Global Information Assurance Certification Paper

An Overview of Remote Operating System Fingerprinting

Chris Trowbridge
Submitted for SANS GIAC GSEC Practical
Assignment version: 1.4b
16 July 2003

Abstract

Operating System (OS) fingerprinting is the science of determining the operating system of a remote computer on the Internet. This may be accomplished passively by sniffing network packets travelling between hosts, actively by sending carefully crafted packets to the target machine and analysing the response, or through non-technical means. It is used by Security Professionals (known as "White-hats") and Hackers ("Black-hats") alike for mapping remote networks and determining which vulnerabilities might be present to exploit. This paper presents an overview of the various approaches to OS fingerprinting, some current tools available on the Internet together with their features, the underlying techniques they use, and suggestions for defeating these tools.

'Classical' Fingerprinting

Even without resorting to stealth techniques of any kind, hosts will often announce their OS to anyone making a connection to them through welcome banners or header information. For example, when connecting to a host via the standard Telnet protocol the OS version is often sent to the client as part of a welcome message.

FTP will also often provide this information either as a welcome banner or if prompted via a SYST command. HTTP can also be used by connecting to the host and initiating a simple `GET / HTTP/1.0\n` query.

If the Simple Network Management Protocol (SNMP) service is present on the machine (UDP/161), it is often left with the default 'public' community name, which will allow remote detailed querying of the service and hence host.

Other services that send back 'free' useful information include IMAP, POP2, POP3, SMTP, SSH, NNTP and FINGER. This technique is reasonably reliable even now and automated tools exist to make the process simple and painless[1,2].

A slightly more subtle approach is to use the anonymous ftp account (if supported), download a public binary required for ftp to function (eg /bin/ls) and examine it to determine what platform it was built for[8].

A more primitive approach is to port scan the machine using any of the common port scanners freely available (eg Nessus, SAINT) and examine the returned list of listening ports for patterns common to a particular OS. Note that this approach is only effective against less secure hosts; particularly earlier Windows servers and those that do not implement accepted basic computer security concepts as detailed in the SANS GSEC courseware for example[3,4].

Finally, it may be possible to determine the OS of a system by a non-technical solution, such as "social engineering". Learning about the target through phone calls, chatting to the System Administrator, or even a public site tour are all possibilities.

Active IP Packet Fingerprinting

This is the predominant form of OS fingerprinting, provoking the target into eliciting a response and analysing it carefully. A huge amount of information can be gleaned about the response to a carefully crafted network packet. The three common IP packet types (ICMP, TCP, UDP) are all used in this technique and various valid (and invalid) packets are sent to the host to refine the guess of the OS.

The most common techniques in use include the following[5]:

- **FIN Probing** – here a single packet is sent to a known open port with the FIN flag set. This flag usually signals the end of a communication and as such is not expected without a connection being previously established. The standard behaviour (as defined in RFC 793) is to simply ignore the packet; however, many stacks send a RST packet back. This difference is the means to begin creating a fingerprint.
- **TCP ISN Sampling** – TCP uses sequence numbers to keep track of the number of bytes successfully transferred across a connection. When an initial connection attempt is made to a host the OS chooses an initial sequence number to begin the process. This choice can be anything from a constant value, through random increments of previous values, algorithms based on the host's internal clock, to true random systems. It is important to note that the predictability of this ISN also has security implications, leaving the host open to attacks similar to the Mitnick attack[6].
- **ICMP Error Quoting** – Various aspects of the structure of ICMP error packets are useful. ICMP error packets are required to return a small portion of the original message for identification purposes; however, some stack implementations return more than expected. This is especially useful as it allows some basic OS identification of machines that no listening ports open at all.
- **ICMP Error Message Echo Integrity** – ICMP error message packets are required to include some of the original ICMP packet that caused the error. This makes it simple for implementations to use a copy of the original as a template for constructing the reply packet. Using the packet space as a 'scratch' area can leave telltale spurious data that identifies the OS that created it.
- **ICMP Error Message Type of Service (TOS)** – Nearly all implementations return a zero in the TOS field for ICMP port unreachable messages. Linux, however, currently returns a different value in this field making it easy to broadly identify.
- **TCP Options** – These are enhancements to the TCP protocol to improve performance in unreliable or high latency networks[7]. TCP Options have been added as TCP RFCs over time as needs dictated and the patterns of compliance in responses can reveal the underlying OS. Interestingly it is not just the number of options a stack supports that can identify it, but also the order in which the options are returned[8].

Many other differences also exist and are used to a lesser extent. These include:

- **IPID Sampling** – Many OSs utilise a system wide counter for IPID generation. Other, more advanced implementations either randomise this number or set it to 0. Information can be gleaned from the choice of IPID as to the source OS. Again, predictable IPID values can have important security implications outside of OS fingerprinting, including completely silent portscanning[9].
- **TCP Timestamp** – This is a TCP option (and hence is not supported by all IP implementations); however, can be used to determine OS type. It can also be used to determine host uptime if implemented and the update frequency is known.
- **ICMP Error Message Limiting** – RFC 1812 suggests limiting the rate at which ICMP error messages are sent. Some IP stacks implement this suggestion (including Linux, Solaris) whereas Windows hosts do not[10]. This technique is only viable over reliable connections to the remote host and extends scanning time; so, is generally not implemented.
- **Fragmentation Handling** – Fragmented packets, particularly the handling of packets that contain overlapping fragments, are another source of anomalies. Some stacks retain the first version of the overlapped data, and some the second.

Nearly all active fingerprinting tools use some or all of the above tests to obtain data on a host and compare the results with a database of known OSs. As tools become established the OS fingerprint database becomes more extensive, enhancing the resolution of the tool. The age and popularity of a tool is often evidenced by the size of its database. Tools in this category include the much respected Nmap[10], a very flexible scanner that implements all of the above techniques and more. Nmap runs on a wide variety of platforms including various Unices, Linux, Mac OS X, FreeBSD and Windows. Version 3.30 of this tool is reputed to accurately identify 874 different types and patch levels of Operating System by conducting 7 tests. It is widely acclaimed as the best scanning tool available[11] and is constantly being enhanced. Many references to techniques for best using Nmap are to be found on the Internet, with the Nmap site itself being a good place to start.

QueSO is another tool to employ and refine this technology. (*Queso* is the Hispanic shortcut for *"Que Sistema Operativo?"* which translates into *"which operating system?"*[12]) This tool was again widely acclaimed and used by “white hats” and “black hats” alike. Sadly, QueSO appears to be no longer being developed and the original website is now inaccessible. Notably, this was the first such tool to separate the fingerprint database from the code, creating a pseudo standard for packet-based fingerprint databases in the process, including Nmap.

Xprobe 2 demonstrates the most innovative step forward in this approach[13], one of fuzzy fingerprint matching. Strict fingerprint matching relies on a ‘best case’ scenario, with no packets being lost or interfered with while in transit. This is recently not the case as sites scramble to effect perimeter security, often putting a smart firewall to spoof responses, Network Address Translation systems that filter/modify packets, or Scrubber systems that deliberately manipulate packet fields and flags to confuse ID attempts.

Xprobe 2 utilises a basic fuzzy logic scoring system to provide more than one guess at the remote OS, along with a probabilistic score. As discussed in the paper, this approach could even be extended to attempt to discover the source of any packet manipulation, eg identifying the firewall that is obfuscating the tests, providing the target OS can be identified.

Xprobe 2 has been designed to be extensible and an API has been provided to facilitate new test modules being added to the package.

Many, many other tools and "proof of concept" code exists on the subject of Active IP Packet Fingerprinting. These are easily found on the Internet, mostly written by hackers and commonly quite rudimentary in comparison to tools like Nmap and QueSO.

Passive IP Packet Fingerprinting

Passive fingerprinting is where a sniffer is set up on a network and passively collects/analyses packets in the flow of information between hosts. This obviously has limited functionality in a switched network; however, may be useful if attached in a shared network space with a router or well-utilised network host. The techniques employed to identify the source OS are usually a subset of those utilised by Active IP packet fingerprinting, as these tools obviously monitor predominantly legitimate traffic, and hence cannot introduce anomalous packets. Nevertheless, substantial tools can be built using this technology.

Ettercap[14] is a package that is available for most common operating systems (Windows, Mac OS X, Linux, and FreeBSD) which collects and dissects packets from a network. It can also take on an active role, injecting data into the information stream while still maintaining synchronisation. Ettercap is also extensible to encompass new tests as they become available. Interestingly it also offers to identify the network card that created the packet, presumably through the encapsulated MAC address from the Ethernet header. This appears to be the most supported of these tools, with 1279 fingerprint records available in the current version (0.6.b), nearly double that of Nmap. However these include many networked devices, rather than only OS records.

Fingerprinting Through Service (Daemon) Querying

Services that appear on a portscan list can be further queried to produce identifying information. These techniques are more recent in concept and the tools still in the early stages of development.

One example involves querying the LPD daemon. The technique employed by lpdfp[15], a "proof of concept" tool, involves sending malformed Control File commands to the daemon and gauging the response. Various printer daemons respond to error conditions differently allowing a fingerprint to be built up for comparison with a database of expected results.

Ftpmap[16] is another tool still in the early stages of development, which attempts to fingerprint via an ftp daemon.

Ftpmap then inspired SMTPscan[17], a similar tool attempting to exploit differences in mail daemons. In this case 15 tests are employed to provoke

error responses, and the author claims to have correctly identified 77 daemons to date.

More tools exist which build on this technology aiming to generalise the approach to service identification. One notable tool is Amap[18], which proudly proclaims to send a trigger packet to an unknown port and again compares the response to a database of fingerprint packets. It makes little assumptions as to convention and, as such, claims to be able to detect daemons listening on non-standard ports. This tool is being actively worked on to develop the database and even supplies an additional program called 'amapcrap' which the Readme states "sends random crap to a udp, tcp or ssl'ed port, to illicit a response", which you can then send in to add to the database.

It is worth noting that this approach relies heavily on a direct match of the default application daemon to the underlying OS as you are inferring the nature of the OS rather than specifically testing it. Further, as can be seen by the SMTPmap example this technology is not as refined or perhaps reliable, with some 15 tests being required to resolve 77 daemons, compared to Nmap's 7 tests, for 500+ OS. This may be due to the fact that daemons operate at the TCP/IP model application layer and hence part or all of the code can easily be ported between OS variants, particularly on Unix where source code is commonly available.

Fingerprinting Through TCP Retransmission Timeouts

The latest technique for OS fingerprinting uses the novel approach of TCP Retransmission timing. TCP relies on IP for packet transfer over the Internet. TCP is designed to be connection oriented and reliable. To enforce these design rules a connection must be requested and acknowledged before transfer can take place, and every packet must be acknowledged as having been successfully received. If a packet is not received in a predetermined period of time, it must be resent. The original RFC 793 defining the TCP protocol does not explicitly impose an algorithm for the timing of resending 'lost' packets, instead only suggesting one. RFC 2988 notes the sensitivity of TCP to retransmission problems and specifies an algorithm for use when retransmission should occur. Unfortunately not all IP stacks implement the latter RFC or interpret the first in different ways.

This then allows for a novel approach to be taken in which a new connection is requested with a SYN packet, but ignores all SYN-ACK response packets, instead measuring the timing between successive attempts to acknowledge the connection and comparing these times to a fingerprint database. Requesting the TCP timestamp option (when possible) also serves to mitigate the fact that the Internet itself is prone to timing problems. Interestingly, the timestamp option is dependent on IP stack implementation, which further serves to differentiate OS.

An obvious advantage of this method is that only one valid TCP SYN packet is used to initiate the process. Thus the procedure is 'firewall friendly' allowing fingerprinting of hosts behind a firewall with at least one exposed TCP port.

The initial proof of concept tool, RING (Remote Identification Next Generation)[19] demonstrates this methodology, as implemented in C. RING is also implemented as an add-on to Nmap, making a very powerful

combination. The technique was ported to Perl shortly thereafter and named Snacktime[20], independently confirming the concept and increasing portability.

The Final Straw – Chronological Exploits

If all else fails, less tactful attempts at OS identification can be made by launching known exploits for a given OS type against a target host, in chronological order. The theory is that exploits are patched as they are discovered so by starting with the oldest known exploit against a given host and working forward should yield a point at which an attack succeeds, which should thereby identify the revision of OS in use. As an example, Microsoft Windows 95, 98 and NT4 are difficult to distinguish supposedly because the IP stack code was only marginally revised between OS versions. Starting with a basic WinNuke attack and moving forward to more complex attacks such as Teardrop can eventually yield a vulnerability that points to the type and/or hotfix revision that is missing from the OS, thus indicating the current patch level[8].

Avoiding Fingerprinting

Just as there is a vast plethora of tools and techniques available for remote OS identification, there are also techniques and tools for OS obfuscation. Papers on the subject of defeating OS fingerprinting acknowledge that its use is limited[5], not because it is technically infeasible or particularly performance limiting, but because the majority of exploit attacks are by 'script kiddies' who do no more than download the latest exploit script from a website and mass-scan the Internet looking for vulnerable hosts. OS obfuscation tends to be more useful in deterring the more advanced hacker who is explicitly looking for specific OS types to attack.

It is worth digressing here to reinforce the usual "obscurity is not security" rule. That is, you should ensure the system you are trying to protect is fully patched and secured. As discussed above, OS obfuscation is not something to be relied upon as a security measure.

Nevertheless, steps that could be taken include:

- Altering all public banners on services to something non-committal. This can range from the simple editing of files (eg issue, issue.net, motd on Unix) to the more difficult editing of application source code and recompilation (eg Apache Server type response). This is, of course, much more difficult if using proprietary software or software you do not possess the source code to (try changing your IIS Server type response).
- Searching content files for 'incriminating' strings that can give away the OS. Web pages may include comments automatically generated that identify the authoring tool itself, which in turn is known to be commonly used for authoring to a particular platform. Microsoft Frontpage is particularly bad for this. A tool may produce HTML code that allows inference to the server OS. Microsoft Office products produce characteristic HTML which is commonly uploaded to IIS webservers, again identifying a 'Microsoft shop'. Other indicators might include scripted webpages with .aspx URL suffixes

indicating a Microsoft Windows 2000 onwards host, or adopting the three letter .htm suffix, again a Microsoft convention.

- Take stock of the number and types of services you are running on the machine. Do they resemble the target you are trying to impersonate? Can these applications be queried further using a Service Query tool to reveal the nature of the service itself, and hence infer the underlying OS? Follow standard security practice and remove any unwanted services. Tools exist to fake ones that should appear to be there if you wish to impersonate a different OS.
- Changing the way your IP stack responds to probe queries. This is much easier to accomplish on platforms that allow direct manipulation of incoming packets through kernel mode OS hooks (code intervention points), or altered network card drivers. Tools are available for this purpose on some Unix/Linux platforms. Of note is the netfilter/iptables[21] framework inside the 2.4.x Linux kernel. This is a firewall package, implemented as a set of OS hooks. These allow a user routine to be registered as a module and be called whenever a packet matching some criteria is encountered. Thus packets can be inspected and 'mangled' as they are received by the host. Berrueta[22] describes the use of this technology in conjunction with the "IP Personality"[23] netfilter module to make a Linux 2.4.19 based machine appear as a Sega Dreamcast console. Other kernel level solutions simply look to detect odd IP packets and silently drop them with or without logging them for further analysis.
  Note that these approaches work well with strict rule-based fingerprinting, eg Nmap, QueSO. They may be less effective against the newer 'fuzzy matching' techniques of Xprobe 2, or the 'normal' packets of RING.
- Insulating the host from probe packets via a firewall or packet 'Scrubber'. Many firewall products now provide this functionality out of the box. Checkpoint's Firewall-1 software provides a language that allows responses to be crafted to particular packets, thus actively spoofing the target OS. BlackICE PC Protection and Sygate's Personal Firewall also proclaim OS fingerprint protection; however, this appears to be centred around the more usual practice of dropping strange packets rather than dealing with them.
- Insulating the host from probe packets by using Network Address Translation (NAT) technology. In this scenario the host's network is typically given a 'private' network designation, eg 10.0.0.0, or 192.168.0.0. An intelligent gateway accepts outgoing packets from hosts and transparently alters them to make it appear the gateway node is the source. Upon return the gateway transparently reinstates the original address and forwards the packet to the original host. This effectively makes all traffic to or from the network appear to be coming from the one node, making identification of hosts on the private network very difficult.

One thing to note through all of this is that your host is presumably on the Internet for a reason, probably to provide a service. Your OS Fingerprint

'hardening' should not impair the function your host is providing. This may not be evident in your machine itself failing to function, but perhaps in your customers not being able to access your services reliably because you are confusing their client software with strange host ID strings, or dropping packets from their 'slightly broken' IP stack.

The future of OS Fingerprinting

Fingerprinting tools continue to improve, as do the defences against them. A current focus of software development houses is one of computer security, with Microsoft launching its "Trustworthy Computing Initiative"[24] and many OS vendors initiating an automated patch download/update service. Examples include Microsoft's Windows Automatic Update service included in Windows 2000 and onwards, and the Redhat Network service available via the up2date utility in Redhat Linux. These developments, coupled with the general improvement in the world's cyber laws and prosecution rates, are slowly 'raising the bar' on cyber attacks. In this climate, general 'script kiddy' mass-scans may prove too dangerous or fruitless to pursue.

One emerging technology is fingerprinting tools that are themselves automated as part of OS refined attack tools. The first products in this area are already in existence. One tool of note is Sscan2kpre6[25]. Sscan2kpre6 represents the current step of evolution in an OS fingerprinting vulnerability scanner that tests for over 200 exploits and is designed to be incorporated into the design of an Internet worm. The flexibility of Sscan is demonstrated in the Sscan-readme file, which states *"you can have sscan launch off programs, initiate tcp connections and have dialogs with hosts that fit certain criterian* [sic]*, negotiate telnet connections and have sscan log into a host and execute shell commands (useful in coding internet worms), etc, all via a simple built in scripting language defined below."*[25]

The attacks of the future may be well directed and customised according to OS and services running on the target. This may be considered normal worm activity in the future.

Conclusion

Remote OS Fingerprinting is a recent development on the Internet and one to watch. The ability to remotely determine, with high accuracy, the Operating System of a remote host on the Internet is a powerful one. The implications of this technology are perhaps not yet fully understood; however, it is seen as enough of a threat that strategies are currently being developed to prevent and spoof OS Fingerprints.

The most relevant concept to remember is the old adage "Obscurity is not Security". The ease with which exploit tools can be scripted and used *en masse* to find vulnerable hosts largely trivialises the benefits of OS obscurity in today's world. This may change over the coming years as the larger software companies put an emphasis on network security and more specialised attacks are required to exploit systems. The general trend towards increasing penalties for getting caught as the world's cyber laws improve may also serve as a driver towards more refined attacks in the future.

The first developments have already occurred in this area, with OS fingerprinting worm toolkits being developed to refine attacks.

[1] Pfeil, Maik. "Arb-Scan – 0.5.0" 19 March 2003. URL: http://arbon.elxsi.de/download/arb-scan-0.5.0.tar.gz (8 July 2003)

[2] Phish, Mr. "Banshee-3.3" URL: http://www.blakhat.co.uk/code/BH-banshee/BH-banshee-3.3.tar.gz (8 July 2003)

[3] "SANS Security Essentials V: Windows Security", The SANS Institute, 2003.

[4] "SANS Security Essentials VI: Unix Security", The SANS Institute, 2003.

[5] Beck, Rob. "Passive-Aggressive Resistance: OS Fingerprint Evasion" 1 September 2001. URL: http://www.linuxjournal.com/article.php?sid=4750 (8 July 2003)

[6] Gulker, Chris. "The Kevin Mitnick/Tsutomu Shimomura affair". 17 September 2001. URL: http://www.gulker.com/ra/hack/ (8 July 2003).

[7] Hall, Eric A. "Advanced TCP Options" 8 February 1999. URL: http://www.networkcomputing.com/1003/1003ws1.html (8 July 2003)

[8] Yarochkin, Fyodor. "Remote OS Detection via TCP/IP Stack Fingerprinting". 11 June 2002. URL: http://www.insecure.org/nmap/nmap-fingerprinting-article.html (8 July 2003)

[9] Yarochkin, Fyodor. "Idle Scanning and related IPID games". URL: http://www.insecure.org/nmap/idlescan.html (8 July 2003)

[10] Yarochkin, Fyodor. "Nmap network security scanner man page". URL: http://www.insecure.org/nmap/data/nmap_manpage.html (8 July 2003)

[11] Yarochkin, Fyodor. "Nmap in the News" 13 June 2002. URL: http://www.insecure.org/nmap/nmap_inthenews.html (8 July 2003)

[12] "The Internet Operating System Counter" (web page). April 1999. URL: http://leb.net/hzo/ioscount/ (9 July 2003)

[13] Arkin, Fyodor. "Xprobe v2.0 A Fuzzy Approach to Remote Active Operating System Fingerprinting" August 2002. URL: http://www.sys-security.com/archive/papers/Xprobe2.pdf (9 July 2003)

[14] AloR, NaGA. "ettercap-0.6.b". 10 July 2003. URL: http://prdownloads.sourceforge.net/ettercap/ettercap-0.6.b.tar.gz?download (11 July 2003)

[15] f0bic. "Remote OS Detection using LPD Querying". 3 March 2001. URL: http://www.securiteam.com/unixfocus/5PP041F3PY.html (8 July 2003)

[16] Denis, Frank. "ftpmap-0.4" 13 November 2002. URL: ftp://ftp.pureftpd.org/pub/pure-ftpd/ftpmap/ ftpmap-0.4.tar.gz (8 July 2003)

[17] Bordet, Julien. "Remote SMTP Server Detection" 4 September 2002. URL: http://www.greyhats.org/outils/smtpscan/remote_smtp_detect.pdf (8 July 2003)

[18] Van Houser, Revmoon, DJ. "amap-2.7" 16 June 2003. URL: http://www.thc.org/download.php?t=r&d=amap-2.7.tar.gz (8 July 2003)

[19] Veysset, Coutay and Heen. "New Tool and Technique for Remote Operating System Fingerprinting" Intranode Research Team. April 2002. URL: http://www.intranode.com/fr/doc/ring-full-paper.pdf (9 July 2003)

[20] Beardsley, Tod. "Snacktime: A Perl Solution for Remote OS Fingerprinting" Plan B Security. June 2003. URL: http://www.planb-security.net/wp/snacktime.html (9 July 2003)

[21] "netfilter/iptables - Documentation" URL: http://www.netfilter.org/documentation/ (9 July 2003)

[22] Berrueta, David B. "A Practical Approach for Defeating Nmap OS-Fingerprinting" 2003. URL: http://voodoo.somoslopeor.com/papers/nmap.html (9 July 2003)

[23] Gael Roualland and Jean-Marc Saffroy, *IP Personality*, URL: http://ippersonality.sourceforge.net/ (9 July 2003)

[24] "Trustworthy Computing Initiative", Microsoft website, 3 July 2003. URL: http://www.microsoft.com/mscorp/innovation/twc/ (10 July 2003)

[25] Smith, Donald. "Mscan, Sscan and Synscan - The evolution of worm - enabling vulnerability scanners that span two Millenniums." 20 May 2000. URL: http://www.whitehats.ca/main/publications/external_pubs/scanner_fingerprints/scanner_fingerprints.html (10 July 2003)