

NMAP Guide

Lamont Granquist

NMAP has been getting a lot of review on what its capabilities are lately, so I thought I'd take a shot at it as well. I skipped over a few things that I didn't think were really worth mentioning you better be able to figure out (-p and -F). Comments more than welcome.

NMAP does three things. First, it will ping a number of hosts to determine if they are alive or not. Second, it will portscan hosts to determine what services are listening. Third, it will attempt to determine the OS of hosts.

Of course NMAP is very configurable, and any of these steps may be omitted, (although portscanning is necessary in order to do an OS scan), and there are multiple ways to accomplish most of these, and many command line switches to tweak the way that NMAP operates.

Target Selection

You can specify NMAP targets both on the command line or give a list of targets in a filename with the -i option. As the NMAP help documentation suggests you can use the hostname/mask method of specifying a range of hosts (cert.org/24 or 192.88.209.5/24) or you can give an explicit IP range (192.88.209.0-255). The '24' in 'cert.org/24' is the number of bits in the mask, so /32 means "just that host", /24 means "the 256 addresses in that Class C", /16 means "the 65536 addresses in that Class B", /8 would be "the 2²⁴ addresses in that Class A" and /0 would scan all possible (IPv4) 2³² IP addresses.

Ping Scans

The default behavior of NMAP is to do both an ICMP ping sweep (the usual kind of ping) and a TCP port 80 ACK ping sweep. If an admin is logging these this will be fairly characteristic of NMAP. This behavior can be changed in several ways. The easiest way is, of course, to simply turn off ping sweeps with -P0.

If you want to do a standard ICMP ping sweep use -PI. If you are trying to get through a firewall, though, ICMP pings will likely be blocked and using packet filtering ICMP pings can even be dropped at the host.

To get around this NMAP tries to do a TCP "ping" to see if a host is up. By default it sends an ACK to port 80 and expects to see a RST from that port if the host is up. To do only this scan and not the ICMP ping scan use -PT. To specify a different port than port 80 to scan for specify it immediately afterwards, e.g. -PT32523 will ACK ping port 32523.

Picking a random high-numbered port in this way may work *much* better than the default NMAP behavior of ACK pinging port 80. This is because many packet filter rules are setup to let through all packets to high numbered ports with the ACK bit set, but sites may filter port 80 on every machine other than their publically accessible web servers. You can also do both an ICMP ping scan and an ACK scan to a high numbered port with, e.g. -PB32523.

However, if a site has a really, really intelligent firewall that recognizes that your ACK packet isn't part of an ongoing TCP connection it might be smart enough to block it. For that reason, you may get better results with a TCP SYN sweep with -PS. In this case, scanning a high-numbered port will probably not work, and instead you need to pick a port which is likely to get through a firewall. Port 80 is not a bad pick, but something like ssh (port 22) may be better.

So the first question to ask yourself is if you care about wasting time scanning machines which are not up and if you care about getting really complete coverage of the network? If you don't care about wasting time and really want to hit all the machines on a network, then use -P0.

NMAP Guide

Lamont Granquist

Pinging machines will only cause you to have more of a signature in any log files and will eliminate machines which might possibly be up. Of course, you will waste time scanning all the IP numbers which aren't assigned.

If you do ping machines, an ICMP ping sweep is probably more likely to be missed or ignored by system administrators. It doesn't look all that hostile. If you think you're up against a firewall you should experiment with which kinds of pings seem to get through it. Do ICMP pings work at all? Can you ping thier webserver? If not, then don't bother with ICMP pings. Can you ACK ping thier webserver? If not, then you have to go with SYN pings.

What if all you want to do is a ping scan? Then use -sP.

Port Scanning

The vanilla scan is a TCP connect() scan (-sT). These are loggable. You probably don't want to do these. SYN scans (-sS) are the workhorse of scanning methods. They are also called "half-open" scans because you simply send a SYN packet, look for the return SYN|ACK (open) or RST (closed) packet and then you tear down the connction before sending the ACK that would normally finish the TCP 3-way handshake. These scans don't depend on the characteristics of the target TCP stack and will work anytime a connect() scan would have worked.

They are also harder to detect -- TCP-wrappers or anything outside of the kernel shouldn't be able to pick up these scans -- packet filters like ipfwadm or a firewall can though. If a box is being filtered NMAP's SYN scan will detect this and report ports which are being filtered.

FIN (-sF), NULL (-sN) and XMAS (-sX) scans are all similar. They all rely on RFC-compliance and as such don't work against boxes like Win95/98/ NT or IRIX. They also work by getting either a RST back (closed port) or a dropped packet (open port). Of course, the other situation where you might get back a dropped packet is if you've got a packet filter blocking access to that port. In that case you will get back a ton of false open ports. A few years back these kinds of scans might have been stealthy and undetectable. These days they probably aren't.

You can combine any of the SYN, FIN, NULL or XMAS scans with the (-f) flag to get a small fragment scan. This splits the packet which is sent into two tiny frags which can sometimes get through firewalls and avoid detection. Unfortunately, if you're not running a recent version of an open source O/S (Linux or Net/Open/FreeBSD) then you probably can't frag scan due to the implimentation of SOCK_RAW on most unices (Solaris, SunOS, IRIX, etc). See Fyodor's NMAP portability chart to see if -f is supported on your platform.

For the initiated out there, you could modify libpcap to allow you to send packets in addition to sniffing them by opening the packet capture device rw instead of ro. Then you need to build a link-layer (probably ethernet) header and then you could impliment your own frag scanner. For bonus points impliment all of the different SYN, FIN, NULL and XMAS scans *and* allow for sending the fragments out in reverse order (which helps for getting through firewalls). This hasn't been done (yet) in NMAP due to the fact that NMAP needs to support multiple different link layer interfaces (not just ethernet) and needs code for dealing with ARP.

If anyone wants to code this up, I'm sure that people would appreciate it. UDP scanning (-sU) in NMAP has the same problem as FIN scans in that packet filtered ports will turn up as being open ports. It also runs extremely slowly against machines with UDP packet filters.

NMAP Guide

Lamont Granquist

Another type of scan is the bounce scan (-b <ftp_relay_host>) which, if there is insufficient logging on the ftp host you're using to bounce, is completely untraceable. Recent FTP servers shouldn't let you do these kinds of scans.

The last scanning option that I'm going to mention is identd scanning (-I) which only works with TCP connect scans (-sT). This will let you know the owner of the daemon which is listening on the port. Provided, of course, that the site is running identd and is not doing something intelligent like using a cryptographic hash (i.e. pidentd -C). You *have* to make complete 3-way TCP handshakes for this to work, so this is not very stealthy. It does, however, give you a lot of information. It only works against machines that have port 113/auth open.

Source IP Deception

You can also take advantage of the fact that you can change your source address. The simplest way to do this is with -S <ip>. If you are on a broadcast ethernet segment you could change your source address to an IP which doesn't exist and then you simply sniff the network for the reply packets. And if you are not on a leaf node/network then as long as the reply packet will get routed by you, you can use it. To turn this on its head: the next time you get scanned, do a traceroute on the machine that scanned you. Any of the machines on any of the networks that those packets went through could have been the machine which was *really* scanning you.

The other deceptive measure is to use decoy scans. You spoof a ton of scans originating from decoy machines and insert your IP in the middle of it somewhere. The admin at the site you are scanning is presented with X number of scans and no way to determine which one actually did it.

For bonus points, combine this with the previous tactic and spoof an IP address which doesn't exist. If you don't spoof your own IP address make sure to use "likely" decoys -- use machines which were connected to the net at the time you made your scans and don't use sites like www.microsoft.com. Ideally you want a lot of linux boxes as decoys.

The more decoys the better, but obviously the slower the scan will go.

[QUESTION: do decoy/spoof scans also decoy/spoof the ping scan? Can you combine decoy scans and "ME" spoofing like this? does a decoy/ spoof scan also decoy/spoof the OSscan?]

OS scanning

This is the -O option. To use it requires one open and one closed port. The closed port is picked at random from a high-numbered port. Machines which do packet filtering on high-numbered ports will cause problems with OS detection (many sites will filter packets to high numbered ports which don't have the ACK bit set). Also excessive packet loss will cause problems with OS detection. If you run into trouble try selecting an open port which isn't being served by inetd (e.g. ssh/22 or portmap/rpcbind/111).

OS scanning also reports the TCP sequence number prediction vulnerability of the system. If you're 31337 you will be able to use this to exploit trust relationships between this machine and other machines. There's a reasonably decent phrack article on this in phrack P48-14, but you should beware that it isn't this easy -- you need to worry about ARP (what's that? how does it work? i suggest familiarizing yourself with tcpdump) and if you're trying to exploit rsh/rlogin you need to worry about spoofing the authorization connection as well.