Global Information Assurance Certification Paper

A Remote OS Identification Primer
Albert Boyle
December 27, 2001

Introduction

Remote OS Identification is a part of the reconnaissance phase of any targeted network attack. In order to identify weaknesses in a system, an attacker must know what OS the target is running, and what software is installed on it. Once this has been determined, it is usually a simple matter to query a vulnerability database, such as those kept by CERT (http://www.kb.cert.org/vuls/), NIST (http://icat.nist.gov/icat.cfm), SecurityFocus (http://www.securityfocus.com/cgi-bin/vulns.pl), etc., to determine what exploits the target may be susceptible to.

There are many methods of determining the OS of a remote system, from simple "banner grabbing" to sophisticated TCP/IP stack fingerprinting techniques. I hope this paper will serve as a reference for those new to security who wish to know more about remote OS identification, and how to defend against it.

"Classic" Techniques

"Port Scanning"

Port scanning is used to determine what ports are open on a system, and thus what services are listening (and available to be attacked). It can also be used as an OS-identification technique. Systems with port 445 open are most likely Windows 2000 systems, while systems with ports 6000-6063 open are probably running X-Windows, and are most likely some flavor of UNIX.

"Banner grabbing"

Banner grabbing is the simplest and easiest technique. Any telnet client can be used to connect to an open port and see what logon information is advertised. Many telnet, Web, SMTP, and FTP servers proudly display their product name, version, and OS information. The following is an example "banner grabbing" session.

First,[1] we telnet to port 25 to see what SMTP server our target is using:

```
>telnet mail.majorisp.net 25
220-server.majorisp.net ESMTP Exim 3.33 #1 Sun, 09 Dec 2001 20:28:18 -0800
220-NO UCE.  MajorISP does not authorize the use of its computers or network
220 equipment to deliver, accept, transmit, or distribute unsolicited e-mail.
>quit
221 server.majorisp.net closing connection
```

[1] Throughout this paper I have intentionally avoided providing examples of "aggressive" identification techniques such as port scanning and active TCP/IP fingerprinting; an attacker would most likely begin with a full port scan. I have also changed names to protect the innocent.

A quick web search reveals that "Exim is a message transfer agent (**MTA**) developed at the University of Cambridge for use on Unix systems connected to the Internet" [1]. So we know a few things already: the MajorISP SMTP server is UNIX (what flavor or version we don't yet know), the message transfer agent is Exim, and the version is 3.33. A quick perusal of the Exim website reveals complete documentation, including apparently much of the author's design philosophy, etc. A dedicated attacker would certainly find a lot of interesting information here. We've also noted the hostname of the particular server in use.

Next, we examine port 110, POP3:

```
>telnet mail.majorisp.net 110
+OK POPserver vMI_3_36 at majorisp.net ready <22321.1007964598@hostname>
```

Not a lot of information is revealed in this case, as a web search on the POP server identified does not reveal anything useful. The implementation appears possibly to be proprietary. However, we again make note of the server's name (different from the SMTP server in this case).

Telnet to port 80 does reveal some interesting information:

```
>telnet www.majorisp.net 80
><cr>
HTTP/1.1 400 Bad Request
Server: Netscape-Enterprise/3.6
```

Now we know that MajorISP employs Netscape-Enterprise/3.6 on their web servers, and can begin searching for potential vulnerabilities. By examining the "Supported Platforms and System Requirements" of the Netscape Enterprise Server [2], we can begin to narrow down the OS in use, but we still don't know which flavor of UNIX they are running.

Next, we try FTP:

```
>telnet ftp.majorisp.net 21
220-Welcome to the anonymous FTP server at MajorISP, Inc.
220-
220-If your FTP client crashes or hangs shortly after login, try using a
220-dash (-) as the first character of your password.  This will turn off
220-the informational messages which may be confusing your ftp client.
220-
220-If you have any questions, please send mail to ftp@majorisp.net.
220-
220 bob FTP server (Version MISP-FTPD(2) Thu Feb 6 17:05:52 PST 1997) ready.
>SYST
215 UNIX Type: L8
```

Unfortunately, "UNIX Type: L8" is useless in determining the OS version [3]. "bob FTP server (Version MISP-FTPD(2) …)" doesn't reveal much about the FTP daemon in use,

except that it appears to be quite old (1997), and it appears to be something homegrown (or at least modified by the target organization). A web search reveals nothing useful.

So, in this example, my target has done a decent job of hiding the OS versions of their servers, at least against simple banner grabbing. However, we did discover that the target organization runs some variant of UNIX, they use Exim 3.33 for SMTP, and they serve web pages with Netscape's Enterprise server 3.6. In many cases, these techniques will be completely successful, and are the first and easiest method of remote OS identification.

Several other ports might give up useful information, such as 22 (SSH), 23 (telnet), 143 (IMAP), 113 (Identd) etc., depending on what ports were found to be open in earlier port scans.

Email Headers

Email headers can reveal significant details about a target system. If simply connecting to port 25 had not revealed any useful information, we may be able to gather it from the email headers of a message delivered through the target mail system. For example, email sent to an invalid user reveals several server names, as well as the SMTP server software and version (again, Exim 3.33):

```
Status:  U
Return-Path: <>
Received: from server1.majorisp.net ([w.x.y.z])
        by server2 (MajorISP SMTP Server) with ESMTP id u18fup.b73.37tiu4s
        for <user@majorisp.net>; Sun, 9 Dec 2001 20:53:45 -0800 (PST)
Received: from exim by server1.majorisp.net with local (Exim 3.33 #1)
        id 16DIRk-00038m-00
        for user@majorisp.net; Sun, 09 Dec 2001 20:53:44 -0800
X-Failed-Recipients: bogustestemail@majorisp.net
From: Mail Delivery System <Mailer-Daemon@server1.majorisp.net>
To: user@majorisp.net
Subject: Mail delivery failed: returning message to sender
Message-Id: <E16DIRk-00038m-00@server1.majorisp.net>
Date: Sun, 09 Dec 2001 20:53:44 -0800
```

Email sent to an auto-reply address at the target (info@majorisp.net) reveals additional useful information:

```
Status:  U
Return-Path: <sales@majorisp.net>
Received: from server3.earthlink.net ([w.x.y.c])
        by server4 (MajorISP SMTP Server) with ESMTP id u18os0.ikc.37tiu8v
        for <user@majorisp.net>; Sun, 9 Dec 2001 23:25:52 -0800 (PST)
Received: (from autoreply@localhost)
        by server3.majorisp.net (8.9.3/8.9.3) id XAA12332
        for From:  "User Name" <user@majorisp.net>; Sun, 9 Dec 2001 23:25:52 -0800 (PST)
Date: Sun, 9 Dec 2001 23:25:52 -0800 (PST)
Message-Id: <200112100725.XAA12332@server3.majorisp.net>
To: user@majorisp.net
Subject: Re: test request for information
References: <001101c1814c$32810b60$fb01a8c0@majorisp.net>
```

```
In-Reply-To: <001101c1814c$32810b60$fb01a8c0@majorisp.net>
Precedence: auto_reply
X-Loop: autoreply@majorisp.net
From: Sales at MajorISP Inc <sales@majorisp.net>
```

Notice that server3 reveals the text "8.9.3/8.9.3". What would you bet this server is running sendmail 8.9.3? A quick telnet to port 25 confirms this:

```
>telnet server3.majorisp.net 25
220 server3.majorisp.net ESMTP Sendmail 8.9.3/8.9.3; Sun, 9 Dec 2001 23:39:48 -0800 (PST)
```

Again, this does not reveal the underlying OS, but it does present us with another SMTP application to check for vulnerabilities. As Sendmail 8.9.3 is not the current version, it seems likely to be an unpatched and possibly vulnerable system (assuming the sendmail.conf file has not been altered to present false version information). A quick search of the NIST vulnerabilities database reveals several potential weaknesses.

It would be interesting at this point to probe this target's email anti-virus defenses by sending the EICAR test string [4] to both valid and invalid email addresses. The EICAR test string is an innocuous string of characters that anti-virus software detects as a virus, and is used by administrators to confirm that anti-virus implementations are functioning properly. An attacker could include the string in the body of an email, or as an "innocuous" attachment (.TXT for example), inside of an attached archive file, inside of an attached password-protected archive file, or as an email attachment with various file extensions (.EXE, .COM, .SCR, etc.). Responses could reveal the anti-virus configuration, and possibly the software and version.

DNS Query

NSLOOKUP and Dig are two tools used to query DNS servers for various information, from full zone transfers to HINFO and TXT records, to the version of the DNS server itself. TXT records contain "descriptive text" [5], exactly the sort of thing attackers might find useful. For example, an administrator might note the function of a particular server, or some other text that would provide clues to the attacker. HINFO records typically contain the Host's CPU type and OS [6]. Luckily, most DNS administrators are wise to this by now, and so don't use HINFO or TXT records in publicly available zone files. From the attacker's perspective it would be worthwhile checking for these, just in case.

The following is an example of an attempted zone transfer, and an attempted BIND version query:

```
>nslookup
Default Server:  ns.majorisp.net
Address:  w.x.y.z

>ls –d majorisp.net
[ns.majorisp.net]
*** Can't list domain majorisp.net: Query refused
> set class=chaos
```

```
> set type=txt
> version.bind
Server:  ns.majorisp.net
Address:  w.x.y.z.

*** ns.majorisp.net can't find version.bind: Query refused
```

My example target has implemented defenses against this information gathering technique, most likely editing the named.conf to allow zone transfers and BIND version queries only from particular hosts, as described in [7]. As an alternative defense against the version query, they could have altered the version information returned as described in [8].

An interesting area of future research would be to create a "DNS version interrogation cookbook", as envisioned by "Mr. DNS" [9], that could "fingerprint" a DNS server, similar to the way that NMAP (discussed later) fingerprints the TCP/IP stack.

SNMP

An initial port scan would have turned up SNMP services running on a target host at port UDP 161. It's undoubtedly wishful thinking that nobody in the world could possibly be running SNMPv1 on an Internet connected host, and certainly an attacker would find it worth investigating. SNMPv1 relies on "Community Strings" for authentication, and passes all data (including the community strings) in clear text. Default community strings are well known (e.g. "public" and "private"). If Read-only access can be achieved, it is trivial to access OS version information [10], [11]. Tools such as SNMPUTIL from the Windows NT Resource Kit (http://www.microsoft.com), and SNMPWALK (http://www.mkssoftware.com) can be used to gather the information. If read-write access can be achieved, then it's game over.

"Social Engineering"

It should be possible to call the target, pose as a vendor, and ask questions about the number of servers, the OS in use, and so on. Nobody at the target is likely to recognize you as an attacker, as this is a standard line of interrogation used by any pushy sales person. The target may be left feeling as though they need to go wash their hands after talking with you, but they are unlikely to realize that you are gathering information that will be used against them in a network attack.

"Modern" Techniques

Active TCP/IP Fingerprinting

No discussion of Remote OS Detection is complete without referring to Fyodor's famous paper on the topic, "Remote OS Detection via TCP/IP Fingerprinting" [12]. He discusses several "classical" techniques in addition to the TCP/IP Stack fingerprinting methods he has implemented in his tool NMAP.

NMAP is considered to be the best tool available for port-scanning and active TCP/IP fingerprinting, and performs a variety of active probes to detect the remote OS. These are: FIN flag probe, bogus flag probe, TCP initial sequence number sampling, don't fragment bit, TCP initial window size, ACK value, ICMP error message quenching, ICMP message quoting, ICMP error message echoing integrity, type of service, fragmentation handling, and TCP options. Fyodor and others have explained these methods in great detail, so I will not repeat their explanations here. Suffice it to say that, as Thomas Glaser's paper makes clear, NMAP can be used to determine nearly any OS "to a high degree of accuracy" [13].

Other methods of TCP/IP stack fingerprinting exist, such as examination of the TTL value, Maximum Segment Size, and source port [13], [14], [15].

Passive TCP/IP Fingerprinting

One problem with active TCP/IP fingerprinting is that it requires the fingerprinter to send a number of strangely formed packets to the target system. This pattern (and even some of these individual packets) is easily detectable and may alert the administrators of the target system (if they are examining their firewall and/or NIDS logs). Another problem is that an active scan can in some rare cases cause the target system to crash. The goal of passive TCP/IP fingerprinting is to determine the OS without sending any packets at all to the target, or at least nothing out of the ordinary. It can also be used by defenders to help ID attackers [16].

This can be achieved by examining many of the same parameters that are examined in an active fingerprint session: Initial Sequence numbers, TTL, window sizes, don't fragment bit, type of service, etc. can be used to identify the OS.

The following is a syn-ack packet from our example web server, captured with Microsoft Network Monitor:

```
00000:  ww ww ww ww ww ww xx xx xx xx xx xx 08 00 45 00
00010:  00 30 F4 1D 40 00 2F 06 53 D2 yy yy yy yy zz zz
00020:  zz zz 00 50 09 5F A9 6F 12 08 D6 A5 B1 66 70 12
00030:  60 F4 D1 0F 00 00 01 01 04 02 02 04 05 B4
```

From this we see that the TTL is 47 (original TTL is most likely 64), the Don't Fragment bit is set, the Type of Service is 0x0, and the Window size is 24820. Consulting the database of signatures provided by [15], we determine (finally!) that the target system from my previous examples is most likely Solaris, version 8.

Conclusion

There are currently no foolproof countermeasures against remote OS identification. An administrator can remove all banner messages and even close all ports, but an attacker can still determine the OS with a high degree of certainty using active TCP/IP fingerprinting techniques.

It has been suggested [17] that a universal TCP/IP stack be produced and supported. This would seem to be a daunting task, as pointed out in the original suggestion. However, I believe there is another even worse flaw to this approach: homogeneity. If a flaw were found in this proposed TCP/IP stack, and were it deployed on all systems across the Internet, then all systems would be vulnerable to that flaw.

When I started research for this paper, my initial thesis was that TCP/IP stack fingerprinting could be successfully defended against if RFCs were written that covered all possible responses to any TCP or IP packet, and then all vendors followed the RFCs to the letter. After some period of time, older OSes would no longer be used, and newer releases would all appear identical to a remote attacker. However, I think this approach might partly share the same homogeneity problem as the previous approach; exactly specified responses could easily lead to extremely similar code, which may contain similar flaws. And at the same time, it seems likely that different implementations, even of the same exacting RFCs, would still contain subtle differences that could make them vulnerable to stack fingerprinting.

Defenses against remote OS identification are also useless against automated, non-targeted attacks, such as the recent Code Red and Nimbda worms. These do not attempt to perform any remote OS identification, and in fact don't even check if the target port is open; they simply attack everything on the network, regardless of OS. This type of attack will undoubtedly continue and grow more popular in the future, which means that defending against remote OS identification will yield fewer and fewer benefits for defenders. Administrators cannot rely on the anonymity of their systems to protect them. Staying up to date with patches and ensuring proper configuration is still the best defense.

References

[1] Metheringham, Nigel. "The Exim Homepage." http://www.exim.org/.

[2] http://home.netscape.com/enterprise/v3.6/datasheet/index.html

[3] Bernstein, D. J. "The SYST, STAT, HELP, and NOOP Verbs."
http://multivac.cwru.edu/mirror/host/cr.yp.to/ftp/syst.html

[4] http://service2.symantec.com/SUPPORT/ent-security.nsf/docid/19975295056

[5] Mockapetris, P. "Domain Names – Implementation and Specification" (RFC 1035).
http://www.faqs.org/rfcs/rfc1035.html

[6] Farrow, Rik. "How DNS Can Divulge Sensitive Information."
http://www.networkmagazine.com/article/NMG20000515S0096

[7] Jones, LaMont. "Bind Version 8.2.2 is Vulnerable to Root Compromise."
http://www.securiteam.com/unixfocus/3Z5Q2Q0Q0C.html

[8] Beale, Jay. "Foiling DNS Attacks." http://www.bastille-linux.org/jay/defending-dns.html

[9] "Mr. DNS." "Is there a way to query the BIND version?" http://www.acmebw.com/askmrdns/archive.php?category=83&question=37

[10] Maheu, Robert. "Don't Let Hackers Gather Your Information." http://www.sans.org/infosecFAQ/threats/gather.htm

[11] Farrow, Rik. "Keeping SNMP's Secrets Safe." http://www.networkmagazine.com/article/NMG20000515S0127

[12] Fyodor. "Remote OS Detection via TCP/IP Fingerprinting." http://www.insecure.org/nmap/nmap-fingerprinting-article.html

[13] Glaser, Thomas. "TCP/IP Stack Fingerprinting Principles." http://www.sans.org/newlook/resources/IDFAQ/TCP_fingerprinting.htm

[14] Lewis, Jason. "Examining Advanced Remote OS Detection Methods/Concepts using Perl." http://www.packetnexus.com/kb/greyarts/docs/981766898:16776.html

[15] Spitzner, Lance. "Know Your Enemy: Passive Fingerprinting," http://project.honeynet.org/papers/finger/ (also available at http://www.net-security.org/text/articles/spitzner/fingerprinting.shtml).

[16] Siefried, Kurt. "Passive OS Detection and Source Ports." http://www.seifried.org/security/network/20011009-passive-os-detection.html

[17] Cowan, Crispin. "Re: Preventing Remote OS Detection". http://www.securityfocus.com/cgi-bin/archive.pl?id=1&start=2001-12-06&end=2001-12-12&mid=12698&threads=1

[18] McClure, Stuart, Scambray, Joel, and Kurtz, Goerge. Hacking Exposed. Berkeley: Osborne/McGraw-Hill, 1999. 51-55.