UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

SEMINAR

Nmap: Scanning the Internet

Marin Koštić

Mentor: *Doc. dr. sc. Predrag Pale*

Zagreb, January 2018.

Contents

1. Introduction

Nmap ("Network Mapper") is a free and open source utility for network discovery and security auditing. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host and service uptime. In this seminar I will talk about how Gordon Lyon (also known by his pseudonym Fyodor Vaskovich) has taken Nmap to a whole new level by scanning millions of Internet hosts as part of the Worldscan project. Fyodor has presented his work on DEFCON 16 conference where he talked about how he used Nmap to determine possible bugs and flaws so he could further improve upon it and also he demonstrated practical advices on how to efficiently use Nmap. In the following pages I will show examples on how to use Nmap for host discovery and port scanning as the main topic, but I will also include additional topics about detecting and subverting firewall and intrusion detection systems.

2. Scanning the Internet

2.1. Scan Challenges

Before making an scan of such scale, first question that comes to mind is what are the actual IP addresses that you want to scan. There are couple of options that one can use to determine IP addresses to scan, one could take BGP routing tables. BGP stands for Border Gateway Protocol, it is a standardized exterior gateway protocol designed to exchange routing and reachability information among autonomous systems on the Internet. Another interesting option is using DNS zone files, these are text files that describe a DNS zone. A DNS zone is a subset, often a single domain, of a hierarchical domain name structure of the DNS. The zone file contains mapping between domain names and IP addresses and other resources. But what in his talk Fyodor used Nmap's own random IP generator and demonstrated how one can use this Nmap tool to get a list of IP addresses without actually scanning the machines.

Listing 2.1: Using Nmap to generate random IP addresses

```
nmap -iR 25200000 -sL -n | grep "not␣scanned"
| awk '{print $2}' | sort -n |
uniq >! tp; head -25000000 tp >! 25M-IPs;
rm tp
```

Next question, when you obtain IP addresses to scan, is what source do you wanna use for the actual scan. It is important to understand when you get a list of 25 million IP addresses and you want to preform such a scan as Fyodor did it rises a legal concern. If you preform this scan from your home using your ISP network you can expect to be denied access to the Internet and legal charges could be pressed upon you. So when Fyodor talked about this concern he made a joke about how he thought to use his neighbours open wireless access point. Although this is just a joke, if you look at it from your neighbours point of view having an open wireless access point is a danger and could be exploited using Nmap scans from anyone with little knowledge on how to perform scans. What he actually did when performing such big scan is that he contacted his ISP and explained what he was doing and why, this is example of best practices that any network administrator should have before executing network scans. The final remark that he makes about large scale scans is that when performing such scans one must be aware of the fact that scanning the Internet is long and hard work and it can be disheartening. He listed an example of an UDP scan with 65 thousand ports and 2048 hosts in a group from his original IP address list.

Listing 2.2: UDP Scan performance example

```
- Stats: 93:57:40 elapsed; 254868 hosts
 completed (2048 up), 2048 undergoing UDP Scan
UDP Scan Timing: About 11.34\%done; ETC:
 03:21 (-688:-41:-48 remaining)
```

As it can be seen from the example this was a scan of such scale that time estimation resulted in an integer overflow. When looking into such examples one can think are such scans even worth doing.

2.2. Host Discovery

In the previous section we described how to use Nmap to random generate a list of IP addresses now we have to tackle a task of host discovery. This is one of the first steps in any network reconnaissance, what it means is to reduce a set of known IP addresses into a list of active or somehow interesting hosts. There are many methods to use within Nmap for host discovery, most common and by default active when using Nmap is Echo request and TCP ACK to port 80. Although these might be sufficient in some cases, when doing a large scale scan as one Fyodor described this would not be a good solution. Maybe this type of host discovery would be efficient in the late 90's when most of hosts did not have strict iptables rules and weren't behind a firewall. But today Internet is quite a dangerous place to have open ports and many big companies have a good practice to block scans that use Echo request and TCP ACK on port 80. But there are other different methods that are available in Nmap for host discovery and can be used in such cases.

TCP Host Discovery methods (-PS, -PA):

- SYN packet discovery (-PS) best against stateful firewall. The SYN flag suggests to the remote system that you are attempting to establish a connection. Normally the destination port will be closed, and a RST (reset) packet sent back. If the port happens to be open, the target will take the second step of a TCP three-way-handshake by responding with a SYN/ACK TCP packet.
- ACK packet discovery (-PA) best against stateless firewall. The TCP ACK ping is quite similar to the just-discussed SYN ping. The difference, as you could likely

guess, is that the TCP ACK flag is set instead of the SYN flag. Such an ACK packet purports to be acknowledging data over an established TCP connection, but no such connection exists. So remote hosts should always respond with a RST packet, disclosing their existence in the process.

As described above we see that different TCP port scanning methods have different scanning results so a best practice would be to use both methods. Now what one could be wondering is what are the actual ports that are good to scan with these methods. It would be futile to scan all ports since we know that there are 65 thousand available ports, and such a scan would just be too long to perform on a big number of hosts. What Fyodor suggests in his talk is the fallowing list of ports, what is important to note is that this information was a result of scans that he did on hosts that were heavily firewalled.
Top TCP Host Discovery Ports:

- 80/http
- 25/smtp
- 22/ssh
- 443/https
- 21/ftp
- 112/auth
- 23/telnet
- 53/domain
- 554/rtsp
- 3389/ms-term-server

Next strategy is to use the UDP discovery method (-PU) for port scanning. When using UDP port scanning the goal is to find closed ports since they are more likely to respond, because an open UDP port won't even respond to an UDP probe. But an close port will send an Port Unreachable packet which is more than enough information to find out whether a host is up or not. A best practice is to use a high number port when using this scan and to include port 53, this port is interesting and worthwhile due to firewall exceptions for DNS.

2.3. Port Scanning

The core functionality when using Nmap is port scanning, a simple command **nmap <target>** by default scans 1000 TCP ports on the host <*target*>. While many port scanners have traditionally lumped all ports into the open or closed states, Nmap is much more granular. It divides ports into six states: *open*, *closed*, *filtered*, *unfiltered*, *open | filtered* or *closed | filtered*.

open An application is actively accepting TCP connections, UDP datagrams or SCTP associations on this port. Finding these is often the primary goal of port scanning. Security-minded people know that each open port is an avenue for attack. Attackers and pentesters want to exploit the open ports, while administrators try to close or protect them with firewalls without thwarting legitimate users.

closed A closed port is accessible (it receives and responds to Nmap probe packets), but there is no application listening on it. They can be helpful in showing that a host is up on an IP address (as I have described in the section before), and as part of OS detection (I will talk about this in the next section).

filtered Nmap cannot determine whether the port is open because packet filtering prevents its probes from reaching the port. The filtering could be from a dedicated firewall device, router rules, or host-based firewall software.

unfiltered The unfiltered state means that a port is accessible, but Nmap is unable to determine whether it is open or closed.

open | filtered Nmap places ports in this state when it is unable to determine whether a port is open or filtered. This occurs for scan types in which open ports give no response.

clised | filtered This state is used when Nmap is unable to determine whether a port is closed or filtered. It is only used for the IP ID idle scan.

Now that that I have described how will Nmap display results when performing port scans, lets have a look at some of the port scanning techniques that Nmap offers. It is important to note that most of the scan types are only available to privileged users. This is because they sent and receive raw packets, which requires root access on Unix systems. Also keep in mind that while Nmap attempts to produce accurate results, these results are based on packets returned by the target machines (or firewall in front of them). Such hosts may be untrustworthy and send responses intended to confuse or mislead Nmap. Now lets look at the list of scanning techniques that Nmap offers.

-sS (TCP SYN scan) SYN scan is the default and most popular scan. It can be performed quickly, scanning thousands of ports per second on fast network not hampered by restrictive firewall. This scan is also relatively unobtrusive and stealthy since it never completes TCP connections. It also allows clear, reliable differentiation between the *open*, *closed* and *filtered* states. How this scan works is quite simple, you send a SYN packet, as if you are going to open a real connection and wait for a response. A SYN/ACK or SYN indicates the port is listening (open), while a RST (reset) is indicative of a non-listener. If no response is received after several retransmissions the port is marked as filtered. Also port is marked filtered if an ICMP unreachable error is received.

-sT (TCP connect scan) this type of a scan is the default TCP scan when SYN scan is not an option. This occurs when a user does not have raw packet privileges. This scan instead of sending raw packets uses the underlying operating system to establish a connection with the target machine and port by issuing the *connect* system call. This is the same high-level system call that web browsers, P2P clients, and most other network enabled applications use to establish a connection. Note SYN scan if available it is always a better option to use. This is because the connect scan completes a three way TCP handshake, and not only that this process takes longer time to complete and requires more packets to obtain the same information, such a scan is more likely to be logged by the target machine. So any decent administrator who sees a bunch of connection attempts in his machine logs from a single system should know that his machine has been connect scanned.

-sA (TCP ACK scan) This scan is different than others discussed so far that it never determines *open* (or event *open* | *filtered*) ports. It is used to map out firewall rulesets, determining whether they are stateful or not and which ports are filtered.

-sU (UDP scans) While most services on the Internet run over TCP protocol, UDP services are widely deployed. Most common three services examples that use UDP are DNS, SNMP, DHCP. Because UDP scanning is generally slower and more difficult than TCP, some security auditors ignore these ports. This is considered as bad practice, as exploitable UDP service are quite common and attackers certainly don't ignore the whole protocol. This scan can be used in combination with TCP scan such as SYN scan (-sS) to check both protocols during the same run (this is very useful). UDP scan works by sending a UDP packet to every targeted port (some common ports are 53 and 161). If an ICMP port unreachable error (type 3, code 3) is returned, the port is *closed*. Other ICMP unreachable errors (type 3, codes 0,1,2,9,10, or 13) mark the port as *filtered*. If a service on the port responds with a UDP packet, port is marked as *open*.

Otherwise if no response is received after retransmissions, the port is classified as *open | filtered*. The biggest challenge when doing UDP scanning is doing it quickly. Open and filtered ports rarely send any response, leaving Nmap to time out and then conduct retransmissions just in case probe or response were lost. Close ports are even bigger problem. They usually send back an ICMP port unreachable error. But unlike the RST packets sent by TCP ports in response to SYN or connect scan, many hosts rate limit ICMP port unreachable messages by default (Linux and Solaris are particularly strict about this). A best practice advice to speed up your UDP scans is to include scanning more hosts in parallel, or doing a quick scan just for the popular ports.

-sN; -sF; -sX (TCP NULL, FIN , and Xmas scan) Null scan (-sN) does not set any bits (TCP flag header is 0), FIN scan (-sF) sets just the TCP FIN bit, and Xmas scan (-sX) sets FIN, PSH, and URG flags. The key advantage to these scans types is that they can sneak through certain non-stateful firewalls and packet filtering routers. Another advantage is that these scans types are a little more stealthy then even a SYN scan. But do not think a these scans can not be detected, modern IDS products can be configured to detect them. There are some disadvantages when doing these scans, not all system follow RFC 793 to the letter. Major operating systems that to this are Microsoft Windows, many Cisco devices, BSDI, and IBM OS/400, but this scan does work agains most Unix-based systems.

2.4. OS Detection

Remote OS (Operating system) detection using TCP/IP stack fingerprinting is one of best-known features that Nmap offers to its users. To determine OS Nmap sends a series of TCP and UDP packets to the remote host and examines practically every bit in the responses. After performing dozens of tests such as TCP ISN sampling, TCP options support and ordering, IP ID sampling, and initial window size check, Nmap compares the results to its *nmap-os-db* database of more than 2600 known OS fingerprints and prints out the OS details if there is a match. OS detection enables some other tests which make use of information that is gathered during the process anyway. One of these is TCP Predictability Classification. This measures approximately how hard it is to establish a forged TCP connection agains the remote host. This is useful for exploiting source-IP based trust relationship (rlogin, firewall, filters, etc.) or for hiding the source of the attack. OS detection is enabled and controlled with the following options:

-O (Enable OS detection) Alternatively, you can use -A to enable OS detection along with other things.

2.5. Firewall/IDS Evasion and Spoofing

Network obstructions such as firewalls can make mapping a network exceedingly difficult. In addition to restricting network activity, companies are increasingly monitoring traffic with intrusion detection systems (IDS). All major IDSs ship with rules designed to detect Nmap scans because scans are sometimes precursor to attacks. Many people suggest that Nmap shound not offer features for evading firewall rules or sneaking past IDSs. They argue that these features are just as likely to be misused by attackers as used by administrators to enhance security. The problem with this logic is that these methods would still be used by attackers, who would just find other tools or even patch Nmap with such functionalities. Meanwhile, administrators would be find it much harder to do their jobs. There is no magic Nmap option for detecting and subverting firewalls and IDS systems. It takes skill and experience to to these kind of scans. To show such examples I do not have enough skill or knowledge. There are many books, lectures and blogs published on several different platforms about this topic, one that is notable and should be mention is BlackHat. But what I can give you is an list of options you can use within Nmap to accomplish such tasks.

-f (fragment packets); --mtu (using the specified MTU) Option -f causes the requested scan (including ping scans) to use tiny fragmented IP packets. The idea is to split up the TCP header over several packets to make it harder for packets filters, IDS, and other annoyances to detect what your are doing. When using this option you must be careful, some programs have trouble handling these packets. In addition to fragmentation you can use option to define your own offset size with --mtu (Do not specify -f if you use --mtu).

-D (Cloak a scan with decoys) This type of scan makes it appear to the remote host that the host you specify as decoys are scanning the target network too. Thus their IDS might report 5-10 port scans unique IP addresses, but they won't know which IP was scanning them and which were decoys. This method can be used for effective way of hiding your IP address.

-S <IP_Address> (Spoof source address) In some cases Nmap may not be able to determine your source address, so you can use this option to specify an interface you wish to send packets through. In other cases it is possible to spoof the source IP address and make the host think that someone else is scanning them. Note that you usually won't receive reply packets back and Nmap will not produce useful reports.

--source-port <portnumber> (Spoof source port number) One surprisingly common misconfiguration is to trust traffic based only on the source port number (example DNS port 53 and FTP port 20). Nmap offers the -g and --source-port options (they are equivalent) to exploit these weaknesses. Simply provide port number and Nmap will send packets from that port where possible.

--spoof-mac <MAC address, prefix, or vendor name> This option only affects raw packets scans such as SYN scan or OS detection (remember for raw scans you must have privileges).

--data <hex string> (Append custom binary data to payload) This option lets you include binary data as payload in sent packets. Just make sure you specify the information in the byte order expected by the receiver.

3. Conclusion

4. Abstract