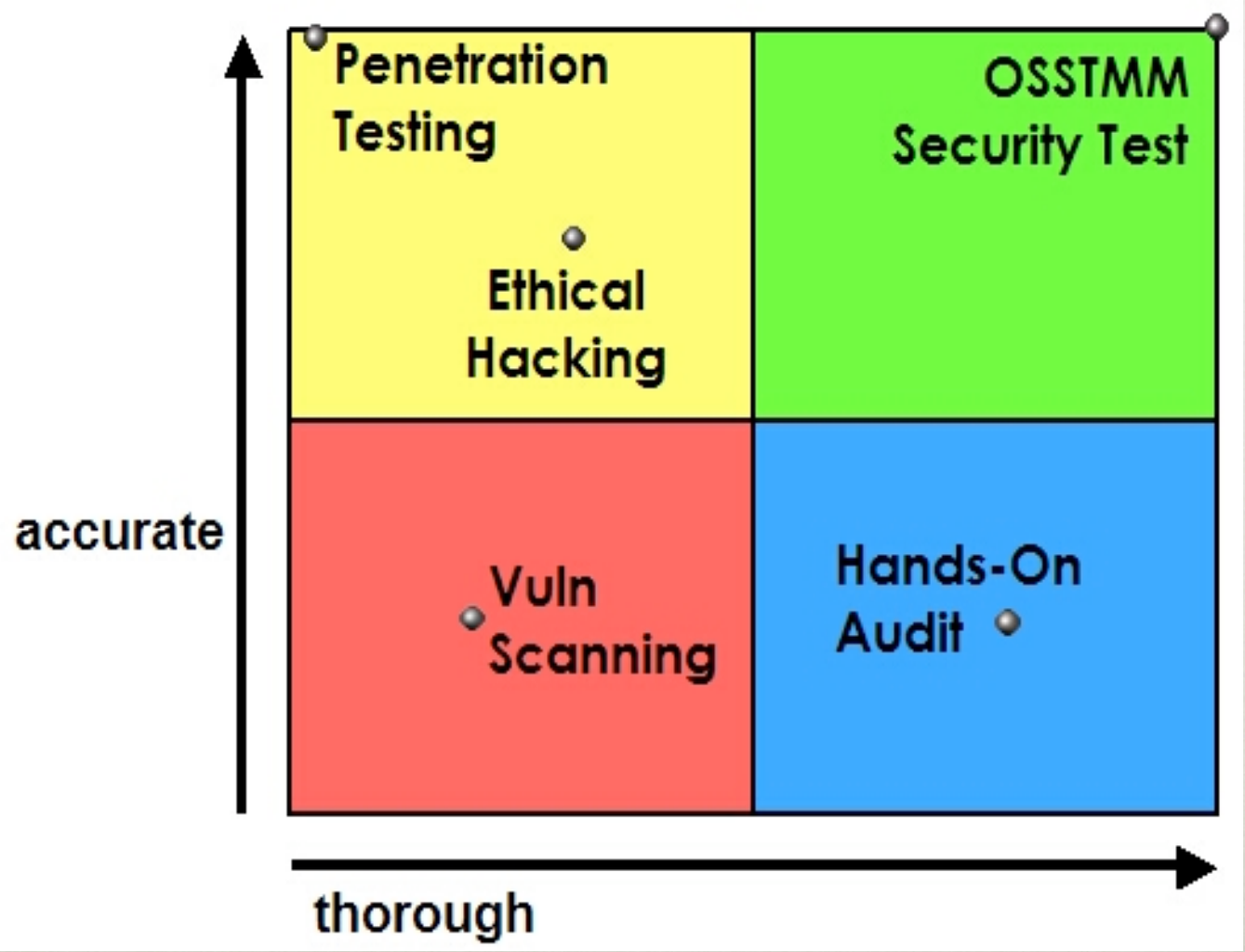
Network Security Testing

CS 155 Elie Bursztein

Why testing security

- Get a snapshot of the current security
- Evaluate the capacity to face intrusion
- Test backup plan

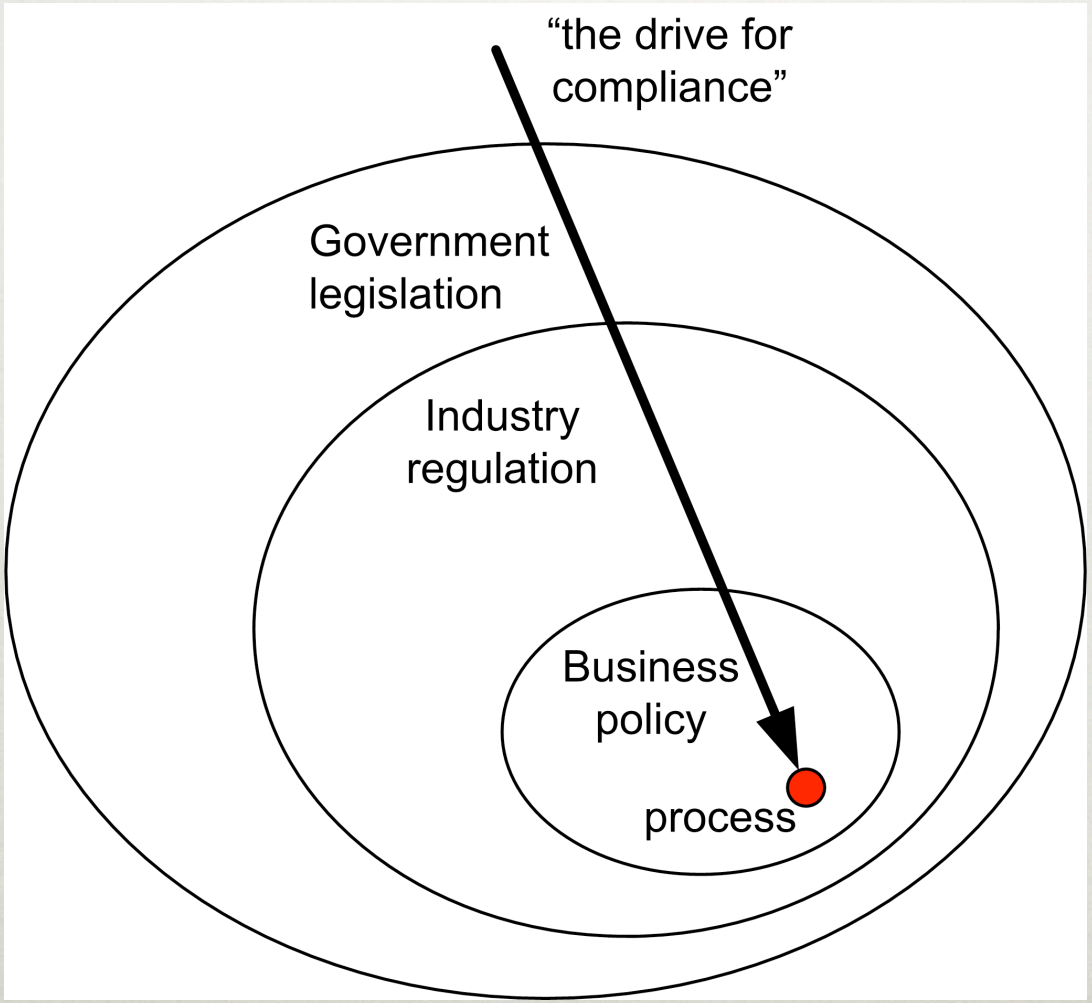
SCOPE


Penetration Testing
OSSTMM Security Test
Ethical Hacking
Vuln Scanning
Hands-On Audit
accurate
thorough

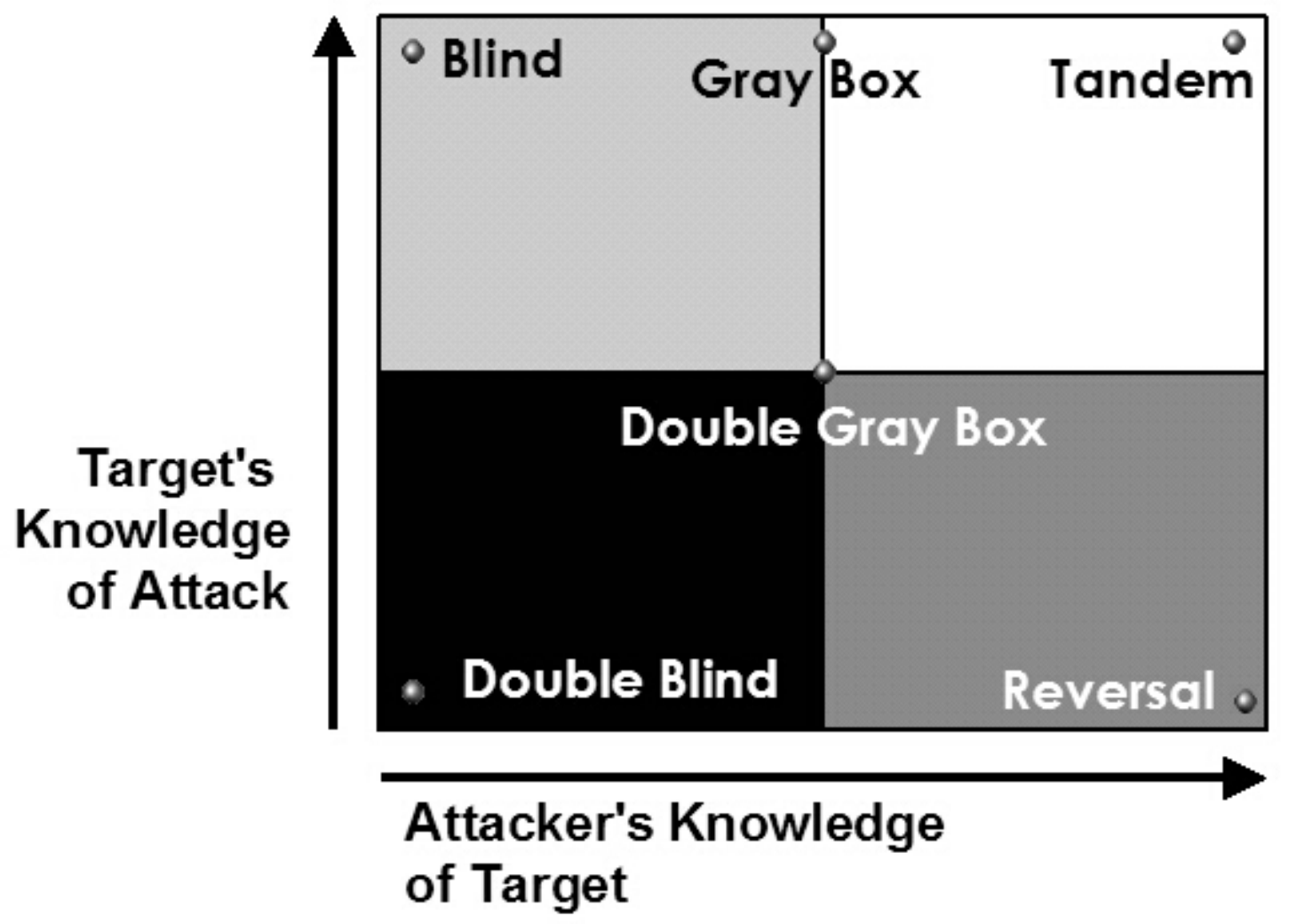
Results

- Date /type
- Duration
- Auditor and analyst associated
- Test type
- Scope
- Test index
- Channel test
- Test vector
- Verified test and metrics calculations of the operational protection levels, loss controls, and security limitations
- Knowledge of which tests have been completed, not completed, or only partially completed, and to what extent
- Any issues regarding the test and the validity of the results
- Test error margins
- Any processes which influence the security limitations
- Any unknowns or anomalies

SCOPE


“the drive for compliance”
Government legislation
Industry regulation
Business policy
process

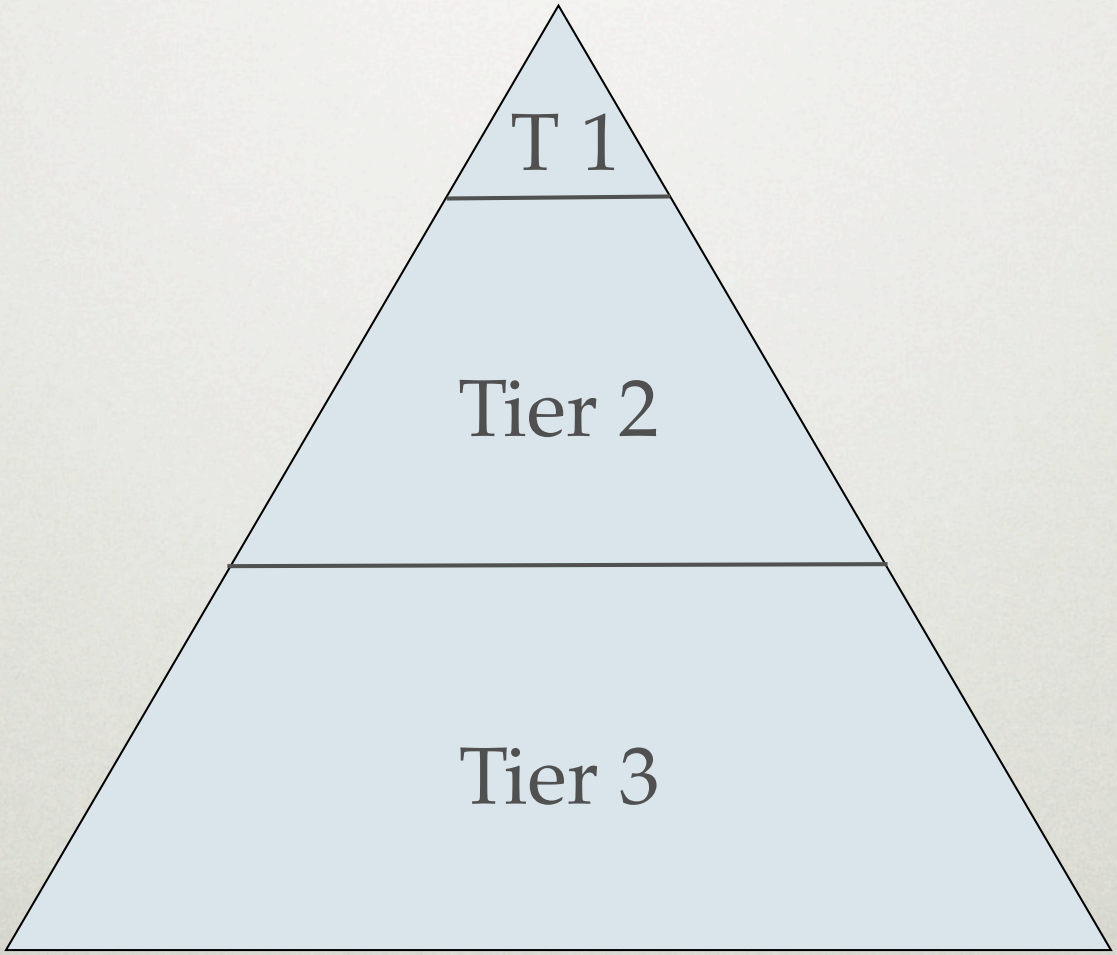
Security Test Type



Channel

- Physsec
  - Human
  - Physical
- SPECSEC
  - Wireless communication
- COMSEC
  - Data networks
  - Telecommunication

Hacker Skill Level



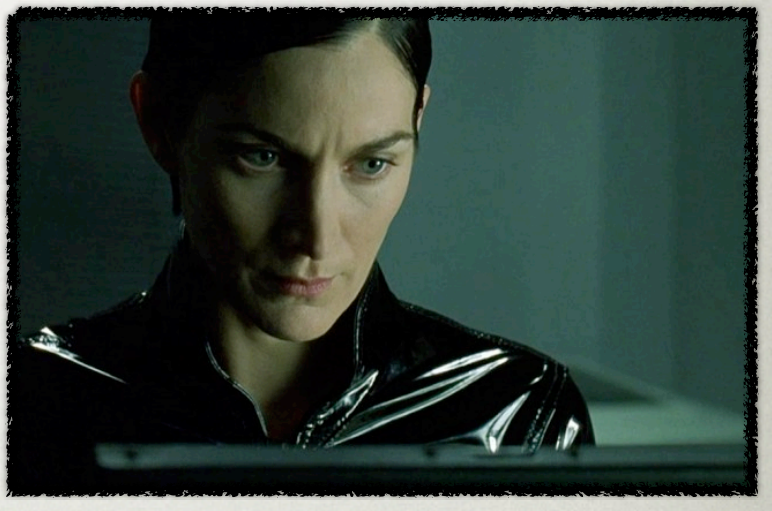
Network techniques toolbox

- Network scouting
- Os fingerprinting
- Vulnerability scanner
- Network trace analysis

Network scouting

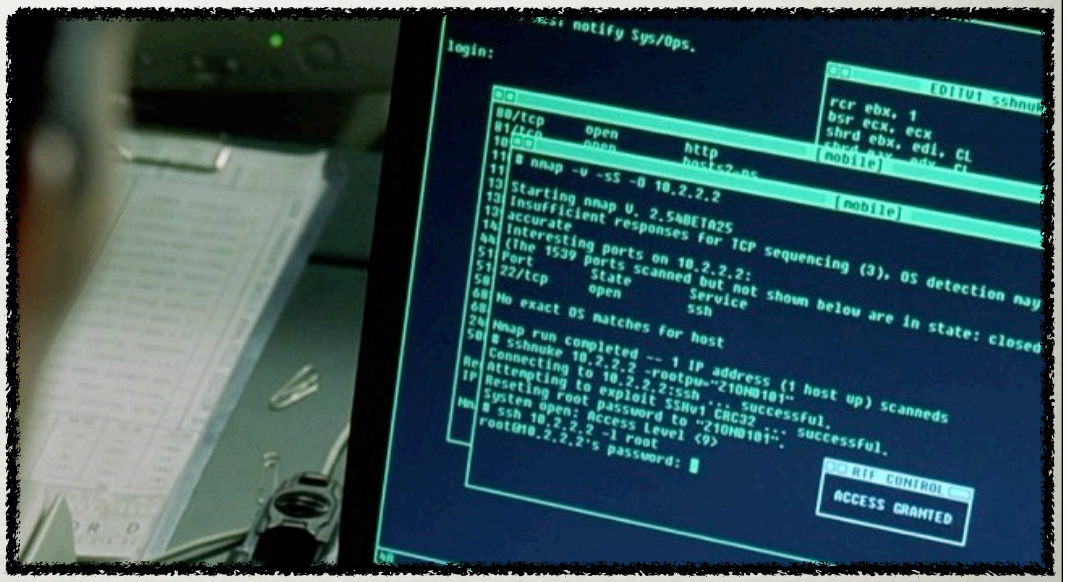
Scouting toolbox

- Unix standard tools
- Nmap ("Network Mapper")
  - Free and open source
  - Leading network scanner

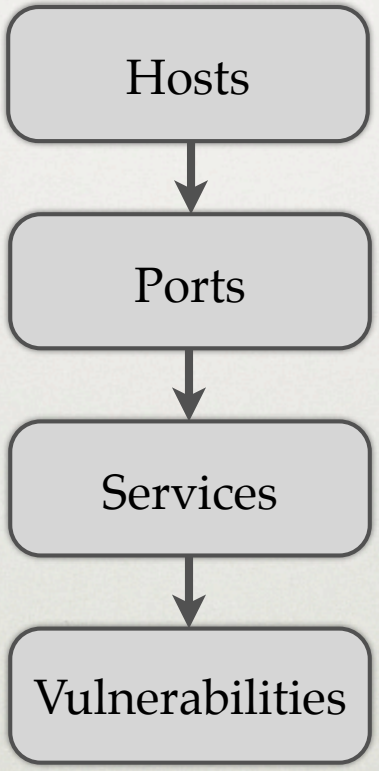


Why scouting is important ?

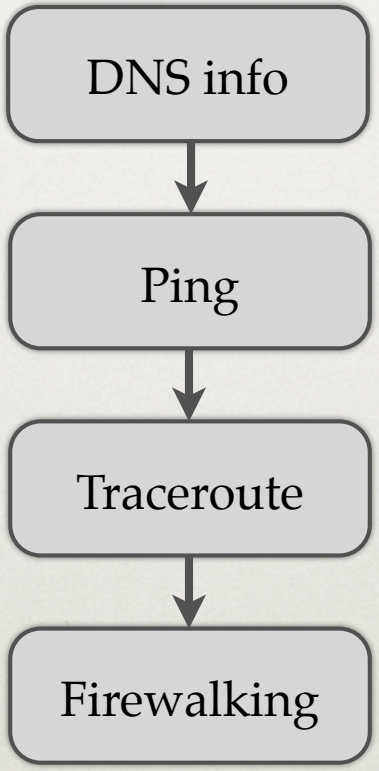
- Scouting is the first step
- You can't attack what you don't know



Scouting Process overview



Topological Mapping



Whois

Domain Name: STANFORD.EDU

Registrant:

Stanford University

The Board of Trustees of the Leland Stanford Junior University

241 Panama Street, Pine Hall, Room 115

Stanford, CA 94305-4122

UNITED STATES

Whois

Administrative Contact:

Domain Admin
Stanford University
241 Panama Street Pine Hall, Room 115
Stanford, CA 94305-4122
UNITED STATES
(650) 723-4328
sunet-admin@stanford.edu

Whois

Name Servers:

ARGUS.STANFORD.EDU 171.64.7.115

AVALLONE.STANFORD.EDU 171.64.7.88

ATALANTE.STANFORD.EDU 171.64.7.61

AERATHEA.STANFORD.EDU 152.3.104.250

Digging DNS record

- Dig **stanford.edu**
- ;; ANSWER SECTION:
- stanford.edu. 3600 IN A 171.67.216.3
- stanford.edu. 3600 IN A 171.67.216.4
- stanford.edu. 3600 IN A 171.67.216.7
- stanford.edu. 3600 IN A 171.67.216.8
- stanford.edu. 3600 IN A 171.67.216.9

- ;; AUTHORITY SECTION:
- stanford.edu. 172800 IN NS Avallone.stanford.edu.
- stanford.edu. 172800 IN NS Argus.stanford.edu.
- stanford.edu. 172800 IN NS Atalante.stanford.edu.
- stanford.edu. 172800 IN NS Aerathea.stanford.edu.

- ;; ADDITIONAL SECTION:
- Argus.stanford.edu. 3600 IN A 171.64.7.115
- Avallone.stanford.edu. 3600 IN A 171.64.7.88
- Atalante.stanford.edu. 3600 IN A 171.64.7.61

Zone transfer

- Allow to dump the entire zone
- Nowadays usually correctly protected
- dig @server domain axfr

Port scanning

- Find which port are open

```
Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-05-11 16:37
PDT
Interesting ports on localhost (127.0.0.1):
Not shown: 996 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
631/tcp  open  ipp
9050/tcp open  tor-socks
```

Ping

- Standard : Use icmp
- TCP work as well: TCP port 80
- ARP ping (lan only)

```
box:~# arping 192.168.0.1
ARPING 192.168.0.1
60 bytes from 00:21:91:f8:48:3a (192.168.0.1): index=0 time=6.410 msec
60 bytes from 00:21:91:f8:48:3a (192.168.0.1): index=1 time=3.351 msec
60 bytes from 00:21:91:f8:48:3a (192.168.0.1): index=2 time=2.839 msec
60 bytes from 00:21:91:f8:48:3a (192.168.0.1): index=3 time=7.165 msec
```

Finding routers

- traceroute
- Play with TTL
- Various protocols produce various results
- Established traceroute

Project 2 !

Traceroute example

traceroute to www.l.google.com (74.125.19.147), 64 hops max, 40 byte packets

1 171.66.32.1 1.329 ms 0.820 ms 0.893 ms

2 171.64.1.17 1.205 ms 0.884 ms 1.045 ms

3 171.64.1.129 1.910 ms 3.633 ms 1.835 ms

4 137.164.50.33 1.962 ms 2.540 ms 3.192 ms

5 137.164.46.203 4.371 ms 4.424 ms 3.677 ms

6 137.164.46.205 2.564 ms 3.099 ms 3.170 ms

7 137.164.131.237 2.594 ms 3.804 ms 2.433 ms

8 137.164.130.94 2.789 ms 2.695 ms 2.715 ms

9 216.239.49.250 3.878 ms 5.500 ms 5.405 ms

10 209.85.251.94 7.837 ms 4.840 ms 12.804 ms

11 74.125.19.147 3.637 ms 4.196 ms 6.283 ms

Asymmetric routing

- Routing policy are complex
- Outgoing route can be different from incoming route

Detecting asymmetric route

- Use the IP record option
- Limited to 9 records
- Some routers do ignore this option
- Use ping -R

Asymmetric route example

traceroute to www.l.google.com (74.125.19.147), 64 hops max, 40 byte packets

1 171.66.32.1 1.329 ms 0.820 ms 0.893 ms

2 171.64.1.17 1.205 ms 0.884 ms 1.045 ms

3 171.64.1.129 1.910 ms 3.633 ms 1.835 ms

4 137.164.50.33 1.962 ms 2.540 ms 3.192 ms

5 137.164.46.203 4.371 ms 4.424 ms 3.677 ms

6 137.164.46.205 2.564 ms 3.099 ms 3.170 ms

7 137.164.131.237 2.594 ms 3.804 ms 2.433 ms

8 137.164.130.94 2.789 ms 2.695 ms 2.715 ms

9 216.239.49.250 3.878 ms 5.500 ms 5.405 ms

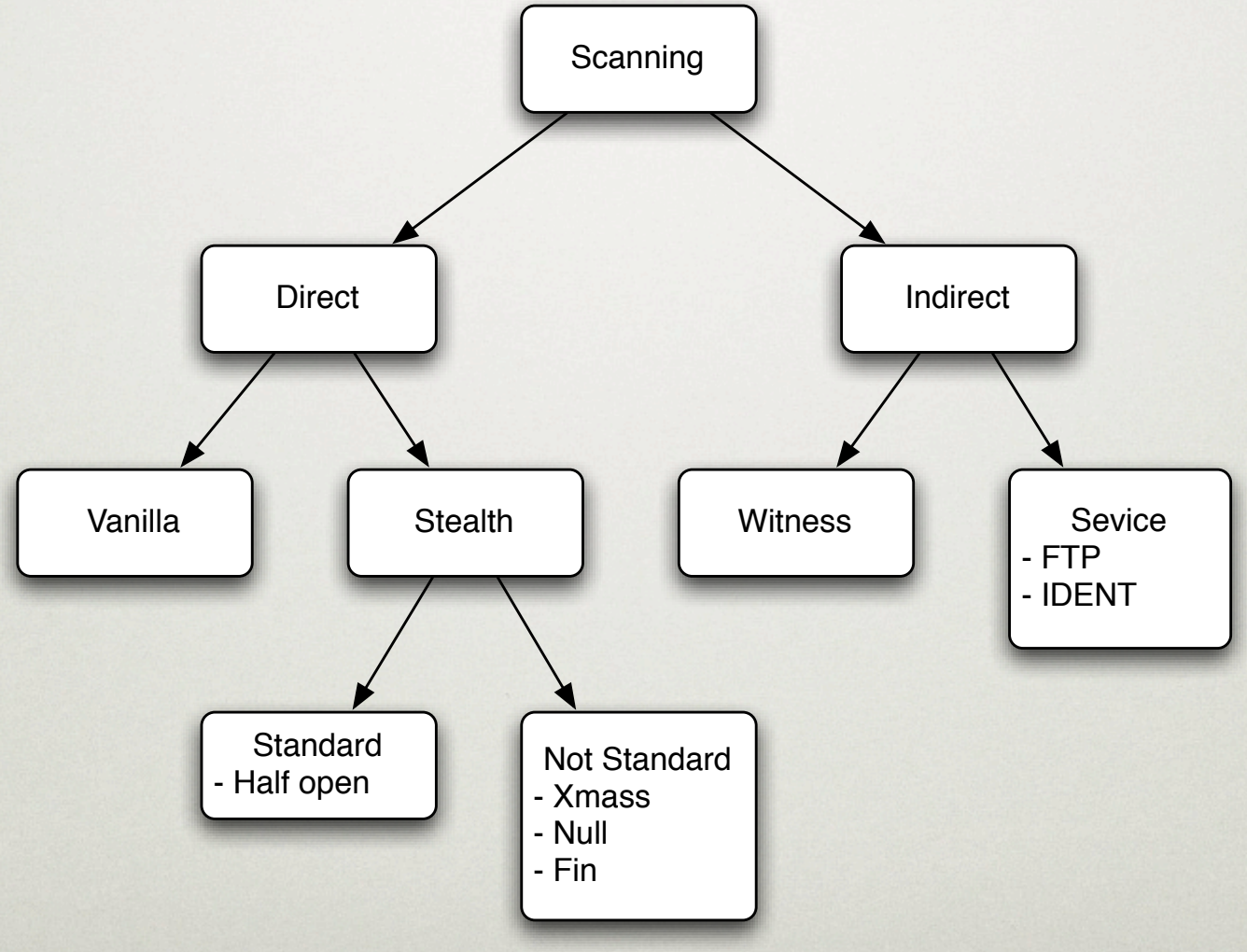
10 209.85.251.94 7.837 ms 4.840 ms 12.804 ms

11 74.125.19.147 3.637 ms 4.196 ms 6.283 ms

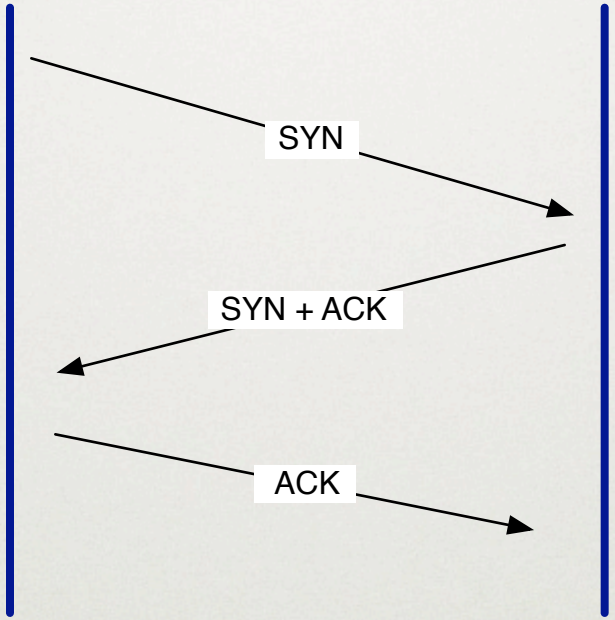
PING 209.85.251.94 (209.85.251.94): 56 data bytes
64 bytes from 209.85.251.94: icmp_seq=0 ttl=55 time=21.108 ms
RR: 171.64.1.18
171.64.1.34
137.164.50.34
137.164.46.200
137.164.131.238
137.164.130.93
74.14.232.245
209.85.251.93

64 bytes from 209.85.251.94: icmp_seq=1 ttl=55 time=13.601 ms (same route)

Scan types

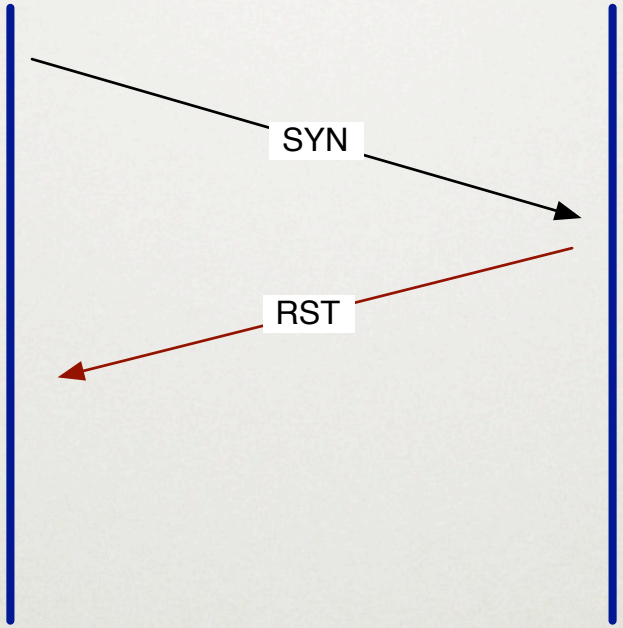

Scanning
Direct
Indirect
Vanilla
Stealth
Witness
Sevice
- FTP
- IDENT
Standard
- Half open
Not Standard
- Xmass
- Null
- Fin

Vanilla scan 1

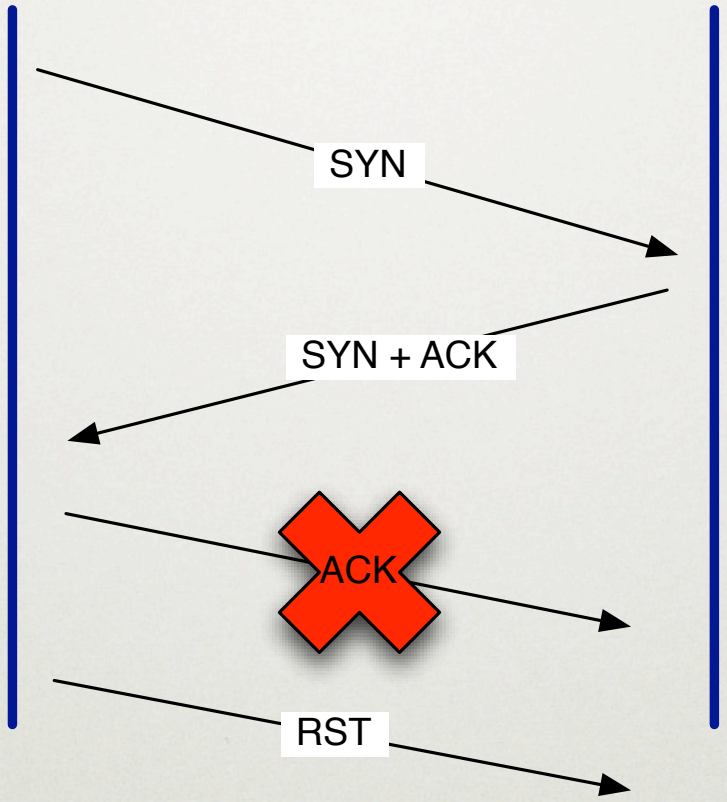


Scanner Server

Vanilla scan 2



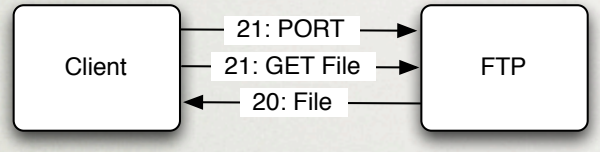
HalfOpen scan



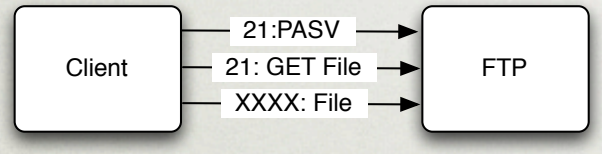
Not standard scan

- Violate the RFC
  - Null scan : no flag
  - Xmas scan : all flag
  - Fin scan: Fin flag
  - Maimon scan
  - ACK scan / Windows scan

FTP

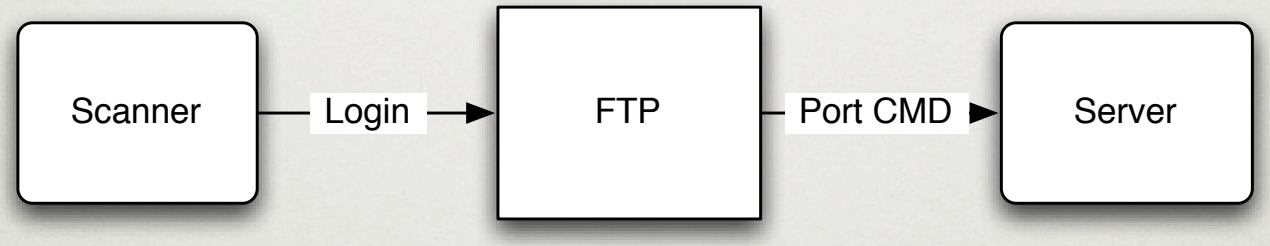


Active



Passive

Bounce Scan



Idle Scan

- Stealthiest scan
- The victim never see the scanner ip
- HomeWork !

Additional Manipulation

- If you can't be stealth, be noisy : decoys
- Try to confuse NIDS :
  - Fragmentation
  - Horizontal scan

Nmap vs Snort

Type	Vanilla	Half	Fin	Null	UDP
	-P0 -sT	-P0 -sF	-P0 -Sx	-P0 -sN	-P0 -sU
Default	✗	✗	✗	✗	✓
Frag (-f)		✗	✗	✗	
Slow (-T2)	✗	✗	✗	✗	✓
Very slow (-T1)	✓	✓	✗	✗	✓
Extremely slow (-T0)	✓	✓	✗	✗	✓

Service identification

- Grab banner
- Defense : slow down null probe
- Nmap have anti-defense....
- Remove / customize the banner

Example

Interesting ports on whispermoon (213.215.31.18):

Not shown: 989 closed ports

PORT STATE SERVICE VERSION

21/tcp open ftp (Generally vsftp or WU-FTPD)

22/tcp open ssh OpenSSH 4.7p1 Debian 8ubuntu1.2 (protocol 2.0)

25/tcp open smtp Postfix smtpd

80/tcp open http Apache httpd 2.2.8 ((Ubuntu) PHP/5.2.4-2ubuntu5.5 with Suhosin-Patch mod_ssl/2.2.8 OpenSSL/0.9.8g)

135/tcp filtered msrpc

139/tcp filtered netbios-ssn

443/tcp open ssl/http Apache httpd 2.2.8 ((Ubuntu) PHP/5.2.4-2ubuntu5.5 with Suhosin-Patch mod_ssl/2.2.8 OpenSSL/0.9.8g)

445/tcp filtered microsoft-ds

993/tcp open ssl/imap Dovecot imapd (SASL enabled)

995/tcp open ssl/pop3

DNS version probing

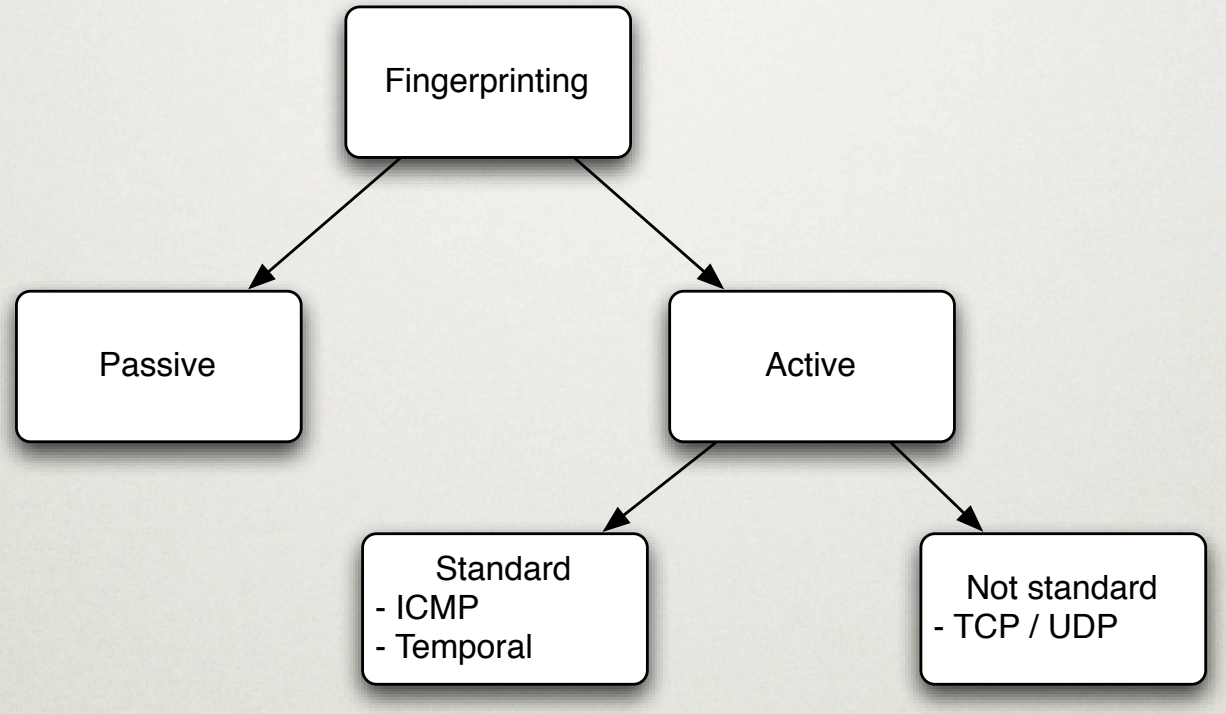
- dig @Argus.stanford.edu txt chaos version.bind
- version.bind. 0 CH TXT "9.4.2-P2"

Fingerprinting

Outline

- Network scouting
- Os fingerprinting
- Vulnerability scanner
- Network trace analysis

Type of fingerprinting



Key idea

- The RFC is not well specified
- Every programmer believe they now better
- Network stack exhibit subtle difference

Passive fingerprinting

- Look at packet that flow through the network
- Four type of passive fingerprinting
  - machines that connect to you (SYN)
  - machines you connect to (SYN+ACK)
  - machine you cannot connect to (RST)
  - machines whose communications you can observe.

POF Discriminators

- Format : wwww:ttt:mmm:D:W:S:N:I:OS Description
  - wwww - window size
  - ttt - time to live
  - mmm - maximum segment size
  - D - don't fragment flag (0=unset, 1=set)
  - W - window scaling (-1=not present, other=value)
  - S - sackOK flag (0=unset, 1=set)
  - N - nop flag (0=unset, 1=set)
  - I - packet size (-1 = irrevelant)

POF output

- <Wed Feb 27 18:26:58 2008> 213.215.x.x:45291 - Linux 2.6 (newer, 2) (up: 1421 hrs) -> 208.83.x.x:2703 (distance 0, link: ethernet/modem)
- <Wed Feb 27 18:27:02 2008> 212.24.x.x:62994 - FreeBSD 5.3-5.4 (up: 4556 hrs) -> 213.215.x.x:80 (distance 9, link: ethernet/modem)
- <Wed Feb 27 18:27:16 2008> 90.2.x.x:1322 - Windows 2000 SP4, XP SP1+ -> 213.215.x.x:80 (distance 9, link: pppoe (DSL))

Computing Distance

- Use fingerprinting
- Use an heuristic based on the closest ^2
  - 62 -> 64 - 62 = 2
  - 118 -> 128 - 118 = 10

Link Type

- Analyze the MTU (Maximum transmission unit)
- Some medium have a very distinct type
  - 1462, "sometimes DSL (5)”
  - 1656, "Ericsson HIS"

M. Zalewski

Firewall detection

- Look at the don't fragment bit (DF)

Nmap Fingerprinting (v2)

- Mix all previous techniques
- 7 TCP probes, 1 ICMP, 1 UDP
  - **TCP** probes are sent exactly 110 milliseconds apart
  - Required to analyze
    - initial sequence numbers
    - IP IDs
    - TCP timestamps

Active fingerprinting

1. ECN notification
2. window scale (10), NOP, MSS (1460), timestamp (TSval: 0xFFFFFFFF; TSecr: 0), SACK permitted. The window field is 1.
3. MSS (1400), window scale (0), SACK permitted, timestamp (TSval: 0xFFFFFFFF; TSecr: 0), EOL. The window field is 63.
4. Timestamp (TSval: 0xFFFFFFFF; TSecr: 0), NOP, NOP, window scale (5), NOP, MSS (640). The window field is 4.
5. SACK permitted, Timestamp (TSval: 0xFFFFFFFF; TSecr: 0), window scale (10), EOL. The window field is 4.
6. MSS (536), SACK permitted, Timestamp (TSval: 0xFFFFFFFF; TSecr: 0), window scale (10), EOL. The window field is 16.
7. MSS (265), SACK permitted, Timestamp (TSval: 0xFFFFFFFF; TSecr: 0). The window field is 512.

Old Nmap (4.11)

nmap -v -O 192.168.0.1

Interesting ports on 192.168.0.1:

Not shown: 1678 closed ports

PORT STATE SERVICE

80/tcp open http

4444/tcp open krb524

MAC Address: 00:21:91:F8:48:3A (Unknown)

No exact OS matches for host (If you know what OS is running on it, see http://www.insecure.org/cgi-bin/nmap-submit.cgi).

New Nmap 4.8x

- **nmap -O -v 192.168.0.1**

  PORT STATE SERVICE

  80/tcp open http

  4444/tcp open krb524

  8099/tcp open unknown

  MAC Address: 00:21:91:F8:48:3A (D-Link)

  Device type: print server | router

  Running: D-Link embedded

  OS details: D-Link DPR-1260 print server, or DGL-4300 or DIR-655 router

  Network Distance: 1 hop

  TCP Sequence Prediction: Difficulty=174 (Good luck!)

  IP ID Sequence Generation: Incremental

Why IDS detect it ?

- ICMP : TOS IP_TOS_RELIABILITY
- UDP : C repeated 300 times
- TCP : non standard packets

OS uptime

- TCP timestamp is incremented each second by a known value x
- No random origin on Unices

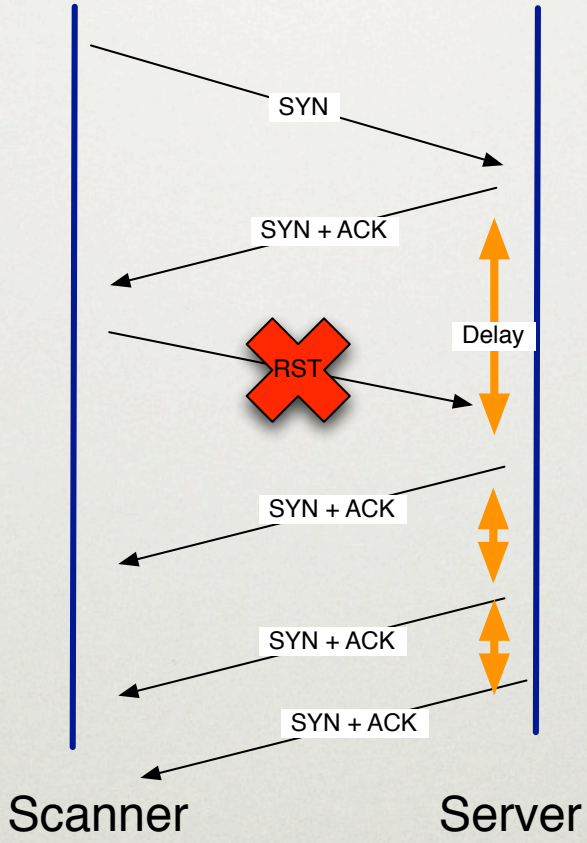
uptime = x * value

Nat Detection

- Inconstancy between the mss and the wss
- Use auto-increment field
  - IP ID (Windows)
  - Timestamp (Unix / Windows server)

Linear regression

Temporal Fingerpinting

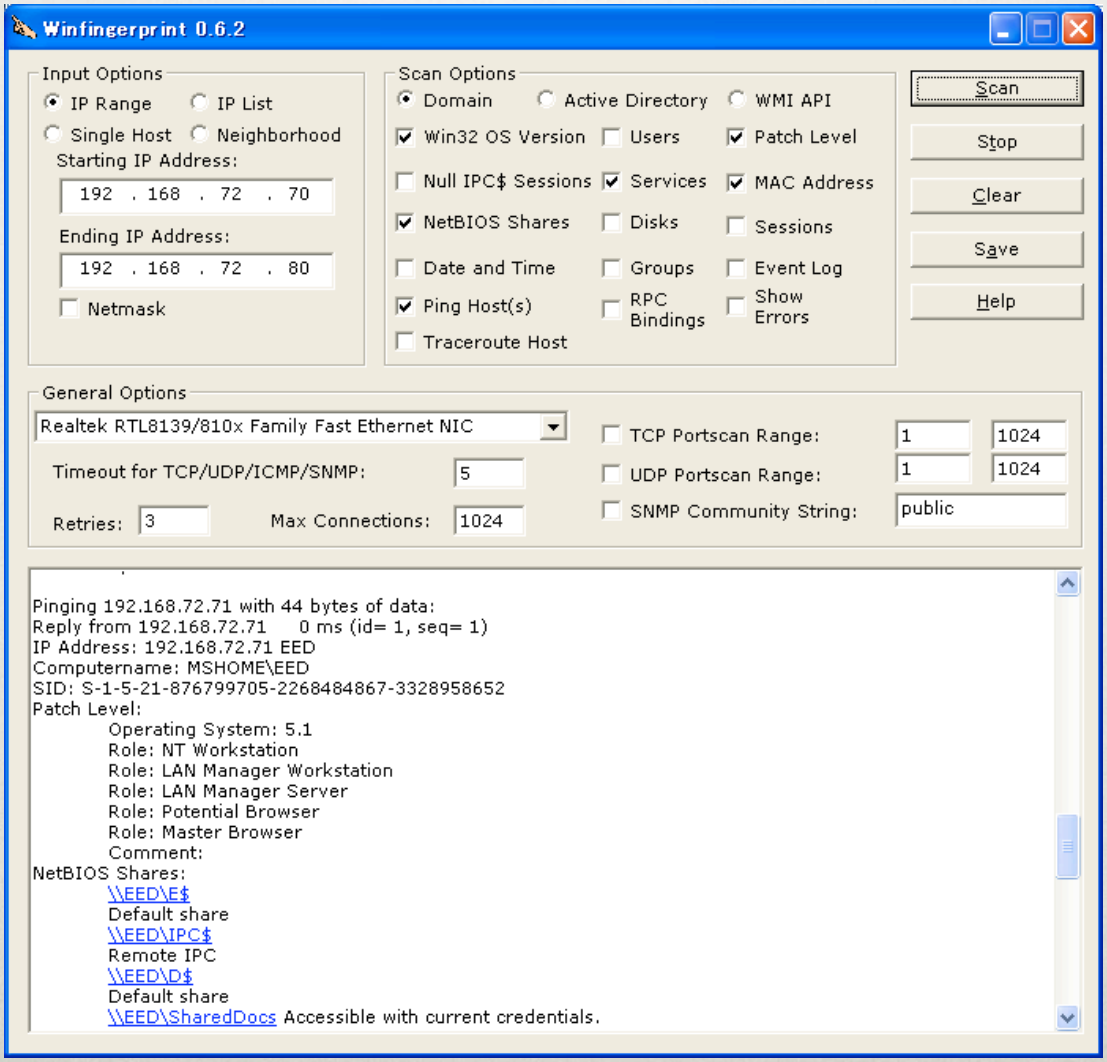


Franck Veysset, Olivier Courtay, Olivier Heen, Intranode Research Team 2002

Protocol Specific

- Netbios (Win)
- WMI (Win)
- SNMP

Winfingerprint



Vulnerability Scanner

Outline

- Network scouting
- Os fingerprinting
- Vulnerability scanner
- Network trace analysis

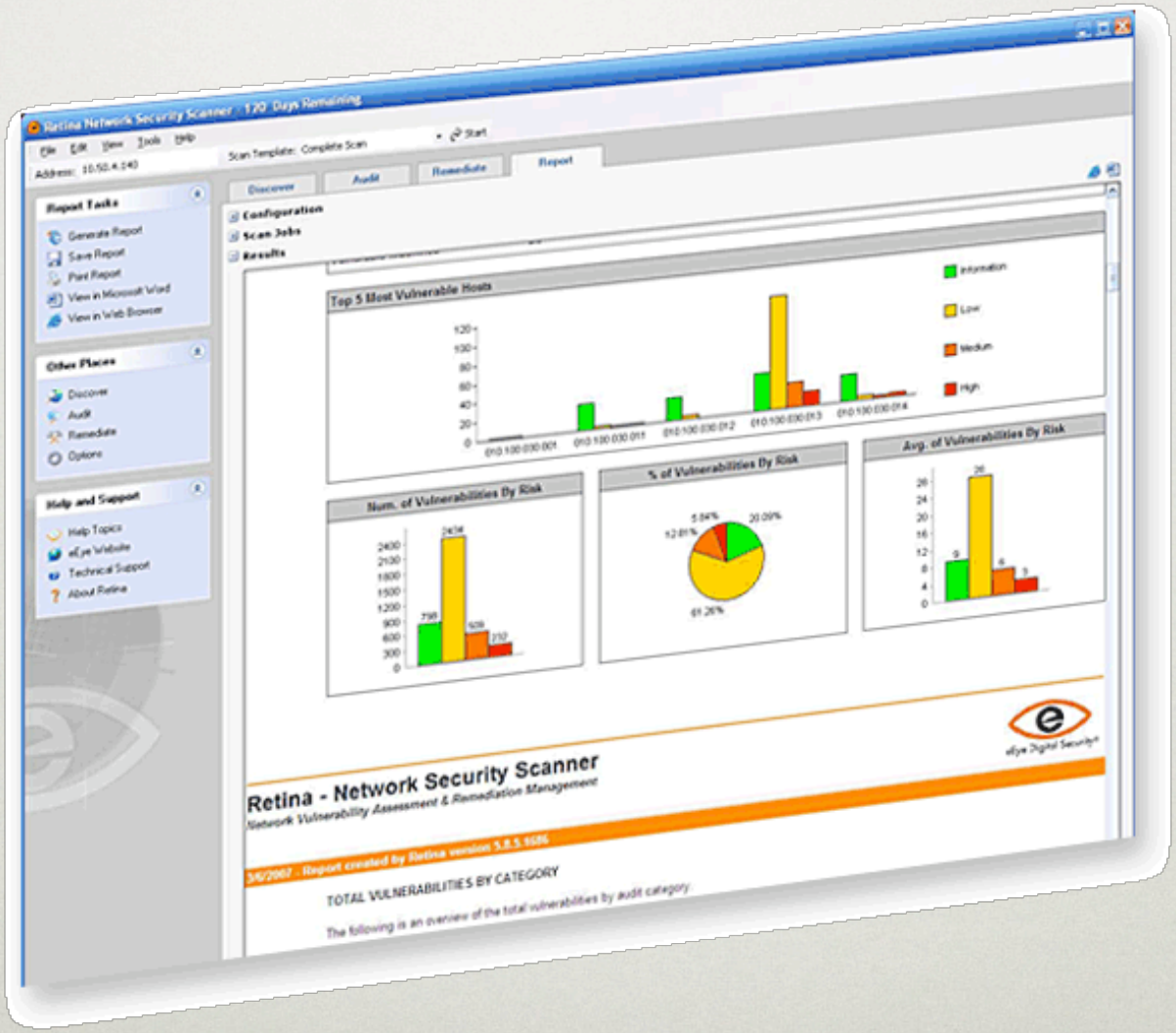
What it is

- A tool that given a set of
  - vulnerabilities (plugins)
  - hosts (scouting)
- Tell which hosts is vulnerable to what

Type of assessment

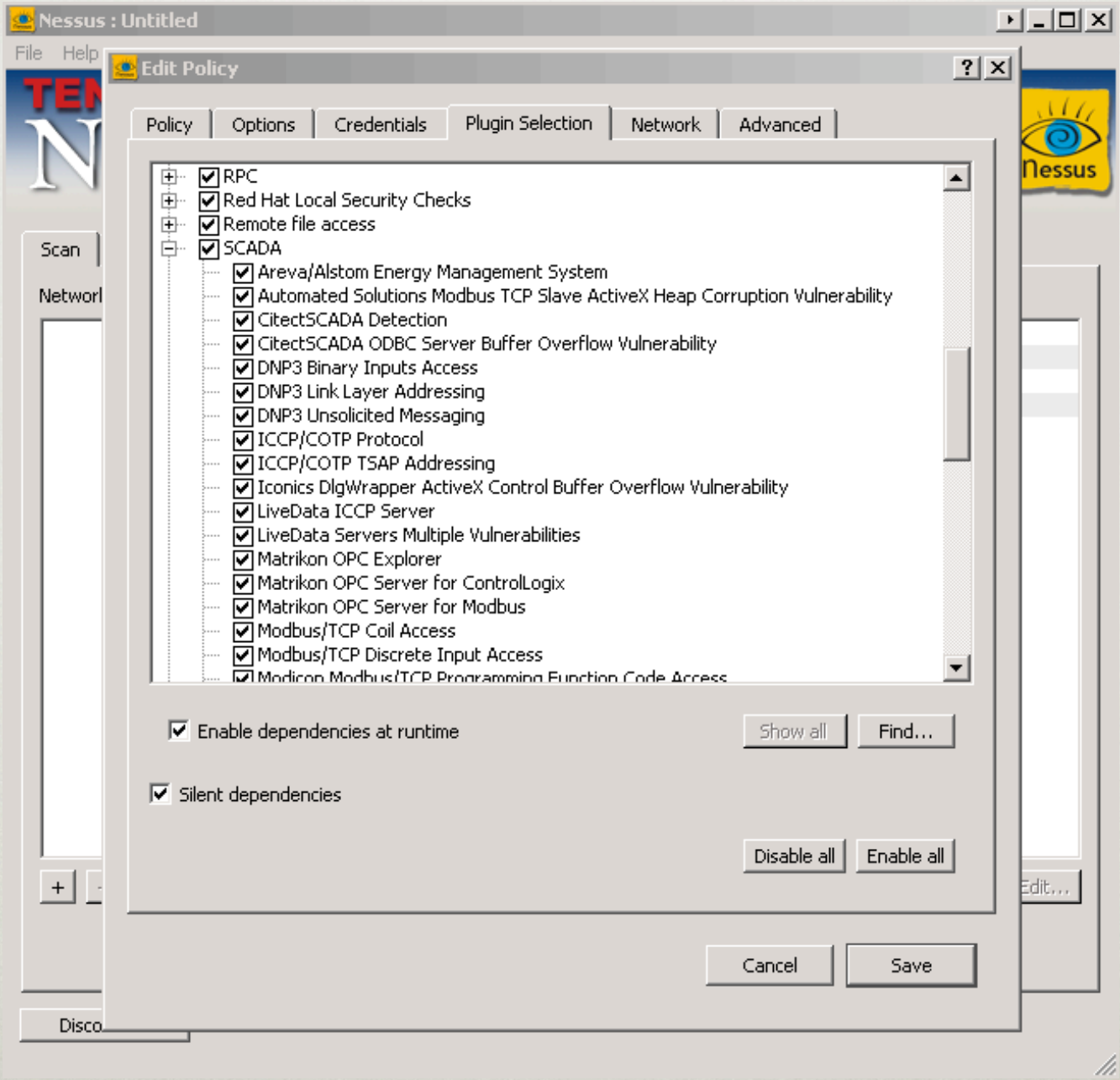
- Local
- Remote
- Modern vulnerability scanner
  - Mix remote/ local test
  - Keep a temporal record

Retina


Retina Network Security Scanner - 120 Days Remaining
Report Tasks
Generate Report
Save Report
Print Report
View in Microsoft Word
View in Web Browser
Other Places
Discover
Audit
Remediate
Options
Help and Support
Help Topics
eEye Website
Technical Support
About Retina
Discover
Audit
Remediate
Report
Configuration
Scan Jobs
Results
Top 5 Most Vulnerable Hosts
Num. of Vulnerabilities By Risk
% of Vulnerabilities By Risk
Avg. of Vulnerabilities By Risk
Retina - Network Security Scanner
Network Vulnerability Assessment & Remediation Management
TOTAL VULNERABILITIES BY CATEGORY

Nessus

- Owned by tenable
- Open souce -> close source


Nessus : Untitled
File Help
Edit Policy
Policy
Options
Credentials
Plugin Selection
Network
Advanced
RPC
Red Hat Local Security Checks
Remote file access
SCADA
Areva/Alstom Energy Management System
Automated Solutions Modbus TCP Slave ActiveX Heap Corruption Vulnerability
CitectSCADA Detection
CitectSCADA ODBC Server Buffer Overflow Vulnerability
DNP3 Binary Inputs Access
DNP3 Link Layer Addressing
DNP3 Unsolicited Messaging
ICCP/COTP Protocol
ICCP/COTP TSAP Addressing
Iconics DlgWrapper ActiveX Control Buffer Overflow Vulnerability
LiveData ICCP Server
LiveData Servers Multiple Vulnerabilities
Matrikon OPC Explorer
Matrikon OPC Server for ControlLogix
Matrikon OPC Server for Modbus
Modbus/TCP Coil Access
Modbus/TCP Discrete Input Access
Enable dependencies at runtime
Show all
Find...
Silent dependencies
Disable all
Enable all
Cancel
Save
Scan
Nessus

Difficulties

- Keep an updated list of vulnerabilities
- Know the OS of the host
- Know the version of each service
- Is able to test without breaking the service

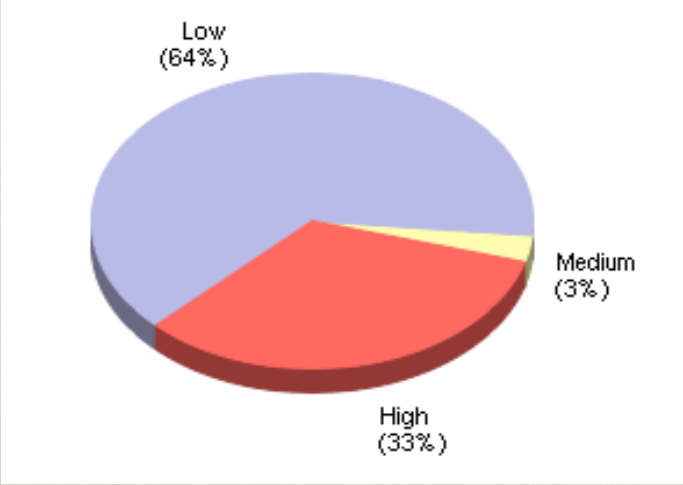
Nessus report

Vulnerability Summary

Network Profile

Host Count	16
Date of First Scan	2007-05-20
Date of Last Scan	2007-05-23

Vulnerabilities - Summary By Severity



Count	Severity
1841	TOTAL
0	Critical
608	High
54	Medium
1179	Low

Nessus report

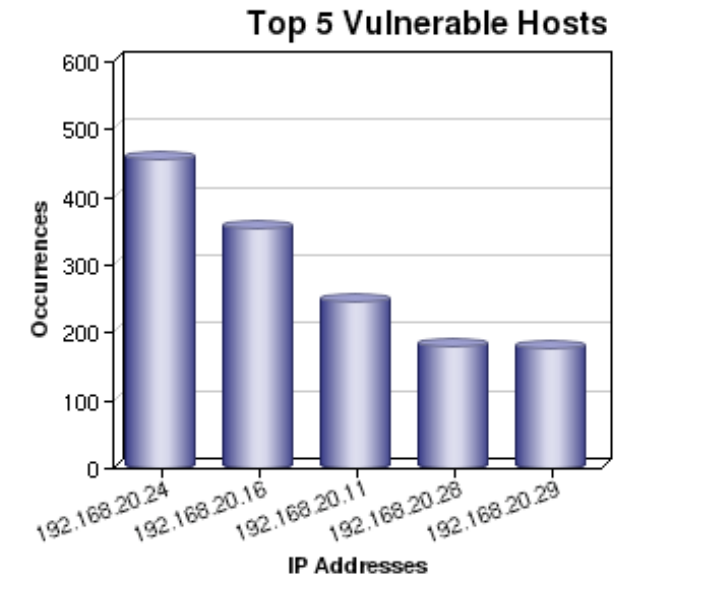
Top 5 Plugin Families

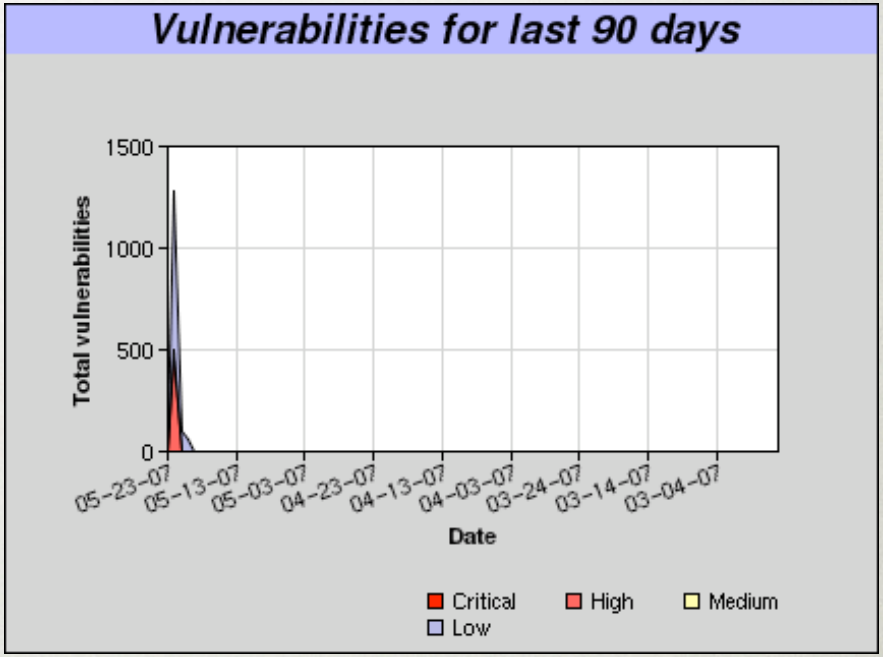
Total	Plugin Family
545	Generic (PVS)
435	Red Hat Local Security Checks
214	Compliance Checks
126	Port scanners
111	General

Vulnerabilities - Summary By Assets

Total	Asset Tag
47	Network Equipment
730	OS Unix
907	OS Windows Managed
79	OS Windows Unmanaged
1695	Service HTTP

Nessus report





Tenable

Nessus report

Nessus ID	Total	Sev	Name	Family
17167	3	High	RHSA-2005-033: alsa	Red Hat Local Security Checks
17169	3	High	RHSA-2005-035: libtiff	Red Hat Local Security Checks
17170	3	High	RHSA-2005-036: vim	Red Hat Local Security Checks
17171	3	High	RHSA-2005-037: ethereal	Red Hat Local Security Checks
17172	3	High	RHSA-2005-040: enscript	Red Hat Local Security Checks
17173	3	High	RHSA-2005-045: krb	Red Hat Local Security Checks
17174	3	High	RHSA-2005-053: cups	Red Hat Local Security Checks

Nessus report

Nessus ID	Total	Sev	Name	Family
10395	6	Medium	SMB shares enumeration	Windows
10758	4	Medium	Check for VNC HTTP	Backdoors
10281	3	Medium	Telnet Server Detection	Service detection
03754	2	Medium	Portable OpenSSH < 4.4.p1	SSH (PVS)
10539	2	Medium	Usable remote name server	General
11853	2	Medium	Apache < 2.0.48	Web Servers
02059	1	Medium	Shareaza P2P fileshare client is installed	PeerToPeer (PVS)
02286	1	Medium	PHP Arbitrary File Upload Vulnerability	Web Servers (PVS)
03112	1	Medium	Apache HTTP Smuggling vulnerability	Web Servers (PVS)

Nessus report

Asset	Total	Critical	High	Medium	Low
Network Equipment	47	0	0	2	45
OS Unix	730	0	439	7	284
OS Windows Managed	907	0	164	37	706
OS Windows Unmanaged	79	0	2	3	74
Service HTTP	1695	0	605	49	1041
Service SSH	427	0	150	10	267
Service Telnet	460	0	164	36	260
VMWare Systems	261	0	23	20	218
Web Server - Apache	383	0	150	10	223
Web Server - IIS	451	0	164	35	252

Tenable

Network Trace analysis

Outline

- Network scouting
- Os fingerprinting
- Vulnerability scanner
- Network trace analysis

WIKIPEDIA
The Free Encyclopedia

navigation

- Main page
- Contents
- Featured content
- Current events
- Random article

search

Go Search

interaction

- About Wikipedia
- Community portal
- Recent changes
- Contact Wikipedia
- Donate to Wikipedia
- Help

toolbox

- What links here
- Related changes

arping

From Wikipedia, the free encyclopedia

The introduction to this article provides **insufficient context** for those unfamiliar with the subject.

Please help improve the article with a good introductory style.

arping is a computer program that can be used to discover information about a computer network. The **arping** command tests whether on the local network, and can get additional information about the device using that address.

The **arping** command is similar in function to ping, but it operates using Address Resolution Protocol (ARP) instead of Internet Control ARP, arping is only usable on the local network; in some cases the response will be coming, not from the arpinged host, but rather from in proxy ARP (such as a router).

There are two popular arping implementations. One is part of Linux iproute2, is Linux-only, and cannot currently cannot resolve MAC addresses to IP addresses. The other arping implementation, written by Thomas Habets, uses the platform-independent libraries libpcap and libnet, and works with a wide range of operating systems.

Example arping (iputils version) session:

```
ARPING 192.168.39.120 from 192.168.39.1 eth0
Unicast reply from 192.168.39.120 [00:01:80:38:F7:4C]  0.810ms
Unicast reply from 192.168.39.120 [00:01:80:38:F7:4C]  0.607ms
Unicast reply from 192.168.39.120 [00:01:80:38:F7:4C]  0.602ms
Unicast reply from 192.168.39.120 [00:01:80:38:F7:4C]  0.606ms
Sent 4 probes (1 broadcast(s))
Received 4 response(s)
```

arping uses the ICMP protocol to ping MAC addresses, so hosts which are configured to ignore ICMP pings will not respond to arping either.

17:31:16.301217 IP (tos 0x0, ttl 42, id 24244, offset 0, flags [none], proto: TCP (6), length: 44) 192.168.0.194.52232 > 192.168.0.1.80: S, cksum 0x6485 (correct), 3647930309:3647930309(0) win 3072 <mss 1460>

17:31:16.301667 IP (tos 0x0, ttl 57, id 37298, offset 0, flags [none], proto: TCP (6), length: 44) 192.168.0.194.52232 > 192.168.0.1.81: S, cksum 0x6884 (correct), 3647930309:3647930309(0) win 2048 <mss 1460>

17:31:16.301987 IP (tos 0x0, ttl 64, id 48783, offset 0, flags [none], proto: TCP (6), length: 44) 192.168.0.1.80 > 192.168.0.194.52232: S, cksum 0xc685 (correct), 2609643106:2609643106(0) ack 3647930310 win 4096 <mss 1460>

17:31:16.417655 IP (tos 0x0, ttl 64, id 48786, offset 0, flags [none], proto: TCP (6), length: 44) 192.168.0.1.80 > 192.168.0.194.52425: S, cksum 0x8030 (correct), 2610399074:2610399074(0) ack 1654600479 win 4096 <mss 1460>

17:31:16.417679 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto: TCP (6), length: 40) 192.168.0.194.52425 > 192.168.0.1.80: R, cksum 0xcaf4 (correct), 1654600479:1654600479(0) win 0

17:31:17.021331 IP (tos 0x0, ttl 61, id 4162, offset 0, flags [none], proto: UDP (17), length: 328) 192.168.0.194.52300 > 192.168.0.1.39695: UDP, length 300

17:31:16.993102 IP (tos 0x4, ttl 58, id 43133, offset 0, flags [none], proto: ICMP (1), length: 178) 192.168.0.194 > 192.168.0.1: ICMP echo request, id 34388, seq 296, length 158

17:31:17.217108 IP (tos 0x0, ttl 41, id 17642, offset 0, flags [none], proto: TCP (6), length: 60) 192.168.0.194.52444 > 192.168.0.1.79: FP, cksum 0x5191 (correct), 1654600478:1654600478(0) win 65535 urg 0 <wscale 15,nop,mss 265,timestamp 4294967295 0,sackOK>

01:25:08.063167 192.168.1.40.http >
192.168.1.40.http: S [bad tcp cksum a8e4!]
3868:3868(0) win 2048 (ttl 255, id 3868, len 40

```
23:57:12.623167 192.168.1.2.40 > 192.168.1.3.netbios-ssn: S [tcp sum ok] 740990201:740990201(0) win 16384 <mss 1460,nop,nop,sackOK>
                (DF) (ttl 128, id 39059, len 48)

23:57:12.623167 192.168.1.3.netbios-ssn > 192.168.1.2.40: S [tcp sum ok] 3674022113:3674022113(0) ack 740990202 win 5840
                <mss 1460,nop,nop,sackOK> (DF) (ttl 64, id 0, len 48)

23:57:12.623167 192.168.1.2.40 > 192.168.1.3.netbios-ssn: . [tcp sum ok] 1:1(0) ack 1 win 17520 (DF) (ttl 128, id 39060, len 40)

23:57:12.623167 192.168.1.2.40 > 192.168.1.3.netbios-ssn: P 1:256(255) ack 1 win 17520 urg 255
>>> NBT Packet
flags=0x42
NBT - Unknown packet type
Type=0x424F4F4F
Data: (251 bytes)
[000] 4F 4F 4F 4F 4F 4F 4F 4F  4F 4F 4F 4F 4D 00 20 00  OOOOOOOO OOOOM. .
[010] 39 00 35 00 00 00 FF FF  C0 F8 12 00 99 2A 41 00  9.5..... .....*A.
[020] 10 F9 12 00 28 F9 00 00  00 00 11 00 00 00 70 6F  ....(... ......po
[030] 72 74 20 34 30 00 00 00  00 00 19 00 00 00 00 00  rt 40... ........
[040] 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
[050] 00 00 19 00 00 00 F0 84  15 08 90 A9 15 08 08 A9  ........ ........
[060] 15 08 00 00 00 00 00 00  00 00 11 00 00 00 5F 59  ........ ......_Y
[070] 23 40 10 B4 01 40 00 00  00 00 11 00 00 00 30 98  #@...@.. ......0.
[080] 15 08 C8 C8 15 08 00 00  00 00 21 00 00 00 2F 6C  ........ ..!.../l
[090] 69 62 2F 6C 69 62 6E 73  73 5F 6E 69 73 70 6C 75  ib/libns s_nisplu
[0A0] 73 2E 73 6F 2E 32 00 00  00 00 19 04 00 00 8B 50  s.so.2.. .......P
[0B0] 60 91 F0 78 47 9B 70 2C  D7 9B 70 91 BC 9C F0 48  `..xG.p, ..p....H
[0C0] C0 9D 70 FE 89 9E F0 2A  A0 9F F0 A5 60 A0 F0 0C  ..p....* ....`...
[0D0] 80 A1 F0 12 2E A2 F0 4C  7A A3 F0 81 35 A4 70 23  .......L z...5.p#
[0E0] 5E A5 F0 35 25 A6 F0 9B  27 A7 70 26 58 A8 F0 7D  ^..5%... '.p&X..}
[0F0] 07 A9 70 34 EE A9 F0 5F  E7 AA F0                 ..p4..._ ...

 (DF) (ttl 128, id 39061, len 295)

23:57:12.623167 192.168.1.3.netbios-ssn > 192.168.1.2.40: . [tcp sum ok] 1:1(0) ack 256 win 5840 (DF) (ttl 64, id 1714, len 40)
23:57:12.633167 192.168.1.3.netbios-ssn > 192.168.1.2.40: R [tcp sum ok] 1:1(0) ack 256 win 5840 (DF) (ttl 64, id 1715, len 40)
```

```
01:22:30.673167 192.168.1.3 > 192.168.1.40: icmp: echo request (frag 4321:1480@0+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@1480+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@2960+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@4440+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@5920+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@7400+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@8880+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@10360+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@11840+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@13320+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@14800+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@16280+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@17760+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@19240+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@20720+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@22200+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@23680+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@25160+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@26640+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@28120+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@29600+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@31080+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@32560+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@34040+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@35520+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@37000+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@38480+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@39960+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@41440+) (ttl 255, len 1500)
01:22:30.673167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@42920+) (ttl 255, len 1500)
01:22:30.683167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@44400+) (ttl 255, len 1500)
01:22:30.683167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@45880+) (ttl 255, len 1500)
01:22:30.683167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@47360+) (ttl 255, len 1500)
01:22:30.683167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@48840+) (ttl 255, len 1500)
01:22:30.683167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@50320+) (ttl 255, len 1500)
01:22:30.683167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@51800+) (ttl 255, len 1500)
01:22:30.683167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@53280+) (ttl 255, len 1500)
01:22:30.683167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@54760+) (ttl 255, len 1500)
01:22:30.683167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@56240+) (ttl 255, len 1500)
01:22:30.683167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@57720+) (ttl 255, len 1500)
01:22:30.683167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@59200+) (ttl 255, len 1500)
01:22:30.683167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@60680+) (ttl 255, len 1500)
01:22:30.683167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@62160+) (ttl 255, len 1500)
01:22:30.683167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@63640+) (ttl 255, len 1500)
01:22:30.683167 192.168.1.3 > 192.168.1.40: (frag 4321:1480@65120) (ttl 255, len 1500)
```

23:28:12.503167 192.168.1.2 > 192.168.0.3: icmp: 212.43.217.98 protocol 6 unreachable for 192.168.0.3.1200 > 212.43.x.x.ircd: [ltcp] (ttl 128, id 25159, len 30, bad cksum 0!) (ttl 128, id 21813, len 56)

23:28:28.693167 192.168.1.2 > 192.168.0.3: icmp: 212.43.217.98 protocol 6 unreachable for 192.168.0.3.1200 > 212.43.x.x.ircd: [ltcp] (ttl 52, id 17098, len 123, bad cksum 0!) (ttl 128, id 21989, len 56)

00:48:06.523167 192.168.1.3.smtp > 192.168.1.2.smtp: [no cksum] udp 28 (frag 1109:36@0+) (ttl 255, len 56)
00:48:06.543167 192.168.1.3.smtp > 192.168.1.2.smtp: [no cksum] udp 28 (frag 1109:36@0+) (ttl 255, len 56)
00:48:06.563167 192.168.1.3.smtp > 192.168.1.2.smtp: [no cksum] udp 28 (frag 1109:36@0+) (ttl 255, len 56)
00:48:06.583167 192.168.1.3.smtp > 192.168.1.2.smtp: [no cksum] udp 28 (frag 1109:36@0+) (ttl 255, len 56)

Kevin Mitnick

- August 6, 1963
- 12: bypass the bus punchcard system
- 1979 Hack DEC to view VMS source code
- Hacking of Motorola, NEC, Nokia, Sun Microsystems and Fujitsu Siemens systems
- Arrested in 1995

U.S. Department of Justice
United States Marshals Service

WANTED
BY U.S. MARSHALS

NOTICE TO ARRESTING AGENCY: Before arrest, validate warrant through National Crime Information Center (NCIC).
United States Marshals Service NCIC entry number: (NIC/ W721460021).

NAME: .....MITNICK, KEVIN DAVID
AKS (S): .....MITNIK, KEVIN DAVID
MERRILL, BRIAN ALLEN

DESCRIPTION:
Sex: .....MALE
Race: .....WHITE
Place of Birth: .....VAN NUYS, CALIFORNIA
Date(s) of Birth: 08/06/63; 10/18/70
Height: .....5'11"
Weight: 190
Eyes: .....BLUE
Hair: .....BROWN
Skintone: .....LIGHT
Scars, Marks, Tattoos: .....NONE KNOWN
Social Security Number (s): 550-39-5695
NCIC Fingerprint Classification: ...DOPM2GPM13DIPM19PM09

ADDRESS AND LOCALE: KNOWN TO RESIDE IN THE SAN FERNANDO VALLEY AREA OF CALIFORNIA AND LAS VEGAS, NEVADA

WANTED FOR: VIOLATION OF SUPERVISED RELEASE
ORIGINAL CHARGES: POSSESSION UNAUTHORIZED ACCESS DEVICE; COMPUTER FRAUD
Warrant Issued: CENTRAL DISTRICT OF CALIFORNIA
Warrant Number: 9312-1112-0154-C

DATE WARRANT ISSUED: NOVEMBER 10, 1992

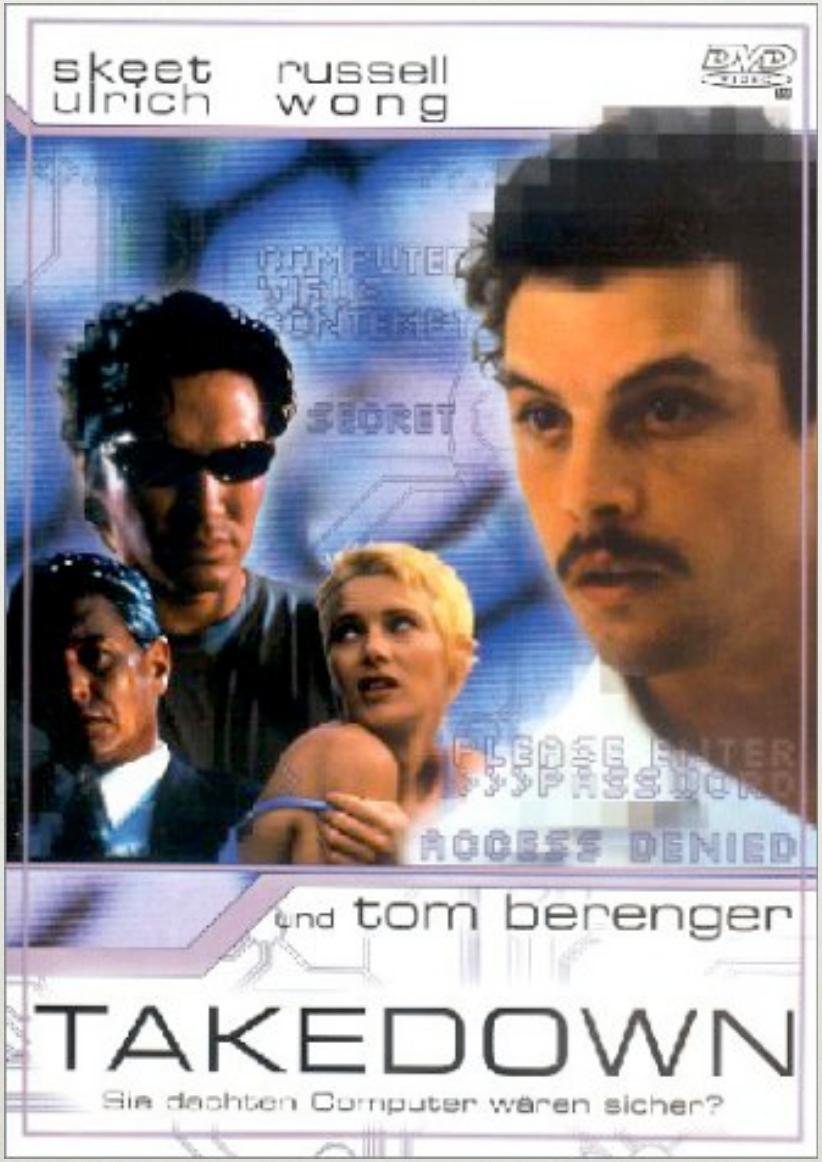
MISCELLANEOUS INFORMATION: SUBJECT SUFFERS FROM A WEIGHT PROBLEM AND MAY HAVE EXPERIENCED WEIGHT GAIN OR WEIGHT LOSS
VEHICLE/TAG INFORMATION: NONE KNOWN OFTEN USES PUBLIC TRANSPORTATION

If arrested or whereabouts known, notify the local United States Marshals Office, (Telephone: 213-894-2485).
If no answer, call United States Marshals Service Communications Center in McLean Virginia.
Telephone (800)336-0102; (24 hour telephone contact) NLETS access code is VAUSM0000.

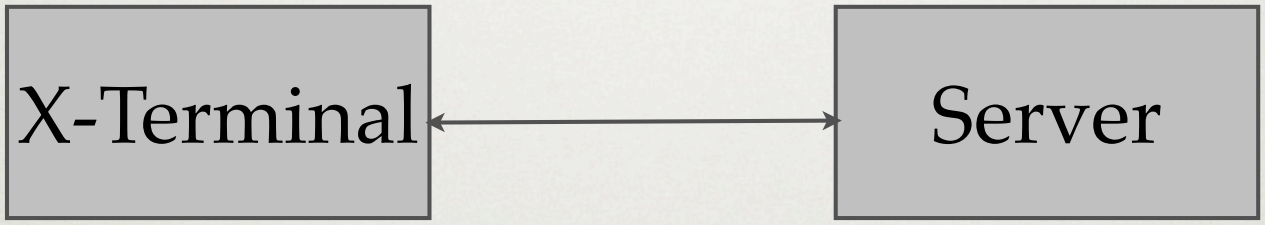
PRIOR EDITIONS ARE OBSOLETE AND NOT TO BE USED

November 1992

Mitnick vs Shimomura


skeet ulrich
russell wong
SECRET
ACCESS DENIED
and tom berenger
TAKEDOWN
Sie dachten Computer wären sicher?

Setup



Step 1 12/25/94

14:09:32 toad.com# finger -l @target

14:10:21 toad.com# finger -l @server

14:10:50 toad.com# finger -l root@server

14:11:07 toad.com# finger -l @x-terminal

14:11:38 toad.com# showmount -e x-terminal

14:11:49 toad.com# rpcinfo -p x-terminal

14:12:05 toad.com# finger -l root@x-terminal

Step 2 flood

14:18:24.382841 130.92.6.97.619 > server.login: S 1382726979:1382726979(0) win 4096

14:18:24.443309 130.92.6.97.620 > server.login: S 1382726980:1382726980(0) win 4096

14:18:24.643249 130.92.6.97.621 > server.login: S 1382726981:1382726981(0) win 4096

14:18:24.906546 130.92.6.97.622 > server.login: S 1382726982:1382726982(0) win 4096

14:18:24.963768 130.92.6.97.623 > server.login: S 1382726983:1382726983(0) win 4096

14:18:25.022853 130.92.6.97.624 > server.login: S 1382726984:1382726984(0) win 4096

14:18:25.153536 130.92.6.97.625 > server.login: S 1382726985:1382726985(0) win 4096

14:18:25.400869 130.92.6.97.626 > server.login: S 1382726986:1382726986(0) win 4096

14:18:25.483127 130.92.6.97.627 > server.login: S 1382726987:1382726987(0) win 4096

14:18:25.599582 130.92.6.97.628 > server.login: S 1382726988:1382726988(0) win 4096

14:18:25.653131 130.92.6.97.629 > server.login: S 1382726989:1382726989(0) win 4096

Step 3 prediction

14:18:34.375641 x-terminal.shell > apollo.it.luc.edu.984: S 2023872000:2023872000(0) ack 1382727007 win 4096

14:18:34.452830 apollo.it.luc.edu.984 > x-terminal.shell: R 1382727007:1382727007(0) win 0

14:18:34.714996 apollo.it.luc.edu.983 > x-terminal.shell: S 1382727007:1382727007(0) win 4096

14:18:34.885071 x-terminal.shell > apollo.it.luc.edu.983: S 2024000000:2024000000(0) ack 1382727008 win 4096

14:18:34.962030 apollo.it.luc.edu.983 > x-terminal.shell: R 1382727008:1382727008(0) win 0

14:18:35.225869 apollo.it.luc.edu.982 > x-terminal.shell: S 1382727008:1382727008(0) win 4096

14:18:35.395723 x-terminal.shell > apollo.it.luc.edu.982: S 2024128000:2024128000(0) ack 1382727009 win 4096

14:18:35.472150 apollo.it.luc.edu.982 > x-terminal.shell: R 1382727009:1382727009(0) win 0

14:18:35.735077 apollo.it.luc.edu.981 > x-terminal.shell: S 1382727009:1382727009(0) win 4096

14:18:35.905684 x-terminal.shell > apollo.it.luc.edu.981: S 2024256000:2024256000(0) ack 1382727010 win 4096

14:18:35.983078 apollo.it.luc.edu.981 > x-terminal.shell: R 1382727010:1382727010(0) win 0

Step 4 blind spoofing

14:18:36.245045 server.login > x-terminal.shell: S 1382727010:1382727010(0) win 4096

14:18:36.755522 server.login > x-terminal.shell: . ack 2024384001 win 4096

Insertion

14:18:37.265404 server.login > x-terminal.shell: P 0:2(2) ack 1 win 4096

14:18:37.775872 server.login > x-terminal.shell: P 2:7(5) ack 1 win 4096

14:18:38.287404 server.login > x-terminal.shell: P 7:32(25) ack 1 win 4096

14:18:37 server# rsh x-terminal "echo + + >>/.rhosts"

Closing up

14:18:41.347003 server.login > x-terminal.shell: . ack 2 win 4096

14:18:42.255978 server.login > x-terminal.shell: . ack 3 win 4096

14:18:43.165874 server.login > x-terminal.shell: F 32:32(0) ack 3 win 4096

14:18:52.179922 server.login > x-terminal.shell: R 1382727043:1382727043(0) win 4096

14:18:52.236452 server.login > x-terminal.shell: R 1382727044:1382727044(0) win 4096

Cleaning up

14:18:52.298431 130.92.6.97.600 > server.login: R 1382726960:1382726960(0) win 4096

14:18:52.363877 130.92.6.97.601 > server.login: R 1382726961:1382726961(0) win 4096

14:18:52.416916 130.92.6.97.602 > server.login: R 1382726962:1382726962(0) win 4096

14:18:52.476873 130.92.6.97.603 > server.login: R 1382726963:1382726963(0) win 4096

14:18:52.536573 130.92.6.97.604 > server.login: R 1382726964:1382726964(0) win 4096

14:18:52.600899 130.92.6.97.605 > server.login: R 1382726965:1382726965(0) win 4096

14:18:52.660231 130.92.6.97.606 > server.login: R 1382726966:1382726966(0) win 4096

14:18:52.717495 130.92.6.97.607 > server.login: R 1382726967:1382726967(0) win 4096

14:18:52.776502 130.92.6.97.608 > server.login: R 1382726968:1382726968(0) win 4096

Terminal hijhacking

x-terminal% modstat

Id Type Loadaddr Size B-major C-major Sysnum Mod Name

1 Pdrv ff050000 1000 59. tap/tap-2.01 alpha

x-terminal% ls -l /dev/tap