International Journal of Computer Applications (0975 – 8887)
Volume 141 – No.7, May 2016

Implementation of Intrusion Detection System and Traffic Analysis – A Case of a Linux Platform

Isaac Bansah
Information Technology Department, Methodist University College, Accra, Ghana

Tonny Montana Adegboyega
Information Technology Department, Methodist University College, Ghana

Stephen Brako Oti
Information Technology Department, Methodist University College, Accra, Ghana

ABSTRACT
The security of a Computer network cannot be compromised in any form as it would actually defeat the exact purpose for which the network exists; to provide connectivity between nodes that would allow exchange of information or resources. It also goes a long way to ensure absolute security for nodes in communication, information at source, in transit or flight and finally at the destination. Security implementations may vary according to network designs but it is essentially suppose to provide Authentication, Data integrity, Confidentiality, Access control and Availability. This paper looks at the implementation of an Intrusion Detection System on a Linux operating systems and analyzing the traffic, threats and vulnerabilities with a configured Firewall

General terms
Access privileges, Packets, Operating System

Keywords
Intrusion Detection System, Linux, Traffic Analysis, Network Security

1. INTRODUCTION
Computerization started in the west as support service technology through it being an appropriate tool and now almost an engine of growth to national economies. Irrespective of the era one finds itself, we have grown to accept it as part of daily lives. Its usefulness or applications manifests itself through all areas of lives including medicine, commerce, communication, manufacturing etc.

Computer networking has made it possible for several communicating nodes to be connected to each other. It therefore becomes an infrastructure through which nodes exchange information and share resources among themselves. These information or resources exchanged on this network now becomes vulnerable to illegitimate or unauthorized users because of the sensitivity of the some of the information exchanged through this medium. This calls for an appropriate implementation of security measures to ensure safety of information exchanged through the network. Availability of operating System variants which provide the basic interface between the computers and the users have provided different responses threats and vulnerabilities to systems. Approach to this must be holistic and not any ad hoc practice just to curb a particular situation.

2. INTRUSION DETECTION SYSTEMS (IDS)
Even though the area of Intrusion Detection Systems (IDS) is relatively young, there has been numerous amount of work done to bring to light the impact it has on the system. The explosion of Internet connectivity and the expansion of the network influence this today. This necessitates the need to keep the network safe from unwarranted attacks hence the importance of IDSs. Most of these works highlight the limitations of the current intrusion detection systems in use.

There are a lot of intrusion detection systems currently in use but all these can be classified into one standard model according to [1]. This model they referred to as the Common Intrusion Detection Framework (CIDF). It generally defines the fundamental components that make up an intrusion detection system. These components are the generators, analysis engines, storage mechanisms and countermeasures; E-boxes, A-boxes, D-boxes and C-boxes respectively.

The E-box serves as the source of information to the system. It provides the system with the necessary information to help in the detection of a malicious infiltration. It does not necessarily matter if the information in itself is an intrusion or not.

The duty of the A-box is to generally analyze the information gathered by the E-box for any intrusion. This may be done using statistical anomaly detection, graph analysis etc.

The information gathered by the E-box and analyzed by the A-box needs to be stored and made available to the systems analyst. This is done by the D-box. It provides the means for the storage of important security information.

The purpose of the C-box is to react to any intrusion by raising an alarm or shutting down the system completely to avoid further attack.

It is not outside the capability of a smart attacker to compromise the various components of the intrusion detection system (IDS) itself hence rendering it unreliable.

Any successful attack on the E-box can lead to the event generator not been able to provide the network with the appropriate and reliable information and raw data needed by the network. It can also prevent it from decoding packets off the network.

The analysis of the information provided by the E-box varies from one IDS to another. Some IDSs rely on sophisticated analytical approach for security information. This can lead to an attacker who is familiar with this situation to attack the system. This report is in no way suggesting that the sophisticated analytical approach should not be employed since making it simplistic can also expose the system to constant attack.

A poorly implemented storage technique can be very dangerous in IDS. It makes it possible for the data collected by the systems to be altered by the attacker. It also allows the activities of a smart attacker not to be recorded.

Even though the C-box mostly serves as a countermeasure mechanism, its importance cannot be over-looked. This countermeasure can be employed by an experience attacker to attack the network the IDS is employed to protect and go undetected, [1].

The basic purpose of an intrusion detection system is to be able to distinguish between a bad traffic and a good one that enters the system. This characteristic leads to the following:

1. Undetected bad traffic
2. Detected bad traffic
3. Good traffic the system identifies as bad (false alarm)
4. Good traffic the system sees as good.

2.1 Undetected bad traffic

This is the most dangerous thing that can happen to the system. A situation like this implies that the intrusion detection system is not doing the work it is employed to do. This may be brought about by one of so many reasons. It could be due to lack of adequate intrusion detection mechanisms in place or the inability of the system to identify new forms of attack. It will be unreasonable to ask of an intrusion detection system to identify all forms of attack but it should be able to identify as many attacks as possible.

2.2 Detected bad traffic

This is the best thing that can happen to an intrusion detection system; the ability to identify a bad traffic with accuracy and speed. The more accurate the system is the more it can be trusted. The system will be of no use if it cannot detect bad traffic with high accuracy and speed.

2.3 Good traffic the system identifies as bad (false alarm)

This situation generally leads to a lot of time- wasting by a systems analyst. It is mostly referred to as false alarm or false positive. This problem comes about when the system identifies traffic as bad and raises an alarm when in the actual sense the traffic is a good one. This situation needs to be investigated by the analyst hence wasting a lot of time. In doing this investigation valuable time and resources needed to investigate real attacks is wasted. This basically leads to lose of trust in the intrusion detection system.

2.4 Good traffic the system sees as good.

This is the case where the system identifies a good traffic for what it is. It is necessary the system identifies this and allows it into the network. If this is not so, the purpose of the network will be undermined.

Considering the importance of the availability of IDS it becomes an obvious target of attack by an experience attacker so that it is rendered ineffective. An attacker capitalizes on any vulnerability in the IDS and mostly causes it to give wrong signals. The accuracy of the system when compromised leads to a lot 'false positives' i.e. the system identifying an intrusion when there is none.

The system can also be compromised such that it produces a lot of 'false negatives', i.e. the system falling to identify an attack. This is referred to as the completeness of the system, [1].

The importance of the IDS within a security system is such that, it is necessary its function is very accurate and the information it offers is devoid of inconsistencies since it can lead to a lot of insecurities. The forensic information gathered by a flawed system can also be misleading.

Most of these problems are brought about when intrusion detection techniques are poorly implemented.

The design of an intrusion detection system basically depends on whether it is to be used on a single host or a network of computers. There are two main kinds namely Network-based intrusion detection systems (NIDs) and Host-based intrusion detection systems (HIDs).

Network intrusion detection systems basically analyze network packets to detect an attack whilst host-based intrusion detection systems analyze audit logs collected by the operating system on activities of users and system applications.

2.5 Host Based IDS

Host based intrusion detection systems use audit logs collected by the operating system on activities of users and system applications to analyze and detect intrusion. It processes audit data and raises an alarm based on deviation from past user activities. This is only determined after the actual attack has already occurred.

The normal behavior of the system is normally defined in the security policy of the computer system and the user settings set by the system administrator. Any deviation from the set rules is deemed intrusive and logged.

According to [2] computer systems '…are likely to remain unsecured for some time to come' due to ' continually shortened software life cycle times' resulting in 'poor designs or inadequate testing'.

[3] States the following as some of the methods of audit trail analysis:

- In depth off-line (after-the-fact) analysis of audit data
- Real time testing of audit data, so that an immediate protective response is possible
- Subsequent analysis of audit data for damage assessment

Computer usage patterns captured by the host system can also be used for security purposes. The trend analysis of user behavior patterns can be used as defense against inside attacks, [3]. Keystroke dynamics according to [3] can also be used to verify the identity of a user.

Modern host-based intrusion detection systems still use audit logs for detection but are now more automated to stand the test of time. Updated and sophisticated responsive detection techniques are now used. Host-based intrusion detection systems monitor system, event and security logs on Window NT and syslog in UNIX environments. When any of these files change, the HIDS compares the new log entry with attack signatures to see if there is a match. If there is, the system responds with administrator alerts and other calls to action.

Other technologies have also been incorporated into modern HIDS. They can now check key system files and executables via checksums at regular intervals for unexpected changes.

They are able to alert the system administrator when some specific ports are accessed thereby introducing some fundamental characteristics of network-based intrusion detection systems.

2.6 Types of Network-Based IDS

A typical Network-based intrusion detection system is Snort. It is a lightweight intrusion detection system.

2.6.1 SNORT

Snort is a lightweight intrusion detection system capable of performing real time traffic analysis and packet logging on IP networks. It is a signature-based intrusion detection system developed by [4]. It uses rules and pre-processors to analyze traffic. It is not resource intensive and the source code is rather small. Snort is mostly used on small to medium sized networks, single hosts, or on segments of a large network.

Snort rules basically offer a flexible way of creating signatures to analyze single packets whereas the pre-processor codes provide the possibility of analyzing data that cannot be analyzed by rules alone, [5].

The presence of pre-processors in the Snort network intrusion detection system allows it to perform various tasks such as TCP stream reassembly, IP de-fragmentation, web traffic normalization; ports can detection and many more.

It is easily configurable and flexible making it possible for users to create their own signatures and also alter its fundamental functionalities by the use of plug-ins.

2.6.2 Snort Overview

Snort is able to compare network traffic with set signatures called rules, which are stored in a database. It can also be used to dump sniffed traffic to the screen.

Its architecture is focused on performance, simplicity and flexibility. Snort has three primary sub-systems according to [4]. These are packet decoder, detection engine and logging and alerting sub-system.

The packet decoder is built around the layers of the protocol stack present in the data-link and TCP/IP protocol definition. Any action in the decoder imposes order on the packet data by overlapping data structures on the network traffic. These actions are organized through the protocol stack. Speed is the point of emphasis here. The decoder basically sets pointers into the packet data for later analysis by the detection engine

The Snort detection rule has two parts namely the Chain Header and Chain Options. They are rules that have been condensed to a list of common attributes Chain Header and the detection modifier options in the Chain Option. These forms what is referred to as the detection engine.

Even though Snort rules are simple to write, they are nonetheless very powerful in detecting a lot of attacks and mere suspicious network traffic.

Snort uses three base actions when there is detection. These are pass, log, or alert. Pass rules drop the packet whereas log rules write the packet to the logging routine chosen at run time and the alert rule produces event notification according to the method specified by the user at the command line.

The choice of what intrusion detection system to use basically depends on what it is supposed to do. Considering the strengths and weaknesses of both host-based and network-based systems, it is very important to combine the features of both. It would be appropriate to take advantage of the capabilities of both but the main concern is figuring out where to use each type and how to integrate the data, [6].

3. DETECTION TECHNIQUES

Detecting intrusions requires either knowledge of possible intrusions or knowledge of the known and expected system behavior, [7].

Intrusion detection techniques can generally be grouped into two according to the data they use for the detection. These are Misuse detection and Anomaly detection, [8].

3.1 Misuse/Signature Detection

This intrusion detection technique employs the services of known attack patterns and matching them with the current system behavior. It is also referred to as signature detection. Any deviation from the known signature is deemed to be malicious and dangerous to the system. This approach works really well in reducing a lot of 'false-positive' error provided the attack matches a pre-determined signature pattern. It has a major shortfall in not been able to identify new forms of attack. As long as the pattern of the attack does not match any known attack signature, it is allowed into the system hence increasing the amount of 'false-negative'. This can be very dangerous since the IDS will not be doing what it is meant to do. It also has a problem with fragmentation. IP datagram are sometimes broken down into smaller bits to augment transmission rates and later reassembled at the final point. The fragmented data is mostly not detected as malicious by the intrusion detection system. They are allowed to get past the system in most cases hence handing the intruder the advantage,[9].

3.2 Anomaly Detection

This technique is also sometimes referred to as protocol analysis detection technique. It has the ability to analyze packet flows. This helps to identify deviations in the general acceptable Internet rules of communication. It is comparable to the immune system, which allows 'self' and opposes 'non-self'. This idea of 'self' and 'non-self' was used by [10] to detect patterns of activity in software processes. This they did by building a database of patterns of system call usage and then performing direct pattern search on subsequent data to detect deviations. There point is that intrusions are normally caused by system calls, which do not match programmed patterns. When the system identifies any deviation from the specified standards then the packet is assumed to be malicious. Some of the attacks this technique can detect are the Buffer Overflow attack and the FTP Bounce Attack.

3.3 Accidental Association

Network Autonomy has led to definition of network reach and coverage. But the proliferation of wireless technology has caused overlaps of network coverage areas. When users find themselves in an intersection of a couple wireless networks, one is capable to access more than one network and that particular position with the right credentials.
A legitimate user can easily render his node for that matter the entire network he is legitimately subscribed to very vulnerable especially if the network is a fixed one. When that legitimate user happens to fall within a coverage area of a wireless network, an illegitimate user of the wireless network could then have access to the fixed network user and finally use him and an access to the fixed network.
This was espoused by [11] as he also indicated this could amount to a security breach in that proprietary company.

3.4 Malicious Association
Hackers associating themselves to the intended target destination gets them a step closer to accomplishing their task. There are several ways which this achieved. It makes mapping activity much easier.

According to [11], hackers or illegitimate users tend to configure their terminals or laptops as 'soft Access points' which allows them access to a company's network not through its Access point but as an access point itself.

And the Access Point as a layer two device would naturally evade all security implementations at the layer three level such as Authentication of users. After having access, hackers then do exhaustive search for passwords to help get full access credentials.

4. NON-TRADITIONAL NETWORKS
According to [11] vulnerabilities also extends to non-traditional networks having appendages of Bluetooth and other wireless linked devices such as handheld PDAs and printers are very vulnerable to cracking. They devices that are easily ignored by administrators as security implementations are concentrated on the traditional networks.

4.1 Identity Theft (MAC Spoofing)
Mac addressing is an address facility available at the link layer that uniquely identifies each hardware at that level. It also provides a means of identification with unique privilege for network access credentials. Nevertheless, it also provides an avenue for hackers to explore as network intrusion can also be possible at the level. It is done by sniffing on network traffic and with an exhaustive search of possible MAC addresses to gain access rights to the network. Mac address filtering is usually engaged to regulate access entry.

According to [11], hackers tend to employ the services of other applications and software with sniffing capabilities which then gives the cracker an arbitrary MAC address of a computer, which looks like the appropriate one for the hacker to accomplish its task.

4.2 Trojan Attacks
Trojans offer another option by which hackers attack computer systems. It manifests in various forms and offers different forms or functionality approaches. It is mostly executed by having access to a client computer remotely and most of the cases compromising the targeted computer. The most commonly used ones are the Sub seven and Netbus approaches. In executing the Netbus approach, the Netbus is primarily made up of the a server file and a Client file. The server file is normally sent to the target client computer and observes a connection from the target client computer. When a connection is made the Trojan is then installed on the target and data can then be accessed and target compromised by possibly modifying, creating or deleting accessed data.
The second approach, the subseven is mostly used for targets on windows platforms. It comes with three file; Server editor, a server and a client. The server file is initially sent to the target client machine as the case of the Netbus. The client file is used to establish a connection to the server and also for control purposes. After a successful connection to the target computer the editor server is used to modify the Trojan during deployment.
According to [12], after successful establishment of connection to the target machine, communication is done through high-order TCP ports, though still detectable despite being configured by users.

4.3 Packet Sniffing
Packet sniffing is an exercise that involves hackers ensures that they by some means strategically attached to the target network mostly at the link layer undetected. It could be through access point or by virtualization once they remain passive. Such strategic positioning allows these hackers to receive all broadcast frames on the network. At this point all they require is the employment of an appropriate application to interpret the frames in to an understandable form.
An example is such application is the Wireshark which can be put to good use by Network administrators to analyze and troubleshoot their networks. On the same vain it can be put to malicious use by hackers by using it to interpret intercepted frames of data for their use. Data collected by sniffing may include sensitive information including network topologies, routes, protocols, access credentials and any data that could shared during a broadcast.
Sniffing does a lot in inhibit successful data communication between two nodes. The possibility of accessing the data being transmitted violates confidentiality. There is a possibility of the integrity of the data being violated as well if especially the a possibility of altering the data being transmitted.
Sniffing is only executed in a mode where the hacker has his Ethernet card physically attached to the infrastructure but manage to remain passive and undetected. This makes all nodes visible to the network but not necessarily the destinations of the frames being sent. [12]

4.4 Key Logging Attacks
Confidentiality, the need to keep meaning of data being transmitted to a particular destination just the source and destination, brought about the need to encrypt data before transmission.
Encryption has proven to be successful and has evaded the hackers penchant to access data illegally.
Key logging attacks comes as an option to get the information ahead of it being encrypted. It is executed by gathering information from the keyboard as you type before it gets encrypted. The name of this tool is called key logger. It involves recording of typed key strokes which might include data like passwords, letters, login credentials and other typed sensitive information. There is also the possibilities of capturing screens at intervals. [12]

5. NETWORK COMMUNICATION
The basic purpose of the TCP is to provide a reliable and securable data transfer between a host computer and a destination computer (RFC 793). The following features make TCP a very reliable protocol:

5.1 Basic Data Transfer
TCP is able to transfer data continuously in each direction between users by putting the data into segments for transmission through the Internet system. It also determines when to block and forward data at their own time.

It is very important to know that the data submitted to the TCP is transmitted. This is catered for in the TCP protocol. To ensure this, the sending user indicates that it is sent to the receiving user. This causes the TCP to immediately forward and deliver data up to a point to the receiver (RFC 793).

5.2 Reliability
Data transferred to a receiving user can possibly be damaged, lost, delivered out of order or duplicated. It is necessary these anomalies are all corrected or reduced as much as possible.

Using sequence numbers and acknowledgement messages does this. TCP can provide a sending node with delivery information about packets transmitted to a destination node. [13] suggests that data transmitted should be assumed to be stream of bytes or octets and should be delivered to the destination in the same order as they were transmitted from the source. Damaged data is checked by adding a checksum to the segments transmitted checking it at the destination and discarding the damaged one.

5.3 Multiplexing

Connections to TCP allow multiplexing between TCP and other application processes. The various streams of data flow are independent from each other. It is also possible to operate half duplex communication over a full duplex. It should be noted that TCP and its interface know nothing of the structure of application data records or other units.

5.4 TCP Flow Control

[13] referred to flow control as “the process of controlling the transmission of data units over a network or set of interconnected networks (Internet)”. The fundamental purpose of flow control is the need to restrict excessive flow of data or packet from one connection to the other, which could eventually overwhelm the system, [14]. It ensures that the sending end of the system reduces its speed when the need arises.

5.5 TCP Connections

TCP uses what is referred to as a three-way handshake to establish connection as shown in the figure below. The first part of the handshake has the SYN bit set, the response has SYN and ACK bits set and the final part has just the ACK bit set. Hypothetically a three-way handshake seems unnecessary but it should be noted the TCP operates over an unreliable IP service so one of the elements of the handshake can get lost or duplicated.

The three-way handshake serves the following purposes:

- It ensures that both called and calling parties are prepared to have a call set up before any data is exchanged.
- It ensures that both parties agree on initial sequence numbers (x and y in the diagram below) for the two streams of data to flow from A to B and B to A over the established connection. X and y are chosen at random.

Closing a connection is also carried out with a three-way handshake to ensure a connection is not closed with data in transit. Here a FIN segment is used to signal closing a connection.

5.6 TCPdump

TCPdump is a UNIX tool used to sniff network packets for analysis. Its windows version is referred to as Windump. Data gathered by TCPdump is in most cases less coherent but mostly becomes coherent and clear ones the user is familiar with the use of this tool, [15]. It is also text-based hence easy to run on a telnet connection. Even though TCPdump does not have the ability to analyze traffic on a network, it is able to present the user with the ability to gather an enormous amount of traffic data that can be analyzed using other tools like ethereal or snort. Because capturing packets require access to devices accessible to root-only, after the installation of TCPdump most operating systems require root access to run.

The behavior of TCPdump can sometime be very irritating or annoying depending on the kind of command issued. The default behavior is for the tool to display or read all traffic on a network when the command tcpdump is issued. To avoid this behavior so many command lines are available.

TCPdump is designed complete with a filter language, which represents the fields in an IP datagram that need attention if TCP records are to be dumped for instant.

It is also possible to locate the file in which a particular filter is stored.

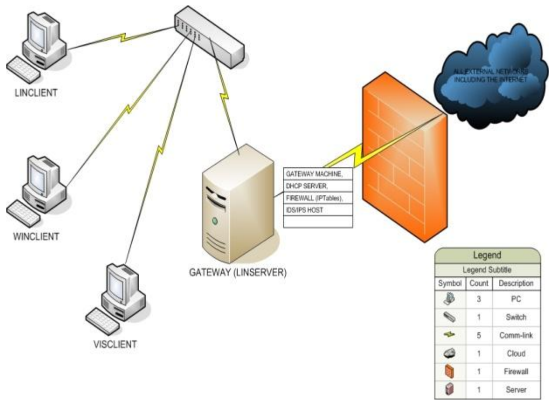


Figure 1: Research Network Design

6. CONFIGURATION OF THE /ETC/RC.D/INIT.D/FIREWALL SCRIPT FILE FOR THE GATEWAY SERVER

There are some services listed in the files of the firewall rules that are not required by the Gateway Server by default can be commented out with ”#" at the beginning of the line. Services that are required by the Gateway Servers but are not turned on by default can be turned ON by taking off the "#" at the beginning of their lines.

The firewall script files was created as follows;

```
touch /etc/rc.d/init.d/firewall
```

and open the iptables file for editing;

```
nano /etc/sysconfig/iptables
```

on the Gateway Server and add:

```
#!/bin/bash
####################################################
##################################
#                                          IPTABLES
                                        #
####################################################
##################################
#
# Invoked from /etc/rc.d/init.d/firewall
        chkconfig iptables on
# description: supposed to Starts and stops the IPTABLES \
# used to provide services of network firewall
####################################################
##########
```

```
# name and location of the iptables.

        IPTABLES=iptables

# The path to the iptables executable.

        PATH="/sbin"

#internal network address space and its supporting network
device.

        OURNET="10.10.2.0/16"

        OURBCAST="10.10.255.255"

        OURDEV="eth1"

# The outside address and the network device that supports it.

        ANYADDR="0/0"

        ANYDEV="eth0"

# TCP services that can go through - "" empty implies all
ports

        TCPIN="smtp,www"

        TCPOUT="smtp,www,ftp,ftp-data,irc"

# The UDP services that can through - "" empty implies all
ports

        UDPIN="domain"

        UDPOUT="domain"

# The ICMP that can go through - "" empty implies all types

# ref: /usr/include/netinet/ip_icmp.h for type numbers

        ICMPIN="0,3,11"

        ICMPOUT="8,3,11"

# Logging; logging of datagrams are enabled by the
following line

# that are blocked by the firewall.

        LOGGING=1

####################################################
#################

# Source function library.
        . /etc/rc.d/init.d/functions

# Source networking configuration.
        . /etc/sysconfig/network

# Check that networking is up
        [ ${NETWORKING} = "no" ] && exit 0

        case "$1" in
        start)
                echo -n "Starting Firewalling Services: "

####################################################
##################
# Flush the Input table rules
        $IPTABLES -F FORWARD

# Denying incoming access by default
        $IPTABLES -P FORWARD DROP
```

```
# All datagrams destined for this host received from outside
dropped.
        $IPTABLES -A INPUT -i $ANYDEV -j DROP

# SPOOFING
# Any datagrams with a source address matching the source
should be dropped
        $IPTABLES -A FORWARD -s $OURNET -i
$ANYDEV -j DROP

# SMURF
# Disallow ICMP to access the broadcast address to prevent
"Smurf"  attack.
        $IPTABLES -A FORWARD -m multiport -p icmp
-i $ANYDEV -d $OURNET -j DROP

# Fragments should be accepted, in iptables must be done
explicitly
        $IPTABLES -A FORWARD -f -j ACCEPT

# TCP
# Accept TCP datagrams belonging to an existing connection
# (i.e. having the ACK bit set) for the TCP ports  allowed
through.
# More than 95 % of valid TCP packets should be caught.
        $IPTABLES -A FORWARD -m multiport -p tcp -d
$OURNET --dports $TCPIN /
                ! --tcp-flags SYN,ACK ACK -j ACCEPT
        $IPTABLES -A FORWARD -m multiport -p tcp -s
$OURNET --sports $TCPIN /
                ! --tcp-flags SYN,ACK ACK -j ACCEPT

# TCP - INCOMING CONNECTIONS
# Connection requests from the outside would only be
accepted when it comes from
# allowed TCP ports.
        $IPTABLES -A FORWARD -m multiport -p tcp -i
$ANYDEV -d $OURNET $TCPIN /
                --syn -j ACCEPT

# TCP - OUTGOING CONNECTIONS
# Accept only outgoing tcp connection requests made from
allowed
# TCP ports.
        $IPTABLES -A FORWARD -m multiport -p tcp -i
$OURDEV -d $ANYADDR /
        --dports $TCPOUT --syn -j ACCEPT

# UDP - INCOMING
# Only UDP datagrams on the in and back allowed ports are
allowed
        $IPTABLES -A FORWARD -m multiport -p udp -i
$ANYDEV -d $OURNET /
                --dports $UDPIN -j ACCEPT
        $IPTABLES -A FORWARD -m multiport -p udp -i
$ANYDEV -s $OURNET /
                --sports $UDPIN -j ACCEPT

# UDP - OUTGOING
# Only allow UDP datagrams out to the allowed ports and
back.
        $IPTABLES -A FORWARD -m multiport -p udp -i
$OURDEV -d $ANYADDR /
                --dports $UDPOUT -j ACCEPT
        $IPTABLES -A FORWARD -m multiport -p udp -i
$OURDEV -s $ANYADDR /
                        --sports $UDPOUT -j ACCEPT
```

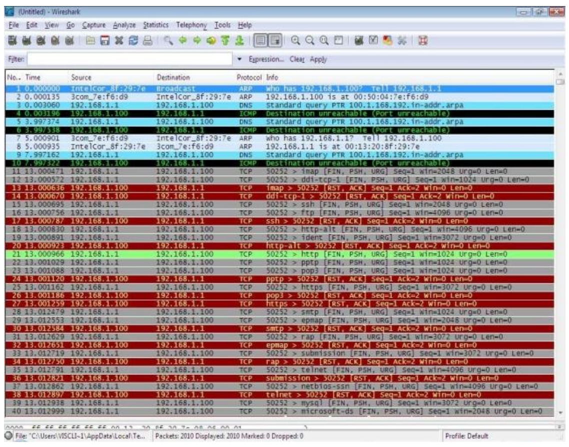
```
# ICMP - INCOMING
# Only allow ICMP datagrams in of the allowed types.
            $IPTABLES -A FORWARD -m multiport -p icmp
-i $ANYDEV -d $OURNET /
                        --dports $ICMPIN -j ACCEPT

# ICMP - OUTGOING
# Only allow ICMP datagrams out of the allowed types.
            $IPTABLES -A FORWARD -m multiport -p icmp
-i $OURDEV -d $ANYADDR /
                        --dports $ICMPOUT -j ACCEPT

# DEFAULT and LOGGING
# All other datagrams that does not fall within the
#cartegories are by default dropped by the rules.
#The default was configured a such

# configured the LOGGING variable above.
#
            if [ "$LOGGING" ]
            then
                        # Log barred TCP
                        $IPTABLES -A FORWARD -m tcp -p
tcp -j LOG
                        # Log barred UDP
                        $IPTABLES -A FORWARD -m udp -p
udp -j LOG
                        # Log barred ICMP
                        $IPTABLES -A FORWARD -m udp -p
            icmp -j LOG
            fi
            #
            end
```

7. NETWORK TRAFFIC ANALYSIS

Wireshark is a powerful protocol assessor (and sniffer) that is used by network professionals to troubleshoot and analyze network traffic under great scrutiny. Since the information revealed by Wireshark can be used to either attack or protect a network, it is a valuable tool for administrators to recognize strategic methods of hacker attacks. Wireshark is a utility that was used to look at how various protocols worked and analysis of network traffic at different stages of the development and implementation this research.

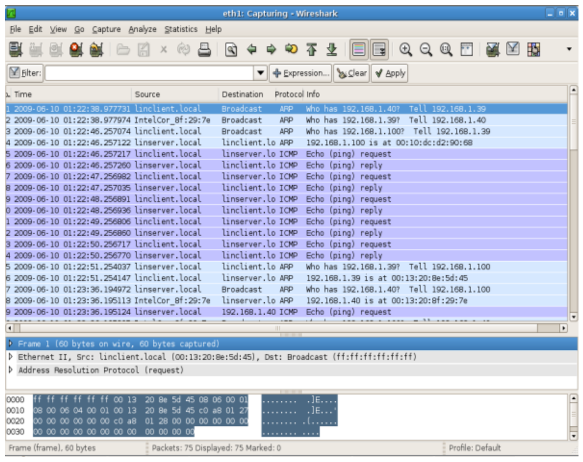
7.1 Capturing Packets

After installing and lunching of the Wireshark, the appropriate interface is selected and packet capturing begins. Advance features are to make view and investigation more precise.
Packets captured are in real time and in includes packets from both to and from the operating end.


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter: Expression... Clear Apply

```
No.. Time      Source             Destination        Protocol Info
  1 0.000000   IntelCor_8f:29:7e  Broadcast          ARP  Who has 192.168.1.100?  Tell 192.168.1.1
  2 0.000135   3com_7e:f6:d9      IntelCor_8f:29:7e  ARP  192.168.1.100 is at 00:50:04:7e:f6:d9
  3 0.003060   192.168.1.1        192.168.1.100      DNS  Standard query PTR 100.1.168.192.in-addr.arpa
  4 0.003196   192.168.1.100      192.168.1.1        ICMP Destination unreachable (Port unreachable)
  5 3.997374   192.168.1.1        192.168.1.100      DNS  Standard query PTR 100.1.168.192.in-addr.arpa
  6 3.997538   192.168.1.100      192.168.1.1        ICMP Destination unreachable (Port unreachable)
  7 5.000901   3com_7e:f6:d9      IntelCor_8f:29:7e  ARP  Who has 192.168.1.1?  Tell 192.168.1.100
  8 5.000935   IntelCor_8f:29:7e  3com_7e:f6:d9      ARP  192.168.1.1 is at 00:13:20:8f:29:7e
  9 7.997162   192.168.1.1        192.168.1.100      DNS  Standard query PTR 100.1.168.192.in-addr.arpa
 10 7.997322   192.168.1.100      192.168.1.1        ICMP Destination unreachable (Port unreachable)
 11 13.000471  192.168.1.1        192.168.1.100      TCP  50252 > imap [FIN, PSH, URG] Seq=1 Win=2048 Urg=0 Len=0
 12 13.000572  192.168.1.1        192.168.1.100      TCP  50252 > ddi-tcp-1 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 13 13.000636  192.168.1.100      192.168.1.1        TCP  imap > 50252 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
 14 13.000670  192.168.1.100      192.168.1.1        TCP  ddi-tcp-1 > 50252 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
 15 13.000695  192.168.1.1        192.168.1.100      TCP  50252 > ssh [FIN, PSH, URG] Seq=1 Win=2048 Urg=0 Len=0
 16 13.000756  192.168.1.1        192.168.1.100      TCP  50252 > ftp [FIN, PSH, URG] Seq=1 Win=4096 Urg=0 Len=0
 17 13.000787  192.168.1.100      192.168.1.1        TCP  ssh > 50252 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
 18 13.000830  192.168.1.1        192.168.1.100      TCP  50252 > http-alt [FIN, PSH, URG] Seq=1 Win=4096 Urg=0 Len=0
 19 13.000891  192.168.1.1        192.168.1.100      TCP  50252 > ident [FIN, PSH, URG] Seq=1 Win=3072 Urg=0 Len=0
 20 13.000923  192.168.1.100      192.168.1.1        TCP  http-alt > 50252 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
 21 13.000966  192.168.1.1        192.168.1.100      TCP  50252 > http [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 22 13.001029  192.168.1.1        192.168.1.100      TCP  50252 > pptp [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 23 13.001088  192.168.1.1        192.168.1.100      TCP  50252 > pop3 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 24 13.001120  192.168.1.100      192.168.1.1        TCP  pptp > 50252 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
 25 13.001162  192.168.1.1        192.168.1.100      TCP  50252 > https [FIN, PSH, URG] Seq=1 Win=3072 Urg=0 Len=0
 26 13.001186  192.168.1.100      192.168.1.1        TCP  pop3 > 50252 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
 27 13.001259  192.168.1.100      192.168.1.1        TCP  https > 50252 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
 28 13.012479  192.168.1.1        192.168.1.100      TCP  50252 > smtp [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 29 13.012553  192.168.1.1        192.168.1.100      TCP  50252 > epmap [FIN, PSH, URG] Seq=1 Win=2048 Urg=0 Len=0
 30 13.012584  192.168.1.100      192.168.1.1        TCP  smtp > 50252 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
 31 13.012629  192.168.1.1        192.168.1.100      TCP  50252 > rap [FIN, PSH, URG] Seq=1 Win=3072 Urg=0 Len=0
 32 13.012651  192.168.1.100      192.168.1.1        TCP  epmap > 50252 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
 33 13.012719  192.168.1.1        192.168.1.100      TCP  50252 > submission [FIN, PSH, URG] Seq=1 Win=3072 Urg=0 Len=0
 34 13.012750  192.168.1.100      192.168.1.1        TCP  rap > 50252 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
 35 13.012791  192.168.1.1        192.168.1.100      TCP  50252 > telnet [FIN, PSH, URG] Seq=1 Win=4096 Urg=0 Len=0
 36 13.012821  192.168.1.100      192.168.1.1        TCP  submission > 50252 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
 37 13.012862  192.168.1.1        192.168.1.100      TCP  50252 > netbios-ssn [FIN, PSH, URG] Seq=1 Win=4096 Urg=0 Len=0
 38 13.012897  192.168.1.100      192.168.1.1        TCP  telnet > 50252 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
 39 13.012938  192.168.1.1        192.168.1.100      TCP  50252 > mysql [FIN, PSH, URG] Seq=1 Win=3072 Urg=0 Len=0
 40 13.012999  192.168.1.1        192.168.1.100      TCP  50252 > microsoft-ds [FIN, PSH, URG] Seq=1 Win=2048 Urg=0 Len=0

File: "C:\Users\VISCLI~1\AppData\Local\Te...   Packets: 2010 Displayed: 2010 Marked: 0 Dropped: 0   Profile: Default
```

Figure 2: Captured Network Packets using Wireshark in Linux.


eth1: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply

```
. Time                       Source             Destination  Protocol Info
1 2009-06-10 01:22:38.977731 linclient.local    Broadcast    ARP  Who has 192.168.1.40?  Tell 192.168.1.39
2 2009-06-10 01:22:38.977974 IntelCor_8f:29:7e  Broadcast    ARP  Who has 192.168.1.39?  Tell 192.168.1.40
3 2009-06-10 01:22:46.257074 linclient.local    Broadcast    ARP  Who has 192.168.1.100?  Tell 192.168.1.39
4 2009-06-10 01:22:46.257122 linserver.local    linclient.lo ARP  192.168.1.100 is at 00:10:dc:d2:90:68
5 2009-06-10 01:22:46.257217 linclient.local    linserver.lo ICMP Echo (ping) request
6 2009-06-10 01:22:46.257260 linserver.local    linclient.lo ICMP Echo (ping) reply
7 2009-06-10 01:22:47.256982 linclient.local    linserver.lo ICMP Echo (ping) request
8 2009-06-10 01:22:47.257035 linserver.local    linclient.lo ICMP Echo (ping) reply
9 2009-06-10 01:22:48.256891 linclient.local    linserver.lo ICMP Echo (ping) request
0 2009-06-10 01:22:48.256936 linserver.local    linclient.lo ICMP Echo (ping) reply
1 2009-06-10 01:22:49.256806 linclient.local    linserver.lo ICMP Echo (ping) request
2 2009-06-10 01:22:49.256860 linserver.local    linclient.lo ICMP Echo (ping) reply
3 2009-06-10 01:22:50.256717 linclient.local    linserver.lo ICMP Echo (ping) request
4 2009-06-10 01:22:50.256770 linserver.local    linclient.lo ICMP Echo (ping) reply
5 2009-06-10 01:22:51.254037 linserver.local    linclient.lo ARP  Who has 192.168.1.39?  Tell 192.168.1.100
6 2009-06-10 01:22:51.254147 linclient.local    linserver.lo ARP  192.168.1.39 is at 00:13:20:8e:5d:45
7 2009-06-10 01:23:36.194972 linserver.local    Broadcast    ARP  Who has 192.168.1.40?  Tell 192.168.1.100
8 2009-06-10 01:23:36.195113 IntelCor_8f:29:7e  linserver.lo ARP  192.168.1.40 is at 00:13:20:8f:29:7e
9 2009-06-10 01:23:36.195124 linserver.local    192.168.1.40 ICMP Echo (ping) request

▷ Frame 1 (60 bytes on wire, 60 bytes captured)
▷ Ethernet II, Src: linclient.local (00:13:20:8e:5d:45), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▷ Address Resolution Protocol (request)

0000  ff ff ff ff ff ff 00 13  20 8e 5d 45 08 06 00 01   ........ .]E....
0010  08 00 06 04 00 01 00 13  20 8e 5d 45 c0 a8 01 27   ........ .]E...'
0020  00 00 00 00 00 00 c0 a8  01 28 00 00 00 00 00 00   ........ .(......
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....

Frame (frame), 60 bytes   Packets: 75 Displayed: 75 Marked: 0   Profile: Default
```

Figure 3: Captured packets in Wireshark.

7.2 Color Coding

Colour coding the traffic flow allows the captured packets to easily indentified at a glance. It provides unique and codified look of captured packets. It is mostly comes in green, dark blue and black colours. The green representing TCP traffic, dark blue for DNS traffic and finally the black for TCP packets that had errors during delivery

As shown in Figures 3 above, Wireshark's main screen is separated in to three sections.

7.3 Packet List Section

Located at the top, this section displays a summary of the packets captured. Clicking on any one of the packets in this section displays more detailed information about that packet in the other two sections.

7.4 Tree View Section

Located in the middle, this section displays detailed information about the packet selected in a tree structure.

7.5 Data View Section

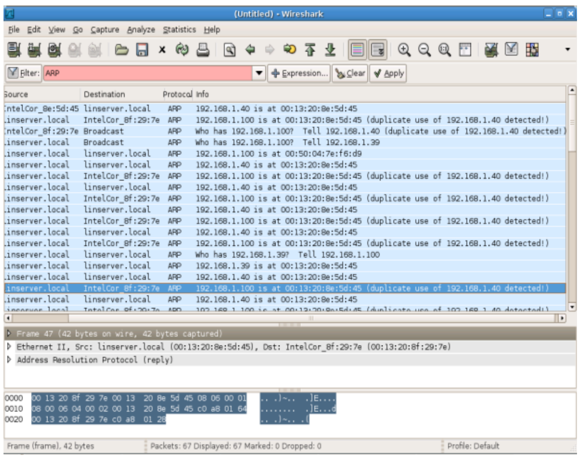
Located at the bottom, this section shows the data captured in hexadecimal format. Hexadecimal is the base 16 numbering system, and is composed of numbers the 0-9 and the letter A-F. Hexadecimal is sometimes used as a short way of representing binary numbers. Any section selected in the Tree View Section is highlighted in this section.

There are six columns in the Packet List Section. Each column provides specific information about the packet:

- No. – The order in which the packets were received
- Time – The time each packet was captured relative to the beginning of the capture
- Source – Source address
- Destination – Destination address
- Protocol – Protocol used to capture the packet
- Info – A summary of what the packet is doing

The highlighted frame in the summary section is what is displayed in the tree view and data view sections.

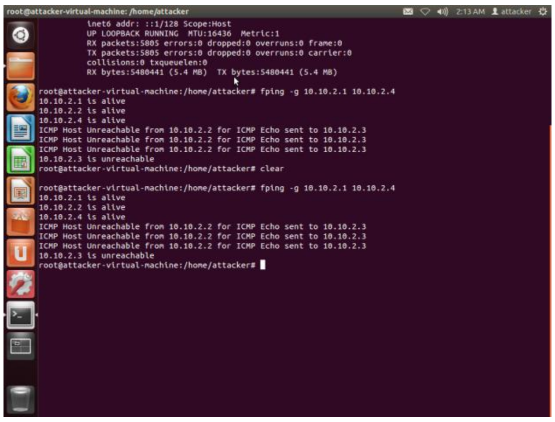
- Filtering packets:


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Expression... Clear Apply

```
Filter: ARP

Source             Destination         Protocol Info
IntelCor_8e:5d:45  linserver.local     ARP      192.168.1.40 is at 00:13:20:8e:5d:45
linserver.local    IntelCor_8f:29:7e   ARP      192.168.1.100 is at 00:13:20:8e:5d:45 (duplicate use of 192.168.1.40 detected!)
IntelCor_8f:29:7e  Broadcast           ARP      Who has 192.168.1.100?  Tell 192.168.1.40 (duplicate use of 192.168.1.40 detected!)
linserver.local    Broadcast           ARP      Who has 192.168.1.100?  Tell 192.168.1.39
linserver.local    linserver.local     ARP      192.168.1.100 is at 00:50:04:7e:f6:d9
linserver.local    linserver.local     ARP      192.168.1.40 is at 00:13:20:8e:5d:45
linserver.local    IntelCor_8f:29:7e   ARP      192.168.1.100 is at 00:13:20:8e:5d:45 (duplicate use of 192.168.1.40 detected!)
linserver.local    linserver.local     ARP      192.168.1.40 is at 00:13:20:8e:5d:45
linserver.local    IntelCor_8f:29:7e   ARP      192.168.1.100 is at 00:13:20:8e:5d:45 (duplicate use of 192.168.1.40 detected!)
linserver.local    linserver.local     ARP      192.168.1.40 is at 00:13:20:8e:5d:45
linserver.local    IntelCor_8f:29:7e   ARP      192.168.1.100 is at 00:13:20:8e:5d:45 (duplicate use of 192.168.1.40 detected!)
linserver.local    linserver.local     ARP      192.168.1.40 is at 00:13:20:8e:5d:45
linserver.local    IntelCor_8f:29:7e   ARP      192.168.1.100 is at 00:13:20:8e:5d:45 (duplicate use of 192.168.1.40 detected!)
linserver.local    linserver.local     ARP      Who has 192.168.1.39?  Tell 192.168.1.100
linserver.local    linserver.local     ARP      192.168.1.39 is at 00:13:20:8e:5d:45
linserver.local    linserver.local     ARP      192.168.1.40 is at 00:13:20:8e:5d:45
linserver.local    IntelCor_8f:29:7e   ARP      192.168.1.100 is at 00:13:20:8e:5d:45 (duplicate use of 192.168.1.40 detected!)
linserver.local    linserver.local     ARP      192.168.1.40 is at 00:13:20:8e:5d:45

Frame 47 (42 bytes on wire, 42 bytes captured)
Ethernet II, Src: linserver.local (00:13:20:8e:5d:45), Dst: IntelCor_8f:29:7e (00:13:20:8f:29:7e)
Address Resolution Protocol (reply)

0000  00 13 20 8f 29 7e 00 13  20 8e 5d 45 08 06 00 01   .. .)~..  .]E....
0010  08 00 06 04 00 02 00 13  20 8e 5d 45 c0 a8 01 64   ........  .]E...d
0020  00 13 20 8f 29 7e c0 a8  01 28                     .. .)~.. .(

Frame (frame), 42 bytes    Packets: 67 Displayed: 67 Marked: 0 Dropped: 0    Profile: Default
```

Figure 4: Using the filter in Wireshark

Attacks using Metasploit FrameworkDoing an fping to see which machines are on the network:



```
root@attacker-virtual-machine: /home/attacker                    2:13 AM  attacker
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:5805 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5805 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:5480441 (5.4 MB)  TX bytes:5480441 (5.4 MB)

root@attacker-virtual-machine:/home/attacker# fping -g 10.10.2.1 10.10.2.4
10.10.2.1 is alive
10.10.2.2 is alive
10.10.2.4 is alive
ICMP Host Unreachable from 10.10.2.2 for ICMP Echo sent to 10.10.2.3
ICMP Host Unreachable from 10.10.2.2 for ICMP Echo sent to 10.10.2.3
ICMP Host Unreachable from 10.10.2.2 for ICMP Echo sent to 10.10.2.3
10.10.2.3 is unreachable
root@attacker-virtual-machine:/home/attacker# clear

root@attacker-virtual-machine:/home/attacker# fping -g 10.10.2.1 10.10.2.4
10.10.2.1 is alive
10.10.2.2 is alive
10.10.2.4 is alive
ICMP Host Unreachable from 10.10.2.2 for ICMP Echo sent to 10.10.2.3
ICMP Host Unreachable from 10.10.2.2 for ICMP Echo sent to 10.10.2.3
ICMP Host Unreachable from 10.10.2.2 for ICMP Echo sent to 10.10.2.3
10.10.2.3 is unreachable
root@attacker-virtual-machine:/home/attacker# 
```

Figure 5: MSFconsole

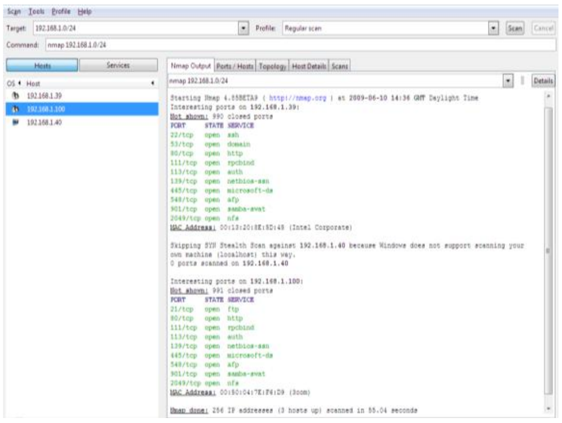
7.6 Filtering Packets

Filtering packets involves parsing packets through pre-determine set of configured setting to regulate packet flow to and from the system. In this case by the typing the appropriate group of packets at the top of the Wireshark window and applying the choice. A choice of TCP would reveal only TCP packets and same applies to UDP and DNS.

Filters can always be added by clicking on the Analyze menu and then select the appropriate filter. Additionally specific packets can be selected to view the TCP stream by right clicking on the appropriate packet and that reveals a possible correspondence between a client and a server.

Packets can be further examined or inspected by selecting the appropriate packet and view additional information such as the source and destination address, sequence number, acknowledgement number Header length etc.

Nmap was used to determine which nodes were active and which were not. This was very necessary as it helped to access traffic generated by these nodes. It was also used to scan and also provide an opportunity for comparison among previous and new scans. Telnet service came in as handy to access IP/port combinations and banners generated from the Nmap activities.



```
Target: 192.168.1.0/24     Profile: Regular scan
Command: nmap 192.168.1.0/24

OS  Host
    192.168.1.39
    192.168.1.100
    192.168.1.40

nmap 192.168.1.0/24

Starting Nmap 4.85BETA9 ( http://nmap.org ) at 2009-06-10 14:36 GMT Daylight Time
Interesting ports on 192.168.1.39:
Not shown: 990 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
53/tcp   open  domain
80/tcp   open  http
111/tcp  open  rpcbind
113/tcp  open  auth
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
548/tcp  open  afp
901/tcp  open  samba-swat
2049/tcp open  nfs
MAC Address: 00:13:20:8E:5D:45 (Intel Corporate)

Skipping SYN Stealth Scan against 192.168.1.40 because Windows does not support scanning your
own machine (localhost) this way.
0 ports scanned on 192.168.1.40

Interesting ports on 192.168.1.100:
Not shown: 991 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
80/tcp   open  http
111/tcp  open  rpcbind
113/tcp  open  auth
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
548/tcp  open  afp
901/tcp  open  samba-swat
2049/tcp open  nfs
MAC Address: 00:50:04:7E:F6:D9 (3com)

Nmap done: 256 IP addresses (3 hosts up) scanned in 55.04 seconds
```

Figure 13: Network scan with Nmap

8. LINUX SECURITY

Windows as prosperity operating system is naturally a superior operating systems in terms security. Windows 8 and earlier versions are typical of such operating systems with good configurable features as far as their firewalls are concern.

Linux on this occasion demonstrated superior security capabilities and has since given an opinion of windows being a weaker one though Linux remains a non-proprietary operating system. Open source operating systems has always being regarded weaker in terms of security features.

Linux has demonstrated robust features in the area of access control and privileges. Linux users always require exclusive permissions and privileges before scripts or programs that have been automated can alter anything the system. And this so because Linux does not run from the root unlike the Administrator privileges by default in windows.

Haven seen people trying to retrieve virus and malware infected windows file on Linux platforms are a manifestation of the robustness of the Linux platform as far as virus infections are concern.

Managing windows systems comes with the burden of virus and malware management which also comes with cost. Linux might not absolutely free from the virus hassle but it is extremely minimal.

9. CONCLUSION

The Linux operating system being an open source operating system has demonstrated superior security performance. Its non-proprietary status did not undermine its robustness as for security issues are concerned. Intrusion Detection System implementation over the Linux platform allowed for network traffic analysis using the Wireshark open source application.

In executing this experiment, the author appreciated the need for computer networks to be very robust and provide an infrastructure that is always available to ensure data communication from source to destination despite the presence of illegitimate users, hackers etc. Linux exhibited resilience to threats and effectiveness of Intrusion Detection Systems implemented over Linux platform. Not sure how the proprietary version, UNIX would have performed as it is believed that proprietary Operating Systems performs better though not in this case. On the other hand Windows as a proprietary Operating System though widely considered to be the safest and the most popular platform has its main feature of giving full administrative rights to users as its main weakness. However running a comparative simultaneous cross platform implementation experiment would offer an opportunity for a fair assessment of responses to common threats.

10. REFERENCES

[1] Ptacek, T. H. & Newsham, T. N. (1998). Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection [Online] Retrieved February 16, 2005 fromhttp://www.snort.org/docs/idspaper/

[2] Kumar, S., (1995), Classification and Detection of Computer Intrusions. (Thesis).

[3] Lunt, T. (1993). Detecting Intruders in Computer Systems [Online] Retrieved 25th March, 2005 from http://www.raptor.com/lib/canada93.ps

[4] Roesch, M. (1999) Snort-Lightweight Intrusion Detection for Networks, Proceedings of LISA '99: 13th Systems Administration Conference [Online] Retrieved 10th April, 2005 fromhttp://www.usenix.org/publications/library/proceedings/lisa99/full_papers/roesch/roesch.pdf

[5] Northcutt, S. and Novak, J., (2003), Network Intrusion Detection: An Analyst's Handbook, Third Edition. New Riders.

[6] Northcutt, S. (2005) What is network intrusion detection? [Online] Retrieved 28th March, 2005 from http://www.sans.org/resources/idfaq/network_based.php

[7] Allen, J. & Christie, A. (2000). "State of the practice of intrusion detection technologies" [Online] Retrieved February 20, 2005 from http://www.cert.org/archive/pdf/99tr028.pdf

[8] Axelsson, S. "Intrusion detection systems: A survey and taxonomy," *Technical Report 99-15*, Department of Computer Engineering, Chalmers University, March 2000.

[9] Burgess, M. (2004). Principles of Networking and System administration, (2nd Ed.). Chichester, John Wiley and Sons, Ltd.

[10] Forrest, S., Hofmeyr, S. A., Somayaji, A. & Longstaff, T.A. (1996). "*A Sense of Self for Unix Processes*" Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy [Online] Retrieved February 24, 2005 fromhttp://citeseer.ist.psu.edu/forrest96sense.html

[11] Choi, M., (2008), Wireless Network Security. International Journal of Multimedia and Ubiquitous Engineering (Vol. 3, No. 3). School of Multimedia, Hannam University, Daejeon, Korea.

[12] Nestler, V. J., et al., (2006), Computer Security Lab Manual. McGraw-Hill/Irwin, New York, USA.

[13] Early, G (2004). Transmission Control Protocol (TCP), Lecture 04, University of Portsmouth.

[14] Kurose, J.F. & Ross, K.W. Computer Networking: A Top- Down Approach Featuring the Internet, (2nd Ed.). Pearson Education, Inc.

[15] Northcut, S. & Novak, J. (2002). Network Intrusion Detection, (3rd Ed.). New Riders Publishing.