

Revolutionizing Active Operating System Fingerprinting The Present & Future of Xprobe2

Ofir Arkin

ofir@sys-security.com

Founder



Agenda

- **Introduction**
 - Ofir Arkin
 - The Xprobe2 project
- **The Problems/Issues with Active OS fingerprinting**
- **The Remedy I**
- **The Future of Active Operating System Fingerprinting**
- **Demo – Xprobe2 v0.2**
- **Future Work**
- **Questions**

Ofir Arkin

- **CISO of an International Telephone Carrier**
- **Founder, The Sys-Security Group**
- **Computer Security Researcher**
 - **Etherleak**: Ethernet frame padding information leakage (with Josh Anderson)
 - **IP Telephony Security** (Security risk factors with IP Telephony based networks, numerous advisories and white papers)
 - **ICMP Usage In Scanning** (Security related issues with the ICMP protocol)
 - **Information Warfare** (trace-back)
- **Member, the HoneyNet project**

Xprobe/Xprobe2 Project

- **Open Source Project**
- **Developers**
 - Ofir Arkin
 - Fyodor Yarochkin
 - Meder Kydyraliev
- **Xprobe2 is a remote active operating system fingerprinting tool**
- **Xprobe2 presents an alternative to other remote active operating system fingerprinting tools**
- **Voted one of the top 75 security tools (at the top 50)**

Xprobe/Xprobe2 Project History

- Initial **alpha release** (Xprobe v0.0.1) at the **Blackhat briefings USA 2001**, June 2001
 - Relying on **ICMP-based active OS fingerprinting methods** found by Ofir Arkin (specified in the “**ICMP Usage In Scanning**” research paper)
 - **Static** decision tree
 - Was not signature-based
 - **It was only a mission statement** - Alpha – **limited in functionality**

Xprobe/Xprobe2 Project History

- **Xprobe2 0.1 beta** was released last year at **Defcon X**:
 - Based on a **signature database**
 - **First open source fingerprinting tool to use fuzzy logic** matching algorithm between **probe results to a signature database** (strict signature matching suffers from a number of accuracy issues)
 - Xprobe2 0.1 beta was using only **ICMP-based fingerprinting tests**

Xprobe/Xprobe2 Project History

- **Xprobe2 0.1 release (April 2003)**
 - Sends RFC compliant packets
 - A lot of bug fixes
 - Support for IP ID = SENT fingerprinting method
 - Major signature DB update
 - Documentation on how to add your own signatures

Issues with Active OS fingerprinting

The Issue with Hardware-based Devices

- When fingerprinting **operating systems** we fingerprint the way an **operating system (the software)** reacts to different fingerprinting probes a tool uses
- With a **hardware based device** we fingerprint the way a **device's firmware** reacts to the different fingerprinting probes
- Hardware based devices of **the same manufacture** will usually **run the same**, or a **slightly different, firmware** (or software) **version**
- It will be either **one version for all**, or a **particular version for a particular functionality**

The Issue with Hardware-based Devices

- Example: A **Cisco 7200 router** will be fingerprinted exactly the same as **Cisco's Aironet 1100/1200 wireless access points**
- It is **not possible** to distinguish between different hardware based products, and their functionality, **manufactured** by **Cisco** and using **IOS**, using traditional active operating system fingerprinting methods
- It is possible to identify these devices as manufactured by Cisco and using IOS
- It is also possible to **divide these devices** into groups according to **fingerprints differences with the IOS versions** they are using, but not to **discover** their **functionality**

The Issue with Hardware-based Devices

- Another example is the Foundry Network's Net/Fast/Big Iron family
- If the designers of the fingerprinting tool of your choice failed to understand these issues, the results received, which are **based on a corrupted database**, will be **unreliable**

The Way Probe Results Are Being Matched

- **A Strict Signature Matching based Tool**
 - Would search for a 100% match between the **received results** and the tool's **signature database**
 - If a 100% match is not found, than **no match is found** and the run fails
 - Extremely **sensitive to environmental affects** on the probed target, and on the network which the probed target resides on

The Way Probe Results Are Being Matched

- **Fuzzy Logic**

- Xprobe2

- First to implement a statistical analysis based mathematical algorithm to provide with a best effort match between probe results, received from a targeted system, to a signature database

- Uses one of the simplest forms of Optical Character Recognition (OCR), by utilizing a matrix based fingerprints matching based on statistical calculation of scores for each test performed

- Using a fuzzy logic approach, provides better resistance against environmental affects which might take their toll on a target system and on probe packets

The Way Probe Results Are Being Matched

■ Fuzzy Logic (continue)

- The **quality of the results** produced with an active operating system fingerprinting tool using a fuzzy logic approach would be **higher**
- This is if the tool will **not suffer from design flaws**, and will **use a large base of fingerprinting tests**
- The fuzzy logic implementation with Xprobe2 **still misses** the ability to **assign different weights to different fingerprinting tests**
- This ability is required since **some fingerprinting tests** should have **bigger impact** over the overall fingerprinting results

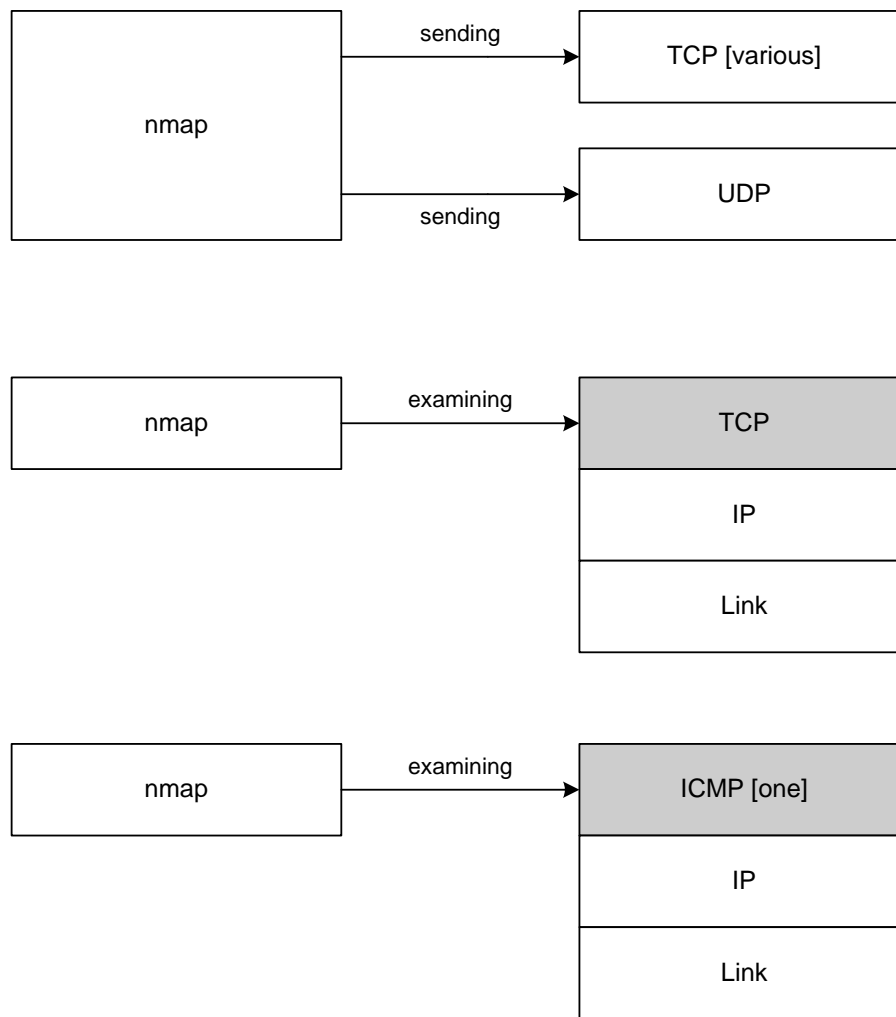
The Use of a Fixed Number of Fingerprinting Tests

- A fixed number of fingerprinting tests is used
- A fixed number of parameters are examined
- In theory: **Possible matches** = **tests** X **parameters examines** X **parameters permutations**
- Although the overall number of possible matches is currently much higher than the number of the current available network elements, **certain test classes cannot deliver the expected results** and provide with a clear distinction between different network elements

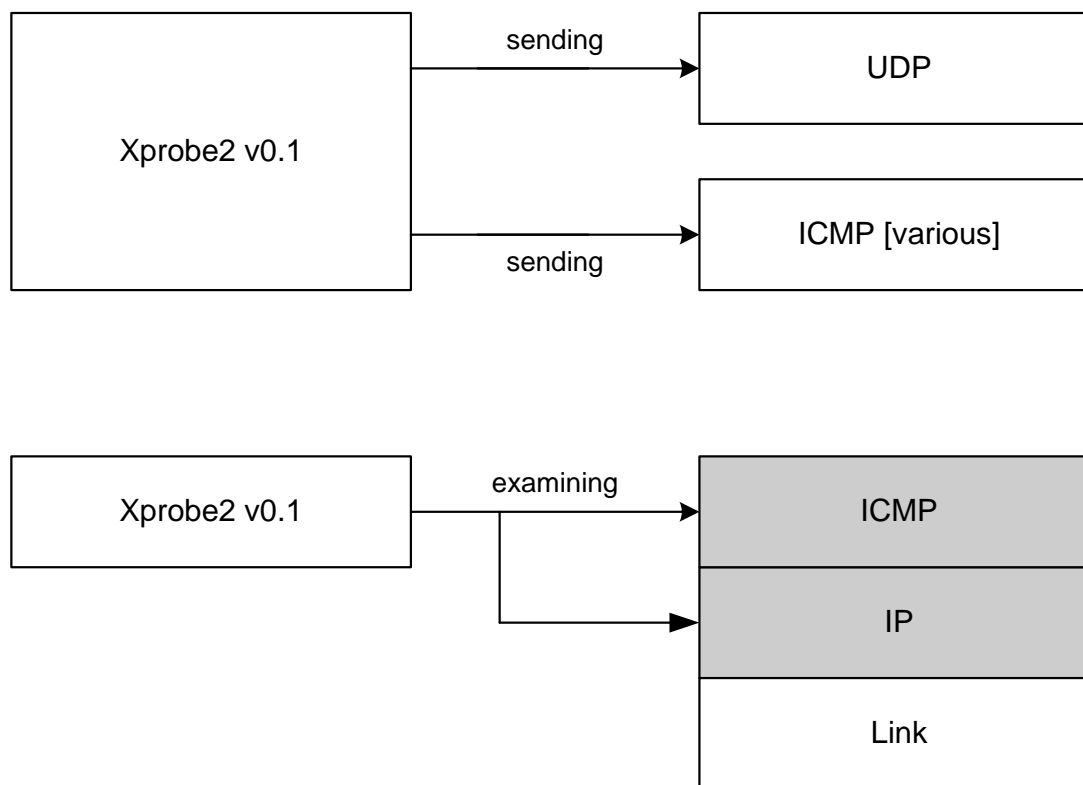
The Use of a Fixed Number of Fingerprinting Tests

- A better tool for active OS fingerprinting would be required to **utilize fingerprinting tests**, which would **examine many parameter values** with the probe's reply
- These parameter values **would need to be different among many network elements**
- Therefore a number of this kind of tests is required to be used in order **to achieve a broader distinction between different network elements**
- It suggests that the **usage** of more **parameter rich fingerprinting tests** with an active operating fingerprinting tool will provide **better overall results**

The Use of a Certain Fingerprinting Niche



The Use of a Certain Fingerprinting Niche



The Use of a Certain Fingerprinting Niche

- This fixation brings into light the **inability** of such tools to deal with situations where the fingerprinting tests they use **do not yield an adequate result** about a **certain operating system** or even **a class of operating systems**

No Changes Are Made To the TCP/IP Stacks Of New Versions Of Operating Systems

- The behavior of the TCP/IP stack of **newly released operating systems** hardly changes compared to an older version of the same operating system, or
- Changes made to a newly released operating system's TCP/IP stack **might affect a certain protocol behavior only**
- The result? **Inability** of some active operating system fingerprinting tools **to distinguish between different versions of the same operating system** or even between a class of the same operating system family

No Changes Are Made To the TCP/IP Stacks Of New Versions Of Operating Systems

```
[root@angelfire NG]# xprobe2 -v x.x.x.x
```

```
XProbe2 v.0.1 Copyright (c) 2002-2003 fygrave@tigerteam.net, ofir@sys-security.com, meder@areopag.net
```

```
[+] Target is x.x.x.x
[+] Loading modules.
[+] Following modules are loaded:
    [x][1] ICMP echo (ping)
    [x][2] TTL distance
    [x][3] ICMP echo
    [x][4] ICMP Timestamp
    [x][5] ICMP Address
    [x][6] ICMP Info Request
    [x][7] ICMP port unreach
[+] 7 modules registered
[+] Initializing scan engine
[+] Running scan engine
[+] Host: x.x.x.x is up (Guess probability: 100%)
[+] Target: x.x.x.x is alive
[+] Primary guess:
[+] Host x.x.x.x Running OS: "Sun Solaris 5 (SunOS 2.5)" (Guess probability: 100%)
[+] Other guesses:
[+] Host x.x.x.x Running OS: "Sun Solaris 6 (SunOS 2.6)" (Guess probability: 100%)
[+] Host x.x.x.x Running OS: "Sun Solaris 7 (SunOS 2.7)" (Guess probability: 100%)
[+] Host x.x.x.x Running OS: "Sun Solaris 8 (SunOS 2.8)" (Guess probability: 100%)
[+] Host x.x.x.x Running OS: "Sun Solaris 9 (SunOS 2.9)" (Guess probability: 100%)
```

No Changes Are Made To the TCP/IP Stacks Of New Versions Of Operating Systems

```
[root@angelfire NG]# /usr/local/bin/nmap -sT -O x.x.x.x
```

```
Starting nmap 3.28 ( www.insecure.org/nmap/ ) at 2003-06-18 19:14 IDT
```

```
Interesting ports on x.x.x.x:
```

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

Port	State	Service
21/tcp	filtered	ftp
22/tcp	filtered	ssh
25/tcp	open	smtp
80/tcp	open	http
135/tcp	open	loc-srv
139/tcp	open	netbios-ssn
443/tcp	open	https
465/tcp	open	smtps
1029/tcp	open	ms-lsa
1433/tcp	open	ms-sql-s
2301/tcp	open	compaqdiag
5555/tcp	open	freeciv
5800/tcp	open	vnc-http
5900/tcp	open	vnc
6000/tcp	filtered	X11

```
Remote operating system guess: Windows NT 3.51 SP5, NT4 or 95/98/98SE
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 3.334 seconds
```

The Inability to Determine the Exact Software Service Pack

- Traditional active operating system fingerprinting tools are usually **unable to identify the installation of software service packs on a targeted machine**
- For example, they will identify a target machine runs Microsoft Windows 2000, but will not be able to determine which service pack version is installed (if any at all)

Some Fingerprinting Tests May Have Bigger Impact on the Overall Results

- Some fingerprinting tests may have bigger **impact** on the **overall accuracy** of the test results compared to other fingerprinting tests used
- If these tests **fail**, for some reason, **the quality of the produced results will be lowered significantly**, especially with tools using strict signature matching
- The affect of a failure of a mark key test on the results a tool using a **fuzzy logic** approach produces will be **less significant**, although it might take its toll as well

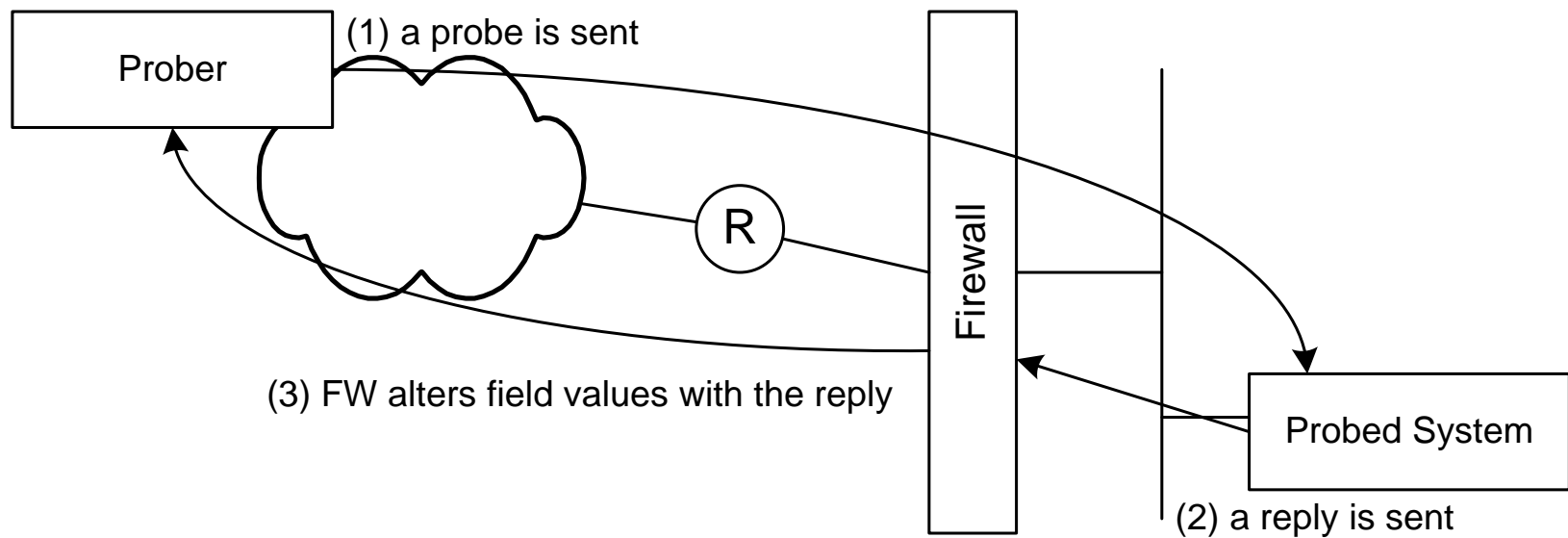
Some Fingerprinting Tests May Have Bigger Impact on the Overall Results

```
spanion:~ # xprobe2 -v x.x.x.x
XProbe2 v.0.1 Copyright (c) 2002-2003 fygrave@tigerteam.net,
ofir@sys-security.com, meder@areopag.net
[+] Target is x.x.x.x
[+] Loading modules.
[+] Following modules are loaded:
...
[+] 7 modules registered
[+] Initializing scan engine
[+] Running scan engine
[+] Host: x.x.x.x is up (Guess probability: 100%)
[+] Target: x.x.x.x is alive
[+] Primary guess:
[+] Host x.x.x.x Running OS: "Microsoft Windows XP Professional / XP
Professional SP1" (Guess probability: 100%)
[+] Other guesses:
[+] Host x.x.x.x Running OS: "Microsoft Windows
2000/2000SP1/2000SP2/2000SP3" (Guess probability: 100%)
```

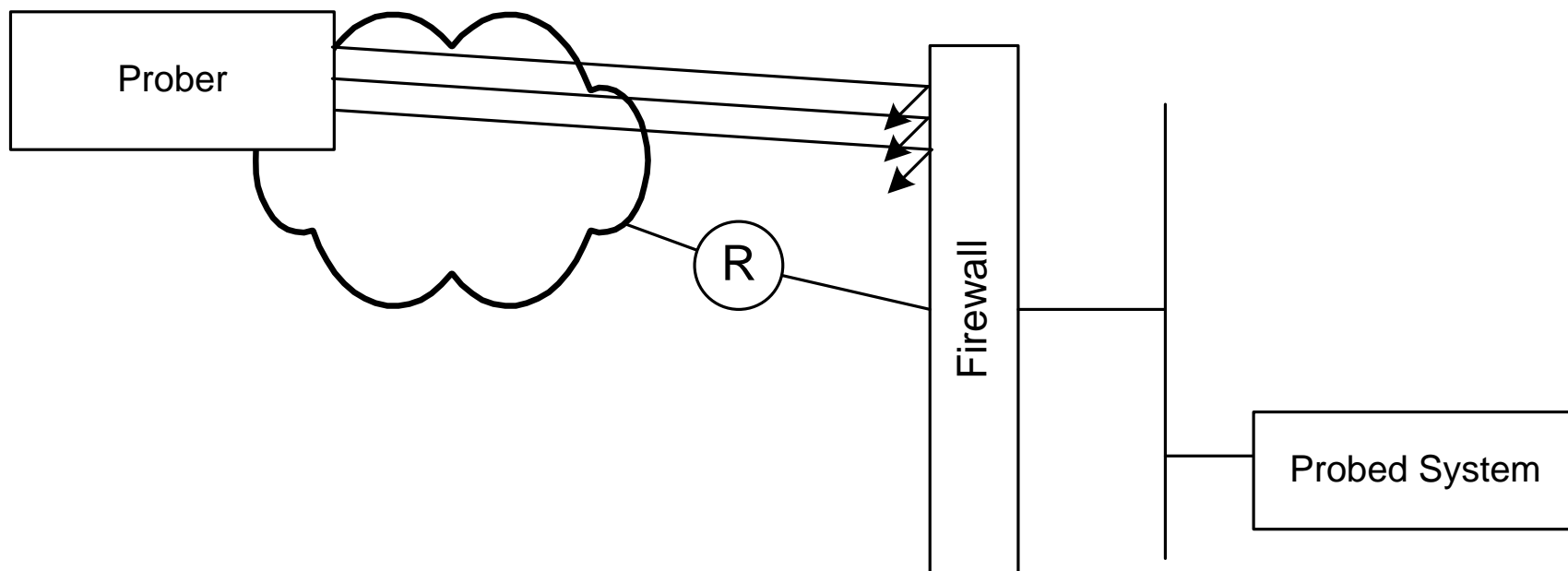
Some Fingerprinting Tests May Have Bigger Impact on the Overall Results

```
spanion:~ # xprobe2 -v -D 1 -D 2 -D 3 x.x.x.x
XProbe2 v.0.1 Copyright (c) 2002-2003 fygrave@tigerteam.net,
ofir@sys-security.com, meder@areopag.net
[+] Target is x.x.x.x
[+] Loading modules.
[+] Following modules are loaded:
      [x][1] ICMP Timestamp
      [x][2] ICMP Address
      [x][3] ICMP Info Request
      [x][4] ICMP port unreachable
[+] 4 modules registered
[+] Initializing scan engine
[+] Running scan engine
[+] All alive tests disabled
[+] Target: x.x.x.x is alive
[+] Primary guess:
[+] Host x.x.x.x Running OS: "Microsoft Windows XP Professional /
XP Professional SP1" (Guess probability: 100%)
[+] Other guesses:
[+] Host x.x.x.x Running OS: "Microsoft Windows
2000/2000SP1/2000SP2/2000SP3" (Guess probability: 100%)
[+] Host x.x.x.x Running OS: "Microsoft Windows ME" (Guess probability:
100%)
```

Different Networking Devices May Alter A Packet's Field Value



A Firewallled Target Systems



A Firewallled Target Systems

- Probed systems **might be firewallled**
- If a remote active operating system fingerprinting tool relies on sending and/or receiving of particular packet types and those packets are **dropped** by a firewall protecting the target system(s) chances are that the **quality of the results** would be **degraded** to the point **false results** or **no results at all** will be produced

The Use of Malformed Packets

- If malformed packets are used, a filtering device **may drop** the packets, if the filtering device analyzes packets for non-legitimate content
- Therefore **the quality** of the results produced by utilizing a fingerprinting tests relying on malformed packets will be **degraded** and in some cases even **fail**
- Malformed packets may have another affect, they **might cause some TCP/IP stacks to crash**

A TCP/IP Stack's Behavior Might Be Altered

- Some **characteristics** of a TCP/IP stack's behavior can be **altered** by a machine's system administrator:
 - **Tunable parameters** of the TCP/IP stack might be changed e.g. the `sysctl` command on the various *BSDs, the `ndd` command on Sun Solaris, etc.
 - Numerous **patches exist** for some open source operating system's kernels that **alter the way** the particular operating system's TCP/IP stack responds to certain packets
- If a remote active operating system fingerprinting tool is using some of the TCP/IP based parameters that can be altered as part of its fingerprinting test, the quality of the results would be **affected** and **questionable** when these parameter values will be altered

The Quality of the Signature Database

- The quality of the results produced by an active operating system fingerprinting tool is not only a factor of programming and terrain
- It is much **affected** from **the way the signature database of the tool was and is built**
- If **signatures submitted** to the database were and are **obtained in a wrongfully manner** than the signature database should be regarded as **corrupt**
- The results produced by the tool **will not be accurate**

The Quality of the Signature Database

- One can find false information quite easily in signature databases of some tools
- For example: `nmap` has a TCP “EOL” in the middle of a TCP Options list of some fingerprints

The Inability to Identify the Underlying Architecture Platform

- Usually, active operating system fingerprinting tools will identify the operating system of a network node, but **not its underlying platform**
- The knowledge about the underlying platform is **extremely important** for tools performing vulnerability assessment, network inventory, etc., which rely on the results of the active operating system fingerprinting tool (i.e. nessus)

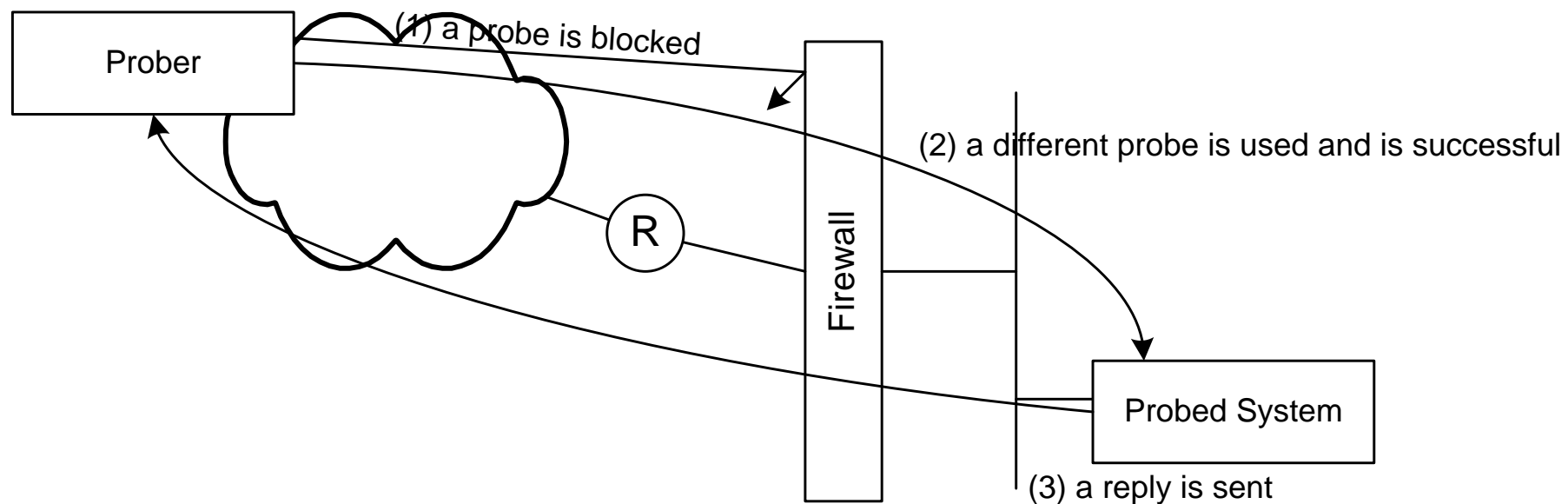
The Inability to Scale

- An active operating system fingerprinting tool should have the **ability to scan large networks**
- Must **not use many packets** to do so
- For any router and switch there is an **upper limit** to the number of **packets per second it can process**. Beyond that limit, some packets will be **dropped**, but more important, **the router/switch might suffer from a denial of service condition**
- Therefore it is very important to **balance the scan rate** with the network and network elements abilities

Inability to Control the Fingerprinting Modules to Be Executed

- **The intent of the OS fingerprinting attempt is important**
 - If the intent of the scan is **malicious**, than it might be more beneficial for the malicious attacker to use **minimum packets to achieve maximum results**
 - If the reason is not malicious, usually, there is **less importance** for being stealth, still needing to achieve **maximum results**. Although we would still like to send minimum packets to probe each targeted system, eliminating the possibility of overwhelming the network and affecting its performance, we **can use whatever fingerprinting tests we have for our disposal**
- **One needs to control the fingerprinting tests a certain tool has to offer according to her/his needs**
- **Furthermore, we would like an active OS fingerprinting tool to be able to detect certain scanning conditions and to react, by switching scanning tactics**

Inability to Control the Fingerprinting Modules to Be Used



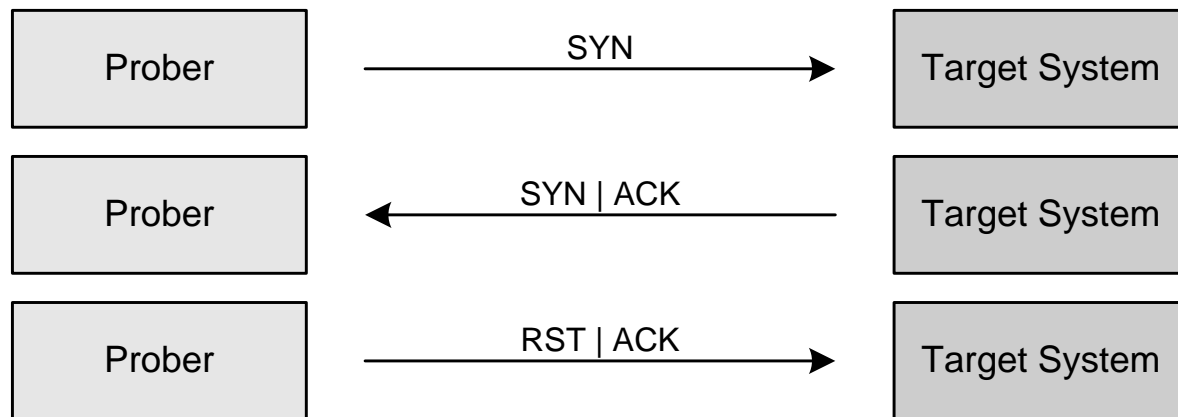
The Remedy I

Added Functionality to Xprobe2

The Use of Best of Breed TCP/IP Stack Fingerprinting Techniques

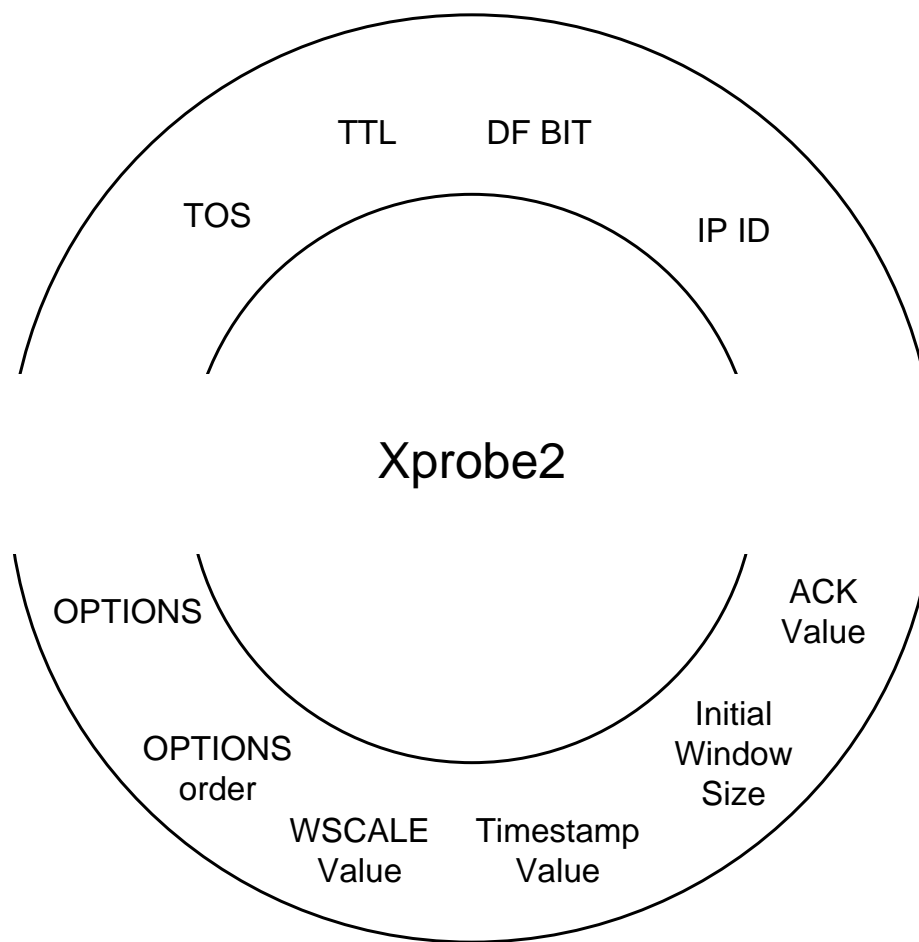
- Searched for a TCP-based fingerprinting test with maximum impact on the overall fingerprinting results
- A test which will use **as much parameters as possible** and provide with a **real added value**
- We have decided on adding a **TCP module based on the TCP 3-way handshake**

The Use of Best of Breed TCP/IP Stack Fingerprinting Techniques



- The **parameters** with the SYN request **resembles** the parameters used with a **Linux telnet** request
- Unlike other tools, which use a similar module, Xprobe2 **examines** parameters found in the **IP** and **TCP** layers

The Use of Best of Breed TCP/IP Stack Fingerprinting Techniques



The Use of Best of Breed TCP/IP Stack Fingerprinting Techniques – Example (without)

```
spanion:~/tmp/xprobe2-demo/src # ./xprobe2 -v -c ../etc/xprobe2.conf -D
8 -p TCP:22:open 192.168.0.203
Xprobe2 v.0.1 Copyright (c) 2002-2003 fygrave@tigerteam.net,
ofir@sys-security.com, meder@areopag.net
...
[+] Following modules are loaded:
    [x][1] ICMP echo (ping)
    [x][2] TTL distance
    [x][3] ICMP echo
    [x][4] ICMP Timestamp
    [x][5] ICMP Address
    [x][6] ICMP Info Request
    [x][7] ICMP port unreachable
[+] 7 modules registered
[+] Initializing scan engine
[+] Running scan engine
[+] Host: 192.168.0.203 is up (Guess probability: 100%)
[+] Target: 192.168.0.203 is alive
[+] Primary guess:
[+] Host 192.168.0.203 Running OS: "FreeBSD 4.4" (Guess probability: 100%)
[+] Other guesses:
[+] Host 192.168.0.203 Running OS: "FreeBSD 5.1" (Guess probability: 100%)
[+] Host 192.168.0.203 Running OS: "FreeBSD 5.0" (Guess probability: 100%)
[+] Host 192.168.0.203 Running OS: "FreeBSD 4.8" (Guess probability: 100%)
[+] Host 192.168.0.203 Running OS: "FreeBSD 4.7" (Guess probability: 100%)
[+] Host 192.168.0.203 Running OS: "FreeBSD 4.6.2" (Guess probability: 100%)
[+] Host 192.168.0.203 Running OS: "FreeBSD 4.6" (Guess probability: 100%)
[+] Host 192.168.0.203 Running OS: "FreeBSD 4.5" (Guess probability: 100%)
...
```

The Use of Best of Breed TCP/IP Stack Fingerprinting Techniques – Example (with)

```
spanion:~/tmp/xprobe2-demo/src # ./xprobe2 -v -c ../etc/xprobe2.conf -p
TCP:22:open 192.168.0.203
Xprobe2 v.0.1 Copyright (c) 2002-2003 fygrave@tigerteam.net,
ofir@sys-security.com, meder@areopag.net
...
[+] Following modules are loaded:
    [x][1] ICMP echo (ping)
    [x][2] TTL distance
    [x][3] ICMP echo
    [x][4] ICMP Timestamp
    [x][5] ICMP Address
    [x][6] ICMP Info Request
    [x][7] ICMP port unreachable
    [x][8] TCP Handshake
[+] 8 modules registered
[+] Initializing scan engine
[+] Running scan engine
[+] Host: 192.168.0.203 is up (Guess probability: 100%)
[+] Target: 192.168.0.203 is alive
[+] Primary guess:
[+] Host 192.168.0.203 Running OS: "FreeBSD 4.4" (Guess probability:
100%)
[+] Other guesses:
[+] Host 192.168.0.203 Running OS: "FreeBSD 4.8" (Guess probability: 96%)
[+] Host 192.168.0.203 Running OS: "FreeBSD 4.7" (Guess probability: 96%)
[+] Host 192.168.0.203 Running OS: "FreeBSD 5.1" (Guess probability: 93%)
[+] Host 192.168.0.203 Running OS: "FreeBSD 5.0" (Guess probability: 93%)
```

The Use of Best of Breed TCP/IP Stack Fingerprinting Techniques

- Combined with Xprobe2's other fingerprinting modules, the TCP handshake module greatly **enhance** Xprobe2's abilities and its overall **accuracy**

Adding a Port Scanner

- The success of executing some of Xprobe2's fingerprinting modules **depends on successfully probing** an **open TCP port** and a **closed UDP port**
- Therefore we have implemented a port scanner module as an **independent module** to Xprobe2
- By default Xprobe2 **does not tie** the port scanner module with its fingerprinting modules and therefore it maintains the minimal usage of packets to discover a targeted system's underlying operating system

Adding a Port Scanner, Usage

```
[root@fremont src]# ./xprobe2 -v -c ../etc/xprobe2.conf -t 1 -s 5 -P -T 20-40,80  
x.x.x.x
```

```
Xprobe2 v.0.1 Copyright (c) 2002-2003 fygrave@tigerteam.net,  
ofir@sys-security.com, meder@areopag.net
```

```
[+] Target is x.x.x.x  
[+] Loading modules.  
[+] Following modules are loaded:  
    [x][1] ICMP echo (ping)  
    [x][2] TTL distance  
    [x][3] ICMP echo  
    [x][4] ICMP Timestamp  
    [x][5] ICMP Address  
    [x][6] ICMP Info Request  
    [x][7] ICMP port unreach  
    [x][8] TCP Handshake  
    [x][9] Portscanner  
[+] 9 modules registered
```

```
[+] Initializing scan engine
[+] Running scan engine
[+] Host: x.x.x.x is up (Guess probability: 100%)
[+] Target: x.x.x.x is alive rtt: 1
[+] Portscan results for x.x.x.x:
[+] Stats:
[+] TCP: 4 - open, 18 - closed, 0 - filtered
[+] UDP: 0 - open, 0 - closed, 0 - filtered
[+] Portscan took 0.95 seconds.
[+] Details:
[+] Proto      Port Num.      State          Serv. Name
[+] TCP        21             open          ftp
[+] TCP        23             open          telnet
[+] TCP        37             open          time
[+] TCP        80             open          http
[+] Other ports are in closed state.
[+] Primary guess:
[+] Host x.x.x.x Running OS: "HP UX 11.0" (Guess probability: 100%)
[+] Other guesses:
[+] Host x.x.x.x Running OS: "HP UX 11.0i" (Guess probability: 96%)
[+] Host x.x.x.x Running OS: "Sun Solaris 9 (SunOS 2.9)" (Guess probability: 90%)
[+] Host x.x.x.x Running OS: "Sun Solaris 6 (SunOS 2.6)" (Guess probability: 87%)
[+] Host x.x.x.x Running OS: "Sun Solaris 7 (SunOS 2.7)" (Guess probability: 87%)
[+] Host x.x.x.x Running OS: "Sun Solaris 8 (SunOS 2.8)" (Guess probability: 87%)
[+] Host x.x.x.x Running OS: "OpenBSD 2.5" (Guess probability: 78%)
[+] Host x.x.x.x Running OS: "OpenBSD 2.9" (Guess probability: 78%)
[+] Host x.x.x.x Running OS: "NetBSD 1.4" (Guess probability: 78%)
[+] Host x.x.x.x Running OS: "NetBSD 1.4.1" (Guess probability: 78%)
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.
```

Adding a Port Scanner, Usage

- When the port scanner module is used, **knowledge** about opened TCP ports, and closed UDP ports **will be used as parameters for other modules**
- For example, the **port used for the TCP handshake module** will be one that **was already discovered as opened** by the port scanner
- The TCP handshake module used TCP port 21 to perform its fingerprints although the “-p” option was not used. The port was discovered as opened by the port scanner

Adding a Port Scanner, Controlling the Sending Stream

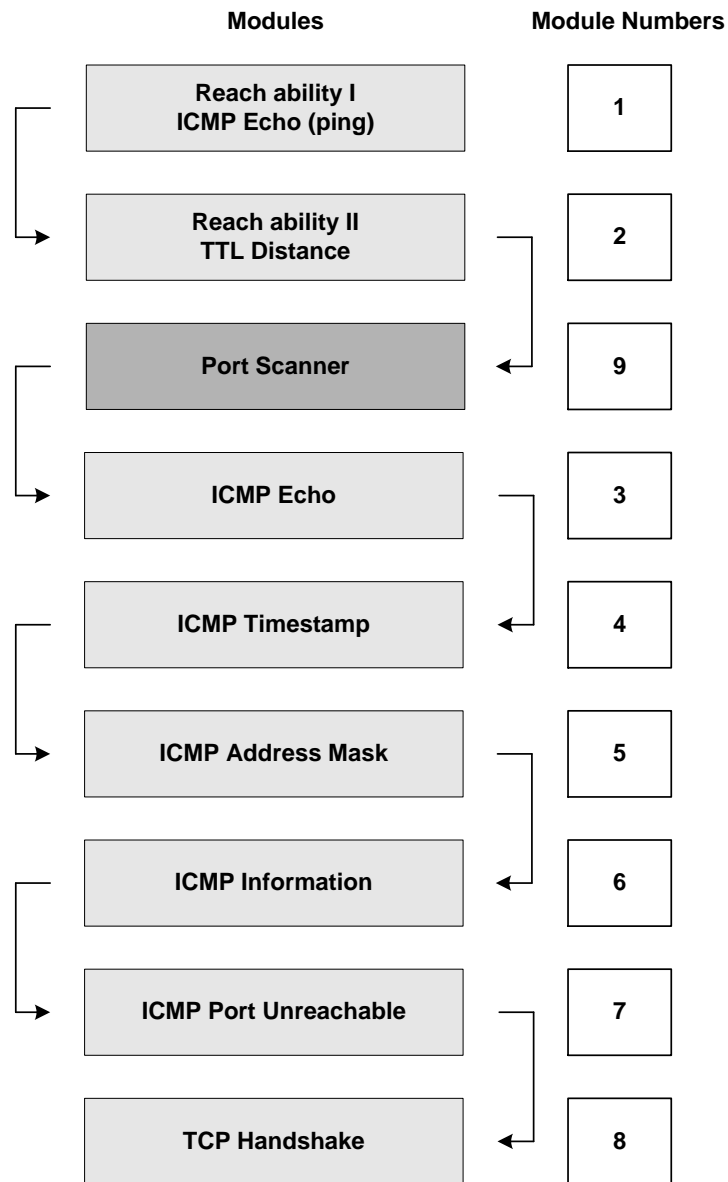
- A command line option, “-s”, was added to **control the sending stream** of the port scanner module
- The command line controls **the time interval between each SYN packet sent**
- The value given is represented in milliseconds
- Controlling the stream of the port scan is an important feature, allowing one to adjust the pace of the scan, not allowing denial of service conditions to be introduced

Enhanced Receive Timeout Specifications

- **There are three receiving timeouts:**
 - Xprobe2 takes use of the **ping discovery module** in order to calculate the **receiving timeout for its fingerprinting modules**. The timeout used is the round-trip time of the ICMP echo request and reply times two (**RTT*2**) measured in milliseconds
 - In order to allow a proper **receive timeout for the ICMP echo discovery module itself**, one can use the “-t” command line option and specify the receiving timeout in milliseconds
 - The port scanner’s receiving timeout is calculated differently:
 - $(\text{number of ports to scan} * ((\text{sending delay defined})+10)) + \text{RTT} * 2$
 - If the **number of received replies equals the number of ports scanned**, the port scanner module will **time out before the receiving timeout has been reached**

The Ability to Totally Control Modules and Features of Xprobe2 – Module Execution

- With the “-D” option one can specify which Xprobe2 modules **not to use**
- With the “-M” option one can specify which Xprobe2 modules to **use**



The Ability to Totally Control Modules and Features of Xprobe2 – Module Execution, Example

```
[root@fremont src]# ./xprobe2 -v -c ../etc/xprobe2.conf -D 2 -D 6 -t 1
-s 10 -P -T 21,22,23,25,80 x.x.x.x
Xprobe2 v.0.1 Copyright (c) 2002-2003 fygrave@tigerteam.net,
ofir@sys-security.com, meder@areopag.net
[+] Target is x.x.x.x
[+] Loading modules.
[+] Following modules are loaded:
    [x] [1] ICMP echo (ping)
    [x] [2] ICMP echo
    [x] [3] ICMP Timestamp
    [x] [4] ICMP Address
    [x] [5] ICMP port unreach
    [x] [6] TCP Handshake
    [x] [7] Portscanner
[+] 7 modules registered
[+] Initializing scan engine
[+] Running scan engine
[+] Host: x.x.x.x is up (Guess probability: 100%)
[+] Target: x.x.x.x is alive rtt: 1
[+] Portscan results for x.x.x.x:
[+] Stats:
[+]   TCP: 1 - open, 4 - closed, 0 - filtered
[+]   UDP: 0 - open, 0 - closed, 0 - filtered
[+]   Portscan took 0.37 seconds.
[+] Details:
[+]   Proto      Port Num.      State          Serv. Name
[+]   TCP        80             open           http
[+] Other ports are in closed state.
[+] Primary guess:
[+] Host x.x.x.x Running OS: "Microsoft Windows 2000 Server Service
Pack 3" (Guess probability: 100%)
```

The Ability to Totally Control Modules and Features of Xprobe2

```
[root@fremont src]# ./xprobe2 -v
Xprobe2 v.0.1 Copyright (c) 2002-2003 fygrave@tigerteam.net,
ofir@sys-security.com, meder@areopag.net
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 (miliseconds).
    -d <debuglv>     Specify debugging level.
    -D <modnum>      Disable module number <modnum>.
    -m <numofmatches> Specify number of matches to print.
    -P                Enable portscanning module
    -T <portspec>    Specify TCP port(s) to scan.
                        Example: -T21-23,53,110
    -U <portspec>    Specify UDP port(s) to scan.
    -f                force fixed round-trip time (-t opt).
```

Maintaining a Quality Signature Database

- Xprobe2's signature database is **tightly controlled**
- New signatures will be added to the database **if, and only if, we can verify them against a test system we control or have legitimate access to**
- We see the signature database issue as a **mandatory issue** for the success of the tool
- It is very easy to corrupt a signature database where it would lead to false and inaccurate results

Maintaining a Quality Signature Database

- **Xprobe2's signature database was re-built from scratch currently containing over 160 signatures**
 - The entire Linux Kernel 2.4.x branch
 - The entire Linux Kernel 2.2.x branch
 - FreeBSD 2.2.7, 2.2.8, 3.1, 3.2, 3.3, 3.4, 3.5.1, 4.0, 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.6.2, 4.7, 4.8, 5.0, 5.1
 - OpenBSD 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 3.1, 3.2, 3.3
 - NetBSD 1.6.1, 1.6, 1.5.3, 1.5.2, 1.5.1, 1.5, 1.4.3, 1.4.2, 1.4.1, 1.4, 1.3.3, 1.3.2, 1.3.1, 1.3
 - Microsoft Windows 2003-based Servers (unique ID), Windows XP and its service packs, Windows 2000 based Server (unique for SP3 and SP4) and Workstation and their service packs, NT4 Server and Workstation with the different service packs, ME, 98SE, 98, 95
 - Cisco IOS 12.2, 12.0, 11.3, 11.2, 11.1
 - And many more...

Parallel Scanning Support

- Support for parallel scanning is being added to Xprobe2 (currently in development)
- Xprobe2 will be able to scan class B networks fairly quickly
- Since Xprobe2 uses a minimal number of packages per host to discover the host's underlying operating system, its overall impact on the network is minimal compared to other active operating system fingerprinting tools

The Future of Active Operating System Fingerprinting

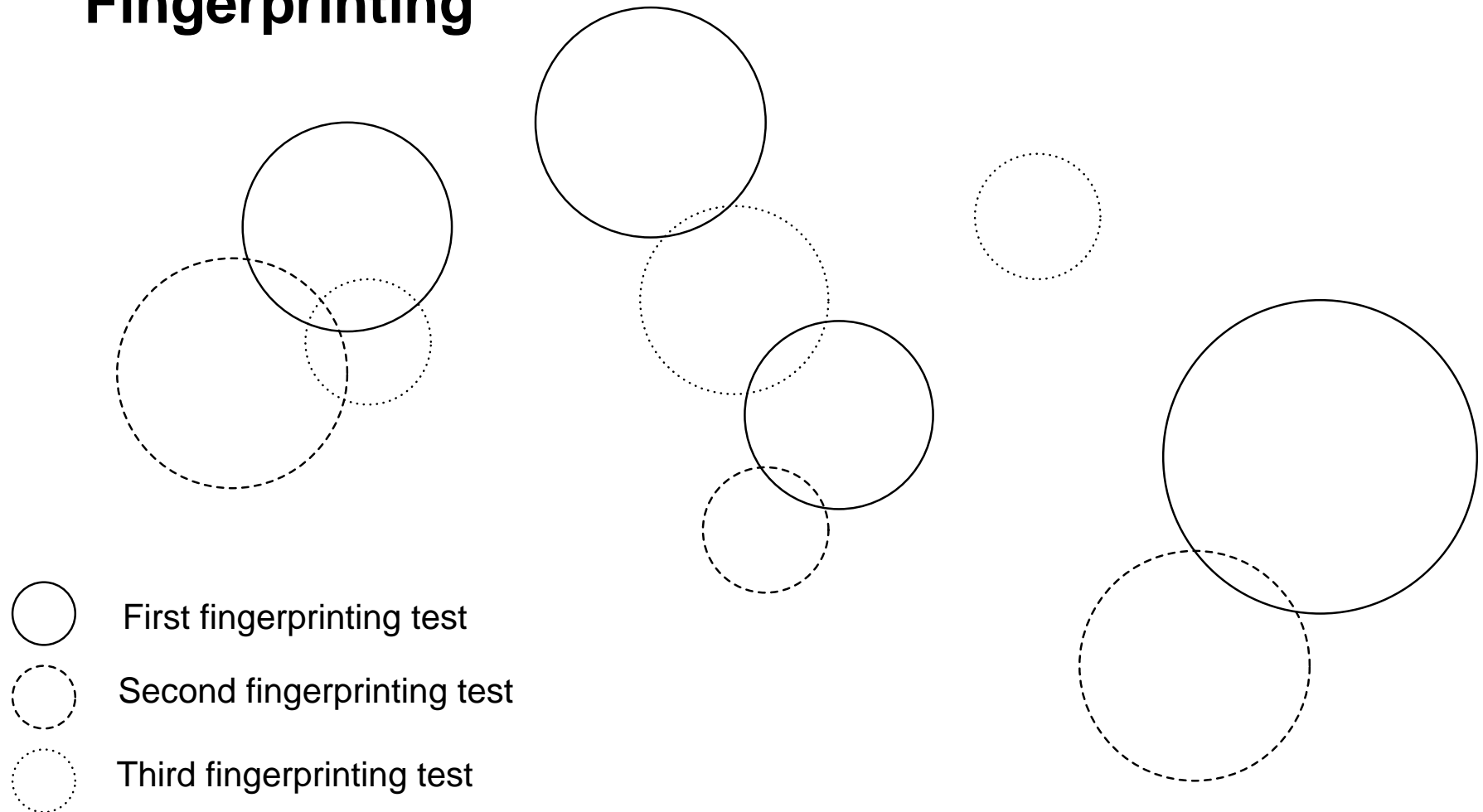
The Future of Active Operating System Fingerprinting

- Although new and/or existing TCP/IP stack based fingerprinting methods can, and will, be added to existing active operating system fingerprinting tools to **create a tool which uses the best of breed active operating system TCP/IP based stack fingerprinting methods**, **in the not so far future**, we would need to **start adding other means for actively identifying the underlying operating system of a remote targeted system**

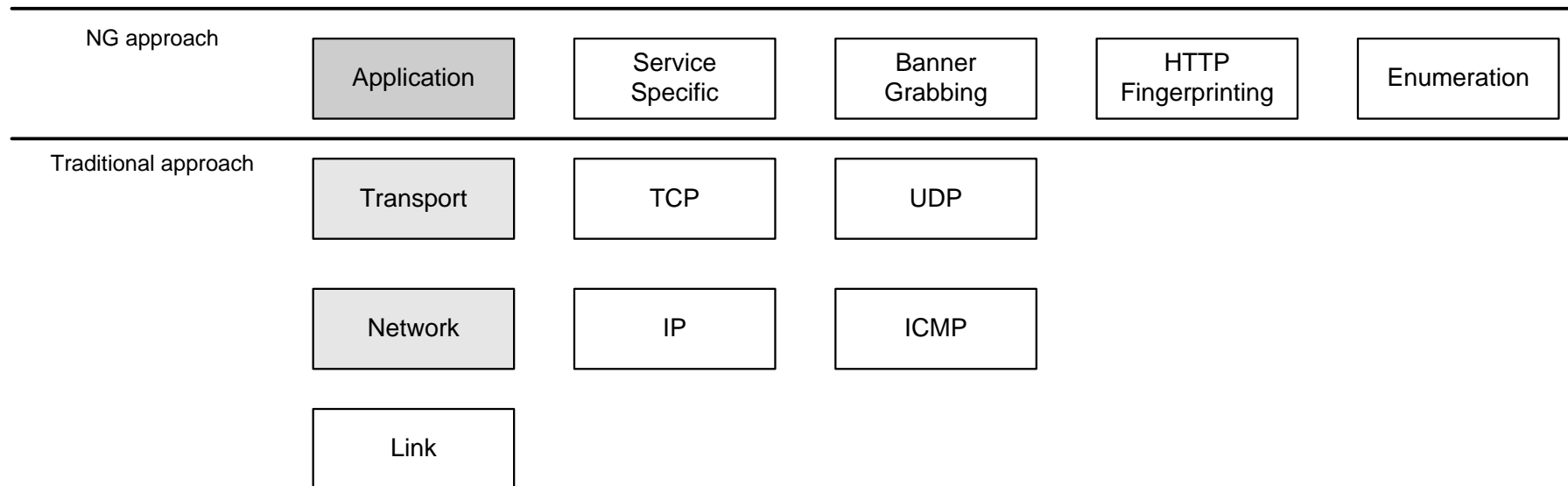
The Future of Active Operating System Fingerprinting

- There is a **certain limit** to the abilities of TCP/IP stack fingerprinting that **other methods might provide the remedy to**
- These other methods might provide with **additional insights** that cannot be gained using the traditional TCP/IP stack fingerprinting methods

The Future of Active Operating System Fingerprinting



Using the application layer



Using the application layer

- When an application layer based fingerprinting test is used, **other fingerprinting tests** which uses the transport and IP layers **can be used at the same time** based on the information exchanged during the “setup stage” of the application based fingerprinting test
- **Several methods:**
 - Generic Tests
 - Application based fingerprinting tests
 - Application based fingerprinting tests targeting a specific operating system family or a certain group of operating systems

Banner Grabbing

- Obtaining a service banner is usually a **simple process**, nearly as simple as it takes one to change the banner a certain service is using
- Many operating system **hardening guides contain instructions** on **how to change the banner** presented with several services
- Therefore we should treat service banners with **cautious**
- There are unique cases in which a service banner cannot be altered

Banner Grabbing

```

06/21-17:02:09.520598 192.168.0.1:80 -> 192.168.0.3:49429
TCP TTL:64 TOS:0x0 ID:22699 IpLen:20 DgmLen:181
***A*** Seq: 0xEBC6279 Ack: 0xCD94564E Win: 0x16D0 TcpLen: 20
48 54 54 50 2F 31 2E 30 20 34 30 31 20 41 75 74 HTTP/1.0 401 Aut
68 6F 72 69 7A 61 74 69 6F 6E 20 52 65 71 75 69 horization Requi
72 65 64 0D 0A 57 57 57 2D 41 75 74 68 65 6E 74 red..WWW-Authent
69 63 61 74 65 3A 20 42 61 73 69 63 20 72 65 61 icate: Basic rea
6C 6D 3D 22 44 2D 4C 69 6E 6B 20 44 49 2D 37 31 lm="D-Link DI-71
34 50 2B 22 0D 0A 43 6F 6E 74 65 6E 74 2D 74 79 4P+"..Content-ty
70 65 3A 20 74 65 78 74 2F 68 74 6D 6C 0D 0A 0D pe: text/html...
0A 34 30 31 20 41 75 74 68 6F 72 69 7A 61 74 69 .401 Authorizati
6F 6E 20 52 65 71 75 69 72 65 64 0D 0A on Required..

```

- The difference from other service banners is that the reply we have received from the D-Link 714P+ wireless broadband router cannot be altered by using any of the device's configuration interfaces**

Banner Grabbing

- When implementing a banner-grabbing module as an active operating system fingerprinting module, **different service banners should get different weights** in the overall matching process **according to the ability and ease of changing them**
- The over cautious can grant a banner grabbing module implementation with **less impact on the overall fingerprinting results** than other fingerprinting modules used

Banner Grabbing

- A banner grabbing module is **not a magical solution**. Like any other fingerprinting module, it suffers from a number of issues such as the **inability to clearly identify certain versions of the operating systems, and/or hardware based devices**

```
[root@angelfire root]$ telnet x.x.x.x
Trying x.x.x.x...
Connected to x.x.x.x.
Escape character is '^]'.
```

```
User Access Verification
```

```
Username:
```

Application-based Stack Fingerprinting

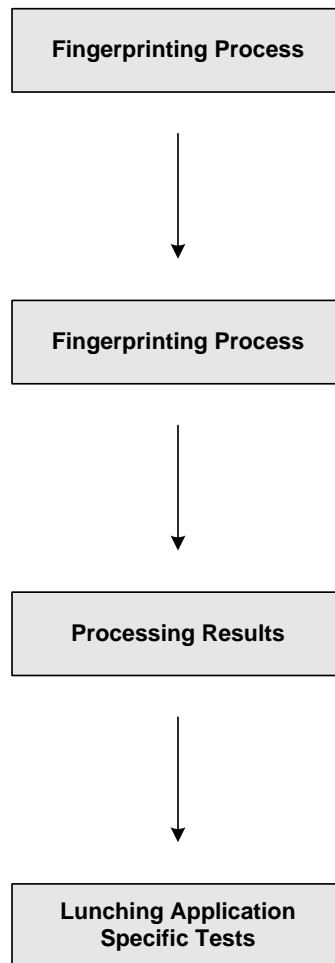
- **Generic Tests**

- Web Servers (HTTP Fingerprinting)
- Any other service which is widely implemented

- **Group specific**

- There is a need to use a specific test which would specifically target an operating system family in question, and would provide with the required information to dismantle the group, and to provide a clear and decisive answer about the underlying operating system of the targeted machine

Application-based Stack Fingerprinting



Xprobe2 v0.2 Live Demo

Further Reading

- Arkin Ofir, “ICMP Usage in Scanning” research project

<http://www.sys-security.com>

- Arkin Ofir, “ICMP Usage in Scanning” version 3.0, June 2001

<http://www.sys-security.com/html/projects/icmp.html>

- Arkin Ofir & Fyodor Yarochkin, “X – Remote ICMP based OS fingerprinting Techniques”, August 2001 (This paper describes the first generation of Xprobe).

http://www.sys-security.com/archive/papers/X_v1.0.pdf

Further Reading

- Arkin Ofir & Fyodor Yarochkin, “ICMP based remote OS TCP/IP stack fingerprinting techniques”, Phrack Magazine, Volume 11, Issue 57, File 7 of 12, Published August 11, 2001.

<http://www.sys-security.com/archive/phrack/p57-0x07>

- Arkin Ofir & Fyodor Yarochkin, “Xprobe2 - A ‘Fuzzy’ Approach to Remote Active Operating System Fingerprinting”, <http://www.sys-security.com/archive/papers/Xprobe2.pdf>, August 2002.
- Arkin Ofir, Fyodor Yarochkin, Meder Kydyraliev, “Next Generation Active Operating System Fingerprinting – The Present & Future of Xprobe2”, July 2003.

Questions?

