POSFinger: A NEW FINGERPRINT FOR PASSIVE REMOTE OPERATING SYSTEM IDENTIFICATION

Ahmed Mansour Manasrah, Muhammad Fermi Pasha, Rahmat Budiarto, Sureswaran Ramadass
National Advanced IPv6 (NAv6) Center
Level 6, School of Computer Sciences Building, University of Sains Malaysia
11800 Penang, Malaysia
{ahmad, fermi, rahmat, sures}@nav6.org

Abstract

In this paper we propose POSFinger a fingerprint for identifying remote operating system as a proof-of-concept. Passive operating system detection is the art of detecting the operating system of other PC remotely without its knowledge. In order to achieve this goal the detection process should be fast enough and reliable whereby the prediction possibilities should narrow down to match the most proper operating system. POSFinger is the software that utilizes the idea of passive operating system fingerprinting by mapping the active methods to be mapped passively to achieve better performance and reliable prediction. Maintaining a secure operating system presents a great challenge for any security or network administrator.

Key words: *passive operating system fingerprinting, Operating system fingerprint, Operating system fingerprint identification,*

1. Introduction

When a user takes illegal control over any information system, this consider as intrusion. Whether or not the action is harmful, it is of concern because it might cause instability to the system or the services provided by the system, and since today's systems consist of multiple nodes executing multiple operating systems in a form of single distributed system, this only changes the complexity, but not the fundamental problems.

In fact, intrusion is not a problem that can be solved once; continual aware is required. Complete physical isolation of a system from all is possible, but it may be unacceptable. Other approaches cannot be used because they are in conflict with other service. It is typically that the operating system does have vulnerabilities that can be exploited. New vulnerabilities discovered daily in each operating system upgrade. Patching the system could eliminate known vulnerabilities. And some would prevent desired functionality. Even though black hat user's attempts to exploit security vulnerabilities in both the operating system (OS) and the applications running on top of it.

To date, system administrators have relied on vendor patches to ensure the safety of there systems but these fixes are seldom available in a timely fashion and fixing one problem often creates new ones. [3]. Black hat users require operating Systems information and the patch level as first step in gaining control of the remote host. And thus, Understanding how to obtain remote OS information is the key to protect it. Detecting remote operating system could be the first step in building a stronger intruder detection system or honeyNet, recalling the fact that most of the security holes depends on the operating system version running on the host, then its wise enough to know how to patch or how to protect the system from illegal users since they could reach that point of detecting the operating system and thus detecting system vulnerabilities [1].

2. Related Work

Many techniques available for fingerprint network stack, Basically, each operating system has different behavior then others in terms of packet content and fields values; and thus by looking for these different behaviors among operating systems and building a database for these differences will absolutely narrow down the operating system estimations very tightly. There are only two techniques in OS fingerprinting; Active and passive techniques.

For Active techniques, all previous work consider active tools such as Nmap in [2,3,4,5] results in categorizing Nmap into "random scan" and "exploits plus". NMAP is a network exploration tool and security scanner that can find all open ports and detects the OS on hosts within an IP address range. Franck Veysset in [6] identify RING tools to one of the unique tools because it uses only one open port and a standard TCP packets instead of malformed packets to avoid machine disturbance and make it impossible to be detected by any IDS system, and give it the stealth nature. QUESO as described in [7] is a port scanner and operating system detection tool that usually sends out seven packets at a time. QUESO sets the SYN | FIN flags to tell the receiving computer to OPEN and CLOSE a connection once.

From the other side a passive OS fingerprinting tools such as p0f [8] consider as a tool to identify the operating system by examining packets being passed over the local network, without sending any packets. It can identify the operating system on machines that connect to your box (SYN mode), machines you connect to (SYN+ACK mode), machine you cannot connect to (RST+ mode), and machines whose communications you can observe. Preston Wood in [9] describe a passive IP discovery and fingerprinting utility designed to sit on segments distributed throughout a network to discover unique IP's on the network. In addition to IP discovery disco has the ability to passively fingerprint TCP SYN packets and TCP SYNACK packets.Ettercap is another tool developed by Alberto Ornaghi, Marco Valleri. It is a suite for man-in-the-middle attacks on LAN. It features sniffing of live connections, content filtering on the fly and many other interesting tricks. It supports active and passive dissection of many protocols (even ciphered ones) and includes many features for network and host analysis.

As a result of these various tools, it is concluded that Ring takes long time for identification compared to NMAP. Moreover Ring tool need certain initialized parameter in order to operate; such as the target host IP address, an open port, the scanner machine IP address and of course the network interface to be scanned . Nmap still in active nature which is detectable by any firewall that can banned it, moreover Nmap require a good conditions to perform the test such as one opened TCP port, one closed one and a closed UDP port. sprint contains advanced banner grapping functionality which could be disabled by the administrator in addition to the active nature of the tool itself. And as a result, a POSFinger is a combination between both active and passive techniques to work in passive nature for more accurate and less false positive detection results, in the coming section the POSFinger system component description will be endorsed.

3. Passive OS Fingerprinting

Instead of crafting packets, listening and capturing packets during normal communication is the core of passive OS identification. Analyzing the captured packets will help in determining the operating system that creates those packets. In contrast with active OS fingerprinting that is based on sending crafted packets to motivate the host to reply. Two states considered for better results in identifying the OS; the normal connection request and the connection establish request.

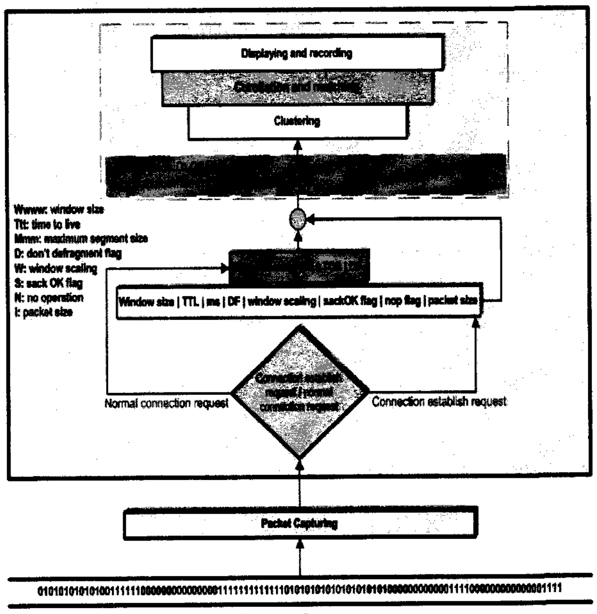


Figure 1: main component

3.1 Normal Connection Request

Detailed information must be collected from captured packets such as Window size, TTL, ToS, and DF values to minimize the possibilities of the fingerprint. To assist in constructing a proper format to be maintained into a tree data structure format for fast matching. And since the TTL is not constant because it depends on the hops number it passes a temporary matching could be accepted but not to ignore the accuracy of matching; this is where the Host Path Projection take place by projecting the path after applying the common rules whenever it comes to TTL values; The rule of thumb when dealing with TTL values is :

$$(\min TTL \geq currentTTL \leq \max TTL + 1).$$

Packet information received from www.lib.usm.my
Window: 9112 / TTL: 232 / ToS: 0 / DF: ON

Table 1: host path projection

TTL values	TTL Good Match
32	0 - 32
64	33 - 64
128	65 - 128
255	129 - 255

Suppose a packet with TTL value of 232, by comparing this value against the table above, we will see that 232 is between 129 – 255 and then the corresponding approximate TTL value is 255 which will give a match for (x86 Solaris Operating System (Solaris,x86,9112,255,0,1)). The solution is not to relay on a single field value, whereas relaying on multiple fields value is more reliable and will to narrow down the operating system guessing possibilities. And thus a file with known OS fingerprint matches is maintained within the system. In summary a combination of Window, TTL, TOS, and DF will be tracked to perform matching and to narrow down the possibility of guessing the operating system, at least one TCP packet and one IP packet needed in order to gather the necessary information.

3.2 Connection Establish request (3-way handshaking)

It is not necessary that all packets contain the same fields of information. Once the fields are extracted {*window size* / *time to live* / *maximum segment size* / *don't fragment flag* / *window scaling* / *sackOK flag* / *nop flag* / *packet size*} the proper OS fingerprinting file is loaded to perform matching against it. Basically both cases are the same but differ in the number of fields used and the format.

4. Passive Operating System Identification (POSFinger)

Recalling that Passive Operating System identification relies on the fact that each operating system TCP/IP stacks implementation differs slightly. This is most accurately done by analyzing TCP SYN, TCP SYN+ACK packets, as these have the greatest number of unique identifiers. The fields used to make this determination are (Initial TTL, Don't Fragment flag, Initial SYN Packet Size, and TCP Options, such as Window Size, MSS Size, Nop Option, Window Scaling Option and SackOK Option).

4.1. Analysis of TCP Options values.

This example from a SYN packet generated from Windows 98 box [10]

Table 2 TCP Option Filed

<mss 1460,nop,nop,sackOK>
4500 0030 ae17 4000 8006 f398 xxxx xxxx yyyy yyyy 0c37 0015 0526 a98f 0000 0000 7002 ffff 6f06 0000 0204 05b4 0101 0402 **Maximum Segment Size (MSS):** 4 bytes **Window Scale (WSCALE):** 3 bytes **Timestamp (TS):** 10 bytes **No Operation (NOP):** 1 byte **Selective Acknowledgment Permitted (SackOK):** 2 bytes **Selective Acknowledgment Data**: 10 bytes (plus 8 bytes for each additional pair of sequence numbers)

Considering another example of a SYN packet content from OpenBSD 2.7 box

Table 3: Sample TCP Option Fields

<mss 1460,nop,nop,sackOK,nop,wscale 0,nop,nop,timestamp 50473 0>
The following describes the additional 16 bytes of fields used in this sample. Offers 1 byte of padding. The operating system supports the Window Scale option but it is not used. This takes up 3 bytes. <nop,nop> supplies 2 bytes of padding. <timestamp 50473 0> is the first timestamp. The Timestamp Value field is set to 50473. The Timestamp Echo Reply field is set to 0 since this is the initial packet. The timestamp option takes 10 bytes.

In order to achieve a better comparison between different OpenBSD X boxes another example captured from OpenBSD 2.9, which produce no TCP option values as follow:

Table 4: Sample Packet Format

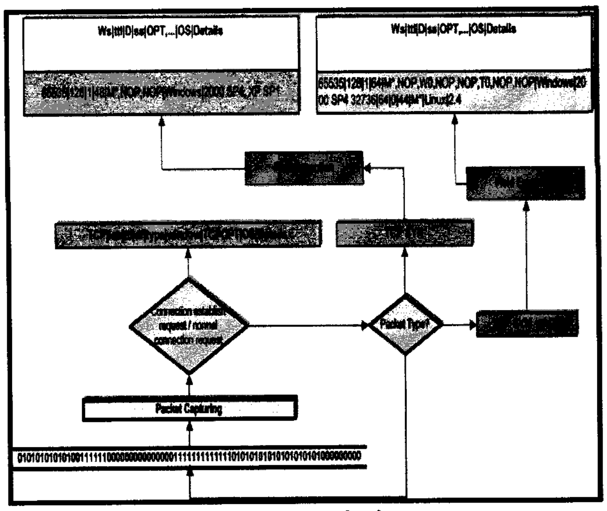
<1145: P 3066603742:3066603806(64) ack 1646168027 win 17520 [tos 0x10]>
t4510 0068 7e87 4000 4006 3862 c0a8 011e c0a8 0128 0016 0479 b6c8 a8de 621e 87db 5018 4470 1813 0000 e492 152f 23c3 8a2b 4ee7 dbf8 0d48 88e8 0110 2b01 4295 39f4 52c9 a05b 31d7 e3ae 1c62 2dbd d955 d604 b5d2 63d1 8fbc 4ab7 1615 b382 571c 70e0 a368 a03f 425b 6211

- The P, ack, and 1646168027 are TCP flags.
- The 3066603742:3066603806(64) is the byte sequence/range.
- The win 17520 indicates the window size in bytes that the sender is presently ready to receive.
- The [tos 0x10] is the TCP type of service.

Another example of a SYN packet content from a RedHat Linux 7.0 system is <mss 1460, sackOK, timestamp 123265 0, nop, wscale 0>. This example also employs the same 4 options at the beginning which is used by the previous examples. The difference in this example is the arrangements of the fields in a dissimilar way. And thus, the operating system has the ability to specify the order of the TCP options fields.

4.2. The Proposed Fingerprints and their Meanings

After reviewing some famous passive operating system detection tools, POSFinger comes up with new strategy in matching against fingerprints and identifying operating systems by considering three different states:


Ws|ttl|D|ss|OPT,...|OS|Details
Ws|ttl|D|ss|OPT,...|OS|Details
65535|128|1|64|M*,NOP,W0,NOP,NOP,T0,NOP,NOP|Windows|2000 SP4 32736|64|0|44|M*|Linux|2.4
Connection establish request / normal connection request
Packet Type?
Packet Capturing
010101010101001111110000000000000000011111111111110101010101010101010100000000
Figure 2: fingerprinting cases

4.2.1 SYN Fingerprint.

Defining the TCP SYN Packets as the default mode of the fingerprinting operation, because SYN Packets are rich in information which are operating system dependant such as

- Window size (WSS) –The value defined depends on the system itself where it could be fixed value, multiple of MSS or MTU (MSS + 40), or random numbers.
- Initial TTL – The real TTL value is determine once a packet is received.
- Maximum segment size (MSS) – This setting is used to identify the link type of the remote host.
- Window Scaling (WSCALE) – Some system uses this to scale WSS which is able to lengthen the TCP/IP window size about 32 bits.
- Timestamp – In some cases the initial SYN is set to zero if the timestamp feature is used.
- Selective ACK permitted – To implement this function a flag is set.
- NOP option – This is used as a separator between different option values to indicate the sequence of these values.
- EOL option – Normally the system uses NOP option to pad at the end of the packet and the EOL option is rarely used.

Table 5: SYN Fingerprint Format

Ws|ttl|D|ss|OPT,...|OS|Details	
ws	- window size.
ttl	- initial TTL.
D	- don't frangment bit (0 – not set, 1 –set).
Ss	- overall SYN packet size.
OPT	- option value and order specification.
OS	- OS Name.
Details	- OS description.
Where the OPT is a comma separated list and could take the following format,	
NOP	- NOP option
EOL	- EOL option
WX	- window scaling option with value X.
MX	- maximum segment size option with value X.
SACKOK	- selective ACK OK
T	- timestamp
65535|128|1|48|M*,NOP,NOP|Windows|2000 SP4, XP SP1	

4.2.2 SYN + ACK Fingerprint

The other way to identify the TCP/IP signature from the packets sent out (SYN + ACK packets). The SYN + ACK fingerprint depends on the SYN value in some cases. If the SYN has enabled the extensions such as [RFC1323], [RFC2018] and [RFC1644], then only it could be revealed. In some cases the WSS value is duplicated from the SYN packets so a different WSS value is used to test this kind of values. The proposed fingerprint for these packets as follow:

Table 6: SYN+ACK Fingerprint Format

Ws|ttl|D|ss|OPT,...|OS|Details	
Ws	- window size.
ttl	- initial TTL.
D	- don't frangment bit (0 – not set, 1 –set).
Ss	- overall SYN packet size.
OPT	- option value and order specification.
OS	- OS Name.
Details	- OS description.
Where the OOO is a comma separated list and could take the following format,	
NOP	- NOP option
EOL	- EOL option
WX	- window scaling option with value X.
MX	- maximum segment size option with value X.
SACKOK	- selective ACK OK
T	- timestamp
65535|128|1|64|M*,NOP,W0,NOP,NOP,T0,NOP,NOP|Windows|2000 SP4 32736|64|0|44|M*|Linux|2.4	

4.2.3 General Fingerprint.

Another way to identify TCP/IP signature, by considering the normal communication and investigate the content of interchanged packets. It is not accurate as much as the previous two techniques, the result could be near to prediction and more then one possible guess could be displayed instead of the exact one. Each fingerprint consists of one line, depending on the packet type, the fingerprint format as follow.

Table 7: Normal Communication Fingerprint Format

TCP|len|ttl|df|type|window|TCPOPT|OS|Details	
TCP	- TCP Protocol.
Len	- Packet Length.
ttl	- initial TTL.
df	- don't frangment bit (0 – not set, 1 –set).
Type	- Packet type (SYN,ACK,SYN+ACK)
window	- window size.
TCPOP	- TCP Options-if available
OS	- OS Name.
Details	- OS description.
Where TCPOP is a comma separated list and could take the following format,	
NOP	- NOP option
EOL	- EOL option
WX	- window scaling option with value X.
MX	- maximum segment size option with value X.
SACKOK	- selective ACK OK
T	- timestamp
TCP|48|128|1|SYN|8192|M*,NOP,NOP|Windows| 98SE	

The default state is the SYN fingerprint which is the accurate guess, then the SYN+ACK then the general fingerprint which is the least accurate among all these states.

5. Result and discussion

An analysis of five different operating systems is conducted which are:

5.1 Linux Box.

Let's Consider the SYN packet contents of a Linux 2.4 operating system to identify the OS. There are two cases for this identification:

Case 1: Linux 2.4 uses 5840 bytes as its initial Window size which is different compared to Linux kernel 2.2 that uses 32120 as its initial Window size. Linux kernel 2.2 and 2.4 uses default TTL value which is 64.

The following TCP Options are used by default by Linux 2.4 and 2.2 as shown in the figure above.

- Maximum Segment Size (MSS)
- Selective Acknowledgment OK (sackOK)
- Timestamp
- Window Scale, and Nop

As discussed above, the TCP Option fields contribute to the total length of a packet that could assist in determining an OS type. The total length of the packet shown in the figure above is 60 bytes. Thus, Linux is easy to be identified as it is the only OS that contains total header length of 60 bytes.

Case 2: Putting Linux Fingerprinting Together:
As a result, the characteristic listed below recognizes only one Operating System which is Linux.

- TTL: 64
- Windows: 5840
- TCP Options: Sets MSS, Timestamps, sackOK, wscale and 1 nop
- Packet Length: 60

5.2 OPENBSD Box

The following describes on the OpenBSD characteristics.

- TTL: 64. This value alone will not help to detect OpenBSD.
- Window Size: 16384. This value is also used by other operating systems.
- TCP Options: Uses the same options as Linux but instead of setting one nop, OpenBSD uses 5 nops
- Total Packet Length: 64 Bytes. This is a key indicator.

As a result, a system which uses OpenBSD can be identified by using the characteristics listed above.

5.3 SOLARIS 7

Solaris can be identified easily by using its initial TTL field. The figure below shows the packet contents of a Solaris 7 operating system. The following list the characteristics of the Solaris 7 operating system.

- TTL: 255
- Window Size: 8760
- TCP Options: MSS
- IP ID: Increments by one ALL of the time.
- Total Packet Length: 44 bytes

A Solaris packet can be identified by using both the TTL value which is 255 and the total packet length which is 44 bytes.

5.4 AIX 4.3.

The following list the characteristics of an AIX 4.3.

- TTL: 64. Resembles Linux and OpenBSD
- Window: 16384. Resembles Windows 2000
- TCP Options: MSS. Like Solaris
- Packet Length: 44 bytes. Like Solaris.

As a result, the AIX 4.3 operating system can be identified by comparing with the characteristics listed above.

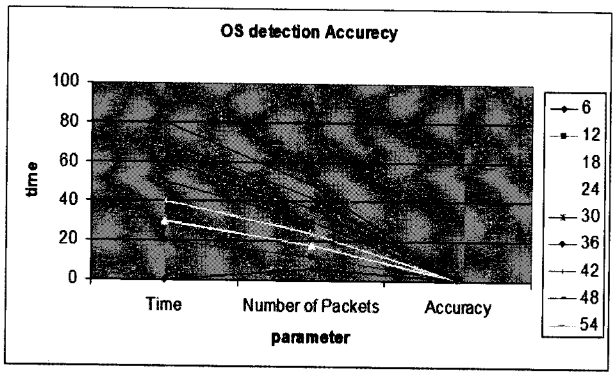
5.5 Window 2000

The following list the characteristics of Window 2000 operating system.

- TTL: 128. Nearly all Windows Operating systems uses this TTL value.
- Window: 16384 - Like AIX 4.3
- TCP Options: MSS, sackOK, 2 nops

As a result, Window 2000 packet can be identified by using the packet length along with the TTL value to assist in giving an accurate result.
The accuracy in detection summarized in the following table

Time	Number of Packets	Accuracy
0	6	50%
20	12	51%
30	18	52%
40	24	70%
50	30	71%
60	36	72%
70	42	72%
80	48	73%
90	54	73%



6. Conclusion

POSFinger is utilizing the idea of Active Operating System fingerprinting by investigating the fingerprint signature that active tools used such as NMAP, and enhancing this fingerprint signature by eliminating the non-operating system dependant fields such as the IP options field, the type of service field, and the identification filed, by this POSFinger minimize the time of comparison by reducing number of parameter to compare against.

References

[1] "Passive Aggressive", By Jon Lasser Jan 30 2002; http://www.securityfocus.com/
[2] "PLUGGING LEAKY HOLES; Port scanners provide an efficient means for finding soft spots on a network's digital perimeter"; June 2001;; BY GARY C. KESSLER
[3] "Remote Host Detection and OS Fingerprinting"; Debasis Mohanty; Orissa, INDIA.
[4] RFC 793 pp. 64
[5] "New Tool And Technique For Remote Operating System Fingerprinting"; Full Paper.
[6] Franck Veysset, Olivier Courtay, Olivier Heen, Intranode Research Team; April 2002, v1.1
[7] Internet, 2004, http://www.safemode.org/sprint/
[8] "The new p0f: 2.0.5", (C) Copyright 2000-2004 by Michal Zalewski (lcamtuf@coredump.cx)
[9] Internet, 2004, http://www.altmode.com/disco/
[10] Understanding tcpdump, Operating System, OpenBSD 2.9, http://www.aei.ca/~pmatulis/pub/tcpdump.html