UDP Scan Using nmap

In this lab, you will use the nmap -sU option to perform a UDP scan.

With this scan type, nmap sends 0-byte UDP packets to each port on the target system. Receipt of an ICMPv4 Destination Unreachable/Port Unreachable (Type 3, Code 3) message signifies the port is closed; otherwise it is assumed open.

One major problem with this technique is that when a firewall blocks outgoing ICMPv4 Type 3, Code 3 messages, the port will appear open. These false-positives are hard to distinguish from real open ports.

Another disadvantage with UDP scanning is the speed at which it can be performed. Most OSes limit the number of ICMPv4 Type 3, Code 3 messages which can be generated in a certain time period, thus slowing the speed of a UDP scan. nmap adjusts its scan speed accordingly to avoid flooding a network with useless packets.

NOTE: Microsoft OSes do not limit the ICMPv4 Type 3, Code 3 error generation frequency, thus, making it is easier to scan a Windows machine's 65,535 UDP ports in very little time.

Exercise 1:

1. From a BackTrack shell, type the following (only type what's in **bold**):

```
user1@pentest:~# nmap -sU -v target_IP_address > /root/ceh/udpscan
```

Syntax breakdown:

nmap: program name

-sU: program option for UDP scan

-v: verbose mode

target_IP_address: the IP address of the target system

> /root/ceh/udpscan: redirect the output to a file called udpscan in the /root/ceh directory

2. Examine your results:

```
user1@pentest:~#cat /root/ceh/udpscan | less
```

3. Enable DNS server and SNMP service in your victim machine (Windows Server 2003) and repeat step#1-2 and write down your results. (If you do not know how to enable DNS AND SNMP services, research on it.)

4. Repeat steps #1-2 using different target IP addresses. Compare your results.

TCP Connect Scan Using nmap

Exercise 1: TCP Connect scan:

1.From a BackTrack shell, type the following (only type what's in **bold**):

```
user1@pentest:~#nmap -sT -vv target_IP_address > /root/ceh/connect_scan
```

Syntax breakdown:

nmap: program name

-sT: program option for TCP connect scan

-vv: program option for double verbose output

target_IP_address: the IP address of the target system

> /root/ceh/connect_scan: redirect the output to a file called connect_scan in the /root/ceh directory

2. Examine your results:

```
user1@pentest:~#cat /root/ceh/connect_scan | less
```

3.Record your results:

```
PORT     STATE                    SERVICE
--------  ----------               --------------
```

4.Enable IIS, FTP, Messenger, Telnet, POP3, and SMTP in your victim machine (Windows Server 2003) and repeat step#1-2 and record your results. (if you do not know how to enable IIS, FTP, Messenger, Telnet, POP3, and SMTP, research on it)

TCP SYN Scan Using nmap

Let's build on this common port scan so that your actions are *stealthier*.

In the next example, you will add two additional options: the -g and the –p options:

The *-g option* specifies the source port on the scanning machine (your system).

The most common option to use here is port 80 (because it's allowed through most

border network devices). The *-p option* sets the ports on the target system to scan.

Exercise 1: TCP SYN scan:

1. Using the -sS flag option, you will send the target system a SYN packet, but upon receiving the SYN/ACK from the target system, your system will not respond with an ACK packet (thus, completing the 3-way handshake). Instead, your system will respond with a RST packet (the port is then considered open). If you receive an RST/ACK packet from the target system, it indicates that there is not a running process on that port, or the process running on the port is not listening for connections.

2. The syntax to perform a TCP SYN scan is (only type what's in **bold**, on one line):

   user1@pentest:~#**nmap -sS -vv -g 80 -p 80,88,135,139,389,445 target_IP_address > /root/ceh/syn_scan**

Syntax breakdown:

nmap: program name

-sS: program option for TCP SYN scan

-vv: program option for double verbose output

-g 80: program option that specifies the source port on the scanning machine

(your system)

-p 80,88,135,139,389,445: specifies the ports on the target system to scan

target_IP_address: the IP address of the target system

> /root/ceh/syn_scan: redirect the output to a file called syn_scan in the /root/ceh directory

3. Examine your results:

   user1@pentest:~#**cat /root/ceh/syn_scan | less**

4. Record your results:

```
PORT     STATE           SERVICE
--------  ----------      --------------
```

5. Repeat steps #2-3 using a different target IP address and different destination port numbers

TCP SYN Scans Using hping

Hping is a command-line-oriented packet-crafting program that allows you to control specific options of the UDP, ICMP, TCP, or raw IP packet. hping can also be used for the following:

- Test firewall rules
- Advanced port scanning
- Test network performance using different protocols, packet size, TOS (type of service) and fragmentation
- Path MTU discovery
- Transferring files
- Route tracing under different protocols
- Firewalk-like usage
- Remote OS fingerprinting, and
- TCP/IP stack auditing

In this lab, you will use hping to perform a few basic port scans.

Exercise 1: TCP SYN scan:

1.From a BackTrack shell, type the following (only type what's in **bold**):

user1@pentest:~# **hping2 target_IP_address -S -p 135 > /root/ceh/hping_out**

Syntax breakdown:

hping2: program name

target_IP_address: the IP address of the target system

-S: SYN flag set

-p 135: scan for TCP port 135

> /root/ceh/hping_out: redirect the output to a file called *hping_out* located in the /root/ceh directory

2. After the scan runs for approximately 5 seconds, hit Ctrl+C to stop it

3. To view the results of this scan type the following:

user1@pentest:~# **cat /root/ceh/hping_out | less**

NOTE: An open port is indicated by an SA (SYN/ACK) return packet. A closed port is indicated by an RA (RST/ACK) packet.

Exercise 2: TCP SYN scan (range of addresses):

1. In this exercise, you will scan all well-known ports and redirect the output to a file called *hping_out1* in the /tmp directory. The resulting file will list the response (if any) from all well-known ports. Notice the *-V option* in the following syntax (only type what's in **bold**, on one line):

user1@pentest:~# **hping2 target_IP_address -S --scan 0-1023 -V > /root/ceh/hping_out1**

Syntax breakdown:

hping2: program name

target_IP_address: the IP address of the target system

-S: SYN flag set

--scan 0-1023: scan well-known ports

-V: see all the replies

> /root/ceh/hping_out1: redirect the output to a file called *hping_out1* located in the /root/ceh directory

2. To view the results of this scan type the following:

   user1@pentest:~# **cat /root/ceh/hping_out1 | less**

3. What ports are open on the target system?

Exercise 3: TCP SYN scan (range of addresses) viewing response from open ports:

1. In this exercise, you will scan all well-known ports again, and redirect the output to a file called *hping_out2* in the /tmp directory. In addition, you will eliminate the *-V option*, which will show results only for open ports (only type what's in **bold**, on one line):

   user1@pentest:~# **hping2 target_IP_address -S --scan 0-1023 > /root/ceh/hping_out2**

   Syntax breakdown:

   hping2: program name

   target_IP_address: target system's IP address

   -S: SYN flag set

   --scan 0-1023: scan well-known ports

   > /root/ceh/hping_out2: redirect the output to a file called *hping_out2* located in the /root/ceh directory

2. To view the results, type the following:

   user1@pentest:~# **cat /root/ceh/hping_out2 | less**

3. What ports are open on the target system?

4. Compare this with the output in Exercise #2, #3. Are they the same? Should they be? Why or why not.

Decoy Scan Using nmap

Exercise 1: in this exercise, you'll use nmap to perform a decoy scan, which mixes your IP address with bogus IP addresses (using the -D option):

1. From a BackTrack shell, type the following (only type what's in **bold**, on one line):

   user1@pentest:~#**nmap -n -D192.168.1.5,10.5.1.2,me,172.1.2.4 target_IP_address > /root/ceh/decoy_scan**

Syntax breakdown:

nmap: program name

-n: program option to never resolve DNS names

-D192.168.1.5,10.5.1.2,me,172.1.2.4: program option to cloak your scan with the supplied bogus and real IP addresses

target_IP_address: the IP address of the target system

> /root/ceh/decoy_scan: redirect the output to a file called decoy_scan in the /root/ceh directory

2. Examine your results:

   user1@pentest:~#**cat /root/ceh/decoy_scan | less**