



YoungDev Interns

Cyber Security Tasks

01 Month Remote Internship



Report on Internship Completion YoungDev Interns - Cyber Security

Intern Name: Abdullah hassan

Internship Duration: 1 Month

Internship Type: Remote

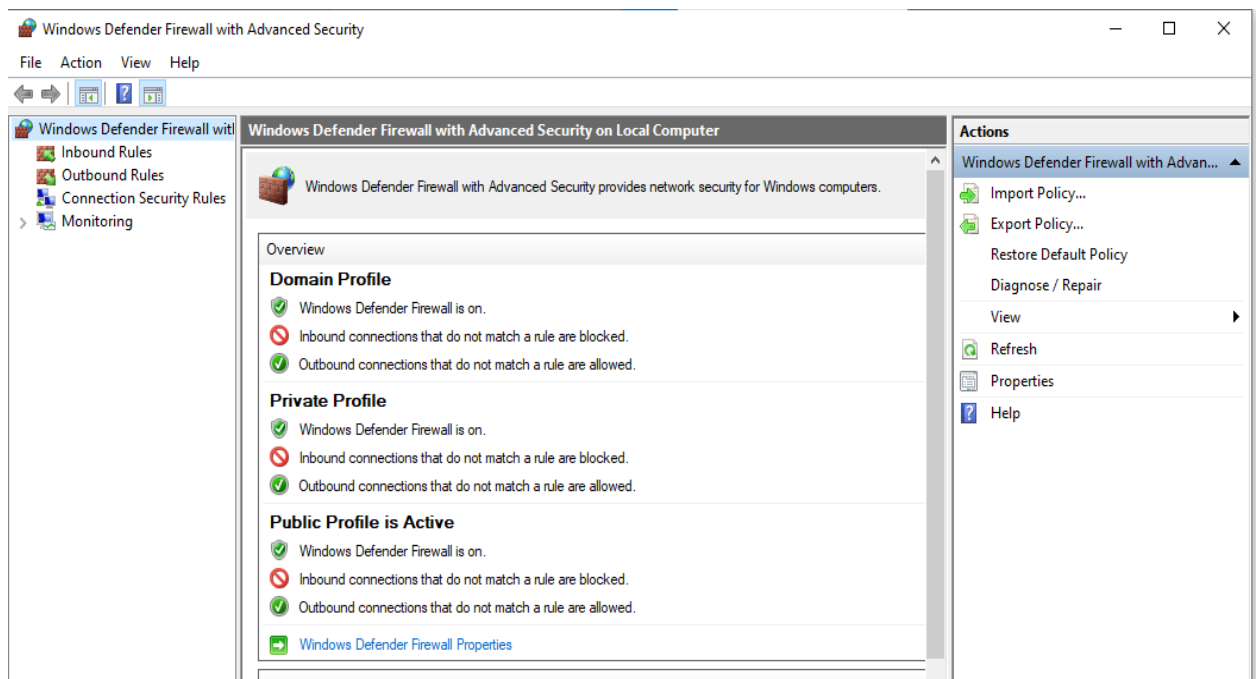
Basic Tasks

Task 1: Set Up Basic Firewall Rules

What I Did:

To strengthen my computer's security, I worked on configuring the built-in Windows Defender Firewall. I carefully navigated through the firewall settings and enabled key security features.

1. I started by accessing the "Advanced Settings" in Windows Firewall.
2. Created both "Inbound" and "Outbound" rules for specific applications to ensure only trusted apps were allowed to communicate.



3. Configured rules to block certain ports, preventing potential unauthorized access to the system.
4. Tested the rules to verify they worked as expected, blocking applications and ports as per my setup.

This activity helped me understand the importance of firewalls in protecting personal and enterprise-level systems.

Task 2: Use a Password Manager

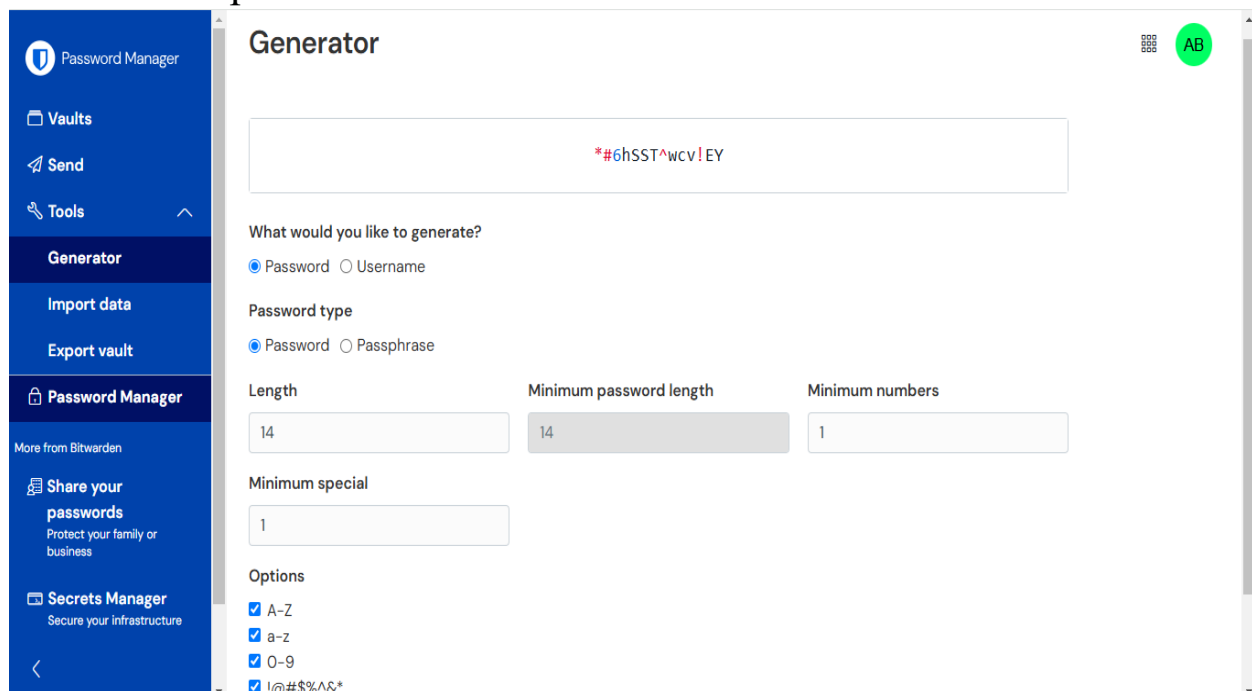
What I Did:

I installed Bitwarden, a trusted and secure password manager, to handle and organize my passwords effectively.

1. Downloaded and installed Bitwarden from its official website.
2. Set up my personal account and started adding login credentials for my online services like email, social media, and other platforms.
3. Experimented with the built-in password generator to create strong and unique passwords.
4. Verified its autofill feature, which securely entered my login details on various websites.

Using Bitwarden simplified password management for me and highlighted how password managers are an essential tool in cybersecurity.

Status: Completed



The screenshot shows the Bitwarden Password Manager interface. On the left is a blue sidebar with navigation options: Password Manager, Vaults, Send, Tools, Generator (highlighted), Import data, Export vault, and Password Manager (with a lock icon). Below these are links for 'Share your passwords' and 'Secrets Manager'. The main area is titled 'Generator' and features a text box displaying a generated password: `*#6hSST^wcv!EY`. Below this, there are settings for 'What would you like to generate?' (Password selected), 'Password type' (Password selected), 'Length' (14), 'Minimum password length' (14), 'Minimum numbers' (1), and 'Minimum special' (1). Under 'Options', checkboxes for 'A-Z', 'a-z', '0-9', and '!@#\$\$%^&*' are all checked. A green circular button with 'AB' is visible in the top right corner of the main area.

Task 3: Identify Phishing Emails

What I Did:

The goal was to learn how to spot phishing attempts effectively. To achieve this:

1. I reviewed guides and examples of phishing emails, identifying traits like fake sender addresses, suspicious links, and urgent messages designed to create panic.
2. Analyzed real-world phishing emails by using phishing test kits.
3. Practiced reporting fake emails by using “Report Phishing” features available in popular email services.

This task sharpened my ability to differentiate between legitimate and fraudulent messages, boosting my online safety awareness.

Status: Completed

Phishing IQ Test

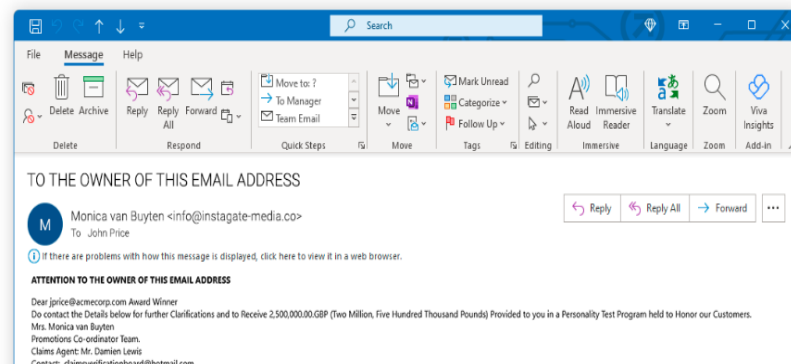
Phishing or Legitimate?

Please select if the following email is legitimate or a phishing attack.

1 2 3 4 5 6 7 8 9 10

Phishing

Legitimate



Intermediate Tasks

Task 1: Perform a Basic Vulnerability Scan

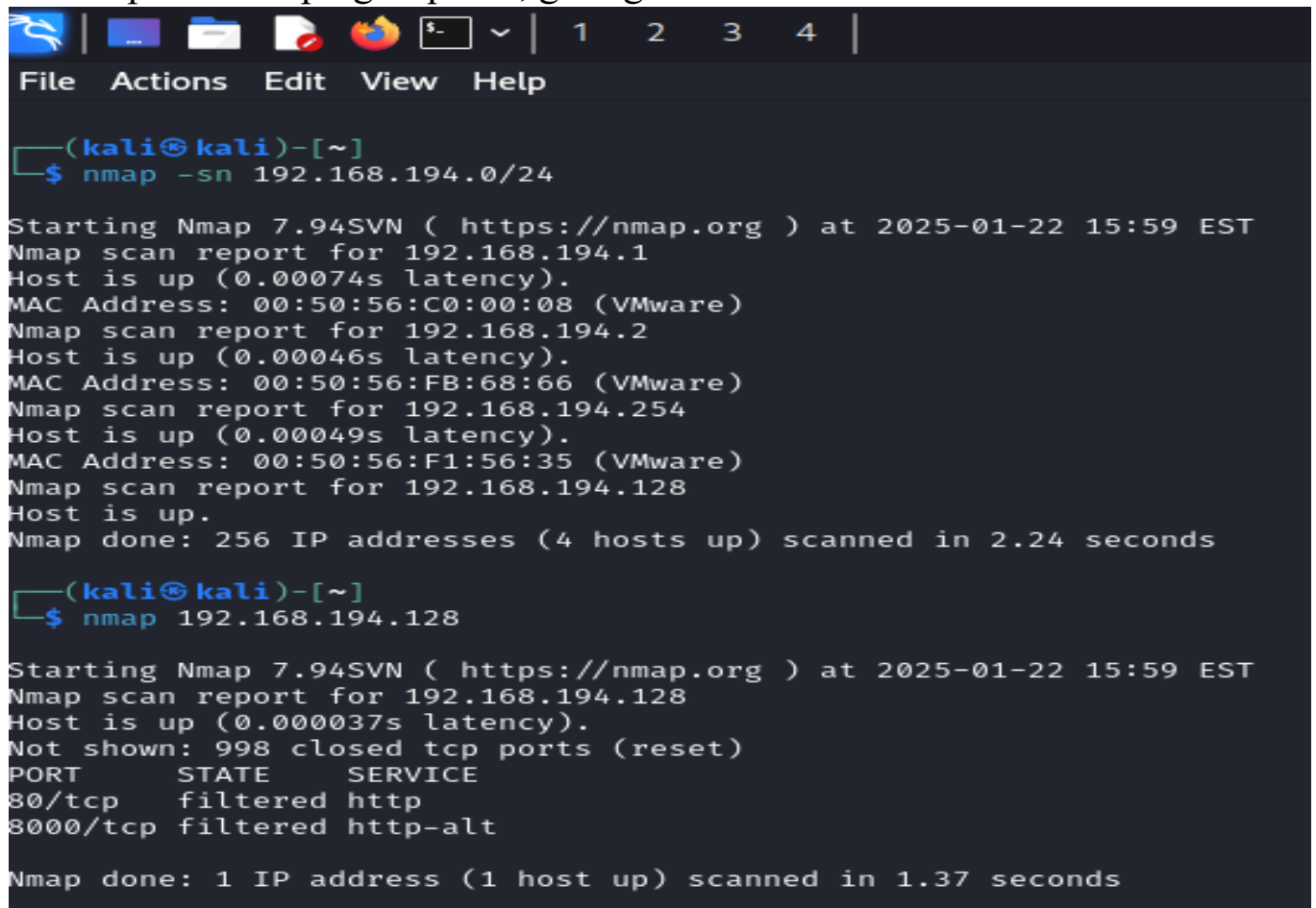
I used Nmap, a powerful network scanning tool, to explore vulnerabilities on a test network. Here's how I conducted the scans steps

1. Basic Scan:

I started with the simplest command, `nmap 192.168.194.128`, to scan a single host. This command provided a basic overview of the open ports and services running on the target machine.

2. Ping Sweep:

To identify all active devices on the network, I used the command `nmap -sn 192.168.194.0/24`. This scan helped me map the network, find devices that responded to ping requests, giving me a list of reachable hosts.

A screenshot of a Kali Linux terminal window. The window has a dark background with a menu bar at the top containing 'File', 'Actions', 'Edit', 'View', and 'Help'. Below the menu bar is a toolbar with icons for a terminal, a folder, a document, a search icon, and a dropdown menu. The terminal shows two Nmap commands and their outputs. The first command is `nmap -sn 192.168.194.0/24`, which performs a ping sweep. The output shows that four hosts are up: 192.168.194.1, 192.168.194.2, 192.168.194.254, and 192.168.194.128, all with VMWare MAC addresses. The second command is `nmap 192.168.194.128`, which performs a basic scan on a single host. The output shows that the host is up and lists open ports: 80/tcp (filtered http) and 8000/tcp (filtered http-alt).

```
(kali㉿kali)-[~]  
$ nmap -sn 192.168.194.0/24  
  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-22 15:59 EST  
Nmap scan report for 192.168.194.1  
Host is up (0.00074s latency).  
MAC Address: 00:50:56:C0:00:08 (VMware)  
Nmap scan report for 192.168.194.2  
Host is up (0.00046s latency).  
MAC Address: 00:50:56:FB:68:66 (VMware)  
Nmap scan report for 192.168.194.254  
Host is up (0.00049s latency).  
MAC Address: 00:50:56:F1:56:35 (VMware)  
Nmap scan report for 192.168.194.128  
Host is up.  
Nmap done: 256 IP addresses (4 hosts up) scanned in 2.24 seconds  
  
(kali㉿kali)-[~]  
$ nmap 192.168.194.128  
  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-22 15:59 EST  
Nmap scan report for 192.168.194.128  
Host is up (0.000037s latency).  
Not shown: 998 closed tcp ports (reset)  
PORT      STATE      SERVICE  
80/tcp    filtered  http  
8000/tcp   filtered  http-alt  
  
Nmap done: 1 IP address (1 host up) scanned in 1.37 seconds
```

3. Aggressive Scan:

Finally, I performed a detailed scan using `nmap -A 192.168.194.128`. This command enabled aggressive scanning, including OS detection, version detection, script scanning, and traceroute. It provided comprehensive information about the target machine, including its services and vulnerabilities.

```
(kali㉿kali)-[~/Downloads]
$ sudo nmap -A 192.168.194.128

Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-22 16:07 EST
Nmap scan report for 192.168.194.128
Host is up (0.00013s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE      SERVICE  VERSION
80/tcp    filtered  http
8000/tcp   filtered  http-alt
Too many fingerprints match this host to give specific OS details
Network Distance: 0 hops

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 3.91 seconds
```

Observations:

These scans gave me valuable insights into the network and its security posture. For example, I could identify open ports and their associated services, which could potentially be exploited if left unpatched or misconfigured.

This task enhanced my understanding of how attackers map and assess systems and how defenders can proactively secure their infrastructure.

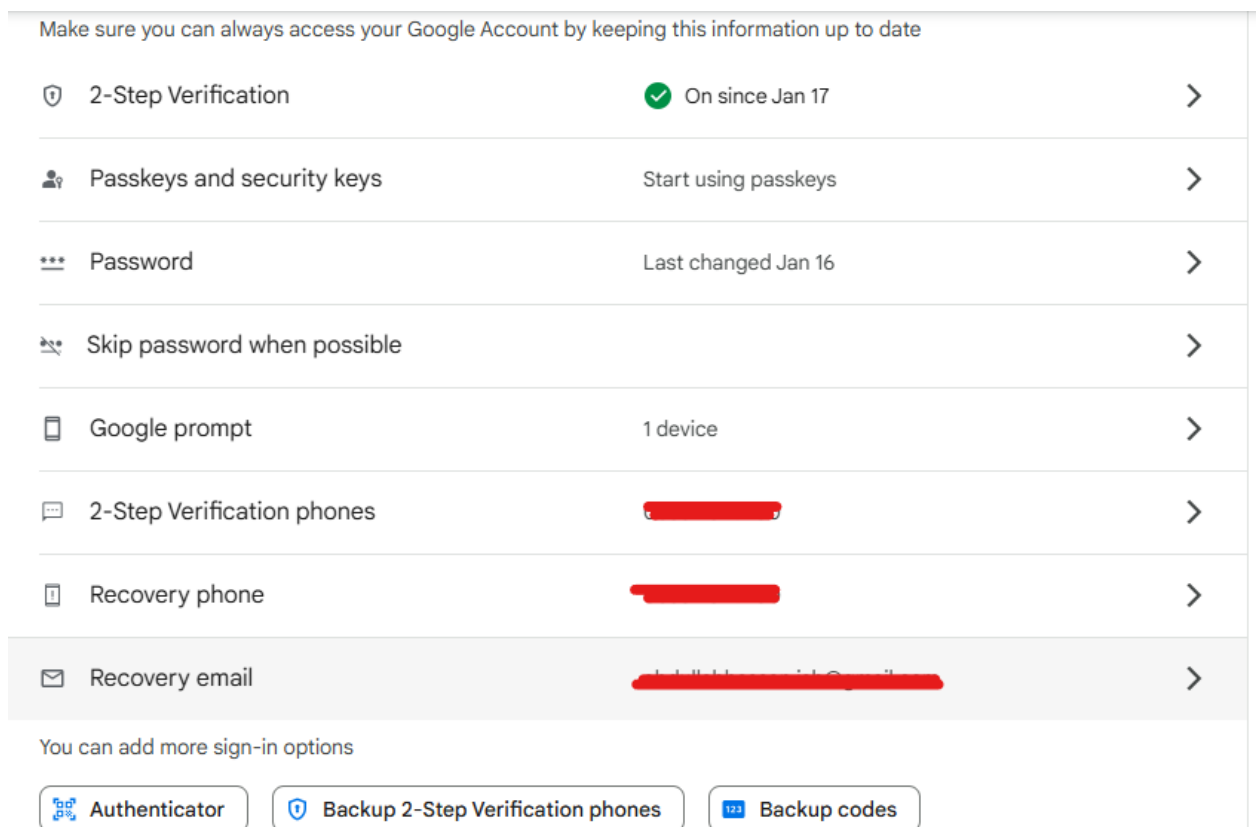
Status: Completed

Task 2: Implement Two-Factor Authentication (2FA)

What I Did:

For this task, I set up two-factor authentication (2FA) on some of my personal accounts to improve account security.

1. Enabled 2FA on my Gmail and Facebook accounts by navigating to their respective security settings.



2. Downloaded and configured an authenticator app, such as Google Authenticator, to generate time-based codes.
3. Tested 2FA to ensure that I could log in securely using both the password and the code from the app.

Enabling 2FA showed me how an additional layer of security could prevent unauthorized access even if passwords are compromised.

Status: Completed

January 17, 1:35 AM

Signing in with 2-Step Verification was turned on

To make sure you don't get locked out of your Google Account, set up a backup phone and get backup codes. [Learn more](#)



Windows

Pakistan

✓ This device

[Manage settings](#)

Task 3: Analyze Network Traffic

What I Did:

To monitor and analyze network traffic, I used Wireshark to capture and interpret packets. As part of the task, I performed a practical test on an HTTP website to see how insecure traffic can be intercepted.

1. Set up Wireshark and selected the active network interface for capturing live network data.
2. Navigated to <http://testphp.vulnweb.com/login.php>, an intentionally vulnerable website designed for testing.

3. Entered test credentials (a username and password) on the login page and captured the HTTP traffic in Wireshark.

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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	142.250.181.174	192.168.193.196	TLSv1.2	127	Application Data
2	0.156662	192.168.193.196	192.168.193.234	DNS	79	Standard query 0xd6d6 A time.cloudflare.com
3	0.157320	192.168.193.196	192.168.193.234	DNS	79	Standard query 0x7678 AAAA time.cloudflare.com
4	0.191881	192.168.193.234	192.168.193.196	DNS	111	Standard query response 0xd6d6 A time.cloudflare.com A 162.159.200.123 A 162.159.200.1
5	0.198611	192.168.193.234	192.168.193.196	DNS	135	Standard query response 0x7678 AAAA time.cloudflare.com AAAA 2606:4700:f1::123 AAAA 2606:4700:f1::1
6	0.199671	192.168.193.196	162.159.200.123	NTP	98	NTP Version 4, client
7	0.252866	162.159.200.123	192.168.193.196	NTP	98	NTP Version 4, server
8	1.433706	142.250.185.42	192.168.193.196	TLSv1.2	127	Application Data
9	2.047906	142.250.181.74	192.168.193.196	TLSv1.2	127	Application Data
10	2.165345	192.168.193.196	172.16.0.2	TCP	66	50771 → 1688 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
11	2.355788	142.250.185.42	192.168.193.196	TLSv1.2	127	Application Data
12	3.180414	192.168.193.196	172.16.0.2	TCP	66	[TCP Retransmission] 50771 → 1688 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
13	3.518916	192.168.193.196	188.138.125.61	TCP	66	50769 → 83 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
14	3.812002	2404:3100:1c01:34f7::2a00:1450:4018:809:...	...	TCP	74	50751 → 443 [FIN, ACK] Seq=1 Ack=1 Win=256 Len=0
15	3.814793	192.168.193.196	192.168.193.234	DNS	83	Standard query 0x2a68 AAAA safebrowsing.google.com

4. Applied the filter http in Wireshark to narrow down the displayed packets related to HTTP traffic.

http

No.	Time	Source	Destination	Protocol	Length	Info
18	3.817970	192.168.193.196	44.228.249.3	HTTP	523	GET /login.php HTTP/1.1
42	4.140734	44.228.249.3	192.168.193.196	HTTP	1342	HTTP/1.1 200 OK (text/html)
137	14.002541	192.168.193.196	44.228.249.3	HTTP	708	POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)
196	14.319387	44.228.249.3	192.168.193.196	HTTP	330	HTTP/1.1 302 Found (text/html)
197	14.328712	192.168.193.196	44.228.249.3	HTTP	570	GET /login.php HTTP/1.1
245	14.653662	44.228.249.3	192.168.193.196	HTTP	1342	HTTP/1.1 200 OK (text/html)

5. Located the login request packet and analyzed its contents in the packet details pane. I found the username and password I had entered in plain text, demonstrating the vulnerability of unsecured HTTP connections.

>	Frame 137: 708 bytes on wire (5664 bits), 708 bytes captured (5664 bits) on interface \Device\NPF{...}
>	Ethernet II, Src: Intel_8f:af:02 (14:ab:c5:8f:af:02), Dst: 7e:df:14:fb:e6:09 (7e:df:14:fb:e6:09)
>	Internet Protocol Version 4, Src: 192.168.193.196, Dst: 44.228.249.3
>	Transmission Control Protocol, Src Port: 50754, Dst Port: 80, Seq: 470, Ack: 2749, Len: 654
>	Hypertext Transfer Protocol
▼	HTML Form URL Encoded: application/x-www-form-urlencoded
>	Form item: "uname" = "abdullah"
>	Form item: "pass" = "air@123"

This task gave me a hands-on understanding of why HTTPS is crucial for secure communication. I also learned how cybersecurity tools like Wireshark can expose potential security risks in real time.

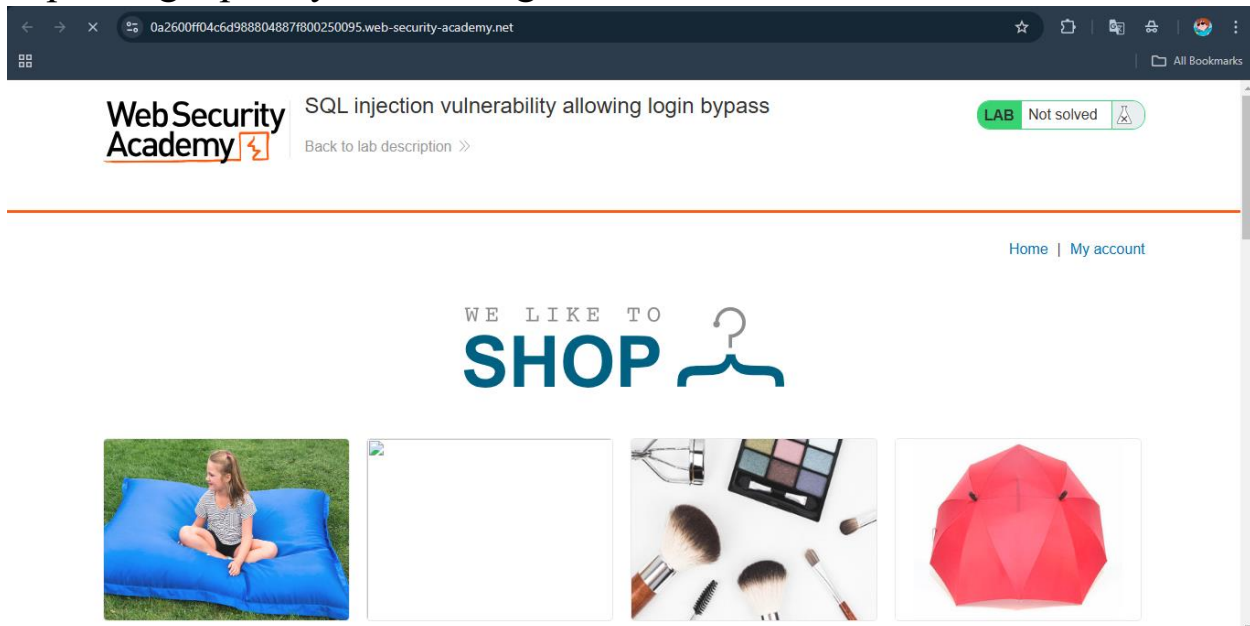
Status: Complete

Advance Tasks

Task 1

SQL Injection Lab and Practical Demonstration:

As part of my practical experience in identifying and exploiting SQL injection vulnerabilities, I performed a **SQL Injection lab** using **PortSwigger Web Security Academy**. This lab allowed me to simulate an attack and gain unauthorized access to an administrator account by exploiting a poorly secured login form.



Lab Objective:

The goal of the lab was to demonstrate how **SQL injection** can be used to bypass authentication systems. The application in the lab was vulnerable to SQL injection because user input in the **username** and **password** fields was directly inserted into an SQL query without sanitization.

The Vulnerable Query:

Upon reviewing the web application's login page, I identified that the form submission was processed by an SQL query similar to:

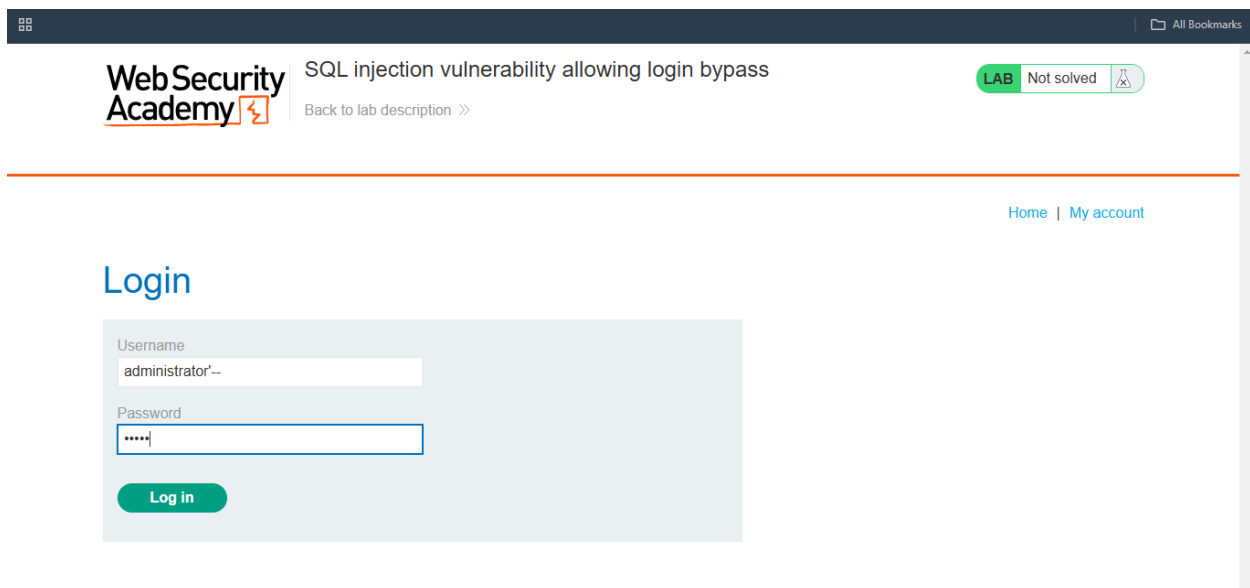
```
SELECT * FROM users WHERE username = 'input_username' AND password = 'input_password';
```

In this case, the values for username and password were being directly concatenated into the query string. This is the type of vulnerability that allows an attacker to inject malicious SQL code into the query, ultimately bypassing the authentication.

Exploiting the Vulnerability:

To perform a **SQL injection**, I entered the following payload into the **username** field:

```
administrator'--
```



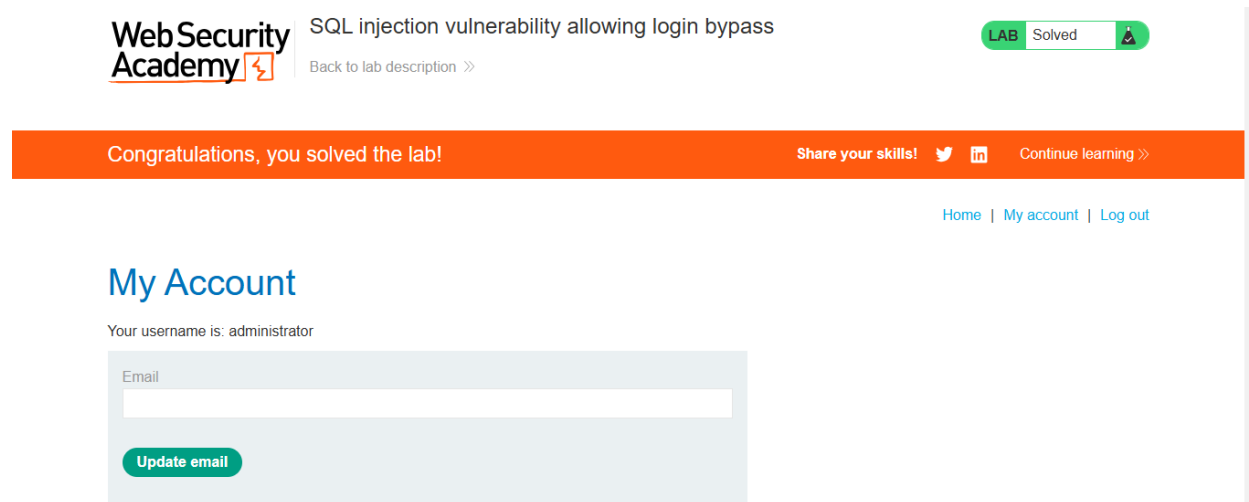
This SQL injection exploits the vulnerability by causing the SQL query to look like this after the payload is injected:

```
SELECT * FROM users WHERE username = 'administrator'--' AND password = 'anything';
```

- The '-- marks the start of a **SQL comment**, which causes the remaining part of the query (including the password check) to be ignored.

- The query now only checks if the **username** is 'administrator' and bypasses the need to validate the password.

Since 'administrator' is a valid username, the query now returns the administrator's record without requiring the correct password. By bypassing the password validation step, I successfully gained **administrator access**.



Results of the Exploit:

By injecting the SQL payload above, I gained access to the application as the **administrator**, showing how such an easily exploitable vulnerability could be used for unauthorized access. This exercise reinforced the need for securing web applications against SQL injection attacks, demonstrating the impact of leaving these vulnerabilities unchecked.

Action Taken:

After performing the lab and exploiting the vulnerability, I recognized the need for securing the web application in the real-world scenario. Here's what I did:

1. I ensured that **user inputs** (such as login credentials) were **parameterized** using **prepared statements** instead of being concatenated into SQL queries directly.
2. I reviewed all areas of the application where user input interacted with SQL queries and updated those sections to use **parameterized queries**, significantly reducing the attack surface.

This real-world lab demonstrated the importance of proper input handling and reinforced the necessity of following secure coding practices to prevent SQL injection vulnerabilities.

By performing the SQL injection lab, I gained hands-on experience with this common web application vulnerability and used that knowledge to implement stronger defenses, ensuring the security of the application from SQL injection attacks

Task 2

Incident Response Plan for YoungDev Interns

Purpose

This plan helps us manage and fix any security problems quickly and effectively. It makes sure we can handle issues without causing much damage and get back to normal operations fast.



Team Roles and Responsibilities

Team Members

1. Team Leader:

- Manages the entire process of solving the issue.
- Communicates updates to the company and other important people.

- Approves actions to fix the issue.

2. Problem Solver:

- Analyzes the situation and figures out what's wrong.
- Keeps records of what is found.
- Suggests solutions to stop the problem.

3. Communication Expert:

- Updates everyone in the company about the problem.
- Shares information with customers if needed.
- Handles public relations to protect the company's reputation.

4. Tech Expert:

- Collects evidence to find out what caused the issue.
- Makes sure laws and rules are followed.
- Tracks how the problem started.

5. IT Support:

- Fixes the systems to stop the problem.
- Minimizes interruptions to daily work.
- Helps restore everything back to normal.

6. Legal Advisor:

- Gives advice on legal responsibilities.
- Looks into any legal consequences.
- Ensures compliance with data protection laws.

Simple Steps to Solve Problems

1. Finding the Problem

- **Monitor:**
 - Use tools and logs to watch for anything unusual.
- **Report Issues:**
 - Train employees to report any suspicious activity.
- **Initial Check:**
 - Identify what kind of issue it is.
 - Find out which systems and data are affected.

2. Controlling the Problem

- **Immediate Actions:**
 - Disconnect affected systems from the network.
 - Block unauthorized access.
- **Temporary Fixes:**
 - Apply quick solutions to stop further damage.
 - Keep an eye on systems to ensure safety.

3. Solving the Root Cause

- **Investigate:**
 - Look for how the issue started.
 - Find the weak spots that allowed the problem.
- **Clean Up:**
 - Remove malware or block unauthorized users.
 - Install updates and security patches.

4. Recovering

- **Restoration:**
 - Restore systems using backup files.
 - Double-check systems are working properly.
- **Extra Precaution:**
 - Monitor activities closely after the recovery.
 - Make sure everything is running smoothly.

5. Learning and Improving

- **Record Everything:**
 - Write down what happened and how it was solved.
- **Review:**
 - Discuss what went well and what can improve.
- **Update Policies:**
 - Change or strengthen rules and security measures.

Real-Life Example

Imagine someone in the office clicks a suspicious email link, and it installs a virus on their computer. Here's how the plan works:

1. **Finding the Problem:** IT notices strange activity on the network.
2. **Controlling the Problem:** The infected computer is disconnected immediately.
3. **Solving the Root Cause:** IT finds the email and blocks similar ones.
4. **Recovering:** The computer is cleaned, and data is restored.

5. **Learning:** The company updates training to help employees avoid such emails in the future.

Conclusion

This simple plan helps YoungDev Interns quickly handle problems like viruses or hackers. It ensures minimal damage and helps everyone learn from the experience to avoid future issues.

Task 3 : Secure a Web Application

Implementing Content Security Policy (CSP) and HTTP Strict Transport Security (HSTS) headers to enhance security.

Deploying the website locally using node.js and exposing it to the internet using ngrok.

Detailed Steps Taken

1. Setting Up the Development Environment

- Installed Node.js and npm (Node Package Manager) on Kali Linux.
- Created a project directory named secure-web-app and initialized it:

```
mkdir secure-web-app
```

```
cd secure-web-app
```

```
npm init -y
```

- Installed necessary packages:

```
npm install express helmet
```

Express: Used to create the web server.

Helmet: Used to secure the application by setting HTTP headers.

2. Creating the Website

- Created a file named app.js in the project directory.
- Added the following content to define the web server:

```

const express = require('express');
const helmet = require('helmet');

const app = express();

// Use Helmet to secure headers
app.use(helmet());
app.use(
  helmet.contentSecurityPolicy({
    directives: {
      defaultSrc: ["'self'"],
      scriptSrc: ["'self'", "https://code.jquery.com"],
      styleSrc: ["'self'", "https://cdn.jsdelivr.net"],
    },
  })
);
app.use(helmet.hsts({ maxAge: 31536000 })); // Enable HSTS for 1 year

// Serve your page
app.get('/', (req, res) => {
  res.send(`
    <html>
      <head>
        <title>Abdullah Hassan - Cybersecurity Tips</title>
        <style>
          body { font-family: Arial, sans-serif; padding: 20px; }
          h1 { color: #333; }
          ul { list-style: none; padding: 0; }

```

```

          li { margin: 10px 0; }
        </style>
      </head>
      <body>
        <h1>Welcome! My Name is Abdullah Hassan</h1>
        <h2>Cybersecurity Tips</h2>
        <ul>
          <li>1. Use strong and unique passwords.</li>
          <li>2. Enable two-factor authentication (2FA).</li>
          <li>3. Keep your software updated.</li>
          <li>4. Avoid clicking on suspicious links.</li>
          <li>5. Use a VPN secure browsing.</li>
        </ul>
      </body>
    </html>
  `);
});

// Start the server
const PORT = 3000;
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});

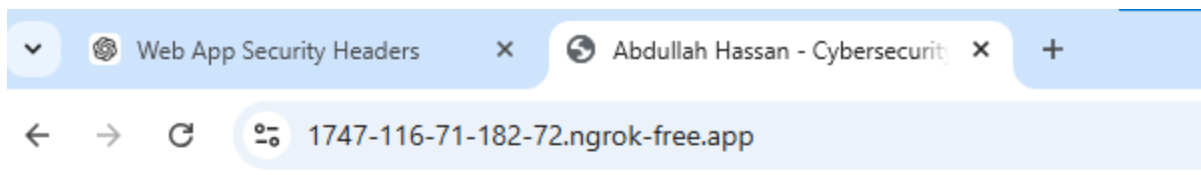
```

3. Running the Application Locally

- Started the server:
- `node app.js`
- Accessed the website on <http://localhost:3000>.

4. Exposing the Website Using Ngrok

- Installed ngrok and authenticated it with my Ngrok token.
- Started ngrok to expose the local server:
- `ngrok http 3000`
- Received a public URL (e.g., <https://abcd-1234.ngrok-free.app>) to access the site over the internet.



Welcome! My Name is Abdullah Hassan

Cybersecurity Tips

- 1. Use strong and unique passwords.
 - 2. Enable two-factor authentication (2FA).
 - 3. Keep your software updated.
 - 4. Avoid clicking on suspicious links.
 - 5. Use a VPN secure browsing.
-

5. Testing Security Headers

- Verified that CSP and HSTS headers were applied using browser developer tools (Network tab).

Conclusion

The task successfully demonstrated the ability to:

1. Create a secure web application using Node.js and Express.
 2. Implement Content Security Policy (CSP) and HTTP Strict Transport Security (HSTS) for enhanced security.
 3. Deploy the application locally and expose it to the internet using Ngrok.
 4. Configure a custom domain name for professional presentation.
-