Keep your network in sight

On the Current State of Remote Active OS Fingerprinting Tools

Ofir Arkin
CTO
ofir.arkin@insightix.com

Defcon 13

What this talk is about?

- This talk examines different aspects of remote active operating system fingerprinting
  - Examines different active OS fingerprinting methods & techniques
  - Discusses their limitations and advantages
  - Explains the state of the current used technology
  - Deals with the question of what can and cannot be accomplished using remote active OS fingerprinting
  - Looks at what should be done in the future
  - Analyzes the accuracy aspects of remote active OS fingerprinting and of several active OS fingerprinting tools
- Presents the new version of Xprobe2 (Xprobe2 v0.3)

Ofir Arkin

- CTO and Co-Founder, Insightix
  http://www.insightix.com
- Founder, The Sys-Security Group
  http://www.sys-security.com
- Computer Security Researcher
  - Infrastructure Discovery
    - ICMP Usage in Scanning
    - Xprobe2 (The Active OS Fingerprinting Tool)
  - VoIP Security
  - Information Warfare
- Member
  - VoIPSA (Board member, chair security research committee)

Remote Active OS Fingerprinting An Introduction

An Introduction

- Remote active operating system fingerprinting is the process of actively determining a targeted network node's underlying operating system by probing the targeted system with several packets and examining the response(s), or lack thereof, received
- The traditional approach is to examine the TCP/IP stack behavior (IP, TCP, UDP, and ICMP protocols) of a targeted network element when probed with several legitimate and/or malformed packets

An Introduction

- The received results would then be compared to a signature database in an effort to find an appropriate match
- Remote active OS fingerprinting is not limited to the IP and Transport layers only
- The application layer can be used as another venue for information gathering about the underlying operating system

An Introduction

- The way of operation of an active OS fingerprinting tool varies from one remote active OS fingerprinting tool to another
  - The OS fingerprinting tests used (i.e. what does it check for)
  - The type of packets sent (i.e. RFC compliant, crafted)
  - The number of packets sent
  - Other variables

An Introduction

- Identifying the underlying operating system of a network element, whether remote or local, is an important parameter for the success of many processes in the networking and security world
- Building a network inventory, getting the right context for network intrusion detection systems (NIDS) and/or network intrusion prevention systems (NIPS), and performing a vulnerability analysis are all good examples among many other for the use of active operating system fingerprinting

Remote Active OS Fingerprinting Strengths

Strengths

- Control over the parameters to scan for (i.e. the stimulus)
- Control over the pace of the scan and its initiation
- Provides with fast results
- Can cover entire IP address ranges*
- Can be used from a single point to scan multi-points
- Can be used from multi points to scan multi-points

Remote Active OS Fingerprinting Weaknesses

Weaknesses

- The weaknesses of remote active OS fingerprinting are divided into:
  - Scanning conditions and environmental effects
  - Operation of the OS fingerprinting process
  - Signature DB related issues
  - Tool related issues

Weaknesses
Scanning conditions and environmental effects

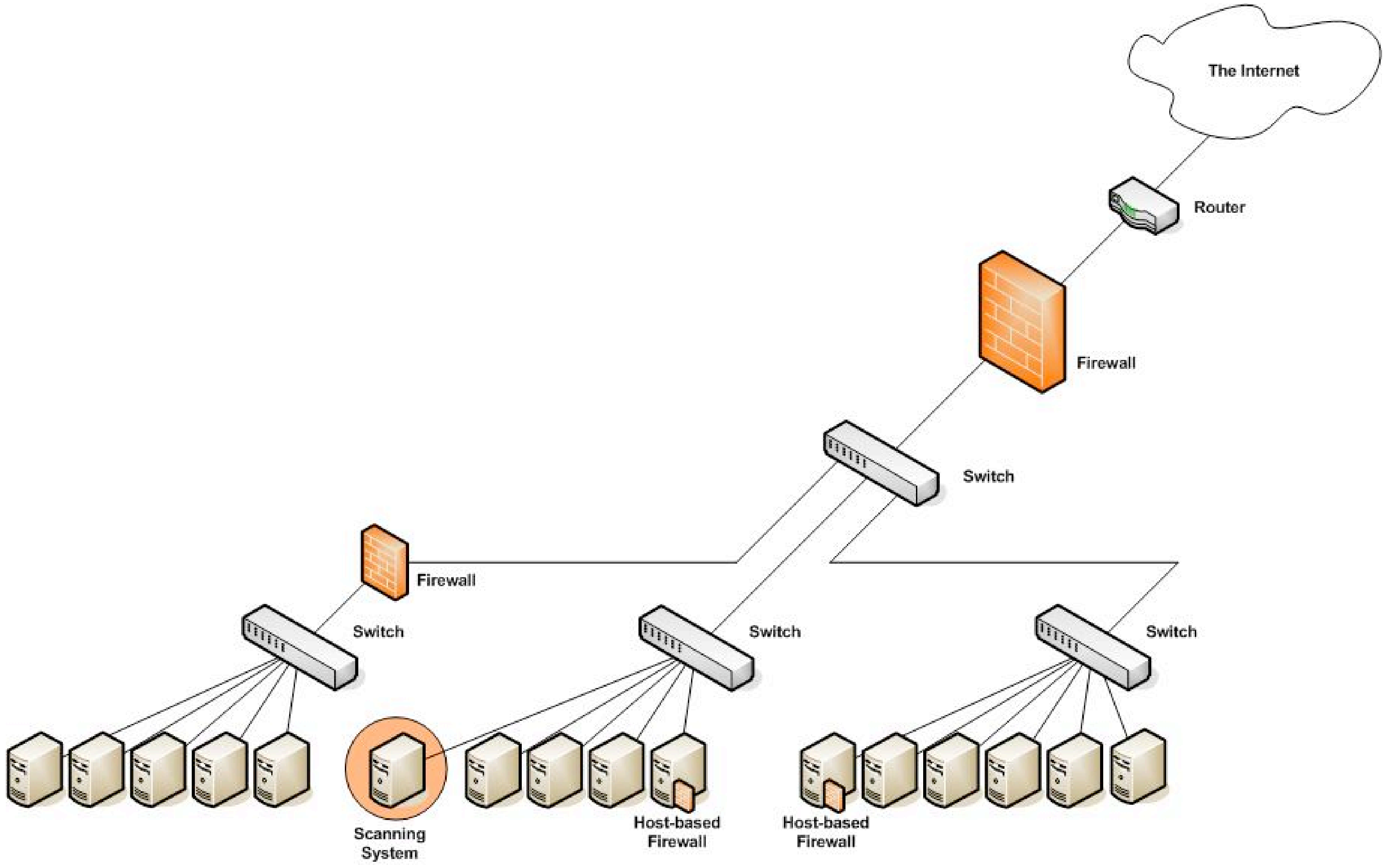
Scanning Conditions & Environmental Effects

- There is no control over the quality of the scan
- The quality of the scan is directly affected by the environment
  - The location of the scanning system and the target system and what is between them (local network, remote network, over the Internet)
    - The path between the scanning element to the target element (firewalls, load balancers, scrubbers, etc.)
    - The target element itself (i.e. personal firewall, tunable parameters, etc.)
- Lack of intelligence (i.e. to determine the terrain and the limitations of the scan, switching scanning tactics, 'understand' results)

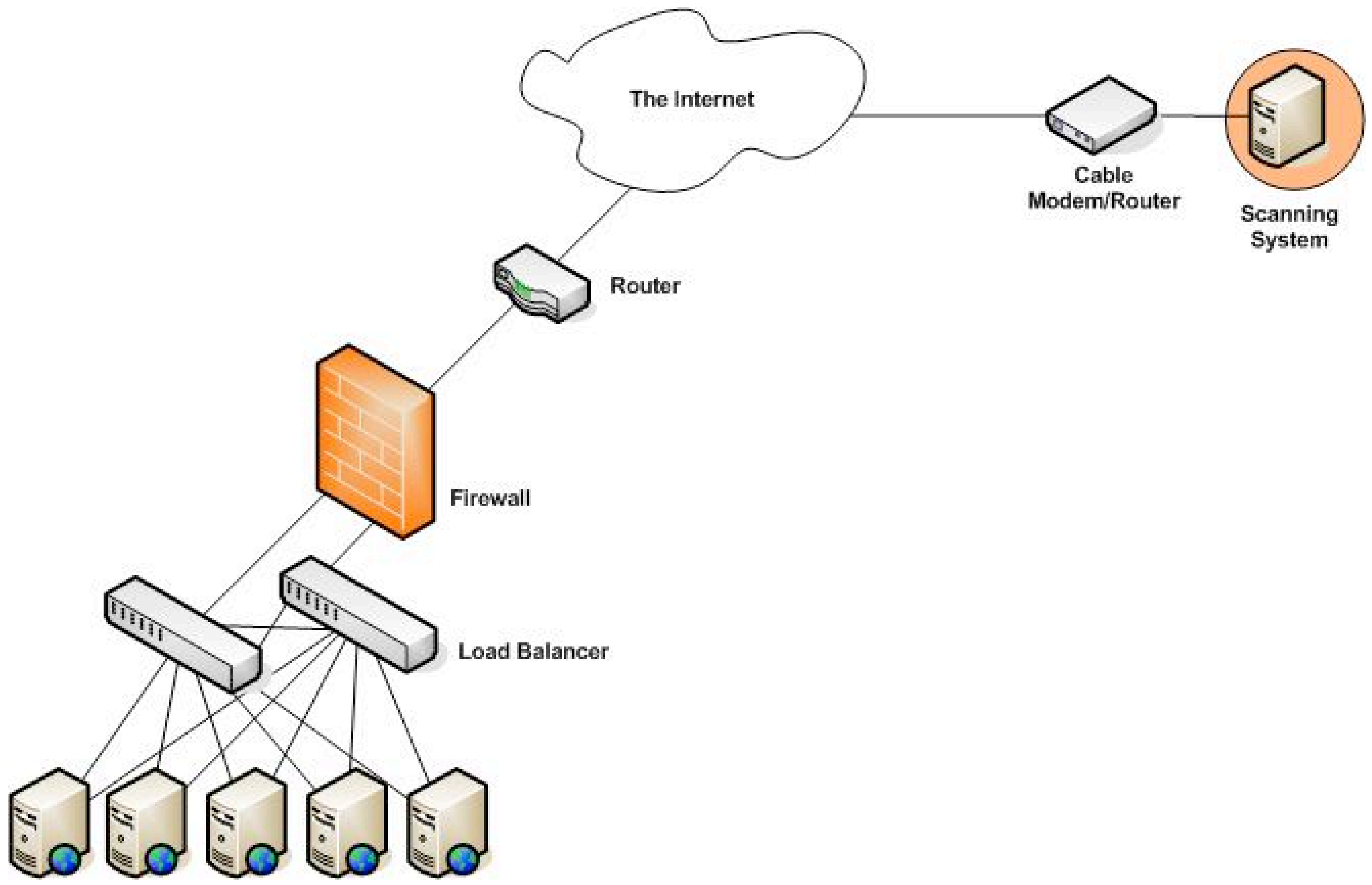
Network Obstacles

- A remote active OS fingerprinting tool will be able to detect the underlying operating system of an element which will be operational ('up') on a network at the time of the scan
- This is if the packets sent by the tool are able to reach the probed elements, and that the probed element's OS signature is included with the tool's signature DB
- Network obstacles such as Network firewalls, host-based firewalls, NAT enabled devices, load balancers and other, may block probe packets from reaching their target

Network Obstacles



Network Obstacles



Network Obstacles

- 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) (or another network obstacle) chances are that the quality of the results would be degraded to the point false results or no results at all will be produced

Different Networking Devices May Alter A Packet's Field Value

- A networking device along the path between the source system to its destination may alter one, or more, field values which are relevant to the OS fingerprinting process
- This would result with issues determining what is the targeted machines underlying operating system
- Example: Scrubbers, QoS devices

The Use of Crafted/Malformed Packets

- If malformed packets are used with the OS fingerprinting process, a filtering device (i.e. firewall, IPS), and even an end-point device, may drop the packets, if the device analyzes packets for non-legitimate content
- Example: Weird TCP flags combinations, i.e. SYN|FIN, SYN|RST
- 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 may 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 may be altered:
  - 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 (i.e. Linux) that alter the way the particular operating system's TCP/IP stack responses to certain packets

A TCP/IP Stack's Behavior Might Be Altered

- 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

Weaknesses
Signature DB Related Issues

Signature DB related issues

- The signature DB is one of the most important parts of an active OS fingerprinting tool
- The quality of the results produced by an OS fingerprinting tool is directly affected by the way the signature database of a tool was built and is maintained
- 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, this even if the tool would use the most advanced fingerprinting tests

Producing Signatures

- Producing signatures is an important process
  - Signatures must be produced in a controlled environment
  - The test device must be controlled
  - The terrain must not harm the process
  - A lab is the must appropriate signature production environment
- Examples for problems
  - Lab environment vs. Internet
  - When there is no understanding of the process there are some weird things like: ‘Let’s fingerprint this firewalled device’

Strictly controlled vs. Loosely controlled signature DB

- Strictly controlled
  - Signatures are inserted into the DB only after verification (usually only by a tool maintainer)
  - Usually it is a slower process
  - Limitation of the number of signatures / devices
  - Extremely accurate
- Loosely controlled
  - Signatures are submitted over the internet
  - Signature creation process is not controlled
  - Many signatures are produced in a wrongfully manner
  - Creates an uncontrolled DB
  - Extremely inaccurate

What do we fingerprinting?

- 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

What do we fingerprinting?

- Hardware based devices of the same manufacture will usually run the same, or a slightly different, firmware (a.k.a software) version
- It will be either one version for all, or a particular version for a particular functionality

What do we fingerprinting?

- Example I: Cisco IOS
  - A Cisco 7200 router will be fingerprinted exactly the same as Cisco's Aironet 1100/1200 wireless access points
  - They run the same operating system - Cisco IOS
  - It is impossible to tell their use (I.e. router, wireless access point) according to traditional TCP/IP stack based active OS fingerprinting
- Example II:
  - Foundry Networks IronWare operating system (Net/Fast/Big Iron family)
- Example III:
  - Printers (i.e. HP Printers – it is not about their modules but rather it is their firmware), etc.

What do we fingerprinting?

- Unfortunately with many active OS fingerprinting tools these issues were not taken into account
- These tools have a corrupted DB

The inability to implement new fingerprinting tests due to DB population and control problems

- When a new fingerprinting test is implemented a signature DB of an active OS fingerprinting tool needs to be updated to reflect the addition of the new test
- An uncontrolled signature DB cannot handle new fingerprinting tests, since some of its signatures cannot be rebuilt, expanded, or recreated to reflect the addition of the new test
- This can creates differences with the quality of the signatures

Political

- Where your work is abused
- …and your signature DB is stolen

Weaknesses
Operation of the OS fingerprinting tool

The Way Probe Results Are Being Matched

- Scan results needs to be compared to the signature DB in order to find a match
- The comparison process can be done either using:
  - Strict signature matching
  - Statistical analysis approach

The Way Probe Results Are Being Matched

- A Strict Signature Matching based Tool
  - Would search for a 100% match between the received results and a tool's signature database
  - If a 100% match is not found, than there are no results
  - 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

- Statistical based algorithms (the best match)
  - Using statistical based algorithms a tool is able to provide with better resistance against environmental affects which might take their toll on a target system and on probe packets
  - Some fingerprinting tests may have bigger impact over the overall accuracy of the test results compared with other tests used. Their failure may or may not harm with the ability to provide with granular results (i.e. not grouped)
  - Remark: Xprobe2 was the first open source tool to implement a statistical analysis based mathematical algorithm (‘fuzzy logic’) to provide with a best effort match between probe results to a signature database

The Use of a Fixed Number of Fingerprinting Tests

- A fixed number of fingerprinting tests are used
- A fixed number of parameters are examined
- In theory:

  Possible matches = the number of tests X number of parameters examines X parameter's 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 to provide with a clear distinction between different OSs

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 a probe's reply
- These parameter values would need to be different among many TCP/IP stack implementations
- Therefore a number of those tests are needed in order to achieve a broader distinction between different TCP/IP stack implementations
- It suggests that the usage of more parameter rich fingerprinting tests with an active operating fingerprinting tool will provide better overall results
- An active operating system fingerprinting tool must, therefore, reserve the ability to be able to support new fingerprinting methods as they are published

Some Fingerprinting Tests May Have Bigger Impact on the Overall Results

- Some fingerprinting tests have bigger impact over the overall accuracy of the test results compared with other tests used
- If these tests fail, for some reason, the quality of the produced results will be significantly lowered

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 which rely on a certain fingerprinting niche 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

- Example:
  - Windows XP SP2 and Windows 2003
  - Windows 2000 and Windows XP
  - Linux Kernel 2.6.10, 2.6.11…

The Inability to Determine the Exact Windows OS 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, traditional active operating system fingerprinting tools will identify a targeted machine runs Microsoft Windows 2000, but will not be able to determine which OS service pack version is installed (if any at all)

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 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

- When scanning different machines on different topologies some tests would be proved useless
- Controlling which tests to use would result with better accuracy and less chance of being detected
- 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

Denial of Service

- It is a known fact that some active OS fingerprinting tools may introduce a denial-of-service condition (i.e. a reboot, a crash) to some network elements when they probe them for information
- This is either due to:
  - The pace of the scan and the number of packets sent to a network element during an active network discovery process (i.e. 1500 packets in 3 seconds or less is not a good idea)
  - The usage of non-RFC compliant probe packets, which some probed network elements cannot withstand due to their unexpected input
- Exmaple: My Kyocera Mita FS-1900 Printer

Weaknesses Summary

Weaknesses - Summary

- The OS fingerprinting methods a certain remote active OS fingerprinting tool uses requires that the scanning conditions would meet several conditions in order to produce with a successful identification of the underlying operating system of a remote machine
- Some of those conditions cannot be met under several scanning terrains
- One good example would be a web server behind a well fortified firewall

Weaknesses - Summary

- Since some of the OS fingerprinting tests a remote active OS fingerprinting tool would use may fail, the accuracy of the tool will be degraded when optimal scanning conditions would not be met
- If the OS fingerprinting tests which would fail, would be those with the bigger impact on the accuracy of the tool's result, the result the tool would produce would be poor at best

Weaknesses - Summary

- The currently used TCP/IP-based OS fingerprinting test are not granular enough with their results (i.e. Microsoft Windows based OSs)

Other approaches and their limitations

Other approaches and their limitations

- Some researchers suggested to use a certain OS fingerprinting niche to fingerprint the underlying operating systems of remote machines in light of Internet conditions
- The suggested tests would use an opened TCP port, and only would examine the TCP stack implementation of the remote machine
- Some of those tests requires specific data to be exchanged between the scanning system to its target element, and a great number of packets to be exchanged

Other approaches and their limitations

- The main problem of this approach is that this approach is ok to use when you wish to identify families of operating systems and not an exact operating system version
- Another issue with this approach is that some other tests, which are currently available with open source remote active OS fingerprinting tools, produces the same quality of results when run against an opened TCP port with a single packet…

The Needed Solution

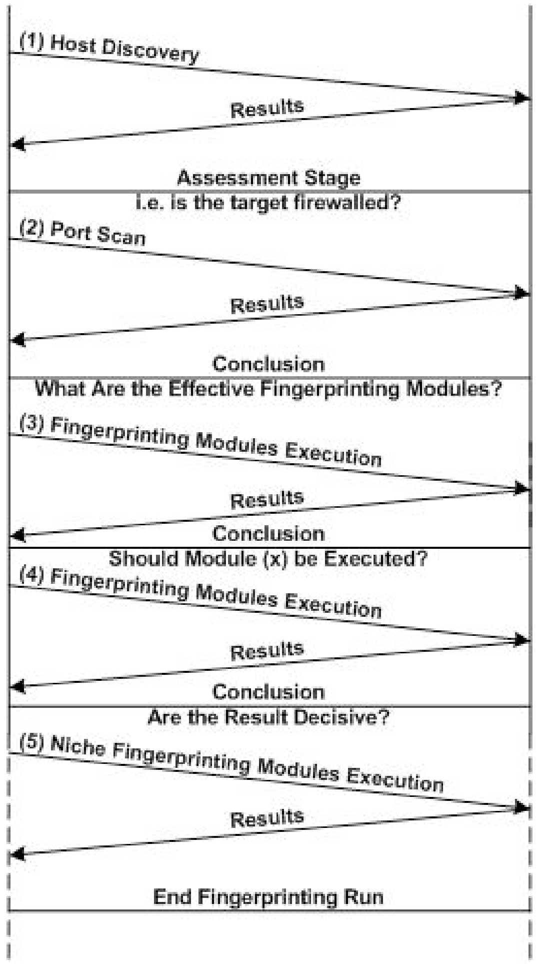
The Needed Solution

- Intelligence in scanning must be introduced
- Understanding of the terrain a tool operates in is crucial
- An active OS fingerprinting tool must understand the quality of the results received
- More tests needs to be evaluated in order to find more OS fingerprinting tests which will have significance in the OS fingerprinting process

The Needed Solution

- An integration between Stack-based OS fingerprinting tests and application layer based fingerprinting tests tailored towards the services found opened on a targeted system(s) and/or a service commonly found with the operating system family in question, must be created

The Needed Solution


(1) Host Discovery
Results
Assessment Stage
i.e. is the target firewalled?
(2) Port Scan
Results
Conclusion
What Are the Effective Fingerprinting Modules?
(3) Fingerprinting Modules Execution
Results
Conclusion
Should Module (x) be Executed?
(4) Fingerprinting Modules Execution
Results
Conclusion
Are the Result Decisive?
(5) Niche Fingerprinting Modules Execution
Results
End Fingerprinting Run

Xprobe2

The Xprobe2 Project

- An open source remote active OS fingerprinting tool, which presents an alternative to other remote active OS fingerprinting tools
- Developers
  - Fyodor Yarochkin
  - Ofir Arkin
  - Meder Kydyraliev
- The project represents our take, beliefs and ideas, and we hope it contributes to the community at large

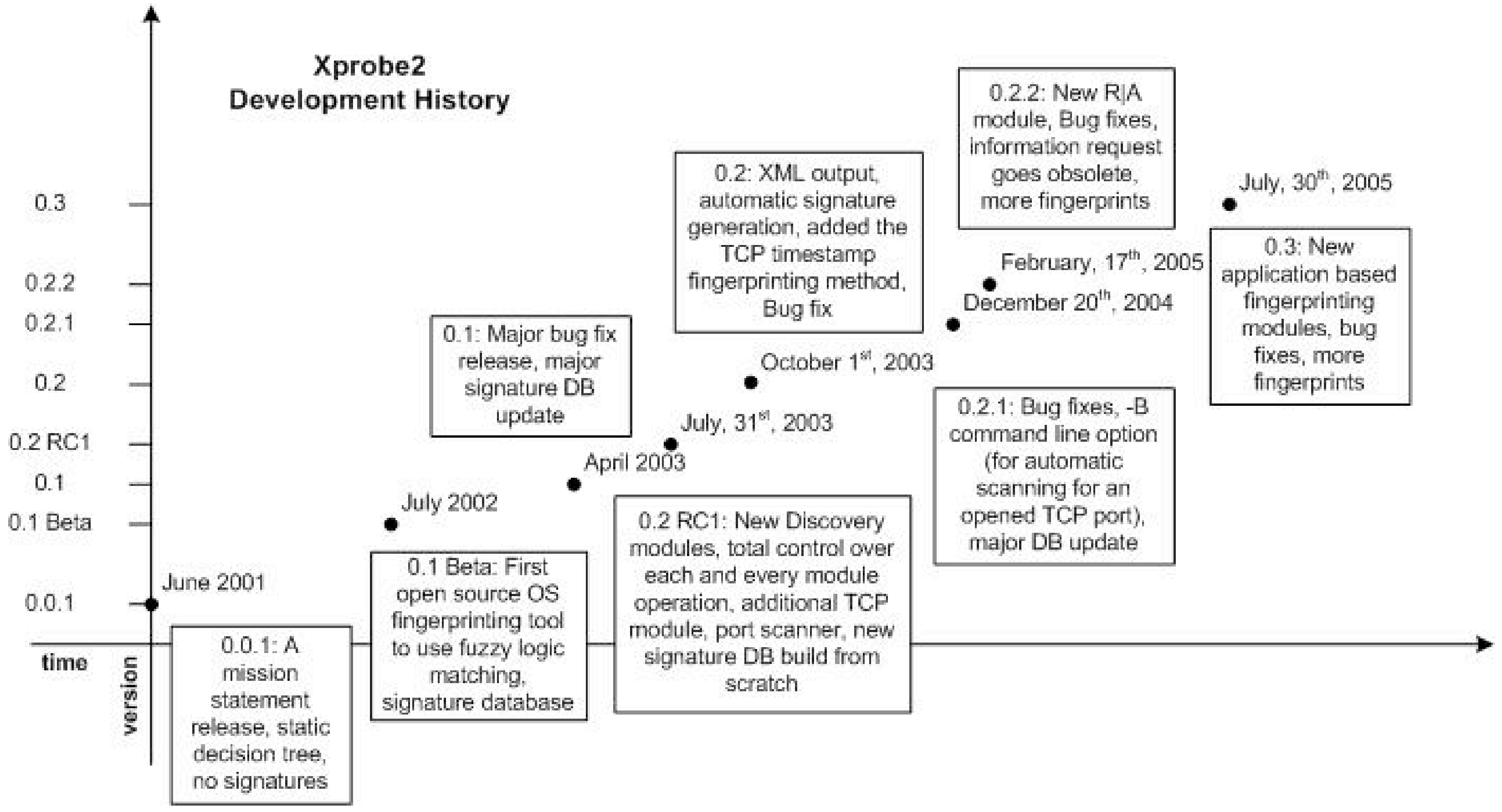
The Xprobe2 Project

- Voted one of the top 75 network security tools
- Replaced NMAP as the OS detection engine for Nessus
- Found with many open source operating system installations

Xprobe2 Highlights

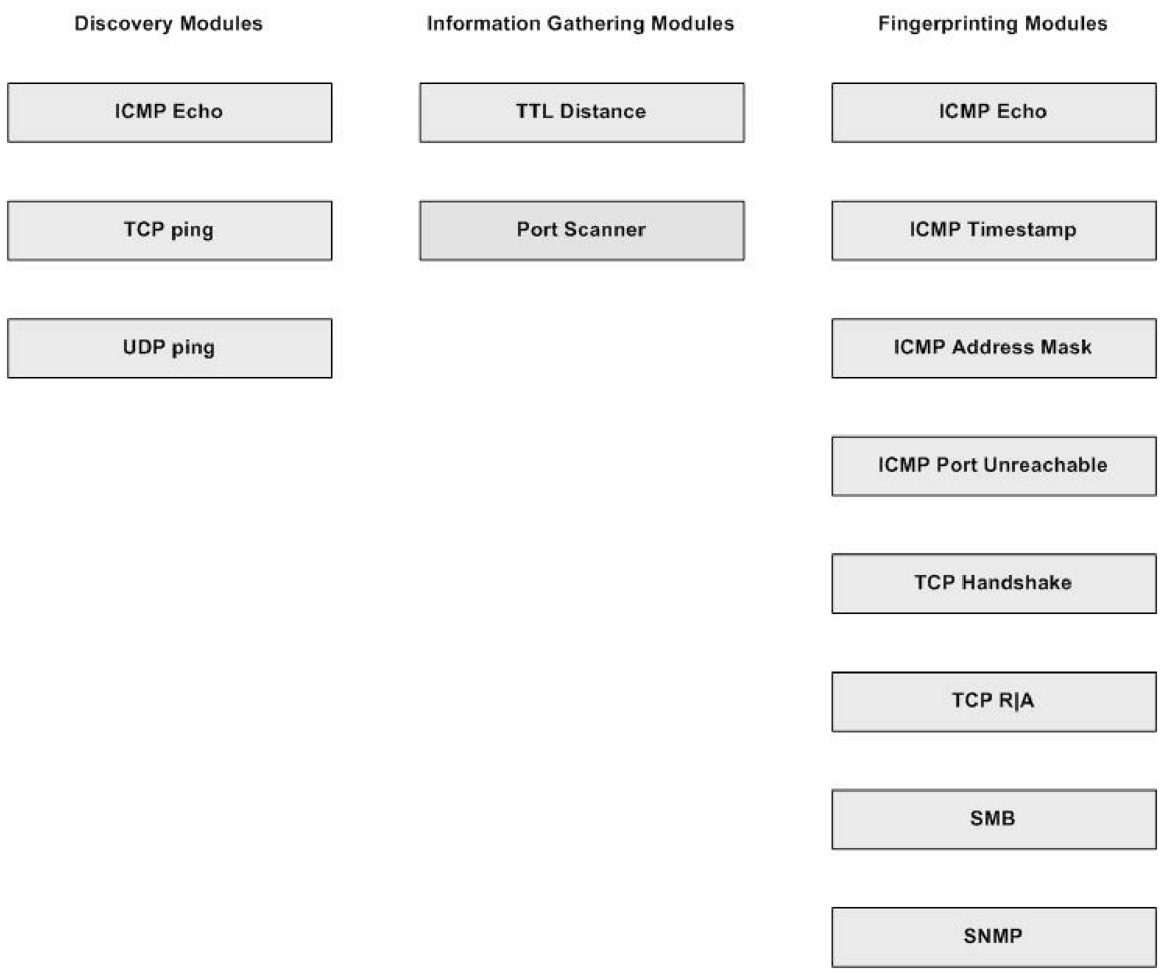
- Uses less packets then any other OS fingerprinting tool
- Does not disrupt the operation of a target system
- Allows complete control over the execution of OS detection modules
- Far more accurate then any other OS detection tool

Xprobe2 - Project History

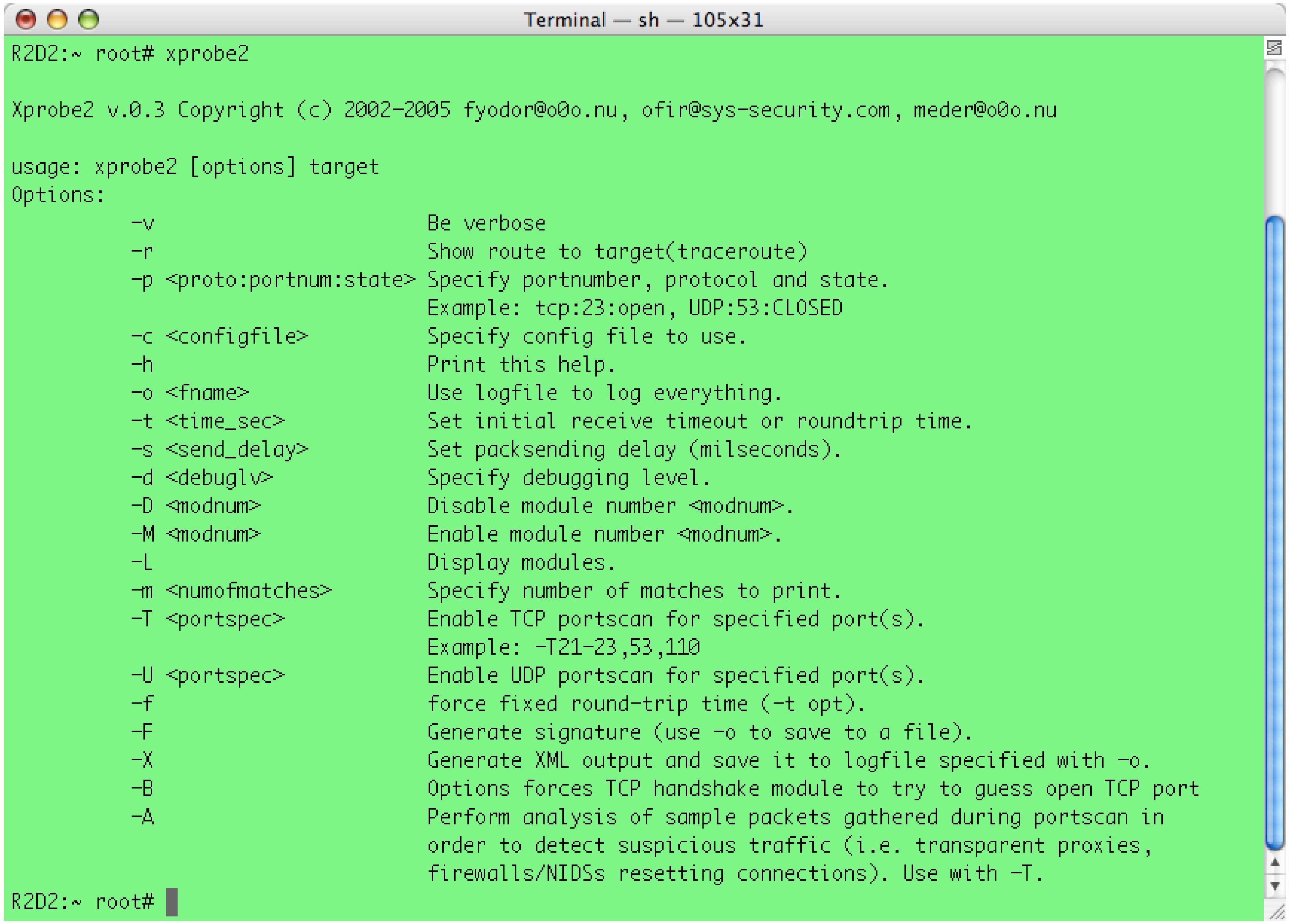

Xprobe2
Development History
0.3
0.2.2
0.2.1
0.2
0.2 RC1
0.1
0.1 Beta
0.0.1
version
time
June 2001
July 2002
April 2003
July, 31st, 2003
October 1st, 2003
December 20th, 2004
February, 17th, 2005
July, 30th, 2005
0.0.1: A mission statement release, static decision tree, no signatures
0.1 Beta: First open source OS fingerprinting tool to use fuzzy logic matching, signature database
0.1: Major bug fix release, major signature DB update
0.2 RC1: New Discovery modules, total control over each and every module operation, additional TCP module, port scanner, new signature DB build from scratch
0.2: XML output, automatic signature generation, added the TCP timestamp fingerprinting method, Bug fix
0.2.1: Bug fixes, -B command line option (for automatic scanning for an opened TCP port), major DB update
0.2.2: New R|A module, Bug fixes, information request goes obsolete, more fingerprints
0.3: New application based fingerprinting modules, bug fixes, more fingerprints

Xprobe2 Software Modules

Software Modules



Command Line Options



```
R2D2:~ root# xprobe2

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

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.
R2D2:~ root#
```

Discovery Modules

- Designed to perform Host Detection
  - Echo request (ping)
  - TCP Ping
  - UDP Ping
- Has an important added value where it provides information for the automatic receive timeout mechanism
- Automatically adjust to the scanning conditions

Port Scanner

- Port Scanner
  - Information about opened and closed ports is used with the OS fingerprinting modules
- -T<VALUES>
- -U<VALUES>
- Examples:
  - -T139
  - -T10-200,1024,5675
  - -T20-40,80

Port Scanner

```
...
[+] Host: x.x.x.x is up (Guess probability: 25%)
[+] Target: x.x.x.x is alive. Round-Trip Time: 0.00149 sec
[+] Selected safe Round-Trip Time value is: 0.00298 sec
[+] Portscan results for x.x.x.x:
[+]  Stats:
[+]   TCP: 4 - open, 18 - closed, 0 - filtered
[+]   UDP: 0 - open, 0 - closed, 0 - filtered
[+]   Portscan took 2.50 seconds.
[+]  Details:
[+]   Proto     Port Num.       State           Serv. Name
[+]   TCP       21              open            ftp
[+]   TCP       22              open            ssh
[+]   TCP       23              open            telnet
[+]   TCP       37              open            time
[+]  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 8 (SunOS 2.8)" (Guess probability: 90%)
[+] 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: "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.
```

The OS Fingerprinting Modules

- What is usually needed?
  - Opened TCP port
  - Closed TCP port
  - Closed UDP port
  - ICMP echo reply/no reply
  - ICMP timestamp reply/no reply
  - ICMP Address Mask reply/no reply

The OS Fingerprinting Modules

- Using the R|A OS fingerprinting module Xprobe2 has the ability to distinguish between the following groups of Linux Kernels:
  - 2.4.0-2.4.3
  - 2.4.4
  - 2.4.5-2.4.18
  - 2.4.19-2.4.28
  - 2.6.0-2.6.7
  - 2.6.8
  - 2.6.9-2.6.12

Useful command line options

- xprobe2 -v -B <IP ADDRESS>
- xprobe2 -v -B -F -o <FILE_NAME> <IP_ADDRESS>

- The -B command line option will use the following TCP ports:
  - 80, 443, 23, 21, 25, 22, 139, 445, 6000
- Automatic signature generation using the -F command line option

Introducing Xprobe2 v0.3

Download From

- http://www.sys-security.com
- http://prdownloads.sourceforge.net/xprobe/xprobe2-0.3.tar.gz?download
- MD5 (xprobe2-0.3.tar.gz) 3ebb89ed9380038d368327816e34ec54
- SHA1 (xprobe2-0.3.tar.gz) c28d48823c1b953f73fd1b1fbced5c77a63d2bf0

Features added to Xprobe2 v0.3

- Application-based OS fingerprinting modules
- New signatures
- Bug fixes

Application-based OS Fingerprinting

- SMB Module
  - Retrieves Native OS and Native LANMAN parameters from SMB Session Setup and X replies
  - Resolves the problem TCP/IP stack based OS fingerprinting tools have with Microsoft Windows-based OSs
  - Allows extraction of information from Windows NT4, 2000, XP, 2003
  - Very useful with 2000,XP,2003
  - Needs file and print sharing to be enabled
  - New keywords:

    smb_lanman = Windows Server 2003 5.2

    smb_nativeos = Windows Server 2003 3790

Microsoft Windows 2000 SP4 (SMB Module Not Enabled)

```
R2D2:~ root# xprobe2 -v -B 192.168.2.103

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

[+] Target is 192.168.2.103
[+] 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
[x] [4] infogather:ttl_calc  -  TCP and UDP based TTL distance calculation
[x] [5] infogather:portscan  -  TCP and UDP PortScanner
[x] [6] fingerprint:icmp_echo  -  ICMP Echo request fingerprinting module
[x] [7] fingerprint:icmp_tstamp  -  ICMP Timestamp request fingerprinting module
[x] [8] fingerprint:icmp_amask  -  ICMP Address mask request fingerprinting module
[x] [9] fingerprint:icmp_port_unreach  -  ICMP port unreachable fingerprinting module
[x] [10] fingerprint:tcp_hshake  -  TCP Handshake fingerprinting module
[x] [11] fingerprint:tcp_rst  -  TCP RST fingerprinting module
[x] [12] fingerprint:smb  -  SMB fingerprinting module
[x] [13] fingerprint:snmp  -  SNMPv2c fingerprinting module
[+] 13 modules registered
[+] Initializing scan engine
[+] Running scan engine
[-] ping:tcp_ping module: no closed/open TCP ports known on 192.168.2.103. Module test failed
[-] ping:udp_ping module: no closed/open UDP ports known on 192.168.2.103. Module test failed
[-] No distance calculation. 192.168.2.103 appears to be dead or no ports known
[+] Host: 192.168.2.103 is up (Guess probability: 50%)
```

Microsoft Windows 2000 SP4 (SMB Module Not Enabled)

```
[+] Target: 192.168.2.103 is alive. Round-Trip Time: 0.00144 sec
[+] Selected safe Round-Trip Time value is: 0.00287 sec
[-] fingerprint:smb need either TCP port 139 or 445 to run
[+] Primary guess:
[+] Host 192.168.2.103 Running OS: "Microsoft Windows XP SP1" (Guess probability: 100%)
[+] Other guesses:
[+] Host 192.168.2.103 Running OS: "Microsoft Windows XP" (Guess probability: 100%)
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Server Service Pack 4" (Guess probability: 100%)
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Server Service Pack 3" (Guess probability: 100%)
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Server Service Pack 2" (Guess probability: 100%)
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Server Service Pack 1" (Guess probability: 100%)
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Server" (Guess probability: 100%)
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Workstation" (Guess probability: 100%)
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Workstation SP1" (Guess probability: 100%)
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Workstation SP4" (Guess probability: 100%)
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.
```

Microsoft Windows 2000 SP4 (SMB Module Enabled)

```
R2D2:~ root# xprobe2 -v -B -T139 192.168.2.103

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

[+] Target is 192.168.2.103
[+] 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
[x] [4] infogather:ttl_calc  -  TCP and UDP based TTL distance calculation
[x] [5] infogather:portscan  -  TCP and UDP PortScanner
[x] [6] fingerprint:icmp_echo  -  ICMP Echo request fingerprinting module
[x] [7] fingerprint:icmp_tstamp  -  ICMP Timestamp request fingerprinting module
[x] [8] fingerprint:icmp_amask  -  ICMP Address mask request fingerprinting module
[x] [9] fingerprint:icmp_port_unreach  -  ICMP port unreachable fingerprinting module
[x] [10] fingerprint:tcp_hshake  -  TCP Handshake fingerprinting module
[x] [11] fingerprint:tcp_rst  -  TCP RST fingerprinting module
[x] [12] fingerprint:smb  -  SMB fingerprinting module
[x] [13] fingerprint:snmp  -  SNMPv2c fingerprinting module
[+] 13 modules registered
[+] Initializing scan engine
[+] Running scan engine
[-] ping:tcp_ping module: no closed/open TCP ports known on 192.168.2.103. Module test failed
[-] ping:udp_ping module: no closed/open UDP ports known on 192.168.2.103. Module test failed
[-] No distance calculation. 192.168.2.103 appears to be dead or no ports known
[+] Host: 192.168.2.103 is up (Guess probability: 50%)
```

Microsoft Windows 2000 SP4 (SMB Module Enabled)

```
[+] Portscan results for 192.168.2.103:
[+]  Stats:
[+]   TCP: 1 - open, 0 - closed, 0 - filtered
[+]   UDP: 0 - open, 0 - closed, 0 - filtered
[+]   Portscan took 0.01 seconds.
[+]  Details:
[+]   Proto     Port Num.       State           Serv. Name
[+]   TCP       139             open            netbios-ssn
[+] SMB [Native OS: Windows 5.0] [Native Lanman: Windows 2000 LAN Manager] [Domain: WORKGROUP]
[+] SMB [Called name: WIN2K-SERVER   ] [MAC: 00:0c:29:88:ce:10]
[+] Primary guess:
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Server Service Pack 4" (Guess probability: 100%)
[+] Other guesses:
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Server Service Pack 3" (Guess probability: 100%)
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Server Service Pack 2" (Guess probability: 100%)
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Server Service Pack 1" (Guess probability: 100%)
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Server" (Guess probability: 100%)
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Workstation" (Guess probability: 100%)
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Workstation SP1" (Guess probability: 100%)
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Workstation SP4" (Guess probability: 100%)
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Workstation SP3" (Guess probability: 100%)
[+] Host 192.168.2.103 Running OS: "Microsoft Windows 2000 Workstation SP2" (Guess probability: 100%)
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.
```

Application-based OS Fingerprinting

- SNMP Module
  - Extracts information from SNMP `sysDescr`
  - Supports SNMP v2
  - Will support SNMP v1 & v3 with the next release
  - List of community strings under xprobe2.conf
  - Executed when UDP port 161 is open

An Example with SNMP

```
[+] SNMP [Community: public] [sysDescr.0: FreeBSD beastie 5.3-STABLE FreeBSD 5.3-STABLE #1:
 Sun Jan 23 15:15:20 CST 2005     meder@beastie:/usr/obj/usr/src/sys/BEASTIE i386]
[+] Primary guess:
[+] Host 127.0.0.1 Running OS: "FreeBSD 5.3" (Guess probability: 100%)
[+] Other guesses:
[+] Host 127.0.0.1 Running OS: "FreeBSD 5.2" (Guess probability: 96%)
[+] Host 127.0.0.1 Running OS: "FreeBSD 5.2.1" (Guess probability: 96%)
[+] Host 127.0.0.1 Running OS: "FreeBSD 5.4" (Guess probability: 96%)
[+] Host 127.0.0.1 Running OS: "Apple Mac OS X 10.2.6" (Guess probability: 96%)
[+] Host 127.0.0.1 Running OS: "Apple Mac OS X 10.2.7" (Guess probability: 96%)
[+] Host 127.0.0.1 Running OS: "Apple Mac OS X 10.2.8" (Guess probability: 96%)
[+] Host 127.0.0.1 Running OS: "FreeBSD 4.8" (Guess probability: 93%)
[+] Host 127.0.0.1 Running OS: "FreeBSD 5.1" (Guess probability: 93%)
[+] Host 127.0.0.1 Running OS: "Apple Mac OS X 10.2.2" (Guess probability: 93%)
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.
```

Features added to Xprobe2 v0.3

- New signatures
  - Mac OS X 10.2.x, 10.3.x, 10.4.x
  - Linux Kernels 2.4.29, 2.4.30, 2.6.11, 2.6.12
  - FreeBSD 4.11, 5.4
  - OpenBSD 3.7
- Bug fixes (i.e. Bug with pcap sniffing)

Remote Active OS Fingerprinting Future Directions

Future Directions

- Automating the active OS fingerprinting scan to understand the terrain
- Switch scanning tactics according to the terrain
- Evaluate the quality of the results received
- Determine if the results received are good enough to be presented
- Present results

Questions?

STIF

- Security Tools Integration Framework (STIF) is aimed to provide a unified environment and data exchange platform for automated security assessments in heterogeneous environments.
- It is a platform for "hacking" automation, where STIF emulates the logic of a security analyst to perform repetitive tasks.
- http://o0o.nu/sec/STIF/

Resources

- Ofir Arkin's Web Site: http://www.sys-security.com
- Arkin Ofir, "ICMP Usage in Scanning" version 3.0, June 2001
- Arkin Ofir & Fyodor Yarochkin, "X – Remote ICMP based OS fingerprinting Techniques", August 2001.
- 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.

Resources

- Arkin Ofir & Fyodor Yarochkin, “Xprobe2 - A ‘Fuzzy’ Approach to Remote Active Operating System Fingerprinting”, August 2002.
- Arkin Ofir, Fyodor Yarochkin, Meder Kydyraliev, “The Present & Future of Xprobe2 – Next Generation Active Operating System Fingerprinting ”, July 2003.

insightix

Thanks!