New Tool And Technique For Remote Operating System Fingerprinting

– Short Paper –

Franck Veysset, Olivier Courtay, Olivier Heen, *Intranode Research Team*

April 2002; v1.1

***Abstract* – Information gathering is an essential part of acute vulnerability assessment, especially when the whole process is automated. In this context, host operating system detection must be precise, in particular when networks are well defended. We present an original Operating System detection method, based on temporal response analysis. The 'RING' open source tool – for Remote Identification Next Generation – was developed as a proof of concept. We also stress the interesting synergy of using RING together with state-of-the-art tools, such as NMAP [1] or X-Probe [2], for a better overall accuracy.**

***Keywords* – Remote Operating System Detection, Operating System Fingerprinting, Automated Vulnerability Assessment, Internet Security.**

Introduction

In recent years, the need for automated Internet vulnerability assessment software has been identified and has resulted in the very fast growth of widely available solutions.

As an essential part of the assessment process, remote Operating Systems detection, a.k.a. *OS Fingerprinting*, must meet several requirements:

- **Accuracy**: no falsely detected OS.
- Firewall and IDS **neutrality**: not be disturbed by / do not disturb existing firewalls and IDS.
- **Politeness**: low network traffic and no dangerous segments.
- **Handiness**: easily extensible signature database and automation functions.
- **Speed**: depending on the usage, a fast fingerprinting tool might allow large network scans**.**

We introduce a new OS Fingerprinting method, with such good properties and fairly acute results in practical cases where other tools may fail.

We developed open source software called RING for both *proof of concept* and test purpose. Moreover, we strongly believe that complete access to source code will encourage and speed-up collaborative improvements. RING relies on a signature database that may be enhance, thanks to the built-in learning mode.

State-of-The-Art

A brief history of OS Detection

Security assessors already have a choice of detection techniques and tools, each of which may be suitable for some application context.

- **Banner grabbing** allows OS deduction from services banner and is appreciated by most security assessors. This can be completed by binary file collect and analysis for better accuracy.
- **TCP segments** (standard or not) response analysis relies on different Operating System responses to specifically prepared segments, particularly when response behavior is not clearly specified in RFCs [3]. Furthermore, vendors have introduced fine-tuning and proprietary extensions into their TCP/IP stack, which will clearly identify those systems in case of such solicitations. Popular

© Intranode Software Technologies – 2002

tools such as Savage's QueSO [5] and Fyodor's NMAP [1] use many variants of this technique.

- **ICMP replies** analysis consists in sending UDP or ICMP solicitations and analyzing various ICMP responses. A tool such as Ofir Arkin's X-Probe [2] may give precise indications except if needed protocols are blocked at firewall level.
- **Initial Sequence Number** statistical analysis exploits differences in TCP stacks random generators, identified through a sufficient number of tests [6].

For a most comprehensive description of various techniques, see also [7].

Recalling TCP/IP timeout principle

TCP is a connected mode, reliable protocol. So, among other mechanisms, hosts react to lost segments by regenerating them after an adapted timeout. Segment regeneration may happen in various states of the TCP transition diagram.

As an example the SYN_RCVD state is reached at the very beginning of a tentative TCP connection (see figure 1). If no ACK segment is received before timeout expires, the system will generate a new SYN-ACK segment.

Sometimes, simply regenerating one segment will *not* permit connection process continuation. The responding host will then assume network congestion, wait a bit longer, and tentatively regenerate more segments, and so on [4].

Fig. 1: Simplified TCP/IP state diagram and transitions

For more information about TCP state transition diagram, see [8]. RING takes advantage of this regeneration mechanism for Operating System detection.

RING: Principle & Implementation

Principle description

As timeout values and regeneration cycles are loosely RFC specified, almost each OS uses its own method and set of values (see figure 2). Moreover, OS that share the same IP stack technologies may have slightly adapted timeout values.

Resend SYN-ACK	Microsoft® Windows 2000	FreeBSD 4.4
1st	3 sec.	3 sec.
2nd	6 sec.	6 sec.
3rd	No more retries	12 sec.
4th	No more retries	24 sec.
Reset	No Reset Sent	Reset after 30 sec.

Fig. 2: Stressing implementation differences between some Operating Systems

By forcing timeout, then carefully measuring delays between successive SYN-ACK resents and before optional Reset, and by comparing measures to a reference suite, one may identify the possible Operating Systems.

Important things to checks are:

- Signature suites are contrasted enough.
- Signature suites are obtained under constant network conditions.
- Network conditions during measures don't vary too much.

Here are the four steps of a typical identification session:

1. RING sends a SYN segment to an open port over the target, exactly as would occur for a normal TCP connection.
2. The target shifts from LISTEN state to SYN_RCVD state while sending back a SYN-ACK segment.
3. RING ignores the SYN-ACK segment, and do not send the normally awaited ACK segment.
4. According to TCP state transition diagram, the target remains in state SYN_RCVD while reinjecting SYN-ACK segments, from time to time. RING measures times between these segments (see figure 3).

Fig. 3: Sending SYN, then ignoring SYN-ACK replies

RING Implementation

First, let us stress that using no more than common tools (*iptable*, *tcpdump*, *sendip*) and a Linux host, it is possible as well as easy to perform manual testing.

RING merely automates this method:

- Blocking incoming target segments, thanks to filtering functions controlled through Dug Song's *libdnet*.
- Listening incoming segments and measuring time intervals. RING does this through *libpcap* (from Berkeley National Laboratory) as modified for NMAP [1]. Note that blocking segments don't inhibit listening functions.
- Sending a single SYN segment to an open TCP port on the target host.

Distance used in RING for suite comparison is a straightforward adaptation of usual distance between series: $\Sigma|\delta_i - \lambda_i|$ where δ_i = signature values and λ_i = measured values.

The guessed Operating System is the one whose recorded signature is closest to measured values.

Practical Results & Improvements

Firewall & IDS matters

Most representative security architecture encountered while performing automated assessment, is a well configured firewall, opening no more than a few "needle holes" (i.e. selected open ports on selected internal or service hosts) to protect hosts and applications.

No closed TCP / UDP port is visible through the firewall. Moreover, it is likely that entering ICMP traffic is blocked.

This common configuration prevents X-Probe detection, as needed segments are not allowed to reach the target. NMAP accuracy will also decrease, as it generally benefits from open and close port usage, and no close port is visible. TCP sequence number analysis is likely to be interpreted as a synflood attack and blocked at firewall level.

By sending **only one standard segment** through the needle hole, RING is not disturbed by the firewall. Moreover, target replies are no more than standard SYN-ACK segments that will cross the firewall (if not, normal traffic wouldn't be possible).

As only standard TCP dialog occurs during both learning and detection stages, it is likely that no IDS can block or even detect RING.

One good protection against RING, as well as against other tools, may be the use of SYN relaying functions. In this case, the firewall will partially or totally undertake the TCP connection process. RING will then detect the firewall instead of the protected host.

Further improvements

Other states in TCP transition diagram show similar behavior, trying to reinject supposedly lost segments. This is the case for FIN_WAIT_1 state that can be used to corroborate previous OS deductions and / or bypass some SYN defenders.

Further researches on this topic could concern independence in regard to network performance variation and global robustness of RING detection. Repetitive measures with some aberrant value detection may help in cases where the network is very unsteady. We also may use the timestamp TCP field in some cases, in order to calculate precise duration between successive segments sending.

Lastly, we feel that known signature database must grow, and guess that open source developer's community will help. We encourage sending comments and newly found signatures to ring@intranode.com

Conclusions

RING uses a brand new Operating Systems detection technique, that relies on very common and noiseless TCP traffic. Automated vulnerability assessment engines may greatly benefit form RING, especially when used in conjunction with other techniques (see figure 4).

For further reading and information concerning RING, a full paper can be found at this URL:

www.intranode.com/site/techno/ring-full-paper.pdf

The open source version of RING, with associated libraries, man page and an evolutive signature database can be found at this URL:

www.intranode.com/site/techno/techno_articles.htm

Any comment or suggestion may be sent to the alias ring@intranode.com

Main References

[1] Fyodor, *NMAP*, www.insecure.org/nmap

[2] Arkin, O., *X-Probe*, www.sys-security.com/html/projects/X.html

[3] Postel, J. (Sep, 1981), *RFC 793 – Transmission Control Protocol*

[4] Comer, D and Lin, J. (1994), *Probing TCP implementations* www.cs.purdue.edu/homes/lin/probe.tcp.html

[5] Savage, *QueSO*, savage.apostols.org/projects.html

[6] Zalewski, M. (Apr, 2001), *Strange Attractors and TCP/IP Sequence Number Analysis*

[7] Veysset, F. (Jun, 2001), *13th Annual FIRST Conference – OS Fingerprinting Revisited*

[8] Stevens, W. R. (1994), *TCP/IP Illustrated, Vol. 1*

The authors would like to thank Fyodor, D. Fort, P. Auffret, F. Frade, A. Floch and C. Patel for valuable comments.

OS Fingerprinting Techniques Comparison	Banner Grabbing	Non standard segments	ICMP replies	ISN Sampling	Temporal analysis
History					
Classical implementation	Plenty	NMAP	X-PROBE	?	RING
Created by	Hackers	Fyodor	Ofir Arkin F. Yarochkin	M. Zalewsky Guardent	Intranode
First released	?	Jun, 1998	Aug, 2001	Apr, 2001	Mar, 2002
IP Protocol & Service					
Used protocols	Service related	IP, TCP, UDP & ICMP	UDP & ICMP	TCP	TCP
Open TCP port required	Service related	The more the better	No	Yes	Yes
Closed TCP port required	No		No	No	No
Closed UDP port required	No		Yes	No	No
Firewall concerns					
Bypass filtering routers	Always	Generally	Rarely	Generally	Generally
Bypass SYN relays	Always	Rarely	Always	Never	Never
Bypass application proxies	Possible	Never	Never	Never	Never
Outgoing firewall neutral	Always	Generally	Generally	Always	Always
IDS concerns					
Detection	Hard	Easy	Possible	Possible	Hard
Blocking	Hard	Possible	Possible	Possible	Hard
Misc.					
Learning functions	No	Yes	No	?	Yes
KB size on Marsh 2002	?	> 600	~ 20	?	~ 30
Target hosts disturbance	None	Rare	None	Possible	None
Best match feature	No	Yes	?	?	Yes
Defensive measures	Banner rewriting	Firewall or host stack tuning	ICMP blocking at firewall	SYN Relaying	SYN Relaying or host stack tuning

Fig. 4: Possible synergies mixing most efficient techniques and tools