Detecting Live Systems

Active Reconnaissance

Overview

- Overview of port scanning
- Port scan types: open scan, half-open scan, UDP scan.
- Port scanning tools
- OS fingerprinting
- TCP fragmentation scanning
- Countermeasures

Introduction to Port Scanning

Scanning is a method for discovering exploitable communication channels.

Port Scanning is how attackers identify open and available TCP/IP ports, services and applications on a system.

Applications and services on a system are associated with well known port numbers.

For example port 80 is HTTP, port 23 is Telnet, and port 25 is SMTP

Port Scan Tips

Understand the three-way handshake. Generally, as long as the three-way handshake has not been completed, the law has not been broken.

A port scan is like checking to see if the door is unlocked but not entering to see whether someone's at home. No crime has been committed yet, so in most cases the police can't do anything at this point.

If a computer system is attacked many times by a port scan, one can argue that the port scan was, in fact, a denial-of-service (DoS) attack, which is usually an offense.

Be Careful!

Laws differ in states and countries; check with your legal department!

Expected Results

Is the system active and responsive? What ports are open or filtered?

What services are running and what information can be gleaned

Services & versions	Type of OS running and patch level	Network Blueprint

Popular Port Scanning Tools

Tool	Platforms	Website
Look@LAN	Windows	www.lookatlan.com
SuperScan	Windows	www.foundstone.com
Unicornscan	Unix	www.unicornscan.org
NMAP	Windows and Unix	www.insecure.org
AutoScan	Unix	www.icewalkers.com/Linux/Software/521810/AutoScan.html
Hping2	Unix	www.hping.org

Stealth Online Ping

CentralOps.net Advanced online Internet utilities

Ping See if a host is reachable

domain or IP address: www.mile2.com

packets to send: 5 timeout (ms): 1000

data size (bytes): 32 ttl (hops): 255

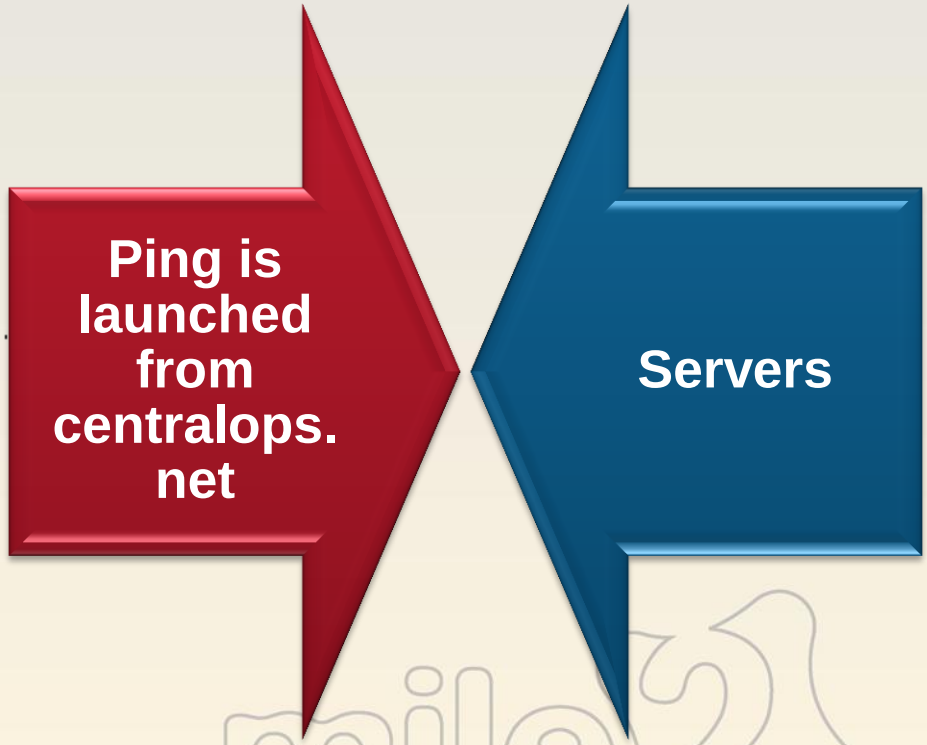
☐ don't fragment go

source code: view | download CentralOps.net

Pinging **www.mile2.com [209.59.165.80]** with 32 bytes of data...

Results

count	ttl (hops)	rtt (ms)	from
1	53	40	209.59.165.80
2	53	40	209.59.165.80
3	53	40	209.59.165.80
4	53	40	209.59.165.80
5	53	40	209.59.165.80



NMAP: Is the Host online

Shell - Konsole

Shell | Shell No. 2

```
bt ~ # nmap -sP 194.111.80.1

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2007-02-08 14:19
 GMT
Host homerun7.wlan.inet.fi (194.111.80.1) appears to be up.
MAC Address: 00:A0:8E:1C:A3:24 (Nokia Internet Communications)
Nmap finished: 1 IP address (1 host up) scanned in 0.468 seconds
bt ~ #
```

```
-sP (Ping Scan)
      This option tells Nmap to only perform a ping scan (host discovery), then
      print out the available hosts that responded to the scan. No further testing
      (such as port scanning or OS detection) is performed. This is one step more
      intrusive than the list scan, and can often be used for the same purposes. It
      allows light reconnaissance of a target network without attracting much
      attention. Knowing how many hosts are up is more valuable to attackers than
      the list provided by list scan of every single IP and host name.
```

Nmap Reference Guide: http://nmap.org/book/man.html

ICMP Disabled?

Most of the time you will need to disable ping on NMAP in order to test the system!

-PN and -P0 – Disables ping test (i.e. host discovery)

```
bt ~ # nmap -PN 192.168.1.135

Starting Nmap 4.60 ( http://nmap.org ) at 2009-06-18 22:30 GMT
Interesting ports on 192.168.1.135:
Not shown: 1711 closed ports
PORT     STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
990/tcp open  ftps
MAC Address: 00:0D:61:0D:A5:BE (Giga-Byte Technology Co.)

Nmap done: 1 IP address (1 host up) scanned in 2.531 seconds
```

NMAP TCP Connect Scan

Zenmap

Scan Tools Profile Help

Target: Profile: Scan

Command: nmap -sT -v 192.168.1.135

Hosts | Nmap Output | Ports / Hosts | Topology | Host Details | Scans

OS | Host

192.168.1.135

nmap -sT -v 192.168.1.135 Details

```
Discovered open port 135/tcp on 192.168.1.135
Increasing send delay for 192.168.1.135 from 5 to 10 du
max_successful_tryno increase to 5
Completed Connect Scan at 14:38, 296.87s elapsed (1000
ports)
Host 192.168.1.135 appears to be up ... good.
Interesting ports on 192.168.1.135:
Not shown: 996 closed ports
PORT     STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
990/tcp open  ftps
MAC Address: 00:0D:61:0D:A5:BE (Giga-Byte Technology Co

Read data files from: C:\Program Files\Nmap
Nmap done: 1 IP address (1 host up) scanned in 305.33 s
           Raw packets sent: 1 (42B) | Rcvd: 1 (42B)
```

-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.

Zenmap

Nmap ("Network Mapper") is an open source utility for network exploration or security auditing.

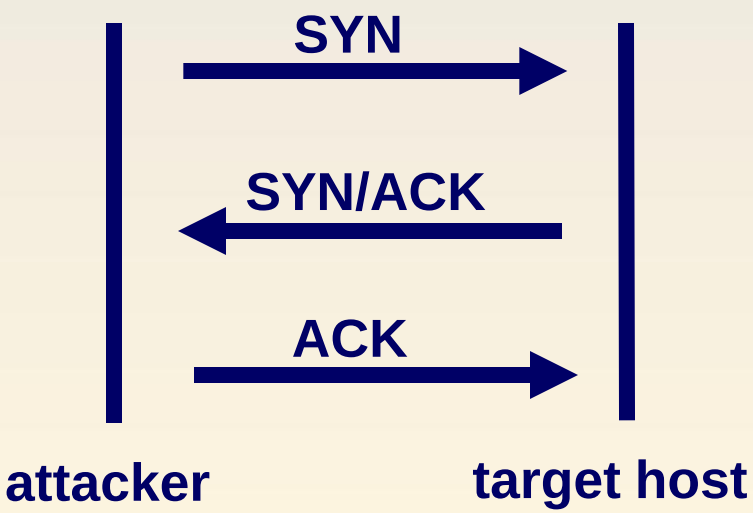
It is designed to scan a large network very quickly. It can be used for port scanning, OS detection, ping sweeps, and version detection just to name a few.

TCP Connect Port Scan

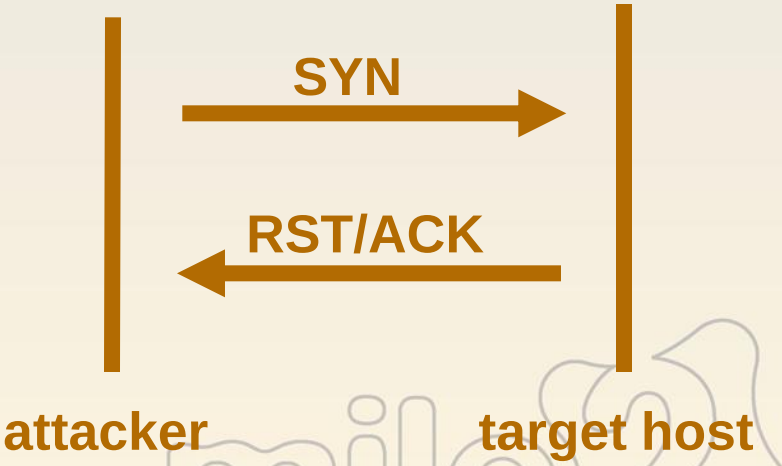
With a TCP Connect port scan, the attacker sends SYN packets to sequential port numbers on a target to see which port numbers reply. A connection is attempted to port 1, then port 2, then port 3, etc.

An open port will reply with a SYN/ACK, a closed port will reply with a RST/ACK, or no reply if filtered.

Open port response



Closed port response



Nmap (cont.)

Shell - Konsole

Shell

```
root@slax:~# nmap -v -P0 -sS 192.168.1.190 -p 135-139,443

Starting Nmap 4.00 ( http://www.insecure.org/nmap/ ) at 2006-05-25 03:33 GMT
DNS resolution of 1 IPs took 0.08s. Mode: Async [#: 1, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
Initiating SYN Stealth Scan against 192.168.1.190 [6 ports] at 03:33
The SYN Stealth Scan took 6.04s to scan 6 total ports.
Host 192.168.1.190 appears to be up ... good.
Interesting ports on 192.168.1.190:
PORT     STATE    SERVICE
135/tcp filtered msrpc
136/tcp filtered profile
137/tcp filtered netbios-ns
138/tcp filtered netbios-dgm
139/tcp filtered netbios-ssn
443/tcp filtered https

Nmap finished: 1 IP address (1 host up) scanned in 6.331 seconds
               Raw packets sent: 15 (600B) | Rcvd: 6 (408B)
root@slax:~#
```

This example shows verbose (-v) output from nmap, doing a half-open scan (-sS) of ports 135-139 & 443.

Nmap by default pings the target host before doing a port scan; you can disable this by using –P0 or -PN.

Notice that the ports are “filtered” ports, which means that a firewall is blocking the responses.

Tool Practice : TCP half-open & Ping Scan

```
-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 scan-
       ning technique is that fewer sites will log it. Unfortunately you need root
       privileges to build these custom SYN packets. This is the default scan type
       for privileged users.
```

Practice: nmap –sS <IP address>

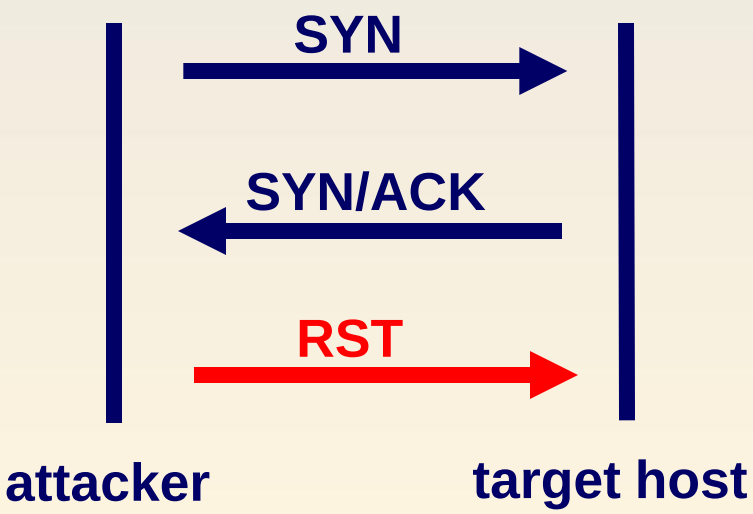
Half-open Scan

A half-open TCP SYN port scan is the same as the vanilla TCP open scan, however the attacker does not complete the 3-way handshake.

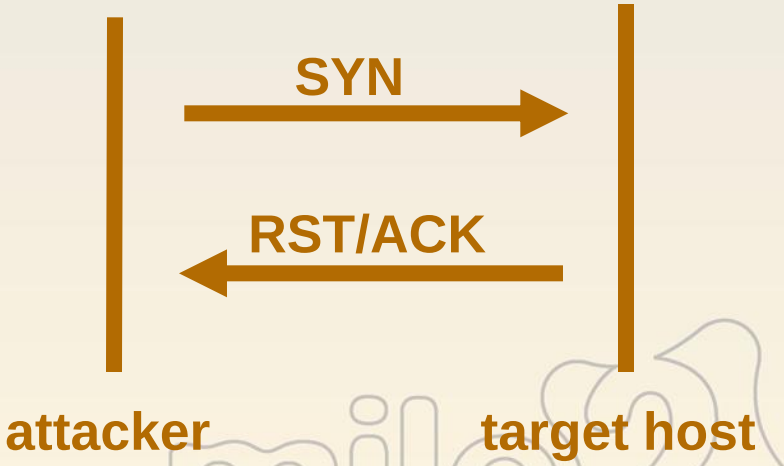
An open port will still reply with a SYN/ACK, a closed port will reply with a RST/ACK.

Advantage over TCP Connect scan: may not be detected by simple IDS and no law has been broken at this time.

Open port response



Closed port response



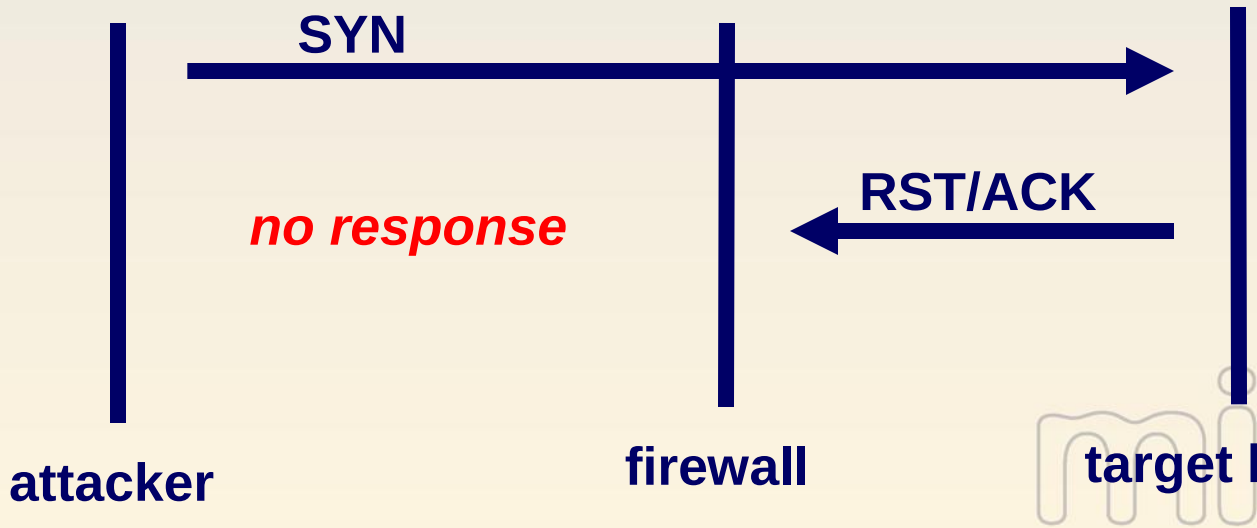
Firewalled Ports

A TCP Connect or half-open scan should receive either a SYN/ACK or a RST/ACK packet.

However, a third possibility exists: No response

This is often due to a firewalled port being filtered, or possibly the packets being lost due to network congestion.

Firewalled port response



NMAP Service Version Detection

```
-sV    Version detection: After TCP and/or UDP ports are discovered using one of  the
       other scan methods, version detection communicates with those ports to try and
       determine more about what is actually running.  A  file  called  nmap-service-
       probes is used to determine the best probes for detecting various services and
       the match strings to expect.  Nmap tries to  determine  the  service  protocol
       (e.g.  ftp,  ssh,  telnet,  http), the application name (e.g. ISC Bind, Apache
       httpd, Solaris telnetd),  the  version  number,  and  sometimes  miscellaneous
       details  like  whether  an X server is open to connections or the SSH protocol
```

```
bt ~ # nmap -sV -p 80 209.59.165.80

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2007-02-08 21:55 GMT
Warning: Servicescan failed to fill info_template (subjectlen: 1460). Too long?
ne 2683: v/Apache httpd/$1/$2
Interesting ports on 209.59.165.80:
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 1.3.37

Nmap finished: 1 IP address (1 host up) scanned in 9.094 seconds
```

Additional NMAP Scans

-O – OS Detection

-A – OS and Service Version Detection

-T – Timing (Built in Timing Templates)

- Paranoid (0)
- Sneaky (1)
- Polite (2)
- Normal (3)
- Aggressive (4)
- Insane (5)

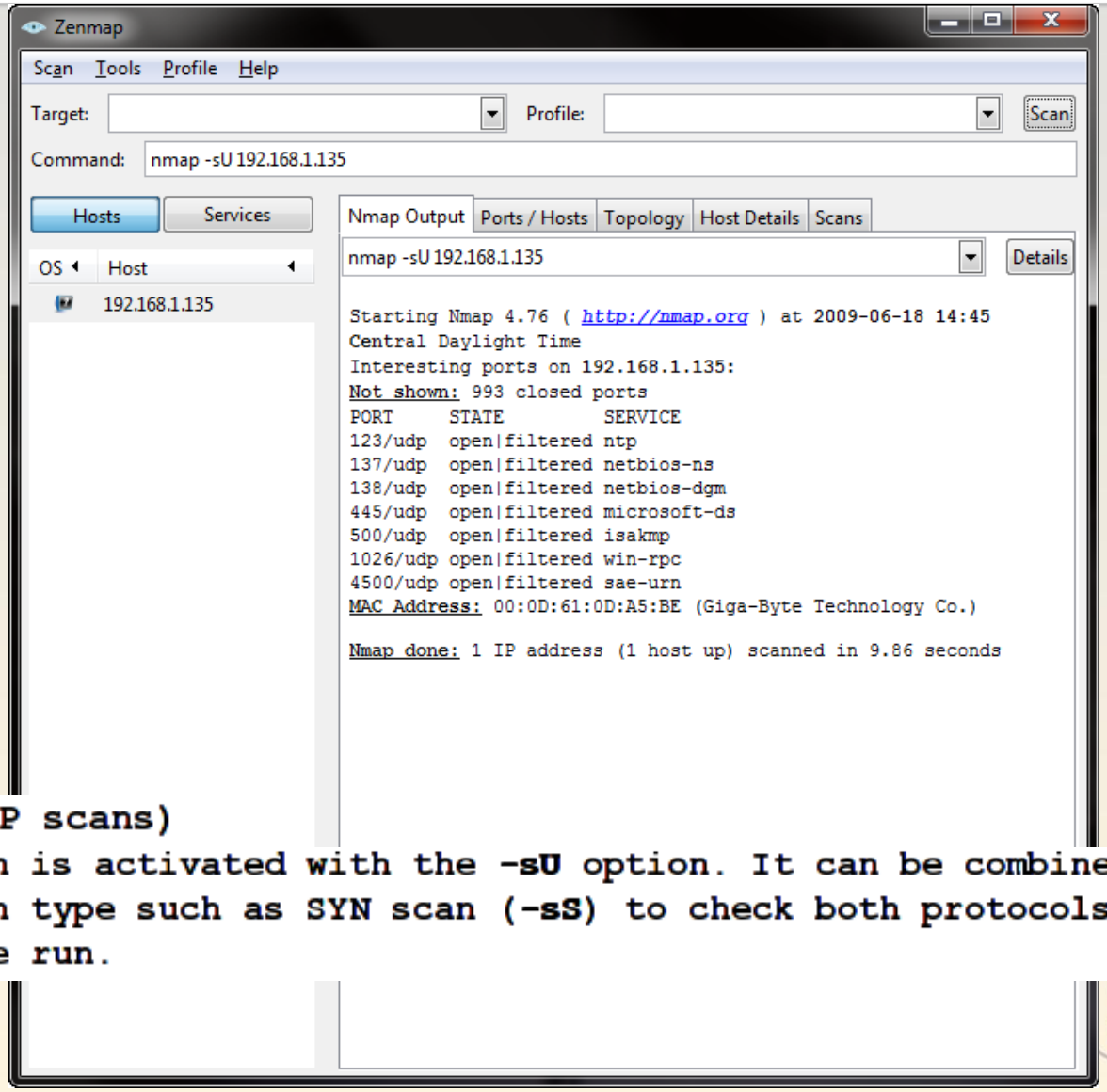
Saving NMAP results

NMAP Output Formats

```
OUTPUT:
  -oN/-oX/-oS/-oG <file>: Output scan in normal, XML, s|<rIpt kIddi3,
     and Grepable format, respectively, to the given filename.
  -oA <basename>: Output in the three major formats at once
```

```
# nmap -v -oG - -sO localhost
# Nmap 4.68 scan initiated [time] as: nmap -v -oG - -sO localhost
# Ports scanned: TCP(0;) UDP(0;) PROTOCOLS(256;0-255)
Host: 127.0.0.1 (localhost)
      Protocols: 1/open/icmp/, 2/open|filtered/igmp/, 6/open/tcp/,
                 17/open/udp/, 136/open|filtered/udplite/, 255/open|filtered//
      Ignored State: closed (250)
# Nmap done at [time] -- 1 IP address (1 host up) scanned in 2.345 seconds
```

NMAP UDP Scans



-sU (UDP scans)

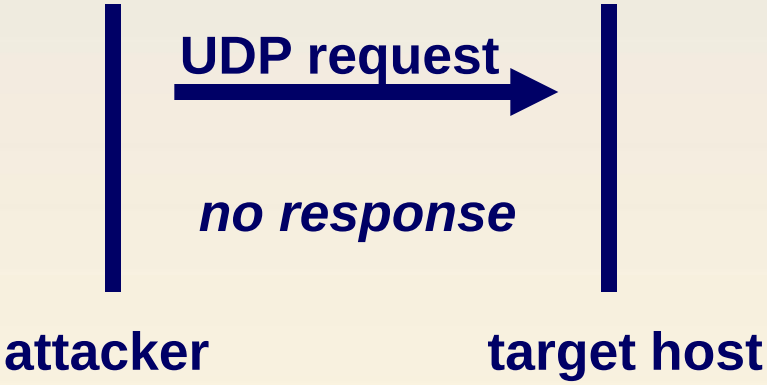
UDP scan is activated with the **-sU** option. It can be combined with a TCP scan type such as SYN scan (**-sS**) to check both protocols during the same run.

UDP Port Scan

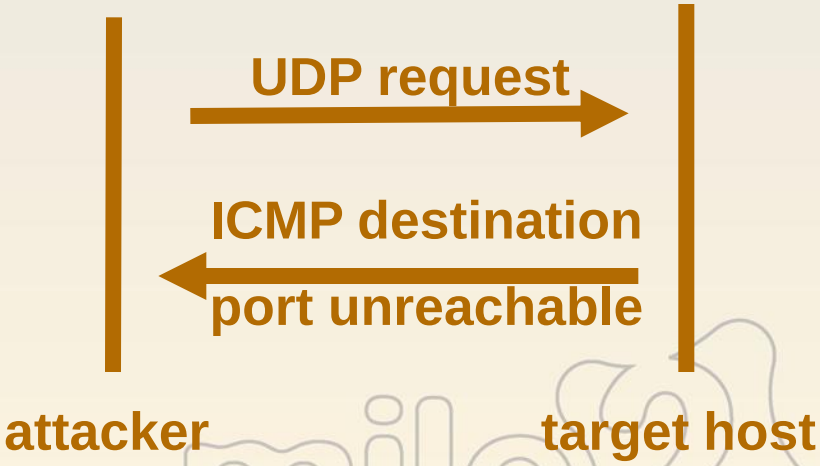
Open UDP ports can be identified with port scans, even though UDP is connectionless.

Sending a UDP request to a particular port will result in no response for an open port, ICMP port unreachable for a closed port.

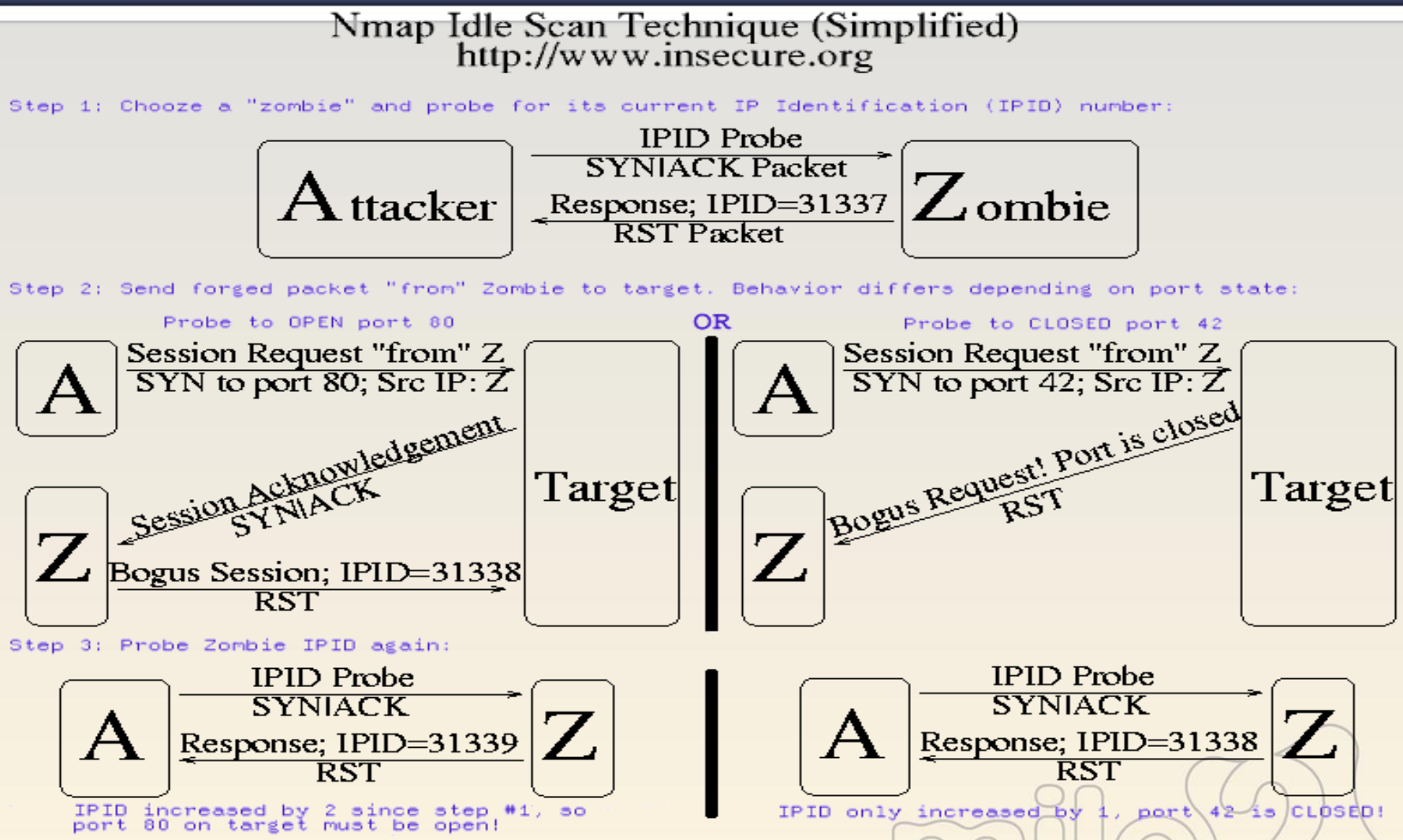
Open port response



Closed port response

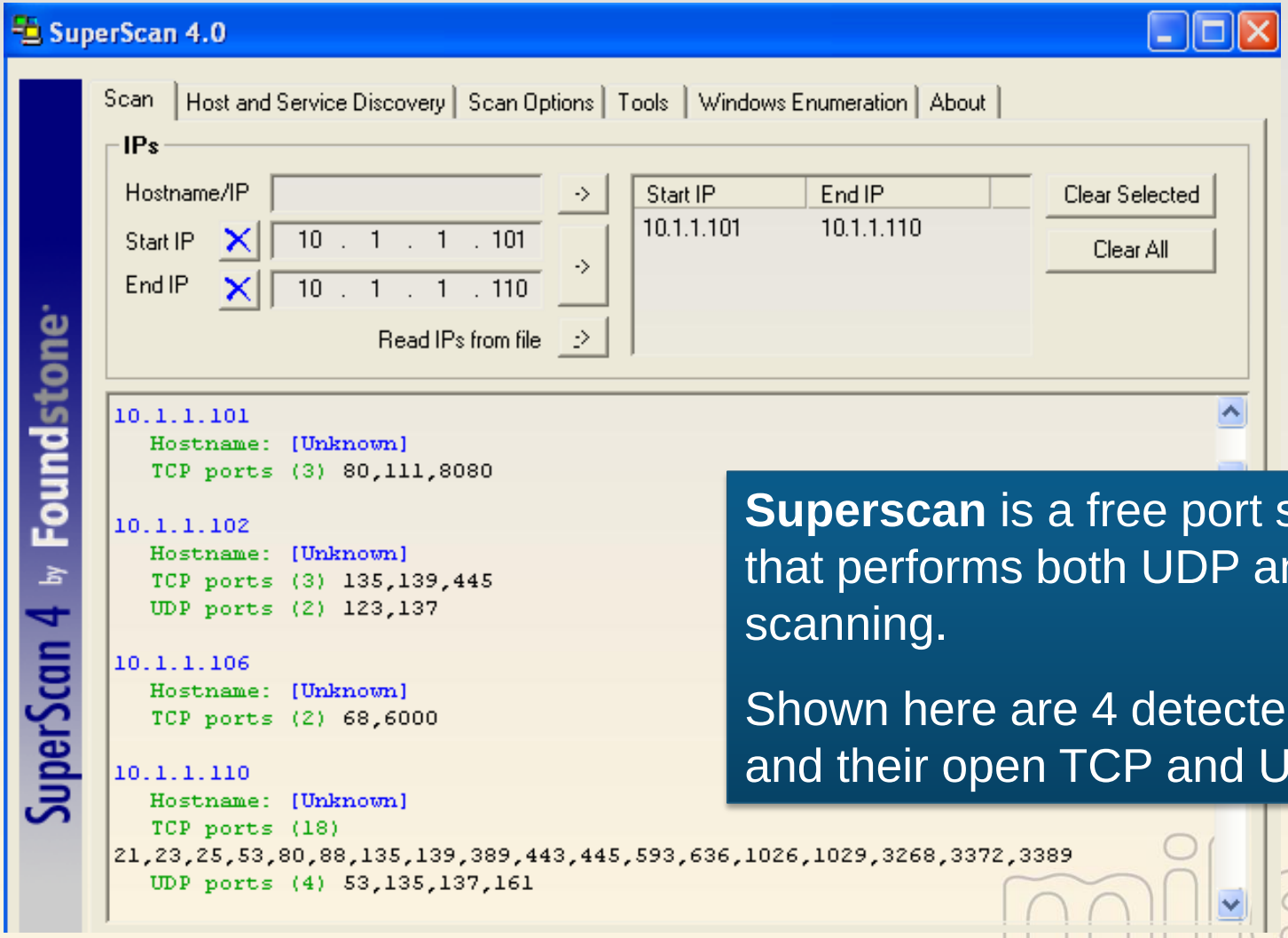


Advanced Technique



http://insecure.org/nmap/idlescan.html

Tool: Superscan



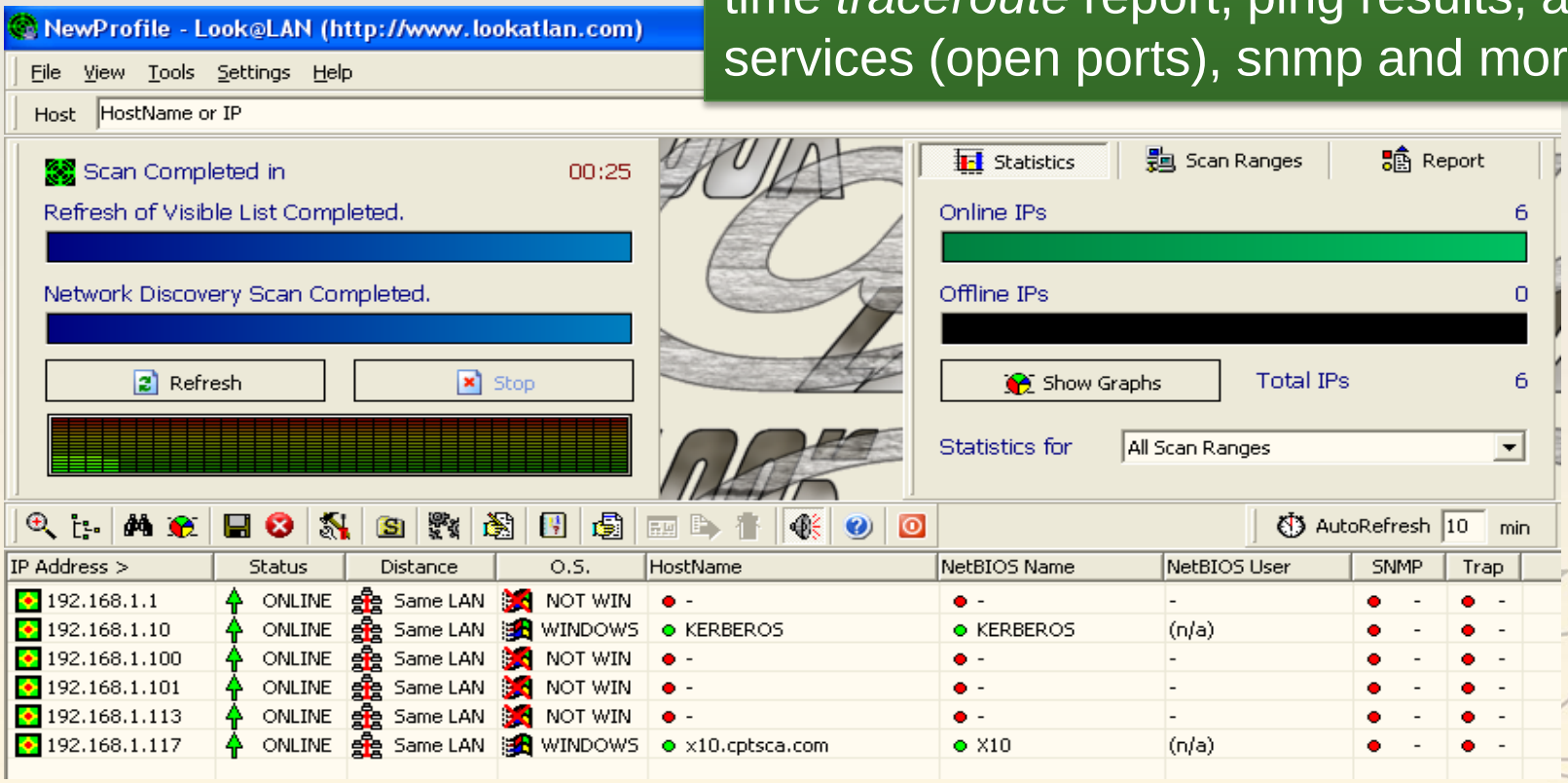
Superscan is a free port scanner that performs both UDP and TCP port scanning.

Shown here are 4 detected hosts, and their open TCP and UDP ports.

Tool: Look@LAN

Look@LAN is a free port scanner

Lists all available nodes and detailed statistics and scan results are available for each individual machine, including a real-time *traceroute* report, ping results, active services (open ports), snmp and more.



UNICORNSCAN

Unicorns are fast! :)

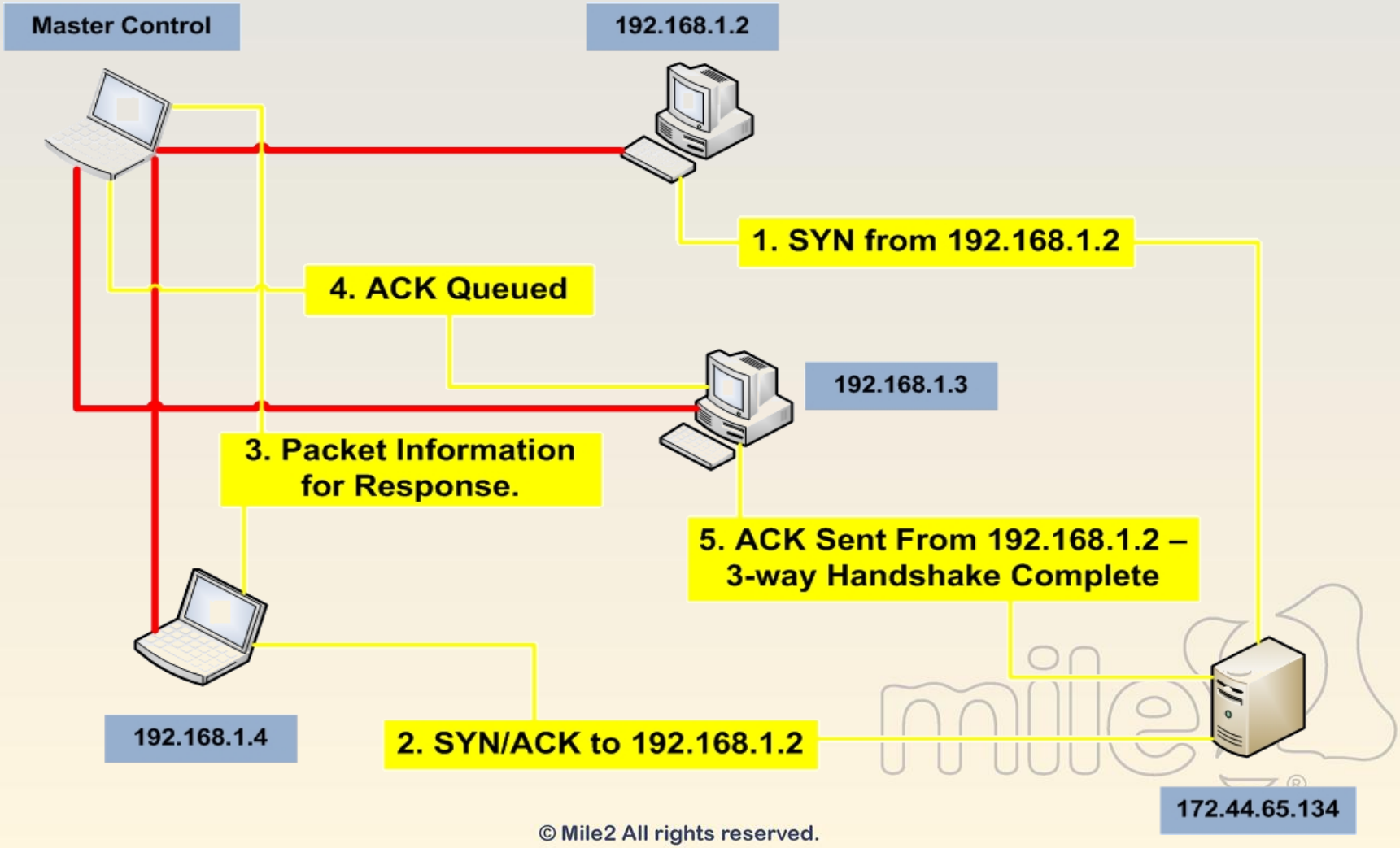
A new method for TCP state tracking (Scatter Connect)

Unicornscan is a distributed Stimulus/Response framework (not a port scanner)

Although it currently has hundreds of individual features, a main set of abilities include:

- Asynchronous stateless TCP scanning with all variations of TCP Flags.
- Asynchronous stateless TCP banner grabbing
- Asynchronous protocol specific UDP Scanning (sending enough of a signature to elicit a response).
- Active and Passive remote OS, application, and component identification by analyzing responses.
- PCAP file logging and filtering
- Relational database output
- Custom module support
- Customized data-set views

Scatter Connect



Tool: Hping2

A subset of the things you can do using hping:

- Firewall testing
- Advanced port scanning
- Network testing, using different protocols, TOS, fragmentation
- Manual path MTU discovery
- Advanced traceroute, under all the supported protocols
- Remote OS finger-printing
- Remote uptime guessing
- TCP/IP stacks auditing

hping

http://www.hping.org/

Tool: Hping2

Shell - Konsole

Shell

```
root@slax:~# hping2 -S -p 80 -c 1 192.168.6.190
HPING 192.168.6.190 (eth0 192.168.6.190): S set, 40 headers + 0 data bytes
len=46 ip=192.168.6.190 ttl=128 DF id=2142 sport=80 flags=SA seq=0 win=16616 rtt=5.3 ms

--- 192.168.6.190 hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 5.3/5.3/5.3 ms
root@slax:~# hping2 -S -p 81 -c 1 192.168.6.190
HPING 192.168.6.190 (eth0 192.168.6.190): S set, 40 headers + 0 data bytes
len=46 ip=192.168.6.190 ttl=128 id=2150 sport=81 flags=RA seq=0 win=0 rtt=3.0 ms

--- 192.168.6.190 hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 3.0/3.0/3.0 ms
root@slax:~#
```

Hping2 is a UNIX tool that can send arbitrary packets to a host in order to see how the host will respond.

Shown here is a SYN packet sent to host 192.168.6.190 on port 80, and the response back is “SA” flags, for SYN/ACK (port is open). A SYN packet sent to port 81 results in “RA” flags, for Reset/ACK (port is closed).

More Hping2

Shell - Konsole <2>

Shell

```
root@slax:~# hping2 --scan 1-30,70-90,135-139 -S 192.168.1.10
Scanning 192.168.1.10 (192.168.1.10), port 1-30,70-90,135-139
56 ports to scan, use -V to see all the replies
+----+-----------+---------+---+-----+-----+
|port| serv name |  flags  |ttl| id  | win |
+----+-----------+---------+---+-----+-----+
   25 smtp        : .S..A... 128 21540 16384
   80 http        : .S..A... 128 25636 16384
   88 kerberos-se: .S..A... 128 27684 16384
  135 loc-srv     : .S..A... 128 28452 16384
  139 netbios-ssn: .S..A... 128 29476 16384
All replies received. Done.
```

```
--- 192.168.1.10 hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 1.0/1.5/2.0 ms
root@slax:~# hping2 -S -p 80 -c 2 192.168.1.10
HPING 192.168.1.10 (eth0 192.168.1.10): S set, 40 headers + 0 data bytes
len=46 ip=192.168.1.10 ttl=128 id=15703 sport=80 flags=SA seq=0 win=16384 rtt=1.0 ms
len=46 ip=192.168.1.10 ttl=128 id=15704 sport=80 flags=SA seq=1 win=16384 rtt=2.0 ms
```

Hping2 --scan 1-30,70-90,135-139 –S <destinationIp>

Hping2 –help (to view other more advanced syntax usage)

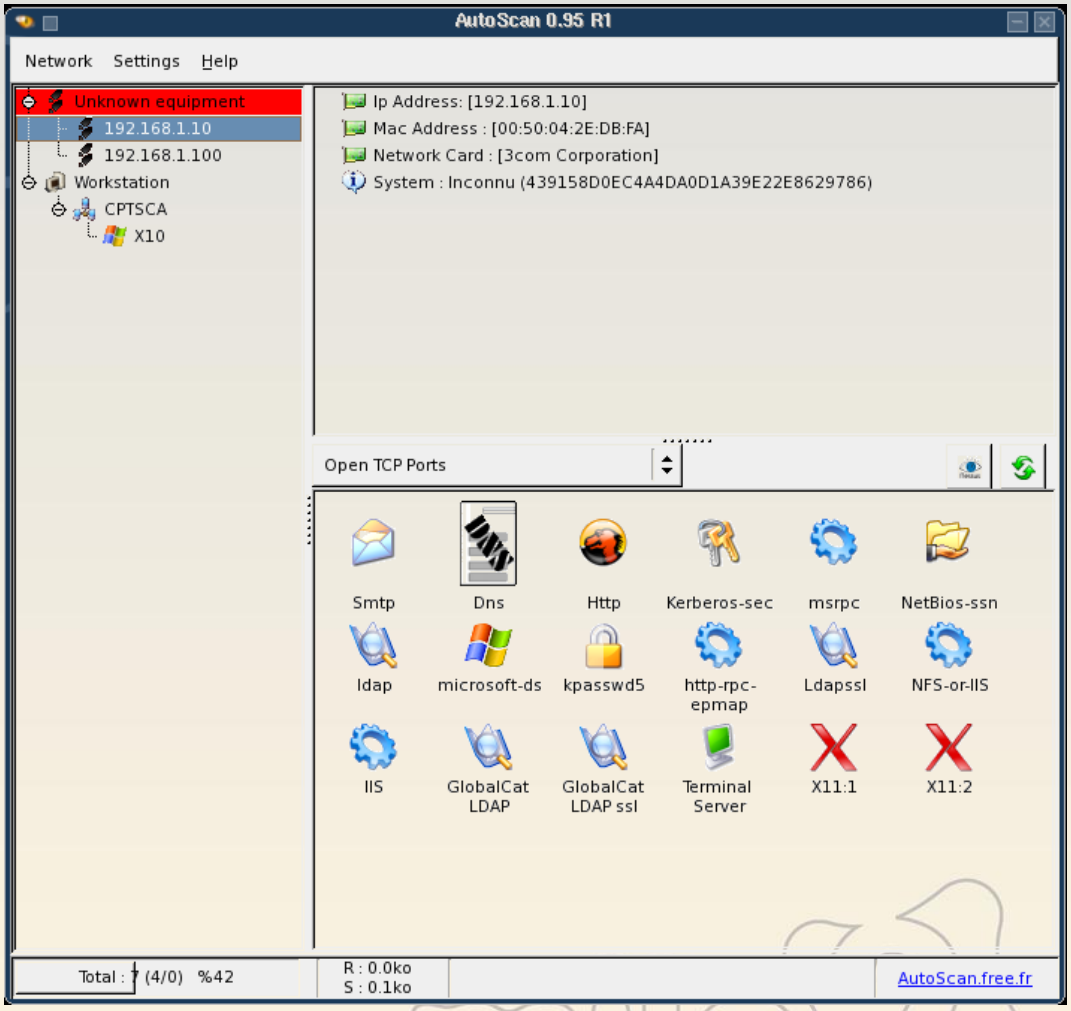
Now add: -V after –S as it will list details verbosely

Tool: Auto Scan

AutoScan is an application designed for exploring and managing your network.

Entire subnets can be scanned simultaneously without intervention from you.

It features OS detection, automatic network discovery, a port scanner, a Nessus client, a Samba share browser, and the ability to save the network state.



OS Fingerprinting: Xprobe2

```
bt ~ # xprobe2 -v -T21-500 209.59.165.80

Xprobe2 v.0.3 Copyright (c) 2002-2005 fyodor@o0o.nu, ofir@sys-security.com,

[+] Target is 209.59.165.80
[+] Loading modules.
[+] Following modules are loaded:
[x] [1] ping:icmp_ping  -  ICMP echo discovery module
[x] [2] ping:tcp_ping  -  TCP-based ping discovery module
[x] [3] ping:udp_ping  -  UDP-based ping discovery module
```

```
[+] Primary guess:
[+] Host 209.59.165.80 Running OS: "Linux Kernel 2.0.30" (Guess probability: 95%)
[+] Other guesses:
[+] Host 209.59.165.80 Running OS: "Foundry Networks IronWare Version 03.0.01eTc1" (Guess probabili
ty: 95%)
[+] Host 209.59.165.80 Running OS: "Linux Kernel 2.6.11" (Guess probability: 95%)
```

Xprobe2 is an OS fingerprinting tool that runs on Unix systems.

Xprobe2 relies on fuzzy signature matching, probabilistic guesses, multiple matches simultaneously, and a signature database.

Xprobe2 Options

- **Xprobe2 –v –T21-500 192.168.XXX.XXX**

```
bt ~ # xprobe2 --help

Xprobe2 v.0.3 Copyright (c) 2002-2005 fyodor@o0o.nu, ofir@sys-security.com, meder@o0o.nu

xprobe2: invalid option -- -
usage: xprobe2 [options] target
Options:
          -v                        Be verbose
          -r                        Show route to target(traceroute)
          -p <proto:portnum:state> Specify portnumber, protocol and state.
                                    Example: tcp:23:open, UDP:53:CLOSED
          -c <configfile>           Specify config file to use.
          -h                        Print this help.
          -o <fname>                Use logfile to log everything.
          -t <time_sec>             Set initial receive timeout or roundtrip time.
          -s <send_delay>           Set packsending delay (milseconds).
          -d <debuglv>              Specify debugging level.
          -D <modnum>               Disable module number <modnum>.
          -M <modnum>               Enable module number <modnum>.
          -L                        Display modules.
          -m <numofmatches>         Specify number of matches to print.
          -T <portspec>             Enable TCP portscan for specified port(s).
                                    Example: -T21-23,53,110
          -U <portspec>             Enable UDP portscan for specified port(s).
          -f                        force fixed round-trip time (-t opt).
          -F                        Generate signature (use -o to save to a file).
          -X                        Generate XML output and save it to logfile specified with -o.
          -B                        Options forces TCP handshake module to try to guess open TCP port
          -A                        Perform analysis of sample packets gathered during portscan in
                                    order to detect suspicious traffic (i.e. transparent proxies,
                                    firewalls/NIDSs resetting connections). Use with -T.
```

Tool: P0f

Passive OS Finger Printing Utility

```
bt ~ # p0f
p0f - passive os fingerprinting utility, version 2.0.8
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'eth0', 262 sigs (14 generic, cksum 0F1F5CA2), rule: 'all'.
194.111.80.217:2181 - UNKNOWN [S4:64:1:60:M1460,S,T,N,W3:.:?:?] (up: 18 hrs)
  -> 89.149.202.26:80 (link: ethernet/modem)
194.111.80.217:2182 - UNKNOWN [S4:64:1:60:M1460,S,T,N,W3:.:?:?] (up: 18 hrs)
  -> 89.149.202.26:80 (link: ethernet/modem)
194.111.80.217:4763 - UNKNOWN [S4:64:1:60:M1460,S,T,N,W3:.:?:?] (up: 18 hrs)
  -> 213.206.121.69:80 (link: ethernet/modem)
```

Tool Practice: Amap

AMAP – Application Mapper Identifies Services using banner grabbing and also simulates handshaking services:

Shell - Konsole

Shell | Shell No. 2

```
bt ~ # amap -A www.mile2.com 80
amap v5.2 (www.thc.org/thc-amap) started at 2007-02-08 21:47:51 - MAPPING mode

Protocol on 209.59.165.80:80/tcp matches http
Protocol on 209.59.165.80:80/tcp matches http-apache-1
Protocol on 209.59.165.80:80/tcp matches http-apache-2
Protocol on 209.59.165.80:80/tcp matches webmin

Unidentified ports: none.

amap v5.2 finished at 2007-02-08 21:47:57
bt ~ #
```

Tool: Fragrouter: Fragmenting Probe Packets

TCP fragmentation scanning involves fragmenting the TCP probe packets into thousands of pieces before sending them to the target host.

By sending fragmented traffic, simple non-stateful firewalls will allow the fragmented packets to pass through it.

When the fragments reach their target host, they are reassembled and processed. Replies will be sent to the probe packets, thus allowing the scanner to identify open and closed TCP ports.

The tool fragrouter can be used to send these fragmented packets to a host.

```
root@slax:~# fragrouter
Version 1.6
Usage: fragrouter [-i interface] [-p] [-g hop] [-G hopcount] ATTACK

 where ATTACK is one of the following:

 -B1: base-1: normal IP forwarding
 -F1: frag-1: ordered 8-byte IP fragments
 -F2: frag-2: ordered 24-byte IP fragments
 -F3: frag-3: ordered 8-byte IP fragments, one out of order
 -F4: frag-4: ordered 8-byte IP fragments, one duplicate
 -F5: frag-5: out of order 8-byte fragments, one duplicate
 -F6: frag-6: ordered 8-byte fragments, marked last frag first
 -F7: frag-7: ordered 16-byte fragments, fwd-overwriting
 -T1: tcp-1:  3-whs, bad TCP checksum FIN/RST, ordered 1-byte segments
 -T3: tcp-3:  3-whs, ordered 1-byte segments, one duplicate
 -T4: tcp-4:  3-whs, ordered 1-byte segments, one overwriting
 -T5: tcp-5:  3-whs, ordered 2-byte segments, fwd-overwriting
 -T7: tcp-7:  3-whs, ordered 1-byte segments, interleaved null segments
 -T8: tcp-8:  3-whs, ordered 1-byte segments, one out of order
 -T9: tcp-9:  3-whs, out of order 1-byte segments
 -C2: tcbc-2: 3-whs, ordered 1-byte segments, interleaved SYNs
 -C3: tcbc-3: ordered 1-byte null segments, 3-whs, ordered 1-byte segments
 -R1: tcbt-1: 3-whs, RST, 3-whs, ordered 1-byte segments
 -I2: ins-2:  3-whs, ordered 1-byte segments, bad TCP checksums
 -I3: ins-3:  3-whs, ordered 1-byte segments, no ACK set
 -M1: misc-1: Windows NT 4 SP2 - http://www.dataprotect.com/ntfrag/
 -M2: misc-2: Linux IP chains - http://www.dataprotect.com/ipchains/
```

Countermeasures: Scanning

- Disable all ICMP both inbound and outbound at the firewall.
- Configure the firewall to drop all invalid packets and anomalies .
- Enable application-layer monitoring of data at the firewall or IDS.
- Use an intrusion detection system to detect port scans and then terminate that connection.
- Use an advanced firewall that obfuscates port scans by indicating that all ports are open.
- Use software (e.g. BSDFPF) that emulates a different TCP/IP stack in order to give an attacker a false OS fingerprint.
- Use XP SP2 and Vista since it limits the number of simultaneous open sockets and hence drastically slows down a port scan.

Review

- Port scanning intro
- Port scan types: open scan, half-open scan, UDP scan
- Port scanning tools
- OS fingerprinting
- TCP fragmentation scanning
- Countermeasures

Module 4 Lab Scanning Active Recon