An NMAP Primer

Nmap is *the* definitive port scanner. If you have any need for this sort of tool it would behoove you to become familiar with at least the basics of this most excellent application.

This guide is by no means comprehensive — it's nothing more than a short walkthrough of how I have come to use some of the various options. For a thorough look at Nmap's capabilities, I suggest you also consult the man page.

BASICS

This very simple Nmap command below contains nothing but the invocation of Nmap and then the target as an argument — in this case a hostname: ** Note : assume all scans are run as root, which by default performs a TCP SYN scan `(-sS)`

```
nmap localhost
```

We could have specified the target as an IP address as well, or even a network in CIDR notation like so:

```
nmap 192.168.10.0/24
```

This would scan all of the hosts in the 24-bit network and return results like those above for each.

SPECIFYING PORTS

By default, nmap scans 1663 ports (in version 3.81), but it's possible and often prudent to change how many and/or which ports are scanned. You can do this by specifying the `-p` option like so:

```
nmap -p1-10000 192.168.10.0/24
```

This would scan TCP ports 1-10,000 on the entire 24-bit network. In order to pick and choose between certain ports *and* ranges, you can do the following:

```
nmap -p22,23,10000-15000 192.168.10.0/24
```

You can even do this when adding UDP ports by specifying first that you're doing a UDP scan `(-sU)`and then defining which ports on each protocol to scan:

```
nmap -sU -pT:21,22,23,U:53,137
192.168.10.0/24
```

When running Nmap as root, the default scan type is TCP SYN `(-sS)`. This type of scan sends, as one might expect, TCP packets with only the synchronize bit set in the TCP options. Again, if you're running as root, you don't need to specify `-sS` to perform this type of scan.

UDP SCANS

Another type of scan that Nmap can do is the UDP scan. As mentioned, the `-sU` option must be given to Nmap in order for it to scan using the UDP protocol.

```
nmap -sU host
```

SPECIAL TCP SCANS

Nmap is also able to do specialized TCP scans such as the FIN scan, the XMAS scan, the ACK scan, and the NULL scan. The FIN scan sends a TCP packet with only the FIN (finish) bit set. The ACK scan sends a TCP packet with only the ACK flag set. The XMAS scan sends a TCP packet with the FIN, URG, and PSH flags. The NULL scan, as can be expected, sends a packet with no TCP options set at all. These are done like like so, respectively:

```
nmap -sF host
```

```
nmap -sX host
```

```
nmap -sA host
```

```
nmap -sN host
```

As an interesting aside, the XMAS scan is named for the mental picture of the TCP flags being, “lit up like a Christmas tree.” These options are sometimes useful when there is network and/or host filtering taking place (and you wish to bypass it and learn about the hosts anyway), but the TCP SYN and UDP scan types make up the majority of my scans.

THE “PING” SCAN

You can also use Nmap to merely check and see what hosts are out there, i.e. *without doing a portscan afterwards.* This is done with the `-sP` option:

```
nmap -sP network
```

It’s interesting to note that this sends both an ICMP echo *and* a TCP ACK to the hosts in the target range. Again, the key thing to remember is that this option is used only when you **don’t** also want to do a portscan. In other words, unless explicitly stated, regular port scans also “ping” the hosts as well; the difference is that they then go on to scan it.

VERSION SCANNING

An interesting, relatively new feature in Nmap is its ability to attain version information for various TCP and UDP services on target machines. This is done in conjunction with whatever other scans you choose to run:

```
nmap -p1-10000 -sV host
```

This scan would scan the first 10,000 ports on the host and give version information where possible about the services found on the system.

A TWO STEP PROCESS — DISCOVERY AND SCANNING

One thing that helped me understand Nmap infinitely better was to embrace the concept of Nmap kicking off in two distinct phases — *the **discovery** phase, and the **scan** phase.*

The key concept here is that if discovery fails for a particular host, Nmap **doesn't scan it**. This means you have to ensure that the options you give to Nmap will find hosts in the discovery phase.

One option that helps you do this is the `-P0` (pee-zero, not pee-oh) option. This basically says to Nmap, "Don't worry about discovery — just scan." So if you find yourself scanning a massive list of hosts that you *know* are likely to have services running, `-P0` is one option.

```
nmap -P0 network
```

Again, by default ICMP and a TCP ACK on port 80 are used for the discovery portion, but options are available. My current favorite is `-PS`, which does discovery using TCP SYNs on ports that you specify:

```
nmap -PS21,22,23,80 network
```

Here I've not only allowed the ping option to remain, but I've added some additional discovery methods. This gives me much more of a chance of finding hosts — hosts that will then in turn be scanned by Nmap vs. being passed over due to failing discovery.

You may remember that you can scan every host in your range by specifying `-P0`, so you may be asking why one would even use discovery options like `-PS` in the first place.

Well, the reason is speed. If you're dealing with 16-bit networks and/or saturated network links it's far better to make a concerted effort toward discovery than to just scan everything. This, I imagine, is why Fyodor made it such a pivotal part of Nmap.

OPERATING SYSTEM DISCOVERY

Another interesting Nmap option is the ability to attempt to determine what operating system a target host is running. It does this by analyzing a number of subtelties exhibited by a given host and comparing them to a database of known behavior. This is accomplished with the `-O` (oh, not zero) option:

```
nmap -O network
```

In addition to getting a fairly solid operating system guess, this option also provides uptime and TCP sequence prediction difficulty information.

OUTPUT

One of the most neglected yet powerful options in Nmap is its ability to output in various formats. Few realize that you can create reports in a myriad of formats — human readable, grepable, XML, etc.

My personal recommendation is to use the `-oA` output option for every Nmap scan. This creates output in *all three formats*. The XML format is particularly interesting in that you can put it on a webserver and point the document to nmap.xsl, which will format it nicely.

```
nmap -oA output_file network
```

This will create three files in the current directory — output_file.nmap (human readable), output_file.gnmap (grepable), and output_file.xml (XML). You can also produce these seperately via `-oN`, `-oG`, and `-oX`respectively.

One *very cool* thing you can do with the grepable output (outputfile.gnmap) is you can programatically extract hostnames from it — something thats very helpful when you need a list of hosts to feed another application. This can be done like so from a Bash prompt:

```
cat output.gnmap | cut -d” ” -f2 | grep
^[0-9] > newapp_input
```

This essentially says, “read the output file, take out the second field, make sure it starts with a digit, and write the results to a file named newapp_input”. Assuming you did your due dilligence during the discovery phase of the Nmap scan, this will produce a quality list of hosts for you so that your additional, more thourough tools won’t have to waste time performing another discovery.

RESUME AN INTERRUPTED SCAN

I commonly have to scan 16-bit networks and I can tell you that the ability to recover after a scan crashing is a very useful feature.

This option requires that you have a logfile in either the human readable or grepable format, so it’s yet another reason to go ahead and use the `-oA` option when performing all scans. Essentially though, all you do is call one of the file names and it picks up where it left off.

```
nmap –resume logfile_name
```

VERBOSITY

Adding verbosity to your output is often desired, and this can be accomplished via the single or double `-v` options.

```
nmap -vv -oA output target
```

SCAN SPEED

If you do these scans in production environments (especially those that you don’t own), it’s important to take into consideration the availablility of the network.

Nmap offers options to throttle the speed of your scans by running its probes serially rather than in parallel and by varying the time between each probe. The option is `-T` and it has several parameters — Paranoid, Sneaky, Polite, Normal, Aggressive, and Insane.

The first three are different than the default scan type (normal) in that they send their probes *serially* instead of in parallel. The difference between them is in how long they delay between each packet they send. Paranoid waits 5 minutes, Sneaky waits 15 seconds, and Polite waits at least .4 seconds.

At the Normal stage the parallel scan types start, and it tries to go as fast as possible while monitoring for bandwidth starvation. Agressive and Insane get progressivly faster and less concerned for the well-being of the network being scanned. Don’t try these on any network where availability is an issue.

FUN

One little known option that Fyodor threw in there is the ability to output to Leetspeak. The option for this is `-oS`.

```
nmap -oS output_file target
```

WRAPPING IT UP

Well, that’s it for now. Again, I know this isn’t anywhere near comprehensive, but it’s a decent start and tends to get me through most sessions. If you spot any errors or just have a comment, do drop me an email at daniel@danielmiessler.com.

REFERENCES

The Official Nmap Manpage
http://www.insecure.org/nmap/data/nmap_manpage.html