Port Scanning with Nmap

23 October 2016 • 14 mins read

• Reconnaissance • Port scanning • Nmap • TCP 3-way handshake • TCP scan • UDP scan • Metasploitable 2 • Network packet analysis • Wireshark

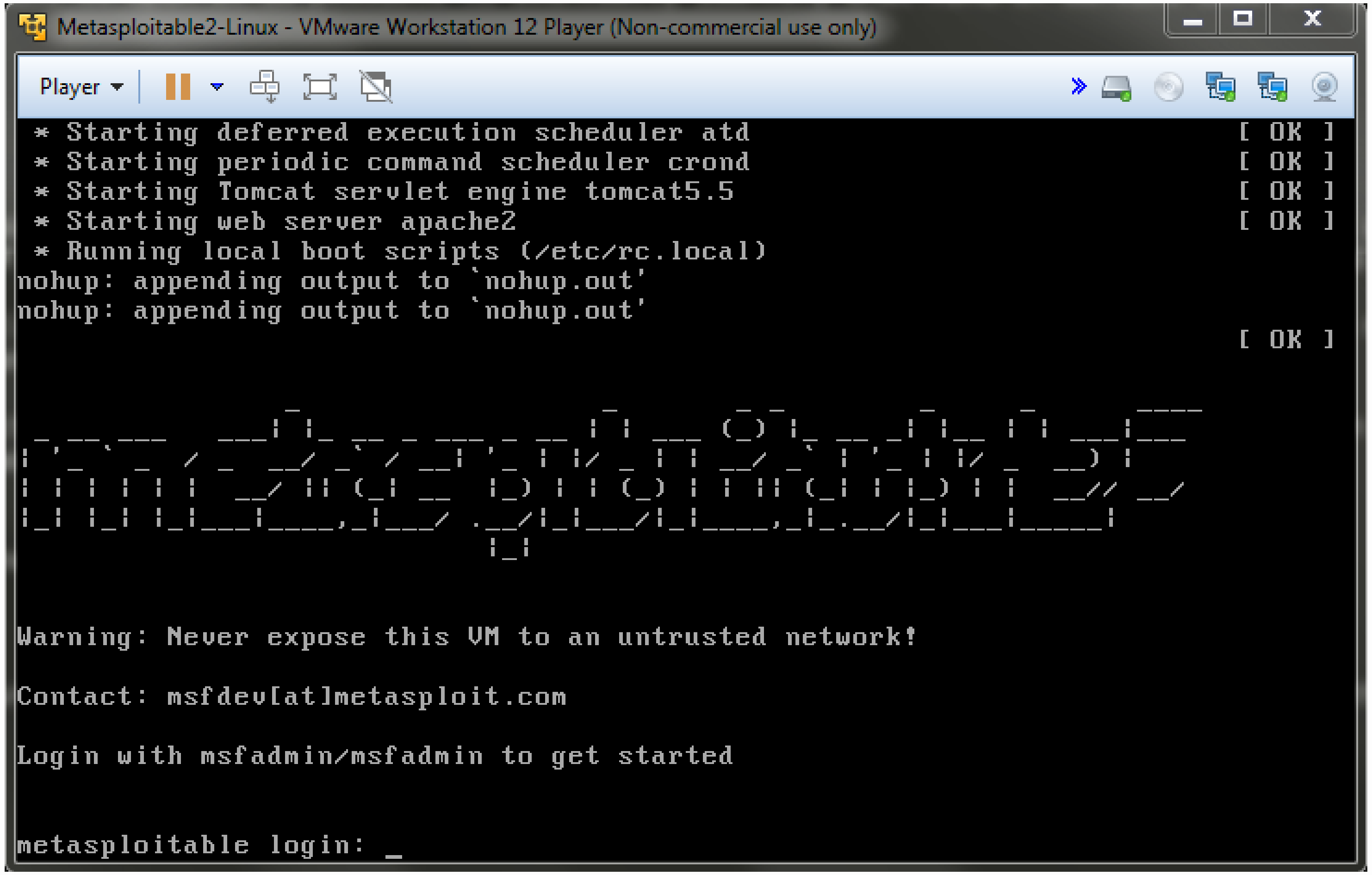
Port scanning is a technique used to identify if a port on the target host is open or closed; a port can be open if there is a service that uses that specific port to communicate with other systems. This is the reason why if a port is open it is possible to eventually identify what kind of service uses it by sending specially crafted packets to the target.
This activity represents an important step in the active reconnaissance phase.

Setting up a testing environment

Since we need a target against which we can launch our port scanning attacks, we need to create a very basic testing laboratory that includes our attacking machine, in my case Kali Linux, and a target machine: for this lab session I suggest to download a VM called Metasploitable 2. This system, created by Metasploit team, has been build intentionally vulnerable to a series of attacks by exposing compromised services through open ports.

The virtual machine can be downloaded for free from here; because of its nature, do not expose this VM on the Internet, i.e. be sure to run it in a local network environment behind a Router Firewall.

Once the VM image is downloaded it is just a matter of extracting the files from the archive and import it in your Hypervisor: for example, in VMware Workstation Player click on “Open a Virtual Machine”, select the extracted Metasploitable 2 image and you are ready to launch it. If everything has been done correctly you should get this terminal prompt:


Metasploitable2-Linux - VMware Workstation 12 Player (Non-commercial use only)
Player

```
 * Starting deferred execution scheduler atd                            [ OK ]
 * Starting periodic command scheduler crond                            [ OK ]
 * Starting Tomcat servlet engine tomcat5.5                             [ OK ]
 * Starting web server apache2                                          [ OK ]
 * Running local boot scripts (/etc/rc.local)
nohup: appending output to `nohup.out'
nohup: appending output to `nohup.out'
                                                                        [ OK ]


                _                  _       _ _        _     _      ____
 _ __ ___   ___| |_ __ _ ___ _ __ | | ___ (_) |_ __ _| |__ | | ___|___ \
| '_ ` _ \ / _ \ __/ _` / __| '_ \| |/ _ \| | __/ _` | '_ \| |/ _ \ __) |
| | | | | |  __/ || (_| \__ \ |_) | | (_) | | || (_| | |_) | |  __// __/
|_| |_| |_|\___|\__\__,_|___/ .__/|_|\___/|_|\__\__,_|_.__/|_|\___|_____|
                            |_|


Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started


metasploitable login: _
```

Remember to configure it so as its IP address is in the same LAN of the attacking machine (in my case the LAN is 192.168.1.0/24).

Port scanning Metasploitable 2

Nmap, which we have already analyzed for Network Discovery in this topic, is the most famous tool for port scanning: by sending probes to the target it is able to find which ports are open and which services are running on them (this is just one of its capabilities).

We can start by taking a look at the huge list of scanning options:

```
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
  -p <port ranges>: Only scan specified ports
    Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
  --exclude-ports <port ranges>: Exclude the specified ports from scanning
  -F: Fast mode - Scan fewer ports than the default scan
  -r: Scan ports consecutively - don't randomize
  --top-ports <number>: Scan <number> most common ports
  --port-ratio <ratio>: Scan ports more common than <ratio>
SERVICE/VERSION DETECTION:
  -sV: Probe open ports to determine service/version info
  --version-intensity <level>: Set from 0 (light) to 9 (try all probes)
  --version-light: Limit to most likely probes (intensity 2)
  --version-all: Try every single probe (intensity 9)
  --version-trace: Show detailed version scan activity (for debugging)
SCRIPT SCAN:
  -sC: equivalent to --script=default
  --script=<Lua scripts>: <Lua scripts> is a comma separated list of
           directories, script-files or script-categories
  --script-args=<n1=v1,[n2=v2,...]>: provide arguments to scripts
  --script-args-file=filename: provide NSE script args in a file
  --script-trace: Show all data sent and received
  --script-updatedb: Update the script database.
  --script-help=<Lua scripts>: Show help about scripts.
           <Lua scripts> is a comma-separated list of script-files or
           script-categories.
OS DETECTION:
  -O: Enable OS detection
  --osscan-limit: Limit OS detection to promising targets
  --osscan-guess: Guess OS more aggressively
TIMING AND PERFORMANCE:
  Options which take <time> are in seconds, or append 'ms' (milliseconds),
  's' (seconds), 'm' (minutes), or 'h' (hours) to the value (e.g. 30m).
  -T<0-5>: Set timing template (higher is faster)
  --min-hostgroup/max-hostgroup <size>: Parallel host scan group sizes
  --min-parallelism/max-parallelism <numprobes>: Probe parallelization
  --min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <time>: Specifies
      probe round trip time.
  --max-retries <tries>: Caps number of port scan probe retransmissions.
  --host-timeout <time>: Give up on target after this long
  --scan-delay/--max-scan-delay <time>: Adjust delay between probes
  --min-rate <number>: Send packets no slower than <number> per second
  --max-rate <number>: Send packets no faster than <number> per second
FIREWALL/IDS EVASION AND SPOOFING:
  -f; --mtu <val>: fragment packets (optionally w/given MTU)
  -D <decoy1,decoy2[,ME],...>: Cloak a scan with decoys
  -S <IP_Address>: Spoof source address
  -e <iface>: Use specified interface
  -g/--source-port <portnum>: Use given port number
  --proxies <url1,[url2],...>: Relay connections through HTTP/SOCKS4 proxies
  --data <hex string>: Append a custom payload to sent packets
  --data-string <string>: Append a custom ASCII string to sent packets
  --data-length <num>: Append random data to sent packets
  --ip-options <options>: Send packets with specified ip options
  --ttl <val>: Set IP time-to-live field
  --spoof-mac <mac address/prefix/vendor name>: Spoof your MAC address
  --badsum: Send packets with a bogus TCP/UDP/SCTP checksum
OUTPUT:
  -oN/-oX/-oS/-oG <file>: Output scan in normal, XML, s|<rIpt kIddi3,
     and Grepable format, respectively, to the given filename.
  -oA <basename>: Output in the three major formats at once
  -v: Increase verbosity level (use -vv or more for greater effect)
  -d: Increase debugging level (use -dd or more for greater effect)
  --reason: Display the reason a port is in a particular state
  --open: Only show open (or possibly open) ports
  --packet-trace: Show all packets sent and received
  --iflist: Print host interfaces and routes (for debugging)
  --append-output: Append to rather than clobber specified output files
  --resume <filename>: Resume an aborted scan
  --stylesheet <path/URL>: XSL stylesheet to transform XML output to HTML
  --webxml: Reference stylesheet from Nmap.Org for more portable XML
  --no-stylesheet: Prevent associating of XSL stylesheet w/XML output
```

```
MISC:
  -6: Enable IPv6 scanning
  -A: Enable OS detection, version detection, script scanning, and traceroute
  --datadir <dirname>: Specify custom Nmap data file location
  --send-eth/--send-ip: Send using raw ethernet frames or IP packets
  --privileged: Assume that the user is fully privileged
  --unprivileged: Assume the user lacks raw socket privileges
  -V: Print version number
  -h: Print this help summary page.
EXAMPLES:
  nmap -v -A scanme.nmap.org
  nmap -v -sn 192.168.0.0/16 10.0.0.0/8
  nmap -v -iR 10000 -Pn -p 80
SEE THE MAN PAGE (https://nmap.org/book/man.html) FOR MORE OPTIONS AND EXAMPLES
```

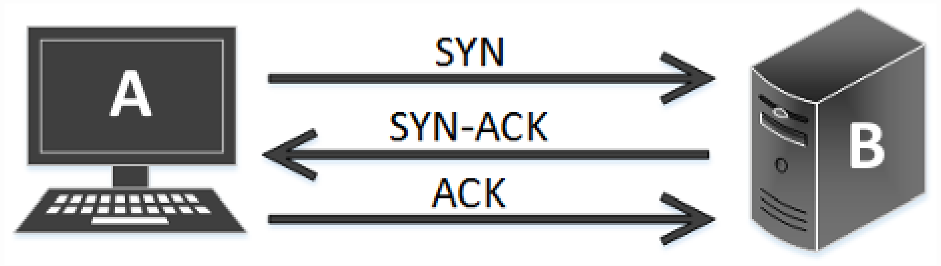
Even if we will not go through all of them, we will analyze the most useful ones.

The first thing we need to know to start a scan is the IP address of the target host. We could easily acquire that by simply logging into the just launched Metasploitable VM and then using `ifconfig` command, but, since we want to mimic the actions of a real attacker that has no direct access to the machine, we have to move in a different way.
Remembering the previous post we can either use Nmap or Netdiscover for this matter: `nmap -sn 192.168.1.0/24` or `netdiscover -r 192.168.1.0/24` will do the job. We discover that Metasploitable 2 machine has IP address 192.168.1.100 (by default the IP address is assigned by DHCP, but we can set a static IP address modifying `/etc/network/interfaces`).

Once we know the target IP address we can launch the port scanning attack. By default, if no option is selected, Nmap runs a TCP SYN Scan also known as Stealth Scan.

TCP 3-Way Handshake

To understand this type of scan it can be useful to refresh the TCP 3-way handshake theory which represents the way a TCP connection starts:



When a system A wants to establish a connection with another system B, A sends a SYN (Synchronize) packet to B; when B receives the message it sends back to A a SYN-ACK (Synchronize-Acknoledgement). Once A receives that signal, it sends to B an ACK. Finally B receives the ACK and the TCP socket connection is established.

TCP Scan

A TCP SYN Scan works this way: system A, that represents our attacking machine, sends to the target system B the SYN and waits for the SYN-ACK. If B responds, which means the port is open, A does not send the final ACK. If A does not receive the SYN-ACK the port can be either closed or filtered (this can indicate the presence of a Firewall). In this way we have performed a TCP port scan without establishing a full connection with the target.
Resuming and detailing:

- Open port: A sends SYN to B and B responds with SYN-ACK;
- Closed port: A sends SYN to B and B responds with RST-ACK (Reset-Acknoledgement);
- Filtered port: A sends SYN to B, but does not receive a response or receives an ICMP port unreachable error message.

Even if this type of scan is the default one, we can set it up with the “-sS” parameter followed by the IP address of the target:

```
root@kali:~# nmap -sS 192.168.1.100

Starting Nmap 7.30 ( https://nmap.org ) at 2016-10-22 20:07 CEST
Nmap scan report for 192.168.1.100
Host is up (0.00020s latency).
Not shown: 977 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
23/tcp   open  telnet
25/tcp   open  smtp
53/tcp   open  domain
80/tcp   open  http
111/tcp  open  rpcbind
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
512/tcp  open  exec
513/tcp  open  login
514/tcp  open  shell
1099/tcp open  rmiregistry
1524/tcp open  ingreslock
2049/tcp open  nfs
2121/tcp open  ccproxy-ftp
3306/tcp open  mysql
5432/tcp open  postgresql
5900/tcp open  vnc
6000/tcp open  X11
6667/tcp open  irc
8009/tcp open  ajp13
8180/tcp open  unknown
```

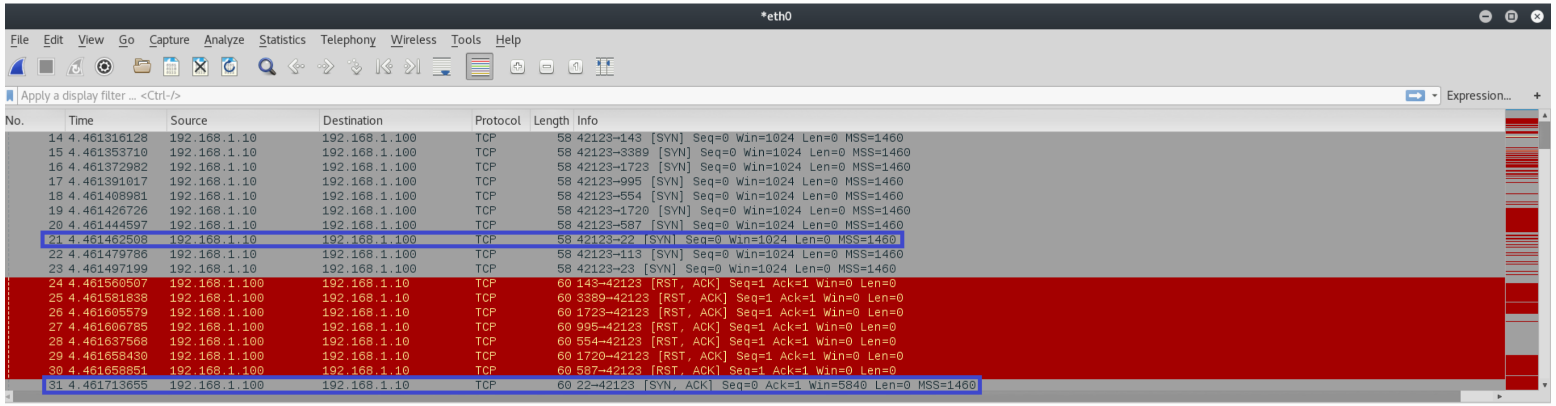
```
MAC Address: 00:0C:29:59:72:BC (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.20 seconds
```

Nmap, if not specified differently, sets the scan to probe the most common 1000 ports and goes through them in a random way (“-r” option for scanning ports consecutively). As we can see from the results, we have scanned 1000 ports in 0.20 seconds and 977 of them are reported as closed; for the opened ones, Nmap gives us information about the service that is running on them.

Traffic analysis with Wireshark

We can inspect Nmap SYN Scan activity with Wireshark by launching it and then running the port scan:


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

No.	Time	Source	Destination	Protocol	Length	Info
14	4.461316128	192.168.1.10	192.168.1.100	TCP	58	42123→143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
15	4.461353710	192.168.1.10	192.168.1.100	TCP	58	42123→3389 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
16	4.461372982	192.168.1.10	192.168.1.100	TCP	58	42123→1723 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
17	4.461391017	192.168.1.10	192.168.1.100	TCP	58	42123→995 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
18	4.461408981	192.168.1.10	192.168.1.100	TCP	58	42123→554 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
19	4.461426726	192.168.1.10	192.168.1.100	TCP	58	42123→1720 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
20	4.461444597	192.168.1.10	192.168.1.100	TCP	58	42123→587 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
21	4.461462508	192.168.1.10	192.168.1.100	TCP	58	42123→22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
22	4.461479786	192.168.1.10	192.168.1.100	TCP	58	42123→113 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
23	4.461497199	192.168.1.10	192.168.1.100	TCP	58	42123→23 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
24	4.461560507	192.168.1.100	192.168.1.10	TCP	60	143→42123 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
25	4.461581838	192.168.1.100	192.168.1.10	TCP	60	3389→42123 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
26	4.461605579	192.168.1.100	192.168.1.10	TCP	60	1723→42123 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
27	4.461606785	192.168.1.100	192.168.1.10	TCP	60	995→42123 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
28	4.461637568	192.168.1.100	192.168.1.10	TCP	60	554→42123 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
29	4.461658430	192.168.1.100	192.168.1.10	TCP	60	1720→42123 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
30	4.461658851	192.168.1.100	192.168.1.10	TCP	60	587→42123 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
31	4.461713655	192.168.1.100	192.168.1.10	TCP	60	22→42123 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460

In the above image it is clear the attacking machine probes target ports by sending SYN packets: for example, we can look at the packet number 21 which probes port 22 (SSH service) and receives a SYN-ACK at packet number 31 meaning the port is open. The ones highlighted in red are RST-ACK meaning that specific port is closed.

A more aggressive port scan

We can add some parameters to acquire additional details about the version of services running on those open ports (“-sV”) and to identify the Operating System (“-O”). We can also specify we want to scan not only the top 1000 ports, but all of them (“-p 1-65535”). Finally we can set up a time aggressive scan with “-T4”, since we do not care of being detected in a testing lab environment (the less the number after “T” the bigger the time between the probes). Of course, because this is a more complex scan, it will take more time to execute w.r.t the previous one:

```
root@kali:~# nmap -sV -O -T4 -p 1-65535 192.168.1.100

Starting Nmap 7.30 ( https://nmap.org ) at 2016-10-22 20:10 CEST
Nmap scan report for 192.168.1.100
Host is up (0.00017s latency).
Not shown: 65505 closed ports
PORT      STATE SERVICE     VERSION
21/tcp    open  ftp         vsftpd 2.3.4
22/tcp    open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet      Linux telnetd
25/tcp    open  smtp        Postfix smtpd
53/tcp    open  domain      ISC BIND 9.4.2
80/tcp    open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec        netkit-rsh rexecd
513/tcp   open  login?
514/tcp   open  tcpwrapped
1099/tcp  open  rmiregistry GNU Classpath grmiregistry
1524/tcp  open  shell       Metasploitable root shell
2049/tcp  open  nfs         2-4 (RPC #100003)
2121/tcp  open  ftp         ProFTPD 1.3.1
```

```
3306/tcp  open  mysql       MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd     distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  X11         (access denied)
6667/tcp  open  irc         Unreal ircd
6697/tcp  open  irc         Unreal ircd
8009/tcp  open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
8787/tcp  open  drb         Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drb)
43534/tcp open  status      1 (RPC #100024)
45768/tcp open  nlockmgr    1-4 (RPC #100021)
52690/tcp open  mountd      1-3 (RPC #100005)
58138/tcp open  unknown
MAC Address: 00:0C:29:59:72:BC (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
Service Info: Hosts:  metasploitable.localdomain, localhost, irc.Metasploitable.LAN;

OS and Service detection performed. Please report any incorrect results at https://n
Nmap done: 1 IP address (1 host up) scanned in 152.14 seconds
```

As reported above, we have details about services version: for example, we now know that the ftp server running on Metasploitable 2 is vsftpd (Very Secure FTPD) version 2.3.4. Moreover we got the Operating System which is Linux and the Kernel version detected as 2.6.x (between 2.6.9 and 2.6.33).

Keep in mind that an aggressive scan is more likely to be detected and there is also the possibility that it brings down the service that runs on the corresponding scanned port.

UDP Scan

Until now we have performed TCP port scan. UDP scan is really different since UDP is a connectionless protocol. It can happen that even if a UDP port is open it might not respond to any received UDP packet. During a UDP scan the attacker machine sends a UDP packet to the target port: if the port is open the attacker machine receives a response; if the port is closed Nmap receives an ICMP port unreachable message. If the attacker machine does not receive any response there are two possibilities: the port is open but the service is not responding to Nmap probes or the traffic is filtered due to the presence of a Firewall.

A UDP scan can be launched with the option “-sU”:

```
root@kali:~# nmap -sU 192.168.1.100

Starting Nmap 7.30 ( https://nmap.org ) at 2016-10-22 20:14 CEST
Nmap scan report for --- (192.168.1.100)
Host is up (0.00021s latency).
Not shown: 994 closed ports
PORT     STATE         SERVICE
53/udp   open          domain
69/udp   open|filtered tftp
111/udp  open          rpcbind
137/udp  open          netbios-ns
138/udp  open|filtered netbios-dgm
2049/udp open          nfs
MAC Address: 00:0C:29:59:72:BC (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1082.85 seconds
```

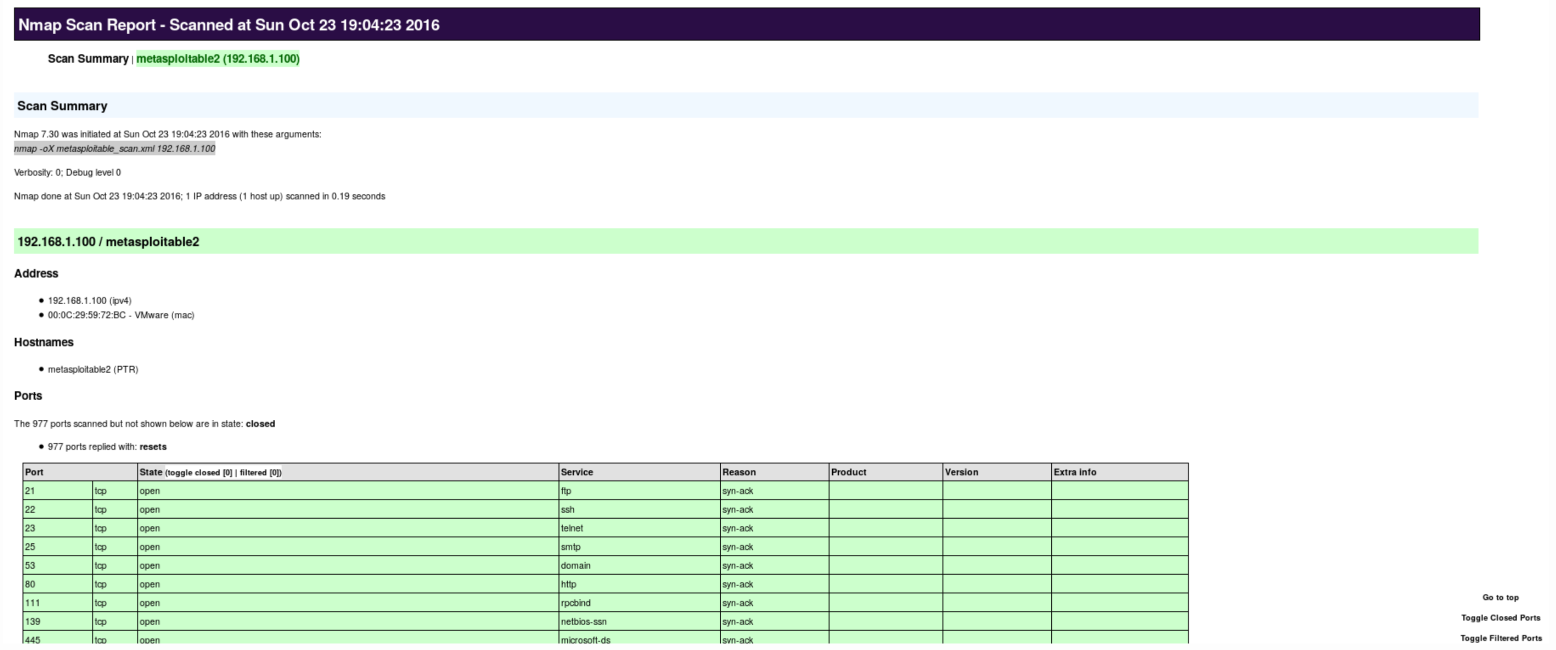
As reported, we have scanned 1000 ports and 994 of them are identified as closed. As stated before, when Nmap does not receive any response it classifies the port either as open or filtered. Moreover UDP scan is time consuming: 1082.85 seconds, which are about 18 minutes.

Reporting

An interesting feature is the possibility to report results into different formats like XML. In particular, it is useful to convert XML results into HTML files, so they can be displayed in your favourite Web Browser. The following command generates the XML report, process it and gives as output the HTML file:

```
root@kali:~# nmap 192.168.1.100 -oX metasploitable_scan.xml && xsltproc metasploitab
```

And this is the HTML report:



Nmap Scan Report - Scanned at Sun Oct 23 19:04:23 2016

Scan Summary | metasploitable2 (192.168.1.100)

Scan Summary

Nmap 7.30 was initiated at Sun Oct 23 19:04:23 2016 with these arguments:
nmap -oX metasploitable_scan.xml 192.168.1.100

Verbosity: 0; Debug level 0

Nmap done at Sun Oct 23 19:04:23 2016; 1 IP address (1 host up) scanned in 0.19 seconds

192.168.1.100 / metasploitable2

Address

- 192.168.1.100 (ipv4)
- 00:0C:29:59:72:BC - VMware (mac)

Hostnames

- metasploitable2 (PTR)

Ports

The 977 ports scanned but not shown below are in state: **closed**

- 977 ports replied with: **resets**

Port		State (toggle closed [0] | filtered [0])	Service	Reason	Product	Version	Extra info
21	tcp	open	ftp	syn-ack			
22	tcp	open	ssh	syn-ack			
23	tcp	open	telnet	syn-ack			
25	tcp	open	smtp	syn-ack			
53	tcp	open	domain	syn-ack			
80	tcp	open	http	syn-ack			
111	tcp	open	rpcbind	syn-ack			
139	tcp	open	netbios-ssn	syn-ack			
445	tcp	open	microsoft-ds	syn-ack			

Conclusions

We have seen how Nmap can perform both TCP and UDP port scan; it is a really good tool to identify open ports on the target system and which services are running on them. We have also seen how it is possible to discover target services versions and O.S. through a more agressive scan. The results reporting capability has been showed with an example of HTML report.
There are a lot of other interesting features inside this tool and we will take a look at them in the next lab sessions.