ARYAMAN MISHRA 19BCE1027

LAB 6

EX-6-METASPLOIT INFORMATION GATHERING USING NMAP

PREPARING METASPLOIT FOR PORT SCANNING

Scanners and most other auxiliary modules use the ‘RHOSTS’ option instead of ‘RHOST’. RHOSTS can take IP ranges (192.168.1.20-192.168.1.30), CIDR ranges (192.168.1.0/24), multiple ranges separated by commas (192.168.1.0/24, 192.168.3.0/24), and line-separated host list files (file:/tmp/hostlist.txt). This is another use for a grepable Nmap output file.

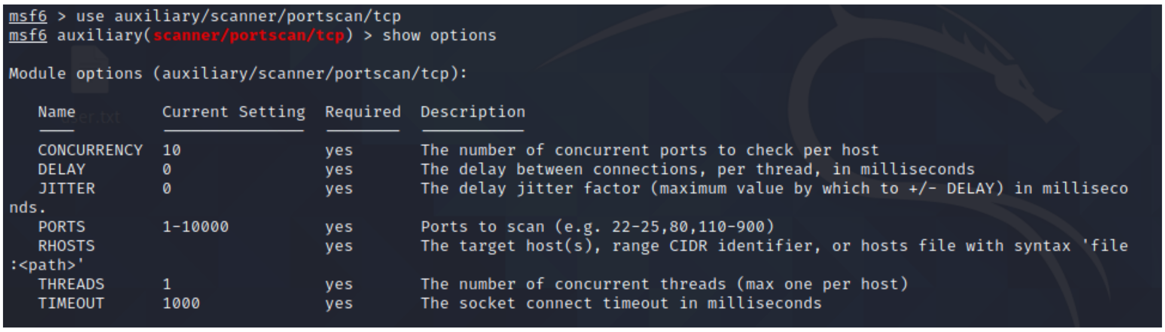
By default, all of the scanner modules will have the ‘THREADS’ value set to ‘1’. The ‘THREADS’ value sets the number of concurrent threads to use while scanning. Set this value to a higher number in order to speed up your scans or keep it lower in order to reduce network traffic but be sure to adhere to the following guidelines:

- Keep the THREADS value under 16 on native Win32 systems
- Keep THREADS under 200 when running MSF under Cygwin
- On Unix-like operating systems, THREADS can be set as high as 256.

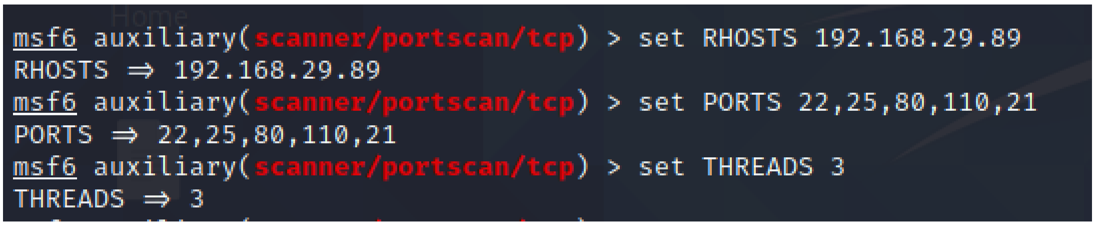
NMAP & DB_NMAP

We can use the **db_nmap** command to run Nmap against our targets and our scan results would than be stored automatically in our database. However, if you also wish to import the scan results into another application or framework later on, you will likely want to export the scan results in XML format. It is always nice to have all three Nmap outputs (xml, grepable, and normal). So we can run the Nmap scan using the **-oA** flag followed by the desired filename to generate the three output files, then issue the **db_import** command to populate the Metasploit database.

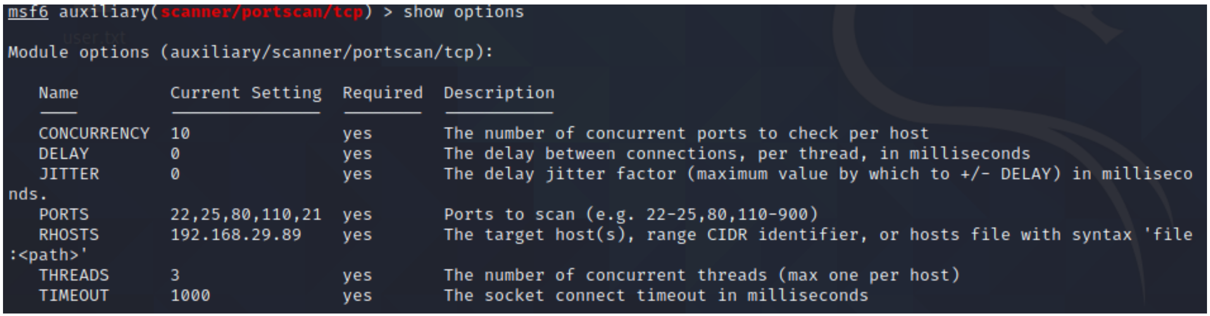
Run Nmap with the options you would normally use from the command line. If we wished for our scan to be saved to our database, we would omit the output flag and use **db_nmap**. The example below would then be **db_nmap -v -sV 192.168.1.0/24**.



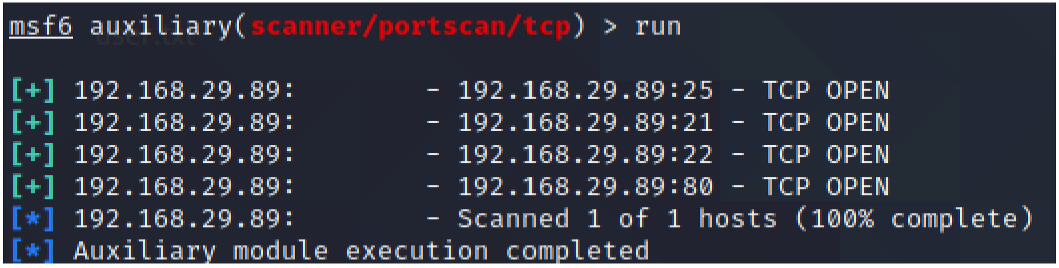
```
msf6 > use auxiliary/scanner/portscan/tcp
msf6 auxiliary(scanner/portscan/tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CONCURRENCY  10               yes       The number of concurrent ports to check per host
   DELAY        0                yes       The delay between connections, per thread, in milliseconds
   JITTER       0                yes       The delay jitter factor (maximum value by which to +/- DELAY) in milliseco
nds.
   PORTS        1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS                        yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file
:<path>'
   THREADS      1                yes       The number of concurrent threads (max one per host)
   TIMEOUT      1000             yes       The socket connect timeout in milliseconds
```



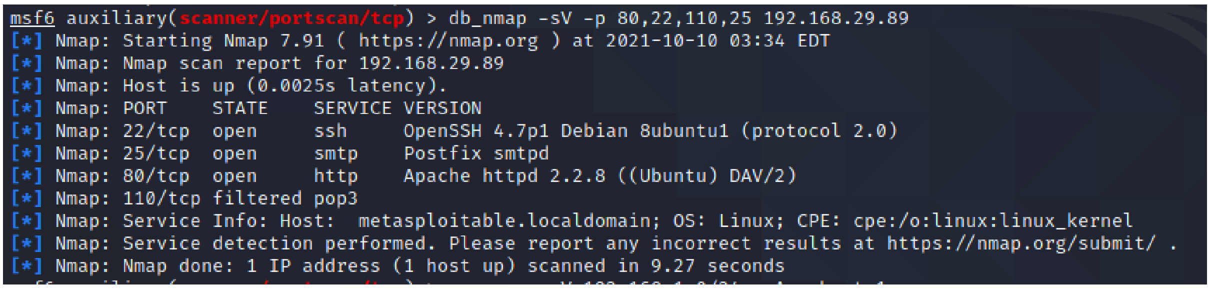
```
msf6 auxiliary(scanner/portscan/tcp) > set RHOSTS 192.168.29.89
RHOSTS ⇒ 192.168.29.89
msf6 auxiliary(scanner/portscan/tcp) > set PORTS 22,25,80,110,21
PORTS ⇒ 22,25,80,110,21
msf6 auxiliary(scanner/portscan/tcp) > set THREADS 3
THREADS ⇒ 3
```



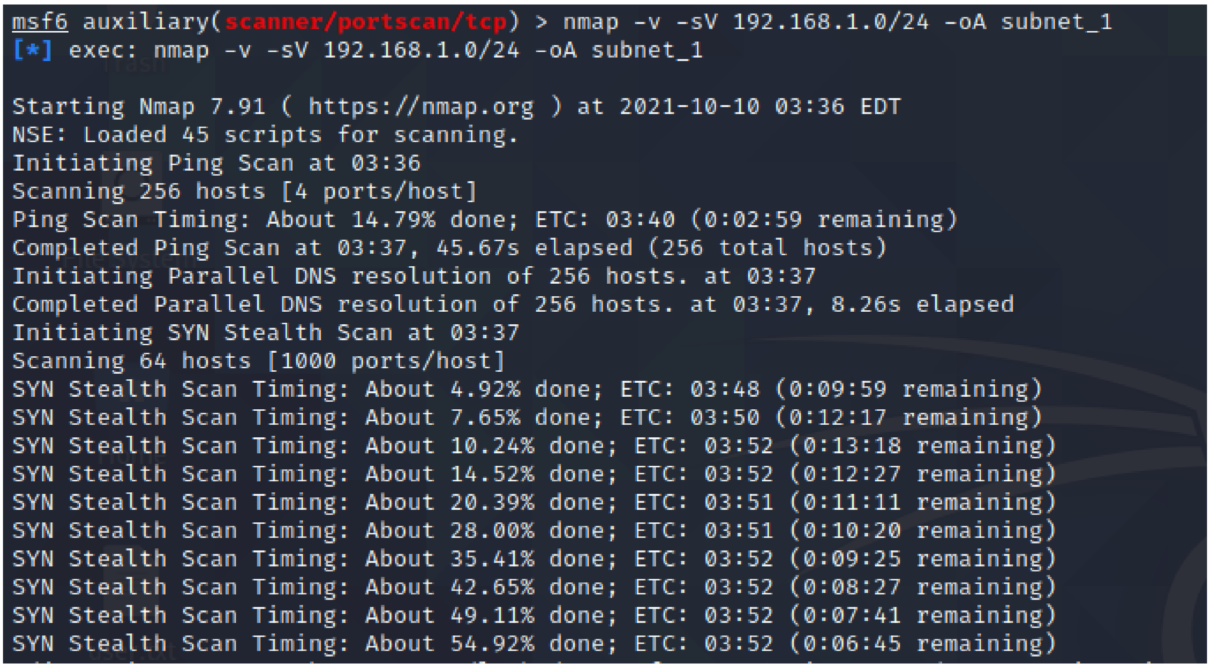
```
msf6 auxiliary(scanner/portscan/tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting   Required  Description
   ----         ---------------   --------  -----------
   CONCURRENCY  10                yes       The number of concurrent ports to check per host
   DELAY        0                 yes       The delay between connections, per thread, in milliseconds
   JITTER       0                 yes       The delay jitter factor (maximum value by which to +/- DELAY) in milliseco
nds.
   PORTS        22,25,80,110,21   yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS       192.168.29.89     yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file
:<path>'
   THREADS      3                 yes       The number of concurrent threads (max one per host)
   TIMEOUT      1000              yes       The socket connect timeout in milliseconds
```



```
msf6 auxiliary(scanner/portscan/tcp) > run

[+] 192.168.29.89:        - 192.168.29.89:25 - TCP OPEN
[+] 192.168.29.89:        - 192.168.29.89:21 - TCP OPEN
[+] 192.168.29.89:        - 192.168.29.89:22 - TCP OPEN
[+] 192.168.29.89:        - 192.168.29.89:80 - TCP OPEN
[*] 192.168.29.89:        - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

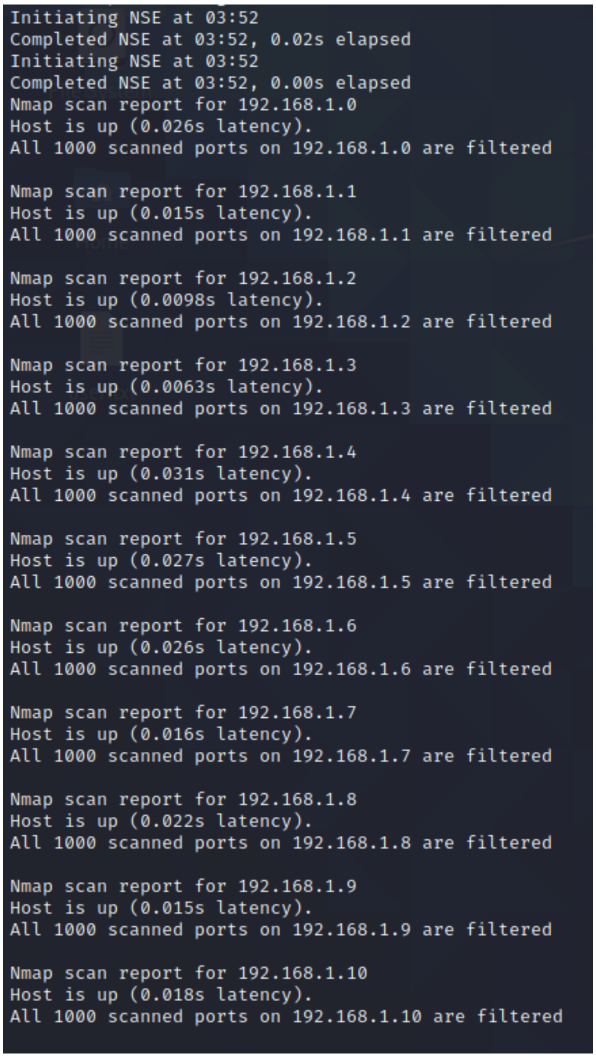


```
msf6 auxiliary(scanner/portscan/tcp) > db_nmap -sV -p 80,22,110,25 192.168.29.89
[*] Nmap: Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-10 03:34 EDT
[*] Nmap: Nmap scan report for 192.168.29.89
[*] Nmap: Host is up (0.0025s latency).
[*] Nmap: PORT    STATE    SERVICE VERSION
[*] Nmap: 22/tcp  open     ssh     OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
[*] Nmap: 25/tcp  open     smtp    Postfix smtpd
[*] Nmap: 80/tcp  open     http    Apache httpd 2.2.8 ((Ubuntu) DAV/2)
[*] Nmap: 110/tcp filtered pop3
[*] Nmap: Service Info: Host:  metasploitable.localdomain; OS: Linux; CPE: cpe:/o:linux:linux_kernel
[*] Nmap: Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 9.27 seconds
```



```
msf6 auxiliary(scanner/portscan/tcp) > nmap -v -sV 192.168.1.0/24 -oA subnet_1
[*] exec: nmap -v -sV 192.168.1.0/24 -oA subnet_1

Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-10 03:36 EDT
NSE: Loaded 45 scripts for scanning.
Initiating Ping Scan at 03:36
Scanning 256 hosts [4 ports/host]
Ping Scan Timing: About 14.79% done; ETC: 03:40 (0:02:59 remaining)
Completed Ping Scan at 03:37, 45.67s elapsed (256 total hosts)
Initiating Parallel DNS resolution of 256 hosts. at 03:37
Completed Parallel DNS resolution of 256 hosts. at 03:37, 8.26s elapsed
Initiating SYN Stealth Scan at 03:37
Scanning 64 hosts [1000 ports/host]
SYN Stealth Scan Timing: About 4.92% done; ETC: 03:48 (0:09:59 remaining)
SYN Stealth Scan Timing: About 7.65% done; ETC: 03:50 (0:12:17 remaining)
SYN Stealth Scan Timing: About 10.24% done; ETC: 03:52 (0:13:18 remaining)
SYN Stealth Scan Timing: About 14.52% done; ETC: 03:52 (0:12:27 remaining)
SYN Stealth Scan Timing: About 20.39% done; ETC: 03:51 (0:11:11 remaining)
SYN Stealth Scan Timing: About 28.00% done; ETC: 03:51 (0:10:20 remaining)
SYN Stealth Scan Timing: About 35.41% done; ETC: 03:52 (0:09:25 remaining)
SYN Stealth Scan Timing: About 42.65% done; ETC: 03:52 (0:08:27 remaining)
SYN Stealth Scan Timing: About 49.11% done; ETC: 03:52 (0:07:41 remaining)
SYN Stealth Scan Timing: About 54.92% done; ETC: 03:52 (0:06:45 remaining)
```

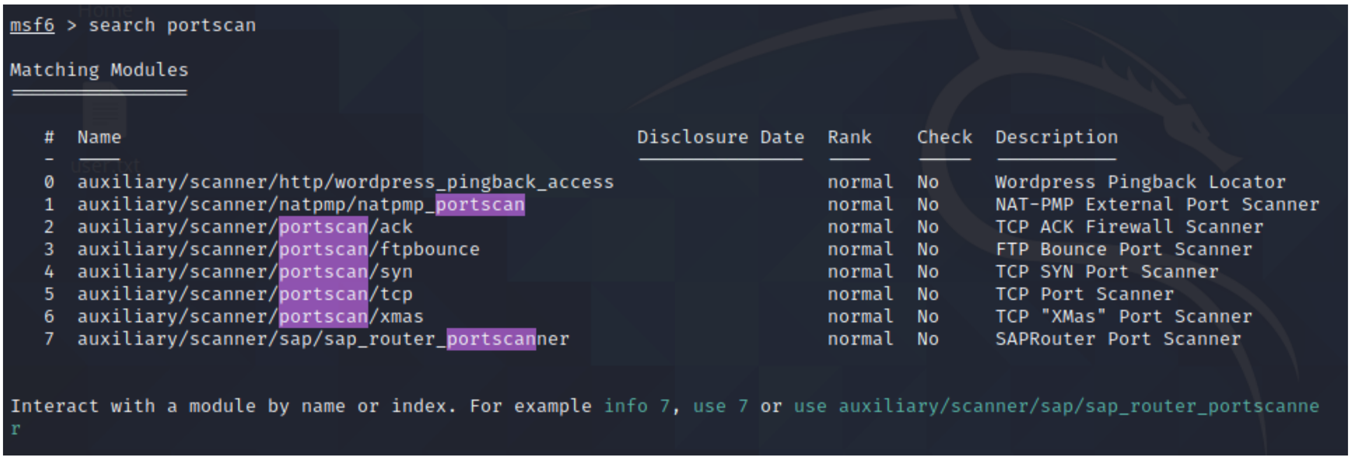
For the sake of comparison, we’ll compare our Nmap scan results for port 80 with a Metasploit scanning module. First, let’s determine what hosts had port 80 open according to Nmap.



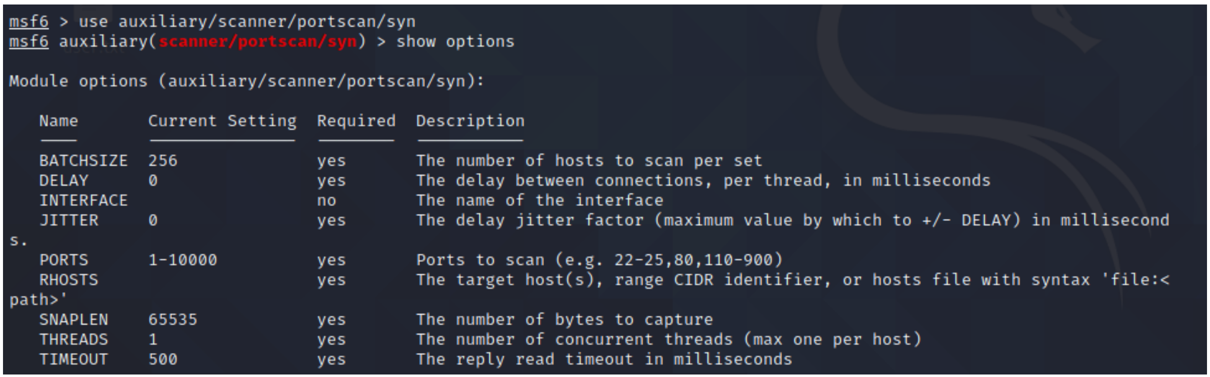
```
Initiating NSE at 03:52
Completed NSE at 03:52, 0.02s elapsed
Initiating NSE at 03:52
Completed NSE at 03:52, 0.00s elapsed
Nmap scan report for 192.168.1.0
Host is up (0.026s latency).
All 1000 scanned ports on 192.168.1.0 are filtered

Nmap scan report for 192.168.1.1
Host is up (0.015s latency).
All 1000 scanned ports on 192.168.1.1 are filtered

Nmap scan report for 192.168.1.2
Host is up (0.0098s latency).
All 1000 scanned ports on 192.168.1.2 are filtered

Nmap scan report for 192.168.1.3
Host is up (0.0063s latency).
All 1000 scanned ports on 192.168.1.3 are filtered

Nmap scan report for 192.168.1.4
Host is up (0.031s latency).
All 1000 scanned ports on 192.168.1.4 are filtered

Nmap scan report for 192.168.1.5
Host is up (0.027s latency).
All 1000 scanned ports on 192.168.1.5 are filtered

Nmap scan report for 192.168.1.6
Host is up (0.026s latency).
All 1000 scanned ports on 192.168.1.6 are filtered

Nmap scan report for 192.168.1.7
Host is up (0.016s latency).
All 1000 scanned ports on 192.168.1.7 are filtered

Nmap scan report for 192.168.1.8
Host is up (0.022s latency).
All 1000 scanned ports on 192.168.1.8 are filtered

Nmap scan report for 192.168.1.9
Host is up (0.015s latency).
All 1000 scanned ports on 192.168.1.9 are filtered

Nmap scan report for 192.168.1.10
Host is up (0.018s latency).
All 1000 scanned ports on 192.168.1.10 are filtered
```

PORT SCANNING

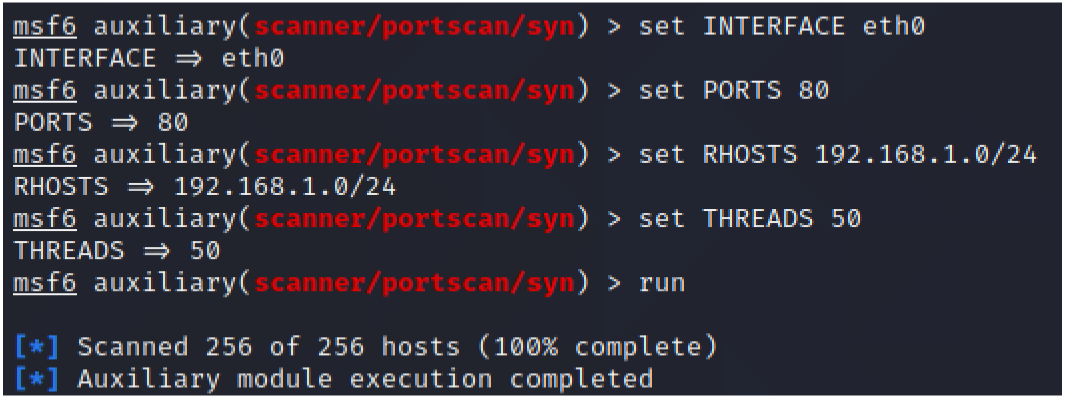
In addition to running Nmap, there are a variety of other port scanners that are available to us within the framework.



```
msf6 > search portscan

Matching Modules
================

   #  Name                                              Disclosure Date  Rank    Check  Description
   -  ----                                              ---------------  ----    -----  -----------
   0  auxiliary/scanner/http/wordpress_pingback_access                   normal  No     Wordpress Pingback Locator
   1  auxiliary/scanner/natpmp/natpmp_portscan                           normal  No     NAT-PMP External Port Scanner
   2  auxiliary/scanner/portscan/ack                                     normal  No     TCP ACK Firewall Scanner
   3  auxiliary/scanner/portscan/ftpbounce                               normal  No     FTP Bounce Port Scanner
   4  auxiliary/scanner/portscan/syn                                     normal  No     TCP SYN Port Scanner
   5  auxiliary/scanner/portscan/tcp                                     normal  No     TCP Port Scanner
   6  auxiliary/scanner/portscan/xmas                                    normal  No     TCP "XMas" Port Scanner
   7  auxiliary/scanner/sap/sap_router_portscanner                       normal  No     SAPRouter Port Scanner


Interact with a module by name or index. For example info 7, use 7 or use auxiliary/scanner/sap/sap_router_portscanne
r
```

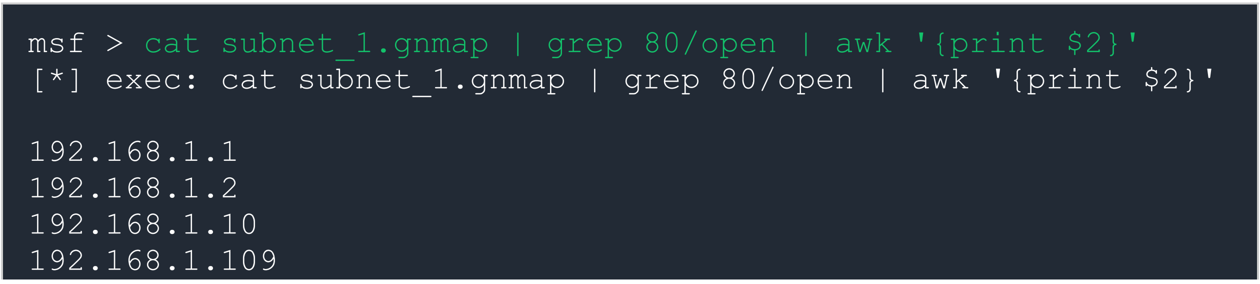


```
msf6 > use auxiliary/scanner/portscan/syn
msf6 auxiliary(scanner/portscan/syn) > show options

Module options (auxiliary/scanner/portscan/syn):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   BATCHSIZE  256              yes       The number of hosts to scan per set
   DELAY      0                yes       The delay between connections, per thread, in milliseconds
   INTERFACE                   no        The name of the interface
   JITTER     0                yes       The delay jitter factor (maximum value by which to +/- DELAY) in millisecond
s.
   PORTS      1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS                      yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<
path>'
   SNAPLEN    65535            yes       The number of bytes to capture
   THREADS    1                yes       The number of concurrent threads (max one per host)
   TIMEOUT    500              yes       The reply read timeout in milliseconds
```



```
msf6 auxiliary(scanner/portscan/syn) > set INTERFACE eth0
INTERFACE => eth0
msf6 auxiliary(scanner/portscan/syn) > set PORTS 80
PORTS => 80
msf6 auxiliary(scanner/portscan/syn) > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf6 auxiliary(scanner/portscan/syn) > set THREADS 50
THREADS => 50
msf6 auxiliary(scanner/portscan/syn) > run

[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
```

For the sake of comparison, we'll compare our Nmap scan results for port 80 with a Metasploit scanning module. First, let's determine what hosts had port 80 open according to Nmap.



```
msf > cat subnet_1.gnmap | grep 80/open | awk '{print $2}'
[*] exec: cat subnet_1.gnmap | grep 80/open | awk '{print $2}'

192.168.1.1
192.168.1.2
192.168.1.10
192.168.1.109
```

```
192.168.1.116
192.168.1.150
```

The Nmap scan we ran earlier was a SYN scan so we'll run the same scan across the subnet looking for port 80 through our eth0 interface, using Metasploit.

```
msf > use auxiliary/scanner/portscan/syn
msf auxiliary(syn) > show options

Module options (auxiliary/scanner/portscan/syn):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   BATCHSIZE  256              yes       The number of hosts
to scan per set
   DELAY      0                yes       The delay between
connections, per thread, in milliseconds
   INTERFACE                   no        The name of the
interface
   JITTER     0                yes       The delay jitter
factor (maximum value by which to +/- DELAY) in milliseconds.
   PORTS      1-10000          yes       Ports to scan (e.g.
22-25,80,110-900)
   RHOSTS                      yes       The target address
range or CIDR identifier
   SNAPLEN    65535            yes       The number of bytes
to capture
   THREADS    1                yes       The number of
concurrent threads
   TIMEOUT    500              yes       The reply read
timeout in milliseconds

msf auxiliary(syn) > set INTERFACE eth0
INTERFACE => eth0
msf auxiliary(syn) > set PORTS 80
PORTS => 80
msf auxiliary(syn) > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf auxiliary(syn) > set THREADS 50
THREADS => 50
msf auxiliary(syn) > run

[*] TCP OPEN 192.168.1.1:80
[*] TCP OPEN 192.168.1.2:80
[*] TCP OPEN 192.168.1.10:80
[*] TCP OPEN 192.168.1.109:80
[*] TCP OPEN 192.168.1.116:80
[*] TCP OPEN 192.168.1.150:80
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
```

Here we’ll load up the ‘tcp’ scanner and we’ll use it against another target. As with all the previously mentioned plugins, this uses the ‘RHOSTS’ option. Remember we can issue the **hosts -R** command to automatically set this option with the hosts found in our database.

```
msf > use auxiliary/scanner/portscan/tcp
msf  auxiliary(tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CONCURRENCY  10               yes       The number of
concurrent ports to check per host
   DELAY        0                yes       The delay between
connections, per thread, in milliseconds
   JITTER       0                yes       The delay jitter
factor (maximum value by which to +/- DELAY) in milliseconds.
   PORTS        1-10000          yes       Ports to scan (e.g.
22-25,80,110-900)
   RHOSTS                        yes       The target address
range or CIDR identifier
   THREADS      1                yes       The number of
concurrent threads
   TIMEOUT      1000             yes       The socket connect
timeout in milliseconds

msf  auxiliary(tcp) > hosts -R

Hosts
=====

address         mac                name  os_name  os_flavor
os_sp  purpose  info  comments
-------         ---                ----  -------  ---------  -
----  -------  ----  --------
172.16.194.172  00:0C:29:D1:62:80        Linux    Ubuntu
server

RHOSTS => 172.16.194.172

msf  auxiliary(tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CONCURRENCY  10               yes       The number of
concurrent ports to check per host
```

```
   FILTER                        no        The filter string
for capturing traffic
   INTERFACE                     no        The name of the
interface
   PCAPFILE                      no        The name of the
PCAP capture file to process
   PORTS        1-1024           yes       Ports to scan (e.g.
22-25,80,110-900)
   RHOSTS       172.16.194.172   yes       The target address
range or CIDR identifier
   SNAPLEN      65535            yes       The number of bytes
to capture
   THREADS      10                yes       The number of
concurrent threads
   TIMEOUT      1000             yes       The socket connect
timeout in milliseconds

msf  auxiliary(tcp) > run

[*] 172.16.194.172:25 - TCP OPEN
[*] 172.16.194.172:23 - TCP OPEN
[*] 172.16.194.172:22 - TCP OPEN
[*] 172.16.194.172:21 - TCP OPEN
[*] 172.16.194.172:53 - TCP OPEN
[*] 172.16.194.172:80 - TCP OPEN
[*] 172.16.194.172:111 - TCP OPEN
[*] 172.16.194.172:139 - TCP OPEN
[*] 172.16.194.172:445 - TCP OPEN
[*] 172.16.194.172:514 - TCP OPEN
[*] 172.16.194.172:513 - TCP OPEN
[*] 172.16.194.172:512 - TCP OPEN
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf  auxiliary(tcp) >
```

We can see that Metasploit’s built-in scanner modules are more than capable of finding systems and open ports for us. It’s just another excellent tool to have in your arsenal if you happen to be running Metasploit on a system without Nmap installed.

SMB VERSION SCANNING

Now that we have determined which hosts are available on the network, we can attempt to determine the operating systems they are running. This will help us narrow down our attacks to target a specific system and will stop us from wasting time on those that aren’t vulnerable to a particular exploit.

Since there are many systems in our scan that have port 445 open, we will use the **`scanner/smb/version`** module to determine which version of Windows is running on a target and which `Samba` version is on a Linux host.

```
msf > use auxiliary/scanner/smb/smb_version
msf auxiliary(smb_version) > set RHOSTS 192.168.1.200-210
RHOSTS => 192.168.1.200-210
msf auxiliary(smb_version) > set THREADS 11
THREADS => 11
msf auxiliary(smb_version) > run

[*] 192.168.1.209:445 is running Windows 2003 R2 Service Pack
2 (language: Unknown) (name:XEN-2K3-FUZZ) (domain:WORKGROUP)
[*] 192.168.1.201:445 is running Windows XP Service Pack 3
(language: English) (name:V-XP-EXPLOIT) (domain:WORKGROUP)
[*] 192.168.1.202:445 is running Windows XP Service Pack 3
(language: English) (name:V-XP-DEBUG) (domain:WORKGROUP)
[*] Scanned 04 of 11 hosts (036% complete)
[*] Scanned 09 of 11 hosts (081% complete)
[*] Scanned 11 of 11 hosts (100% complete)
[*] Auxiliary module execution completed
```

Also notice that if we issue the **hosts** command now, the newly-acquired information is stored in Metasploit's database.

```
msf auxiliary(smb_version) > hosts

Hosts
=====

address        mac  name  os_name            os_flavor  os_sp
purpose  info  comments
-------        ---  ----  -------            ---------  -----
-------  ----  --------
192.168.1.201             Microsoft Windows  XP         SP3
client
192.168.1.202             Microsoft Windows  XP         SP3
client
192.168.1.209             Microsoft Windows  2003 R2    SP2
server
```

IDLE SCANNING

Nmap's IPID Idle scanning allows us to be a little stealthy scanning a target while spoofing the IP address of another host on the network. In order for this type of scan to work, we will need to locate a host that is idle on the network and uses IPID sequences of either Incremental or Broken Little-Endian Incremental. Metasploit contains the module **scanner/ip/ipidseq** to scan and look for a host that fits the requirements.

In the free online Nmap book, you can find out more information on Nmap Idle Scanning.

```
msf > use auxiliary/scanner/ip/ipidseq
msf auxiliary(ipidseq) > show options

Module options (auxiliary/scanner/ip/ipidseq):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   INTERFACE                   no        The name of the
interface
   RHOSTS                      yes       The target address
range or CIDR identifier
   RPORT      80               yes       The target port
   SNAPLEN    65535            yes       The number of bytes
to capture
   THREADS    1                yes       The number of
concurrent threads
   TIMEOUT    500              yes       The reply read
timeout in milliseconds

msf auxiliary(ipidseq) > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf auxiliary(ipidseq) > set THREADS 50
THREADS => 50
msf auxiliary(ipidseq) > run

[*] 192.168.1.1's IPID sequence class: All zeros
[*] 192.168.1.2's IPID sequence class: Incremental!
[*] 192.168.1.10's IPID sequence class: Incremental!
[*] 192.168.1.104's IPID sequence class: Randomized
[*] 192.168.1.109's IPID sequence class: Incremental!
[*] 192.168.1.111's IPID sequence class: Incremental!
[*] 192.168.1.114's IPID sequence class: Incremental!
[*] 192.168.1.116's IPID sequence class: All zeros
[*] 192.168.1.124's IPID sequence class: Incremental!
[*] 192.168.1.123's IPID sequence class: Incremental!
[*] 192.168.1.137's IPID sequence class: All zeros
[*] 192.168.1.150's IPID sequence class: All zeros
[*] 192.168.1.151's IPID sequence class: Incremental!
[*] Auxiliary module execution completed
```

Judging by the results of our scan, we have a number of potential zombies we can use to perform idle scanning. We'll try scanning a host using the zombie at 192.168.1.109 and see if we get the same results we had earlier.

```
msf auxiliary(ipidseq) > nmap -Pn -sI 192.168.1.109
192.168.1.114
[*] exec: nmap -Pn -sI 192.168.1.109 192.168.1.114

Starting Nmap 5.00 ( http://nmap.org ) at 2009-08-14 05:51 MDT
Idle scan using zombie 192.168.1.109 (192.168.1.109:80);
Class: Incremental
Interesting ports on 192.168.1.114:
```

```
Not shown: 996 closed|filtered ports
PORT STATE SERVICE
135/tcp open msrpc
139/tcp open netbios-ssn
445/tcp open microsoft-ds
3389/tcp open ms-term-serv
MAC Address: 00:0C:29:41:F2:E8 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 5.56 seconds
```