**Redynox**

**Cyber Security Internship**

**1. Learn Network Security Concepts**

**Types of Network Threats**

| **Threat Type** | **Description** | **Example/Impact** |
| --- | --- | --- |
| **Virus** | A type of malicious software that attaches itself to clean files and spreads through a system. | Can corrupt files or slow down the network. |
| **Worm** | A self-replicating malware that spreads across networks without needing to attach to files. | Rapidly spreads and consumes bandwidth or system resources. |
| **Trojan Horse** | Malware disguised as legitimate software. It grants unauthorized access when executed. | Creates a backdoor for attackers to steal data or control the system. |
| **Phishing Attack** | A social engineering attack that tricks users into revealing sensitive info like passwords or credit card numbers. | Appears as a legitimate email or message (e.g., fake login pages). |

**Basic Security Concepts**

**Firewalls**

* Definition: A firewall monitors and controls incoming/outgoing network traffic based on predetermined security rules.
* **Types:**
  + **Software Firewalls:** Like Windows Defender Firewall.
  + **Hardware Firewalls:** Standalone devices that filter traffic before it reaches the internal network.
* **Function**: Blocks unauthorized access and monitors traffic.

**Encryption**

* **Definition:** The process of encoding data to prevent unauthorized access.
* **Types:**
  + Symmetric Encryption (same key to encrypt/decrypt) – e.g., AES.
  + Asymmetric Encryption (public/private key pair) – e.g., RSA.
* **Use Case: Secure email, VPNs, HTTPS, etc.**

**Secure Network Configurations**

* Password Protection: Use strong, complex passwords.
* Network Segmentation: Isolate sensitive systems from the rest of the network.
* Disable Unused Services: Reduces the attack surface.
* Regular Updates: Patch systems and devices to fix known vulnerabilities.
* Use of Antivirus/Antimalware Tools: Scans and removes threats proactively.

**Tools for Practical Implementation**

* **Windows Defender Firewall:**
  + Built-in firewall in Windows OS.
  + Allows setting inbound/outbound rules.
* **Wireshark:**
  + A network protocol analyzer used to capture and inspect data packets.
  + Useful to detect suspicious traffic or monitor unsecured data transfers.

**2. Implement Basic Security Measures**

**Step 1: Set Up a Simple Network Environment**

**Option A: Home Network**

* Use your existing Wi-Fi router.
* Connect at least two devices (e.g., laptop and phone).

**Option B: Virtual Lab (Optional for Practice)**

* Use a virtualization platform like VirtualBox or VMware.
* Create a virtual router (e.g., pfSense) and add two VMs as clients (Windows/Linux).

**Step 2: Enable and Configure Windows Defender Firewall**

* **Enable Firewall:**

1. Go to **Control Panel > System and Security > Windows Defender Firewall**.
2. Ensure it says **"Windows Defender Firewall is on"** for both **Private** and **Public** networks.

* **Block Unauthorized Access:**

1. Click **"Advanced settings"** on the left.
2. Under **Inbound Rules**, click **"New Rule"**.
3. Select **Port** > Choose **TCP** or **UDP** > Enter specific port (e.g., block port 23 for Telnet).
4. Choose **"Block the connection"**, then apply to all profiles.
5. Name the rule (e.g., "Block Telnet") and save.

* **Step 3: Basic Security Configurations**
* **Change Default Router Password:**

1. Access your router: Open a browser and enter your router’s IP (e.g., 192.168.1.1 or 192.168.0.1).
2. Login with the default credentials (often on a label under the router).
3. Go to **Administration or Security Settings**.
4. Change both:
   * **Admin username & password**
   * **Wi-Fi network name (SSID)** and **password**

* **Enable Network Encryption (WPA2/WPA3):**

1. In your router settings, navigate to **Wireless > Security**.
2. Select **WPA2-Personal** or **WPA3-Personal** (if supported).
3. Set a strong Wi-Fi password (minimum 12 characters, include letters/numbers/symbols).

* **Additional Best Practices**
* **Turn off WPS (Wi-Fi Protected Setup)** — it’s a known vulnerability.
* **Enable automatic firmware updates** if your router supports it.
* **Disable remote management** unless absolutely necessary.
* **Use MAC address filtering** (optional): restrict network access to specific devices.
* **Optional: Test Security**
* Use **Wireshark** to monitor network traffic and ensure unencrypted traffic is minimized.
* Try a port scanner like **Nmap** (in safe, local-only mode) to verify which ports are open.

**3. Monitor Network Traffic with Wireshark**

**A. Capture and Analyze Network Traffic**

1. **Start a Capture Session**:
   * Click the interface > then the **shark fin icon** (top-left) to begin capturing.
   * Open a few websites, send a ping (ping google.com), or check your email to generate traffic.
2. **Stop Capture**:
   * Click the **red square icon** to stop.
3. **Apply Display Filters**:
   * Filter common protocols to analyze them one by one:
     + **http** – Web traffic
     + **dns** – Domain Name System queries
     + **tcp** – Transmission Control Protocol packets
     + **icmp** – Pings
     + **tls** – Encrypted traffic (HTTPS)

**B. Identify & Understand Common Traffic Types**

| **Protocol** | **Use/Description** | **Filter** |
| --- | --- | --- |
| **HTTP** | Unencrypted web traffic (URLs, headers, sometimes content) | http |
| **HTTPS/TLS** | Encrypted web traffic (you won’t see content, but you’ll see handshakes and domains via SNI) | tls |
| **DNS** | Converts website names into IP addresses | dns |
| **ICMP** | Used for ping, traceroute, error reporting | icmp |
| **ARP** | Resolves MAC addresses on local network | arp |

**Tip**: Right-click a packet > "Follow" > "TCP Stream" to view entire conversations.

**C. Spot Unusual or Suspicious Traffic**

Look out for:

| **Suspicious Pattern** | **What to Watch For** |
| --- | --- |
| Repeated DNS queries | Potential malware communication |
| Connections to unknown IPs | Especially in countries or regions not relevant to you |
| Large volume of outbound traffic | Could suggest data exfiltration |
| Plain HTTP logins or form submissions | Sensitive data not encrypted |
| ARP spoofing | Multiple IPs resolving to one MAC address |
| High ICMP traffic | Possible ping flood/DoS |

**Example of Unusual Behavior:**

* Seeing frequent connections to IPs like 185.225.xxx.xxx not associated with any site you're using.
* DNS requests to suspicious domain names (.ru, .tk, or random letters).

**Optional: Save Your Capture**

* Go to **File > Save As** to export the .pcapng file for analysis or reporting.

**4.Network Security Basics – Findings Report**

**Task 1: Summary of Network Threats**

**1. Virus**

* A program that attaches itself to files and spreads to other files/computers.
* Can corrupt, delete, or encrypt files and slow down systems.

**2. Worm**

* Self-replicates across networks without user action.
* Consumes bandwidth and overloads systems.

**3. Trojan Horse**

* Disguised as legitimate software.
* Grants unauthorized access or installs malware.

**4. Phishing**

* Social engineering attack to trick users into revealing personal information (e.g., passwords, bank details).
* Usually done via email or fake login pages.

**Task 2: Security Measures Implemented**

**1. Network Setup**

* Used a **home Wi-Fi network** with two connected devices (Laptop + Mobile Phone).

**2. Configured Windows Defender Firewall**

* Enabled firewall for both public and private networks.
* Created custom inbound rule to **block Telnet (TCP port 23)**.
* Ensures unused and insecure services are not accessible.

**3. Secured Router Configuration**

* Changed default router admin password.
* Changed SSID and set **WPA2 encryption** for Wi-Fi.
* Disabled WPS and remote management.

**Why These Measures Matter:**

* Prevent unauthorized access.
* Reduce attack surface.
* Encrypt communication to prevent data theft.
* Protect against simple automated attacks and sniffing.

**Task 3: Wireshark Traffic Capture**

**Captured Protocols**

* **HTTP**: Clear text communication with websites.
* **HTTPS/TLS**: Encrypted traffic; only metadata visible.
* **DNS**: Domain name resolution.
* **ICMP**: Ping messages.
* **ARP**: Device identification on local network.

**Screenshots or Description:**

*(Insert screenshots here, or use the descriptions below)*

**Example 1: HTTP Traffic**

* Captured visiting a non-HTTPS site (http://example.com).
* Able to see GET request and host headers in plain text.

**Example 2: DNS Query**

* Saw query to google.com resolving to IP address.
* Verified DNS protocol in use and query types (A, AAAA).

**Example 3: ICMP Ping**

* Sent ping 8.8.8.8 from terminal.
* Captured request/reply with ICMP filter.

**Task 4: How These Measures Help Protect the Network**

| **Measure** | **Benefit** |
| --- | --- |
| **Firewall Rules** | Block unused ports to prevent remote exploitation. |
| **WPA2 Encryption** | Ensures that Wi-Fi traffic is encrypted and not readable by eavesdroppers. |
| **Changed Default Credentials** | Prevents attackers from logging in with known router defaults. |
| **Traffic Monitoring** | Helps identify suspicious activity (e.g., unusual DNS lookups, outbound connections). |
| **Wireshark Analysis** | Allows deep visibility into network behavior to catch vulnerabilities or intrusions early. |

5.**Reflect on Security Best Practices:**

* **Additional Security Measures for Larger Networks**

In larger, more complex networks, it's essential to implement advanced, layered security strategies. These may include Intrusion Detection and Prevention Systems (IDPS) to monitor and block threats in real-time, and Network Access Control (NAC) to enforce policies on which devices can connect. Segmentation using VLANs can isolate sensitive departments, while VPNs ensure secure access for remote workers. Multi-factor authentication (MFA) adds identity verification, and automated patch management ensures systems stay up to date. Finally, regular security audits, employee training, and user behavior analytics provide a proactive defense against evolving threats.

* **Educating Others About Network Security**

To educate others about network security, I would emphasize how digital safety impacts everyday life—from protecting personal banking details to preventing identity theft. I would use real-world examples, like phishing scams and Wi-Fi snooping, to show how easy it is to be targeted. Then, I’d introduce simple habits: using strong passwords, enabling MFA, avoiding public Wi-Fi for sensitive tasks, and staying alert to suspicious emails. The goal would be to make security relatable, practical, and part of everyone's daily routine, regardless of technical background.

**Task 2: Introduction to Web Application Security**

**1. Setup WebGoat**

**A. What Is WebGoat?**

WebGoat is a deliberately insecure application maintained by OWASP to teach common web vulnerabilities and secure coding practices.

**B. System Requirements**

* OS: Windows, Linux, or macOS
* Java 8 or newer (JDK required)
* Internet connection for initial setup

**C. Install WebGoat (Standalone Method - Easiest)**

**1. Download WebGoat JAR File**

* Go to: https://owasp.org/www-project-webgoat/
* Under "Downloads", get the latest .jar file (e.g., webgoat-server-8.2.0.jar)

**2. Install Java (if not already installed)**

* Download and install from: https://www.oracle.com/java/technologies/javase-downloads.html
* Confirm install:

**bash**

**CopyEdit**

**java -version**

**3. Run WebGoat**

**In your terminal or command prompt:**

**bash**

**CopyEdit**

java -jar webgoat-server-8.2.0.jar

It will start a local server.

**4. Access the Application**

* **Open your browser and go to:**[**http://localhost:8080/WebGoat**](http://localhost:8080/WebGoat)

**D. Using WebGoat**

* Default credentials (if needed):  
  Username: guest  
  Password: guest
* **Once logged in, you’ll see lessons organized by vulnerability type:**
* SQL Injection
* Cross-Site Scripting (XSS)
* Insecure Deserialization
* Broken Authentication, and more.

**Alternative: Docker Installation (Optional)**

**If you prefer Docker:**

bash

CopyEdit

docker pull webgoat/webgoat

docker run -d -p 8080:8080 webgoat/webgoat

Then access via:  
<http://localhost:8080/WebGoat>

**2. Perform Basic Vulnerability Analysis**

**A. Set Up OWASP ZAP**

**1. Download & Install**

* Get ZAP from: https://www.zaproxy.org/download/
* Install and launch it (choose "Manual Explore" if prompted).

**2. Configure Your Browser to Use ZAP Proxy**

* ZAP runs on localhost:8080 (by default).
* Set your browser's proxy settings to:
  + **HTTP Proxy**: 127.0.0.1
  + **Port**: 8080

**3. Start Capturing**

* Open WebGoat in the same browser (http://localhost:8080/WebGoat)
* ZAP will start recording your interactions with WebGoat.

**B. Scan WebGoat with ZAP**

**1. Spidering / Crawling**

* In ZAP: right-click on the site tree for WebGoat → **Attack > Spider**.
* This discovers all pages and forms.

**2. Active Scan**

* After spidering, right-click the site again → **Attack > Active Scan**.
* Select the target (WebGoat site), and click **Start Scan**.

**C. Identify Vulnerabilities**

**1. SQL Injection**

* **Where to Find It in WebGoat**:
  + *Injection Flaws > SQL Injection (advanced)*
  + Try submitting ' OR '1'='1 in form fields.
* **ZAP Detection**:
  + Look for alerts like:
    - *"SQL Injection"*
    - *"SQL Error Messages in Response"*
  + ZAP may show error-based SQLi or boolean-based test results.

**2. Cross-Site Scripting (XSS)**

* **Where to Find It in WebGoat**:
  + *Cross-Site Scripting > Stored or Reflected XSS*
  + Enter script payloads like <script>alert(1)</script>.
* **ZAP Detection**:
  + ZAP tries payloads in inputs and looks for JavaScript execution in responses.
  + Alerts may be labeled:
    - *"Reflected XSS"*
    - *"Persistent XSS"*

**3. Cross-Site Request Forgery (CSRF)**

* **Where to Find It in WebGoat**:
  + *CSRF > CSRF Tokens or General CSRF Lab*
* **ZAP Detection**:
  + ZAP alerts may say:
    - *"CSRF Vulnerability"*
    - *"Absence of Anti-CSRF Token"*
  + ZAP inspects forms and POST requests for missing CSRF protections.

**Optional Notes to Document:**

| **Vulnerability** | **Location in WebGoat** | **ZAP Alert/Message** | **Screenshot (if needed)** |
| --- | --- | --- | --- |
| SQL Injection | Injection Flaws section | "SQL Injection Detected" | (Insert screenshot) |
| XSS | XSS Labs | "Reflected XSS" | (Insert screenshot) |
| CSRF | CSRF Labs | "CSRF Vulnerability" | (Insert screenshot) |

**Next Steps (Optional)**

* Save the scan report:  
  Go to **Report > Generate HTML Report** and export your findings.
* Consider analyzing HTTP requests in ZAP’s Request/Response tab to understand how payloads are injected.

**3. Explore Vulnerabilities**

**A. Understand How Each Vulnerability Works**

When you run scans with **OWASP ZAP**, you can:

* **Click on an alert** in the **"Alerts" tab**
* Read:
  + A **description** of the vulnerability
  + **Risk level**
  + **Evidence** (where ZAP found it)
  + **Recommendations** for mitigation

These are based on OWASP's top web security threats.

**B. Manually Exploit Each Vulnerability**

Here’s how you can try **basic manual techniques** using WebGoat’s labs (designed for safe, hands-on learning).

**1. SQL Injection**

**Where:**

WebGoat > *Injection Flaws > SQL Injection (advanced)*

**Exploit Example:**

1. In the login form, input:
   * **Username**: ' OR '1'='1
   * **Password**: (anything)
2. Submit the form.
3. If successful, you’re logged in without valid credentials.

**What’s Happening:**

* The input breaks the SQL query logic:

sql

CopyEdit

SELECT \* FROM users WHERE username = '' OR '1'='1';

* '1'='1' always evaluates to true, bypassing authentication.

**2. Cross-Site Scripting (XSS)**

**Where:**

WebGoat > *Cross-Site Scripting > Stored XSS or Reflected XSS*

**Exploit Example:**

1. Find a comment or input field.
2. Input:

html

CopyEdit

<script>alert('XSS')</script>

1. Submit and reload or view the response.
2. A JavaScript alert should pop up.

**What’s Happening:**

* Your input is rendered directly into the page without sanitization.
* The browser runs your script—this simulates a malicious payload.

**3. Cross-Site Request Forgery (CSRF)**

**Where:**

WebGoat > *CSRF > General CSRF Lab*

**Exploit Example:**

1. The lab shows a form that performs an action (e.g., changes user data).
2. Recreate that form in your own HTML file **without a CSRF token**:

html

CopyEdit

<form method="POST" action="http://localhost:8080/WebGoat/CSRF/example">

<input type="hidden" name="email" value="hacker@example.com">

<input type="submit" value="Exploit!">

</form>

1. Load this file in a browser (same session) and click **Submit**.
2. If CSRF protection is missing, the request goes through.

**What’s Happening:**

* The browser auto-sends session cookies.
* Without CSRF tokens, the server can’t distinguish between a real user action and an attack.

**Summary Table**

| **Vulnerability** | **Manual Technique** | **Outcome** |
| --- | --- | --- |
| **SQL Injection** | ' OR '1'='1 in login | Logged in without valid credentials |
| **XSS** | <script>alert()</script> | Alert box displayed |
| **CSRF** | Custom HTML form w/ no token | Action executes via unauthorized request |
| **4. Report:**  **1. SQL Injection**  **Location**  **WebGoat > *Injection Flaws > SQL Injection (advanced)***  **How It Was Discovered**   * Performed a manual scan with OWASP ZAP * ZAP flagged a SQL Injection alert on the login form * Verified manually by injecting ' OR '1'='1 into the username field   **Exploitation Process**   * **Input:**   + Username: ' OR '1'='1   + Password: any value * **Result: Bypassed login and accessed restricted area**   **Why It’s Dangerous**   * Attackers can bypass authentication, access sensitive data, or modify the database * A common entry point for data breaches   **Screenshot**  *(Replace with actual screenshot)*  **Mitigation Steps**   * Use parameterized queries (prepared statements) instead of string concatenation * Implement input validation/sanitization * Apply least privilege on database accounts   **2. Cross-Site Scripting (XSS)**  **Location**  **WebGoat > *Cross-Site Scripting > Stored XSS***  **How It Was Discovered**   * ZAP flagged Reflected XSS on input forms * Manually injected <script>alert('XSS')</script> into the comment box   **Exploitation Process**   * Input script into a comment field * Reloaded the page and observed a JavaScript alert pop up   **Why It’s Dangerous**   * Attackers can run arbitrary JavaScript in users' browsers * Leads to session hijacking, credential theft, defacement, etc.   **🖼️ Screenshot**  (Replace with actual screenshot)  **🛡️ Mitigation Steps**   * Encode all output (e.g., use HTML entities) * Use Content Security Policy (CSP) * Validate and sanitize all user input   **3. Cross-Site Request Forgery (CSRF)**  **Location**  **WebGoat > *CSRF > General CSRF Lab***  **How It Was Discovered**   * ZAP showed absence of anti-CSRF tokens in form submissions * Recreated a POST request in a custom HTML page to simulate the exploit   **Exploitation Process**   * Created an HTML form that submitted a POST request to WebGoat while the user was logged in * The action executed without any user confirmation or CSRF protection   **Why It’s Dangerous**   * Allows attackers to perform actions on behalf of an authenticated user * Can lead to data modification, transfers, or account manipulation   **🖼️ Screenshot**  *(Replace with actual screenshot)*  **🛡️ Mitigation Steps**   * Use anti-CSRF tokens (unique, per-request) * Require user interaction (e.g., re-enter password or CAPTCHA) * Implement SameSite cookies with appropriate flags (SameSite=Strict)   **✅ Conclusion**  These vulnerabilities are among the most critical listed in the OWASP Top 10. Through tools like OWASP ZAP and intentional targets like WebGoat, developers and testers can learn to:   * Recognize dangerous patterns * Understand how exploits work * Apply secure coding practices |  |  |