# Penetration Testing Logbook

*Student:*
Alessandro Buonerba

*Module Leader:*
Dr Anatolij Bezemskij

Computer Science (Cybersecurity)
Penetration Testing and Ethical Vulnerability Scanning
COMP-1671

Department of Computing & Mathematical Sciences
Liberal Arts & Sciences

University of Greenwich
London, United Kingdom

December 2021

# CONTENTS

# LIST OF FIGURES

# 1 LAB 1: PASSIVE ENUMERATION

Contrarily from active enumeration, passive enumeration is a technique that does not rely on explicit communication with a target system (Cooper, 2020). To perform a passive enumeration, a network monitor tool such as Wireshark is often used.

## 1.1 CONNECT TO FTP

The first part of the task is to connect to the FTP server and download the .pcap file with all the captured network traffic.

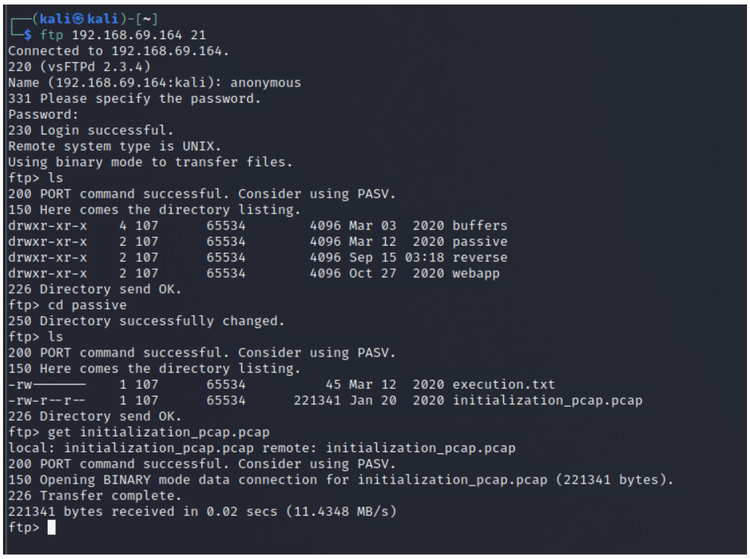

```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.69.164 21
Connected to 192.168.69.164.
220 (vsFTPd 2.3.4)
Name (192.168.69.164:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    4 107      65534        4096 Mar 03  2020 buffers
drwxr-xr-x    2 107      65534        4096 Mar 12  2020 passive
drwxr-xr-x    2 107      65534        4096 Sep 15 03:18 reverse
drwxr-xr-x    2 107      65534        4096 Oct 27  2020 webapp
226 Directory send OK.
ftp> cd passive
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-------    1 107      65534          45 Mar 12  2020 execution.txt
-rw-r--r--    1 107      65534      221341 Jan 20  2020 initialization_pcap.pcap
226 Directory send OK.
ftp> get initialization_pcap.pcap
local: initialization_pcap.pcap remote: initialization_pcap.pcap
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for initialization_pcap.pcap (221341 bytes).
226 Transfer complete.
221341 bytes received in 0.02 secs (11.4348 MB/s)
ftp>
```

Figure 1.1: Connect to the FTP and get the .pcap file

Now that the file has been downloaded, it can be found in the home directory and we can start the analysis of the network traffic through Wireshark following the tasks assigned to this lab.

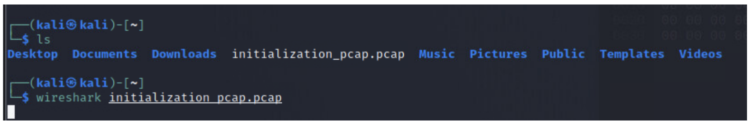
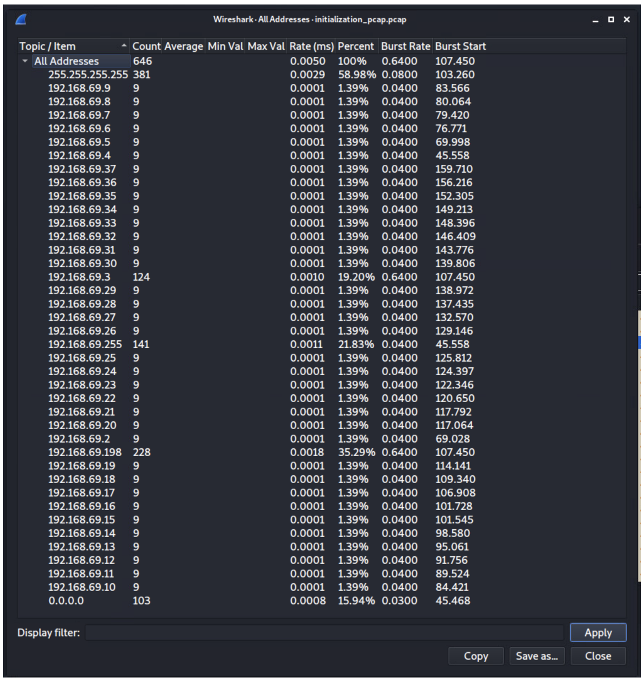

```
┌──(kali㉿kali)-[~]
└─$ ls
Desktop  Documents  Downloads  initialization_pcap.pcap  Music  Pictures  Public  Templates  Videos

┌──(kali㉿kali)-[~]
└─$ wireshark initialization_pcap.pcap
```

Figure 1.2: Open .pcap with Wireshark

## 1.2 FIND UNIQUE IPV4 ADDRESSES

The first tasks asks to find the unique IPs that are stored and captured. We can achieve that through the top menu, selecting statistics and IPv4 addresses. The result is shown in the figure below.



Wireshark · All Addresses · initialization_pcap.pcap

| Topic / Item | Count | Average | Min Val | Max Val | Rate (ms) | Percent | Burst Rate | Burst Start |
|---|---|---|---|---|---|---|---|---|
| All Addresses | 646 | | | | 0.0050 | 100% | 0.6400 | 107.450 |
| 255.255.255.255 | 381 | | | | 0.0029 | 58.98% | 0.0800 | 103.260 |
| 192.168.69.9 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 83.566 |
| 192.168.69.8 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 80.064 |
| 192.168.69.7 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 79.420 |
| 192.168.69.6 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 76.771 |
| 192.168.69.5 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 69.998 |
| 192.168.69.4 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 45.558 |
| 192.168.69.37 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 159.710 |
| 192.168.69.36 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 156.216 |
| 192.168.69.35 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 152.305 |
| 192.168.69.34 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 149.213 |
| 192.168.69.33 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 148.396 |
| 192.168.69.32 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 146.409 |
| 192.168.69.31 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 143.776 |
| 192.168.69.30 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 139.806 |
| 192.168.69.3 | 124 | | | | 0.0010 | 19.20% | 0.6400 | 107.450 |
| 192.168.69.29 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 138.972 |
| 192.168.69.28 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 137.435 |
| 192.168.69.27 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 132.570 |
| 192.168.69.26 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 129.146 |
| 192.168.69.255 | 141 | | | | 0.0011 | 21.83% | 0.0400 | 45.558 |
| 192.168.69.25 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 125.812 |
| 192.168.69.24 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 124.397 |
| 192.168.69.23 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 122.346 |
| 192.168.69.22 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 120.650 |
| 192.168.69.21 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 117.792 |
| 192.168.69.20 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 117.064 |
| 192.168.69.2 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 69.028 |
| 192.168.69.198 | 228 | | | | 0.0018 | 35.29% | 0.6400 | 107.450 |
| 192.168.69.19 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 114.141 |
| 192.168.69.18 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 109.340 |
| 192.168.69.17 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 106.908 |
| 192.168.69.16 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 101.728 |
| 192.168.69.15 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 101.545 |
| 192.168.69.14 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 98.580 |
| 192.168.69.13 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 95.061 |
| 192.168.69.12 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 91.756 |
| 192.168.69.11 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 89.524 |
| 192.168.69.10 | 9 | | | | 0.0001 | 1.39% | 0.0400 | 84.421 |
| 0.0.0.0 | 103 | | | | 0.0008 | 15.94% | 0.0300 | 45.468 |

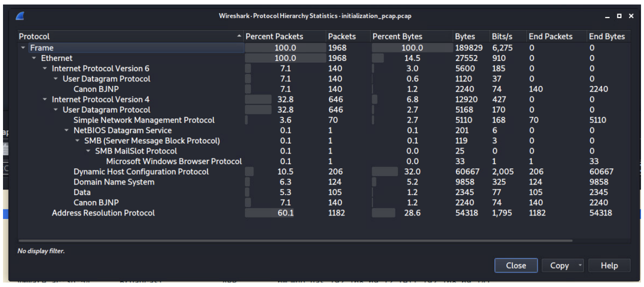
Display filter: Apply

Copy Save as… Close

Figure 1.3: Unique IPv4 addresses

## 1.3 APPLICATION-LAYER PROTOCOLS

The second task asks to find the application-layer protocols that are used in the captured network traffic. This can be displayed using the Protocol Hierarchy command. The result is shown in the figure below.



Wireshark · Protocol Hierarchy Statistics · initialization_pcap.pcap

| Protocol | Percent Packets | Packets | Percent Bytes | Bytes | Bits/s | End Packets | End Bytes |
|---|---|---|---|---|---|---|---|
| Frame | 100.0 | 1968 | 100.0 | 189829 | 6,275 | 0 | 0 |
| Ethernet | 100.0 | 1968 | 14.5 | 27552 | 910 | 0 | 0 |
| Internet Protocol Version 6 | 7.1 | 140 | 3.0 | 5600 | 185 | 0 | 0 |
| User Datagram Protocol | 7.1 | 140 | 0.6 | 1120 | 37 | 0 | 0 |
| Canon BJNP | 7.1 | 140 | 1.2 | 2240 | 74 | 140 | 2240 |
| Internet Protocol Version 4 | 32.8 | 646 | 6.8 | 12920 | 427 | 0 | 0 |
| User Datagram Protocol | 32.8 | 646 | 2.7 | 5168 | 170 | 0 | 0 |
| Simple Network Management Protocol | 3.6 | 70 | 2.7 | 5110 | 168 | 70 | 5110 |
| NetBIOS Datagram Service | 0.1 | 1 | 0.1 | 201 | 6 | 0 | 0 |
| SMB (Server Message Block Protocol) | 0.1 | 1 | 0.1 | 119 | 3 | 0 | 0 |
| SMB MailSlot Protocol | 0.1 | 1 | 0.0 | 25 | 0 | 0 | 0 |
| Microsoft Windows Browser Protocol | 0.1 | 1 | 0.0 | 33 | 1 | 1 | 33 |
| Dynamic Host Configuration Protocol | 10.5 | 206 | 32.0 | 60667 | 2,005 | 206 | 60667 |
| Domain Name System | 6.3 | 124 | 5.2 | 9858 | 325 | 124 | 9858 |
| Data | 5.3 | 105 | 1.2 | 2345 | 77 | 105 | 2345 |
| Canon BJNP | 7.1 | 140 | 1.2 | 2240 | 74 | 140 | 2240 |
| Address Resolution Protocol | 60.1 | 1182 | 28.6 | 54318 | 1,795 | 1182 | 54318 |

No display filter.

Close Copy Help

Figure 1.4: Protocol Hierarchy

## 1.4 NAME OF THE PROTOCOLS

The application-layer protocol are the following.

- SNMP (Single Network Management Protocol): responsible for the management of network devices, allows the communication between them independently of their spec (Scarpati, 2020).
- DNS (Domain Name System): responsible for the resolution of domain names to IP addresses (Insam, 2020).
- DHCP (Dynamic Host Configuration Protocol): responsible for the dynamic configuration of network devices. This protocol is used to automatically assign IPs to network devices (IBM, 2021).
- SMB (Server Message Block): responsible for the communication between shared devices such as printers on a network (Sheldon and Scarpati, 2020).

## 1.5 NETWORK DIAGRAM

This task will allow us to have a visual representation of the analysis of the network. Below the diagram with the active protocols and devices.

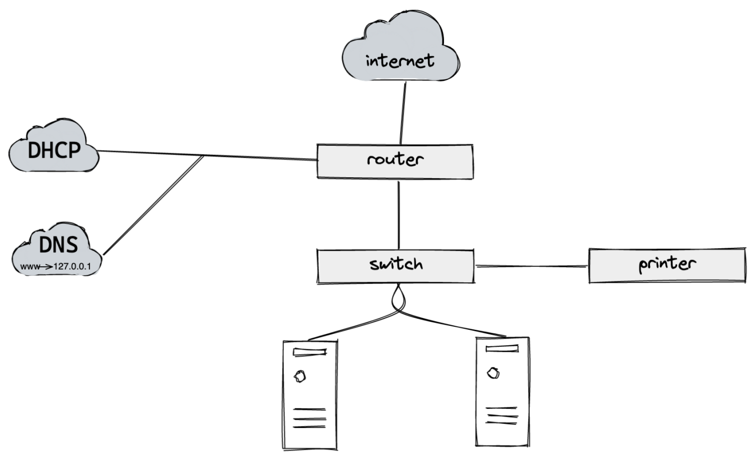



Figure 1.5: Network Diagram

Following the explanation of the protocols, further analysis portrays the use of internet protocol. The protocols in use are UDP, SNMP, DHCP and DNS, meaning computers and shared devices on the network. We can also certify using a BJNP protocol, meaning that the shared device on the network is a Canon printer.

## 1.6 DISCUSSION

The network traffic analysis suggests that a user uses the shared device since there is a BJNP protocol. There are also ACKs and NAKs portraying active communication between the devices of the network. Some of the UDP packets were broadcasting an std discovery all to find all the services on the network.

## 1.7 TCP DUMP

Following the instructions and the man page for the tcpdump command, I have been able to reproduce a one liner to output a number of unique MAC addresses in the provided and previously used .pcap file. Below a picture with the result.

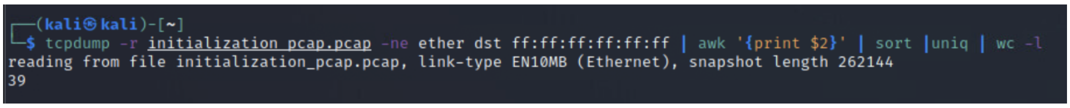



Figure 1.6: TCP Dump

The flag `-r` is used to read the file and the flag `-ne` before `ether dst` looks for ethernet destinations with the MAC address format specified right after it. The command `awk` is used to separate them while printing the second argument to get the second column. It will then sort and check for unique entries for then count everything with the last `wc -l` command

## 1.8 REFLECTION

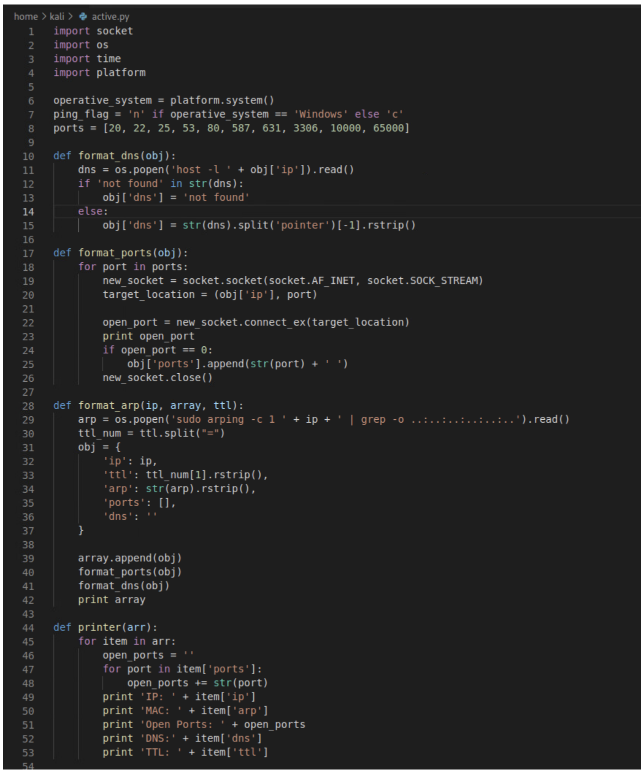
This has been a very fun lab. I have learned a lot more about Wireshark and how to analyse a .pcap file. Even though I have never used `tcpdump`, there were manyexamples and exhaustive official documentation.

# 2 LAB 2: ACTIVE ENUMERATION

## 2.1 INTRODUCTION

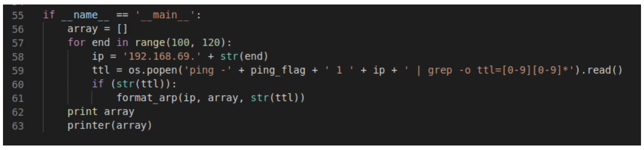
Active enumeration is when a user programmatically gather informations on a system through the use of a set of predefined commands.The most common set of informations that is usually gathered through enumeration are DNS, IPs, ports, and services.

## 2.2 CODE



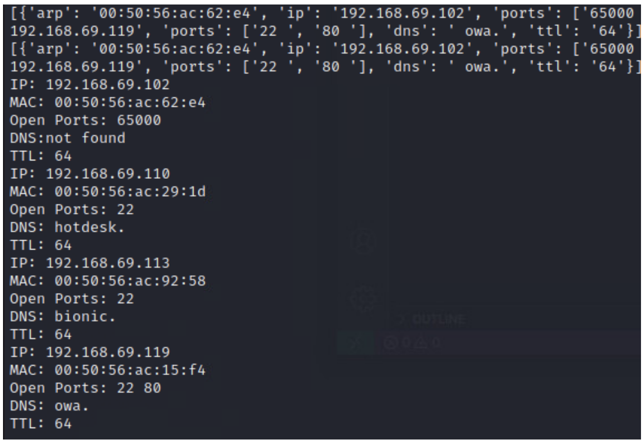
```python
home > kali > active.py
import socket
import os
import time
import platform

operative_system = platform.system()
ping_flag = 'n' if operative_system == 'Windows' else 'c'
ports = [20, 22, 25, 53, 80, 587, 631, 3306, 10000, 65000]

def format_dns(obj):
    dns = os.popen('host -l ' + obj['ip']).read()
    if 'not found' in str(dns):
        obj['dns'] = 'not found'
    else:
        obj['dns'] = str(dns).split('pointer')[-1].rstrip()

def format_ports(obj):
    for port in ports:
        new_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        target_location = (obj['ip'], port)

        open_port = new_socket.connect_ex(target_location)
        print open_port
        if open_port == 0:
            obj['ports'].append(str(port) + ' ')
        new_socket.close()

def format_arp(ip, array, ttl):
    arp = os.popen('sudo arping -c 1 ' + ip + ' | grep -o ..:..:..:..:..:..').read()
    ttl_num = ttl.split("=")
    obj = {
        'ip': ip,
        'ttl': ttl_num[1].rstrip(),
        'arp': str(arp).rstrip(),
        'ports': [],
        'dns': ''
    }

    array.append(obj)
    format_ports(obj)
    format_dns(obj)
    print array

def printer(arr):
    for item in arr:
        open_ports = ''
        for port in item['ports']:
            open_ports += str(port)
        print 'IP: ' + item['ip']
        print 'MAC: ' + item['arp']
        print 'Open Ports: ' + open_ports
        print 'DNS:' + item['dns']
        print 'TTL: ' + item['ttl']
```

Figure 2.1: Functions of the Program



```python
if __name__ == '__main__':
    array = []
    for end in range(100, 120):
        ip = '192.168.69.' + str(end)
        ttl = os.popen('ping -' + ping_flag + ' 1 ' + ip + ' | grep -o ttl=[0-9][0-9]*').read()
        if (str(ttl)):
            format_arp(ip, array, str(ttl))
    print array
    printer(array)
```

Figure 2.2: Main of the Program



```
[{'arp': '00:50:56:ac:62:e4', 'ip': '192.168.69.102', 'ports': ['65000
192.168.69.119', 'ports': ['22 ', '80 '], 'dns': ' owa.', 'ttl': '64'}]
[{'arp': '00:50:56:ac:62:e4', 'ip': '192.168.69.102', 'ports': ['65000
192.168.69.119', 'ports': ['22 ', '80 '], 'dns': ' owa.', 'ttl': '64'}]
IP: 192.168.69.102
MAC: 00:50:56:ac:62:e4
Open Ports: 65000
DNS:not found
TTL: 64
IP: 192.168.69.110
MAC: 00:50:56:ac:29:1d
Open Ports: 22
DNS: hotdesk.
TTL: 64
IP: 192.168.69.113
MAC: 00:50:56:ac:92:58
Open Ports: 22
DNS: bionic.
TTL: 64
IP: 192.168.69.119
MAC: 00:50:56:ac:15:f4
Open Ports: 22 80
DNS: owa.
TTL: 64
```

Figure 2.3: program-result

# BIBLIOGRAPHY


Cooper, Zach (2020). *What's the Difference between Active and Passive Reconnaissance?* URL: https://www.itpro.co.uk/penetration-testing/34465/whats-the-difference-between-active-and-passive-reconnaissance (visited on 10/07/2021).

IBM (2021). *IBM Docs*. URL: https://prod.ibmdocs-production-dal-6099123ce774e592a519d7c33db8265e-0000.us-south.containers.appdomain.cloud/docs/en/aix/7.1?topic=tcpp-tcpip-address-parameter-assignment-dynamic-host-configuration-protocol (visited on 10/07/2021).

Insam, Edward (2020). *Application Layer Protocol - an Overview*. URL: https://www.sciencedirect.com/topics/computer-science/application-layer-protocol (visited on 10/07/2021).

Scarpati, Jessica (2020). *What Is Simple Network Management Protocol (SNMP)? Definition from SearchNetworking*. URL: https://www.techtarget.com/searchnetworking/definition/SNMP (visited on 10/07/2021).

Sheldon, Robert and Jessica Scarpati (2020). *What Is the Server Message Block (SMB) Protocol? How Does It Work?* URL: https://www.techtarget.com/searchnetworking/definition/Server-Message-Block-Protocol (visited on 10/07/2021).