

# Network Vulnerabilities Identification Through Network Security Scanner

*By Rashed Al Fuhed*

*2000791840*

*School of Science*

*rashalyami79@gmail.com*

*10/1/2023*

## Table of Contents

1. Introduction.....	3
2. Background .....	4
2.1 Testing types.....	4
2.2 Scanner Types .....	4
2.2.1 Application Scanners.....	4
2.2.2 Database Scanners .....	5
The scanners which are used to find the vulnerabilities in a database are known as Database scanners. They also used to identify whether the stored information is protected from the attacks or not. Both internal and external overview is checked through these scanners for potential security risks. The most common database attack is a SQL injection which is used to test the security of databases.....	5
2.2.3 Network based Scanners .....	5
These scanners are used to find out the vulnerable and security attacks on the networks. The unknown devices can be identified on a network through network scanners. Unauthorized or unknown remote access is the best example for this type of attack which makes network to insecure [7]. There are various scans by network-based scanners which can be executed to test networks. The weak passwords can be identified and settled out through it. ....	5
3. Vulnerability Scanning importance .....	5
4. Main Network Vulnerabilities .....	5
4.1 Unpatched systems .....	5
4.2 Misconfigured devices .....	5
4.3 Human error .....	6
5. Steps to avoid Security Risks .....	6
1. Network Vulnerability Scanning.....	6
2. IDS or IPS installation .....	6
3. Network segmentation.....	6
4. Backup.....	6
6. Important tools for vulnerability scanning .....	6
The vulnerability scanners use different tools to scan the network, lot of them are important from anyhow. Some main tools for open source are listed below:..	6
7. Result and discussion.....	6
7.1 Wireshark results.....	6
7.1.1 ARP Port scanning .....	7
7.1.2 Password identification.....	7
7.1.3 Outdated software identification .....	8

7.2 Python implementation:.....	9
7.2.1 Python Code .....	9
7.2.2 Output: .....	11
7.2.3 Python code .....	11
7.2.4 Output: .....	13
8. Conclusion .....	13
9. Extra Credit.....	13
10. References .....	14

## 1. Introduction

A software device or tool which is used to scan the full network with its nodes for security loopholes and vulnerabilities is known as a network security scanner. This is a computerized solution that evaluates, assesses, and scans the underlying network security strength and posture. Initially the network scanner is used by network administrators or network security to assess the security of different networks. Generally, a security scanner is used to scan all possible and known threats and vulnerabilities. All the devices can be scanned which includes:

- Servers
- Client computers
- Routers
- Firewalls

Network security scanning is also known as vulnerability scanning of a network. This is the way of finding the problems in network services, systems, and devices. These vulnerabilities or problems happen due to the running network outdated software, open ports or misconfiguration and hackers can utilize them very easily. These scanners are an important representation of any IT company arsenal which is used to expose network security problems by executing a detailed analysis of the network to notice the network security holes. The following vulnerabilities can be check through network security scanners, such as:

- Open ports
- Operating system controls
- Password strength
- Scripts

These scanners are critical tools to support security parameters from cybersecurity attacks and threats. The vulnerabilities scanners are especially functional when compensating with open-source code or 3<sup>rd</sup> party providers. Engaging in an open source, because of nature of these projects, there is no surety of implementation of best security as every developer is contributing to the project. That's why without the use of scanner it can be harder to step out the code vulnerabilities or verify the representation of no vulnerabilities. These scanners are also not short to just scan the code, they have a huge range to scan the different features and targets. It's also quite challenging to find which scanner is suitable for one's requirements. That's why a lot of scanners or build and important of them are represented in this report so that anyone can choose them easily according to his needs.

## **2. Background**

The background study of various features related to the vulnerability scanners like different scanner types, tested vulnerabilities, and targets. Due to the concern of understanding the various available scanners vulnerability, it's critical to give the information on some of associated features with the scanners.

### **2.1 Testing types**

The vulnerability scanners include various types of penetration testing, which are called black and white box testing. This testing imitates a cyber-attack against the system in computer which checks for vulnerable issues or threats. The testing of white box is the testing of developer view in which the tester has complete knowledge and full access according to the tested target and its features. Due to a lot of information, it is more time-consuming. On the other hand, according to the attacker's view, the black box testing is a type of external testing. In this test, the tester has no information about technologies and the target. With penetration testing, there is authenticate and unauthenticated testing as well.

### **2.2 Scanner Types**

#### **2.2.1 Application Scanners**

The application scanners are used to find the misconfigurations and vulnerabilities of the software in an application. These scanners used various methods for analysis like SCA (as Software Composition Analysis). The static analysis is used for the web applications which referred as SAST and DAST. In this report, both names for these methods will be used mutually. There are specific pitfalls and merits for all these methods, ideally used all to make sure the application is safe.

### **2.2.2 Database Scanners**

The scanners which are used to find the vulnerabilities in a database are known as Database scanners. They also used to identify whether the stored information is protected from the attacks or not. Both internal and external overview is checked through these scanners for potential security risks. The most common database attack is a SQL injection which is used to test the security of databases.

### **2.2.3 Network based Scanners**

These scanners are used to find out the vulnerable and security attacks on the networks. The unknown devices can be identified on a network through network scanners. Unauthorized or unknown remote access is the best example for this type of attack which makes network to insecure [7]. There are various scans by network-based scanners which can be executed to test networks. The weak passwords can be identified and settled out through it.

## **3. Vulnerability Scanning importance**

This is an important part of any IT company or the infrastructure of the security department. There is a little confusion as well about network scanning that what it does and why it's essential. This is an integral parameter of the process of security assessment of any system or network. We must settle out the best way to utilize it as it's complex process. The scanning of vulnerability of a network is the best solution to maintain network security. It's very helpful in the identification of system weak points that required to patch up.

## **4. Main Network Vulnerabilities**

To understand network vulnerabilities scanning in the best way, it's crucial to understand first about the network vulnerabilities forms and types. Basically, there are three main vulnerabilities of the network which are discussed in the following.

### **4.1 Unpatched systems**

Mostly the companies avoid updating their network and systems that allow the attackers to make their system vulnerable. The attackers can find these holes easily. In this way the risk is highly increase for the customers and organizations as well.

### **4.2 Misconfigured devices**

The devices which are not configured well are easily reachable to hackers through that they can enter other system or devices in the network. In many cases, the hackers used those types of misconfigured devices to make their

presentation within specific networks. The entire network is compromised through a single misconfigured device.

### **4.3 Human error**

This is also the weakest link in the chain of security, this is very easy for any hacker to contact an employee to get access throughout the entire network. Social engineering is a common method and example of this type of vulnerability.

## **5. Steps to avoid Security Risks**

The risks of security of a network are a major threat for any organization and business. There should be some preventive parameters which should be followed up to avoid that type of risk. Some of the important practices and steps implementation to make the system secure are listed in the following.

1. Network Vulnerability Scanning
2. IDS or IPS installation
3. Network segmentation
4. Backup

## **6. Important tools for vulnerability scanning**

The vulnerability scanners use different tools to scan the network, lot of them are important from anyhow. Some main tools for open source are listed below:

1. Wireshark
2. Nmap
3. Machine learning
4. Metasploit

## **7. Result and discussion**

The Python code and the Wireshark tool is used here to identify the different types of vulnerabilities which are open ports, weak password, and outdated software.

### **7.1 Wireshark results**

network security scanner.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Encrypted Application Data
1	0.000000	192.168.8.100	192.168.8.255	UDP	82	
2	0.810533	192.168.8.100	www.huaweimobilewif...	DNS	86	
3	0.811568	192.168.8.100	www.huaweimobilewif...	DNS	86	
4	1.481321	www.huaweimobilewif...	192.168.8.100	DNS	163	
5	1.481321	www.huaweimobilewif...	192.168.8.100	DNS	163	
6	1.527394	192.168.8.100	python.repl-web.pro...	TLSv1.2	83	8455300407ed8645bd87b2998027dd8bea0f9b74c9023c68
7	1.819629	192.168.8.100	www.huaweimobilewif...	DNS	84	
8	1.827349	www.huaweimobilewif...	192.168.8.100	DNS	122	
9	2.292730	192.168.8.100	python.repl-web.pro...	TCP	83	
10	2.833599	192.168.8.100	www.huaweimobilewif...	DNS	84	
11	3.722928	python.repl-web.pro...	192.168.8.100	TCP	54	
12	4.617445	python.repl-web.pro...	192.168.8.100	TLSv1.2	79	542315a7184de73f02715e6f1473a27924c4b177
13	4.617445	python.repl-web.pro...	192.168.8.100	TCP	79	
14	4.617527	192.168.8.100	python.repl-web.pro...	TCP	66	
15	4.968375	python.repl-web.pro...	192.168.8.100	TCP	79	
16	4.968437	192.168.8.100	python.repl-web.pro...	TCP	66	
17	5.033269	www.huaweimobilewif...	192.168.8.100	DNS	134	
18	7.132797	www.huaweimobilewif...	IntelCor_0b:88:f3	ARP	42	
19	7.132833	IntelCor_0b:88:f3	www.huaweimobilewif...	ARP	42	
20	7.669409	www.huaweimobilewif...	IntelCor_0b:88:f3	ARP	42	

> Frame 1: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface \Device\NPF\_{D99582A5-D995-4000-8000-000000000000} on 0:00:00:00:00:00  
 > Ethernet II, Src: IntelCor\_0b:88:f3 (80:86:f2:0b:88:f3), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
 > Internet Protocol Version 4, Src: 192.168.8.100 (192.168.8.100), Dst: 192.168.8.255 (192.168.8.255)  
 > User Datagram Protocol, Src Port: 51609 (51609), Dst Port: sentinelsrm (1947)  
 > Data (40 bytes)

0000 ff ff ff ff ff 80 86 f2 0b 88 f3 08 00 45 00 .....E  
 0010 00 44 9a a9 00 00 00 11 0d 4c c0 a8 08 64 c0 a8 ..D.....L...d..  
 0020 08 ff c9 99 07 0b 00 30 ef 26 4a 69 65 63 4a 73 .....0 .&jiecJs  
 0030 66 58 6b 65 33 71 2f 4b 51 47 58 4b 45 55 42 76 fxke3q/K QGXKEUv  
 0040 49 41 74 79 49 64 73 6f 58 62 6d 76 4a 59 74 64 lAtyIdso XbmJYtd  
 0050 4d 41 MA

## 7.1.1 ARP Port scanning

network security scanner.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

arp.dst.hw\_mac==00:00:00:00:00:00

No.	Time	Source	Destination	Protocol	Length	Encrypted Application Data
18	7.132797	www.huaweimobilewif...	IntelCor_0b:88:f3	ARP	42	
20	7.669409	www.huaweimobilewif...	IntelCor_0b:88:f3	ARP	42	
53	32.465755	www.huaweimobilewif...	IntelCor_0b:88:f3	ARP	42	
73	87.528417	www.huaweimobilewif...	IntelCor_0b:88:f3	ARP	42	
112	166.020896	www.huaweimobilewif...	IntelCor_0b:88:f3	ARP	42	
305	237.701522	www.huaweimobilewif...	IntelCor_0b:88:f3	ARP	42	
398	317.996760	www.huaweimobilewif...	IntelCor_0b:88:f3	ARP	42	
400	318.624262	www.huaweimobilewif...	IntelCor_0b:88:f3	ARP	42	
402	319.722695	www.huaweimobilewif...	IntelCor_0b:88:f3	ARP	42	
443	375.496471	www.huaweimobilewif...	Broadcast	ARP	42	
468	385.695799	www.huaweimobilewif...	IntelCor_0b:88:f3	ARP	42	
474	386.248331	www.huaweimobilewif...	IntelCor_0b:88:f3	ARP	42	
505	427.690522	www.huaweimobilewif...	Broadcast	ARP	42	
506	432.912976	www.huaweimobilewif...	Broadcast	ARP	42	
507	432.912976	www.huaweimobilewif...	Broadcast	ARP	42	
508	432.912976	www.huaweimobilewif...	Broadcast	ARP	42	
509	432.912976	www.huaweimobilewif...	Broadcast	ARP	42	
510	433.729430	CooseaGr_b7:24:91	Broadcast	ARP	42	
511	435.809469	CooseaGr_b7:24:91	Broadcast	ARP	42	
512	435.809469	CooseaGr_b7:24:91	Broadcast	ARP	42	

> Frame 18: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF\_{D99582A5-D995-4000-8000-000000000000} on 0:00:00:00:00:00  
 > Ethernet II, Src: www.huaweimobilewif... (90:2b:d2:0d:ba:1f), Dst: IntelCor\_0b:88:f3 (80:86:f2:0b:88:f3)  
 > Address Resolution Protocol (request)  
 Hardware type: Ethernet (1)  
 Protocol type: IPv4 (0x0800)  
 Hardware size: 6  
 Protocol size: 4  
 Opcode: request (1)  
 Sender MAC address: www.huaweimobilewif... (90:2b:d2:0d:ba:1f)  
 Sender IP address: www.huaweimobilewif... (192.168.8.1)  
 Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)  
 Target IP address: 192.168.8.100 (192.168.8.100)

0000 80 86 f2 0b 88 f3 90 2b d2 0d ba 1f 08 06 00 01  
 0010 08 00 06 04 00 01 90 2b d2 0d ba 1f c0 a8 08 01  
 0020 00 00 00 00 00 00 c0 a8 08 64

## 7.1.2 Password identification

network security scanner.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.stream eq 1

No.	Time	Source	Destination	Protocol	Length	Encrypted Application Data
22	18.628145	192.168.8.100	20.198.119.143	TLSv1.2	153	00000000000000005152d08adccc8bcb45348b10672f1ff016021af7c53c9
25	19.234798	192.168.8.100	20.198.119.143	TLSv1.2	153	00000000000000005152d08adccc8bcb45348b10672f1ff016021af7c53c9
27	19.523908	20.198.119.143	192.168.8.100	TLSv1.2	153	00000000000000005152d08adccc8bcb45348b10672f1ff016021af7c53c9
29	19.578188	192.168.8.100	20.198.119.143	TLSv1.2	153	00000000000000005152d08adccc8bcb45348b10672f1ff016021af7c53c9
655	618.626194	192.168.8.100	20.198.119.143	TLSv1.2	153	00000000000000005152d08adccc8bcb45348b10672f1ff016021af7c53c9
656	619.237556	192.168.8.100	20.198.119.143	TLSv1.2	153	00000000000000005152d08adccc8bcb45348b10672f1ff016021af7c53c9
657	619.982348	192.168.8.100	20.198.119.143	TLSv1.2	153	00000000000000005152d08adccc8bcb45348b10672f1ff016021af7c53c9
658	620.725048	20.198.119.143	192.168.8.100	TLSv1.2	153	00000000000000005152d08adccc8bcb45348b10672f1ff016021af7c53c9
659	620.773081	192.168.8.100	20.198.119.143	TLSv1.2	153	00000000000000005152d08adccc8bcb45348b10672f1ff016021af7c53c9
660	621.119255	20.198.119.143	192.168.8.100	TLSv1.2	153	00000000000000005152d08adccc8bcb45348b10672f1ff016021af7c53c9
661	622.062040	20.198.119.143	192.168.8.100	TLSv1.2	153	00000000000000005152d08adccc8bcb45348b10672f1ff016021af7c53c9
662	622.062105	192.168.8.100	20.198.119.143	TLSv1.2	153	00000000000000005152d08adccc8bcb45348b10672f1ff016021af7c53c9
663	622.866582	20.198.119.143	192.168.8.100	TLSv1.2	153	00000000000000005152d08adccc8bcb45348b10672f1ff016021af7c53c9

Wireshark · Follow TCP Stream (tcp.stream eq 1) · network security scanner.pcapng

peers:

- peer: 0  
host: 192.168.8.100  
port: 61471
- peer: 1  
host: 20.198.119.143  
port: https

packets:

- packet: 22  
peer: 0  
index: 0  
timestamp: 1699450150.691528000  
data: !!binary |  
FwMDAF4AAAAAABRUtCK3MyLy0U0ix8nLx/wFgIa98U8noGtM1/GvRuFjmvFvQSKZzIQDCF/dE  
lfjfAczw9cRWsk5Aqmp0qpe9Q+p8rMOItEoA+cZqr9A1x0a5Z8n3c70
- packet: 27  
peer: 1  
index: 0  
timestamp: 1699450151.587291000  
data: !!binary |  
FwMDAQAAAAAABVJYinn1buxJVn8E0R188GI3K40x5YqF3j0FQv4SSkED6n+GpaiTC91Q70+0  
oA3i0NgpmEoYUCaeu1N2okEm0Vcjcmm6fyBT/gx3FCRblga9XLtHy+rP13A0LYJBCGR+wkGALqJ  
WeScs8qU1jIjIvYksLsn+CCoq24no1uu0VxFriPr3dh4xpecCnqa8rua9h0QqzTnXjwoNjMI/g==
- packet: 655  
peer: 0  
index: 1  
timestamp: 1699450750.689577000  
data: !!binary |  
FwMDAF4AAAAAABvxyW9gDRwR8dVaGuFofOW5ZbsuV/EHh6lHkUdFM+dT4PRz+i8SmAIMP3s7y  
zbxaHnMbbhVcq3mQ9nJuto4lu7ZvkH3CjFCsGPS07HXWEjo4g+w18zwf
- packet: 658  
peer: 1  
index: 1  
timestamp: 1699450752.788431000  
data: !!binary |  
FwMDAQAAAAAABmtL02YhhKnxfGvH9mBd+EbEpfnHi/xi5dWBe4BH/a6yp8CXbYuUErEKH6P2  
azNaFqnQxMEuC9mOAEdTExsIJPQL+LzKUwGRSDspwEOOYNZnqTpY8XVKyX0z88vLZ15j9NmdKm3  
pYwG40NMRR4AQKrXsrLdgrXiYn/zdfP6ciJJdIj2Tcr9JzzdMTTaK43wb4LcDxRcg3Tg41h0IQ==

<

- > Frame 29: 54 bytes on wire (432 bits)
- > Ethernet II, Src: IntelCor\_0b:88:f3
- > Internet Protocol Version 4, Src: 19
- > Transmission Control Protocol, Src P

Wireshark · Follow TCP Stream (tcp.stream eq 0) · FTP Demo\_anon

```

220 FTP Service ready.
USER demo
331 Password required for demo
PASS s3cr3t!
230 Logged on
QUIT
221 Goodbye

```

4 client pkts, 3 server pkts, 6 turns.

Entire conversation (115 bytes) Show and save data as ASCII Stream 0

Find:  Find Next

Filter Out This Stream Print Save as... Back Close Help

### 7.1.3 Outdated software identification



network security scanner.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

http

No.	Time	Source	Destination	Protocol	Length	Encrypted Application Data
32	20.420495	192.168.8.100	e15275.g.akamaiedge...	HTTP	267	
40	22.225096	e15275.g.akamaiedge...	192.168.8.100	HTTP/X...	400	

> Frame 32: 267 bytes on wire (2136 bits), 267 bytes captured (2136 bits) on interface \Device\NPF\_{D99582A5-D94...}

> Ethernet II, Src: IntelCor\_0b:88:f3 (80:86:f2:0b:88:f3), Dst: www.huaweimobilewifi.com (90:2b:d2:0d:ba:1f)

> Internet Protocol Version 4, Src: 192.168.8.100 (192.168.8.100), Dst: e15275.g.akamaiedge.net (104.120.125.70)

> Transmission Control Protocol, Src Port: 61495 (61495), Dst Port: http (80), Seq: 1, Ack: 1, Len: 213

▼ Hypertext Transfer Protocol

> GET /en-GB/livetile/preinstall?region=PK&appid=C98EA580842D8B94058BF071E1DA76512D21FE36&FORM=Threshold HTTP/1.1

Connection: Keep-Alive\r\n

User-Agent: Microsoft-WNS/10.0\r\n

Host: tile-service.weather.microsoft.com\r\n

\r\n

[Full request URI: http://tile-service.weather.microsoft.com/en-GB/livetile/preinstall?region=PK&appid=C98EA580842D8B94058BF071E1DA76512D21FE36&FORM=Threshold]

[HTTP request 1/1]

[Response in frame: 40]

## 7.2 Python implementation:

### 7.2.1 Python Code

This file is for scanning the vulnerabilities of a network.

networkSecurityScan.py

```

C: > Users > DELL > Downloads > network security scan.py > ...
1 #####
2 # Python Code #
3
4 ##### Port Scanner using Socket
5
6 from socket import *
7 import time
8 startTime = time.time()
9
10 if __name__ == '__main__':
11     target = input('Enter the host to be scanned: ')
12     t_IP = gethostbyname(target)
13     print('Starting scan on host: ', t_IP)
14
15     for i in range(50, 500):
16         s = socket(AF_INET, SOCK_STREAM)
17
18         conn = s.connect_ex((t_IP, i))
19         if(conn == 0):
20             print('Port %d: OPEN' % (i,))
21             s.close()
22     print('Time taken:', time.time() - startTime)
23
24
25     ## Port Scanner using ICMP (ping sweep)
26
27
28 import os
29 import platform
30
31 from datetime import datetime
32 net = input("Enter the Network Address: ")
33 net1= net.split('.')
34 a = '..'
35

```

```

36 net2 = net1[0] + a + net1[1] + a + net1[2] + a
37 st1 = int(input("Enter the Starting Number: "))
38 en1 = int(input("Enter the Last Number: "))
39 en1 = en1 + 1
40 oper = platform.system()
41
42 if (oper == "Windows"):
43     ping1 = "ping -n 1 "
44 elif (oper == "Linux"):
45     ping1 = "ping -c 1 "
46 else :
47     ping1 = "ping -c 1 "
48 t1 = datetime.now()
49 print ("Scanning in Progress:")
50
51 for ip in range(st1,en1):
52     addr = net2 + str(ip)
53     comm = ping1 + addr
54     response = os.popen(comm)
55
56     for line in response.readlines():
57         if(line.count("TTL")):
58             break
59         if (line.count("TTL")):
60             print (addr, "--> Live")
61
62 t2 = datetime.now()
63 total = t2 - t1
64 print ("Scanning completed in: ",total)
65
66
67 | | |   ### Port Scanner using TCP scan ###
68
69 import socket
70 from datetime import datetime
71
72 net = input("Enter the IP address: ")
73 net1 = net.split('.')
74 a = '.'
75
76 net2 = net1[0] + a + net1[1] + a + net1[2] + a
77 st1 = int(input("Enter the Starting Number: "))
78 en1 = int(input("Enter the Last Number: "))
79 en1 = en1 + 1
80 t1 = datetime.now()
81
82 def scan(addr):
83     s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
84     socket.setdefaulttimeout(1)
85     result = s.connect_ex((addr,135))
86     if result == 0:
87         return 1
88     else :
89         return 0
90
91 def run1():
92     for ip in range(st1,en1):
93         addr = net2 + str(ip)
94         if (scan(addr)):
95             print (addr , "is live")
96
97 run1()
98 t2 = datetime.now()
99 total = t2 - t1
100 print ("Scanning completed in: " , total)
101

```

### 7.2.2 Output:

```

Enter the host to be scanned:
localhost
Starting scan on host: 127.0.0.1
Time taken: 11.748804807662964
Enter the Network Address:
127.0.0.1
Enter the Starting Number:
1
Enter the Last Number:
5
ping: permission denied (are you root?)
ping: permission denied (are you root?)
ping: permission denied (are you root?)
ping: permission denied (are you root?)
ping: permission denied (are you root?)
Scanning in Progress:
Scanning completed in: 0:00:00.162236
Enter the IP address:
127.0.0.1
Enter the Starting Number:
1
Enter the Last Number:
10
Scanning completed in: 0:00:00.000659

```

```

Enter the Last Number: 10
127.0.0.1 is live
127.0.0.2 is live
127.0.0.3 is live
127.0.0.4 is live
127.0.0.5 is live
127.0.0.6 is live
127.0.0.7 is live
127.0.0.8 is live
127.0.0.9 is live
127.0.0.10 is live
Scanning completed in: 0:00:00.032995

```

### 7.2.3 Python code

This file is for scanning vulnerable open ports.

## Ports.py

```

C: > Users > DELL > Downloads > ports.py > ...
1  #####
2  # Python Code #
3
4  import socket
5
6
7  class portscan():
8      banners = []
9      open_ports = []
10     def __init__(self, target, port_num):
11         self.target = target
12         self.port_num = port_num
13
14     def scan(self):
15         for port in range(1, self.port_num):
16             self.scan_port(port)
17
18
19     def check_ip(self):
20         try:
21             IP(self.target)
22             return(self.target)
23         except ValueError:
24             return socket.gethostbyname(self.target)
25
26
27     def scan_port(self, port):
28         try:
29             converted_ip = self.check_ip()
30             sock = socket.socket()
31             sock.settimeout(0.5)
32             sock.connect((converted_ip, port))
33             self.open_ports.append(port)
34             try:
35                 banner = sock.recv(1024).decode().strip('\n').strip('\r')
36
37                 self.banners.append(banner)
38             except:
39                 self.banners.append(' ')
40             sock.close()
41         except:
42             pass
43
44     import portscanner
45
46     targets_ip = input('[+] * Enter Target To Scan For Vulnerable Open Ports: ')
47     port_number = int(input('[+] * Enter Amount Of Ports You Want To Scan (500 - First 500 Ports): '))
48     vul_file = input('[+] * Enter Path To The File With Vulnerable Softwares: ')
49     print('\n')
50
51     target = portscanner.portscan(targets_ip, port_number)
52     target.scan()
53     with open(vul_file, 'r') as file:
54         count = 0
55         for banner in target.banners:
56             file.seek(0)
57             for line in file.readlines():
58                 if line.strip() in banner:
59                     print('[!!] VULNERABLE BANNER: "' + banner + '" ON PORT: ' + str(target.open_ports[count]))
60                     count += 1

```

### 7.2.4 Output:

```
[+] * Enter Target To Scan For Vulnerable Open Ports: 192.168.1.11
[+] * Enter Amount Of Ports You Want To Scan (500 - First 500 Ports): 120
[+] * Enter Path To The File With Vulnerable Softwares: vulbanners.txt

[!!] VULNERABLE BANNER: "220 (vsFTPd 2.3.4)" ON PORT: 21
[!!] VULNERABLE BANNER: "SSH-2.0-OpenSSH 4.7p1 Debian-8ubuntu1" ON PORT: 22
```

## 8. Conclusion

Network security through scanners is very important in this era for each organization and business. In this project report the full details about the network security scanning are provided. Also, the implementation through Wireshark and the python code is provided to identify these all vulnerabilities of the network that included weak passwords, outdated software and open ports scanning. In future more suitable tools and methods can be developed according to the vast needs of modern network security.

## 9. Extra Credit

I used multiple network protocols in this project, including **TCP**, **Socket**, **ICMP (ping sweep)**, and **Wireshark** to enhance the depth of the security assessment. Socket programming, which involves the use of sockets to establish communication channels between devices, is essential for creating custom tools or scripts that interact with network services. Wireshark, a powerful network protocol analyzer, aids in capturing and dissecting packets, providing detailed insights into network activities. Implementing TCP and ICMP ping sweeps allows for efficient scanning of hosts and identification of live systems on a network. This topic is also high-level and very time-consuming because of how important it is. It enables a more comprehensive understanding of potential vulnerabilities and security gaps within the network infrastructure using the features above.

## 10. References

- [1] OWASP. (n.d.). LDAP Injection. OWASP. Retrieved from [https://owasp.org/wwwcommunity/attacks/LDAP\\_Injection](https://owasp.org/wwwcommunity/attacks/LDAP_Injection).
- [2] OWASP. (n.d.). Vulnerabilities. OWASP. Retrieved from <https://owasp.org/wwwcommunity/vulnerabilities/>.
- [3] Georgieva, E. (2022, October 9). Black Box Penetration Testing: When Do You Need One? PurpleSec. Retrieved from <https://purplesec.us/learn/black-box-penetrationtesting/>.
- [4] Synopsys. (n.d.). Static Application Security Testing. Synopsys. Retrieved from <https://www.synopsys.com/glossary/what-is-sast.html>.
- [5] Check Point. (n.d.). What is Dynamic Code Analysis? Check Point. Retrieved from <https://www.checkpoint.com/cyber-hub/cloud-security/what-is-dynamiccode-analysis/>.
- [6] Synopsys. (n.d.). Black Duck Software Composition Analysis. Synopsys. Retrieved from <https://www.synopsys.com/software-integrity/security-testing/softwarecomposition-analysis.html>.
- [7] Balbix. (n.d.). What are vulnerability scanners. Balbix. Retrieved from <https://www.balbix.com/insights/what-to-know-about-vulnerabilityscanning-and-tools/>.
- [8] RSI Security. (2021, November 9). 7 Types of Vulnerability Scanners. RSI Security. Retrieved from <https://blog.rsisecurity.com/7-types-of-vulnerabilityscanners/>.
- [9] RSI Security. (2020, August 28). Top 5 Types of Penetration Testing. RSI Security. Retrieved from <https://blog.rsisecurity.com/top-5-types-of-penetrationtesting/>.
- [10] Georgieva, E. (2022, November 2). White Box Penetration Testing: When Do You Need One? PurpleSec. Retrieved from <https://purplesec.us/learn/white-boxpenetration-testing/>.
- [11] TutorialsPoint. (n.d.). Database Testing – Security. TutorialsPoint. Retrieved from [https://www.tutorialspoint.com/database\\_testing/database\\_testing\\_security.htm#](https://www.tutorialspoint.com/database_testing/database_testing_security.htm#).
- [12] Sharma, L. (2023, March 16). 10 DevSecOps to Know as a Developer or Sysadmin. Geekflare. Retrieved from <https://geekflare.com/devsecops-tools/>.
- [13] NIST. (n.d.). Database Scanning Tools. NIST. Retrieved from <https://www.nist.gov/itl/ssd/software-quality-group/database-scanningtools>.
- [14] OWASP. (n.d.). Cross Site Scripting (XSS). OWASP. Retrieved from <https://owasp.org/www-community/attacks/xss/>.

- [15] OWASP. (n.d.). SQL Injection. OWASP. Retrieved from [https://owasp.org/wwwcommunity/attacks/SQL\\_Injection](https://owasp.org/wwwcommunity/attacks/SQL_Injection).
- [16] Invicti. (n.d.). Local file inclusion. Invicti. Retrieved from <https://www.invicti.com/learn/local-file-inclusion-lfi/>.
- [17] Invicti. (n.d.). CRLF injection. Invicti. Retrieved from <https://www.invicti.com/learn/crlf-injection/>.
- [18] Invicti. (n.d.). Server-side request forgery (SSRF). Invicti. Retrieved from <https://www.invicti.com/learn/server-side-request-forgery-ssrf/>.
- [19] Invicti. (n.d.). XML external entity (XXE). Invicti. Retrieved from <https://www.invicti.com/learn/xml-external-entity-xxe/>.
- [21] Greenbone OpenVAS. (n.d.). Greenbone OpenVAS. Greenbone OpenVAS. Retrieved from <https://www.openvas.org/>.
- [22] Tenable. (n.d.). Nessus. Tenable. Retrieved from <https://www.tenable.com/products/nessus>.
- [23] Nmap.org. (n.d.). News. Nmap.org. Retrieved from <https://nmap.org/>.
- [24] Rapid7. (n.d.). Metasploitable 2. Rapid7. Retrieved from <https://docs.rapid7.com/metasploit/metasploitable-2/>.
- [25] Oracle. (n.d.). VirtualBox. Oracle. Retrieved from <https://www.virtualbox.org/>.
- [26] NIST. (n.d.). National Vulnerability Database. NIST. Retrieved from <https://nvd.nist.gov/>.
- [27] Greenbone Security Manager. (n.d.). 21.19 Quality of Detection (QoD). Greenbone Security Manager. Retrieved from <https://docs.greenbone.net/GSM-Manual/gos20.08/en/glossary.html#quality-of-detection-qod>.
- [28] Shankdhar, P. (2020, July 13). 14 best open-source web application vulnerability scanners [updated for 2020]. Infosec. Retrieved from <https://resources.infosecinstitute.com/topic/14-popular-webapplication-vulnerability-scanners/>.
- [29] Kumar, C. (2022, September 6). 12 Open Source Web Security Scanner to Find Vulnerabilities. Geekflare. Retrieved from <https://geekflare.com/open-source-websecurity-scanner/>.
- [30] Aqua. (n.d.). Open-Source Vulnerability Scanning: Methods and Top 5 Tools. Aqua. Retrieved from <https://www.aquasec.com/cloud-native-academy/vulnerabilitymanagement/open-source-vulnerability-scanning/>.
- [31] Snyk. (n.d.). 7 Reasons to use an open-source vulnerability scanner. Snyk. Retrieved from <https://snyk.io/series/open-source-security/open-source-vulnerabilityscanners/>.

- [32] Breachlock. (2023, March 6). Top 5 open-source tools for network vulnerability scanning. Breachlock. Retrieved from <https://www.breachlock.com/resources/blog/top-5open-source-tools-for-network-vulnerability-scanning/>. Breachlock. Retrieved from <https://www.breachlock.com/resources/blog/top-5-open-source-toolsfor-network-vulnerability-scanning/>.
- [33] Aqua. (n.d.). Aqua Trivy: Vulnerability and Misconfiguration Scanning. Retrieved from <https://www.aquasec.com/products/trivy/>.
- [34] Github. (n.d.). Clair. Github. Retrieved from <https://github.com/quay/clair>.
- [35] Github. (n.d.). Anchore Engine. Github. Retrieved from <https://github.com/anchore/anchore-engine>.
- [36] Sqlmap. (n.d.). Introduction. Sqlmap. Retrieved from <https://sqlmap.org/>.
- [37] Wapiti. (2023, January 16). Wapiti. Wapiti. Retrieved from <https://wapitiscanner.github.io/>.
- [38] Github. (n.d.). VCG (VisualCodeGrepper). Github. Retrieved from <https://github.com/nccgroup/VCG>.
- [39] Github. (n.d.). Brakeman. Github. Retrieved from <https://github.com/presidentbeef/brakeman>.
- [40] Github. (n.d.). Bandit. Github. Retrieved from <https://github.com/PyCQA/bandit>.
- [41] Splint. (n.d.). Splint. Splint. Retrieved from <https://splint.org/>.
- [42] Github. (n.d.). UNO. Github. Retrieved from <https://github.com/nimblecode/Uno>.
- [43] Checkstyle. (2023, March 25). Checkstyle. Checkstyle. Retrieved from <https://checkstyle.org/>.
- [44] OWASP. (n.d.). OWASP Zed Attack Proxy (ZAP). OWASP. Retrieved from <https://www.zaproxy.org/>.
- [45] PortSwigger. (n.d.). What do you want to do with Burp Suite? PortSwigger. Retrieved from <https://portswigger.net/burp>.
- [46] Subgraph. (n.d.). Vega helps you find and fix cross-site scripting (XSS), SQL injection, and more. Subgraph. Retrieved from <https://subgraph.com/vega/>.
- [47] Github. (2022, October 8). Home. Github. Retrieved from <https://github.com/sullo/nikto/wiki>.
- [48] Rapid7. (n.d.). Nexpose Vulnerability Scanner. Rapid7. Retrieved from <https://www.rapid7.com/products/nexpose/>.



- [49] OpenSCAP. (n.d.). Tools. OpenSCAP. Retrieved from <https://www.open-scap.org/>.
- [50] Wireshark. (n.d.). Download Wireshark. Wireshark. Retrieved from <https://www.wireshark.org/download.html>.
- [51] Aircrack-ng. (n.d.). Downloads. Aircrack-ng. Retrieved from <https://www.aircrackng.org/downloads.html>.
- [52] Github. (n.d.). vaf. Github. Retrieved from <https://github.com/d4rckh/vaf>.
- [53] Gaucher, R. (n.d.). Grabber. Retrieved from <http://rgaucher.info/beta/grabber/>.
- [54] Detectify. (n.d.). Complete External Attack Surface Management for AppSec & ProdSec teams. Detectify. Retrieved from <https://detectify.com/>.
- [55] Rapid7. (n.d.). Get Metasploit. Rapid7. Retrieved from <https://www.metasploit.com/download>.
- [56] Deepfence. (n.d.). ThreatMapper. Deepfence. Retrieved from <https://deepfence.io/threatmapper/>.
- [57] Github. (n.d.). Watchdog. Github. Retrieved from <https://github.com/flipkartincubator/watchdog>.
- [58] Snyk. (n.d.). Developer loved, Security trusted. Snyk. Retrieved from <https://snyk.io/>.
- [59] Mend.io. (n.d.). Shift Left. Secure at Scale. Mend.io. Retrieved from <https://www.mend.io/>.
- [60] Arachni. (n.d.). Arachni. Arachni. Retrieved from <https://www.arachniscanner.com/>.
- [61] Github. (n.d.). XssPy – Web Application XSS Scanner. Github. Retrieved from <https://github.com/faizann24/XssPy>.
- [62] W3af. (n.d.). SQL injection, Cross-Site scripting and much more. W3af. Retrieved from <https://w3af.org/>.
- [63] Github. (n.d.). Wfuzz – The Web Fuzzer. Github. Retrieved from <https://github.com/xmendez/wfuzz>.
- [64] Greenbone Security Manager. (n.d.). CVEs 212793 of 212793. Greenbone Security Manager. Retrieved from <https://secinfo.greenbone.net/cves>.
- [65] Tenable. (n.d.). CVEs. Tenable. Retrieved from <https://www.tenable.com/cve>.