Global Information Assurance Certification Paper

TCP/IP Stack Fingerprinting Principles

Thomas Glaser

Introduction

Reconnaissance is a practice used by skilled hackers to size up and gather information about their target. There are several ways to go about gathering any given piece of information regarding a target that would yield vulnerability. One of the most important pieces of information that a hacker could have is the flavor and version operating system of a remote host. With information in regards to the flavor and version of the operating system, a hacker could look for any number of possible vulnerabilities via information on the web that are specific to that operating system and version.

For instance, if someone were to determine that an organization's router to the Internet was running Cisco IOS v11.x, that person could lookup a number of known vulnerabilities at a web site like http://www.securityfocus.com/ that are relevant to IOS v11.x. One such vulnerability that affects IOS v11.x is a denial of service type attack if the router is running a web server. A packet could be crafted such that it would pass the contents: *http://<router-ip>/%%* to the server port with a spoofed source address that would cause the router to crash. Since the source address was spoofed, it would be extremely difficult if not impossible to track down the perpetrator.

The proliferation of several new tools have been made available on the Internet that to a high degree of accuracy can tell the operating system of a host just by examining subtle details in the way the TCP/IP stack was implemented within that operating system. This method is called TCP/IP fingerprinting.

Examples of these tools include nmap and queso. This paper examines the specific methodology used by probably the most popular tool for TCP/IP fingerprinting, nmap.

TCP/IP Fingerprinting via NMAP

Since each developer of an operating system implements TCP/IP a bit differently than another developer of an operating system, different operating system's TCP/IP stack could respond differently given the same situation in a TCP/IP conversation.

At the time of this writing, NMAP interrogates the target machine's TCP/IP stack by sending it eight different packets and observing the response. Each of the eight different packets are specially crafted to put the target machine in a position where there is a high probability that two things will happen:

1. The target operating system's TCP/IP stack will respond unique in comparison to another operating system's TCP/IP stack.
2. The target operating system TCP/IP stack will respond in a consistent manner.

Knowing how a given operating system's TCP/IP stack would respond in advance to each of the eight tests allows NMAP to determine with a high degree of accuracy not only which operating

system the target is running, but also what version it is running as well.

The crafted test packets are sent one at a time by the source machine running nmap. The tests are documented in the table below:

TSeq	A series of SYN packets are sent to the target machine to see how TCP sequence numbers are derived.
T1	A SYN packet with options (WNMTE) set is sent to an open TCP port.
T2	A NULL packet with options (WNMTE) set is sent to an open TCP port.
T3	A SYN,FIN,PSH,URG packet with options (WNMTE) set is sent to an open TCP port.
T4	An ACK packet with options (WNMTE) set is sent to an open TCP port.
T5	A SYN packet with options (WNMTE) set is sent to a closed TCP port.
T6	A ACK packet with options (WNMTE) set is sent to a closed TCP port.
T7	A FIN,PSH,URG packet with options (WNMTE) set is sent to a closed TCP port.
PU	A packet to a closed UDP port.

Several different metrics are observed for each of the first seven tests to help determine the target operating system. They are:

1. Whether or not the target host responded.
2. Whether or not the target host responded with a packet that had the "Don't Fragment" bit set.
3. The Window Size set by the target host in the response packet.
4. The relationship of the acknowledgement number of the TCP packet sent in response to NMAP's prior TCP packet.
5. Flags set in the TCP packet sent in response.
6. TCP options that are in the responding packet.

The first test (Tseq) and last test (PU) uses different metrics that will not be covered in this paper, but the same principles apply.

All of these metrics can measure something different between operating systems and different versions of the same operating system given a certain test. But these metrics are consistent with the same version of a given operating system given any of the tests that NMAP implements.

NMAP holds all of its known operating system fingerprints in a text file called *nmap-os-fingerprints*. There are a few hundred fingerprints documented that include at least one entry for all the popular operating systems. An entry in the file typically looks like:

```
Fingerprint Windows 2000
TSeq(Class=RI%gcd=<5%SI=>BBB&<FFFF)
T1(DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=Y%DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)
```

T4(DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%
DAT=E)

The line that states **Fingerprint Windows 2000** identifies the operating system (and sometimes version) that owns the fingerprint. The next line that begins with a **TSeq,** identifies the method for calculating TCP sequence numbers for a given TCP session. The lines that follow, starting with T1 through PU, are descriptive of how that operating system fingerprinted would respond to the given test.

The tests T1 through T7 are all TCP tests. The symbol '%' delimits the metrics used in fingerprinting. The symbol '|' is used to represent "or" in a given set of answers to a metric to state that a number of given results would satisfy the fingerprint.

The metrics are detailed in the following table:

Resp	**Y** = There was a response **N** = There was no response	Whether or not the host responded to the test packet by sending a reply.
DF	**Y** = DF was set **N** = DF was not set	Whether or not the host responding to the test packet sent the "Don't Fragment" bit in response.
W	Can be a two-byte integer expressed in hexadecimal.	Window advertisement size sent by the host responding to the test packet.
ACK	**0** = ack zero **S** = ack sequence number **S++** = ack sequence number + 1	The acknowledgement sequence number response type.
Flags	**S** = SYN **A** = ACK **R** = RST **F** = FIN **U** = URG **P** = PSH	Indicate what flags were set in the responding packet.
Ops	**M** = MSS **E** = Echoed MSS **W** = Window Scale **T** = Timestamp **N** = No Option	Options sent back by the host responding to the test packet. There can be any number of options set (including none) in any order.

For example, let us take the first test in the previous fingerprint example:

T1(DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)

This test states that the response packet of the target host to NMAP sending a SYN packet with options to it had the following characteristics:

Resp=	Resp is not defined; which means the metric is satisfied whether or not the target replies
DF=Y	The "Don't Fragment" bit was set
W=402E	The window size was 402E
ACK=S++	Acknowledgement number was one plus the initial sequence number
Flags=AS	The packet had the SYN/ACK flags set
Ops=MNWNNT	The packet had the following option flags set in this order: MNWNNT

The following is a trace (using dump) that illustrates how the tests are implemented. There is a source machine running NMAP (10.0.2.3) and a target machine (10.0.2.6) that we would like to test to see if NMAP can find a TCP/IP stack fingerprint for in its fingerprint file.

Source Test Packet 1 (NMAP)	10:37:42.324053 10.0.2.3.34031 > 10.0.2.6.135: S 1338197984:1338197984(0) win 4096 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol> (ttl 59, id 9783)
Target's Response	10:37:42.324518 10.0.2.6.135 > 10.0.2.3.34031: S 2863638239:2863638239(0) ack 1338197985 win 16430 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0> (DF) (ttl 128, id 15245)

The first test packet NMAP sends out is a SYN packet some TCP options set. Looking at the above trace and going through our metrics we can deduce the following:

- The target responded so the metric Resp=Y
- The "Don't Fragment" bit (DF) was set in the target's response so the metric DF=Y
- The window size is 16430; however, in hex that is 402E. So the metric W=402E
- The acknowledgement number equals the source sequence number plus 1. So the metric Ack=S++
- A SYN/ACK packet was sent in response. So the metric Flags=AS
- The TCP options MNWNNT was sent in response. So the metric Ops=MNWNNT.

The fingerprint for T1 (Test 1) would look like:
T1(DF=Y%W=402E%Ack=S++%Flags=AS%Ops=MNWNNT)

Source Test Packet 2 (NMAP)	10:37:42.324315 10.0.2.3.34032 > 10.0.2.6.135: . win 4096 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol> (ttl 59, id 1545)
Target's Response	10:37:42.324718 10.0.2.6.135 > 10.0.2.3.34032: R 0:0(0) ack 1338197984 win 0 (ttl 128, id 15246)

The second test packet NMAP sends out is a NULL packet some TCP options set. Looking at the above trace and going through our metrics we can deduce the following:

- The target responded so the metric Resp=Y
- The "Don't Fragment" bit (DF) was not set in the target's response. So the metric DF=N
- The window size is 0. So the metric W=0

- The acknowledgement number equals the most recent source sequence number. So the metric Ack=S
- An ACK and a RESET were sent in response. So the metric Flags=AR
- There were no TCP options sent in response. So the metric Ops=.

The fingerprint for T2 would look like:
T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)

Source Test Packet 3 (NMAP)	10:37:42.327823 10.0.2.3.34033 > 10.0.2.6.135: SFP 1338197984:1338197984(0) win 4096 urg 0 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol> (ttl 59, id 4649)
Target's Response	10:37:42.328265 10.0.2.6.135 > 10.0.2.3.34033: S 2863675212:2863675212(0) ack 1338197985 win 16430 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0> (DF) (ttl 128, id 15247)

The third test packet NMAP sends out is a SYN/FIN/PSH/URG packet some TCP options set to a known open port. The fingerprint for T3 would look like:
T3(Resp=Y%DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)

Source Test Packet 4 (NMAP)	10:37:42.334937 10.0.2.3.34034 > 10.0.2.6.135: . ack 0 win 4096 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol> (ttl 59, id 6192)
Target's Response	10:37:42.335359 10.0.2.6.135 > 10.0.2.3.34034: R 0:0(0) win 0 (ttl 128, id 15248)

The fourth test packet NMAP sends out is an ACK packet some TCP options set to a known open port. The fingerprint for T4 would look like:
T4(DF=N%W=0%ACK=O%Flags=R%Ops=)

Source Test Packet 5 (NMAP)	10:37:42.340712 10.0.2.3.34035 > 10.0.2.6.32775: S 1338197984:1338197984(0) win 4096 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol> (ttl 59, id 62508)
Target's Response	10:37:42.341113 10.0.2.6.32775 > 10.0.2.3.34035: R 0:0(0) ack 1338197985 win 0 (ttl 128, id 15249)

The fifth test packet NMAP sends out is an SYN packet some TCP options set to a known closed port. The fingerprint for T5 would look like:
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)

Source Test Packet 6 (NMAP)	10:37:42.343991 10.0.2.3.34036 > 10.0.2.6.32775: . ack 0 win 4096 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol> (ttl 59, id 18026)
Target's Response	10:37:42.344416 10.0.2.6.32775 > 10.0.2.3.34036: R 0:0(0) win 0 (ttl 128, id 15250)

The sixth test packet NMAP sends out is an ACK packet some TCP options set to a known closed port. The fingerprint for T6 would look like:
T6(DF=N%W=0%ACK=O%Flags=R%Ops=)

Source Test Packet 7 (NMAP)	10:37:42.349443 10.0.2.3.34037 > 10.0.2.6.32775: FP 1338197984:1338197984(0) win 4096 urg 0 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol> (ttl 59, id 37913)
Target's Response	10:37:42.349840 10.0.2.6.32775 > 10.0.2.3.34037: R 0:0(0) ack 1338197985 win 0 (ttl 128, id 15251)

The seventh test packet NMAP sends out is an FIN/PSH/URG packet some TCP options set to a known closed port. The fingerprint for T7 would look like:
T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)

Adding up the results of the cumulative testing results in the examples shows a fingerprint that matches the fingerprint shown for Windows 2000 above.

Countermeasures to TCP/IP Fingerprinting?

There has been a little bit of talk on the Internet regarding possible counter measures to stack fingerprinting. The two main solutions that come up most often:

1. Use a TCP/IP filtering technique (or firewall) to keep hosts from getting tested or responding to the tests
2. Modifying the TCP/IP stack behavior via option set within the operating system itself.

At this point properly configured TCP/IP filtering techniques seem to provide the best protection.

Operating systems such as Microsoft Windows, Linux, and Sun Solaris have added the ability to modify TCP behavior. Such things as TCP options and window sizes can now be configured. The use of these new features as countermeasures alone have drawn criticism from those who participated in a BugTraq thread called "Preventing Remote OS Detection":

The fact of the matter is that most of these fixes are stopgap measures and not very extensible. The proper fix is standards compliance and ubiquitous support. Of course the question then becomes which standards do we adhere to, and what is to be done if there isn't standard defined behavior for some instance...? That's a good one. I say, close the Internet for repairs.
http://www.securityfocus.com/archive/1/12670
"route"
Feb 22 1999

As route suggested, the true fix is "standards compliance and ubiquitous support" that he and another colleague suggested is near impossible:

You probably can't, at least without a significant, tedious, and error-prone code audit. We've been doing research on OS fingerprinting for the past few years, and there are hundreds of different stack-specific idiosyncrasies.
http://www.securityfocus.com/archive/1/12675
Thomas H. Ptacek
Feb 22 1999

References

[1] Fyodor. “Remote OS detection via TCP/IP Stack FingerPrinting.” October 18, 1998. http://www.insecure.org/nmap/nmap-fingerprinting-article.html
[2] Various. “nmap-os-fingerprints.” October 12, 2000. Included with NMAP tool.
[3] Gilbert, Patrick. “PREVENTING REMOTE OS DETECTION.” http://www.pgci.ca/fingerprint.html (October 12, 2000)
[4] Ptacek, Thomas. “BugTraq Article: Re: Preventing remote OS detection.” February 22, 1999. http://www.securityfocus.com/archive/1/12663 (October 24, 2000)
[5] route. “BugTraq Article: Re: Preventing remote OS detection.” February 22, 1999. http://www.securityfocus.com/archive/1/12670 (October 24, 2000)
[6] Information Sciences Institute. “RFC793.” September 1981. ftp://ftp.cis.ufl.edu/pub/ietf/rfc/rfc793.txt
[7] Microsoft Corporation. “Description of Windows 2000 TCP Features.” December 29, 1999. http://support.microsoft.com/support/kb/articles/Q224/8/29.ASP (October 24, 2000)
[8] Hall, Eric. “Advanced TCP Options.” February 8, 1999. http://www.networkcomputing.com/1003/1003ws1.html (October 24, 2000)

Last Updated: May 26th, 2017

Upcoming Training

SANS Stockholm 2017	Stockholm, Sweden	May 29, 2017 - Jun 03, 2017	Live Event
Community SANS Ottawa SEC401	Ottawa, ON	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS San Francisco Summer 2017	San Francisco, CA	Jun 05, 2017 - Jun 10, 2017	Live Event
Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS Charlotte 2017	Charlotte, NC	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Rocky Mountain 2017 - SEC401: Security Essentials Bootcamp Style	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
SANS Secure Europe 2017	Amsterdam, Netherlands	Jun 12, 2017 - Jun 20, 2017	Live Event
Community SANS Portland SEC401	Portland, OR	Jun 12, 2017 - Jun 17, 2017	Community SANS
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Minneapolis 2017	Minneapolis, MN	Jun 19, 2017 - Jun 24, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Cyber Defence Canberra 2017	Canberra, Australia	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Minneapolis SEC401	Minneapolis, MN	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Phoenix SEC401	Phoenix, AZ	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Munich Summer 2017	Munich, Germany	Jul 10, 2017 - Jul 15, 2017	Live Event
Mentor Session - SEC401	Ventura, CA	Jul 12, 2017 - Sep 13, 2017	Mentor
Mentor Session - SEC401	Macon, GA	Jul 12, 2017 - Aug 23, 2017	Mentor
Community SANS Atlanta SEC401	Atlanta, GA	Jul 17, 2017 - Jul 22, 2017	Community SANS
Community SANS Colorado Springs SEC401	Colorado Springs, CO	Jul 17, 2017 - Jul 22, 2017	Community SANS
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANSFIRE 2017 - SEC401: Security Essentials Bootcamp Style	Washington, DC	Jul 24, 2017 - Jul 29, 2017	vLive
Community SANS Charleston SEC401	Charleston, SC	Jul 24, 2017 - Jul 29, 2017	Community SANS
Community SANS Fort Lauderdale SEC401	Fort Lauderdale, FL	Jul 31, 2017 - Aug 05, 2017	Community SANS
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event