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

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 These vulnerabilities or problems happen due to the running network outdated software, open ports or misconfiguration and hackers can utilize them very easily.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 3rd 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

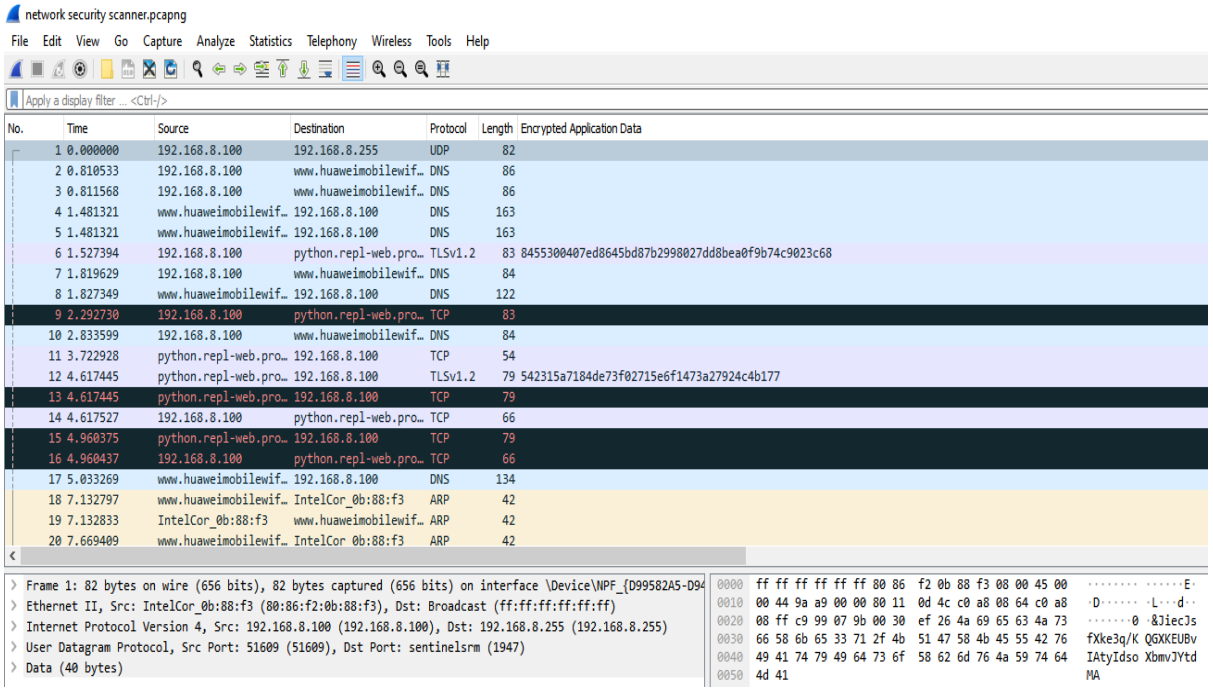
The vulnerability scanners use different tools to scan the network, lot of them are important from anyhow. Some main tools for open source are listed below:

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

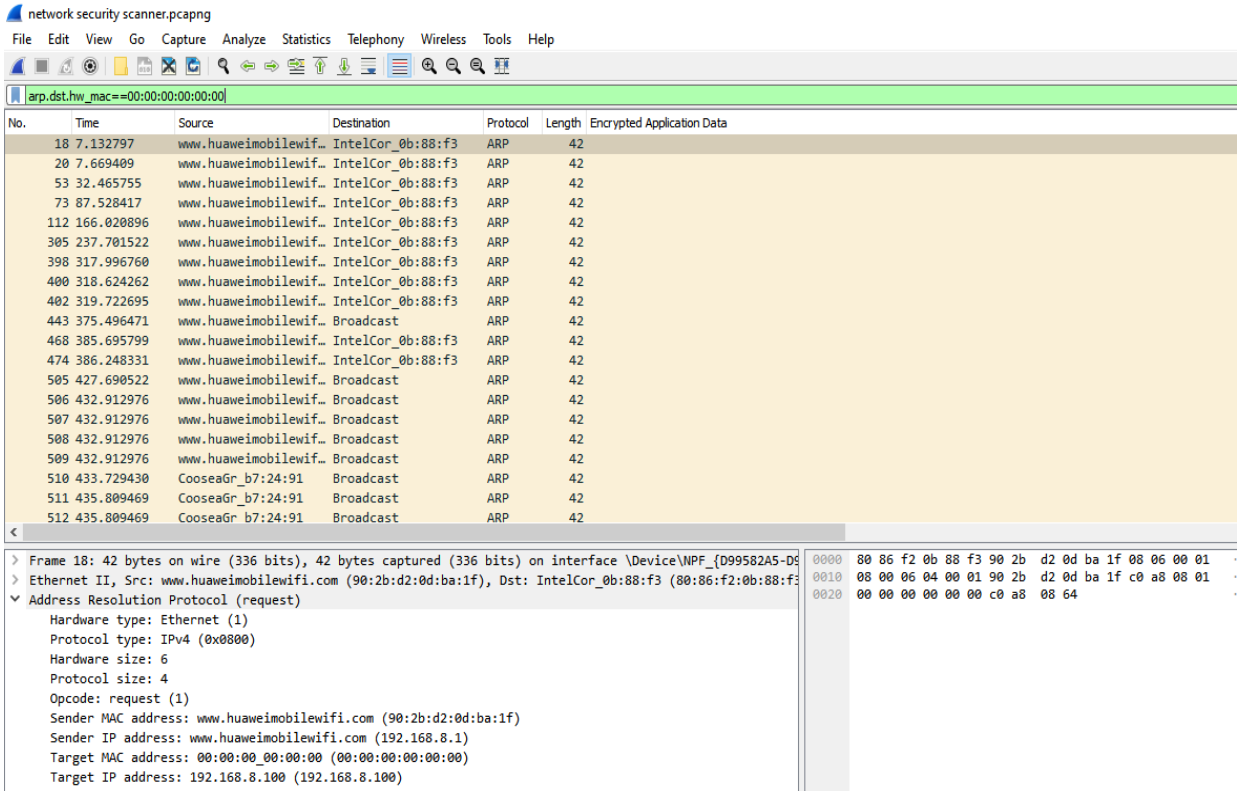
7. Result and discussion

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.

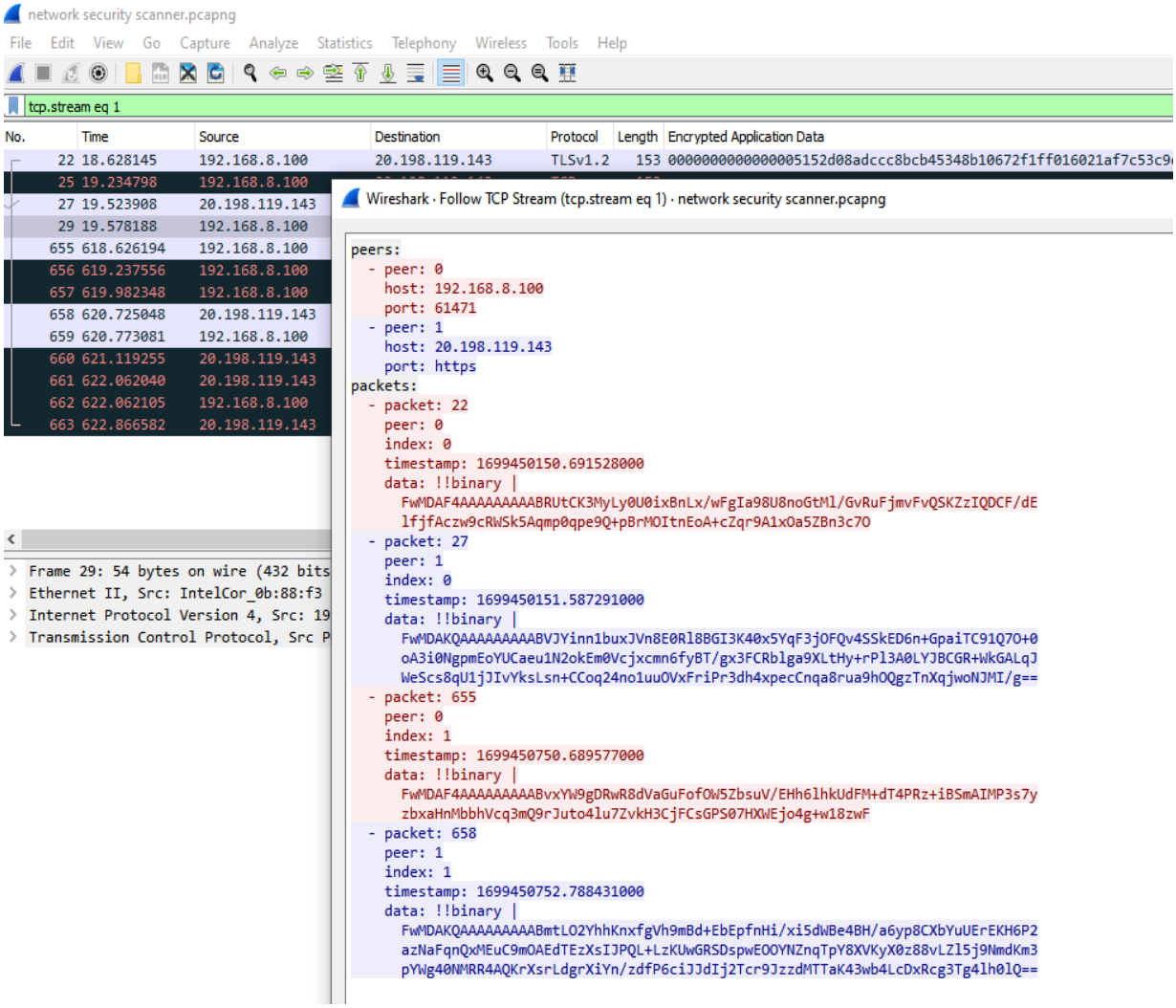
7.1 Wireshark results



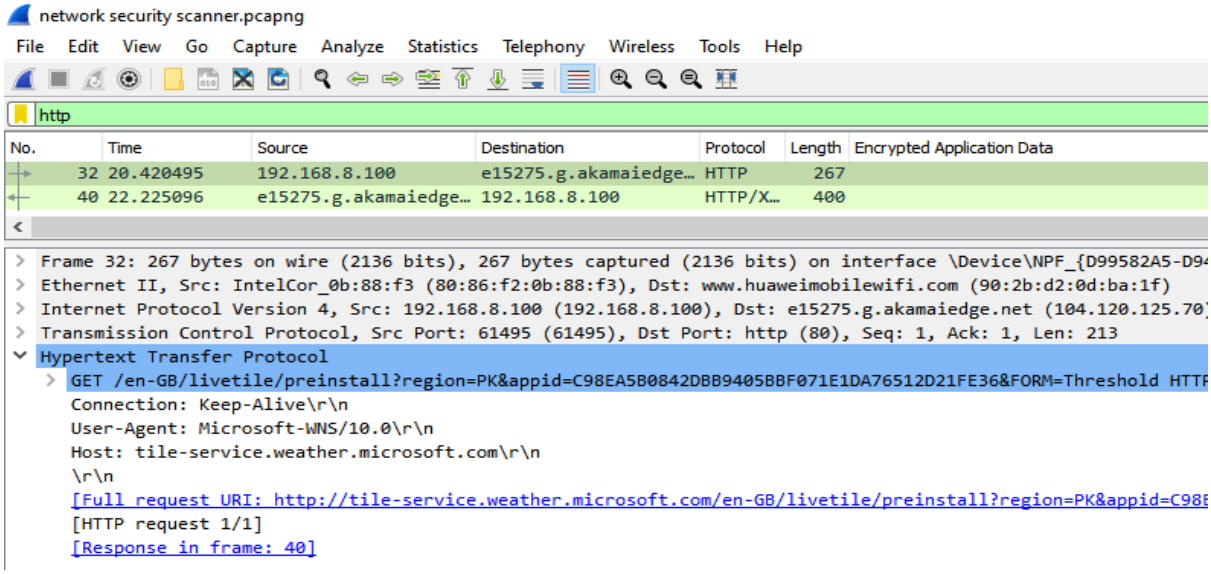
7.1.1 ARP Port scanning



7.1.2 Password identification



7.1.3 Outdated software identification

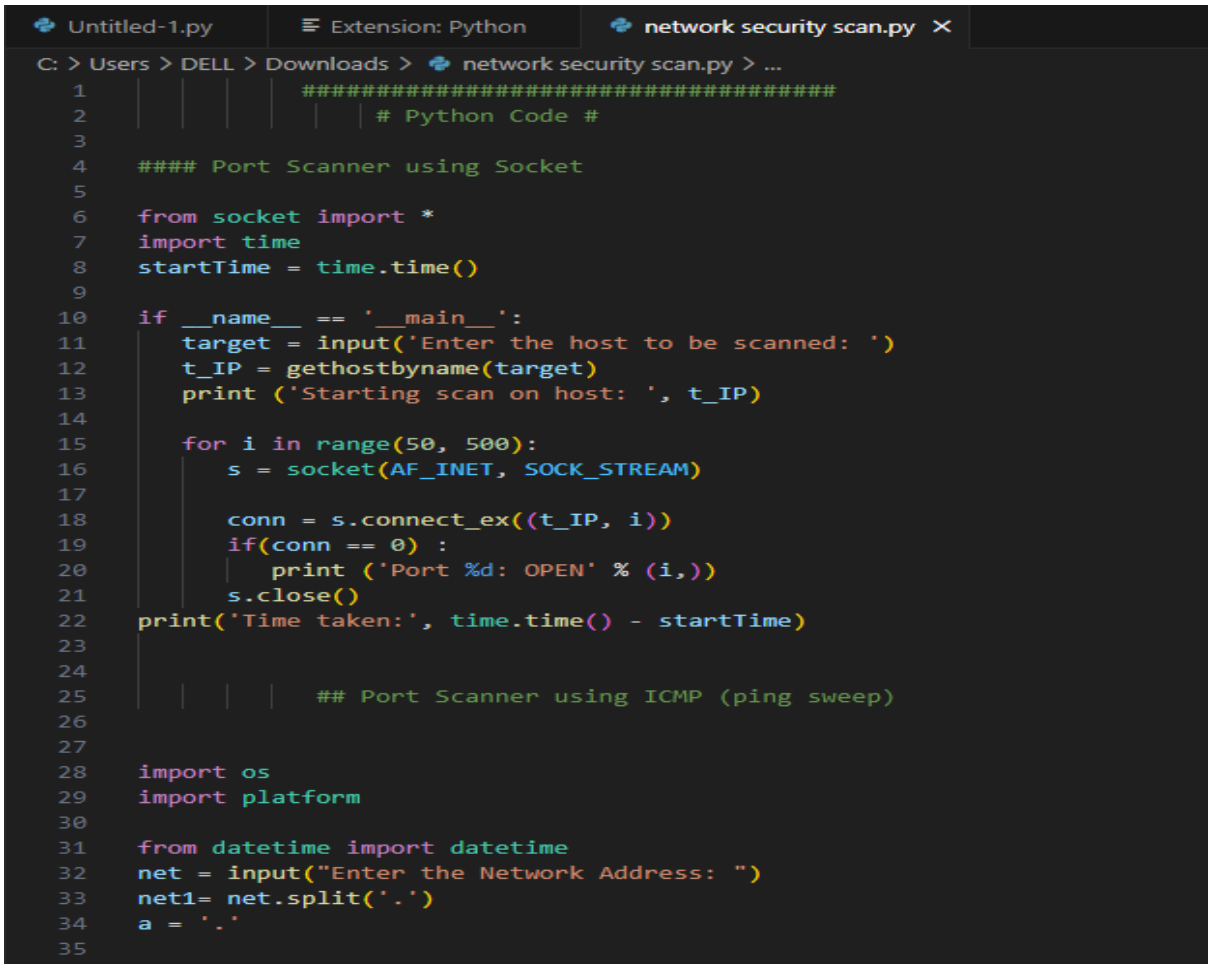


7.2 Python implementation:

7.2.1 Python Code

This file is for scanning the vulnerabilities of a network.

networkSecurityScan.py



```
net2 = net1[0] + a + net1[1] + a + net1[2] + a
st1 = int(input("Enter the Starting Number: "))
en1 = int(input("Enter the Last Number: "))
en1 = en1 + 1
oper = platform.system()

if (oper == "Windows"):
    ping1 = "ping -n 1 "
elif (oper == "Linux"):
    ping1 = "ping -c 1 "
else :
    ping1 = "ping -c 1 "
t1 = datetime.now()
print ("Scanning in Progress:")

for ip in range(st1,en1):
    addr = net2 + str(ip)
    comm = ping1 + addr
    response = os.popen(comm)

    for line in response.readlines():
        if(line.count("TTL")):
            break
        if (line.count("TTL")):
            print (addr, "--> Live")

t2 = datetime.now()
total = t2 - t1
print ("Scanning completed in: ",total)


        ### Port Scanner using TCP scan ###

import socket
from datetime import datetime
net = input("Enter the IP address: ")
net1 = net.split('.')
a = '.'

net2 = net1[0] + a + net1[1] + a + net1[2] + a
st1 = int(input("Enter the Starting Number: "))
en1 = int(input("Enter the Last Number: "))
en1 = en1 + 1
t1 = datetime.now()

def scan(addr):
    s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    socket.setdefaulttimeout(1)
    result = s.connect_ex((addr,135))
    if result == 0:
        return 1
    else :
        return 0

def run1():
    for ip in range(st1,en1):
        addr = net2 + str(ip)
        if (scan(addr)):
            print (addr , "is live")

run1()
t2 = datetime.now()
total = t2 - t1
print ("Scanning completed in: " , total)
```

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

Untitled-1.py | Extension: Python | network security scan.py | ports.py 2

C: > Users > DELL > Downloads > ports.py > ...

```python
#####################################
    # Python Code #

import socket


class portscan():
    banners = []
    open_ports = []
    def __init__(self, target, port_num):
        self.target = target
        self.port_num = port_num

    def scan(self):
        for port in range(1, self.port_num):
            self.scan_port(port)


    def check_ip(self):
        try:
            IP(self.target)
            return(self.target)
        except ValueError:
            return socket.gethostbyname(self.target)


    def scan_port(self, port):
        try:
            converted_ip = self.check_ip()
            sock = socket.socket()
            sock.settimeout(0.5)
            sock.connect((converted_ip, port))
            self.open_ports.append(port)
            try:
                banner = sock.recv(1024).decode().strip('\n').strip('\r')
                self.banners.append(banner)
            except:
                self.banners.append(' ')
            sock.close()
        except:
            pass

        import portscanner

targets_ip = input('[+] * Enter Target To Scan For Vulnerable Open Ports: ')
port_number = int(input('[+] * Enter Amount Of Ports You Want To Scan (500 - First 500 Ports): '))
vul_file = input('[+] * Enter Path To The File With Vulnerable Softwares: ')
print('\n')

target = portscanner.portscan(targets_ip, port_number)
target.scan()
with open(vul_file,'r') as file:
    count = 0
    for banner in target.banners:
        file.seek(0)
        for line in file.readlines():
            if line.strip() in banner:
                print('[!!] VULNERABLE BANNER: "' + banner + '" ON PORT: ' + str(target.open_ports[count]))
        count += 1

```

7.2.4 Output:

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.

[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#.

[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.

[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. Greebone 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 Manger. 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.