An Overview of OS Fingerprinting Tools on the Internet

Ruoshi Li, Markus Sosnowski, Patrick Sattler*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: ruoshi.li@tum.de, sosnowski@net.in.tum.de, sattler@net.in.tum.de

***Abstract*—Operating System (OS) fingerprinting is an important technique that can be used to identify OS. Currently, there are many fingerprinting tools on the internet. This paper offers an overview of some popular tools for OS fingerprinting including Nmap, Xprobe, p0f, Satori etc. We introduce the differences between their mechanisms and information gains. We also introduce some papers that show the automation of OS fingerprinting is possible. The accuracy and efficiency of the fingerprinting technique can be improved using machine learning.**

***Index Terms*—operating system fingerprinting**

1. Introduction

Operating system (OS) fingerprinting can recognize OS that is running on the host. It can be used in the enterprise network to detect malware, which works through virtual machines [1]. OS fingerprinting is also an important skill in the penetration test, which is performed to assess system security. Since different OSs have their unique vulnerabilities, the attack method can be determined after the OS is identified. Another use of OS fingerprinting is to detect outdated OS versions that contain vulnerabilities [2]. It is also useful in generating network statistics and research.

This paper provides an overview of some popular OS fingerprinting tools. In section 2 of this paper, some necessary background knowledge is introduced. In section 3, the mechanism of some popular tools for OS fingerprinting and the information that can be gained with them are briefly introduced. These include active fingerprinting tools (Nmap, Xprobe2, RING), passive fingerprinting tools (p0f, Ettercap, Satori, NetworkMiner) and other tools that can be used for fingerprinting (SinFP, ZMap, Scapy). Since most of them work manually, some research focusing on automation of OS fingerprinting is also introduced in section 4. These papers show that the process of gathering, normalizing information, and classifying OSs can be automated with higher accuracy. Finally, a brief conclusion is in section 5.

2. Background

The name of OS fingerprinting vividly describes the way it works. Just like we can uniquely identify a human through the human fingerprint, we can identify an OS by the information contained in the packets it sends. For example, the values of Time-To-Live (TTL) field are one of the indicators. While in the RFC 791 a prescribed value for TTL has not been defined, different OSs then determine their own TTL values [3]. A recommended default TTL is given in RFC 1700 [4], but it is not followed in most implementations. A signature can then be generated uniquely from integrating the different values for every single OS. Fig.1 shows the TTL values and window size values of popular OSs. Through comparing the generated signature to the signatures stored in the database, an OS can be identified.

OS	TTL	Window Size (bytes)
Linux 2.4 and 2.6	64	5840
Google customized Linux	64	5720
Linux kernel 2.2	64	32120
FreeBSD	64	65535
OpenBSD, AIX 4.3	64	16834
Windows 2000	128	16834
Windows XP	128	65535
Windows 7, Vista, and Server 8	128	8192
Cisco Router IOS 12.4	255	4128
Solaris 7	255	8760
OS X	64	65535

Figure 1: Popular OSs' Time-To-Live (TTL) and window size values [1]

OS fingerprinting has two types: active and passive. With active OS fingerprinting, specific packets are sent to the target host, and the information inside the response packets is analyzed. Passive OS fingerprinting sniffs the network traffic rather than sending packets to the hosts. According to their different ways of working, their advantages are complementary and their information gains are also different. Active OS fingerprinting has higher accuracy and is easier to operate. In contrast, passive OS fingerprinting is more concealed and can reduce the likelihood of being discovered.

3. Different kinds of fingerprinting tools

In this section, popular tools are categorized and introduced based on active fingerprinting, passive fingerprinting and other types.

3.1. Active fingerprinting

We introduce Nmap, RING, Xprobe and its evolution Xprobe2 as examples of active fingerprinting tools.

doi: 10.2313/NET-2020-11-1_14

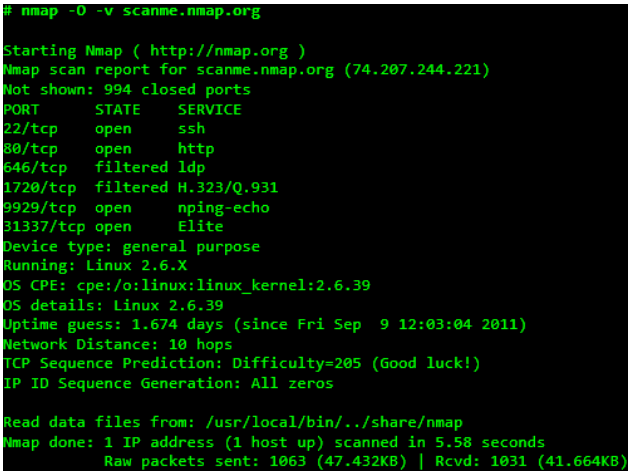

Figure 2: The output of OS detection with verbosity [5]

3.1.1. Nmap. Network Mapper (Nmap) is the most popular tool that can be used for active OS fingerprinting. Nmap has two effective OS detection methods. One is OS detection with verbosity, and the other is using a version scan to detect the OS. We can receive the output shown in Fig.2 with the following information [5]:

1. Device type. This field indicates the type of device, e.g. router, firewall.

2. Running. This field shows the OS Family and OS generation.

3. OS CPE. This field shows the Common Platform Enumeration (CPE) representation of the OS or the hardware. CPE is a naming standard for IT products and platforms, which is machine-readable [6].

4. OS details. This field represents the detailed description for each matched fingerprint.

5. Uptime guess. This is a guess because Nmap can't ensure its accuracy.

6. Network Distance. Nmap can compute the number of routers between it and a target host.

7. TCP Sequence Prediction. This field shows the Initial Sequence Number (ISN) generation algorithm and the difficulty of how hard it is to attack the host using blind TCP spoofing.

8. IP ID Sequence Generation. This field indicates the algorithm of how the host generates the lowly 16-bit ID field in IP packets.

In the case of no fingerprinting matching, the "Service Info" field might reveal the type of OS and Nmap provides the most similar result. Using a fuzzy approach can let Nmap guess more aggressively and offer a list of possible matches with their confidence level in percentage. Nmap also offers detailed information about the host with subject fingerprints, when it cannot identify the host or when we force it to print. An example of a subject fingerprint is shown in Fig.3.

The scan line of the subject fingerprint describes the conditions of the scan, which helps to integrate the fingerprints in the database. Nmap conducts five response tests through sending up to 16 special designed probes (13 TCP, 2 ICMP, 1 UDP). Below the scan line, the response lines are related to the sent probes. The gained information in each line according to the tests is shown in Fig.4.

```
SCAN(V=5.05BETA1%D=8/23%OT=22%CT=1%CU=42341%PV=N%DS=0%DC=L%G=Y%TM=4A91CB90%
     P=i686-pc-linux-gnu)
SEQ(SP=C9%GCD=1%ISR=CF%TI=Z%CI=Z%II=I%TS=A)
OPS(O1=M400CST11NW5%O2=M400CST11NW5%O3=M400CNNT11NW5%
    O4=M400CST11NW5%O5=M400CST11NW5%O6=M400CST11)
WIN(W1=8000%W2=8000%W3=8000%W4=8000%W5=8000%W6=8000)
ECN(R=Y%DF=Y%T=40%W=8018%O=M400CNNSNW5%CC=N%Q=)
T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)
T2(R=N)
T3(R=Y%DF=Y%T=40%W=8000%S=O%A=S+%F=AS%O=M400CST11NW5%RD=0%Q=)
T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)
IE(R=Y%DFI=N%T=40%CD=S)
```

Figure 3: A subject fingerprint [5]

Test	Line name	Result
Sequence generation	SEQ	GCD, SP, ISR, TI, II, TS, SS
	OPS	O1-O6
	WIN	W1-W6
	T1	R, DF, T, TG, W, S, A, F, O, RD, Q
ICMP echo	IE	R, DFI, T, TG, CD
TCP ECN	ECN	R, DF, T, TG, W, O, CC, Q
TCP	T2-T7	R, DF, T, TG, W, S, A, F, O, RD, Q
UDP	U1	R, DF, T, TG, IPL, UN, RIPL, RID, RIPCK, RUCK, RUD

Figure 4: The result obtained by each line classified by the test

In the sequence generation test, six TCP probes are sent. In the response line "SEQ", seven results are shown. Among them, the first three results (TCP ISN greatest common divisor (GCD), TCP ISN counter rate (ISR), TCP ISN sequence predictability index (SP)) are related to the ISN generation algorithm. TI, II and SS (Shared IP ID sequence Boolean) are related to the IP ID sequence. TS is about the TCP timestamp option algorithm. The response line "OPS" contains the TCP options of the six received packets (O1-O6) and the "WIN" line includes the TCP initial window sizes of them (W1-W6). The line "T1" indicates values in the received packets that responses to the first probe. The results are introduced together with the TCP test below.

Next is the ICMP echo test. SS represents the result of testing whether the IP ID sequence is shared between TCP and ICMP. During the test two ICMP echo requests are sent to the host. R represents responsiveness. The value is true when reply is received. DFI indicates the Don't Fragment bit for ICMP. T means initial Time-To-Live, and TG represents its guess if Nmap can't get the value of T. CD represents ICMP response code.

And then is the TCP explicit congestion notification (ECN) test. ECN is a method that intends to reduce network congestion without dropping packets and improve network performance [7]. Nmap sends an SYN packet that sets the ECN CWR (Congestion Window Reduced) and ECE (ECN-Echo) congestion control flags in this test. The value of CC (explicit congestion notification) indicates the setting of CWR and ECE congestion control flags in the response SYN/ACK packet.

Afterwards is the TCP test. Six TCP probe packets with specific TCP options data are sent to the host. Except for R, T, TG, W and O which have already introduced above, DF (Don't Fragment), S (TCP sequence number), A (TCP acknowledgement number), F (TCP flags), RD

(TCP RST data checksum), and Q (TCP miscellaneous quirks) are also recorded as results. For RD, Nmap checks if the host returns data like error message in the reset packets. Q indicates the result for checking two quirks in some implementation.

Finally, in the UDP test, a UDP packet is sent to a closed port. IPL (IP total length) indicates the length of the response packet. UN (Unused port unreachable field nonzero) represents the result of checking whether the last four bytes in the header is zero or not. RIPL (Returned probe IP total length value) indicates the value of the returned IP total length. RID (Returned probe IP ID value) represents if the probe IP ID value has been altered by the host. RIPCK (Integrity of returned probe IP checksum value) shows if the IP checksum still matches the packet, although it has been altered by network hops during transit. RUCK (Integrity of returned probe UDP checksum) shows if the UDP checksum remains the same. RUD indicates the integrity of returned UDP data.

3.1.2. RING. Nmap has a great performance under the condition of one opened, one closed TCP port and one closed UDP port. RING is a tool as a patch against Nmap that can be used when there's only one opened port. RING exploits the mechanism of packet retransmission in TCP three-way handshake, which has not been considered in other tools. An OS's signature can be established by forcing timeouts and then measuring the delay between the packet retransmission or analyzing the information such as TCP flags, sequence number or acknowledge number [8].

3.1.3. Xprobe. Xprobe is based on analysis of ICMP instead of TCP. Xprobe has advantages when the differences between the TCP implementation are subtle, for example, some Microsoft based OS. The number of datagrams that Xprobe sends is very small. Thus Xprobe is stealthier. It can send only one datagram and receive the corresponding reply and then recognize up to eight different OSs [9].

Xprobe uses the following methods to fingerprint OS: ICMP error message quoting size, ICMP error message echoing integrity (based on the different implementations of IP total length field, IPID, 3bits flags and offset fields, IP header checksum, UDP header checksum, precedence bits issues with ICMP error messages), DF bit echoing with ICMP error messages, the IP time-to-live field value with ICMP messages, using code field values different from 0 with ICMP echo requests, and TOS echoing [10].

3.1.4. Xprobe2. The tools like Nmap and Xprobe rely on a static decision tree to perform the results of identification. This approach reduces accuracy because of network topology or the nature of the fingerprinting process itself. For instance, a packet might be affected while in transit, or the user can alter some characteristics of a TCP/IP stack's behaviour [11]. To improve the problem, Xprobe2 utilizes a fuzzy approach with OS fingerprinting. The fuzzy approach is a matrix-based fingerprinting matching approach using the OCR (Optical Character Recognition) technique. The results are shown in a matrix that includes the scores (from 0 (NO) to 3 (YES)) of each test for each OS, and a total score will be calculated for each OS. However, Nmap has already implemented this fuzzy approach [12].

Xprobe2 allows users to modify the modules used for testing. In general, there are seven modules loaded: ICMP echo, time-to-live distance, ICMP echo, ICMP timestamp, ICMP address, ICMP info request and ICMP port unreachable. After the tests, Xprobe2 offers a list of possible OSs with their corresponding probability in percentage sort from the highest to the lowest.

3.2. Passive fingerprinting

The most well-known passive fingerprinting tool p0f and other tools such as Ettercap, Satori and NetworkMiner are introduced in this section.

3.2.1. p0f. The name of p0f is the acronym for passive OS fingerprinting. Almost all passive fingerprinting tools today reuse p0f for TCP-level checks [13]. Though p0f can never be as accurate as Nmap, because its analysis is based on the packet sent by the host itself, it is still an ideal and precise tool and can be helpful when Nmap is confused.

Except source IP address, port, the TCP flag, OS, network distance, uptime, the output also offers a raw signature. The 67-bit signature consists of information of window size (16 bits), initial time-to-live (8 bits), maximum segment size (16 bits), "Don't fragment" flag (1 bit), window scaling option (8 bits), sackOK option (1 bit), nop option (1 bit), initial packet size (16 bits) [12].

3.2.2. Ettercap. Ettercap is an open-source tool and is "a multipurpose sniffer/interceptor/logger for switch LANs" [14]. It is popular for detecting man-in-the-middle attacks and enables OS fingerprinting. As a generalist, Ettercap's function of OS fingerprinting is not as precise as p0f. This information can be gained by Ettercap's fingerprinting: IP address, port, network distance, hostname, device type, fingerprint and OS.

3.2.3. Satori. Satori is a tool based on the analysis of DHCP. The advantage of DHCP is unicast. One of the methods is to use the difference of actual time, seconds elapsed, transaction ID fields of the captured packages to recognize OSs. Another method, also the main method of Satori, is to analyze the parameters of DHCP's option 55, which indicates the parameter request list. Other options can also be used to help to analyze specific OS, for example, option 51 (IP Address Lease Time) and option 57 (Maximum DHCP Segment Size) for Linux [15]. The lease of DHCP can also be exploited to recognize OSs [16]. As the result of OS fingerprinting, Satori offers IP address, fingerprint and a list of possible OSs with their weight up to 12. The higher the weight, the bigger the possibility.

3.2.4. NetworkMiner. NetworkMiner is an open-source tool for Windows that can be used as a passive network sniffer/packet capturing to detect OSs [17]. NetworkMiner uses the database from p0f, Ettercap, and Satori. It also makes use of the MAC-vendor list from Nmap. The information is shown according to hosts but not to packets or frames on NetworkMiner. IP address, MAC address,

vendor, hostname, OS, time-to-live, network distance and open TCP ports of each host can be gained using NetworkMiner. The results of OS fingerprinting are listed according to different databases, containing guesses with their confidence interval in percentage.

3.3. Others

There are other types of tools that also can be used for OS fingerprinting. Some of them work with the help of the functions of the tools that we have introduced in section 3.1 and 3.2. Some combine active and passive fingerprinting. We briefly introduce them in this section.

3.3.1. SinFP. SinFP is a hybrid OS fingerprinting tool that is active and passive. It is designed for addressing the increasingly strict limitations and accurately identify OSs in worst network conditions. For active OS fingerprinting, only three standard requests will be sent to the target, and these standard-compliant frames ensure that the responses are replied from the right target but not the devices in-between. After receiving the responses, the active signature with 15 elements (5 elements for each of the three packets) can be established. The 5 elements are respectively: a list of constant values, TCP flags, TCP window size, TCP options and MSS (Maximum Segment Size). For passive OS fingerprinting, SinFP implements the method that modifies a passive signature and then compares it with active signatures. The database of SinFP has strict conditions for each signature. The principle is that the response is not from devices in-between and the target has at least one open TCP port [18].

3.3.2. ZMap+p0f. ZMap is a free and open-source network scanner that has a very fast scanning speed. ZMap can scan the whole public IPv4 address space within 45 minutes with a gigabit Ethernet connection [19]. To achieve OS fingerprinting, one possible way is to run ZMap and to launch p0f in the background at the same time.

3.3.3. Scapy. Scapy is a program that can be used for packet manipulation. It supports active and passive OS fingerprinting by using the functions of Nmap and p0f. On the other hand, it can actively fingerprint Linux kernel 2.4+ through establishing a TCP three-way handshake with the target and then sending a segment with no TCP flags and arbitrary payload. If the response set the flag ACK, then the target is Linux server (2.4+ kernel) because no other well-known current OS accepts this kind of segment and this behaviour is hard to alter for Linux. [20]

4. Machine Learning applied to OS fingerprinting

Machine learning can realize the automatic process of OS fingerprinting. In [21], Aksoy et al. develop a mechanism that classifies OS with high accuracy using machine learning. They collect header information of IP, ICMP, UDP, DNS, HTTP, IGMP, TCP, FTP, SSH and SSL protocol manually to train the classifiers, instead of using

Type	Tool	Protocol
Active fingerprinting	Nmap	TCP, ICMP, UDP
	RING	TCP
	Xprobe	ICMP
	Xprobe2	
Passive Fingerprinting	p0f	TCP
	Ettercap	TCP
	Satori	DHCP
	NetworkMiner	Combines p0f, Ettercap and Satori
Others	SinFP	TCP
	Zmap+p0f	Use p0f
	Scapy	TCP + functions of Nmap and p0f

Figure 5: A brief conclusion of the protocols that are used by each tool

available OS fingerprinting approaches. The classifiers can successfully detect OSs automatically. Since the approach sniffs packets passively, it is seen as passive fingerprinting, but the approach can also apply for active fingerprinting.

In [22] Schwartzenberg automates the process of updating the database and recognizing the new OSs with the help of machine learning. The database of p0f has been updated manually all the time, and this is a complicated work. The accuracy of p0f decreases because the database has not been updated for new OSs. The approach in [22] attempts to address this problem. The automatic process of extracting the characteristics and normalizing the information is proved to be possible. In addition to this, after using the traditional approach to identify OS, the unrecognizable OS can be classified with higher accuracy using the process that implemented machine learning and trained by existing system set.

Anderson and McGrew [2] present a new approach that integrates TCP/IP, HTTP and TLS features to identify OSs. This approach uses a machine-learning classifier and can identify minor versions with an accuracy of 97.5%. Even under the condition of applying obfuscation, the performance of this approach is robust. The accuracy decreases to 94.95% against an obfuscation level of 25%.

5. Conclusion

According to the introduction above, we can find out that Nmap sends more probes and gained more information than others. Therefore, the accuracy of Nmap is higher. Different from Nmap, Xprobe2 is based on ICMP and implements a fuzzy approach. RING is more like a patch for Nmap, it detects OSs in another point of view. But since Nmap is popular, there are already many measures against Nmap. Hence, the tool like SinFP helps to steer by the limitations. In the field of passive OS fingerprinting, p0f is already a very mature tool. Other passive OS fingerprinting tools reuse p0f more or less. Fig.5 briefly concludes the protocols that these tools use for fingerprinting. Using the machine learning technique, the OS fingerprinting process can be automated to increase accuracy and efficiency.

This paper only offers a part of an overview of TCP/IP OS fingerprinting. For instance, Nmap has specific tests for IPv6 fingerprinting. The mechanism and the information gained by it differ. Further work can focus on

doi: 10.2313/NET-2020-11-1_14

a wider overview of fingerprinting tools, that use the characteristics of IPv6 to recognize OS.

References

[1] R. Tyagi, T. Paul, B. S. Manoj, and B. Thanudas, "Packet inspection for unauthorized os detection in enterprises," *IEEE Security Privacy*, vol. 13, no. 4, pp. 60–65, 2015.

[2] B. Anderson and D. McGrew, "Os fingerprinting: New techniques and a study of information gain and obfuscation," in *2017 IEEE Conference on Communications and Network Security (CNS)*, 2017, pp. 1–9.

[3] J. Postel, "Internet Protocol," Internet Requests for Comments, RFC Editor, RFC 791, September 1981. [Online]. Available: https://tools.ietf.org/html/rfc791

[4] J. K. Reynolds and J. Postel, "Assigned Numbers," Internet Requests for Comments, RFC Editor, RFC 1700, October 1994. [Online]. Available: https://tools.ietf.org/html/rfc1700

[5] G. F. Lyon, "Chapter 8. remote os detection," https://nmap.org/book/osdetect.html, [Online; accessed 18-June-2008].

[6] "About cpe - archive," https://cpe.mitre.org/about/, Mar 2013, [Online; accessed 18-June-2008].

[7] D. B. K. Ramakrishnan, S. Floyd, "The Addition of Explicit Congestion Notification (ECN) to IP," Internet Requests for Comments, RFC Editor, RFC 3168, September 2001. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3168.txt

[8] F. Veysset, O. Courtay, and O. Heen, "New tool and technique for remote operating system fingerprinting," 2002.

[9] O. Arkin and F. Yarochkin, "X remote icmp based os fingerprinting techniques," August 2001.

[10] O. Arkin, "A remote active os fingerprinting tool using icmp," vol. 27, no. 2, pp. 1–6, April 2002.

[11] O. Arkin and F. Yarochkin, "A 'fuzzy' approach to remote active operating system fingerprinting," August 2002.

[12] C. Peikari and A. Chuvakin, *Security Warrior*. USA: O'Reilly & Associates, Inc., 2004.

[13] M. Zalewski, "p0f v3 (version 3.09b)," https://lcamtuf.coredump.cx/p0f3/, 2012, [Online; accessed 19-June-2008].

[14] A. Ornaghi and M. Valleri, "Ettercap," https://www.ettercap-project.org/, [Online; accessed 19-June-2008].

[15] S. Alexander and R. Droms, "DHCP Options and BOOTP Vendor Extensions," Internet Requests for Comments, RFC Editor, RFC 2132, March 1997. [Online]. Available: https://tools.ietf.org/html/rfc2132

[16] E. Kollmann, "Chatter on the wire: A look at dhcp traffic," September 2007.

[17] N. AB, "Networkminer," https://www.netresec.com/?page=NetworkMiner, [Online; accessed 18-July-2008].

[18] P. Auffret, "Sinfp, unification of active and passive operating system fingerprinting," *Journal in Computer Virology*, vol. 6, pp. 197–205, August 2010.

[19] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications," in *22nd USENIX Security Symposium (USENIX Security 13)*. Washington, D.C.: USENIX Association, Aug. 2013, pp. 605–620. [Online]. Available: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/durumeric

[20] P. Biondi, "The art of packet crafting with scapy," https://0xbharath.github.io/art-of-packet-crafting-with-scapy/, [Online; accessed 18-July-2008].

[21] A. Aksoy, S. Louis, and M. H. Gunes, "Operating system fingerprinting via automated network traffic analysis," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 2502–2509.

[22] J. Schwartzenberg, "Using machine learning techniques for advanced passive operating system fingerprinting," 2010.