Information gathering: scanning

"Give me six hours to chop down a tree and I will spend the first four sharpening the axe." -- Abraham Lincoln

License

Contents

General notes about the labs

Often the lab instructions are intentionally open ended, and you will have to figure some things out for yourselves. This module is designed to be challenging, as well as fun!

However, we aim to provide a well planned and fluent experience. If you notice any mistakes in the lab instructions or you feel some important information is missing, please feel free to add a comment to the document by highlighting the text and click the comment icon ( ), and I (Cliffe) will try to address any issues. Note that your comments are public.

If you notice others are also reading the lab document, you can click the chat icon ( ) to discuss the lab with each other.

Preparation

As with all of the labs in this module, start by loading the latest version of the LinuxZ template from the IMS system. If you have access to this lab sheet, you can read ahead while you wait for the image to load.

To load the image: press F12 during startup (on the boot screen) to access the IMS system, then login to IMS using your university password. Load the template image: LinuxZ.

Once your LinuxZ image has loaded, log in using the username and password allocated to you by your tutor.

The root password -- **which should NOT be used to log in graphically** -- is “tiaspbiqe2r” (**t**his **i**s **a** **s**ecure **p**assword **b**ut **i**s **q**uite **e**asy **2** **r**emember). Again, never log in to the desktop environment using the root account -- that is bad practice, and should always be avoided.

Using the VM download script (as described in the previous lab), download and **start** these VMs:

- Kali Linux (Bridged and Host-only networks), user:root password:toor
- Win2K - vulnerable web server (Host-only network)
- Metasploitable 2 (Host-only network)

Note: you don’t need to login to the target VMs (you don’t need to know the passwords for Win2k or Metasploitable), just start the VMs.

Feel free to read ahead while the VMs are downloading.

Note the IP address(es) of the Kali Linux system, using “ifconfig”. Ensure that the VMs are networked as indicated above: that is, they all share a “host only” network, and the Kali Linux VM also has a “bridged” network. If you have downloaded the VMs as instructed, this should already be set correctly.

Introduction to scanning

Scanning is an active phase of an attack or security test, involving identifying IP addresses, ports, and services. Scanning is often conducted by network administrators or security specialists to get an understanding of and and to map out a network. Scanning is also a critical stage for an attacker, since it can give them all the information they need in order to launch an attack. Once an attacker knows the IP address of a service (open port), and the version of the software that the server is running, they can lookup and use known attacks against the target. A well executed scanning stage is extremely important when looking for potential security problems.

Ping sweeps for finding live hosts

Using ping

The Ping command (available on most operating systems) is used for network troubleshooting. It works by sending an Internet Control Message Protocol (ICMP) “echo request”, to which *most* hosts will reply with an ICMP “echo response”.

Read the manual. **From a terminal on your host OS (LinuxZ)**, run:

```
man ping
```

Read the synopsis and description, and press “q” to quit, when you are ready.

Note that the most basic use is “ping *destination*” (where *destination* is the IP address you want to test for a response). Ping your own Kali VM from your local host (with the IP address you noted earlier):

```
ping Kali-IP-address
```

Note that since you haven’t specified how many times to send the echo request, ping keeps going until you press Ctrl-C, at which point ping will present some statistics about how long it took for the remote system (or in this case, the VM) took to reply.

Run a command to send 3 echo requests. Hint: “man ping”, and read through the available options.

On your Kali Linux VM (attacker), note the first three octets of your IP address of the bridged network connection (run “ifconfig”). For example, “192.168.202”. In the Leeds Met IMS labs the third octet corresponds to the room number.

Suppose you want to identify all the live hosts in a given network range (which is often the first step after an attacker has found an IP range using Whois). One way to achieve this is to send an echo request to each IP address, and see which ones reply.

To check all (Class C) IPs starting with “192.168.202” (which can usually be written as 192.168.202.1-254 or 192.168.202.0/24, although the ping command does not understand this):

You could manually run ping for each IP address. Run:

```
ping -c 1 -W 1 192.168.202.1

ping -c 1 -W 1 192.168.202.2

ping -c 1 -W 1 192.168.202.3
ping -c 1 -W 1 192.168.202.4

ping -c 1 -W 1 192.168.202.5
```

You could continue for all 254 IP addresses on this subnet. This is obviously not very efficient... It is easy enough to write your own ping sweep tool as a Bash script.

Creating a ping sweep bash script

Remember, a Bash script is basically a file containing commands that you could otherwise run directly in a Linux Bash shell (terminal) prompt. You may be surprised to learn that you can actually do all kinds of programming tricks at the command prompt, such as writing loops.

Run:

```
vi pingsweep.sh
```

Press “i” to enter insert mode, and enter these lines in the file (you can leave out the (blue) comments, if you like):

```
#!/bin/bash

# if the script is not started with one argument, tell them how it is used
if [ $# -ne 1 ]
then
    # print a message to the screen describing how the script should be used
    echo "Usage: `basename $0` {three octets of IP address, for example 192.168.0}"
    # exit with an error (1)
    exit 1
fi # end of if

# define a variable and set it to the value passed as the first argument ($1)
ip_address_start=$1
# for loop, where “i” starts at 1 and each time increments up to 254
for i in {1..254}
do
    # run ping with one echo request and short timeout for the IP address
    # and pipe to grep to only show if it replies
    ping -c 1 -W 1 $ip_address_start.$i | grep 'from'
done
```

Press “Esc” to exit insert mode, and type “:wq” to write to the file and quit vi.

Note: remember, if you are using a live disk VM, any changes you make will be lost when restarting the VM, so **you may want to save a copy of your script to a USB drive**.

Read the comments (in blue above) to understand this ping sweep script.

Set your script so it can be executed, then run your script:

```
chmod +x pingsweep.sh
```

```
./pingsweep.sh
```

If you get an error message, edit the file and find where you have made a mistake. Check that the spaces around the if statements are exactly as above (that is, "if [$# -ne 1]").

Run your script with the first three octets of your local network's IP address (as you noted earlier):

```
./pingsweep.sh 192.168.202
```

This may take awhile, but should discover the IP addresses of the other hosts in the lab. While you wait for the results, you could open a new console tab (Ctrl-Shift-T) and continue.

How long will this take to complete? Hint: consider what the "-W 1" command argument to ping means. Why does this take so long?

Assuming you have both of the other VMs (Windows 2000 and Metasploitable running), you have two other systems on your host only network.

Use your pingsweep.sh script to discover the IP addresses of the VMs on the host only network. (Hint: use "ifconfig" and identify the IP address assigned to the host only network.)

Nmap

The most popular tool for scanning is, without a doubt, Nmap.

Read the man page:

```
man nmap
```

From the man page:

> "Nmap ("Network Mapper") is an open source tool for network exploration and security auditing. It was designed to rapidly scan large networks, although it works fine against single hosts. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. While Nmap is commonly used for security audits, many systems and network administrators find it useful for routine tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime."
>
> (by Insecure.Com LLC, licensed under the Creative Commons Attribution License version 3.0)
>
> When you are ready, press "q" to quit.

Ping sweeps and host discovery using Nmap

Do a ping sweep of the lab using nmap:

```
nmap -sn -PE 192.168.202.1-254
```

This will be completed fairly quickly, since Nmap does not wait for each response before sending a request to another computer.

Note that often firewalls are configured to block ICMP echo requests. In this case, an ICMP "timestamp request" may still be allowed through, and also reveals whether the host is available.

Nmap's default host discovery does a bit more than just send an echo request. From the man page: "The default host discovery done with -sn consists of an ICMP echo request, TCP SYN to port 443, TCP ACK to port 80, and an ICMP timestamp request by default."

```
nmap -sn 192.168.202.1-254
```

The "-sn" specifies that you don't want Nmap to do a port scan (described later below).

Read the output from the above commands, and note that the the MAC address is also reported. MAC addresses are related to the network interface card, and can also reveal information, since they include prefixes based on the vendor, such as "Cisco Systems" or "Apple".

The responses to the commands above also include DNS name resolution against each IP address. This by itself can be very informative. To only list the hosts along with their hostnames run a “List scan”:

```
nmap -sL 192.168.202.1-254
```

Note that the above List scan does not actually test whether the host is responding.

Use nmap to discover the IP addresses of the VMs *on the host only network*. (Hint: use “ifconfig” and identify the IP addresses on the subnet assigned to the host only network.) You should see three live hosts, one of which will be your Kali Linux VM (the attacker), the other two are your targets.

Note the IP addresses of the two other systems (which are the Metasploitable and Win2k VMs, at this point you won't know which is which).

Ports and port scanning

After establishing a list of live hosts that are targets to attack, the next stage for an attacker or security tester is to examine the *attack surface*; that is, what there is that could be attacked on each of those systems. Any way that a remote computer accepts communication has the potential to be attacked.

All TCP and UDP traffic (which accounts for almost all network traffic over the Internet) involves the use of port numbers to establish which applications are communicating. For example, a server could be running the Apache web server program, which will typically listen to port 80 for connections. Why port 80?: Many different types of servers have standard ports numbers, so for example if I want to access a web page my web browser knows to connect to port 80, while if I want to access email or FTP on the same server, those programs know what port to connect to. The list of all officially registered port numbers can be found here: http://www.iana.org/assignments/port-numbers

On a Linux system there is also a file containing a list of these, plus other common uses:

```
less /etc/services
```

A server *binds* to a port, and listens for connections from a client.

Any open port on a target system is a service that we could test the security of.

Manually you can check whether a port is open on a system, by connecting using Telnet:

```
telnet IP-address 80
```

Where *IP-address* is one of the two host-only network VM IP addresses you noted earlier.

If the connection succeeds you see the message “Connected to [...]” (press “.” and press Enter and the server will reply and then close the connection), if it stays on “Trying [...]” for a while, then the port is not accessible to you (so you know it is probably closed).

Creating a port scanner using a bash script

Using (new versions of) Bash you can easily connect to ports using TCP IP.

For example, **on your local system (LinuxZ):**

Start a netcat listener:

```
nc -l 4444
```

Leave that running, and from another terminal, send a message using the cat command:

```
cat >/dev/tcp/localhost/4444
```

Type in a message, and check that it was sent through. Ctl-D to end (only press it once or you may close your terminal window).

Note: if the above does not work, then your version of Bash does not support making TCP connections. Make sure you are using the LinuxZ host.

You can check whether the command was successful by reading the Bash variable $?:

```
echo $?
```

If you try to connect to a port that is not open:

```
echo >/dev/tcp/localhost/7777
```

The $? will report failure:

```
echo $?
```

What value does $? have when the connection succeeds?

What value does $? have when the connection fails?

Now that we know how to check for open ports using Bash, we can create our own Bash script port scanner:

```
vi portscanner.sh
```

Press “i” to enter insert mode, and enter these lines in the file (you can leave out the blue comments, if you like):

```
#!/bin/bash

if [ $# -ne 1 ]
then
    echo "Usage: `basename $0` {IP address or hostname}"
    exit 1
fi
# define a variable and set it to the value passed as the first argument ($1)
ip_address=$1
# write the current date to the output file
echo `date` >> $ip_address.open_ports
# for loop, where “i” starts at 1 and each time increments up to 65535
for port in {1..65535}
do
    # use a short timeout, and write to the port on the IP address
    timeout 1 echo >/dev/tcp/$ip_address/$port
    # if that succeeded (checks the return value stored in $?)
    if [ $? -eq 0 ]
    then
        # append results to a file named after the date and host
        echo "port $port is open" >> "$ip_address.open_ports"
    fi
done
```

Press “Esc” to exit insert mode, and type “:wq” to write to the file and quit vi.

You may want to save a copy of your script to a USB drive.

Set your script so it can be executed, then run your script:

```
chmod +x portscanner.sh

./portscanner.sh
```

Run your script on your own system:

```
./portscanner.sh localhost
```

This may take awhile, but should discover all the open ports, and create a file in your working directory (where you are) with a list of open ports.

Have a look at the resulting file:

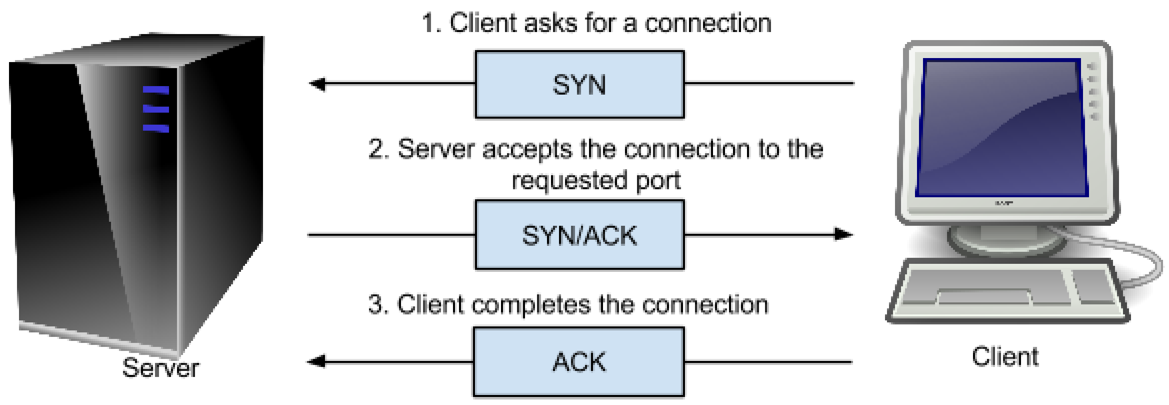
```
less localhost.open_ports
```

Run your script on another IP address in the lab.

Extra challenge: write your own port scanner in another programming language of your choice. Look online, there are lots of examples you can use for help.

TCP three-way handshake

Many port scanners are quite simple, and do more than is strictly necessary to know whether a port is open. In order to establish a TCP connection and start sending and receiving data, a full three way handshake takes place.


1. Client asks for a connection
SYN
2. Server accepts the connection to the requested port
SYN/ACK
3. Client completes the connection
ACK
Server
Client

TCP three-way handshake

As illustrated above, 1) the client starts by sending a TCP packet that includes the port that it wants to talk to (say port 80), and the SYN flag set, meaning that it wants to start a new connection. 2) If the server accepts the connection (there is indeed a program listening on that port), then it responds with the SYN and ACK flags set, meaning it accepts the connection. 3) The client can now complete the connection by sending a packet with the ACK flag set.

If the port is closed it will send a RST at stage 2. If there is a firewall that is filtering connections to the port, then the client may not receive any reply.

SYN port scans

Notice that if all we are interested in is whether the port is open, we can skip the third step, since step 2 already tells us whether the port is open. This approach to port scanning is known as a SYN scan. However, this requires the ability to write to the raw network directly rather than using libraries that establish connections for us. On Linux this means the program needs some extra privileges, such as run by the root user.

Nmap port scanning

Nmap is primarily a port scanner, and is quite possibly the most advanced port scanner that exists. It has many port scanning features, such as the ability to do SYN scans or full TCP connect scans, but also can do some advanced analysis.

On your local system (LinuxZ):

Alternatively, you can run each of these nmap scans from Kali Linux, but may not find any ports open on localhost.

Run a simple nmap port scan against your own system:

```
nmap localhost
```

Have a look through and make sure you understand the output.

Again from the Nmap man page: “The state is either open, filtered, closed, or unfiltered. Open means that an application on the target machine is listening for connections/packets on that port. Filtered means that a firewall, filter, or other network obstacle is blocking the port so that Nmap cannot tell whether it is open or closed. Closed ports have no application listening on them, though they could open up at any time. Ports are classified as unfiltered when they are responsive to Nmap's probes, but Nmap cannot determine whether they are open or closed. Nmap reports the state combinations open|filtered and closed|filtered when it cannot determine which of the two states describe a port.”

Nmap port scanning techniques

“As a novice performing automotive repair, I can struggle for hours trying to fit my rudimentary tools (hammer, duct tape, wrench, etc.) to the task at hand. When I fail miserably and tow my jalopy to a real mechanic, he invariably fishes around in a huge tool chest until pulling out the perfect gizmo which makes the job seem effortless. The art of port scanning is similar. Experts understand the dozens of scan techniques and choose the appropriate one (or combination) for a given task. Inexperienced users and script kiddies, on the other hand, try to solve every problem with the default SYN scan. Since Nmap is free, the only barrier to port scanning mastery is knowledge. That certainly beats the automotive world, where it may take great skill to determine that you need a strut spring compressor, then you still have to pay thousands of dollars for it." -- Nmap man page.

Open the Nmap man page ("man nmap") and scroll down to the title "PORT SCANNING TECHNIQUES". Read the entries for:

- -sS (TCP SYN scan)
- -sT (TCP connect scan)
- -sU (UDP scans)
- -sN; -sF; -sX (TCP NULL, FIN, and Xmas scans)

Run a SYN scan on your own system:

```
sudo nmap -sS localhost
```

Now, based on what you have learned:

1. Run a SYN scan against both of the two host-only network IP addresses you noted earlier.
2. Run a UDP scan against the IP addresses. Any interesting results? Why are UDP scans unreliable?
3. Run a Xmas scan against each of the two VM IP addresses. Did both work? Why?
4. Which of these two IP addresses do you think is the Win2k and which is the Metasploitable VMs?
5. What services appear to be running on each of the VMs?

You may wish to read about the other port scanning techniques that Nmap supports.

Nmap port specification

By default Nmap scans 1000 "interesting ports" (of the possible 65535); however, you can tell Nmap which ports to scan.

Open the Nmap man page ("man nmap") and scroll down to the title "PORT SPECIFICATION AND SCAN ORDER". Read the entries for:

- -p *port ranges* (Only scan specified ports)
- -F (Fast (limited port) scan)
- -r (Don't randomize ports)

Run:

```
nmap -p 80-85 IP-address
```

Where *IP-address* is one of the VM IP addresses.

Look through the output and note which ports are open and which are closed.

Based on what you have learned:

6. Run a SYN scan against both of the VMs, but only checking port 80.
7. Run a TCP connect scan against both of the VMs, checking only the very most likely ports.
8. Run a SYN scan against one of the VMs, checking every possible port (this may take a while).
9. Run a SYN scan against both of the VMs, but don't randomise the port order during scanning.

Nmap timing and performance

By default Nmap scans 1000 "interesting ports" (of the possible 65535); however, you can override this and tell Nmap which ports to scan.

Open the Nmap man page ("man nmap") and scroll down to the title "TIMING AND PERFORMANCE". Read the entry for:

- -T paranoid|sneaky|polite|normal|aggressive|insane (Set a timing template)

Run:

```
nmap -T5 IP-address
```

Where *IP-address* is one of the VM IP addresses.

Based on what you have learned:

10. Run a SYN scan of ports 80-85 against one of the VMs, using the "sneaky speed".

11. Why would anyone choose this speed?
12. Run a scan of all possible ports as fast as possible.
13. Why might it not be a good idea to choose "insane" speed for scans across the Internet?

Service identification

In order to know what attacks could be successful against a system, we usually need to know more than which ports are open. For example, knowing that port 80 is open tells us that some software on the other system is listening to that port number, and that we can connect to that port, and can probably send data to it, and may get a reply. Since port 80 is almost always used by web servers hosting websites, it is very likely that the software on the other side is Apache, IIS, or one of the many other web servers available. However, in order to know what types of attacks are likely to succeed, we want to know as much as we can about the software running on the server.

Banner grabbing

The simplest way of determining what software is running on a port, is to connect and check whether the response tells us what software is running. Many services present a banner whenever a new connection is established, which often states what software version the server is running. Collecting this information is known as *banner grabbing*.

Manually connect to port 21 on each of the VMs' IP addresses you noted earlier:

```
nc IP-address 21
```

Netcat (nc) is similar to Telnet, in that it can be used for manual raw TCP communication.

Press Ctrl-C to kill the processes, after you receive each response.

Note the software version this reports (repeat for each of the two IP addresses you noted earlier).

Manually connect to port 80 on each of the VMs' IP addresses you noted earlier:

```
nc IP-address 80
```

(enter "." and press the Enter key a few times)

Note that the output from a webserver often contains the line "Server:"... Use this to note the software on port 80 for each VM IP, if available.

You can actually talk to the server and request web pages, by sending the commands that a web browser sends:

```
nc IP-address 80
```

(type "**GET / HTTP/1.0**" then press Enter a few times) This may be more likely include a "Server:" response.

Based on what you have learned:

14. What is port 21 used for?
15. What software and versions of software are running on those systems?
16. Can you use this to figure out which IP address is what each of the VMs?

Automated banner grabbing

If there are a number of ports to banner grab from, it is more efficient to use an automated approach.

Netcat can be used to banner grab over a range:

```
nc IP-address 1-2000 -w 1
```

The above command will connect to each port from 1 to 2000 (with a timeout of 1 second), and display any responses.

Extra challenge: update your port scanner (the above bash script or your own one in another language) with your own code to do banner grabbing. Hint: you want to read from each port, rather than write to them.

Protocol analysis and fingerprinting

Q: Consider this: Why can't you trust this information?

A: A problem with banner grabbing is that the software could be lying! Just because a web server is telling you that it is Apache or IIS, does not guarantee that it is.

The solution to this problem (from an attacker/security tester's perspective) is the use of protocol analysis / fingerprinting of the service to determine what software is listening to the port. The way this works is by interacting with the server, sending different kinds of requests (also known as *triggers*), and comparing the way the server responds with a database of fingerprints.

The software that made this a popular security testing approach was Amap. Amap has two main features: banner grabbing (using the "-B" flag), and protocol analysis (using "-A").

On your attacking system (Kali Linux):

Run Amap protocol analysis on each of the VMs web servers (repeat for each IP address):

```
amap -A IP-address 80
```

This will tell you what protocol is in use on the port (HTTP), and an idea of what software is used.

17. Can you use this to figure out which IP address is what each of the VMs?

Note that Amap is now somewhat outdated, and has been superseded by Nmap's service and version detection. Some security testers prefer to use both, to compare results.

Open the Nmap man page ("man nmap") and scroll down to the title "SERVICE AND VERSION DETECTION". Read the text after the title, and the entry for:

- -sV (Version detection)

Run Nmap service and version detection on each of the VMs web servers (repeat for each IP address):

```
nmap -sV -p 80 IP-address
```

Attempt detection for the 1000 ports that are scanned by default (repeat for each IP address):

```
nmap -sV IP-address
```

Based on what you have learned:

18. What web servers (and versions) are running on the two VMs?

19. What other services are running?

Operating system detection

Knowing the versions of software on a remote system is often enough to start looking for vulnerabilities. However, another important piece of information is knowing what operating system it is running. We need to know that if we want to launch an exploit (so we can choose the correct payload), or just to get a better idea of whether that version of the software listening to the port is actually vulnerable.

Nmap can do some clever analysis of the way the system communicates to detect the operating system running on the remote system. The official RFC (request for comment) specifications that define the way protocols such as TCP works contains some ambiguity (not 100% clear or prescriptive), so each OS works slightly differently in the way it handles network packets. Nmap detects the OS by sending lots of specially crafted packets to open and closed port on the system, and analyses the response.

Open the Nmap man page ("man nmap") and scroll down to the title "OS DETECTION". Read the text after the title, and the entry for:

- -O (Enable OS detection)

Run Nmap OS detection on each of the VMs web servers (repeat for each IP address):

```
nmap -O IP-address
```

Some other important Nmap features

We will also cover some other advanced Nmap features in other labs.

Nmap and output

Having the output to your screen is obviously helpful, but usually you will want to keep a record of Nmap's findings, to look back over later, or to import into another security program for reporting, vulnerability analysis (looking for vulnerabilities), or even exploitation (attack). Here are a few of the main ways that Nmap can save output:

- -oN filespec (normal output)
  Essentially saves a copy of the output in a named file
- -oX filespec (XML output)
  Saves an XML document containing the results. This is the most useful format for importing the results into another program.

- -oG filespec (grepable output)
  Saves to a very simple column format, but is deprecated (that is, not encouraged for use, since it may be removed from later versions).

- -oA basename (Output to all formats)
  Saves all of the above.

You may wish to read the man page for more details about the above features.

Run Nmap on each of the VMs and save the normal output:

```
nmap -oN output IP-address
```

Have a look at the output file:

```
less output
```

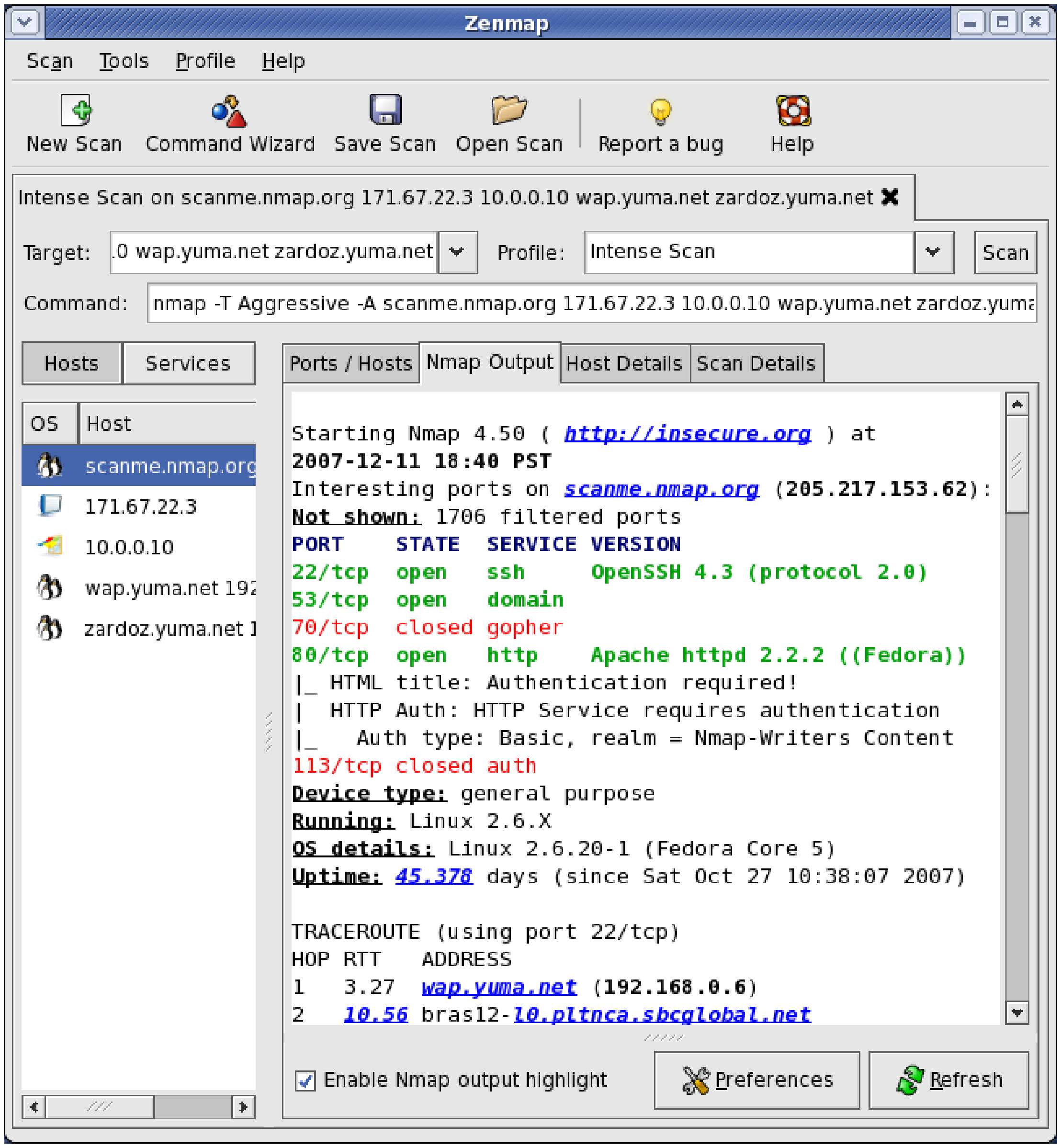
Based on what you have learned:

20. For each IP, run a scan with OS and version detection and output to all three output formats

Ensure you have saved a copy (on USB or via email) of the scan results of the two VMs in XML format for next week's lab.

Nmap GUIs

There are various graphical user interface (GUI) front-ends for Nmap, including the official Zenmap, which is illustrated in the figure below. Essentially the gui can help to create the command for starting Nmap, and then runs Nmap and displays the results. The gui can be helpful for beginners, and for saving scanning profiles, although most experts will generally prefer the command line.



Command: nmap -T Aggressive -A scanme.nmap.org 171.67.22.3 10.0.0.10 wap.yuma.net zardoz.yuma

```
Starting Nmap 4.50 ( http://insecure.org ) at
2007-12-11 18:40 PST
Interesting ports on scanme.nmap.org (205.217.153.62):
Not shown: 1706 filtered ports
PORT     STATE  SERVICE VERSION
22/tcp   open   ssh     OpenSSH 4.3 (protocol 2.0)
53/tcp   open   domain
70/tcp   closed gopher
80/tcp   open   http    Apache httpd 2.2.2 ((Fedora))
|_ HTML title: Authentication required!
|  HTTP Auth: HTTP Service requires authentication
|_   Auth type: Basic, realm = Nmap-Writers Content
113/tcp closed auth
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.20-1 (Fedora Core 5)
Uptime: 45.378 days (since Sat Oct 27 10:38:07 2007)

TRACEROUTE (using port 22/tcp)
HOP RTT   ADDRESS
1   3.27  wap.yuma.net (192.168.0.6)
2   10.56 bras12-10.pltnca.sbcglobal.net
```

Zenmap: a GUI for Nmap

Open challenge: install Zenmap in the Installed Kali Linux VM (“`apt-get install zenmap`”), and start Zenmap with “`zenmap &`”

Alternatively, load the “Kali Linux - with Armitage” VM, which also includes Zenmap.

Tip: note that the ampersand tells bash to start Zenmap in the background, so you can continue to run commands at the terminal while it is running. Alternatively you can start another tab with Ctrl-Shift-Tab, if you are using Konsole.

You may like to experiment with Zenmap's features, and run some scans on your VMs, or other systems in the lab. Note the output provided in the "Host Details" and "Scan Details" tabs.

What next?

Now that you know what software is running on the host (even down to the version of the software running), you now have everything you need to plan an attack on the system. From this point there may be more advanced methods of extracting information about the system (known as enumeration), or you may have gathered information that suggests the system is likely vulnerable to certain attacks.

Conclusion

At this point you have:

- learned about ping sweeps, and how they work (and even worked with Bash code for a ping sweep script)
- learned about port scanning, run a script, and used many important port scanning methods and command line options using Nmap
- performed banner grabbing manually and automatically using protocol analysis
- experimented with other features of Nmap (the most used scanner), such as file output, and used a graphical frontend for Nmap

Congratulations! This was a lengthy lab, but these concepts are very important, and these are essential stages in security testing.

When you have some time, it is highly recommended to try the extra challenge of writing your own port scanner from scratch in a programming language of your choice (such as Python, Perl, Java, or C).

Reminder: ensure you have saved a copy (on USB or via email) of the scan results of the two VMs in XML format for next week's lab.