2018

On Assessing the Impact of Ports Scanning on the Target Infrastructure

Dr Mahdi Aiash

4/24/2018

1. Introduction

A port scan is a method for determining which ports on a network are open. As ports on a computer are the place where information is sent and received, port scanning is analogous to knocking on doors to see if someone is home. Running a port scan on a network or server reveals which ports are open and listening (receiving information), as well as revealing the presence of security devices such as firewalls that are present between the sender and the target. This technique is known as fingerprinting. It is also valuable for testing network security and the strength of the system's firewall. Due to this functionality, it is also a popular reconnaissance tool for attackers seeking a weak point of access to break into a computer.

Generally speaking, ports scanning probes a number of well-known ports (for TCP and UDP) by sending raw network packets to these ports, and based on the responses from these ports, it classifies ports in one of these states:

- Open or Accepted: The host sent a reply indicating that a service is listening on the port.
- Closed or Denied or Not Listening: The host sent a reply indicating that connections will be denied to the port.
- Filtered, Dropped or Blocked: There was no reply from the host.

2. Problem Definition

Fingerprinting and port scanning are the longest stage during pentesting engagements. The thoroughness and completeness of this stage is a key for the success of the vulnerability assessment and exploitation stages. Therefore, vendors have developed large number of port scanning tools (both commercial and open source). However, experienced pentesters might choose to deploy their own bespoke tool that suits their needs. It is important also to highlight that there are dozens of different types (techniques) of port scanning which are different from each other in the following ways:

- The way they parse the response of scanned ports, this effectivity affects the accuracy of the different scans.

- Different types of port scans send a distinct volumes of raw packets to probes the scanned ports consequently resulting in a varying level of overhead that might potentially affect the performance of the scanned host and network.

For pentesters, it is crucial to understand the difference between the scan techniques, and to be able to choose the appropriate one (or combination) for a given task. Therefore, in this report we conduct an experiment to evaluate the accuracy of different scans and to assess their impact on the scanned infrastructure. The experiment is based on testbed created using Vmware workstation and simulates a traditional production infrastructure. We conducted the following two experiments:

- Experiment one: We developed our own information gathering tools that runs ping sweep to discover live hosts and then a TCP port scan these hosts. The tool finally performs banners grabbing to identify the version of the service listening on the open ports. We evaluated the impact of using this tool on the performance of the scanned network in the testbed in terms of latency and the number of the generated packets.
- Experiment two: We used Nmap (a well-known scanning tool) to run different types of scans (both TCP and UDP) and compare the accuracy and the resulted overhead of these scans.

We believe that the results presented in this report will help security researchers and pentesters understanding the difference between port scan techniques and assess the potentially impact of their reconnaissance on the target infrastructure.

3. Testbed setup

The lab test will take place on a LAN network designed for the purpose of this project as shown in Fig 1:

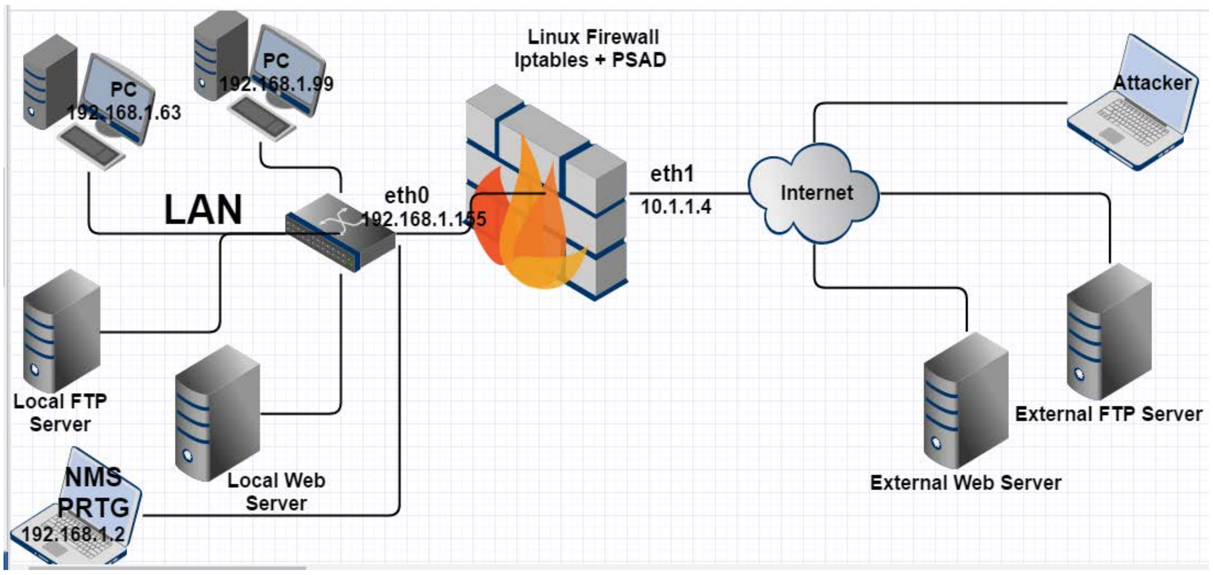


Figure 1 Testbed

As shown in figure 1, a Linux firewall is placed between the internal and external network in order to protect the internal network and prevent unauthorised access from the Internet. This network is designed and built in a virtualised environment and is constructed close to a real enterprise network design. The attacker (scanner) machine located in the outside network will then run a python port scan script against the LAN network (192.168.1.0/24). Another computer (192.168.1.2) located in the target network will be closely monitoring the performance of the network without alerting the attacker that their actions are observed.

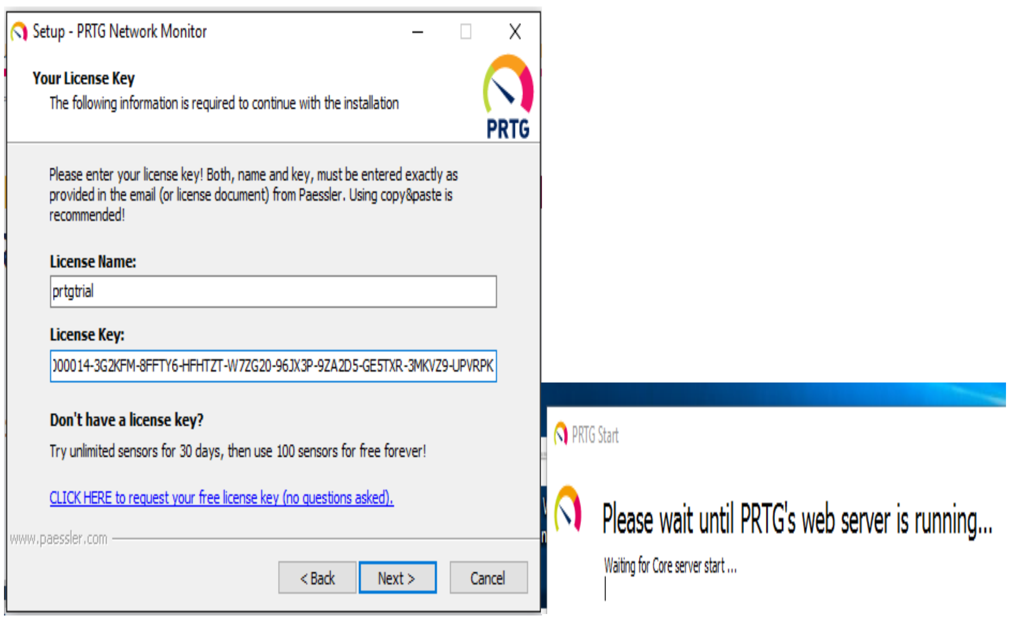
For the purpose of this project the LAN network will be monitored in two different occasions. In the first experiment a threshold is set as a base-line for acceptable level of network traffic, where in the second scenario a port scan attack will be launched against the target network while using the same base-line in the first scenario the impact will be measured and the gathered data will be analysed further. In experiment 2 of this project, the network design will remain the same but Nmap will be used to evaluate different scanning techniques.

3.1. Installation Phase

Both the attacker's machine which run Kali-Linux OS and the target network machines were installed using VMware workstation Pro 12 and the network interfaces were configured according to the network design in figure 1.

3.1.1. Monitoring Software

To monitor the LAN performance, the PRTG monitoring software is downloaded from the "www.paessler.com website to collect data and statistics from all the components in the network. After completing the installation of PRTG, the user interface opens up the standard browser and automatically starts discovering the network in the background as in Fig 2. Additionally, users can launch this software using the PRTG enterprise console which uses a native user interface.



Setup - PRTG Network Monitor

Your License Key

The following information is required to continue with the installation

Please enter your license key! Both, name and key, must be entered exactly as provided in the email (or license document) from Paessler. Using copy&paste is recommended!

License Name: prtgtrial

License Key:)00014-3G2KFM-8FFTY6-HFHTZT-W7ZG20-96JX3P-9ZA2D5-GE5TXR-3MKVZ9-UPVRPK

Don't have a license key?

Try unlimited sensors for 30 days, then use 100 sensors for free forever!

CLICK HERE to request your free license key (no questions asked).

www.paessler.com

PRTG Start

Please wait until PRTG's web server is running...

Waiting for Core server start ...

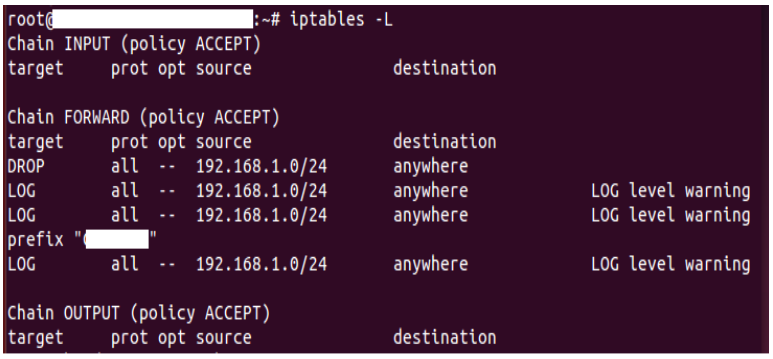
Figure 2: PRTG

3.1.2. Installing and Setting up Iptables firewall on Ubuntu Server:

Iptables is a Linux based firewall developed by Netfilter project (netfilter.org) and has been a pre-installed application in Linux kernels released after early 2001. Iptables are divided into two main components (userland administration program and kernel modules) where both the userland and the kernel are compiled. The latest releases of Linux distributions provide pre built-in kernels that have Iptables compiled in and does not require it to be installed as a separate feature. Many Netfilter subsystems such as packet filtering and connection tracking capabilities are included in the kernel source code and are enabled by default. It is very important to configure and compile Linux kernel appropriately as any misconfiguration error can lead to the Iptables firewall not functioning properly and risk the system becoming vulnerable to more serious threats. All the packet filtering and other matching operations in Iptables are part of the kernel process.

After setting up the network as in the Figure 1, we start the process of configuring the firewall as follows:

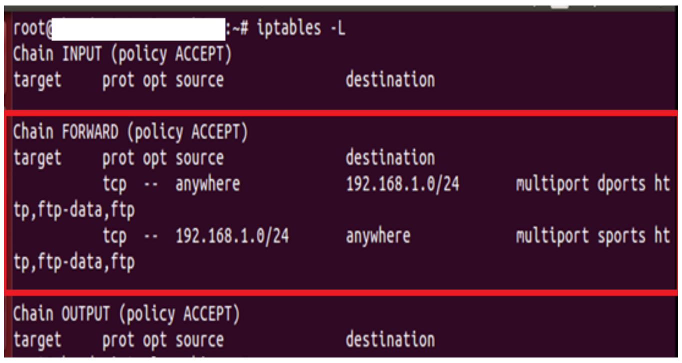
- A network that uses Iptables firewall works by accepting packets that are mandatory in order for the network to function, all other packets are logged or dropped which provides an important information for log analysing tools. The firewall has two NICs with the following IP addresses:
  Eth0: 10.1.1.0/24
  Eth1: 192.168.1.0/24
  The objective of having two different NICs is to configure the firewall and enable eth1 network access the internet and use the web, FTP, mail and other services. A good way to start configuring Iptables firewall is to flush all the existing rules using the Iptables "–F" option. The next step is to create the anti-spoofing rules which are an essential factor of configuring any firewall. Attackers achieve this technique by faking their source address and make it resemble the internal network of an organisation. To avoid the danger of attackers bypassing the firewall a simple rules are created and assigned to the internet interface that inspects the source address of all incoming packets and drops any packets with a source address that is identical to the trusted internal addresses as shown in Fig 3.



```
root@[redacted]:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  192.168.1.0/24       anywhere
LOG        all  --  192.168.1.0/24       anywhere             LOG level warning
LOG        all  --  192.168.1.0/24       anywhere             LOG level warning
prefix "[redacted]"
LOG        all  --  192.168.1.0/24       anywhere             LOG level warning

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Figure 3: Iptables rules

LAN users are allowed to access the Web and the FTP server through the internet. The firewall is configured to allow the inbound and outbound packets destined for port 80, 20, and 21 as shown in Fig 4.



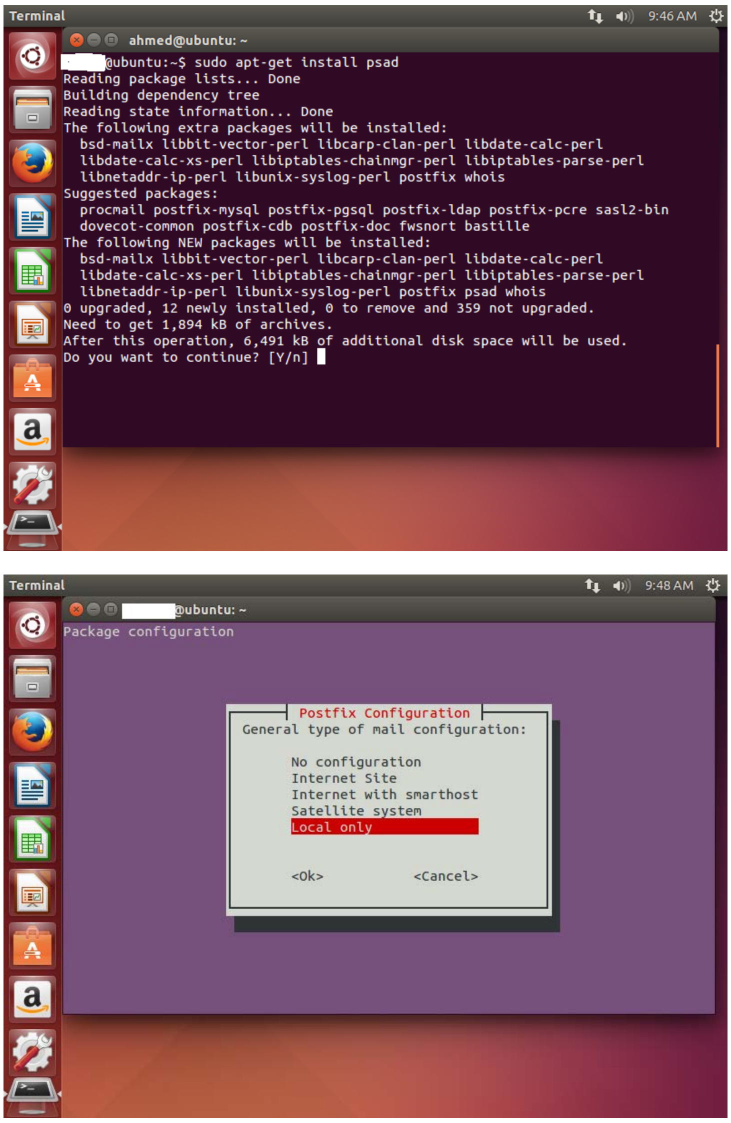
```
root@[redacted]:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
           tcp  --  anywhere             192.168.1.0/24       multiport dports ht
tp,ftp-data,ftp
           tcp  --  192.168.1.0/24       anywhere             multiport sports ht
tp,ftp-data,ftp

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Figure 4: Iptables LAN rules

3.1.3. Installing and Setting up PSAD

PSAD is an application that runs on Linux hosts which monitors the Iptables log messages in order to detect, alert, and block a port scan or any other malicious activities that are taking place. Originally, PSAD started as part of the Bastille Linux project in the late 90s. Bastille Linux Project developers decided it was time to develop a network intrusion detection system (IDS). The idea of the team was to create something new and different from the existing IDS and that fits well with the Linux Firewall. As a result, the PSAD software was launched in 2001 to actively monitor and analyse the Iptables logs. For instance, in the event of TCP port scan, the PSAD is able to inspect the TCP flags and identify the type of the TCP scan e.g. SYN, XMAS, FIN, etc.). PSAD also makes use of many ICMP, UDP, and TCP signatures stored in the Snort IDS database to detect any suspicious traffic. For years, Firewalls have been the inline device to safeguard modern day networks and are usually positioned at the entry and exit point of the network. Due to some complexity in configuring firewall rules and the inability to block attacks at the higher level of the protocol stack, security experts have recommended that Firewalls should never be the only line of defence in any network. Therefore, combining Firewalls with PSAD will provide a strong security protection in any network.

PSAD is downloaded from the default repositories of Ubuntu. To obtain the PSAD program from Ubuntu's Advanced Packaging Tool (APT) the command "sudo apt-get install PSAD" was issued. During the installation process several pieces of input are requested including the Postfix mail server (Fig 5) which is an e-mail address where any alerts generated will be sent to in the event of unusual activity within the network. The PSAD IDS will also allow the visualisation of the security status of the entire network which helps humans to recognise visual patterns than thousand lines of log messages.



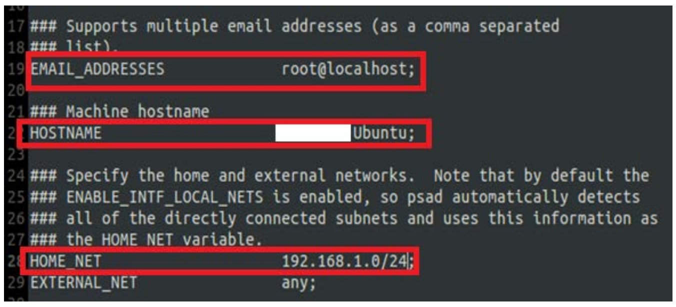
```
[redacted]@ubuntu:~$ sudo apt-get install psad
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  bsd-mailx libbit-vector-perl libcarp-clan-perl libdate-calc-perl
  libdate-calc-xs-perl libiptables-chainmgr-perl libiptables-parse-perl
  libnetaddr-ip-perl libunix-syslog-perl postfix whois
Suggested packages:
  procmail postfix-mysql postfix-pgsql postfix-ldap postfix-pcre sasl2-bin
  dovecot-common postfix-cdb postfix-doc fwsnort bastille
The following NEW packages will be installed:
  bsd-mailx libbit-vector-perl libcarp-clan-perl libdate-calc-perl
  libdate-calc-xs-perl libiptables-chainmgr-perl libiptables-parse-perl
  libnetaddr-ip-perl libunix-syslog-perl postfix psad whois
0 upgraded, 12 newly installed, 0 to remove and 359 not upgraded.
Need to get 1,894 kB of archives.
After this operation, 6,491 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

```
Package configuration

              ┤ Postfix Configuration ├
        General type of mail configuration:

              No configuration
              Internet Site
              Internet with smarthost
              Satellite system
              Local only

              <Ok>            <Cancel>
```

Figure 5: PSAD Installation

The PSAD log analyser relies heavily on the Iptables policy configuration of that system. However, there are few necessary configurations needed in order PSAD to function properly.

All the PSAD daemons references are defined under the "*/etc/psad/psad.conf*" main file which contains many configuration variables and controls the different aspect of how PSAD works. Since this is a large file, most of the configurations were not modified. The most important configuration variable are highlighted in Fig 6.

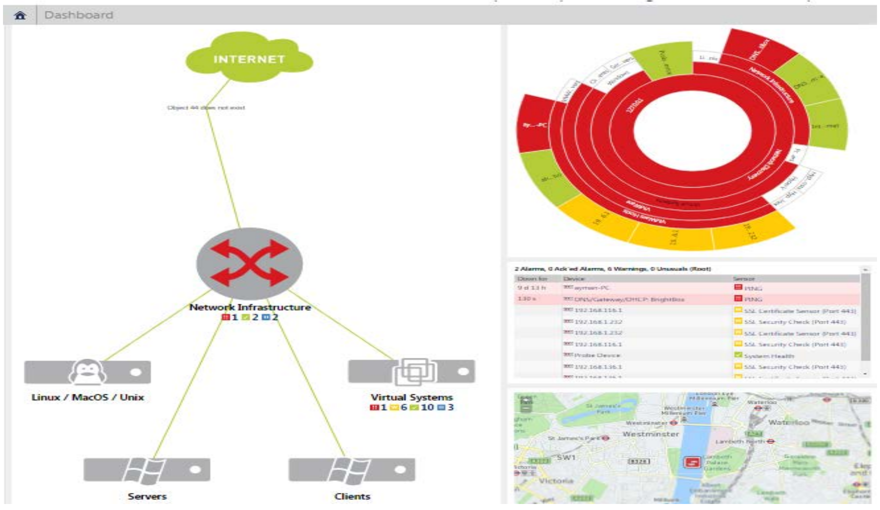


```
### Supports multiple email addresses (as a comma separated
### list).
EMAIL_ADDRESSES             root@localhost;

### Machine hostname
HOSTNAME                    [redacted]Ubuntu;

### Specify the home and external networks.  Note that by default the
### ENABLE_INTF_LOCAL_NETS is enabled, so psad automatically detects
### all of the directly connected subnets and uses this information as
### the HOME_NET variable.
HOME_NET                    192.168.1.0/24;
EXTERNAL_NET                any;
```

Figure 6:

3.1.4. PRTG Configuration

After setting up the Apache and FTP servers, we move on to configure the PRTG to measure the performance of the elements of our testbed. As shown in Fig 7, the PRTG is started and all the current devices of the network are discovered. We can see an overall view of multiple instances of all network devices in hierarchal and categorised way. This was made easy by the dashboard feature in PRTG where user have the option to choose different dashboard viewers. Furthermore, the current state of each device is visible to admin which eases the troubleshooting method in the event of failure.



2 Alarms, 0 Ack'ed Alarms, 6 Warnings, 0 Unusuals (Root)

Down for	Device	Sensor
9 d 13 h	aymen-PC	PING
130 s	DNS/Gateway/DHCP: BrightBox	PING
	192.168.116.1	SSL Certificate Sensor (Port 443)
	192.168.1.232	SSL Security Check (Port 443)
	192.168.1.232	SSL Certificate Sensor (Port 443)
	192.168.116.1	SSL Security Check (Port 443)
	Probe Device	System Health
	192.168.136.1	SSL Security Check (Port 443)

Figure 7: PRTG

4. The Conducted Experiments:

After carefully configuring the network in the previous sections, different tests have been performed to inspect the impact of port scans on a network infrastructure. This section explains the conducted experiments and presents the results obtained. Two different experiment are conducted in this project. In the first experiment, an external scan has been performed in which the LAN network was scanned from the Internet (external network). Using PRTG network monitoring tool, the network was monitored for a period of around 20 minutes in three different occasions. Then the overall network performance is compared before and during the scan. In this scan we used a bespoke tool rather than relying on existing scanning tools.

In the second experiment, the scan was conducted and different scanning techniques were performed using Nmap scanner. This experiment will assist a pen tester to choose the most efficient scanning technique in terms of performance and OS detection accuracy.

4.1. The First Experiment

In order to simulate an external attack or a black-box pentesting, we developed our own script that achieves:

- Ping Sweep: to discover live hosts in the LAN
- PortScanning: to identify open ports on the discovered, live hosts
- Banner Grabbing: After identifying the open ports, our script will fingerprint discovered service as to be able to find out any vulnerabilities specific to the discovered services.

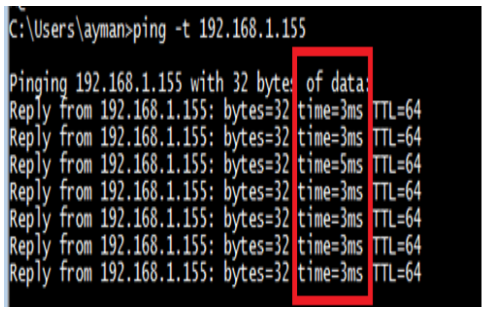
The goal of this scenario is to evaluate how a detected port scan can impact the overall network performance and increase latency.

4.1.1. Latency

In the Internet Protocol (IP) network Latency is defined to be the amount of time it takes for a packet to travel from a source to a destination, or the time it takes from a source to destination and back to the source, this is also known as Round Time Trip (RTT). There are numerous ways network latency can occur including queuing delay, TCP handshake delay, routing and switching delay. Common Internet applications such as FTP and HTTP utilize TCP to transfer data from server to client. Because the TCP is a core protocol of the TCP/IP protocol suite and uses the three-way handshake process prior exchanging data. This process can cause a total packet loss to an already congested network by adding extra congestion. In this project latency is referred to as the amount of time it takes for a LAN network user to access and retrieve data from the external web and FTP servers. Multiple protocols were used including HTTP, FTP and ICMP to generate some traffic across the network in order to calculate the latency and response time of each protocol.

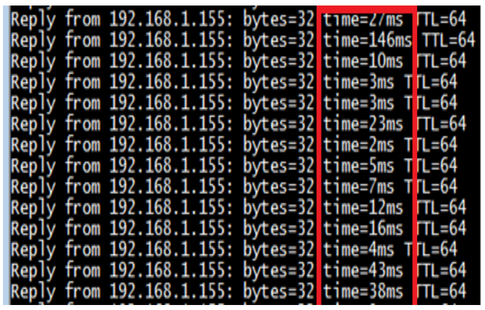
4.1.1.1. ICMP Latency:

The ping utility uses the ICMP protocol to test connectivity between two IP hosts. Ping works by sending ICMP packet called echo-request to the IP host, and the host responds back with the echo-reply. The built-in ping utility in windows is used to ping the 192.168.1.155 host and to determine the RTT value. Multiple packets are sent before and during the scan. When the network is operational accordingly and no scan is taking place, a user on the 192.168.1.0/24 network pings the 192.168.1.155 host (as the Gateway) and the result is recorded. The same step is repeated during the scan and the RTT value is recorded and compared to the value pre the scan. According to the results in in Figures 8 and 9, the RTT value was higher when the scan was taking place. The maximum response time before the scan is 3ms, while this value changes dramatically to a maximum of 146ms during the scan, which shows the significant impact of port scan attack in terms of performance when all the defence mechanisms are in place including Firewall and IDS/IPS



```
C:\Users\ayman>ping -t 192.168.1.155

Pinging 192.168.1.155 with 32 bytes of data:
Reply from 192.168.1.155: bytes=32 time=3ms TTL=64
Reply from 192.168.1.155: bytes=32 time=3ms TTL=64
Reply from 192.168.1.155: bytes=32 time=5ms TTL=64
Reply from 192.168.1.155: bytes=32 time=3ms TTL=64
Reply from 192.168.1.155: bytes=32 time=3ms TTL=64
Reply from 192.168.1.155: bytes=32 time=3ms TTL=64
Reply from 192.168.1.155: bytes=32 time=3ms TTL=64
```

Figure 8: RTT value before the scan



```
Reply from 192.168.1.155: bytes=32 time=27ms TTL=64
Reply from 192.168.1.155: bytes=32 time=146ms TTL=64
Reply from 192.168.1.155: bytes=32 time=10ms TTL=64
Reply from 192.168.1.155: bytes=32 time=3ms TTL=64
Reply from 192.168.1.155: bytes=32 time=3ms TTL=64
Reply from 192.168.1.155: bytes=32 time=23ms TTL=64
Reply from 192.168.1.155: bytes=32 time=2ms TTL=64
Reply from 192.168.1.155: bytes=32 time=5ms TTL=64
Reply from 192.168.1.155: bytes=32 time=7ms TTL=64
Reply from 192.168.1.155: bytes=32 time=12ms TTL=64
Reply from 192.168.1.155: bytes=32 time=16ms TTL=64
Reply from 192.168.1.155: bytes=32 time=4ms TTL=64
Reply from 192.168.1.155: bytes=32 time=43ms TTL=64
Reply from 192.168.1.155: bytes=32 time=38ms TTL=64
```

Figure 9: RTT value during the scan

One of the advantages using PRTG tool is the ability to separate the performance of each protocol. According to figure 10, the maximum RTT time is 2.20 msec which indicates that the network performance is balanced and there are no unusual traffic patterns, otherwise an alarm should have been generated and sent to the admin if the thresholds set by the administrator were exceeded.

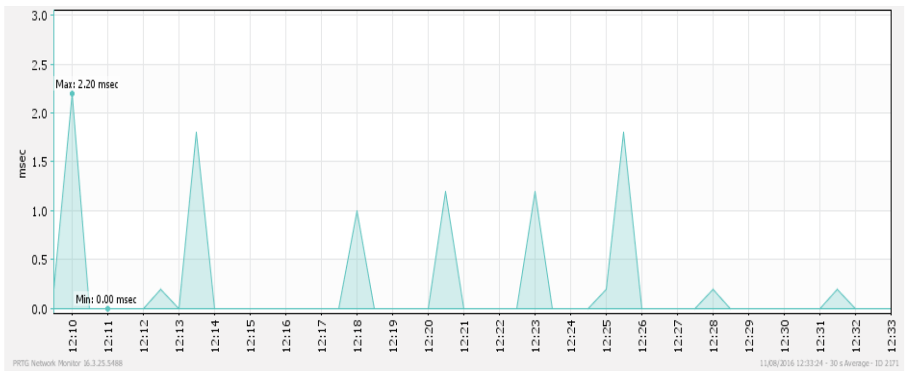


Figure 10: ICMP RTT value pre the scan

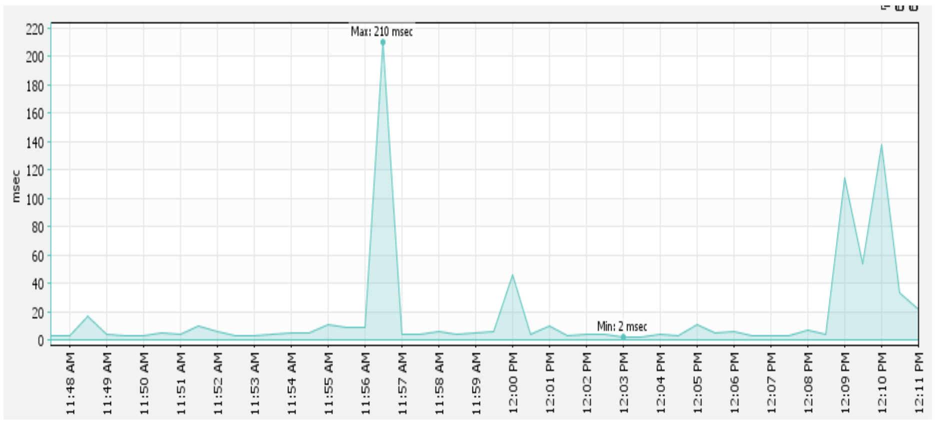


Figure 11: ICMP RTT values during the scan

Viewing the graph in figure 11 clearly displays that the ping time or the time it takes for an ICMP packet to travel back and forth to the destination. The port scan leads widely varying RTT times and the maximum RTT reaches a staggering value of 210 msec. It is obvious that the scan has impacted and degraded the overall network performance where most of the packets sent across the network were encountering large delays.

4.1.1.2. HTTP Latency

Apache 2 Web server is installed on Linux machines. When everything is functioning below the base line set for acceptable traffic level, users on the 192.168.1.0/24 network should access the web server and other resources allocated to them with a minimum delay time. The speed of which HTTP connection can be represented by the responses time, according to Fig 12, the maximum response time for a user to access and retrieve data from the server is 39msec which considered to be an acceptable level of performance since there no port scan or other attacks occurring in the background of the network. Figure 13 illustrates a graph taken from the PRTG monitoring tool which displays how ports scan effects the HTTP response time. This graph was taken when the port scan attack was scanning the network. In contrast to figure 13, the maximum loading time recorded was 39 msec. However, the results were different when the same user accessed the web server during the scan, where the maximum loading time was 63msec.

While the two results do not reflect a considerable delay, this could be ascribed to the fact that we established only one HTTP connection, if we are to simulate the case of a large number of HTTP connections (from the internal network), the delay will be considerably higher

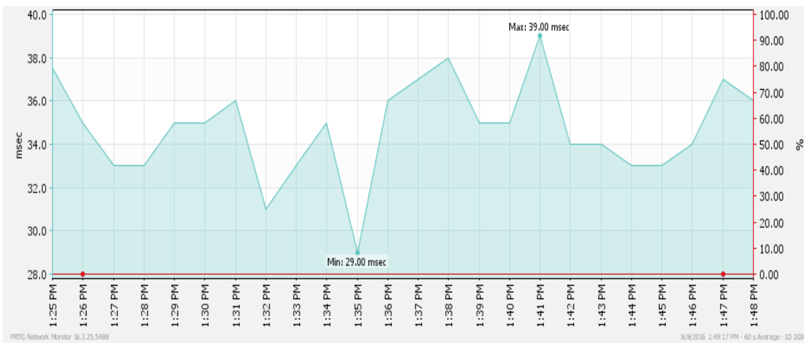


Figure 12: HTTP response time pre the scan

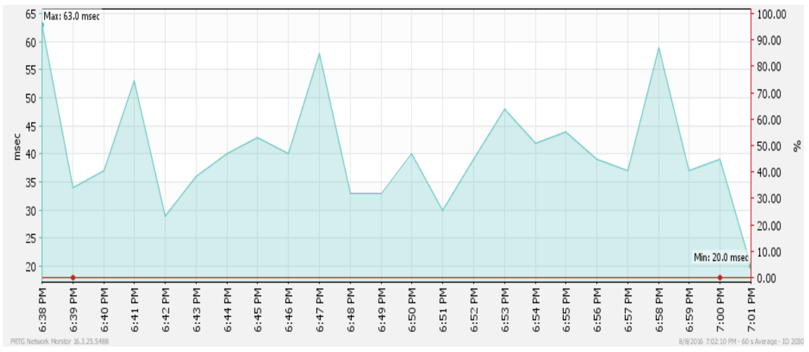


Figure 13: HTTP response time during the scan

4.1.1.3. FTP Latency

A file of 445672KB is downloaded from the FTP server in order to generate FTP traffic across the network and to examine its performance. From figure 14, the maximum response time to the FTP server is 4,208 msec. this value is expected to increase when the port scan test is performed. Figure 15 shows the FTP performance during the port scan. The same file is downloaded from the FTP server to generate some FTP traffic. Because the port scan was conducted, the response time from the FTP server reaches an all-time high of 13,757msec which is almost close to the packet being dropped. It also took longer to download the file from the FTP server compared to the pre scan time.

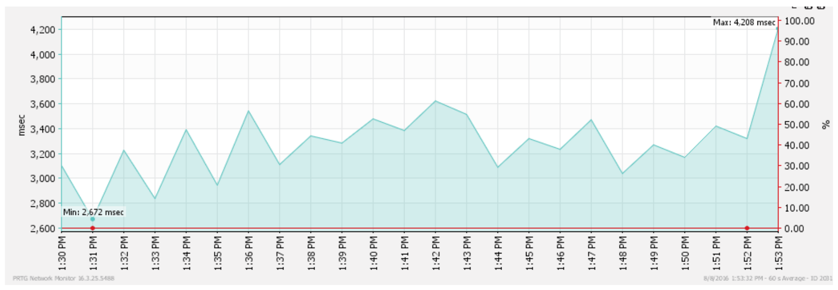


Figure 14: FTP response time pre the scan

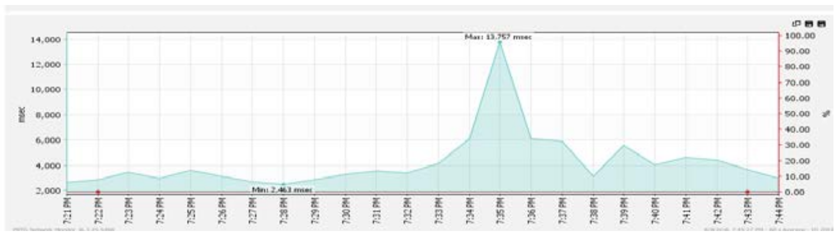


Figure 15: FTP response time during the scan

4.1.2. Analysing Iptables log file with PSAD

Reading log files can be challenging. PSAD is designed to analyse log messages and produce the scan results in a graphical way, which allows users to understand and analyse the source of the scan and its impact in terms of performance. Since PSAD interfaces with Gnuplot to deliver and produce a graph of number of scan packets. From figure 16, we can see how the port scan is impacting the performance levels of the network and ultimately this will cause unwanted delay.

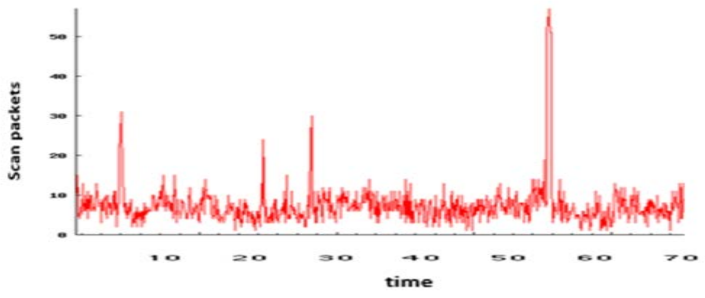


Figure16: analysing Iptables log file with PSAD

4.2. The Second Experiment

In our first experiment we evaluated the impact of a running our scanning script on the network. In this experiment, we will consider different types of port scanning. Therefore, we use Nmap to run a number of scans and then compare between them in terms of accuracy and performance.

4.2.1. Scan Accuracy

The accuracy of information is extremely important in port scanning, as inaccurate information can be misleading for both a penetration tester and attacker. Nmap and other port scanning tools often refer to open ports as filtered or open/filtered which indicates that the scanner is unable to determine the correct state of the scanned port. Although Nmap has exceptionally helpful options to enhance the performance of the scan and provides good scan results, it is crucial not to rely for a single source. Using different scanner tools and techniques does not only improve accuracy but it also allows the scanner to truly determine the state of a particular port

A well-known feature of Nmap is the OS detection which is part of the first stages of mapping out a network. This allows the user to figure out the remote OS version. Administrators often perform fingerprinting in their network to identify any vulnerability and mitigate any possible port scan probes from unauthorised users. Nonetheless, attackers find the OS detection very useful, because it enables them to identify what kind of devices are on the target network. Armed with this information, attackers are then able to make perceptive assumptions and eliminate the need to try vulnerabilities associated with another OS version.

We summarize our scans' result in Table 1. Each scan type prints out the OS details differently, for instance some will display the OS in greater detail while others will printout limited information of the OS. In order to determine the most accurate scan technique, Nmap reveal the user the detection accuracy in form of percentage when the Nmap scan is completed.

Table 1: Port Scanning Accuracy

Scan Technique	Target	Device Type	Vendor	Family/Release	Version	Scan Duration in seconds	Detection Accuracy
Full Connect scan	192.168.1.44	General Purpose	Microsoft	Windows	Win 7	71.08	96%
	192.168.1.45	General Purpose	Linux	Ubuntu	3.14	15.41	95%

	192.168.1.49	General Purpose	Microsoft	Windows	Server 2012	16.03	97%
SYN Scan	192.16..1.44	General purpose	Microsoft	Windows	Win 7	25.41	97%
	192.168.1.45	General purpose	Linux	Ubuntu	3.14	15.13	95%
	192.168.1.49	General purpose	Microsoft	Windows	Server 2012	16.48	96%
Fin Scan	192.168.1.44	General purpose	Microsoft	Windows	Win 7	51.81	95%
	192.168.1.45	General purpose	Linux	Ubuntu	2.4.20, 2.6.14-2.6.32	99.17	93%
	192.168.1.49	General purpose	Microsoft	Windows	Server 2012	46.44	96%
XMAS Scan	192.168.1.44	General purpose	Microsoft	Windows	Win 7	55.95	96%
	192.168.1.45	General purpose	Linux	Ubuntu	2.4.20, 2.6.14-2.6.32	99.09	93%
	192.168.1.449	General purpose	Microsoft	Windows	Server 2012	15.61	95%
Quick Scan	192.168..1.44	General purpose	Microsoft	Windows	Win 7	14.90	94%
	192.168.1.45	General purpose	Linux	3.X	3.11-3.14	15.27	95%
	192.168.1.49	General purpose	Microsoft	Windows	Server 2012	16.22	95%
Intense scan	192.168.1.44	General purpose	Microsoft	Windows	Win 7 ultimate 6.1	75.69	97%
	192.168.1.45	General purpose	Linux	3.X	3.11-3.14	29.86 s	95%
	192.168.1.49	General purpose	Microsoft	Windows	Server 2012 standard 6.2	80.77 s	98%
Slow Comprehensive scan	192.168.1.44	General purpose	Microsoft	Windows	Win 7 ultimate 6.1	43.95 s	96%
	192.168.1.45	General purpose	Linux	Ubuntu	3.11-3.14	30.13 s	97%
	192.168.1.49	General purpose	Microsoft	Windows	Server 2012 standard 6.2	79.44 s	98%

4.2.2. Performance

According to Table 2, the scan duration, raw packets sent, and raw packets received were of various scan techniques were recorded. A total of 11 hosts were scanned ranging from 192.168.1.40 to 192.168.1.50. The goal of this experiment was to identify which scan type will have the lowest impact on the network. It was clear that number of raw packets sent and received had an influence on the level of performance in the network. The TCP connect scan had the least impact in relation to the overall performance and particularly the raw packets sent and received, because this type of scan does not require writing raw packets but alternately uses the connect () system call provided by the operating system. On the other hand, the TCP scan and intense scan have sent the highest raw packets of (1,683 MB).

Table 2: Different scan results

Scan technique	Target	Ports to scan	Scan date & Time	Scan duration	Discovered hosts	Raw-packets sent	Raw-packets received
Full connect	192.168.1.40-50	All ports (*)	27.08.16 11:26	96.94s	8 Hosts	364B	196B
SYN Scan	192.168.1.40-50	All ports (*)	27.08.16 11:29	63.74s	8 Hosts	1.683MB	1.37MB
FIN Scan	192.168.1.40-50	All ports (*)	27.08.16 11:32	51.93s	8 Hosts	1.534MB	1.358MB
XMAS Scan	192.168.1.40-50	All ports (*)	27.08.16 11:35	49.01s	8 Hosts	1.679MB	1.372MB
ACK Scan	192.168.1.40-50	All ports (*)	27.08.16 11:40	167.72.s	8 Hosts	1.534MB	1.363MB
Null Scan	192.168.1.40-50	All ports (*)	27.08.16 11:46	57.8s	8 Hosts	1.539MB	1.358MB
UDP Scan	192.168.1.40-50	All ports (*)	27.08.16 11:49	17675.36s	8 Hosts	4.437MB	6.91MB
Quick Scan	192.168.1.40-50	All ports (*)	27.08.16 18:17	28.74s	8 Hosts	35.900KB	28.716KB
Intense Scan	192.168.1.40-50	All ports (*)	27.08.16 18:23	57.98s	8 Hosts	1.683MB	1.376MB
Slow comprehensive Scan	192.168.1.40-50	All ports (*)	27.08.16 19:21	125.37s	8 Hosts	370.604KB	299.608KB
Aggressive Scan	192.168.1.40-50	All ports (*)	27.08.16 19:45	186.78s	8 Hosts	1.324MB	1.391MB

Nmap scanning tool works fine with all scan types except the UDP scan. As illustrated in Figure 17, the UDP scan duration took very long to complete to the point where that scan was stopped before it finished scanning. It suggested that Nmap is not the right tool for UDP scan as it waits response from each port. If the target port does not respond, the Nmap retries a couple of times before moving on to the next port. This is due to the fact that open and filtered ports barely send any response which forces Nmap to timeout. Other scanners such as Unicorn tend to scan all ports simultaneously and do not wait for a response. Unsurprisingly, it took the aggressive scan 186.78 seconds to finish which makes it the slowest among the scan types.

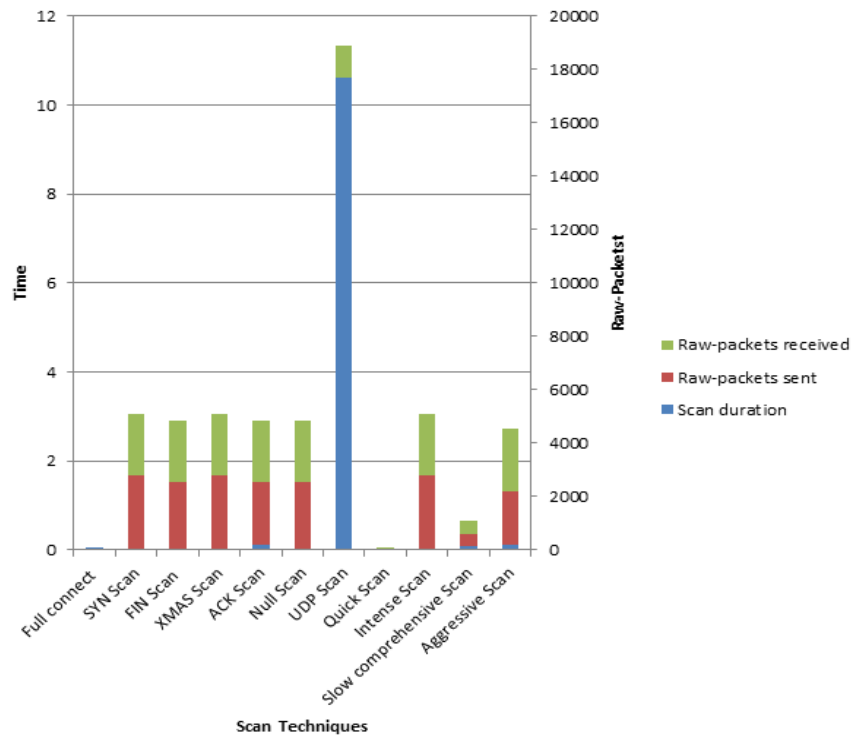


Figure17: Results for different scan techniques

4.2.3.HTTP Response Time (Lowest impact)

The speed in which the HTTP server can be connected is often referred as the HTTP response time and is measured in milliseconds (ms). Using PRTG to monitor the HTTP sensor and examine its performance during the scan process, it was observed that the average web loading times has increased. In Figures 18-21, there are some peaks in the web loading times during the scan where the intense scan had the minimum loading time of (117msec).

- **Slow comprehensive scan**

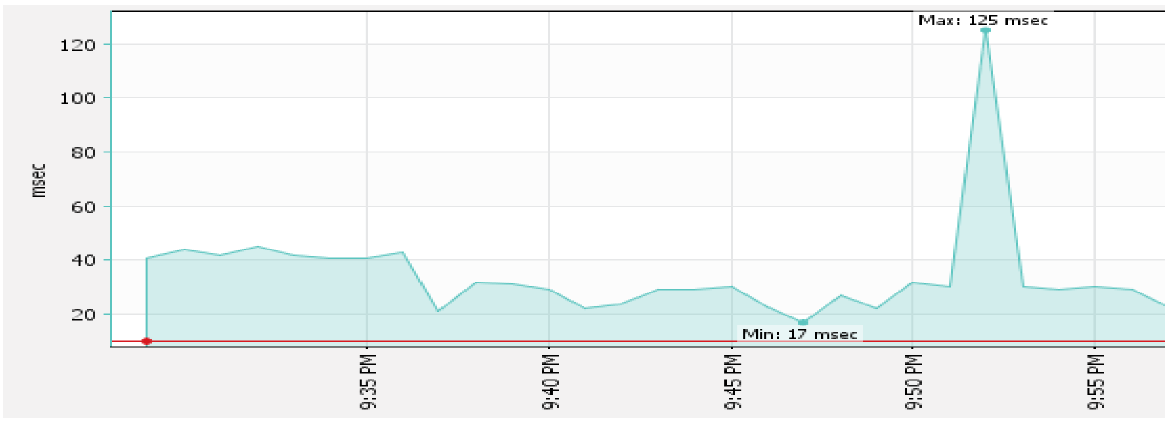


Fig 18: Slow Comprehensive Scan

Null Scan

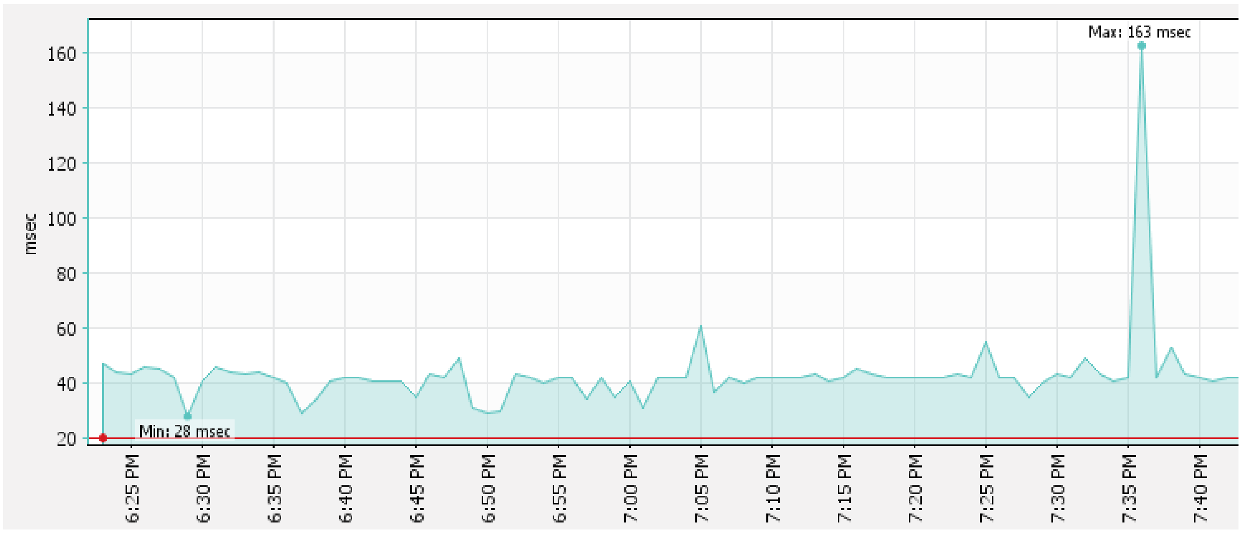


Fig 19: Null Scan

Intense scan

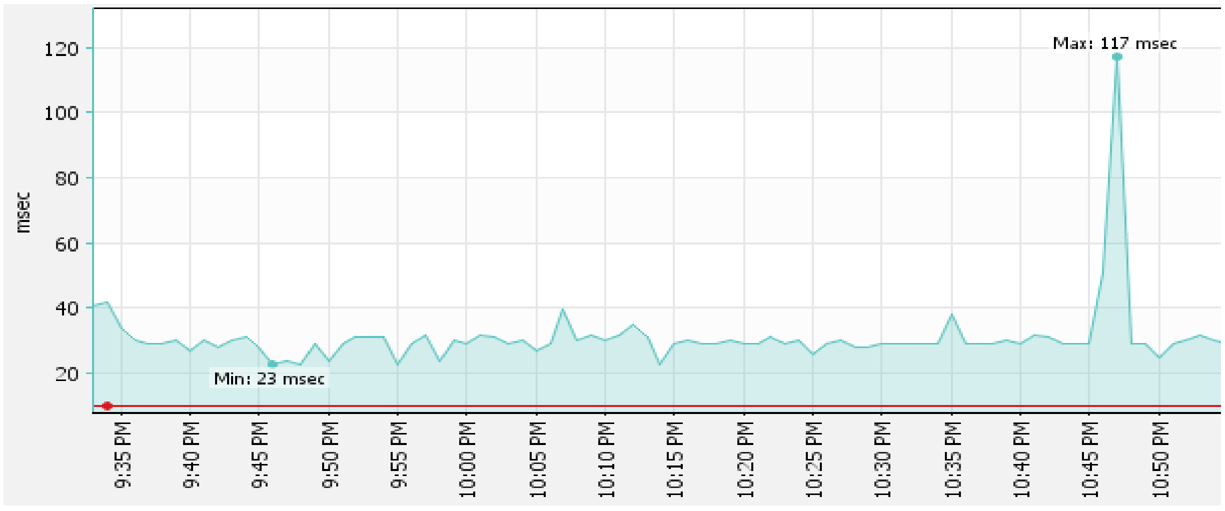


Fig 20: Intense Scan

Quick scan

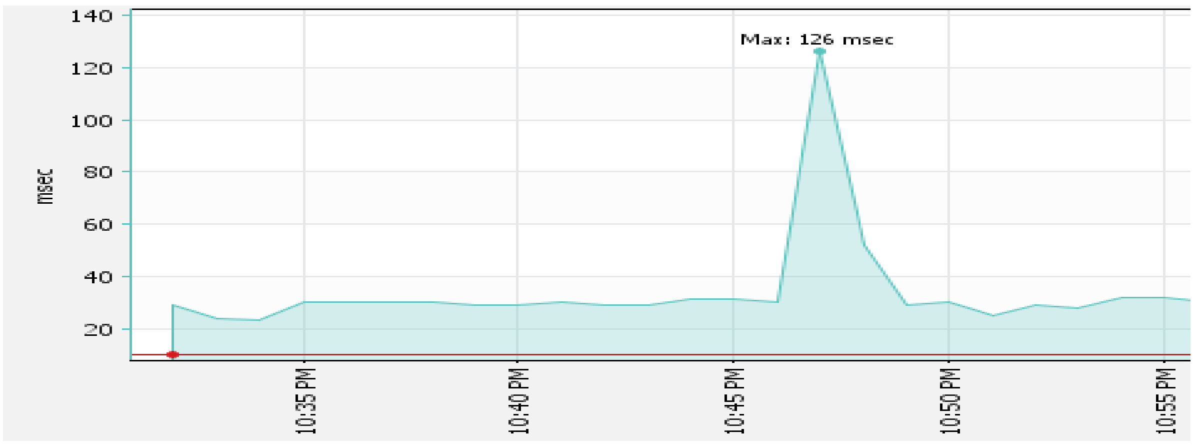
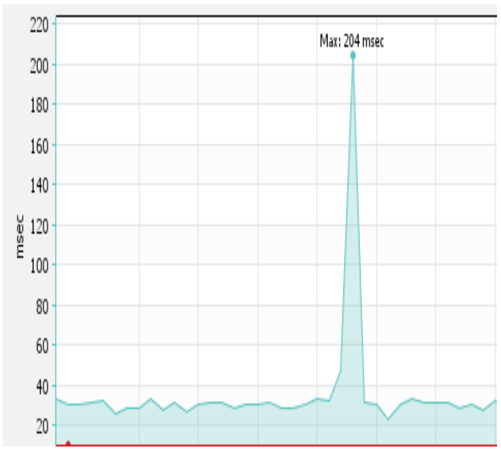


Fig 21: Quick Scan

4.2.4. HTTP Response Time (Highest impact)

During this experiment the scan with the highest impact on the network performance were captured and was concluded that the ACK scan and the aggressive scan seemed to add more overhead in the network than any other scan type. As could be seen in figures 22 and 23 respectively. The maximum response time in the ACK scan is 340 msec, while with aggressive scan the loading time slightly decreases to 204 msec compared to the ACKS scan.

Aggressive scan **ACK scan**



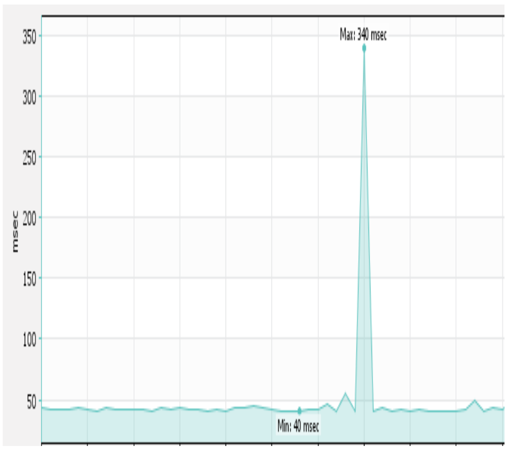


Fig 22: Aggressive Scan. Fig 23: ACK Scan

5. Conclusion and Recommendations:

This report provides an in depth analysis of port scan techniques with extensive information of each scan technique. Two experiments have been conducted. During the first experiment, a TCP scan using a bespoke tool was conducted. The result showed that the network performance was slow during the scan. The scanner was executing the scan from the internet (external network) while the firewall was processing the scan packets and dropping them, the internal network users were encountered an increased access time to the external servers due to the scan. In the second experiment, we compared the accuracy and performance of different types of port scans. The result showed evidently that Nmap is not the right tool to perform UDP scan as open and filtered ports rarely send any response which makes Nmap to timeout before attempting and retrying few more times. The TCP full connect had the least impact in terms of performance considering the packets sent, packets received and scan duration. Because this scan does not need to write raw packets but instead uses the connect () system call of the operating system. However, a drawback for this technique is very easily detected. With regards to the OS detection accuracy, the python port scan script has shown comprehensive information about the target machine, but the goal here was to determine the most accurate technique using Nmap tool. According to the OS detection accuracy results, the Intense and Slow Comprehensive scans were the most accurate with 98% accuracy and have displayed much detailed information compared to other scan types. The author hopes that the results presented in this report will aid security