host machine of the virtual machines which is the default gateway. The contents of snort log did not have anything related to the exploit created.

```
Host:239.255.255.250:1900
ST:urn:schemas-upnp-org:device:InternetGatewayDevice:1
Man:"ssdp:discover"
MX:3

NT:upnp:rootdevice
NTS:ssdp:byebye
Location:http://192.168.20.1:2869/upnphost/udhisapi.dll?content=uuid:8
a706dbc-f34b-4d25-968b-6fb6f2a6c9cc
USN:uuid:8a706dbc-f34b-4d25-968b-6fb6f2a6c9cc::upnp:rootdevice
Cache-Control:max-age=1800
Server:Microsoft-Windows-NT/5.1 UPnP/1.0 UPnP-Device-Host/1.0
OPT:"http://schemas.upnp.org/upnp/1/0/"; ns=01
01-NLS:76a840d27c770234d0349c3b15c137f4
```

Unlike bro, snort does not display the content of the payload.

4.3: COMPARISION:

4.3.1: Time analysis

Figure 3. Time Analysis

From the pie chart above bro used to shortest time to analyze the file.

4.3.2: Speed of analyzing traffic

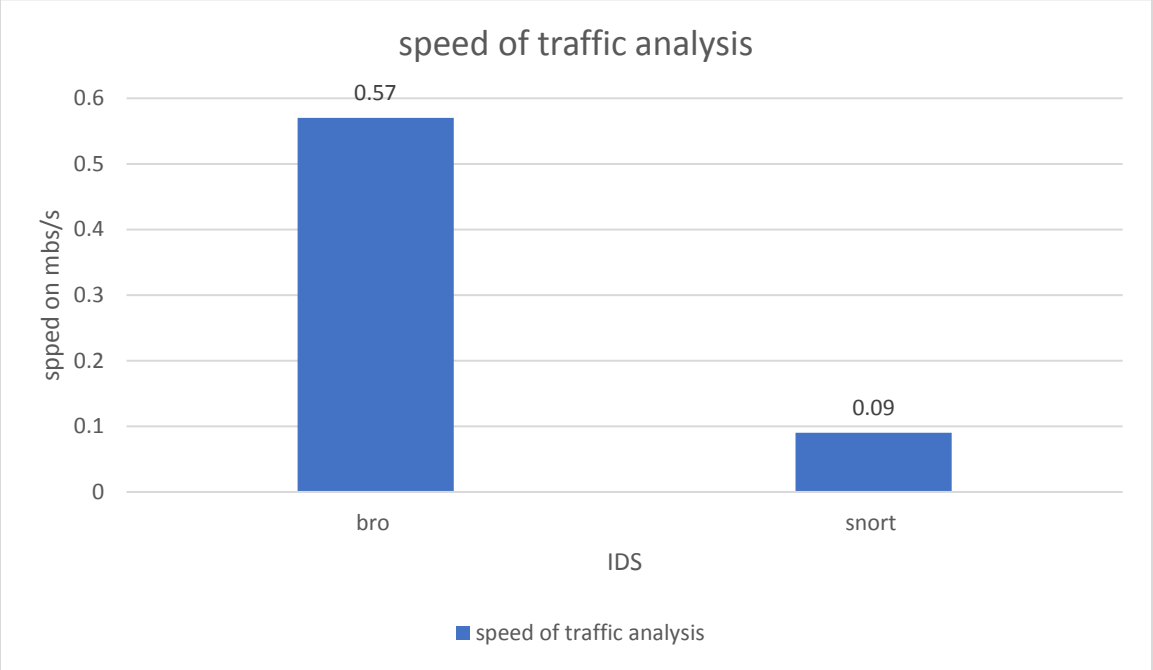

Figure 4 speed for analyzing

4.3.3: Detections.

In the overall implementation snort detected

- Network scan -UPnP scan
- Ssdp discover

While bro detected

- ICMP ping attack
- SSH login,
- Possible split-routing

4.3.4: Table analysis.

Aspect	**bro**	**snort**
Speed	High	medium
Exploits detection	High	low
Implementation method	cluster	standalone
Protocol analysis	DHCP, DNS, SSL, SSH, HTTP	UDP/TCP only
Geo-location capability	Locates IP address with the geographical location	N/A
Installation	difficult	easier
User-friendly	Less friendly	More friendly

Table 2 Table Analysis

CHAPTER FIVE: CONCLUSION AND RECOMMENDATIONS

5.1: Introduction

During the installation of Snort and bro, snort was easier to install than bro. Bro encountered some problems during the installation process. Different pack-ages that were either missing, packages that were not compatible, or it turned out to be wrong version installed.

5.2 Conclusion

Approach 1 and 2.

In approach 1 which involved time take to analyze the same bro took the shortest time to run the tcpdump file of 36mbs. This translates to a higher speed than snort and therefore bro can be used in gigabytes of network.

In the second approach bro made detection of the exploits run while snort was not able to capture that exploit. This shows that bro is able to detect intrusion more than snort.

5.3: Recommendations

In relation to this project we recommend that resource constrained firms to adopt open source method of enhancing intrusion detection system. They should not be a target of attack due to the fact that they cannot afford to buy security tools. These resources constrained set-up should adopt bro intrusion detection system since it can be used as intrusion and also a network monitoring tool.

5.4: Future works

Further research can be done in the following area.

- Software-hardware integration by implementing bro in a raspberry pi and have a customized bro intrusion detection system.

REFERENCES

Charles P. Pfleeger and Shari Lawrence Pfleeger, *(*2006). *Security in computing 4th edition,* Prentice Hall

Installation guide of snort and bro: http: ://blog.securitymonks.com/2016/10/17/three-little-Idsips-engines-build-their-open-source-solutions. 2015

Intrusion detection system. Retrieved from:*http://www.intrusiondetectionsystem.org/, 2016.*

Intrusion detection system. Retrieved from: *www.wikipedia.org/wiki/intrusion detection system,2016.*

Penetration Testing Software, Pen Testing Security | Metasploit. Retrieved from: *www.metasploit.org, 2016.*

Metasploit project. Retrieved from: *www.wikipedia.org/wiki/Metasploit Project, 2016.*

Rodfoss, J. (2011). comparison of open source nids *(1st ed., pp. 5-10).* Norway: Oslo university. retrieved from *https://www.duo.uio.no/bitstream/10852/8951/1/Rodfoss.pdf*

Snort ids.: Retrieved from: *www.snortid.com*, (2016).

Snort ids. Retrieved from: *www.wikipedia.org/wiki/snort (software)*, 2016.

Snort manual.: Retrieved from*: http://www.scribd.com/doc/6777057/Snort-Manual,2016*

Suricata ids: *http://www.openinfosecfoundation.org*, 2016.

Suricata ids. URL: www.wikipedia.org/wiki/suricata (software), 2016.

APPENDIXES

Bro installation.

In this project, we install bro in a Linux platform machine. For bro to installed some dependencies have to be installed.

Required dependencies.

Package	**Description**
BIND8 headers and libraries	BIND (Berkeley Internet Name Domain) is an implementation of the Domain Name System (DNS) protocols. It is an open DNS software.
Bison	Bison is a general-purpose parser generator that converts an annotated context-free grammar into an LALR (1) or GLR parser for that grammar.
Flex	Flex is a tool for generating scanners. A scanner, sometimes called a tokenizer, is a program which recognizes lexical patterns in text.
Libpcap	It is the packet capture library. libpcap is a system-independent interface for user-level packet capture. libpcap provides a portable framework for low-level network monitorin
LibGeoIP	Ability to determine the location of IP addresses. It connects the ip addresses to the specific geographical location
Swig	software development tool that connects programs written in C and C++ with a variety of high-level programming languages. It connects bro to its scripting languages.
zlib	Libz is a compression library. It is used for decompressing HTTP bodies by the HTTP analyzer, and for compressed Bro-to-Bro communication.
Libmagic	Add ability to determine file types, as with the ftp analyzer.

Table 3 Required Dependencies

Installing bro dependencies.

While connected to the internet the bro dependencies are installed as show in the figure below. Apt-get install command is used to install package in Linux platform.

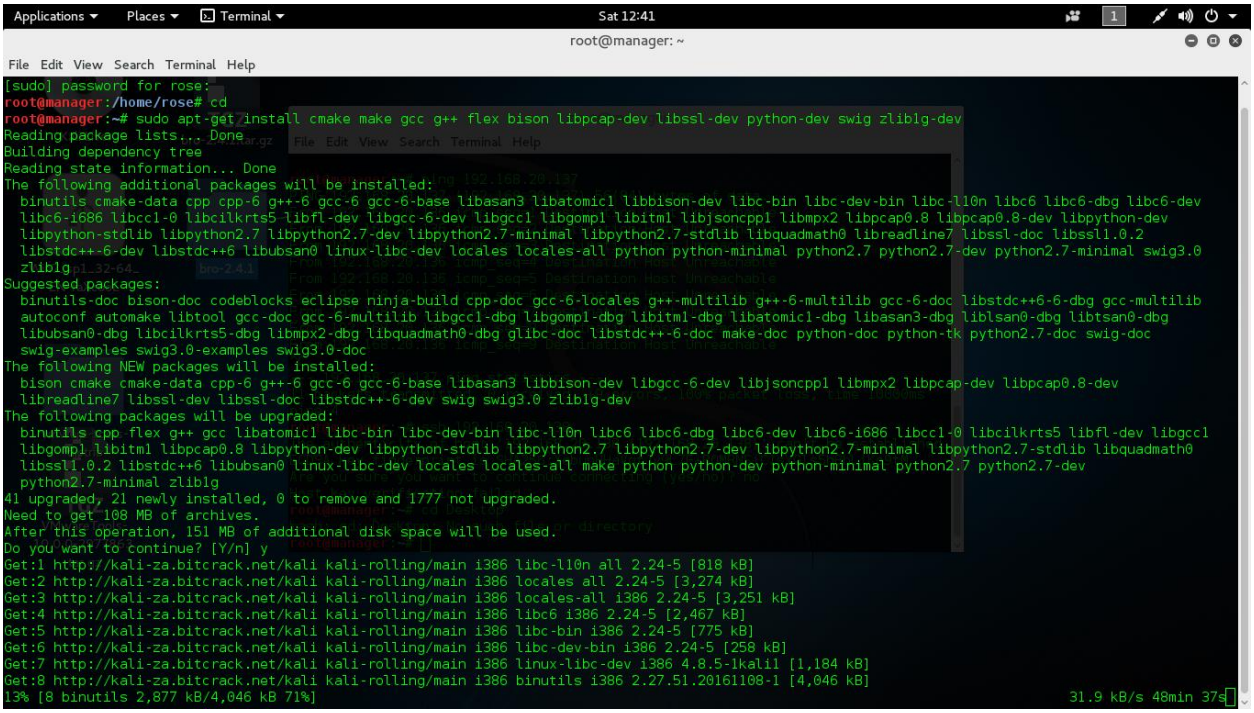

Figure 5. Installation of bro dependencies

Installing bro from the source

The bro network is installed where the packages are fetched from bro.org

git clone --recursive git://git.bro.org/bro

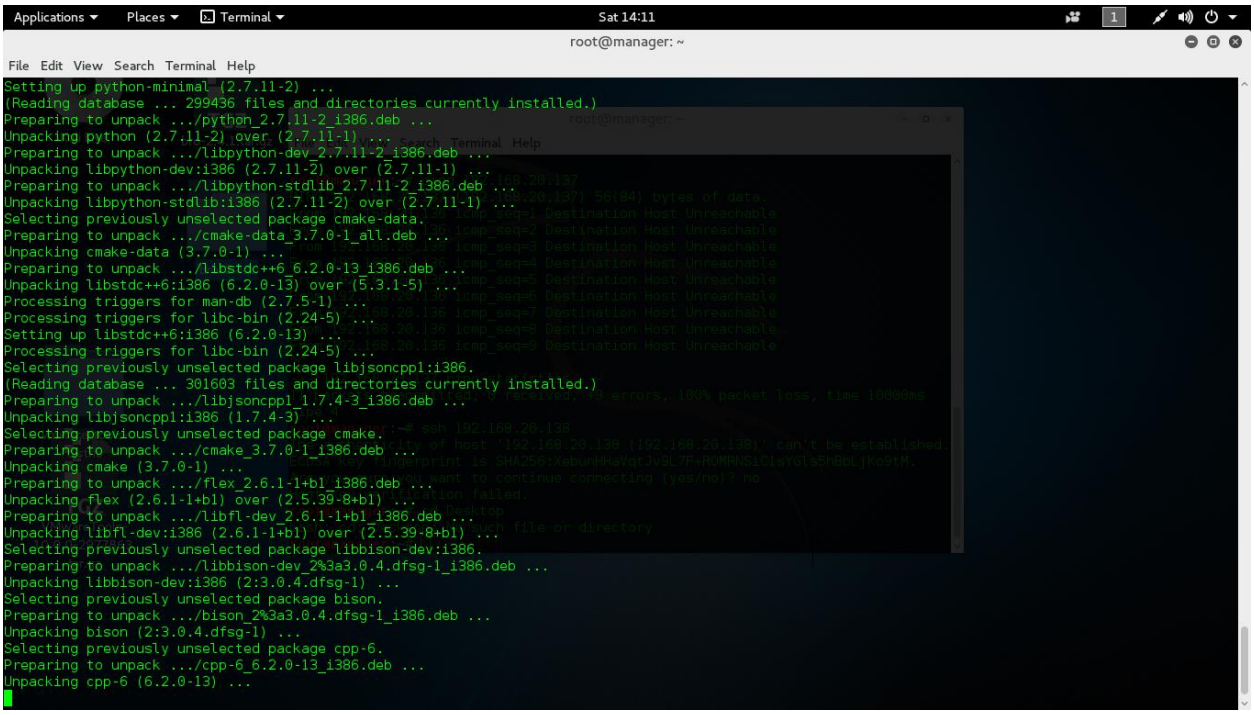

Figure 6. Installation of Bro from the source

After the bro package is installed. Enter in the /root/bro directory to install the package.

. /configure

make

make install

. /configure: tells you whether you are quite ready to build the application. It checks that all bro dependencies are in place.

make: builds (compiles) the source code. Compiler compiles the code, but, most of the times, the code cannot stand alone, it requires external libraries (usually provided by ubuntu packages) to be installed.

make install: moves the application files to the appropriate system directories. This has to be done

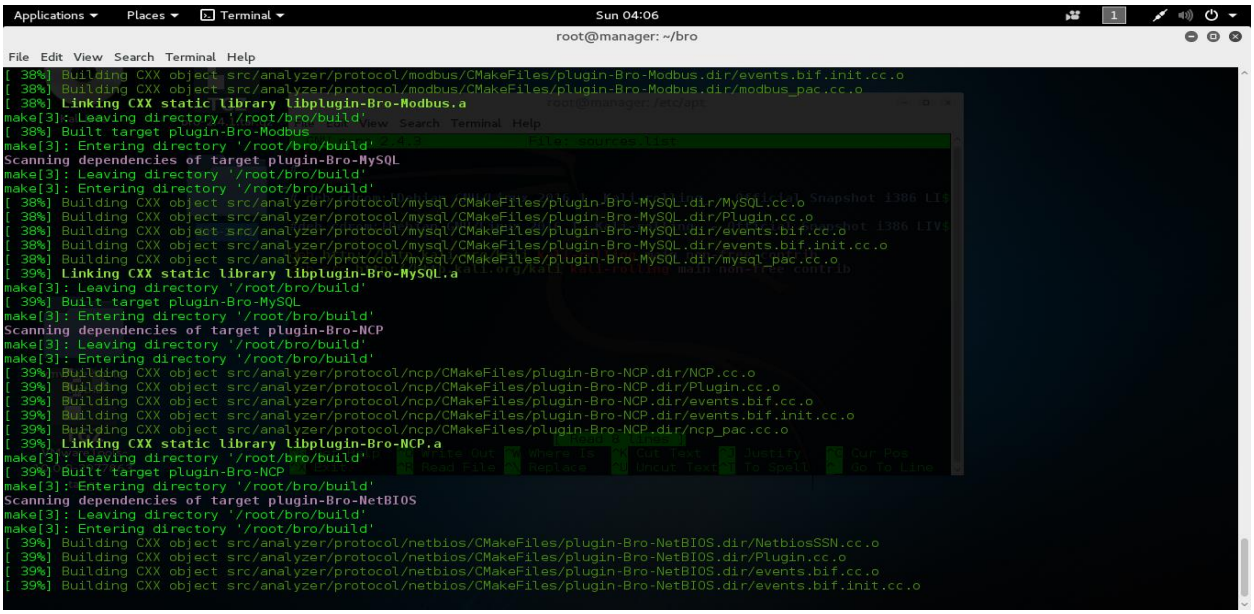

Figure 7. Make and Make install for Bro

creating ssh keys.

Ssh key generating.

Ssh: Secure shell is a protocol operate network services securely over a network that remote access computers in a network through the TCP port 22. After generating the keys the keys is copied to the remote machines (woker and jones) so that they will authenticate connections from the manager machine.

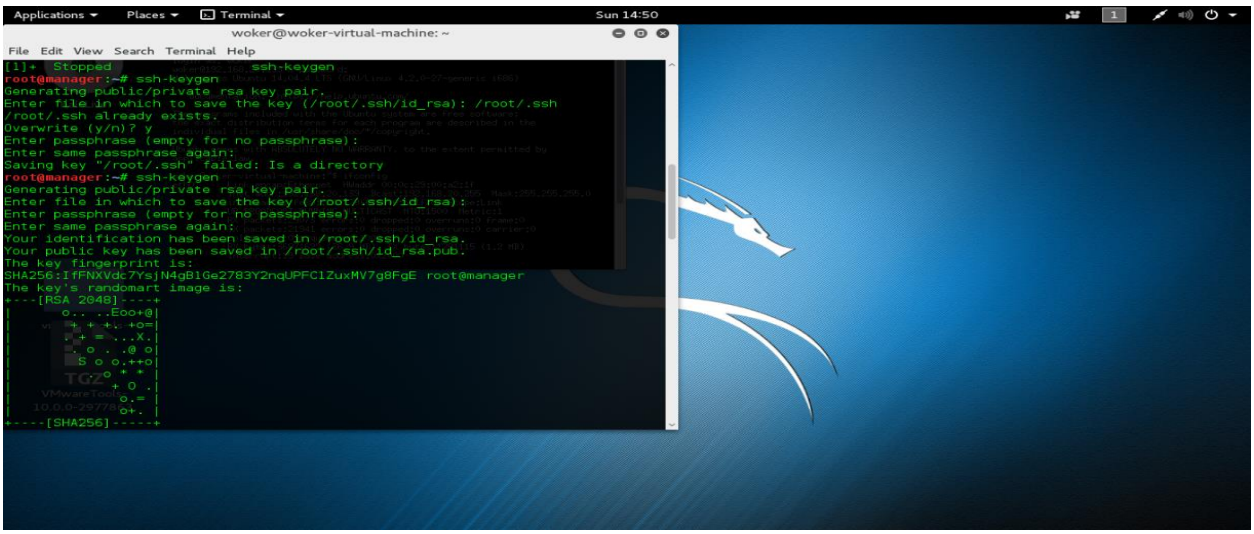

Figure 8 SSH Keys generation

Copying the ssh keys to the remote host

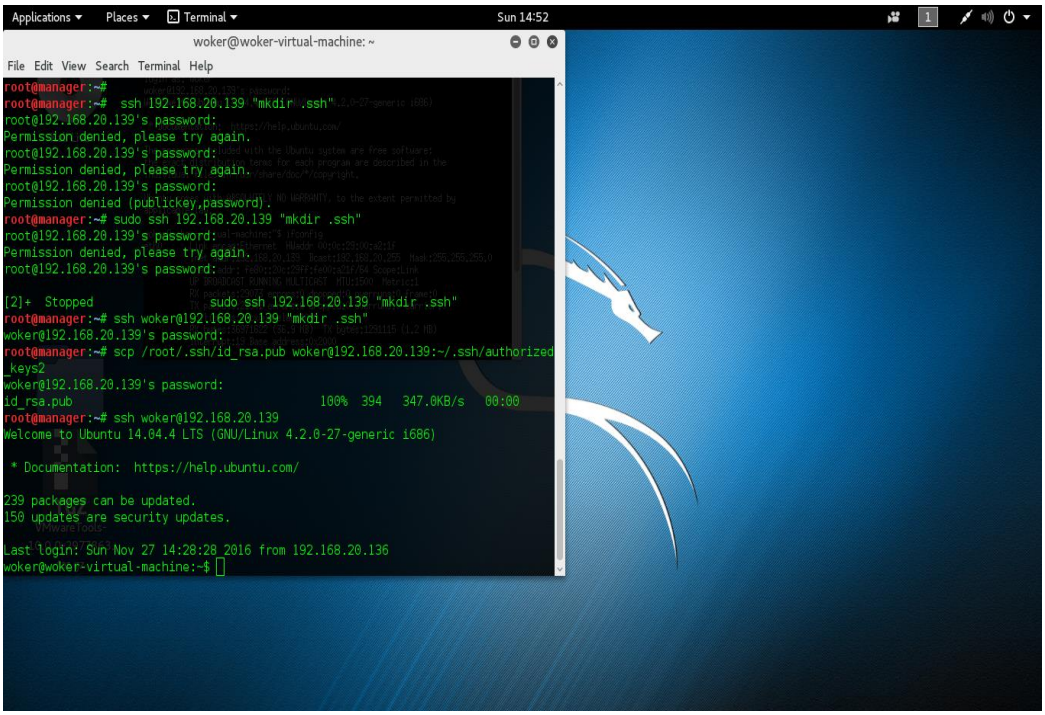

Figure 9. Copying the ssh keys to the remote host

Building a bro cluster: Bro IDS is built on a cluster mode so that it can monitor remote connection. The node file is configured and changed to suite the environment. Where we have on

manager, proxy, and two worker's node machine and ip addresses.

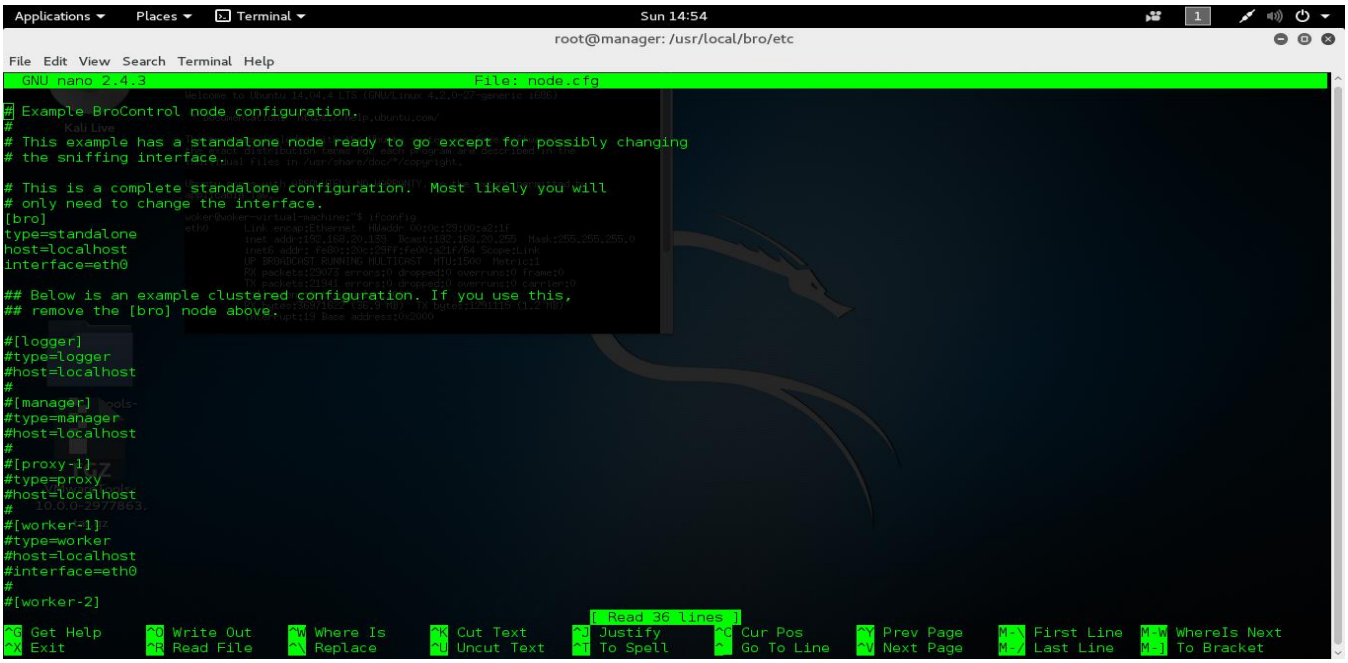

Figure 10 Binding Bro Cluster

Snort installation.

Required dependencies

libdnet	ibdnet provides a simplified, portable interface to several low-level networking routines.
pcre	The PCRE library is a set of functions that implement regular expression pattern matching using the same syntax and semantics as Perl 5.

Table 4 Required dependencies for snort

Snort also require a libpcap dependency that is listed in bro.

Installing snort

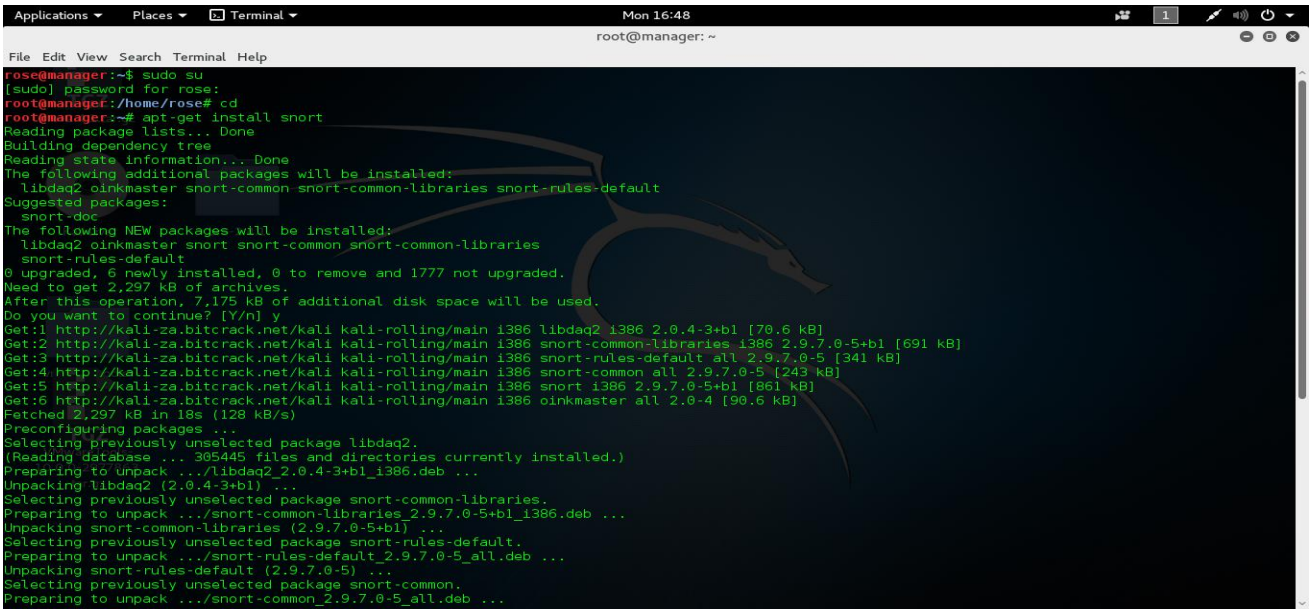

Figure 11 .Snort Installation

Running snort.

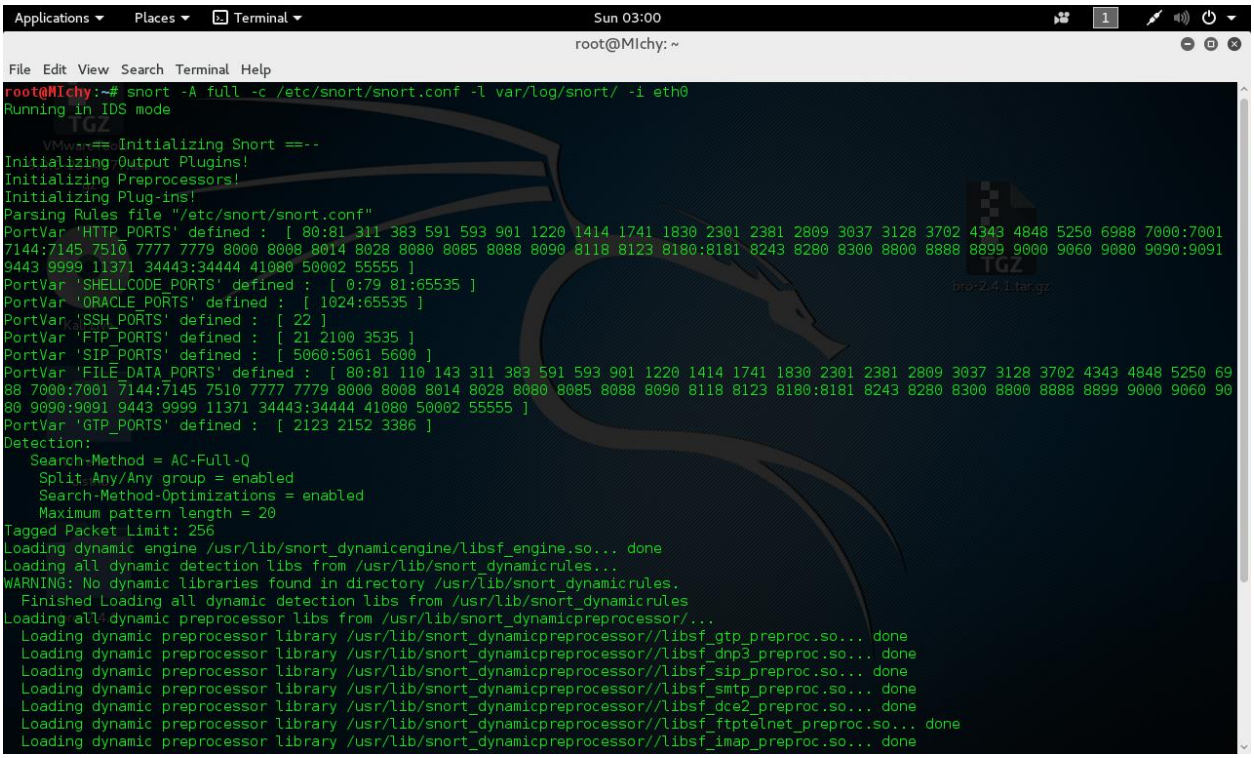

Figure 12: how to run snort

Tcpdump file capture.

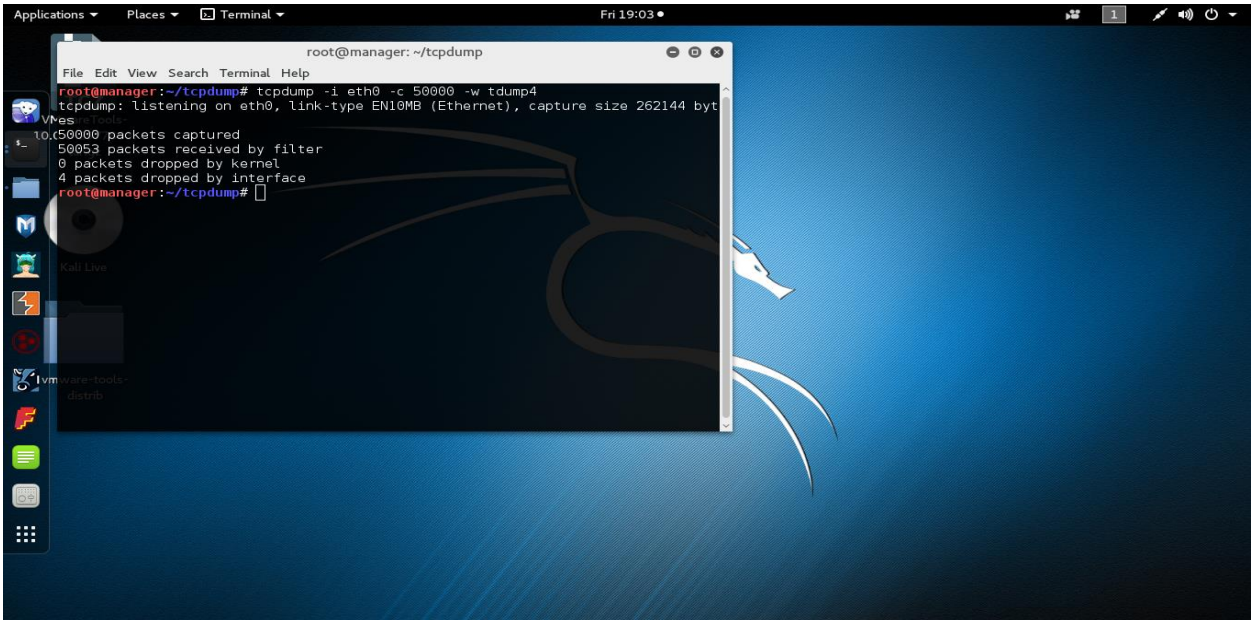

Figure 13 .TCPDUMP File Capturing

1. Running the tcpdump file against bro.

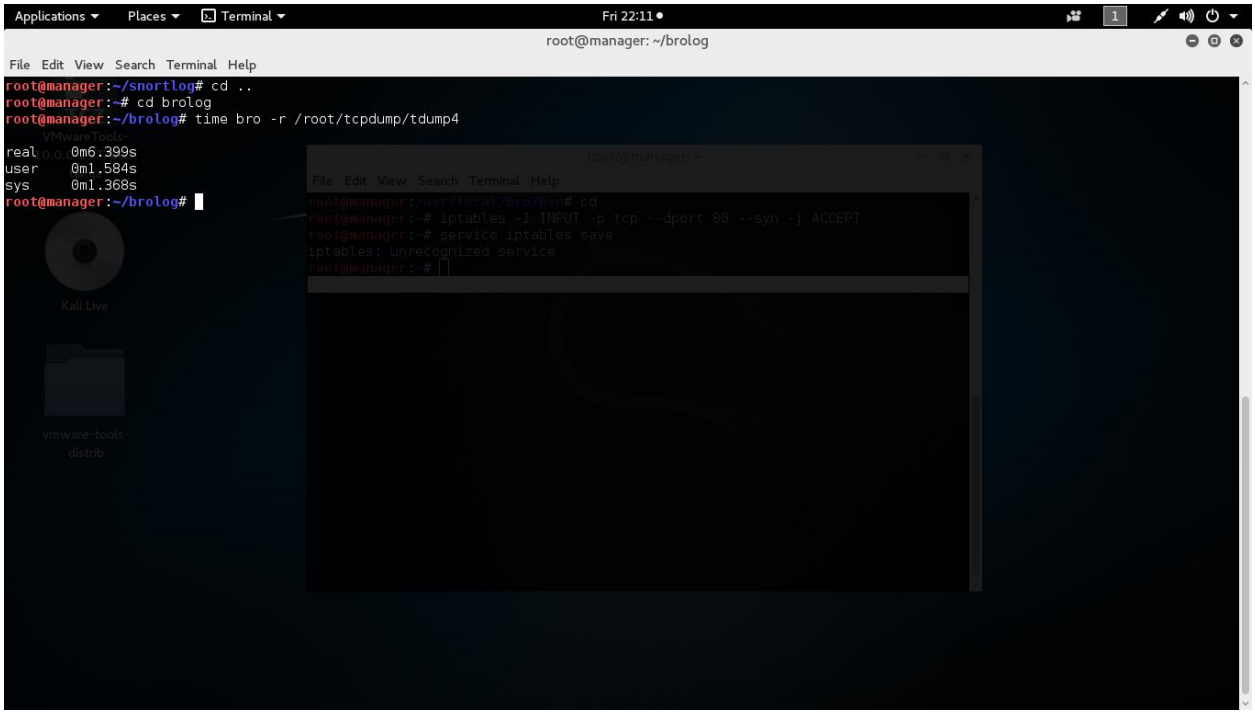

Figure 14. Running tcpdump against Bro

2. Running the tcpdump file against snort.

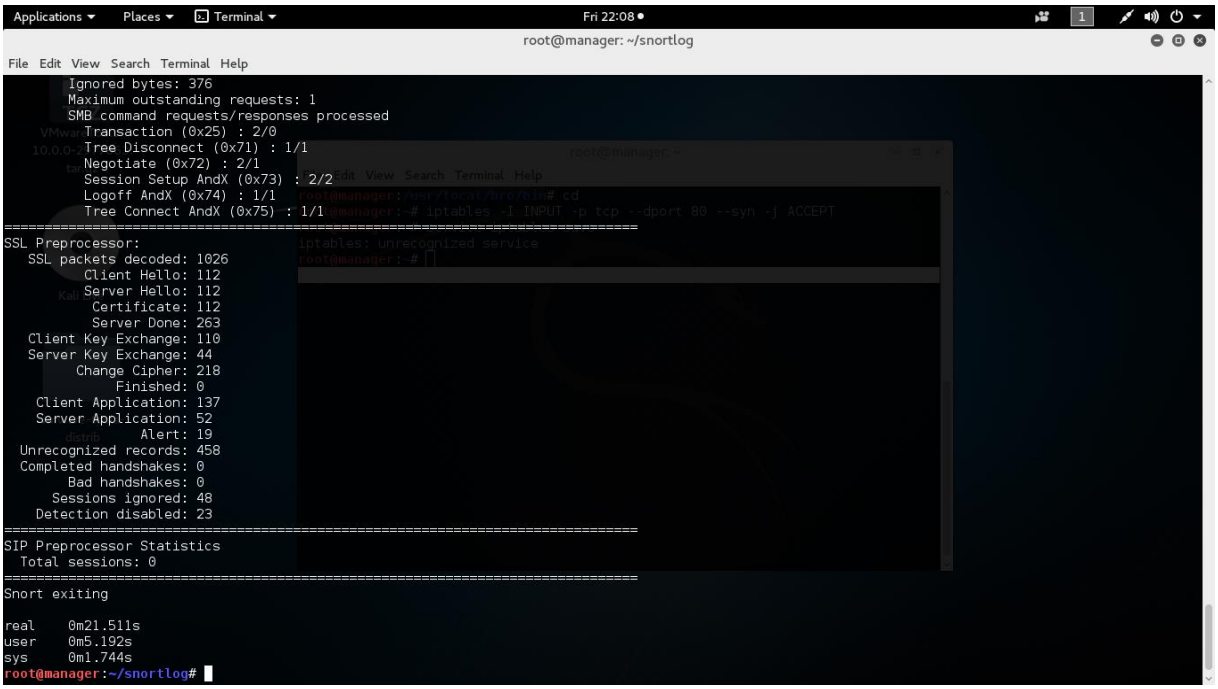

Figure 15 . Running tcpdump against Bro

Metasploit framework.

Creating a postgres database

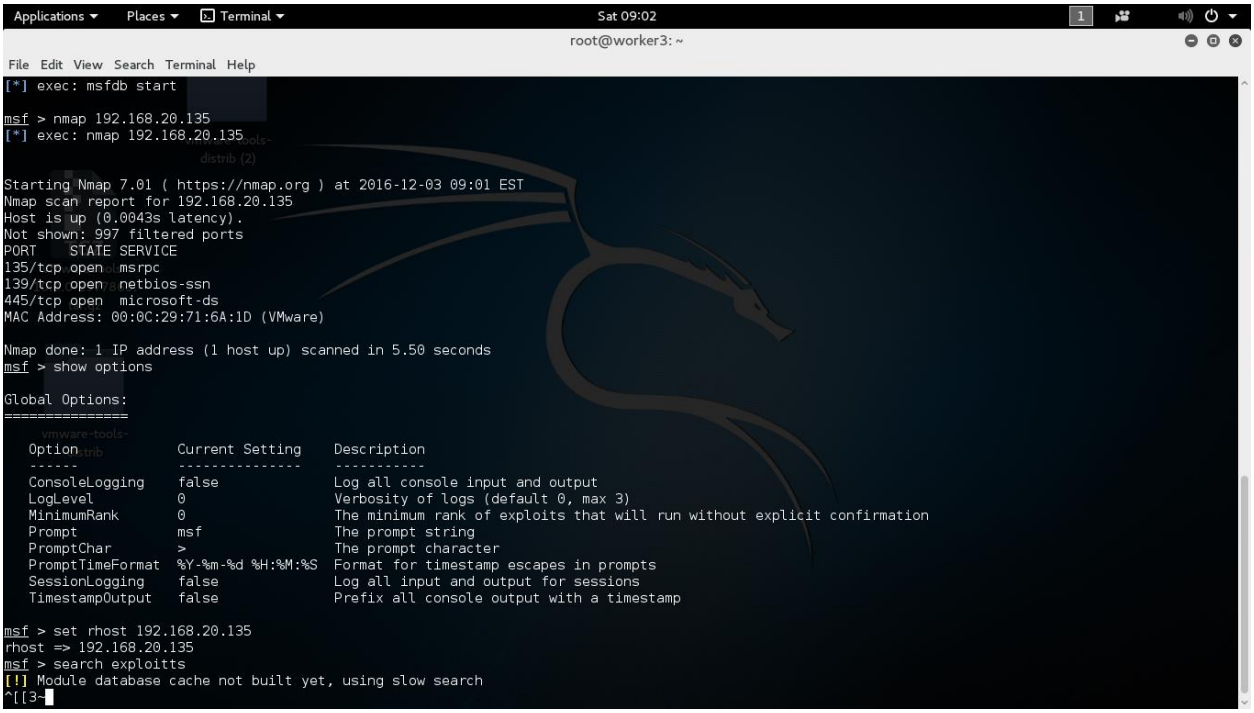

Figure 16 .Creating a postgres database with Metasploit

Creating a payload.

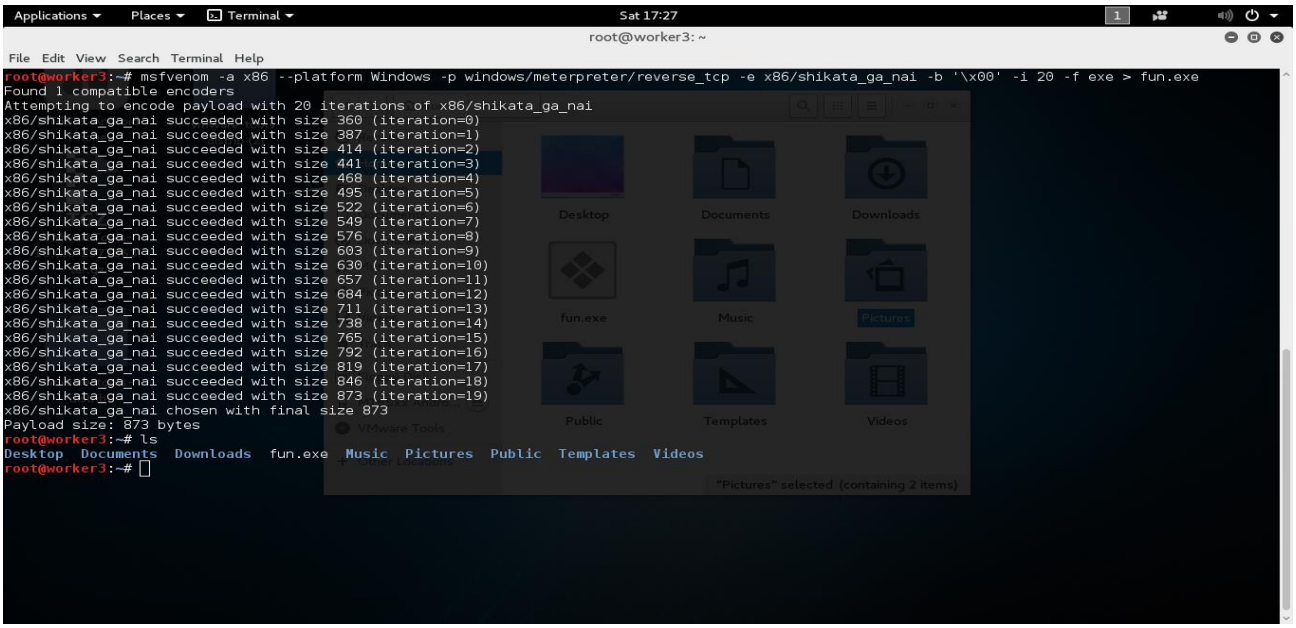

Figure 17 Creating a Payload

Executing a payload to the target machine.

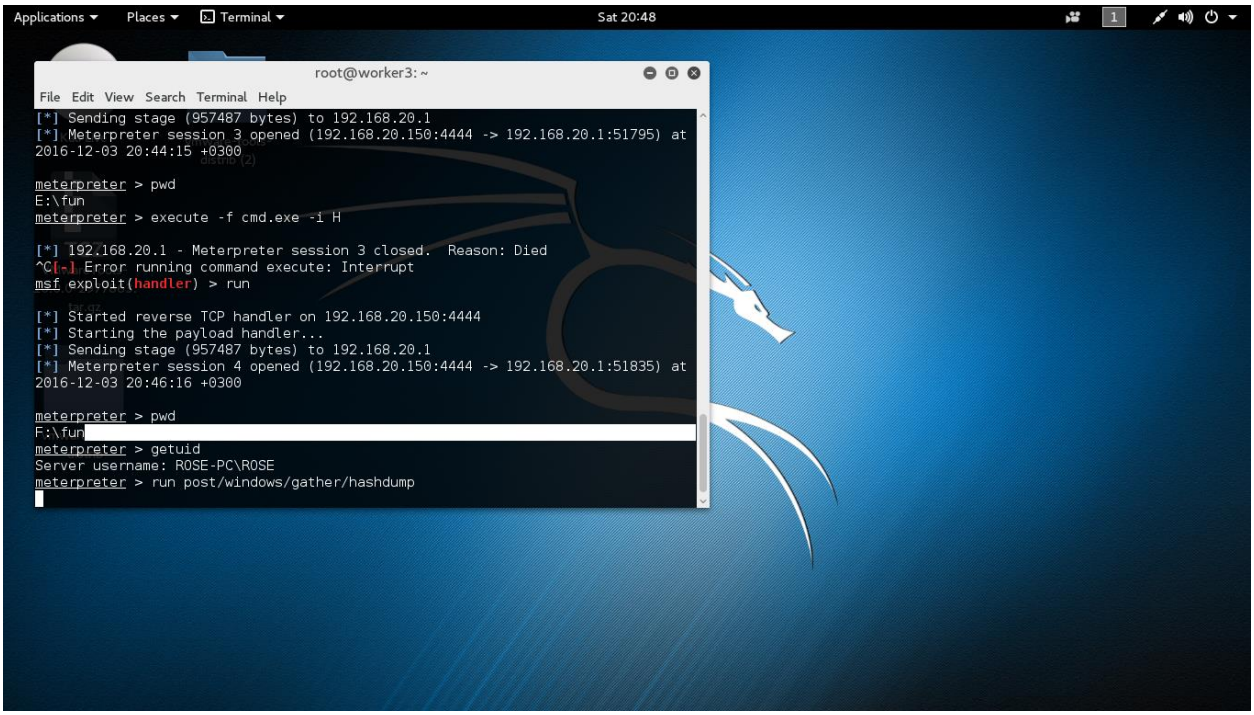

Figure 18 Executing a payload to the target machine.