Cybersecurity Internship Report

Intern: Mudapaka Sailaxman

Internship Organization: The Red Users

Duration: 1 month

Task 1: Introduction to Network Security Basics

Objective:

Understand the fundamentals of network security, identify different network threats, and implement basic security measures.

Skills Utilized:

- Basic Network Security
- Threat Identification
- Security Best Practices

Tools Used:

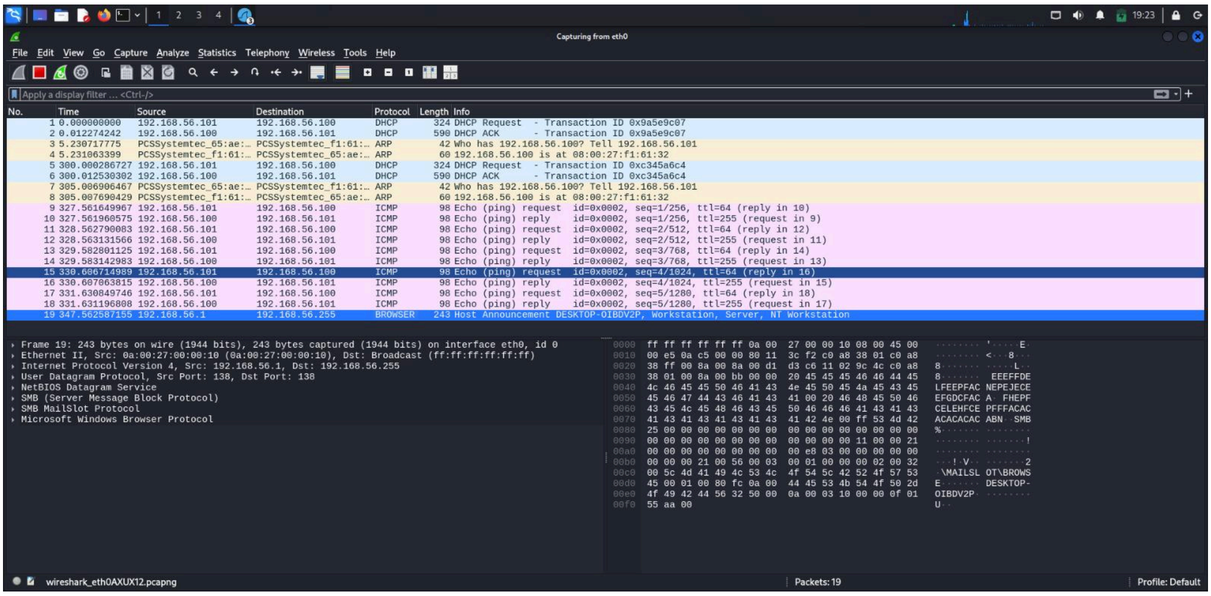
- Windows Defender Firewall
- Wireshark

Work Description:

1. **Network Security Concepts:**
    - Researched network threats such as **viruses, worms, trojans,** and **phishing attacks.**
    - Understood security concepts including **firewalls, encryption,** and **secure network configurations.**
2. **Implementation of Basic Security Measures:**
    - Set up a simple network environment (home network/virtual lab).
    - Configured Windows Defender Firewall to block unauthorized access.
    - Changed default passwords and enabled network encryption using **WPA2/WPA3.**
3. **Network Traffic Monitoring:**
    - Used **Wireshark** to capture and analyze network traffic.
    - Identified traffic types such as **HTTP, DNS, TCP,** and **UDP.**
    - Detected unusual/suspicious traffic that could indicate potential threats.

4. **Findings and Documentation:**

    - Summarized key network threats and their characteristics.
    - Documented implemented security measures with detailed descriptions.
    - Included screenshots from Wireshark showcasing captured traffic patterns.


Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Packets: 19
Profile: Default
19:23

```
No.   Time          Source              Destination         Protocol Length Info
    1 0.000000000   192.168.56.101      192.168.56.100      DHCP      324 DHCP Request  - Transaction ID 0x9a5e9c07
    2 0.012274242   192.168.56.100      192.168.56.101      DHCP      590 DHCP ACK      - Transaction ID 0x9a5e9c07
    3 5.230717775   PCSSystemtec_65:ae:… PCSSystemtec_f1:61:… ARP      42 Who has 192.168.56.100? Tell 192.168.56.101
    4 5.231063399   PCSSystemtec_f1:61:… PCSSystemtec_65:ae:… ARP      60 192.168.56.100 is at 08:00:27:f1:61:32
    5 300.000286727 192.168.56.101      192.168.56.100      DHCP      324 DHCP Request  - Transaction ID 0xc345a6c4
    6 300.012530302 192.168.56.100      192.168.56.101      DHCP      590 DHCP ACK      - Transaction ID 0xc345a6c4
    7 305.006906467 PCSSystemtec_65:ae:… PCSSystemtec_f1:61:… ARP      42 Who has 192.168.56.100? Tell 192.168.56.101
    8 305.007690429 PCSSystemtec_f1:61:… PCSSystemtec_65:ae:… ARP      60 192.168.56.100 is at 08:00:27:f1:61:32
    9 327.561649967 192.168.56.101      192.168.56.100      ICMP       98 Echo (ping) request  id=0x0002, seq=1/256, ttl=64 (reply in 10)
   10 327.561960575 192.168.56.100      192.168.56.101      ICMP       98 Echo (ping) reply    id=0x0002, seq=1/256, ttl=255 (request in 9)
   11 328.562790083 192.168.56.101      192.168.56.100      ICMP       98 Echo (ping) request  id=0x0002, seq=2/512, ttl=64 (reply in 12)
   12 328.563131566 192.168.56.100      192.168.56.101      ICMP       98 Echo (ping) reply    id=0x0002, seq=2/512, ttl=255 (request in 11)
   13 329.582801125 192.168.56.101      192.168.56.100      ICMP       98 Echo (ping) request  id=0x0002, seq=3/768, ttl=64 (reply in 14)
   14 329.583142983 192.168.56.100      192.168.56.101      ICMP       98 Echo (ping) reply    id=0x0002, seq=3/768, ttl=255 (request in 13)
   15 330.606714989 192.168.56.101      192.168.56.100      ICMP       98 Echo (ping) request  id=0x0002, seq=4/1024, ttl=64 (reply in 16)
   16 330.607063815 192.168.56.100      192.168.56.101      ICMP       98 Echo (ping) reply    id=0x0002, seq=4/1024, ttl=255 (request in 15)
   17 331.630849746 192.168.56.101      192.168.56.100      ICMP       98 Echo (ping) request  id=0x0002, seq=5/1280, ttl=64 (reply in 18)
   18 331.631196808 192.168.56.100      192.168.56.101      ICMP       98 Echo (ping) reply    id=0x0002, seq=5/1280, ttl=255 (request in 17)
   19 347.562587155 192.168.56.1        192.168.56.255      BROWSER   243 Host Announcement DESKTOP-OIBDV2P, Workstation, Server, NT Workstation
```

```
› Frame 19: 243 bytes on wire (1944 bits), 243 bytes captured (1944 bits) on interface eth0, id 0
› Ethernet II, Src: 0a:00:27:00:00:10 (0a:00:27:00:00:10), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
› Internet Protocol Version 4, Src: 192.168.56.1, Dst: 192.168.56.255
› User Datagram Protocol, Src Port: 138, Dst Port: 138
› NetBIOS Datagram Service
› SMB (Server Message Block Protocol)
› SMB MailSlot Protocol
› Microsoft Windows Browser Protocol
```

```
0000  ff ff ff ff ff ff 0a 00  27 00 00 10 08 00 45 00   ········ '·····E·
0010  00 e5 0a c5 00 00 80 11  3c f2 c0 a8 38 01 c0 a8   ········ <···8···
0020  38 ff 00 8a 00 8a 00 d1  d3 c6 11 02 9c 4c c0 a8   8······· ·····L··
0030  38 01 00 8a 00 bb 00 00  20 45 45 45 46 46 44 45   8·······  EEEFFDE
0040  4c 46 45 45 50 46 41 43  4e 45 50 45 4a 45 43 45   LFEEPFAC NEPEJECE
0050  45 46 47 44 43 46 41 43  41 00 20 46 48 45 50 46   EFGDCFAC A· FHEPF
0060  43 45 4c 45 48 46 43 45  50 46 46 46 41 43 41 43   CELEHFCE PFFFACAC
0070  41 43 41 43 41 43 41 43  41 42 4e 00 ff 53 4d 42   ACACACAC ABN··SMB
0080  25 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   %······· ········
0090  00 00 00 00 00 00 00 00  00 00 00 00 11 00 00 21   ········ ·······!
00a0  00 00 00 00 00 00 00 00  00 e8 03 00 00 00 00 00   ········ ········
00b0  00 00 00 21 00 56 00 03  00 01 00 00 00 02 00 32   ···!·V·· ·······2
00c0  00 5c 4d 41 49 4c 53 4c  4f 54 5c 42 52 4f 57 53   ·\MAILSL OT\BROWS
00d0  45 00 01 00 80 fc 0a 00  44 45 53 4b 54 4f 50 2d   E······· DESKTOP-
00e0  4f 49 42 44 56 32 50 00  0a 00 03 10 00 00 0f 01   OIBDV2P· ········
00f0  55 aa 00                                           U··
```

wireshark_eth0AXUX12.pcapng

    - Discussed how basic security measures improve overall network safety.

5. **Reflection on Best Practices:**

    - Suggested additional security measures for larger networks, such as **advanced intrusion detection systems (IDS), VPNs,** and **multi-factor authentication (MFA).**
    - Wrote a brief educational note on the importance of network security in daily life.

Task 2: Introduction to Web Application Security

Objective:

Analyze common web application vulnerabilities and understand how attackers exploit these weaknesses.

Skills Utilized:

- Basic Web Security
- Vulnerability Identification

Tools Used:

- **OWASP ZAP**
- **WebGoat** (Intentionally Vulnerable Web Application)

Work Description:

1. **Setup:**
   - Installed and configured WebGoat locally.
   - Explored the application’s structure to understand its functionalities.
2. **Basic Vulnerability Analysis:**
   - Performed vulnerability scans using **OWASP ZAP.**
   - Identified vulnerabilities such as:

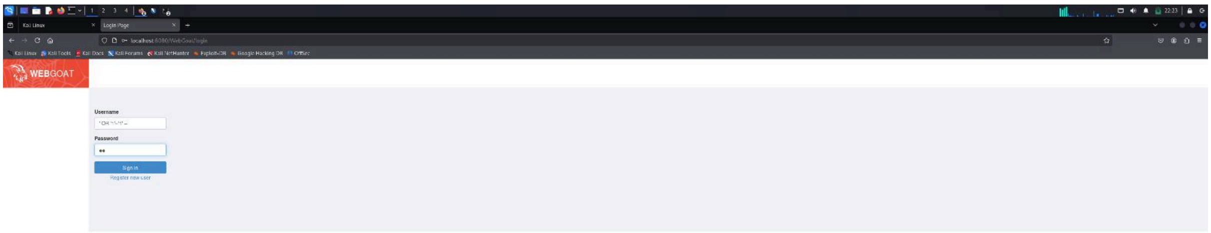
     SQL Injection

     Cross-Site Scripting (XSS)

     Cross-Site Request Forgery (CSRF)

3. **Exploration of Vulnerabilities:**
   - Understood each vulnerability's working mechanism using OWASP ZAP descriptions.
   - Manually exploited vulnerabilities:

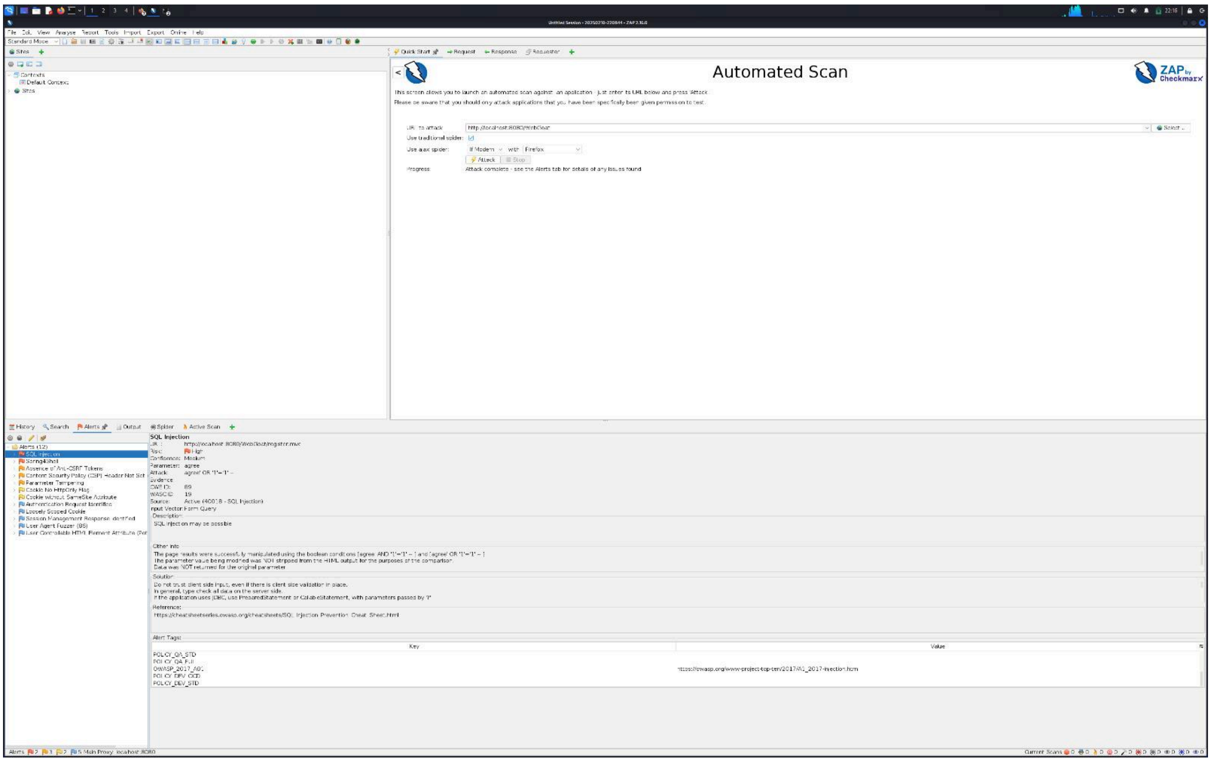
Inserted SQL code into login forms to test for **SQL Injection.**



localhost:8080/WebGoat/login

Username: ' OR '1'='1' --

Password: ●●



Untitled Session - 20250210-220844 - ZAP 2.16.0

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://localhost:8080/WebGoat
Use traditional spider: ☑
Use ajax spider: If Modern with Firefox
Attack Stop
Progress: Attack complete - see the Alerts tab for details of any issues found

Alerts (12)
- SQL Injection
- Spring4Shell
- Absence of Anti-CSRF Tokens
- Content Security Policy (CSP) Header Not Set
- Parameter Tampering
- Cookie No HttpOnly Flag
- Cookie without SameSite Attribute
- Authentication Request Identified
- Loosely Scoped Cookie
- Session Management Response Identified
- User Agent Fuzzer (86)
- User Controllable HTML Element Attribute (Pot

SQL Injection
URL: http://localhost:8080/WebGoat/register.mvc
Risk: High
Confidence: Medium
Parameter: agree
Attack: agree' OR '1'='1' --
Evidence:
CWE ID: 89
WASC ID: 19
Source: Active (40018 - SQL Injection)
Input Vector: Form Query
Description:
SQL injection may be possible

Other Info:
The page results were successfully manipulated using the boolean conditions [agree' AND '1'='1' --] and [agree' OR '1'='1' --]
The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison.
Data was NOT returned for the original parameter.

Solution:
Do not trust client side input, even if there is client side validation in place.
In general, type check all data on the server side.
If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?'

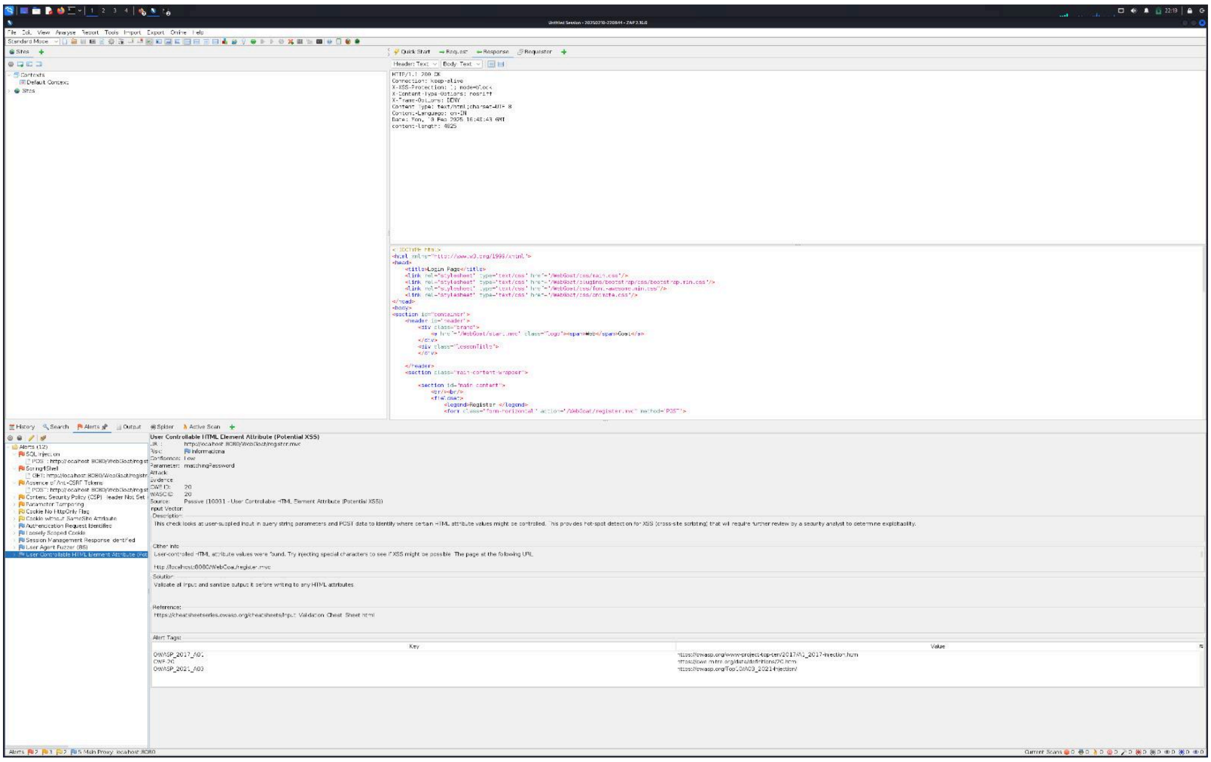
Reference:
https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

Alert Tags:

Key	Value
POLICY_QA_STD	
POLICY_QA_FULL	
OWASP_2017_A01	https://owasp.org/www-project-top-ten/2017/A1_2017-Injection.html
POLICY_DEV_CICD	
POLICY_DEV_STD	

Alerts 2 3 2 5 Main Proxy: localhost:8080
Current Scans 0 0 0 0 0 0 0 0 0 0

Injected malicious scripts to demonstrate **XSS.**



Untitled Session - 20250210-220844 - ZAP 2.16.0

```
HTTP/1.1 200 OK
Connection: keep-alive
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: text/html;charset=UTF-8
Content-Language: en-IN
Date: Mon, 10 Feb 2025 16:48:43 GMT
content-length: 4825
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Login Page</title>
    <link rel="stylesheet" type="text/css" href="/WebGoat/css/main.css"/>
    <link rel="stylesheet" type="text/css" href="/WebGoat/plugins/bootstrap/css/bootstrap.min.css"/>
    <link rel="stylesheet" type="text/css" href="/WebGoat/css/font-awesome.min.css"/>
    <link rel="stylesheet" type="text/css" href="/WebGoat/css/animate.css"/>
</head>
<body>
<section id="container">
    <header id="header">
        <div class="brand">
            <a href="/WebGoat/start.mvc" class="logo"><span>Web</span>Goat</a>
        </div>
        <div class="lessonTitle">
        </div>

    </header>
    <section class="main-content-wrapper">

        <section id="main-content">
            <br/><br/>
            <fieldset>
                <legend>Register </legend>
                <form class="form-horizontal" action="/WebGoat/register.mvc" method="POST">
```

Alerts (12)
- SQL Injection
  - POST: http://localhost:8080/WebGoat/regist
- Spring4Shell
  - GET: http://localhost:8080/WebGoat/registr
- Absence of Anti-CSRF Tokens
  - POST: http://localhost:8080/WebGoat/regist
- Content Security Policy (CSP) Header Not Set
- Parameter Tampering
- Cookie No HttpOnly Flag
- Cookie without SameSite Attribute
- Authentication Request Identified
- Loosely Scoped Cookie
- Session Management Response Identified
- User Agent Fuzzer (86)
- User Controllable HTML Element Attribute (Pot

User Controllable HTML Element Attribute (Potential XSS)
URL: http://localhost:8080/WebGoat/register.mvc
Risk: Informational
Confidence: Low
Parameter: matchingPassword
Attack:
Evidence:
CWE ID: 20
WASC ID: 20
Source: Passive (10031 - User Controllable HTML Element Attribute (Potential XSS))
Input Vector:
Description:
This check looks at user-supplied input in query string parameters and POST data to identify where certain HTML attribute values might be controlled. This provides hot-spot detection for XSS (cross-site scripting) that will require further review by a security analyst to determine exploitability.

Other Info:
User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL:

http://localhost:8080/WebGoat/register.mvc

Solution:
Validate all input and sanitize output it before writing to any HTML attributes.

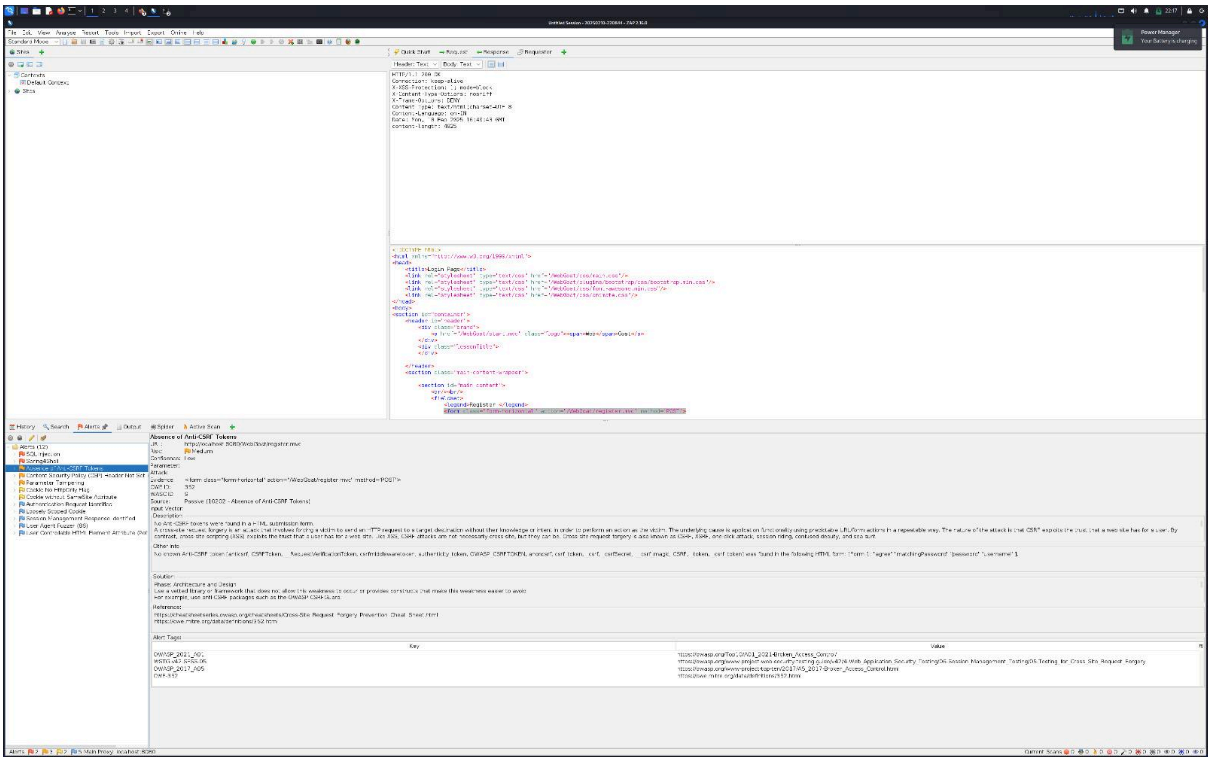
Reference:
https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

Alert Tags:

Key	Value
OWASP_2017_A01	https://owasp.org/www-project-top-ten/2017/A1_2017-Injection.html
CWE-20	https://cwe.mitre.org/data/definitions/20.html
OWASP_2021_A03	https://owasp.org/Top10/A03_2021-Injection/

Alerts 2 3 2 5 Main Proxy: localhost:8080
Current Scans 0 0 0 0 0 0 0 0 0 0

Exploited **CSRF** vulnerabilities to understand session manipulation.



```
HTTP/1.1 200 OK
Connection: keep-alive
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: text/html;charset=UTF-8
Content-Language: en-IN
Date: Mon, 10 Feb 2025 16:48:43 GMT
content-length: 4825
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Login Page</title>
    <link rel="stylesheet" type="text/css" href="/WebGoat/css/main.css"/>
    <link rel="stylesheet" type="text/css" href="/WebGoat/plugins/bootstrap/css/bootstrap.min.css"/>
    <link rel="stylesheet" type="text/css" href="/WebGoat/css/font-awesome.min.css"/>
    <link rel="stylesheet" type="text/css" href="/WebGoat/css/animate.css"/>
</head>
<body>
<section id="container">
    <header id="header">
        <div class="brand">
            <a href="/WebGoat/start.mvc" class="logo"><span>Web</span>Goat</a>
        </div>
        <div class="lessonTitle">
        </div>

    </header>
    <section class="main-content-wrapper">

        <section id="main-content">
            <br/><br/>
            <fieldset>
                <legend>Register </legend>
                <form class="form-horizontal" action="/WebGoat/register.mvc" method="POST">
```

Alerts (12)
- SQL Injection
- Spring4Shell
- Absence of Anti-CSRF Tokens
- Content Security Policy (CSP) Header Not Set
- Parameter Tampering
- Cookie No HttpOnly Flag
- Cookie without SameSite Attribute
- Authentication Request Identified
- Loosely Scoped Cookie
- Session Management Response Identified
- User Agent Fuzzer (96)
- User Controllable HTML Element Attribute (Pot

Absence of Anti-CSRF Tokens

URL: http://localhost:8080/WebGoat/register.mvc
Risk: Medium
Confidence: Low
Parameter:
Attack:
Evidence: <form class="form-horizontal" action="/WebGoat/register.mvc" method="POST">
CWE ID: 352
WASC ID: 9
Source: Passive (10202 - Absence of Anti-CSRF Tokens)
Input Vector:

Description:
No Anti-CSRF tokens were found in a HTML submission form.
A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.

Other Info:
No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token] was found in the following HTML form: [Form 1: "agree" "matchingPassword" "password" "username"].

Solution:
Phase: Architecture and Design
Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.
For example, use anti-CSRF packages such as the OWASP CSRFGuard.

Reference:
https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
https://cwe.mitre.org/data/definitions/352.html

Alert Tags:

Key	Value
OWASP_2021_A01	https://owasp.org/Top10/A01_2021-Broken_Access_Control/
WSTG-v42-SESS-05	https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/06-Session_Management_Testing/05-Testing_for_Cross_Site_Request_Forgery
OWASP_2017_A05	https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control.html
CWE-352	https://cwe.mitre.org/data/definitions/352.html

Main Proxy: localhost:8080
Current Scans

Findings and Documentation:

- Documented the vulnerabilities found, detailing the discovery and exploitation process.
- Included relevant screenshots and technical explanations.
- Suggested mitigation strategies such as **input validation, secure coding practices,** and **token-based authentication** to prevent similar vulnerabilities.