



OSTBAYERISCHE  
TECHNISCHE HOCHSCHULE  
REGENSBURG

---

## **Fakultät Informatik und Mathematik**



**Prof. Dr. Waas**

### **Praktikum zum Fach Kommunikationssysteme/ Rechnernetze**

**Übung**

# **NMAP**

**Linux**

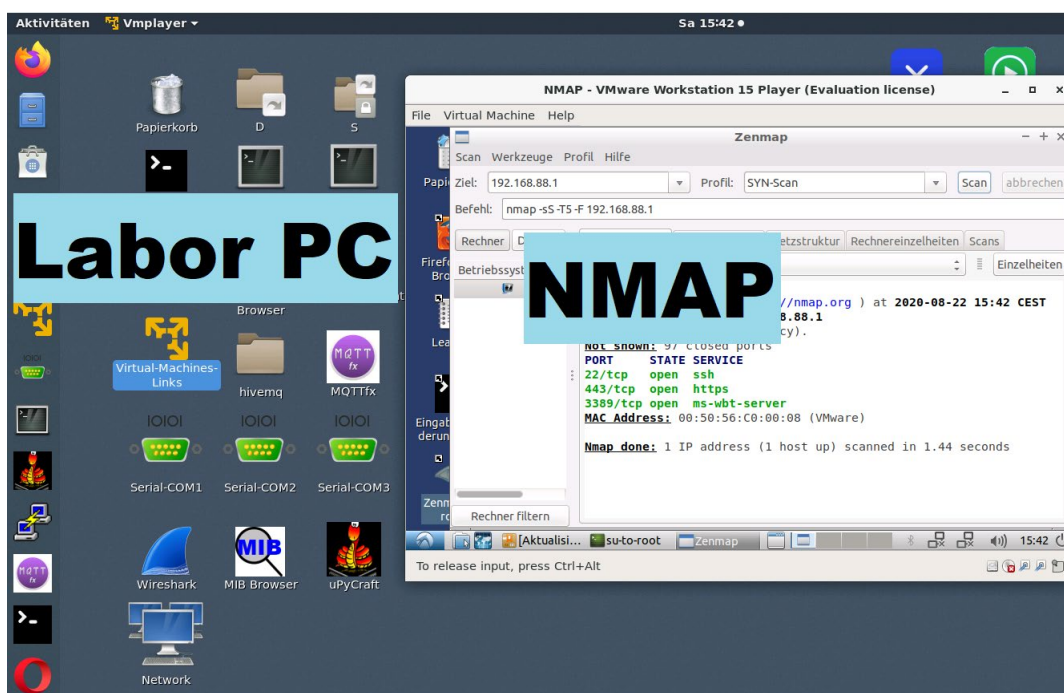
**(Version 27.02.2022 KVM)**

# 1 EINFÜHRUNG

Die folgenden Übungen der Netzwerkanalyse werden in einer virtuellen Umgebung auf dem Labor-PC durchgeführt. Dazu werden sog. virtuellen Maschinen verwendet, die auf Basis von KVM betrieben werden.

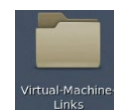
Die im Rahmen der Übung an Ihrem Arbeitsplatz erzielt Messergebnisse, dürfen Sie auf Ihrem Memorystick zur späteren Nachbearbeitung abspeichern (*entfällt bei remote Zugang*) und sind am Ende des Semesters zu löschen.

**Es ist strengstens verboten, irgendwelche anderen Hosts, als die in der Übung vorgesehenen, zu attackieren. Versuche diesbezüglich können zu erheblichen Störungen im Datennetz der OTH Regensburg führen und werden überwacht und strengstens geahndet.**



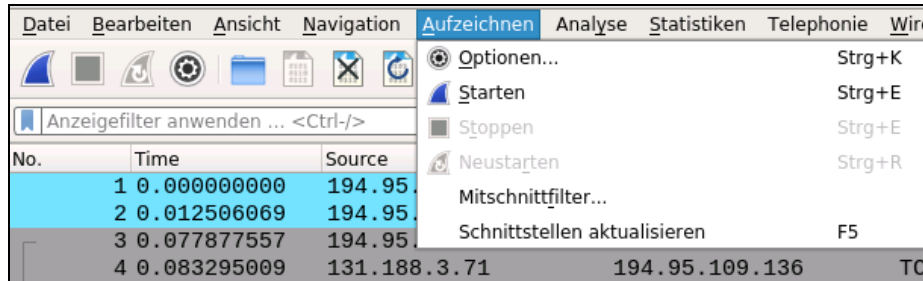
## 2 STARTEN DER ÜBUNGSUMGEBUNG

- Öffnen Sie den Ordner Virtual Machines Links auf dem Desktop.
- Starten Sie die folgende virtuelle Maschine durch Doppelklick:  
**NMAP**

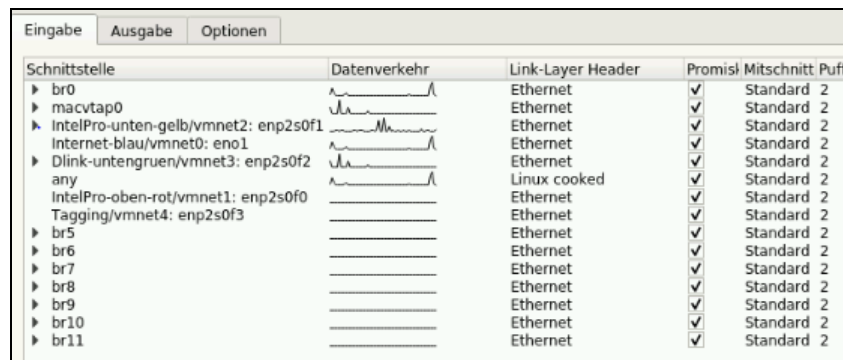


### 3 IPV4 SCANS

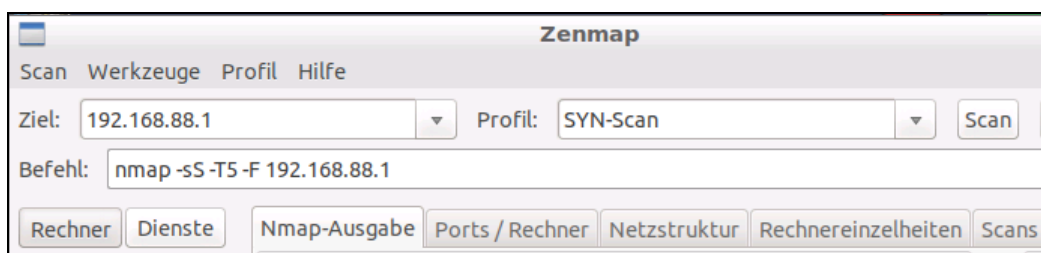
- Starten Sie auf dem Labor-PC das Programm **WireShark**.
- Öffnen Sie das **Aufzeichnen Menü** und wählen Sie **Optionen...** aus.



- Wählen Sie Interface **br8** aus und klicken Sie auf den **Start**-Button. Damit kann Wireshark auf dem Subnetz 192.168.88.0/24 alle IPv4 Datenpakete sehen und aufzeichnen.



- Starten Sie auf dem virtuellen PC NMAP das Programm **Zenmap** (Shortcut auf dem Desktop). Zenmap ist eine GUI für das Security-Scan-Programm NMAP, das selbst nur über ein Komandozeileninterface verfügt.
- Tragen Sie im Eingabefeld **Ziel** die IP-Adresse des eigenen Labor-PC ein: **192.168.88.1**. Dieser Host darf gescannt werden.



#### 3.1 SYN SCAN

- Wählen Sie im Feld **Profil:** den **SYN-Scan** aus. Im Feld Befehlszeile sehen Sie die Parameter, die sie eingeben müssten, wenn Sie das Programm NMAP auf der Befehlszeile ausführen würden. Die Bedeutung der Parameter finden Sie unten im Anhang zu dieser Übung.
- Klicken Sie auf **Scan** um den Scan-Vorgang zu starten. Beobachten Sie die Anzeige im **Zenmap Fenster**.

```

Starting Nmap 7.60 ( https://nmap.org ) at 2020-08-21 13:24 CEST
Nmap scan report for 192.168.88.1
Host is up (0.00025s latency).
Not shown: 97 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
443/tcp    open  https
3389/tcp   open  ms-wbt-server
MAC Address: 00:50:56:C0:00:08 (VMware)
Nmap done: 1 IP address (1 host up) scanned in 1.49 seconds

```

- Warten Sie bis die Meldung: **NMAP done** ... erscheint.
- Interpretieren Sie die Anzeige in Zenmap. Klicken Sie auch auf die anderen Anzeige-Register und betrachten Sie dort die Anzeige.
- Betrachten Sie die Anzeige in Wireshark. Analysieren Sie die aufgezeichneten Pakete und beantworten Sie die folgenden Kontrollfragen.

### Kontrollfragen

- ☒ Welche Ports sind offen?
- ☒ Über welche Protokolle wurden die Ports gefunden?
- ☒ Wie weit wurde der 3-Way Handshake bei der TCP-Eröffnung abgewickelt.
- ☒ Welche Antwort wurde bei den offenen Ports zurückgesendet und welche bei den nicht offenen?

## 3.2 ACK SCAN

- Wiederholen Sie den Scan mit der Option für **ACK SCAN**. Zeichnen Sie den Scan wieder mit Wireshark auf und interpretieren Sie die Aufzeichnung. Beantworten Sie die folgenden Kontrollfragen.

### Kontrollfragen

- ☒ Welches Protokoll wurde verwendet?
- ☒ Wie funktioniert der ACK Scan?
- ☒ Sind die Ports gefiltert oder nicht?
- ☒ Wie unterscheidet er sich von SYN Scan?
- ☒ Was ist der Zweck dieses Scans?

## 3.3 FIN SCAN

- Wiederholen Sie den Scan mit der Option für **FIN-Scan**. Zeichnen Sie den Scan wieder mit Wireshark auf und interpretieren Sie die Aufzeichnung. Beantworten Sie die folgenden Kontrollfragen.

### Kontrollfragen

- ☒ Welche Ports sind offen?
- ☒ Über welche Protokolle wurden die Ports gefunden?
- ☒ Wie funktioniert der FIN Scan?

- ☒ Welche Antwort wurde bei den offenen Ports zurückgesendet und welche bei den nicht offenen?

### 3.4 XMAS-Tree SCAN

- 3.4 Wiederholen Sie den Scan mit der Option für **XMAS-Tree-Scan**. Zeichnen Sie den Scan wieder mit Wireshark auf und interpretieren Sie die Aufzeichnung. Beantworten Sie die folgenden Kontrollfragen.

#### Kontrollfragen

- ☒ Welche Ports sind offen?
- ☒ Über welche Protokolle wurden die Ports gefunden?
- ☒ Wie funktioniert der XMAS Scan?
- ☒ Welche Antwort wurde bei den offenen Ports zurückgesendet und welche bei den nicht offenen?

### 3.5 UDP SCAN

- 3.5 Wiederholen Sie den Scan mit einer Option, für **UDP-Scan**. Zeichnen Sie den Scan wieder mit Wireshark auf und interpretieren Sie die Aufzeichnung. Beantworten Sie die folgenden Kontrollfragen.

#### Kontrollfragen

- ☒ Welche Ports sind offen?
- ☒ Über welche Protokolle wurden die Ports gefunden?
- ☒ Wie ist die Antwort bei einem offenen Port und wie bei einem geschlossenen?
- ☒ Warum ist ICMP involviert?

## 4 WEITERE SCANS

*Die folgenden Scans könne, falls noch Zeit ist, ohne Wireshark Messung durchgeführt werden.*

- Führen Sie noch ein paar Scans durch, aber wählen Sie Scan Profile aus, die noch nicht verwendet wurden. Manche Scans dauern mehrere Minuten.
- Warten Sie, bis der jeweilige Scan zu Ende ist und interpretieren Sie die Anzeigen in den verschiedenen Output Fenstern von Zenmap.

ACK-Scan  
FIN-Scan  
Intense scan  
Intense scan plus UDP  
Intense scan, all TCP ports  
Intense scan, no ping  
Ping scan  
Quick scan  
Quick scan plus  
Quick traceroute  
Regular scan  
SYN-Scan  
Slow comprehensive scan  
UDP-Scan  
XMAS-Tree-Scan

## 5 TRACEROUTE ÜBER TCP

Mit NMAP kann ein Traceroute auf TCP Ebene durchgeführt werden, für den Fall, dass ICMP (Ping) von Firewalls geblockt werden würde. Leider funktioniert das nur auf dem Labor-PC und nicht aus der virtuellen Maschine heraus.

- Öffnen Sie die Eingabeaufforderung auf dem Labor-PC (NICHT auf der virtuellen Maschine NMAP)
- Starten Sie eine Wireshark Aufzeichnung auf dem Interface **Internet-blau**.
- Geben Sie den folgenden Befehl ein und verfolgen Sie die Ausführung auch in Wireshark ( Displayfilter: tcp,port==443 || icmp )

```
sudo nmap -p 443 --traceroute 8.8.8.8
```

```
> Frame 155: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface eno1, id 0
> Ethernet II, Src: HewlettP_2a:f9:38 (10:e7:c6:2a:f9:38), Dst: Cisco_e5:a9:80 (70:ca:9b:e5:a9:80)
▼ Internet Protocol Version 4, Src: 194.95.109.143, Dst: 8.8.8.8
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x31 (DSCP: AF12, ECN: ECT(1))
    Total Length: 44
    Identification: 0x0b46 (2886)
> Flags: 0x00
    Fragment Offset: 0
> Time to Live: 1
    Protocol: TCP (6)
    Header Checksum: 0x6e57 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 194.95.109.143
    Destination Address: 8.8.8.8
```

```
> Frame 156: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface eno1, id 0
> Ethernet II, Src: Cisco_e5:a9:80 (70:ca:9b:e5:a9:80), Dst: HewlettP_2a:f9:38 (10:e7:c6:2a:f9:38)
> Internet Protocol Version 4, Src: 194.95.109.129, Dst: 194.95.109.143
▼ Internet Control Message Protocol
    Type: 11 (Time-to-live exceeded)
    Code: 0 (Time to live exceeded in transit)
    Checksum: 0xf3d0 [correct]
    [Checksum Status: Good]
    Unused: 00000000
▼ Internet Protocol Version 4, Src: 194.95.109.143, Dst: 8.8.8.8
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x31 (DSCP: AF12, ECN: ECT(1))
    Total Length: 44
    Identification: 0x0b46 (2886)
> Flags: 0x00
    Fragment Offset: 0
> Time to Live: 1
    Protocol: TCP (6)
    Header Checksum: 0x6e57 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 194.95.109.143
    Destination Address: 8.8.8.8
```

**Ergebnis:** Betrachten Sie die gemessenen Pakete. NMAP sendet einen Bulk von TCP Paketen mit unterschiedlichen TTL. Nach und nach treffen dann die entsprechenden ICMP-

Fehlerreports (Time to live exceeded) der beteiligten Router ein. In der Payload der ICMP Pakete stecken die TCP-Pakete, die zum Fehler geführt haben und können von NMAP entsprechend ausgewertet werden. Die eindeutige Zuordnung von gesendetem Paket und dazugehöriger ICMP Meldung kann durch die ID auf IP-Ebene gewährleistet werden.

## 6 ENDE DER ÜBUNG

- Beenden Sie alle **Programme** und **virtuelle Maschinen**! Im Rahmen der Übung an Ihrem Arbeitsplatz erzielte Messergebnisse können Sie im Labor auf Ihren Memorystick zur späteren Nachbearbeitung abspeichern. Gewonnene sicherheitsrelevante Informationen insbesondere Passwörter, dürfen nicht weitergegeben oder unbefugt verwendet werden. Geht leider nicht im Remotebetrieb.
- **Loggen** Sie sich aus dem Labor-PC **aus**!
- Lassen Sie den PC weiterlaufen. Er wird automatisch ausgeschaltet.

**Bitte hinterlassen Sie Ihren Arbeitsplatz in ordentlichem Zustand!**

**Entsorgen Sie Mitgebrachtes selbst!**

**Schieben Sie den Stuhl an den Tisch!**

# ANHANG: AUSZUG DER MAN-PAGE FÜR NMAP

## SCAN TYPES

**-sT TCP connect() scan:** This is the most basic form of TCP scanning. The connect() system call provided by your operating system is used to open a connection to every interesting port on the machine. If the port is listening, connect() will succeed, otherwise the port isn't reachable. One strong advantage to this technique is that you don't need any special privileges. Any user on most UNIX boxes is free to use this call.

This sort of scan is easily detectable as target host logs will show a bunch of connection and error messages for the services which accept() the connection just to have it immediately shutdown.

**-sS TCP SYN scan:** This technique is often referred to as "half-open" scanning, because you don't open a full TCP connection. You send a SYN packet, as if you are going to open a real connection and you wait for a response. A SYN|ACK indicates the port is listening. A RST is indicative of a non-listener. If a SYN|ACK is received, a RST is immediately sent to tear down the connection (actually our OS kernel does this for us). The primary advantage to this scanning technique is that fewer sites will log it. Unfortunately you need root privileges to build these custom SYN packets.

**-sF -sX -sN**

**Stealth FIN, Xmas Tree, or Null scan modes:** There are times when even SYN scanning isn't clandestine enough. Some firewalls and packet filters watch for SYNs to restricted ports, and programs like Synlog-

ger and Courtney are available to detect these scans. These advanced scans, on the other hand, may be able to pass through unmolested.

The idea is that closed ports are required to reply to your probe packet with an RST, while open ports must ignore the packets in question (see RFC 793 pp 64). The FIN scan uses a bare (surprise) FIN packet as the probe, while the Xmas tree scan turns on the FIN, URG, and PUSH flags. The Null scan turns off all flags. Unfortunately Microsoft (like usual) decided to completely ignore the standard and do things their own way. Thus this scan type will not work against systems running Windows95/NT. On the positive side, this is a good way to distinguish between the two platforms. If the scan finds open ports, you know the machine is not a Windows box. If a -sF, -sX, or -sN scan shows all ports closed, yet a SYN (-sS) scan shows ports being opened, you are probably looking at a Windows box. This is less useful now that nmap has proper OS detection built in. There are also a few other systems that are broken in the same way Windows is. They include Cisco, BSDI, HP/UX, MVS, and IRIX. All of the above send resets from the open ports when they should just drop the packet.

**-sP Ping scanning:** Sometimes you only want to know which hosts on a network are up. Nmap can do this by sending ICMP echo request packets to every IP address on the networks you specify. Hosts that respond are up. Unfortunately, some sites such as microsoft.com block echo request packets. Thus nmap can also send a TCP ack packet to (by default) port 80. If we get an RST back, that machine is up. A third technique involves sending a SYN



packet and waiting for a RST or a SYN/ACK. For non-root users, a connect() method is used.

By default (for root users), nmap uses both the ICMP and ACK techniques in parallel. You can change the -P option described later.

Note that pinging is done by default anyway, and only hosts that respond are scanned. Only use this option if you wish to ping sweep without doing any actual port scans.

**-sU UDP scans:** This method is used to determine which UDP (User Datagram Protocol, RFC 768) ports are open on a host. The technique is to send 0 byte udp packets to each port on the target machine. If we receive an ICMP port unreachable message, then the port is closed. Otherwise we assume it is open.

Some people think UDP scanning is pointless. I usually remind them of the recent Solaris rcpbind hole. Rcpbind can be found hiding on an undocumented UDP port somewhere above 32770. So it doesn't matter that 111 is blocked by the firewall. But can you find which of the more than 30,000 high ports it is listening on? With a UDP scanner you can! There is also the cDc Back Orifice backdoor program which hides on a configurable UDP port on Windows machines. Not to mention the many commonly vulnerable services that utilize UDP such as snmp, tftp, NFS, etc.

Unfortunately UDP scanning is sometimes painfully slow since most hosts implement a suggestion in RFC 1812 (section 4.3.2.8) of limiting the ICMP error message rate. For example, the Linux kernel (in net/ipv4/icmp.h) limits destination unreachable message generation to 80 per 4 seconds, with a 1/4

second penalty if that is exceeded. Solaris has much more strict limits (about 2 messages per second) and thus takes even longer to scan. nmap detects this rate limiting and slows down accordingly, rather than flood the network with useless packets that will be ignored by the target machine.

As is typical, Microsoft ignored the suggestion of the RFC and does not seem to do any rate limiting at all on Win95 and NT machines. Thus we can scan all 65K ports of a Windows machine very quickly. Woop!

**-sO IP protocol scans:** This method is used to determine which IP protocols are supported on a host. The technique is to send raw IP packets without any further protocol header to each specified protocol on the target machine. If we receive an ICMP protocol unreachable message, then the protocol is not in use. Otherwise we assume it is open. Note that some hosts (AIX, HP-UX, Digital UNIX) and firewalls may not send protocol unreachable messages. This causes all of the protocols to appear "open".

Because the implemented technique is very similar to UDP port scanning, ICMP rate limit might apply too. But the IP protocol field has only 8 bits, so at most 256 protocols can be probed which should be possible in reasonable time anyway.

**-sI <zombie host[:probeport]>**

**Idlescan:** This advanced scan method allows for a truly blind TCP port scan of the target (meaning no packets are sent to the target from your real IP address). Instead, a unique side-channel attack exploits predictable "IP fragmentation ID" sequence generation on the zombie host to glean information about the open ports on the target. IDS systems

will display the scan as coming from the zombie machine you specify (which must be up and meet certain criteria). I am planning to put a more detailed explanation up at [http://www.insecure.org/nmap/nmap\\_documentation.html](http://www.insecure.org/nmap/nmap_documentation.html) in the near future.

Besides being extraordinarily stealthy (due to its blind nature), this scan type permits mapping out IP-based trust relationships between machines. The port listing shows open ports from the perspective of the zombie host. So you can try scanning a target using various zombies that you think might be trusted (via router/packet filter rules). Obviously this is crucial information when prioritizing attack targets. Otherwise, you penetration testers might have to expend considerable resources "owning" an intermediate system, only to find out that its IP isn't even trusted by the target host/network you are ultimately after.

a particular port on the zombie host for IPID changes. Otherwise Nmap will use the port it uses by default for "tcp pings".

**-sA ACK scan:** This advanced method is usually used to map out firewall rulesets. In particular, it can help determine whether a firewall is stateful or just a simple packet filter that blocks incoming SYN packets.

This scan type sends an ACK packet (with random looking acknowledgement/sequence numbers) to the ports specified. If a RST comes back, the ports is classified as "unfiltered". If nothing comes back (or if an ICMP unreachable is returned), the port is classified as "filtered". Note that nmap usually doesn't print "unfiltered" ports, so getting no ports shown in the output is usually a sign that

all the probes got through (and returned RSTs). This scan will obviously never show ports in the "open" state.

**-sW Window scan:** This advanced scan is very similar to the ACK scan, except that it can sometimes detect open ports as well as filtered/nonfiltered due to an anomaly in the TCP window size reporting by some operating systems. Systems vulnerable to this include at least some versions of AIX, Amiga, BeOS, BSDI, Cray, Tru64 UNIX, DG/UX, OpenVMS, Digital UNIX, FreeBSD, HP-UX, OS/2, IRIX, MacOS, NetBSD, OpenBSD, OpenStep, QNX, Rhapsody, SunOS 4.X, Ultrix, VAX, and VxWorks. See the nmap-hackers mailing list archive for a full list.

**-sR RPC scan.** This method works in combination with the various port scan methods of Nmap. It takes all the TCP/UDP ports found open and then floods them with SunRPC program NULL commands in an attempt to determine whether they are RPC ports, and if so, what program and version number they serve up. Thus you can effectively obtain the same info as firewall (or protected by TCP wrappers). Decoys do not currently work with RPC scan, at some point I may add decoy support for UDP RPC scans.

**-sL List scan.** This method simply generates and prints a list of IPs/Names without actually pinging or port scanning them. DNS name resolution will be performed unless you use -n.

**-b <ftp relay host>**

**FTP bounce attack:** An interesting "feature" of the ftp protocol (RFC 959) is support for "proxy" ftp connections. In other words, I should be able to

connect from evil.com to the FTP server of target.com and request that the server send a file ANYWHERE on the internet! Now this may have worked well in 1985 when the RFC was written. But in today's Internet, we can't have people hijacking ftp servers and requesting that data be spit out to arbitrary points on the internet. As \*Hobbit\* wrote back in 1995, this protocol flaw "can be used to post virtually untraceable mail and news, hammer on servers at various sites, fill up disks, try to hop firewalls, and generally be annoying and hard to track down at the same time." What we will exploit this for is to (surprise, surprise) scan TCP ports from a "proxy" ftp server. Thus you could connect to an ftp server behind a firewall, and then scan ports that are more likely to be blocked (139 is a good one). If the ftp server allows reading from and writing to some directory (such as /incoming), you can send arbitrary data to ports that you do find open (nmap doesn't do this for you though).

The argument passed to the 'b' option is the host you want to use as a proxy, in standard URL notation. The format is: username:password@server:port. Everything but server is optional. To determine what servers are vulnerable to this attack, you can see my article in Phrack 51. And updated version is available at the nmap URL (<http://www.insecure.org/nmap>).

## GENERAL OPTIONS

None of these are required but some can be quite useful.

**-P0** Do not try and ping hosts at all before scanning them. This allows the scanning of networks that don't allow ICMP echo requests (or responses) through their firewall. microsoft.com is an example of such a network, and thus you should always use -P0 or -PT80 when portscanning microsoft.com.

**-PT** Use TCP "ping" to determine what hosts are up. Instead of sending ICMP echo request packets and waiting for a response, we spew out TCP ACK packets throughout the target network (or to a single machine) and then wait for responses to trickle back. Hosts that are up should respond with a RST. This option preserves the efficiency of only scanning hosts that are up while still allowing you to scan networks/hosts that block ping packets. For non root users, we use connect(). To set the destination port of the probe packets use -PT<port number>. The default port is 80, since this port is often not filtered out.

**-PS** This option uses SYN (connection request) packets instead of ACK packets for root users. Hosts that are up should respond with a RST (or, rarely, a SYN|ACK). You can set the destination port in the same manner as -PT above.

**-PI** This option uses a true ping (ICMP echo request) packet. It finds hosts that are up and also looks for subnet-directed broadcast addresses on your network. These are IP addresses which are externally reachable and translate to a broadcast of incoming IP packets to a subnet of computers. These should be eliminated if found as they allow for numerous denial of service attacks (Smurf is the most common).

**-PB** This is the default ping type. It uses both the ACK ( -PT ) and ICMP ( -PI ) sweeps in parallel. This way you can get firewalls that filter either one (but not both). The TCP probe destination port can be set in the same manner as with -PT above.

**-O** This option activates remote host identification via TCP/IP fingerprinting. In other words, it uses a bunch of techniques to detect subtleties in the

underlying operating system network stack of the computers you are scanning. It uses this information to create a 'fingerprint' which it compares with its database of known OS fingerprints (the `nmap-os-fingerprints` file) to decide what type of system you are scanning.

If Nmap is unable to guess the OS of a machine, and conditions are good (eg at least one open port), Nmap will provide a URL you can use to submit the fingerprint if you know (for sure) the OS running on the machine. By doing this you contribute to the pool of operating systems known to nmap and thus it will be more accurate for everyone. Note that if you leave an IP address on the form, the machine may be scanned when we add the fingerprint (to validate that it works).

The `-O` option also enables several other tests. One is the "Uptime" measurement, which uses the TCP timestamp option (RFC 1323) to guess when a machine was last rebooted. This is only reported for machines which provide this information.

Another test enabled by `-O` is TCP Sequence Predictability Classification. This is a measure that describes approximately how hard it is to establish a forged TCP connection against the remote host. This is useful for exploiting source-IP based trust relationships (rlogin, firewall filters, etc) or for hiding the source of an attack. The actual difficulty number is based on statistical sampling and may fluctuate. It is generally better to use the English classification such as "worthy challenge" or "trivial joke". This is only reported in normal output with `-v`.

When verbose mode (`-v`) is on with `-O`, IPID Sequence Generation is also reported. Most machines are in the "incremental" class, which means that they

increment the "ID" field in the IP header for each packet they send. This makes them vulnerable to several advanced information gathering and spoofing attacks.

`-I` This turns on TCP reverse ident scanning. As noted by Dave Goldsmith in a 1996 Bugtraq post, the ident protocol (rfc 1413) allows for the disclosure of the username that owns any process connected via TCP, even if that process didn't initiate the connection. So you can, for example, connect to the http port and then use `identd` to find out whether the server is running as root. This can only be done with a full TCP connection to the target port (i.e. the `-sT` scanning option). When `-I` is used, the remote host's `identd` is queried for each open port found. Obviously this won't work if the host is not running `identd`.

`-f` This option causes the requested SYN, FIN, XMAS, or NULL scan to use tiny fragmented IP packets. The idea is to split up the TCP header over several packets to make it harder for packet filters, intrusion detection systems, and other annoyances to detect what you are doing. Be careful with this! Some programs have trouble handling these tiny packets. My favorite sniffer segmentation faulted immediately upon receiving the first 36-byte fragment. After that comes a 24 byte one! While this method won't get by packet filters and firewalls that queue all IP fragments (like the `CONFIG_IP_ALWAYS_DEFRAG` option in the Linux kernel), some networks can't afford the performance hit this causes and thus leave it disabled.

Note that I do not yet have this option working on all systems. It works fine for my Linux, FreeBSD, and OpenBSD boxes and some people have reported success with other \*NIX variants.

**-v** Verbose mode. This is a highly recommended option and it gives out more information about what is going on. You can use it twice for greater effect. Use **-d** a couple of times if you really want to get crazy with scrolling the screen!

**-h** This handy option display a quick reference screen of nmap usage options. As you may have noticed, this man page is not exactly a 'quick reference' :)

**-oN <logfilename>**

This logs the results of your scans in a normal human readable form into the file you specify as an argument.

**-oX <logfilename>**

This logs the results of your scans in XML form into the file you specify as an argument. This allows programs to easily capture and interpret Nmap results. You can give the argument **'-'** (without quotes) to shoot output into stdout (for shell pipelines, etc). In this case normal output will be suppressed. Watch out for error messages if you use this (they will still go to stderr). Also note that **'-v'** may cause some extra information to be printed. The Document Type Definition (DTD) defining the XML output structure is available at <http://www.insecure.org/nmap/nmap.dtd>.

**-oG <logfilename>**

This logs the results of your scans in a grepable form into the file you specify as an argument. This simple format provides all the information on one line (so you can easily grep for port or OS information and see all the IPs. This used to be the preferred mechanism for programs to interact with Nmap, but now we recommend XML output (**-oX** instead). This simple format may not contain as much information as the other formats. You can

give the argument **'-'** (without quotes) to shoot output into stdout (for shell pipelines, etc). In this case normal output will be suppressed. Watch out for error messages if you use this (they will still go to stderr). Also note that **'-v'** will cause some extra information to be printed.

**-oA <basefilename>**

This tells Nmap to log in ALL the major formats (normal, grepable, and XML). You give a base for the filename, and the output files will be **base.nmap**, **base.gnmap**, and **base.xml**.

**-oS <logfilename>**

this l0gz th3 r3suLtS of YouR ScanZ iN a s|<ipT kiDd|3 f0rM iNto ThE fiL3 U sPecfy 4s an arGuMENt! U kAn glv3 the 4rgument **'-'** (wIthOUT qUOTEZ) to sh00t output iNT0 stDouT!@!!

**--resume <logfilename>**

A network scan that is cancelled due to control-C, network outage, etc. can be resumed using this option. The logfilename must be either a normal (**-oN**) or machine parsable (**-oM**) log from the aborted scan. No other options can be given (they will be the same as the aborted scan). Nmap will start on the machine after the last one successfully scanned in the log file.

**--append\_output**

Tells Nmap to append scan results to any output files you have specified rather than overwriting those files.

**-iL <inputfilename>**

Reads target specifications from the file specified RATHER than from the command line. The file should contain a list of host or network expressions separated by spaces, tabs, or newlines. Use a hyphen (**-**) as inputfilename if you want nmap to

read host expressions from stdin (like at the end of a pipe). See the section target specification for more information on the expressions you fill the file with.

**-iR** This option tells Nmap to generate its own hosts to scan by simply picking random numbers :). It will never end. This can be useful for statistical sampling of the Internet to estimate various things. If you are ever really bored, try `nmap -sS -iR -p 80` to find some web servers to look at.

**-p <port ranges>**

This option specifies what ports you want to specify. For example `'-p 23'` will only try port 23 of the target host(s). `'-p 20-30,139,60000'` scans ports between 20 and 30, port 139, and all ports greater than 60000. The default is to scan all ports between 1 and 1024 as well as any ports listed in the services file which comes with nmap. For IP protocol scanning (`-sO`), this specifies the protocol number you wish to scan for (0-255).

When scanning both TCP and UDP ports, you can specify a particular protocol by preceding the port numbers by "T:" or "U:". The qualifier lasts until you specify another qualifier. For example, the argument `"-p U:53,111,137,T:21-25,80,139,8080"` would scan UDP ports 53,111, and 137, as well as the listed TCP ports. Note that to scan both UDP & TCP, you have to specify `-sU` and at least one TCP scan type (such as `-sS`, `-sF`, or `-sT`). If no protocol qualifier is given, the port numbers are added to all protocol lists.

**-F** Fast scan mode.

Specifies that you only wish to scan for ports listed in the services file which comes with nmap (or the protocols file for `-sO`). This is obviously much faster than scanning all 65535 ports on a

host.

**-D <decoy1 [,decoy2][,ME],...>**

Causes a decoy scan to be performed which makes it appear to the remote host that the host(s) you specify as decoys are scanning the target network too. Thus their IDS might report 5-10 port scans from unique IP addresses, but they won't know which IP was scanning them and which were innocent decoys. While this can be defeated through router path tracing, response-dropping, and other "active" mechanisms, it is generally an extremely effective technique for hiding your IP address.

Separate each decoy host with commas, and you can optionally use 'ME' as one of the decoys to represent the position you want your IP address to be used. If you put 'ME' in the 6th position or later, some common port scan detectors (such as Solar Designer's excellent scanlogd) are unlikely to show your IP address at all. If you don't use 'ME', nmap will put you in a random position.

Note that the hosts you use as decoys should be up or you might accidentally SYN flood your targets. Also it will be pretty easy to determine which host is scanning if only one is actually up on the network. You might want to use IP addresses instead of names (so the decoy networks don't see you in their nameserver logs).

Also note that some (stupid) "port scan detectors" will firewall/deny routing to hosts that attempt port scans. Thus you might inadvertently cause the machine you scan to lose connectivity with the decoy machines you are using. This could cause the target machines major problems if the decoy is, say, its internet gateway or even "localhost". Thus you might want to be careful of this option. The real moral of the story is that detectors of

spoofable port scans should not take action against the machine that seems like it is port scanning them. It could just be a decoy!

Decoys are used both in the initial ping scan (using ICMP, SYN, ACK, or whatever) and during the actual port scanning phase. Decoys are also used during remote OS detection ( `-O` ).

It is worth noting that using too many decoys may slow your scan and potentially even make it less accurate. Also, some ISPs will filter out your spoofed packets, although many (currently most) do not restrict spoofed IP packets at all.

#### `-S <IP_Address>`

In some circumstances, nmap may not be able to determine your source address ( nmap will tell you if this is the case). In this situation, use `-S` with your IP address (of the interface you wish to send packets through).

Another possible use of this flag is to spoof the scan to make the targets think that someone else is scanning them. Imagine a company being repeatedly port scanned by a competitor! This is not a supported usage (or the main purpose) of this flag. I just think it raises an interesting possibility that people should be aware of before they go accusing others of port scanning them. `-e` would generally be required for this sort of usage.

#### `-e <interface>`

Tells nmap what interface to send and receive packets on. Nmap should be able to detect this but it will tell you if it cannot.

#### `-g <portnumber>`

Sets the source port number used in scans. Many naive firewall and packet filter installations make

an exception in their ruleset to allow DNS (53) or FTP-DATA (20) packets to come through and establish a connection. Obviously this completely subverts the security advantages of the firewall since intruders can just masquerade as FTP or DNS by modifying their source port. Obviously for a UDP scan you should try 53 first and TCP scans should try 20 before 53. Note that this is only a request -- nmap will honor it only if and when it is able to. For example, you can't do TCP ISN sampling all from one host:port to one host:port, so nmap changes the source port even if you used `-g`.

Be aware that there is a small performance penalty on some scans for using this option, because I sometimes store useful information in the source port number.

`-n` Tells Nmap to NEVER do reverse DNS resolution on the active IP addresses it finds. Since DNS is often slow, this can help speed things up.

`-R` Tells Nmap to ALWAYS do reverse DNS resolution on the target IP addresses. Normally this is only done when a machine is found to be alive.

`-r` Tells Nmap NOT to randomize the order in which ports are scanned.

#### `--randomize_hosts`

Tells Nmap to shuffle each group of up to 2048 hosts before it scans them. This can make the scans less obvious to various network monitoring systems, especially when you combine it with slow timing options (see below).

#### `-M <max sockets>`

Sets the maximum number of sockets that will be used in parallel for a TCP connect() scan (the default). This is useful to slow down the scan a

little bit and avoid crashing remote machines. Another approach is to use `-sS`, which is generally easier for machines to handle.

## TIMING OPTIONS

Generally Nmap does a good job at adjusting for Network characteristics at runtime and scanning as fast as possible while minimizing that chances of hosts/ports going undetected. However, there are same cases where Nmap's default timing policy may not meet your objectives. The following options provide a fine level of control over the scan timing:

`-T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane>`

These are canned timing policies for conveniently expressing your priorities to Nmap. Paranoid mode scans very slowly in the hopes of avoiding detection by IDS systems. It serializes all scans (no parallel scanning) and generally waits at least 5 minutes between sending packets. Sneaky is similar, except it only waits 15 seconds between sending packets. Polite is meant to ease load on the network and reduce the chances of crashing machines. It serializes the probes and waits at least 0.4 seconds between them. Normal is the default Nmap behaviour, which tries to run as quickly as possible without overloading the network or missing hosts/ports. Aggressive mode adds a 5 minute timeout per host and it never waits more than 1.25 seconds for probe responses. Insane is only suitable for very fast networks or where you don't mind losing some information. It times out hosts in 75 seconds and only waits 0.3 seconds for individual probes. It does allow for very quick network sweeps though :). You can also reference these by number (0-5). For example, `-T 0` gives you Paranoid mode and `-T 5` is Insane mode.

These canned timing modes should NOT be used in combination with the lower level controls given below.

`--host_timeout <milliseconds>`

Specifies the amount of time Nmap is allowed to spend scanning a single host before giving up on that IP. The default timing mode has no host timeout.

`--max_rtt_timeout <milliseconds>`

Specifies the maximum amount of time Nmap is allowed to wait for a probe response before retransmitting or timing out that particular probe. The default mode sets this to about 9000.

`--min_rtt_timeout <milliseconds>`

When the target hosts start to establish a pattern of responding very quickly, Nmap will shrink the amount of time given per probe. This speeds up the scan, but can lead to missed packets when a response takes longer than usual. With this parameter you can guarantee that Nmap will wait at least the given amount of time before giving up on a probe.

`--initial_rtt_timeout <milliseconds>`

Specifies the initial probe timeout. This is generally only useful when scanning firewalled hosts with `-P0`. Normally Nmap can obtain good RTT estimates from the ping and the first few probes. The default mode uses 6000.

`--max_parallelism <number>`

Specifies the maximum number of scans Nmap is allowed to perform in parallel. Setting this to one means Nmap will never try to scan more than 1 port at a time. It also effects other parallel