

Nmap Tutorial

yudhax

Introduction to Nmap

Nmap is the network exploration tool, it is essentially one of the most important tools to a security engineer or pen-tester. It is used as it's name suggests as a network exploration tool. With nmap you can probe a entire network and find out what services are listening on each specific port. Not only that but it incorporates fingerprinting that compares different fingerprints and gives you a estimate on what operating system the machine is running. Nmap has allot of options or flags that let you manipulate how it scans, you can simply do a tcp()connect scan that makes a full connection to the host or a syn scan also known as half connection, test firewall rules and distinguish if they are firewalls or packet filters, idle scan and spoof your ip through another machine or throw out decoys to make your presence less traceable. Nmap runs on linux/bsd and windows, although we will only be discussing it's usage under linux, the windows version is just a port from linux and can still be used as a supplement if you want but you do have access to the linux version in the attack lab.

Options and Flags

The nmap options or flags are a set of inbuilt variables that help you modify how nmap probes machines.

By simply typing nmap at the command prompt you will get a breif explanation of each flag.

```
[root@REDHATBOX root]# nmap
Nmap 3.30 Usage: nmap [Scan Type(s)] [Options]
```

```
Some Common Scan Types ('*' options require root privileges)
* -sS TCP SYN stealth port scan (default if privileged (root))
-sT TCP connect() port scan (default for unprivileged users)
* -sU UDP port scan
-sP ping scan (Find any reachable machines)
* -sF,-sX,-sN Stealth FIN, Xmas, or Null scan (experts only)
-sR/-I RPC/Identd scan (use with other scan types)
Some Common Options (none are required, most can be combined):
* -O Use TCP/IP fingerprinting to guess remote operating system
-p ports to scan. Example range: '1-1024,1080,6666,31337'
-F Only scans ports listed in nmap-services
-v Verbose. Its use is recommended. Use twice for greater effect.
-P0 Don't ping hosts (needed to scan www.microsoft.com and others)
* -Ddecoy_host1,decoy2[,...] Hide scan using many decoys
-6 scans via IPv6 rather than IPv4
-T General timing policy
-n/-R Never do DNS resolution/Always resolve [default: sometimes resolve]
-oN/-oX/-oG Output normal/XML/grepable scan logs to
-iL Get targets from file; Use '-' for stdin
* -S /-e Specify source address or network interface
--interactive Go into interactive mode (then press h for help)
```

```
Example: nmap -v -sS -O www.my.com 192.168.0.0/16 '192.88-90.*.*'
SEE THE MAN PAGE FOR MANY MORE OPTIONS, DESCRIPTIONS, AND EXAMPLES
```

Syn/Stealth Scanning. -sS TCP SYN stealth port scan (default if privileged (root))

The first and most widely used method is syn scanning also known as half open or stealth, unfortunately most decent ids (intrusion detection systems) can detected these packets and some firewalls and packet filterers will drop syn packets not allowing you to get a fingerprint of what the host is running. With a syn/stealth scan you do not actually make a full connection when scanning what happens is you send a syn packet and request a connection, the host being scanned then responds

Nmap Tutorial

yudhax

with a syn/ack packet telling you weather or not the port is open and responding, as soon as you receive the syn/ack packet from the remote host nmap sends a rst packet terminating the connection. So you don't actually make a full connection or 3 way handshake, that's why Syn/stealth scanning is called a half scan, due to the fact that nmap sends a rst packet before a full connection is established.

Issuing a Syn/Stealth scan.

```
[root@REDHATBOX root]#nmap -sS 192.168.0.1

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-17 05:07 EST
Interesting ports on 192.168.0.4:
(The 1637 ports scanned but not shown below are in state: closed)
Port State Service
21/tcp filtered ftp
22/tcp open  ssh
23/tcp open  telnet
111/tcp open  sunrpc
139/tcp open  netbios-ssn
1024/tcp open kdm
6000/tcp open X11

Nmap run completed -- 1 IP address (1 host up) scanned in 3.194 seconds
```

Notice how port 21 is filtered, a filtered port usually indicates the machine is running a firewall.

Here is a snort log of syn/stealth scanning.

```
[**] [111:13:1] spp_stream4: STEALTH ACTIVITY (SYN FIN scan) detection [**]
07/15-14:45:47.877211 192.168.0.3:10004 -> 202.87.19.229:1002
TCP TTL:255 TOS:0x0 ID:2304 IpLen:20 DgmLen:40
*****SF Seq: 0x90AB213 Ack: 0x0 Win: 0x1000 TcpLen: 20

Jul 16 11:52:17 192.168.0.4:1460 -> 192.168.0.3:1109 SYN *****S*
Jul 16 11:52:17 192.168.0.4:1461 -> 192.168.0.3:317 SYN *****S*
Jul 16 11:52:17 192.168.0.4:1462 -> 192.168.0.3:174 SYN *****S*
Jul 16 11:52:17 192.168.0.4:1463 -> 192.168.0.3:504 SYN *****S*
Jul 16 11:52:17 192.168.0.4:1464 -> 192.168.0.3:343 SYN *****S*
Jul 16 11:52:17 192.168.0.4:1465 -> 192.168.0.3:672 SYN *****S*
```

As you can see snort has built a ruleset that is able to identify nmap's syn/stealth scanning sequence.

TCP()Connect Scanning. -sT TCP connect() port scan (default for unprivileged users)

Tcp connect scanning is the most basic or primitive form of scanning, almost all MS windows scanners use this method. TCP()Connect scanning is the fastest method of scanning, as it calls up system connect() and you can specify how many sockets you would like to use ,so 60 sockets means 60 connections at once. Yes that's fast. Now with the down sides of TCP() connect scanning, it's noisy every machine in the world will log a tcp connection (()connect) request so every time you send one your being logged by whatever service you make the connection too. So if you make full connections to every service listening on every port people will know exactly what you are doing.

Issuing a TCP()Connect scan

```
[root@REDHATBOX root]# nmap -sT 192.168.0.3
```

Nmap Tutorial

yudhax

```
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-17 10:02 EST
Interesting ports on 192.168.0.3:
(The 1629 ports scanned but not shown below are in state: closed)
Port State Service
9/tcp open discard
13/tcp open daytime
21/tcp open ftp
22/tcp open ssh
25/tcp open smtp
37/tcp open time
80/tcp open http
111/tcp open sunrpc
113/tcp open auth
139/tcp open netbios-ssn
443/tcp open https
515/tcp open printer
993/tcp open imaps
995/tcp open pop3s
9999/tcp open abyss
```

Another thing you can do with the -sT scan is DOS (Denial Of Service) a machine.

I issued nmap the following

```
[root@REDHATBOX root]# nmap -T 5 -M 1000 -sT 192.168.0.1
Warning: Your max_parallelism (-M) option is absurdly high! Don't complain to
Fyodor if all hell breaks loose!
```

```
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-18 06:44 EST
All 1644 scanned ports on 192.168.0.1 are: filtered
```

The machine i pointed this at was a windows xp professional box, the effects were obvious, the machine stooped responding and the mouse locked up. You can crash windows firewalls with these scans aswell so take care are using this method.

If you look at the command i gave nmap -T 5 -M 1000.

The -M flag sets the maximum amount of sockets nmap uses and 60 is classified as being allot, so 1000 is absurdly high according to fyodor and i must agree. But it was effective.

The -T flag sets the rate or speed nmap scans hosts ranging from 0-5, 0 being the slowest and 5 being the fastest.

0 = Paranoid Tries to avoid IDS detection, no parallel scanning.
Waits 5 minutes before sending each packet, so very very slow.

1 = Sneaky also tries to avoid IDS detection, no parallel scanning.
Waits 15 seconds before sending packets.

2 = Polite Still very slow, and should be used against mission critical systems only,
will be detected by all IDS machines. Waits at least 0.4 seconds between packets,
more like about 1 sec per packet.

3 = Normal is nmaps default scanning speed and goes as fast as possible without

Nmap Tutorial

yudhax

the risk of Denial Of Service.

4 = Aggressive is ok for fast networks, it can help against firewalls and heavily filtered networks. Recommended for people on cable/adsl and t1 networks ect..

5 = Insane is insane times out after 0.3 seconds, so you will miss a lot of information. Great for people on a fast connections and recommended for network sweeps.

Here is a example of the trace from snort you leave on a system if you TCP() connect scan it. Not only will all IDS machines log these scans but so will every daemon or service you connect to e.g ftp,sendmail,telnet,ssh,samba,

```
[**] [100:2:1] spp_portscan: portscan status from 192.168.0.4: 261 connections  
across 1 hosts: TCP(261), UDP(0) [**]  
07/16-12:01:44.071271
```

```
[**] [1:469:1] ICMP PING NMAP [**]  
[Classification: Attempted Information Leak] [Priority: 2]  
07/16-12:01:46.050056 192.168.0.4 -> 192.168.0.3  
ICMP TTL:45 TOS:0x0 ID:17750 IpLen:20 DgmLen:28  
Type:8 Code:0 ID:31266 Seq:8282 ECHO  
[Xref => http://www.whitehats.com/info/IDS162]
```

```
Jul 16 12:19:39 192.168.0.4:2418 -> 192.168.0.3:7006 SYN *****S*  
Jul 16 12:19:39 192.168.0.4:2419 -> 192.168.0.3:7006 SYN *****S*  
Jul 16 12:19:39 192.168.0.4:2420 -> 192.168.0.3:7006 SYN *****S*
```

You will notice that the main protocol is TCP (TCP(261)). And it established a lot of 3 way connections.

UDP scan -sU UDP port scan

The very simple connectionless User Data Protocol that is used for transmitting datagrams only. Basicly what the udp scans does is send datagrams and waits for a error response message from the host, it then calualates what ports are open by not receiveing a error message suggesting it is open. Udp scanning is extemely slow, due to service response limitation of over atleast 1-4 seconds due to rfc compliance on *nix type systems, however windows has no set limitations so it will scan alot faster. I also suggest only scanning with udp as root consideing the limitations non root users face in distinguishing open and closed ports. Sure udp scanning can find all udp services but so can TCP()Connect scans and TCP()Connect. For me personaly i think udp scanning is a waste of time. But to get a true finger print of each machine it might be wise to know what you have got listening on each udp port as allot of trojans listen on udp.

Issuing a UDP scan

```
[root@REDHATBOX root]# nmap -sU 192.168.0.3  
  
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-17 11:05 EST  
Interesting ports on 192.168.0.3:  
(The 1467 ports scanned but not shown below are in state: closed)  
Port State Service  
9/udp open discard  
111/udp open sunrpc  
137/udp open netbios-ns  
138/udp open netbios-dgm  
  
Nmap run completed -- 1 IP address (1 host up) scanned in 1473.816 seconds
```

Nmap Tutorial

yudhax

As you can see all the ports listed above are udp ports. To make things more interesting here is a scan from a windows xp machine.

```
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-17 11:41 EST
Interesting ports on 192.168.0.1:
(The 1463 ports scanned but not shown below are in state: closed)
Port State Service
53/udp open domain
123/udp open ntp
135/udp open loc-srv
137/udp open netbios-ns
138/udp open netbios-dgm
445/udp open microsoft-ds
500/udp open isakmp
1900/udp open UPnP
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 7.285 seconds
```

If you have a look at the bottom of each scan you will notice the scan for the debian linux box took over 200 times longer than the xp pro box. Thats because as mentioned above the *nix systems complying to RFC standards.

During the scan on the debian box we set off some major bells and whistles. And on the windows xp box we fillterd the entire scan until the firewalls where turned off.

Typical firewall, blocks nmap udp scans in it's tracks.

```
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-17 12:10 EST
All 1471 scanned ports on 192.168.0.1 are: filtered
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 94.786 seconds
```

Here is a snippet of the noise the udp scan made on the debian linux box running snort.

```
[**] [1:279:2] DOS Bay/Nortel Nautica Marlin [**]
[Classification: Attempted Denial of Service] [Priority: 2]
07/16-13:18:46.140390 192.168.0.4:40310 -> 192.168.0.3:161
UDP TTL:56 TOS:0x0 ID:43196 IpLen:20 DgmLen:28
Len: 8
[Xref => http://www.securityfocus.com/bid/1009]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0221]

[**] [1:1042:2] WEB-IIS view source via translate header [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
07/16-13:53:47.031804 192.168.0.1:4752 -> 192.168.0.3:80
TCP TTL:128 TOS:0x0 ID:9488 IpLen:20 DgmLen:187 DF
***AP*** Seq: 0x231C3970 Ack: 0x4242DB37 Win: 0xFAF0 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS305]

[**] [1:1042:2] WEB-IIS view source via translate header [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
07/16-13:53:47.090385 192.168.0.1:4752 -> 192.168.0.3:80
TCP TTL:128 TOS:0x0 ID:9490 IpLen:20 DgmLen:204 DF
***AP*** Seq: 0x231C3A03 Ack: 0x4242DC29 Win: 0xF9FE TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS305]
```

Nmap Tutorial

yudhax

Snort did not detect the actual portscan as a portscan but it did report that the system was under attack. Not the greatest scanning technique.

Pinging -sP ping scan (Find any reachable machines)

All pinging does is distinguish if a host is up or down. It sends out ICMP echo request to hosts and if they respond there up.

Simple packet trace of a ping query. You can see the initial ICMP echo request and response by adding the following flag to a nmap scan.

```
[root@REDHATBOX root]# nmap --packet_trace -sP 192.168.0.*

SENT (1.1890s) ICMP 192.168.0.4 > 192.168.0.1 Echo request (type=8/code=0) ttl=54
id=32934 iplen=28
SENT (1.1890s) TCP 192.168.0.4:51585 > 192.168.0.1:80 A ttl=59 id=44623 iplen=40
seq=3782306078 win=4096 ack=3782306078
RCVD (1.1930s) ICMP 192.168.0.1 > 192.168.0.4 Echo reply (type=0/code=0) ttl=128
id=4482 iplen=28
```

Some sites block pings as it is quite effective in cloaking the host from some scanners.

Heres some basic tricks you can do with -sP ping.

Say if you wanted to know every machine that is up on a class C network you would issue nmap.

```
[root@REDHATBOX root]# nmap -sP 192.168.0.*

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-17 13:40 EST
Host 192.168.0.0 seems to be a subnet broadcast address (returned 1 extra pings).
Host 192.168.0.1 appears to be up.
Host 192.168.0.3 appears to be up.
Host 192.168.0.4 appears to be up.
Host 192.168.0.255 seems to be a subnet broadcast address (returned 1 extra pings).
Nmap run completed -- 256 IP addresses (3 hosts up) scanned in 11.721 seconds

Nmap scanned 256 ip address and found 3 host up.
```

You can also scan class b networks by doing the following

```
[root@REDHATBOX root]# nmap -sP 202.12.*.*

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-17 13:53 EST
Host me-202-12-3-129.btw.co.nz (202.12.3.129) appears to be up.
Host me-202-12-3-130.btw.co.nz (202.12.3.130) appears to be up.
Host me-202-12-3-135.btw.co.nz (202.12.3.135) appears to be up.
Host me-202-12-3-137.btw.co.nz (202.12.3.137) appears to be up.
```

If i had not of stopped nmap it would have scanned 65536 hosts. From 202.12.0.0 to 202.12.256.256. There are other ways of scanning class c networks using the slash e.g 192.168.0.0/24 and for class b use /16. Not only that but you can even add more options for example every host in class be that starts with 1. You would issue nmap the following.

```
[root@REDHATBOX root]# nmap -sP 202.12.*.1
```

Nmap Tutorial

yudhax

```
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-17 13:53 EST
Host scorpio.psu.ac.th (202.12.74.1) appears to be up.
Host 202.12.92.1 appears to be down.
Host 202.12.93.1 appears to be down.
Host 202.12.94.1 appears to be down.
Host 202.12.95.1 appears to be down.
Host 202.12.96.1 appears to be down.
Host kkul.kku.ac.th (202.12.97.1) appears to be up.
```

I will get more into this stuff later in the lesson.

Ftp Bounce Attack -b

This was a good fun, unfortunately it's no longer possible. So i won't bother going into detail.

```
[root@REDHATBOX root]# nmap -b 192.168.0.4 192.168.0.1
Hint: if your bounce scan target hosts aren't reachable from here, remember to use
-P0 so we don't try and ping them prior to the scan
```

```
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-18 09:25 EST
Your ftp bounce server doesn't allow priviliged ports, skipping them.
Your ftp bounce server doesn't allow priviliged ports, skipping them.
Your ftp bounce server sucks, it won't let us feed bogus ports!
```

this is the error you will get on every ftp server that you try this on.

Idle scanning -si

Say if you where scanning the pentagon for some unknown reason. Would you want them to know youre IP address ? no. Well you can use Idle scanning. It uses a zombie host to transmit the packets using it's IPID, ip identification number so everything that hit's the machine does not originate from you but the host you are using as the zombie. It uses the same technique as a Syn scan, the other 2 factors is the fragment identification number, nmap can source enough information from the FIN to know how many packets have been transmisted since the last packet you sent. And your zombie host must be idle.

So with idle scanning we can determine a the targets status by a increment in the zombies IPID number, if our target sends a rst packet to our zombie host we know the port is closed, but if we get a syn ack sent to our zombie host we know the port is open because the zombie then has to respond to the connection. You should know by know what the zombie responds with ? remember it's just a Syn scan, so the zombie terminates the connect with a reset, rst packet. Unfortunately not all IPID stacks are predictable, as far as i know due to microsofts ignorant defiance of RFC stands all windows machine should be suitable as zombies, older linux machines should also be fine for zombies. You just have to keep trying until you find a suitable zombie with a predictable IPID. Other factors with Idle scanning is that i recommend using nmaps default tcp port 80 to be accessable on the zombie and a windows machine.

Here is how predictable a windows machine is.

```
SENT (98.9480s) TCP 192.168.0.4:33609 >192.168.0.1:828 A ttl=47 id=14515 iplen=40
seq=1085816310 win=4096 ack=1085816310
RCVD (98.9640s) TCP 192.168.0.1:3344 > 192.168.0.4:22 A ttl=128 id=44516 iplen=40
seq=1375280861 win=64240 ack=1375280861
RCVD (99.1390s) TCP 192.168.0.1:3344 > 192.168.0.4:22 A ttl=128 id=44517 iplen=40
seq=1375280861 win=64100 ack=1375280861
RCVD (99.3390s) TCP 192.168.0.1:3344 > 192.168.0.4:22 A ttl=128 id=44518 iplen=40
seq=1375280861 win=63960 ack=1375280861
```

Nmap Tutorial

yudhax

As you can see the id= sequence is in a increments of 1.

Here is a redhat linux 8.0 machine.

```
RCVD (17.2850s) TCP 127.0.0.1:44076 > 192.168.0.4:360 A ttl=51 id=40722 iplen=40
seq=3848151658 win=4096 ack=3848151658
RCVD (17.2850s) TCP 192.168.0.4:360 > 192.168.0.4:44076 R ttl=64 id=0 iplen=40
seq=3767092268 win=0
RCVD (17.2850s) TCP 127.0.0.1:44076 > 192.168.0.4:3086 A ttl=47 id=63447 iplen=40
seq=3848151658 win=4096 ack=3848151658
RCVD (17.2850s) TCP 192.168.0.4:3086 > 192.168.0.4:44076 R ttl=64 id=0 iplen=40
seq=3767092268 win=0
```

The redhat box is too unpredictable to use as a Idle scan zombie. You must realize that for the idle scan to work the IPID number can only increase by 1 or 2, 1 being closed 2 being open.

To issue a idle scan using nmaps default tcp() port 80 use.

```
[root@REDHATBOX root]# nmap -P0 -p- -sI 192.168.0.1 192.168.0.3

Starting nmap V. 3.30 ( www.insecure.org/nmap/ )
Idlescan using zombie 192.168.0.1; Class: Incremental
Interesting ports on 192.168.0.3:
(The 65522 ports scanned but not shown below are in state: closed)
Port State Service
9/tcp open discard
13/tcp open daytime
21/tcp open ftp
22/tcp open ssh
25/tcp open smtp
37/tcp open time
80/tcp open http
111/tcp open sunrpc
113/tcp open auth
139/tcp open netbios-ssn
443/tcp open https
515/tcp open printer
993/tcp open imaps
995/tcp open pop3s
9999/tcp open abyss
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 1348.167seconds
```

If the target checks it's logs there will be no trace of your ip address, only the zombies. Some people do not realize the effects of Idle scanning, scans like this can crumble an entire network if the conditions are suitable. For example if the network admin has some type of packet filtering device running that co-exists with some form of rule set that blocks offending ip's from the network you could just spoof the dns server or mail server resulting in major denial of service.

To find Predictable IPID's you will have to start trawling the internet with nessus. Here is the information from nessus. It's plugin ID is 10201.

The remote host uses non-random IP IDs, that is, it is possible to predict the next value of the ip_id field of the ip packets sent by this host.

Nmap Tutorial

yudhax

An attacker may use this feature to determine if the remote host sent a packet in reply to another request. This may be used for portscanning and other things.

When you do locate a machine with a predictable IPID you can then exploit certain areas of the network it is located on, considering it will be a trusted host on it's network.

-D Decoy Scanning

Decoy scanning can be used to effectively DOS or confuse the intended target. The decoy method is best described as making invalid connections on the behalf of a unaware host, basically you are sending spoofed packets with a fake source address along with your original address hoping to make it harder to find out exactly who is scanning them. If your ISP has egress filters all spoofing would be pointless but i suggest you still try it, because it is not a common implementation yet. You can also DOS a machine by sending spoofed packets on behalf of a trusted host, this only works if the machine blocks offending ip's from the network. Another thing to note is the more decoys the slower the scan for obvious reasons.

```
Jul 23 17:53:45 192.168.0.7:59292 -> 192.168.0.3:3455 SYN *****S*
Jul 23 17:53:45 192.168.0.7:59292 -> 192.168.0.3:173 SYN *****S*
Jul 23 17:53:44 192.168.0.4:59292 -> 192.168.0.3:32 SYN *****S*
Jul 23 17:53:44 192.168.0.4:59292 -> 192.168.0.3:759 SYN *****S*
```

Judging from the above log we where scanned by 2 offending ip address's one real and the other fake the issued Decoy scan was

```
[root@REDHATBOX test]# nmap -p 80,22,139 -ss -D 192.168.0.7,192.168.0.1
192.168.0.3
```

```
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-24 15:58 EST
Interesting ports on 192.168.0.3:
Port State Service
22/tcp open  ssh
80/tcp open  http
139/tcp open netbios-ssn
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 8.879 seconds
```

there is a huge mistake in this scan though ? why decoy scan someone if your going to ping them ?

```
SENT (0.0060s) ICMP 192.168.0.4 > 192.168.0.3 Echo request (type=8/code=0) ttl=42
id=2717 iplen=28
SENT (0.0070s) ICMP 192.168.0.7 > 192.168.0.3 Echo request (type=8/code=0) ttl=58
id=27663 iplen=28
SENT (0.0070s) ICMP 192.168.0.1 > 192.168.0.3 Echo request (type=8/code=0) ttl=41
id=3618 iplen=28
RCVD (0.0070s) ICMP 192.168.0.3 > 192.168.0.4 Echo reply (type=0/code=0) ttl=64
id=37318 iplen=28
```

```
[**] [1:469:1] ICMP PING NMAP [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/23-18:46:13.256183 192.168.0.4 -> 192.168.0.3
ICMP TTL:42 TOS:0x0 ID:2717 IpLen:20 DgmLen:28
Type:8 Code:0 ID:53476 Seq:52818 ECHO
[Xref => http://www.whitehats.com/info/IDS162]
```

Nmap Tutorial

yudhax

```
[**] [1:469:1] ICMP PING NMAP [**]  
[Classification: Attempted Information Leak] [Priority: 2]  
07/23-18:46:13.256190 192.168.0.7 -> 192.168.0.3  
ICMP TTL:58 TOS:0x0 ID:27663 IpLen:20 DgmLen:28  
Type:8 Code:0 ID:53476 Seq:52818 ECHO  
[Xref => http://www.whitehats.com/info/IDS162]
```

Ok so both ip's show up in a snort log ? who cares. I do because your ip might set of more warnings in snort than than the decoys. I have seen this happen once and I'm unable to reproduce it. And i do know that the real attackers ip in a decoy scan always shows up first in the snort alert log.

Disable pinging with the -P0 flag.

* -sF,-sX,-sN Stealth FIN, Xmas, or Null scan (experts only)

Ok, say we are scanning a very paranoid network that drops syn and tcp()connect packets from the firewall.

heres and example of a major locked down server.

```
root@xboxLINUX:~# nmap -sS 192.168.0.4
```

```
Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )  
Interesting ports on (192.168.0.4):  
(The 1553 ports scanned but not shown below are in state: filtered)  
Port State Service  
22/tcp open  ssh
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 159 seconds
```

As you can see this machine (redhat linux 8.0) has filtered everything except ssh. Mind you this can still give us some information.

Issue the following command telnet 192.168.0.4 22

```
root@xboxLINUX:~# telnet 192.168.0.4 22  
Trying 192.168.0.4...  
Connected to 192.168.0.4.  
Escape character is '^]'.  
SSH-1.99-OpenSSH_3.4p1
```

We got what version of ssh it is running SSH-1.99-OpenSSH_3.4p1. But that still not enough for even a OS fingerprint, so we need to turn to Stealth FIN, Xmas, or Null scanning. Before i get into details lets get a decent fingerprint on our target 192.168.0.4

I issued nmap a stealth FIN scan.

```
root@xboxLINUX:~# nmap -sF 192.168.0.4  
Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )  
Interesting ports on (192.168.0.4):  
(The 1547 ports scanned but not shown below are in state: closed)  
Port State Service  
21/tcp open  ftp  
22/tcp open  ssh  
23/tcp open  telnet  
111/tcp open sunrpc
```

Nmap Tutorial

yudhax

```
139/tcp open netbios-ssn
1024/tcp open kdm
6000/tcp open X11
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 14 seconds
```

Jackpot with this result we can safely say that our target is running a firewall due to the fact that the SYN/stealth scan only gave us one open port but the FIN/stealth scan gave us all available open ports. Why does this beat firewalls? Instead of setting the SYN flag we set a FIN flag and use the exact same method of a half scan, by using this technique and sending a FIN instead we can determine if a port is open by getting a RST packet back, and on a closed port we get no response. You see TCP is designed around rules and there are different rule sets for each communication. The TCP stack is required to respond to FIN packets in this way. Letting us exploit them.

Ok I got a Windows box firewalled up, let's test it with a FIN.

```
root@xboxLINUX:~# nmap -sF 192.168.0.1
```

```
Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
All 1554 scanned ports on (192.168.0.1) are: filtered
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 105 seconds
```

Ok Nmap is telling us that the Windows box is all filtered. The reason is that Windows boxes just send a RST packet to every FIN. So even if they are open we still get a RST. So FIN scanning does not work on Windows machines. So I go at the Windows box with a SYN/stealth scan.

```
[root@REDHATBOX root]# nmap -sS 192.168.0.1
```

```
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-20 17:06 EST
Note: Host seems down. If it is really up, but blocking our ping probes, try -P0
Nmap run completed -- 1 IP address (0 hosts up) scanned in 12.206 seconds
```

Now Nmap is telling me that the Windows box is down? or blocking our pings. To get around firewalls like this that block incoming ICMP requests disable pinging with the -P0

```
[root@REDHATBOX root]# nmap -P0 -sS 192.168.0.1
```

```
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-21 08:39 EST
Interesting ports on 192.168.0.1:
(The 1639 ports scanned but not shown below are in state: filtered)
Port State Service
135/tcp open loc-srv
1002/tcp open unknown
1025/tcp open NFS-or-IIS
1720/tcp open H.323/Q.931
5000/tcp open UPnP
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 99.125 seconds
```

Another thing to note with Windows boxes is that usually a SYN/stealth scanning can get around these lame firewalls. And with some Windows firewalls you can actually crash them with simple TCP()connect scans. I just crashed a Sygate firewall when running this scan.

```
[root@REDHATBOX root]# nmap -P0 -sT 192.168.0.1
```

Nmap Tutorial

yudhax

```
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-21 08:43 EST
Interesting ports on 192.168.0.1:
(The 1632 ports scanned but not shown below are in state: closed)
Port State Service
98/tcp filtered linuxconf
135/tcp open loc-srv
139/tcp open netbios-ssn
389/tcp open ldap
445/tcp open microsoft-ds
528/tcp filtered custix
1002/tcp open unknown
1025/tcp open NFS-or-IIS
1337/tcp open waste
1720/tcp open H.323/Q.931
1995/tcp filtered perf-port
5000/tcp open UPnP
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 293.771 seconds
```

Sorry got side tracked there but it is important information when pen-testing a network.

Now the other 2 scan types will give you the same results, the Xmas scan just turns on 2 extra flags and the null scan has no flags set.

Snort will log FIN scans. As you can see below.

```
Jul 21 13:10:21 192.168.0.4:39290 -> 192.168.0.3:571 FIN *****F
Jul 21 13:10:21 192.168.0.4:39290 -> 192.168.0.3:27007 FIN *****F
Jul 21 13:10:22 192.168.0.4:39291 -> 192.168.0.3:22 FIN *****F
Jul 21 13:10:22 192.168.0.4:39291 -> 192.168.0.3:9 FIN *****F
Jul 21 13:10:23 192.168.0.4:39291 -> 192.168.0.3:80 FIN *****F
Jul 21 13:10:22 192.168.0.4:39290 -> 192.168.0.3:139 FIN *****F
Jul 21 13:10:22 192.168.0.4:39290 -> 192.168.0.3:515 FIN *****F
Jul 21 13:10:22 192.168.0.4:39290 -> 192.168.0.3:443 FIN *****F
Jul 21 13:10:22 192.168.0.4:39290 -> 192.168.0.3:9999 FIN *****F
```

Snort will identify all nmap scanning techniques, another interesting thing with snort is that it will also log what flags are set in each probe.

```
Jul 13 12:31:46 192.168.0.4:54468 -> 192.168.0.3:9 NULL *****
Jul 13 12:31:44 192.168.0.4:54469 -> 192.168.0.3:9 NMAPID **U*P*SF
Jul 13 12:31:44 192.168.0.4:54471 -> 192.168.0.3:1 SYN *****S*
Jul 13 12:31:44 192.168.0.4:54473 -> 192.168.0.3:1 XMAS **U*P**F
```

Remember Xmas scans set 3 flags URG, PUSH and FIN, and Null scans set no flags ?

* -sA Ack scanning, Firewall mapping

The Ack scan continually sends Ack Acknowledge packets, and is another useful firewall mapping scan, you can successfully scan a firewalled machine for open ports, with that information you can map the firewalls rule sets and get a understanding of what role this machine plays in the network. Acknowledge packets should get through most firewalls, because as far as non-state full firewall can tell a machine on it's network is requesting a outside connection to the internet because there are ack packets coming in acknowledging there so called syn-ack packets.

Nmap Tutorial

yudhax

```
Nmap run completed -- 1 IP address (1 host up) scanned in 7.426 seconds
[root@REDHATBOX root]# nmap -sA 192.168.0.1
```

```
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-22 10:51 EST
Interesting ports on 192.168.0.1:
(The 1641 ports scanned but not shown below are in state: filtered)
Port State Service
135/tcp UNfiltered loc-srv
1025/tcp UNfiltered NFS-or-IIIS
5000/tcp UNfiltered UPnP
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 72.043 seconds
```

* -sO IP scanning or internet protocol scanning.

No major information can be found using IP protocol scans, it just lists what protocols the host supports. But i would avoid this method unless you really need to determine what protocols are in use, most firewalls will not send back ICMP unreachable messages and you will get a massive list of useless protocols that you will not know if they are in use or not. Nmap sends raw ip packets and waits for a ICMP response to determine what protocols are in use.

```
[root@REDHATBOX root]# nmap -sO 192.168.0.4
```

```
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-21 09:26 EST
Interesting protocols on 192.168.0.4:
(The 251 protocols scanned but not shown below are in state: closed)
Protocol State Name
1 open icmp
2 open igmp
6 open tcp
17 open udp
255 open unknown
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 13.741 seconds
```

```
1 open icmp (Internet Control Message Protocol)
2 open igmp (Internet Group Management Protocol)
6 open tcp (Transmission Control Protocol)
17 open udp (User Datagram Protocol)
255 open unknown
```

Some machines will give false positives. AIX, HP-UX, Digital UNIX, and Windows machines. You could determine a windows machine with this scan if you had too, considering it will list every possible protocol.

-I Ident scanning

Ident scanning list the owners of each process thought a tcp connection. So it's logged and will be filterd by a decent firewall. This scan is usefull for obvious reasons. Exspecially for hackers, they can determine what services are running as root and target them. Not every service has to be root, so why waste youre time hacking a low level user. In order to get this information we must make a full tcp connection to the machine.

```
[root@REDHATBOX root]# nmap -sT -I 192.168.0.3
```

```
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-22 15:11 EST
Interesting ports on 192.168.0.3:
(The 1629 ports scanned but not shown below are in state: closed)
```

Nmap Tutorial

yudhax

```
Port State Service Owner
9/tcp open discard root
13/tcp open daytime root
21/tcp open ftp root
22/tcp open ssh root
25/tcp open smtp root
37/tcp open time root
80/tcp open http www-data
111/tcp open sunrpc daemon
113/tcp open auth identd
139/tcp open netbios-ssn root
443/tcp open https root
515/tcp open printer root
993/tcp open imaps root
995/tcp open pop3s root
9999/tcp open abyss root
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 9.202 seconds
```

As you can see there are a lot of root processes on this machine, and if we were to exploit any one of them we would have full access to the entire system. This scan will not work on windows.

-f Fragmentation scanning

This method exploits a simple flaw in TCP/IP networks, a fragmented packet is required to be reassembled by the receiving host. This allows us to bypass most packet filters and firewalls unless they queue fragmented packets. Fragmentation scanning can be used with the following scan types, Xmas, Syn, Fin and null. Because this method is not widely used it can cause sniffers and even firewalls to crash or seg fault. Nmap splits the TCP header into several fragments, like dicing up a pizza, the target host then must reassemble them, by using this technique we are trying to totally bypass firewall and fool IDS machines.

```
[root@REDHATBOX root]# nmap -f -sS 192.168.0.3
```

```
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-23 11:34 EST
```

```
Interesting ports on 192.168.0.3:
```

```
(The 1628 ports scanned but not shown below are in state: closed)
```

```
Port State Service
9/tcp open discard
13/tcp open daytime
21/tcp open ftp
22/tcp open ssh
25/tcp open smtp
37/tcp open time
80/tcp open http
111/tcp open sunrpc
113/tcp open auth
139/tcp open netbios-ssn
443/tcp open https
515/tcp open printer
993/tcp open imaps
995/tcp open pop3s
1241/tcp open msg
9999/tcp open abyss
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 4.343 seconds
```

So just add the -f for a fragmentation scan.

Nmap Tutorial

yudhax

The Fragmentation scan was a great success, it fooled snort into thinking it could have been bad traffic, it was not logged as a scan.

```
[**] [1:522:1] MISC Tiny Fragments [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
07/22-13:37:26.450093 192.168.0.4 -> 192.168.0.3
TCP TTL:225 TOS:0x0 ID:59880 IpLen:20 DgmLen:36 MF
Frag Offset: 0x0 Frag Size: 0x10
```

```
[**] [1:522:1] MISC Tiny Fragments [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
07/22-13:37:26.450097 192.168.0.4 -> 192.168.0.3
TCP TTL:225 TOS:0x0 ID:57511 IpLen:20 DgmLen:36 MF
Frag Offset: 0x0 Frag Size: 0x10
```

```
[**] [1:522:1] MISC Tiny Fragments [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
07/22-13:37:26.450101 192.168.0.4 -> 192.168.0.3
TCP TTL:225 TOS:0x0 ID:50069 IpLen:20 DgmLen:36 MF
Frag Offset: 0x0 Frag Size: 0x10
```

Fragscan only works with ACK, FIN, Maimon, NULL, SYN, Window, and XMAS scan types

-O OS fingerprinting

The -O flag tells nmap to try and calculate what operating system is running. This is one of most useful techniques nmap has to offer. Nmap uses TCP stack fingerprinting, it then compares it's results to a database from that database it can then determine what operating system the host is running. TCP fingerprinting is only one method to determine what OS a specific host is running but this lesson is on nmap so we will discuss TCP fingerprinting. The reason this is possible is every OS has a different implementation of the TCP stack, a windows machine will have different IPID's, sequence numbers and timing than a Linux or BSD machine. Because this involves analyzing the TCP stack you can also source other information from the TCP stack, for example how predictable it's IPID's are, weather or not it's incremental or positive increments, if you find a machine has a incremental stack it might be possible to guess it's responses and spoof a connection. Hackers use OS fingerprinting and database entire networks, and when a new exploit is released they just search through there logs for target that matches the exploit criteria.

```
[root@REDHATBOX root]# nmap -O -sS 192.168.0.1
```

```
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-24 09:16 EST
Warning: OS detection will be MUCH less reliable because we did not find at least
1 open and 1 closed TCP port
```

```
Interesting ports on 192.168.0.1:
```

```
(The 1639 ports scanned but not shown below are in state: filtered)
```

```
Port State Service
```

```
135/tcp open loc-srv
```

```
1002/tcp open unknown
```

```
1025/tcp open NFS-or-IIS
```

```
1720/tcp open H.323/Q.931
```

```
5000/tcp open UPnP
```

```
Device type: general purpose
```

```
Running: Microsoft Windows NT/2K/XP
```

```
OS details: Microsoft Windows XP Professional RC1+ through final release
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 87.142 seconds
```

Nmap Tutorial

yudhax

Nmap successfully identified the operating system as Windows XP Professional, this is valuable information to hackers and pen-testers. Now you can configure your pen- test to target a Windows system.

```
[root@REDHATBOX root]# nmap -O -sS 192.168.0.4
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-24 09:24 EST
Interesting ports on 192.168.0.4:
(The 1637 ports scanned but not shown below are in state: closed)
Port State Service
21/tcp open ftp
22/tcp open ssh
23/tcp open telnet
111/tcp open sunrpc
139/tcp open netbios-ssn
1024/tcp open kdm
6000/tcp open X11
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux Kernel 2.4.0 - 2.5.20
Uptime 16.176 days (since Tue Jul 8 05:11:43 2003)

Nmap run completed -- 1 IP address (1 host up) scanned in 16.112 seconds
```

In this test we get what kernel is running and this lets us know it is linux.

To get the TCP sequence prediction and IPID you need to include -v (verbose) flag you can even use -vv for even more output from nmap.

```
[root@REDHATBOX root]# nmap -v -O -sS 192.168.0.4
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-24 11:45 EST
Host 192.168.0.4 appears to be up ... good.
Initiating SYN Stealth Scan against 192.168.0.4 at 11:45
Adding open port 6000/tcp
Adding open port 139/tcp
Adding open port 21/tcp
Adding open port 1024/tcp
Adding open port 111/tcp
Adding open port 23/tcp
Adding open port 22/tcp
The SYN Stealth Scan took 2 seconds to scan 1644 ports.
For OSScan assuming that port 21 is open and port 1 is closed and neither are
firewalled
Interesting ports on 192.168.0.4:
(The 1637 ports scanned but not shown below are in state: closed)
Port State Service
21/tcp open ftp
22/tcp open ssh
23/tcp open telnet
111/tcp open sunrpc
139/tcp open netbios-ssn
1024/tcp open kdm
6000/tcp open X11
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux Kernel 2.4.0 - 2.5.20
Uptime 16.274 days (since Tue Jul 8 05:11:43 2003)
```

Nmap Tutorial

yudhax

```
TCP Sequence Prediction: Class=random positive increments
Difficulty=2621869 (Good luck!)
IPID Sequence Generation: All zeros
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 14.787 seconds
```

Here is the difference in the TCP stack, below is a Windows XP pro machine, it's difficulty is allot easier and is incremental, meaning in theory it is possible to try a TCP sequence attack.

```
[root@REDHATBOX root]# nmap -v -O -sS 192.168.0.1
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-24 11:50 EST
Host 192.168.0.1 appears to be up ... good.
Initiating SYN Stealth Scan against 192.168.0.1 at 11:50
Adding open port 135/tcp
Adding open port 5000/tcp
Adding open port 1002/tcp
Adding open port 1025/tcp
Adding open port 1720/tcp
adjust_timeout: packet supposedly had rtt of 9020013 microseconds. Ignoring time.
adjust_timeout: packet supposedly had rtt of 9021101 microseconds. Ignoring time.
The SYN Stealth Scan took 71 seconds to scan 1644 ports.
Warning: OS detection will be MUCH less reliable because we did not find at least
1 open and 1 closed TCP port
For OSScan assuming that port 135 is open and port 38216 is closed and neither are
firewalled
Interesting ports on 192.168.0.1:
(The 1639 ports scanned but not shown below are in state: filtered)
Port State Service
135/tcp open loc-srv
1002/tcp open unknown
1025/tcp open NFS-or-IIS
1720/tcp open H.323/Q.931
5000/tcp open UPnP
Device type: general purpose
Running: Microsoft Windows NT/2K/XP
OS details: Microsoft Windows XP Professional RC1+ through final release
TCP Sequence Prediction: Class=random positive increments
Difficulty=23841 (Worthy challenge)
IPID Sequence Generation: Incremental
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 85.411 seconds
```

-iR randomize hosts

I like this option, i think of it as trawling. Nmap just picks random hosts and scans them. It's not as direct as scanning subnets but you can discover some cool networks. And thats what nmap is all about, discovering networks.

```
Host 217.227.180.44 appears to be down, skipping it.
Host 183.221.113.67 appears to be down, skipping it.
Host 210.143.252.40 appears to be down, skipping it.
Host 49.233.209.60 appears to be down, skipping it.
Host 188.131.254.141 appears to be down, skipping it.ng it.
Host 190.52.101.229 appears to be down, skipping it.
```

If you look at the above address you can see how nmap just randomly picks hosts and scans them. If you wanted you could do this forever on youre networks. Everytime the connection drops out just do --resume (logfile). And run it as a background process.

Nmap Tutorial

yudhax

There is another way to randomize nmap scans with the `--randomize_hosts`

-iL read targets from file.

If you have your networks or WLAN mapped out into a file, give nmap the `-iL` flag and it will read host names from that file and begin scanning them.

-F Fast scan

Only scans for known services, instead of scanning all 65535 it only scans 1190 and decreases scan times greatly.

-p Specify Ports

Even better if you are scanning for rogue web servers or something you only have to scan for ports 80,443. You can specify the ports like so.

```
[root@REDHATBOX root]# nmap -p 22,21 -sT 192.168.0.4

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-24 12:09 EST
Interesting ports on 192.168.0.4:
Port State Service
21/tcp open ftp
22/tcp open ssh

Nmap run completed -- 1 IP address (1 host up) scanned in 10.252 seconds
```

Here we just scanned for ssh and ftp. For more options you can try 21,22,80-200

```
[root@REDHATBOX root]# nmap -p 21,22,80-200 -sT 192.168.0.4
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-24 12:11 EST
Interesting ports on 192.168.0.4:
(The 119 ports scanned but not shown below are in state: closed)
Port State Service
21/tcp open ftp
22/tcp open ssh
111/tcp open sunrpc
139/tcp open netbios-ssn

Nmap run completed -- 1 IP address (1 host up) scanned in 8.128 seconds
```

So we scanned for ftp,ssh and every open port from 80-200.

Nmap scan logs.

You can log nmap scans in 4 different formats.

```
-oN logs nmap scans in human readable format.
-oX logs nmap scans in XML format.
-oG logs nmap scans in grepable format
-oS logs nmap scans in script kiddie format.
-oA logs nmap scans in all of the above formats except script kiddie.
```

here is the script kiddie log format. Not recommended.

```
[root@REDHATBOX test]# cat kiddie

Starting nmap 3.30 ( Http://www.n3cure.org/nmap/ ) At 2003-07-24 13:39 e$T
```

Nmap Tutorial

yudhax

```
!nt3R3$t|ng pOrTz On 192.168.0.4:
(TH3 1637 Portz $canN3d but nOt $hOwn b3l0w ArE iN $TAt3: c10s3d)
P0rT $tate s3rVICe
21/tcp Op3n ftp
22/tcp oP3N $$h
23/Tcp Open t3lnEt
111/tcp Op3n sunrpc
139/tcp OpEn n3tbioz-$$n
1024/tCp opEn kdm
6000/tcP op3n X11
```

```
nmap run c0mpl3ted -- 1 |P adDr3sz (1 h0St up) scannEd |n 8.570 $3conDs
```

this is my preferred logging method with nmap.

```
[root@REDHATBOX test]# nmap -oA test -sT 192.168.0.4;ls
```

```
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-07-24 14:43 EST
Interesting ports on 192.168.0.4:
(The 1637 ports scanned but not shown below are in state: closed)
Port State Service
21/tcp open ftp
22/tcp open ssh
23/tcp open telnet
111/tcp open sunrpc
139/tcp open netbios-ssn
1024/tcp open kdm
6000/tcp open X11
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 7.713 seconds
kiddie test.gnmap test.nmap test.xml
```

resume your scan by specifying the following flag in nmap --resume logfile.

Nmapfe Nmap front end.

The nmap front end is a gui version of the nmap scanner for *nix variants.

Nmap for windows

We have focused the major part of this lesson on nmap for linux, but there is a windows version that is capable of the same functionality as the linux version. The windows version is allot slower than the linux version but apart from that it is very capable.