Technical Report

Operative System fingerprinting

By

Vsevolod Ivanov

40004286

A report submitted in fulfilment of the requirements of ENGR-411

Concordia University

April 2020

ABSTRACT

By the means of this proposal, I would like to determine precise ways of fingerprinting an Operative Systems over the network. As we move forward with a steady increase of IoT (Internet of Things), self-driving cars and satellite constellations, the network interconnectivity is rapidly increasing. In order to leverage the analysis of a network, there should be a precise way of identifying the connected entities. The scope of this problem is my local network and its devices. The same methodology could be applied to any other network with perhaps, some variations. The challenge is to identify an Operative System up to a certain probability. In order to solve this issue, I would like to demonstrate a method of OS fingerprinting to uncover a device identity. First, I will determine a network tool to identify a system using OS scanning techniques. Then, I will plan the analysis of a specific device that I will set up on a clean machine. The schedule of these steps will be defined in terms of the time it takes me to set up a lab environment and learn OS scanning techniques to attest firmly the results accuracy. For the budget, I will aim for FOSS (Free and Open Source Software) to ensure anyone can perform this task. The qualifications of the networking tool will be verified based on standards of Computer Security industry and availability for Arch Linux, my desktop Operative System. Finally, I will determine with high certitude the targeted OS known beforehand from a blackbox point of view.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Purpose

In a continuously increasing technological century, there are emerging devices almost every day. From IoT (Internet of Things) to Wearables in Fashion, all of these devices are meant to be integrated into our daily routine. To leverage the complexity of the hardware and to reduce costs, these machines are often provisioned & controlled through the network. This technological variety makes us wonder what kind of devices are present in a network. An Operative System name and version can tell us a lot about the device identity. In network security, the identification of a machine found on a network is called OS fingerprinting. I would like to propose a solution to determine a way of fingerprinting an Operative System over the network.

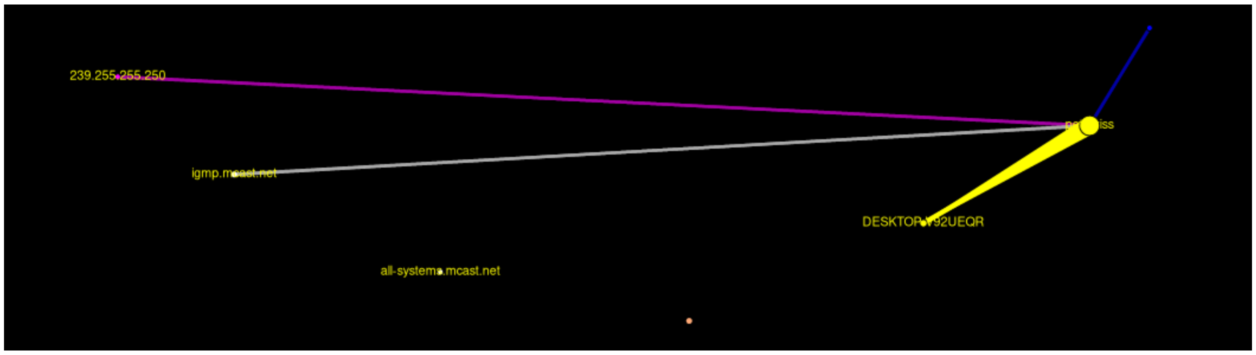
1.2 Background

The key problem is a lack of standard procedure for querying the Operative System details. The latter is meant to be that way to avoid increasing visibility of the attack vectors. The processes running in an Operative System are dependent on its type and purpose. Therefore, vulnerabilities found in applications and protocols are often bound to a specific system. For instance, a personal laptop won't be running a HTTP server like a dedicated IoT device such as an IP Camera serving the video feed and controls over the network to be accessed via a browser. Identifying a system over a network would involve a probabilistic deduction based on the exposed information of the system in question.

The second challenge would require a proper choice of scanning tools and techniques to attest up to a certain probability the identity of an Operative System. In order to confirm the outcome, the produced fingerprint will be compared with the actual system known beforehand.

1.3 Scope

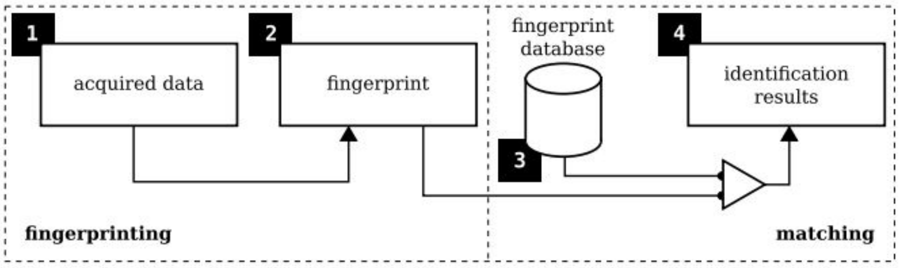
The scope would be my personal network with two devices connected. This network will allow me to control the filtering and to avoid committing any involuntary illegal acts during this network scanning to identify the target device. The current location of my network is at home. I have full access to my router to ensure completion by debugging of any encountered issues with full network controls. Considering the complexity and the extent of a multitude of OS scanning techniques, my report will remain vulgralized to cover the entire process rather than the bit-wise details of each probing technique. In the image below, we have a network visualisation using Etherape tool during the scan of my second target machine represented with a yellow line.



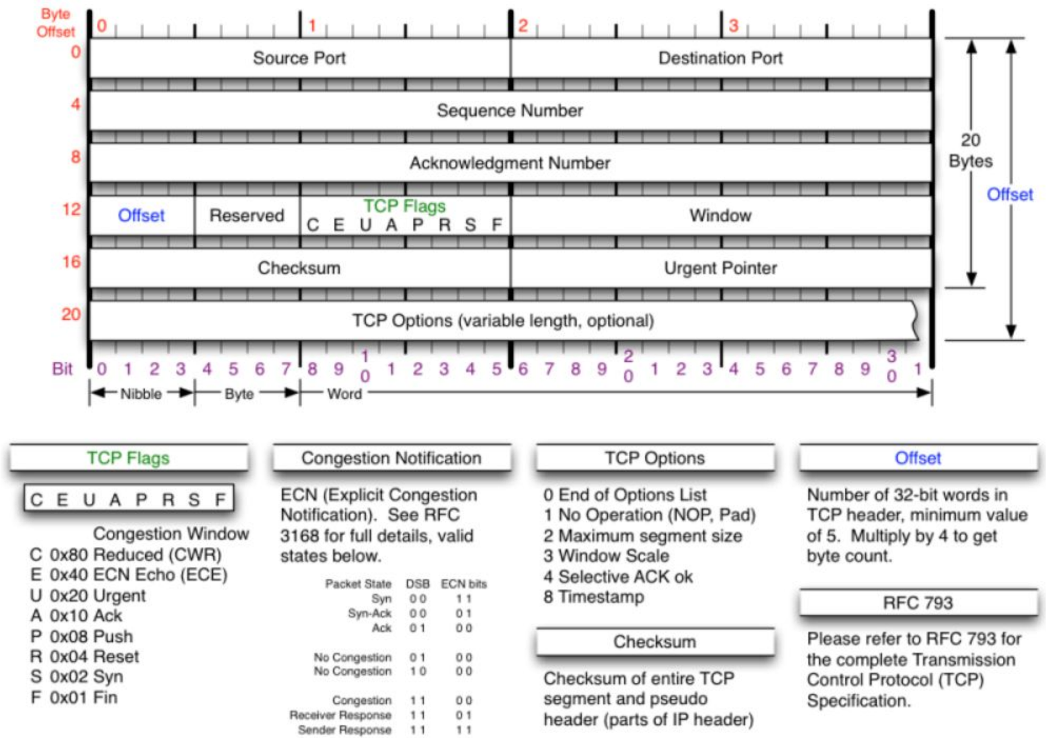
2. SOLUTION CRITERIA

2.1 Approach

To solve the OS Fingerprinting problem, I will use Nmap (Network Mapper) a free and open source network scanning tool developed by Gordon Lyon. This terminal tool is an industry standard to perform passive and active scanning methods. A passive scan avoids direct contact with AP (Access Point) and listens to its beacons (packets) sent periodically. In our study case, we will perform active scans which will interact directly with the AP by sending probes to the host and listening to the replies. One of the most researched techniques for OS detection is TCP/IP stack fingerprinting. This feature is activated by the -O (Enable OS detection) argument passed to Nmap command line binary. More precisely, Nmap will send TCP probes to our Windows 10 host and examine the bits of interest [1] in the response packets. Here is the Nmap diagram covering the internal process of OS fingerprinting:



There are dozens of built-in tests [1] that will be performed including but not limited to TCP ISN (initial sequence number), TCP ISN ordering, IP total length, TCP timestamp, TCP flags, etc. The basic idea is to analyse various options specified in the request according to a well-known OS behavior. At the following page we have a TCP header [2] with a legend explaining various fields. This packet structure gives a good overview of the variety of information that we can specify in the request that will influence the response leveraging the Operative System fingerprinting. Here is an image of the TCP header with various fields and their meanings:



TCP Flags

C E U A P R S F

	Congestion Window
C	0x80 Reduced (CWR)
E	0x40 ECN Echo (ECE)
U	0x20 Urgent
A	0x10 Ack
P	0x08 Push
R	0x04 Reset
S	0x02 Syn
F	0x01 Fin

Congestion Notification

ECN (Explicit Congestion Notification). See RFC 3168 for full details, valid states below.

Packet State	DSB	ECN bits
Syn	0 0	1 1
Syn-Ack	0 0	0 1
Ack	0 1	0 0
No Congestion	0 1	0 0
No Congestion	1 0	0 0
Congestion	1 1	0 0
Receiver Response	1 1	0 1
Sender Response	1 1	1 1

TCP Options

0 End of Options List
1 No Operation (NOP, Pad)
2 Maximum segment size
3 Window Scale
4 Selective ACK ok
8 Timestamp

Checksum

Checksum of entire TCP segment and pseudo header (parts of IP header)

Offset

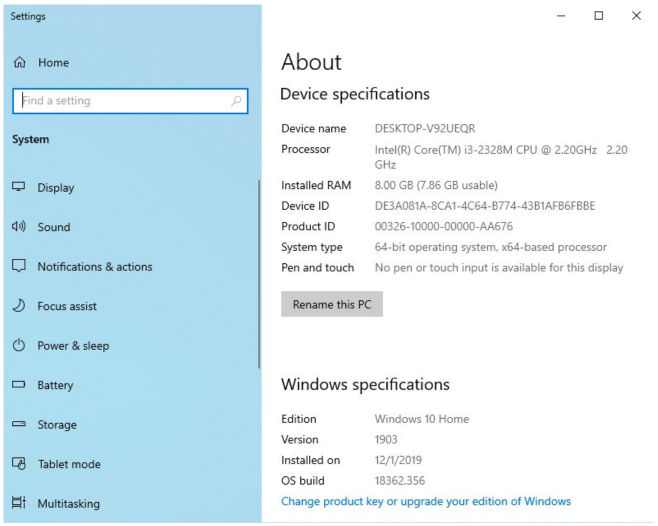
Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

RFC 793

Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.

For the scope of this Technical Report, I will remain on the surface covering the general OS Fingerprinting behavior and the underlying logic. Once the probes are received, it will produce an TCP/IP stack fingerprint. If we activated the probabilistic guessing with --osscan-guess argument, Nmap will compare the fingerprint to its nmap-os-db database with more than 2’600 Operative System fingerprints and it will output the probabilities of being each one of them.

This output will contain a textual description as well as a classification with vendor name, underlying OS with its generation and the device type. For the reproducibility of the results, I decided to set up a computer from scratch with a common Operative System. On my personal computer I'm running Arch Linux. However, it's rather uncommon for an individual to run a Linux distribution. Therefore, I decided to download and install Windows 10 Home edition as if I was a new user coming to this OS in our current times. In fact, most of the computers today come with a pre-installed Windows because this home edition is completely free. This report will not only provide results with high fidelity but it will underline a major issue present for most of the Windows users. This issue is in the lack of protection of the OS identity which makes it easy for malicious people to target specifically vulnerabilities found in Windows rather than being in a very large OS spectrum. Here is a capture of the installed system that we will be scanning. We can see by looking at the Device name DESKTOP-V92UEQR that this installed system corresponds to the target found in the previous Etherape capture which was visualizing my home network during Nmap scan.



About

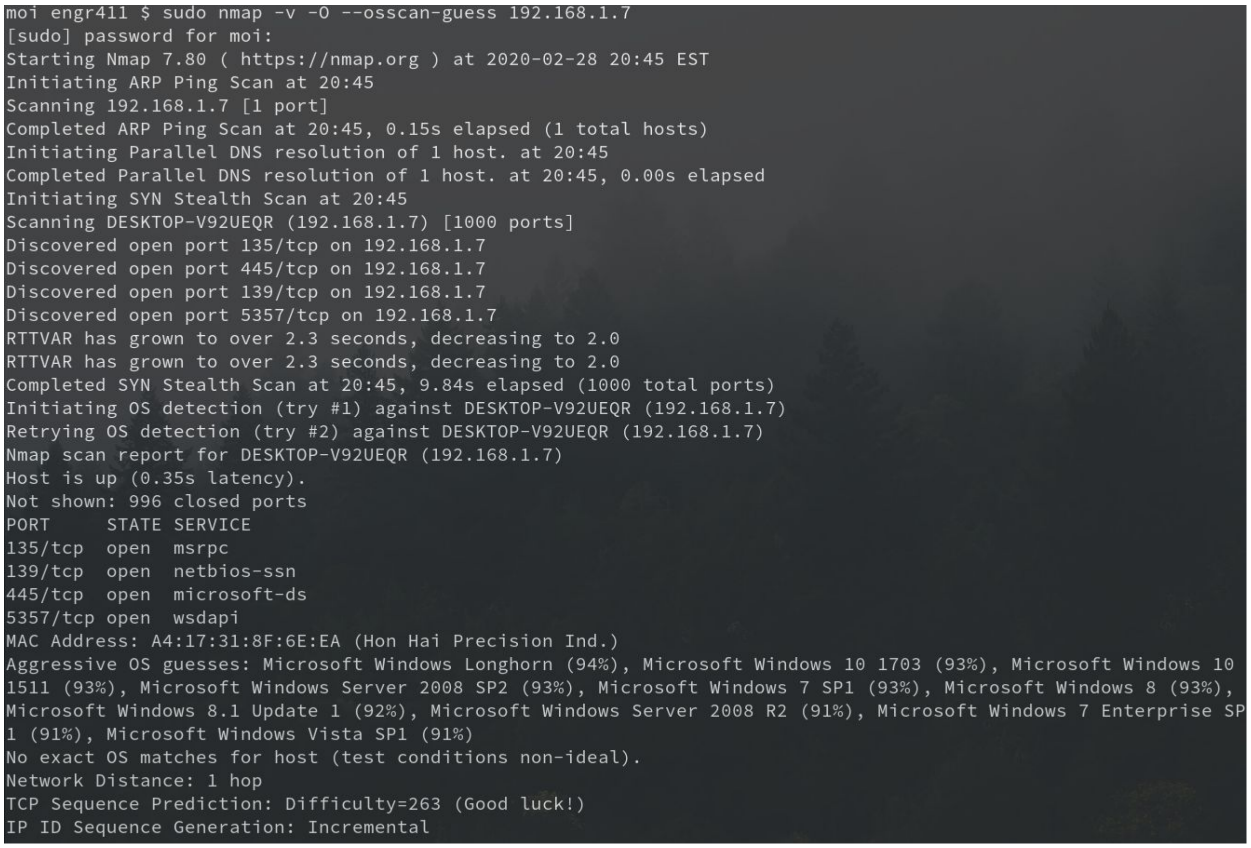
Device specifications

Device name	DESKTOP-V92UEQR
Processor	Intel(R) Core(TM) i3-2328M CPU @ 2.20GHz 2.20 GHz
Installed RAM	8.00 GB (7.86 GB usable)
Device ID	DE3A081A-8CA1-4C64-B774-43B1AFB6FBBE
Product ID	00326-10000-00000-AA676
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Windows specifications

Edition	Windows 10 Home
Version	1903
Installed on	12/1/2019
OS build	18362.356

In the below output, I have scanned with Nmap from my Arch Linux laptop a freshly installed Windows 10 Home edition located on a different computer.

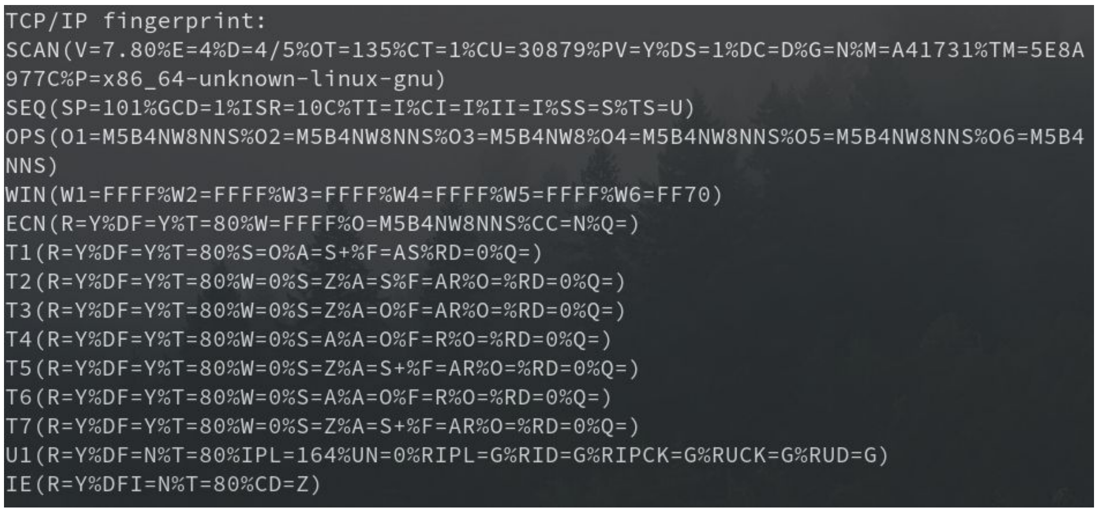


```
moi engr411 $ sudo nmap -v -O --osscan-guess 192.168.1.7
[sudo] password for moi:
Starting Nmap 7.80 ( https://nmap.org ) at 2020-02-28 20:45 EST
Initiating ARP Ping Scan at 20:45
Scanning 192.168.1.7 [1 port]
Completed ARP Ping Scan at 20:45, 0.15s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 20:45
Completed Parallel DNS resolution of 1 host. at 20:45, 0.00s elapsed
Initiating SYN Stealth Scan at 20:45
Scanning DESKTOP-V92UEQR (192.168.1.7) [1000 ports]
Discovered open port 135/tcp on 192.168.1.7
Discovered open port 445/tcp on 192.168.1.7
Discovered open port 139/tcp on 192.168.1.7
Discovered open port 5357/tcp on 192.168.1.7
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
Completed SYN Stealth Scan at 20:45, 9.84s elapsed (1000 total ports)
Initiating OS detection (try #1) against DESKTOP-V92UEQR (192.168.1.7)
Retrying OS detection (try #2) against DESKTOP-V92UEQR (192.168.1.7)
Nmap scan report for DESKTOP-V92UEQR (192.168.1.7)
Host is up (0.35s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
5357/tcp open  wsdapi
MAC Address: A4:17:31:8F:6E:EA (Hon Hai Precision Ind.)
Aggressive OS guesses: Microsoft Windows Longhorn (94%), Microsoft Windows 10 1703 (93%), Microsoft Windows 10
1511 (93%), Microsoft Windows Server 2008 SP2 (93%), Microsoft Windows 7 SP1 (93%), Microsoft Windows 8 (93%),
Microsoft Windows 8.1 Update 1 (92%), Microsoft Windows Server 2008 R2 (91%), Microsoft Windows 7 Enterprise SP
1 (91%), Microsoft Windows Vista SP1 (91%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=263 (Good luck!)
IP ID Sequence Generation: Incremental
```

In this command, we used the arguments -O (Enable OS detection) along with --osscan-guess to perform automatically an Operative System detection following with a probabilistic guess of the OS identity with the TCP/IP stack fingerprint. We can clearly see that the hostname of the system is a match to our Windows capture with its name being DESKTOP-V92UEQR. This Windows host is located at the IP address 192.168.1.7 of my home network which explains the last argument passed to Nmap as such. The results of this performed Nmap scan are explained in great detail in the following section.

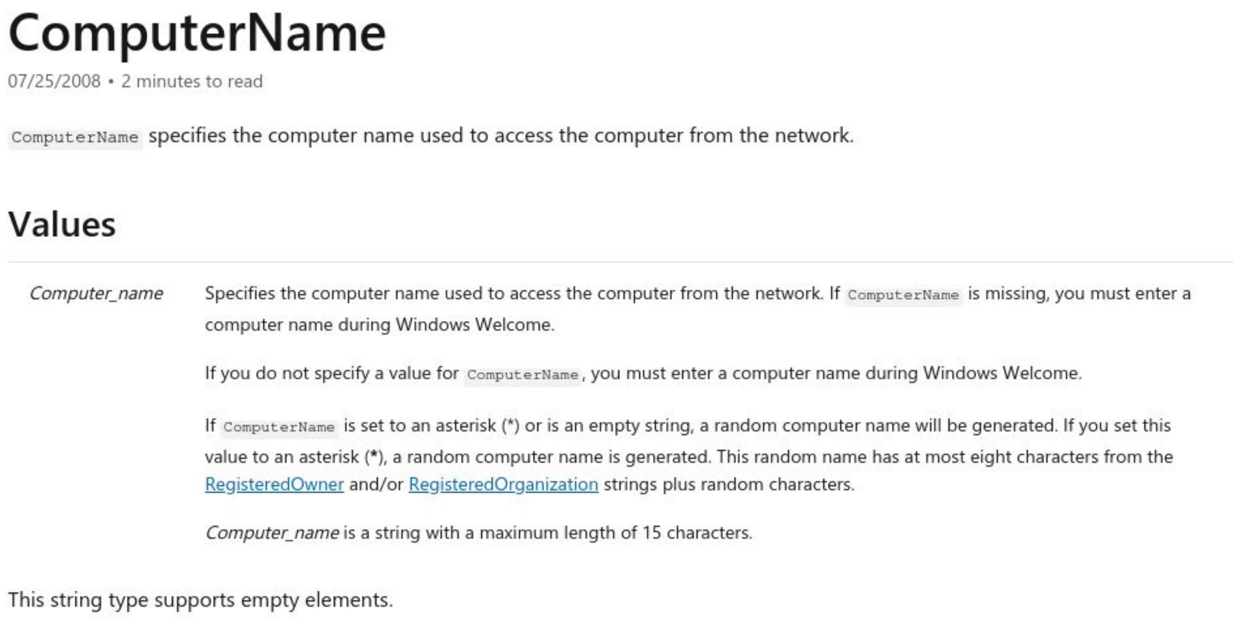
2.2 Result

The initial problem was to solve the OS fingerprinting of a specific device. In our setup, we scanned a Windows 10 machine. We used an Active scan meaning that we directly sent packets and listened to responses to all of the available services to identify with very high accuracy the system identity of this device. We can clearly see in the previous figure that the services “microsoft-ds”, “msrpc” and “wsdapi” with open TCP ports are already indicating the Microsoft OS of this device. The probabilistic OS-scanning guess indicated us “Microsoft Windows 10 (93%)” which confirms the accurate identification of the targeted computer. The first 94% march is a generic Longhorn Windows release but there are two consecutive Windows 10 matches of 93% with different fingerprint IDs 1703 & 1511. This result confirms with high certitude that the OS behind the IP 192.168.1.7 is indeed a Windows 10 system. Here is the fingerprint:



```
TCP/IP fingerprint:
SCAN(V=7.80%E=4%D=4/5%OT=135%CT=1%CU=30879%PV=Y%DS=1%DC=D%G=N%M=A41731%TM=5E8A
977C%P=x86_64-unknown-linux-gnu)
SEQ(SP=101%GCD=1%ISR=10C%TI=I%CI=I%II=I%SS=S%TS=U)
OPS(O1=M5B4NW8NNS%O2=M5B4NW8NNS%O3=M5B4NW8%O4=M5B4NW8NNS%O5=M5B4NW8NNS%O6=M5B4
NNS)
WIN(W1=FFFF%W2=FFFF%W3=FFFF%W4=FFFF%W5=FFFF%W6=FF70)
ECN(R=Y%DF=Y%T=80%W=FFFF%O=M5B4NW8NNS%CC=N%Q=)
T1(R=Y%DF=Y%T=80%S=O%A=S+%F=AS%RD=0%Q=)
T2(R=Y%DF=Y%T=80%W=0%S=Z%A=S%F=AR%O=%RD=0%Q=)
T3(R=Y%DF=Y%T=80%W=0%S=Z%A=O%F=AR%O=%RD=0%Q=)
T4(R=Y%DF=Y%T=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)
T5(R=Y%DF=Y%T=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=Y%DF=Y%T=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)
T7(R=Y%DF=Y%T=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
U1(R=Y%DF=N%T=80%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)
IE(R=Y%DFI=N%T=80%CD=Z)
```

For the rest of the probabilities, the spectrum is rather reduced to very few systems following up with a lower percentage for Windows 7 and Windows Server which means that range of possibilities is well limited to what a system might be. From this point a vulnerability scanner may be launched optimizing its runtime to a very few Windows systems with minor differences. From the blackbox perspective assuming we didn't know the identity of this OS, we can clearly see a rather generic default hostname DESKTOP-V92UEQR generated during the install of Windows 10 Home edition. In case of the probabilistic guess failure, an individual scanning a network device could rely on this information to confirm the identity of the Operative System using common sense. In the below image, Microsoft is explaining [3] this generic name:



ComputerName

07/25/2008 • 2 minutes to read

`ComputerName` specifies the computer name used to access the computer from the network.

Values

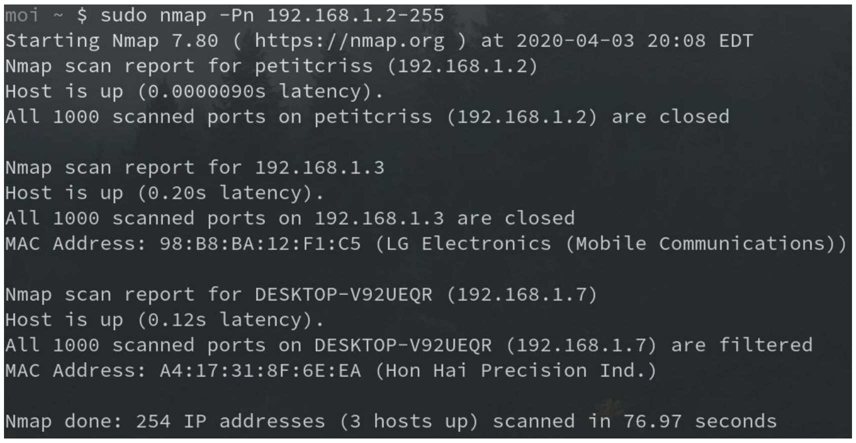
Computer_name	Specifies the computer name used to access the computer from the network. If `ComputerName` is missing, you must enter a computer name during Windows Welcome. If you do not specify a value for `ComputerName`, you must enter a computer name during Windows Welcome. If `ComputerName` is set to an asterisk (*) or is an empty string, a random computer name will be generated. If you set this value to an asterisk (*****), a random computer name is generated. This random name has at most eight characters from the RegisteredOwner and/or RegisteredOrganization strings plus random characters. *Computer_name* is a string with a maximum length of 15 characters.

This string type supports empty elements.

Overall, OS fingerprinting with Nmap is a very promising solution as it provides an automated method to run a complex TCP/IP stack fingerprinting.

3. PLAN OF ACTION

1. The first step is to find a device on the network. This can be achieved using a tool called Nmap by running with an argument of No ping scan (-Pn) which skips host discovery and scans all target hosts ports. We can see our Desktop Windows machine is at IP 192.168.1.7 with its generic Windows specific default hostname. The reason I used this type of scan is because most of the machines nowadays avoid giving sensitive information over a network such as online status information. This measure is put in place to reduce attackers visibility which makes my target host invisible to a List scan (-sL) that is using the host discovery without sending any packets to the host.



```
moi ~ $ sudo nmap -Pn 192.168.1.2-255
Starting Nmap 7.80 ( https://nmap.org ) at 2020-04-03 20:08 EDT
Nmap scan report for petitcriss (192.168.1.2)
Host is up (0.0000090s latency).
All 1000 scanned ports on petitcriss (192.168.1.2) are closed

Nmap scan report for 192.168.1.3
Host is up (0.20s latency).
All 1000 scanned ports on 192.168.1.3 are closed
MAC Address: 98:B8:BA:12:F1:C5 (LG Electronics (Mobile Communications))

Nmap scan report for DESKTOP-V92UEQR (192.168.1.7)
Host is up (0.12s latency).
All 1000 scanned ports on DESKTOP-V92UEQR (192.168.1.7) are filtered
MAC Address: A4:17:31:8F:6E:EA (Hon Hai Precision Ind.)

Nmap done: 254 IP addresses (3 hosts up) scanned in 76.97 seconds
```

2. Afterwards, we performed a TCP/IP stack fingerprinting with Nmap using the OS detection (-O) argument. This step determines the identity of the OS with a unique fingerprint.
3. As the last step, we included an argument (--osscan-guess) to perform a probabilistic OS guess comparing the found fingerprint at the previous step to the Nmap OS database storing more than 2’600 potential matches.

4. SCHEDULE

For the representation of the schedule, I decided to go with a slight variation of the Gantt Chart to illustrate the schedule of environment setup, network scanning tools and techniques identification. The approximate timeframe of each stage is illustrated with a different color.

	Business hours							
Day	9am	10am	12pm	13pm	14pm	15pm	16pm	17pm
Monday	Windows 10 Setup							
Tuesday	Network Scanners & Techniques Research				Identification of the target on the network with Nmap and exploration of OS scanning techniques.			
Wednesday				Results analysis according to Nmap documentation				
Thursday				Redaction of the analysis in the Report				
Friday	Report redaction & references documentation							
Saturday					Report verification			

5. BUDGET

For the budget, Nmap is Free and Open Source. Moreover, Windows 10 and Linux distributions are completely free. Therefore, the only requirement is that a person possesses or rents one computer. Then, both of the machines can be emulated with VirtualBox at no cost.

6. QUALIFICATIONS

Nmap is a Free (GPLv2 license) and Open Source [4] network scanning tool. It was created by Gordon Lyon with its initial release in September 1997. Today, it has become an industry standard for performing network scanning. In the Computer Security industry made of white hat hackers, Nmap is required for basic intrusion detection certification such as CISSP (Certified Information Systems Security Professional) [5]. CISSP is supported by the DoD (U.S. Department of Defense) and it's considered a foundation for Cybersecurity specialists. Nmap contains multiple scanning techniques and it is actively developed. It has been popularized by its appearance in the 2003 movie Matrix Reloaded when Trinity is using it as a high fidelity network scanning tool. This appearance has sparked multiple forum interests and popularized its usage. Since then, Nmap has become a research topic of its own in academia. One of these research topics is experimental validation of correlation systems of cyberattacks published [6] by IEEE (Institute of Electrical and Electronics Engineers). In light of these facts, I must assert that Nmap is a highly qualified tool to perform the OS fingerprinting to identify the system of a device.

It has been in the Computer Security industry for more than 20 years and it is now included in popular Cybersecurity Virtual Machines such as Kali Linux. It's crossplatform availability of both systems (Windows, OSX, Linux) and embedded devices (Android, iOS) is a demonstration of very stable and robust code. Finally, multiple graphical tools such as Zenmap [8] are built on top of Nmap by emulating it or by using the various Nmap's outputs (XML, grepable, normal etc.). Therefore, Nmap is a very stable and reliable tool to solve the initial problem.

7. CONCLUSION

7.1 Summary

Operative System fingerprinting is a rather vast and complex task to perform. In fact, there are thousands of systems and there are all built differently aimed for a wide range of purposes. The main issue is to find a mature networking tool to perform such a complex duty with high accuracy. To avoid this problematic situation, I proposed to use a network scanner called Nmap. This tool allows us to scan a device on a network with various prebuilt procedures to identify a system using TCP/IP fingerprinting. At first, we established the plan of action by understanding entirely the steps to perform. At the first stage of the plan, we performed a No ping scan (-Pn) to find a device on the network using the active scanning method which interacts directly with the target system. Then, we performed TCP/IP stack fingerprinting with Nmap using the OS detection (-O) argument along with the argument from the final stage (--osscan-guess) which performs a probabilistic guess from the output of the OS detection outcome.

Afterwards, we defined an appropriate approach along with the results analysis to make sure the OS scanning methods answers the requirement of identifying an Operative System in the scope of our network. The schedule and budget was planned according to the availability of my schedule and in favor of the FOSS (Free and Open Source Software) to make this study reproducible for anyone at no cost. Finally, we verified the qualifications of all the involved Nmap tools to make sure its results are trust-worthy and to attest the legitimacy of its legal use for study purposes. Overall, I think that performing an OS fingerprinting with Nmap will greatly enhance the portability as well as the accuracy of this task by reducing the deprecated or non-maintained software with an old database not indexing recent Operative Systems. The possibility of performing such a diverse range of OS fingerprinting techniques really expands the methods to investigate. Nmap sheds light on scanning methods which are documented in great detail on their website [1]. The availability of the source code leverages the understanding of the inner and low level implementation as well as it presents the possibility of integrating new Operative System fingerprinting procedures.

7.2 Contact

I would be glad to answer any further inquiries on this matter by email.

Here is my contact information:

Vsevolod Ivanov

seva@binarytrails.net

Computer Applications - Computation Arts

Concordia University

REFERENCES

[1] Nmap, “TCP/IP Fingerprinting Methods Supported by Nmap”.

Accessed on: April 05, 2020. [Online]. Available:

https://nmap.org/book/osdetect-methods.html

[2] Nmap, “TCP/IP Reference”.

Accessed on: April 05, 2020. [Online]. Available:

https://nmap.org/book/tcpip-ref.html

[3] MSDN, “ComputerName”.

Accessed on: April 05, 2020. [Online]. Available:

https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-vista/cc749460(v=ws.10)?redirectedfrom=MSDN

[4] Nmap, “Nmap - the Network Mapper”.

Accessed on: April 05, 2020. [Online]. Available:

https://github.com/nmap/nmap

[5] ISC2, “Become a CISSP – Certified Information Systems Security Professional”,

Accessed on: April 05, 2020. [Online]. Available:

https://www.isc2.org/Certifications/CISSP

[6] IEEE Security & Privacy, "Validation of sensor alert correlators".

Accessed on: April 05, 2020. [Online]. Available:

https://ieeexplore.ieee.org/document/1176995