Bangladesh University of Professionals (BUP)

Assignment on LAB TEST

Topic

Application Security Assessment & Suggestion for Remediation

Submitted To

Muhammad Abul Kalam Azad

Submitted By

Jannatul Ferdous Katha

Masters in Cyber Security

Department of Computer Science and Engineering

Submission Date: 11-05-2025

Contents

💡Part 1: Theoretical Knowledge

⁉️Question - 1. List and briefly describe at least five common types of application attack.

1. SQL Injection (SQLi)

Description:
An attacker manipulates input fields to inject malicious SQL queries into the application's database.

Example:
Entering ' OR '1'='1 in a login form to bypass authentication.

2. Cross-Site Scripting (XSS)

Description:
Malicious scripts are injected into trusted websites, which then execute in the browser of other users.

Example:
Submitting <script>alert('XSS');</script> in a comment section to pop up an alert box when others view it.

3. Cross-Site Request Forgery (CSRF)

Description:
An attacker tricks a logged-in user into unknowingly submitting a request to perform actions on their behalf.

Example:
Sending a hidden form that triggers a money transfer when a user is logged into their bank account.

4. Command Injection

Description:
An attacker injects system commands through an input field, which the server executes unintentionally.

Example:
Entering ; rm -rf / in a vulnerable shell-based input to delete server files.

5. Broken Authentication

Description:
Weak authentication mechanisms allow attackers to compromise user accounts or session tokens.

Example:
Using automated tools to brute-force login credentials due to lack of rate-limiting.

⁉️ Question - 2: Explain the differences between white hat, black hat, and grey hat hackers.

👨‍💻 White Hat Hackers (Ethical Hackers)

Definition:
White hat hackers are security professionals who use their skills to find and fix security vulnerabilities. They have authorization and work to protect systems.

Example:
A penetration tester hired by a company to test its web application for vulnerabilities and provide a report on how to fix them.

🕵️ Black Hat Hackers (Malicious Hackers)

Definition:
Black hat hackers are cybercriminals who exploit vulnerabilities for personal gain, such as stealing data, installing malware, or causing damage. Their actions are illegal and unethical.

Example:
A hacker breaches a company's server to steal customer credit card data and sells it on the dark web.

⚖️ Grey Hat Hackers (In-between Hackers)

Definition:
Grey hat hackers operate between ethical and unethical boundaries. They may find and disclose vulnerabilities without permission, but usually without malicious intent.

Example:
A hacker finds a flaw in a government website and informs them without authorization, but doesn't exploit it or demand a reward.

⁉️ Question - 3: Describe the five phases of ethical hacking methodology (Reconnaissance, Scanning, Gaining Access, Maintaining Access, and Covering Tracks).

🔍 1. Reconnaissance (Information Gathering)

Description:
This is the first phase where the ethical hacker collects as much information as possible about the target system or network. This can include domain names, IP addresses, employee details, or system technologies.

Example:
Using tools like WHOIS, Google search, or social engineering to gather data about the organization.

📡 2. Scanning

Description:
In this phase, the hacker identifies live hosts, open ports, services, and vulnerabilities on the target system using scanning tools.

Example:
Using tools like Nmap or Nessus to detect which ports are open and which software versions are running.

🔓 3. Gaining Access

Description:
The hacker attempts to exploit identified vulnerabilities to gain unauthorized access to the system or application.

Example:
Exploiting a weak password or outdated software to log into a system.

🔄 4. Maintaining Access

Description:
After gaining access, the hacker tries to maintain a persistent connection for future use, often by installing backdoors or creating new user accounts.

Example:
Placing a remote access Trojan (RAT) on the system to reconnect later.

🧹 5. Covering Tracks

Description:
The final phase involves erasing any evidence of the attack to avoid detection and maintain system integrity.

Example:
Deleting logs, clearing browser history, or disabling alarms that might alert administrators.

⁉️ Question -4: Discuss the legal and ethical considerations that ethical hackers must follow.

⚖️ Legal and Ethical Considerations for Ethical Hackers

Ethical hackers, also known as white hat hackers, must operate within the boundaries of the law and professional ethics. Their goal is to improve security, not to cause harm. The following are key considerations they must follow:

✅ 1. Proper Authorization

Ethical hackers must always obtain written permission from the organization before performing any security testing. Hacking without consent is illegal—even with good intentions.

Example:
Testing a company's website only after signing a legal agreement that outlines scope, methods, and rules.

✅ 2. Scope Limitation

Ethical hackers must work strictly within the defined scope of the engagement. Testing systems outside of the approved boundaries can lead to legal consequences.

Example:
If authorized to test only the company's public web app, they should not attempt to access internal databases or employee email systems.

✅ 3. Data Privacy and Confidentiality

Hackers must protect sensitive data they encounter during testing and not misuse, copy, or disclose it.

Example:
If they find customer data during testing, it must be reported securely and not shared with unauthorized parties.

✅ 4. Responsible Disclosure

Any vulnerabilities discovered must be reported only to the authorized party and in a responsible manner, giving the organization time to fix the issues.

Example:
Reporting a bug to the company's security team rather than posting it on social media.

✅ 5. Compliance with Laws and Standards

Ethical hackers must comply with local and international cybersecurity laws, as well as standards like GDPR, HIPAA, or PCI-DSS, depending on the industry.

Example:
Not accessing or storing personal medical data in violation of HIPAA regulations.

✅ Conclusion

Ethical hackers play a vital role in securing systems, but they must act with integrity, transparency, and full legal compliance to ensure their work is beneficial, professional, and lawful.

🔧Part 2: Practical Application

🔐 WebGoat Setup and Access:

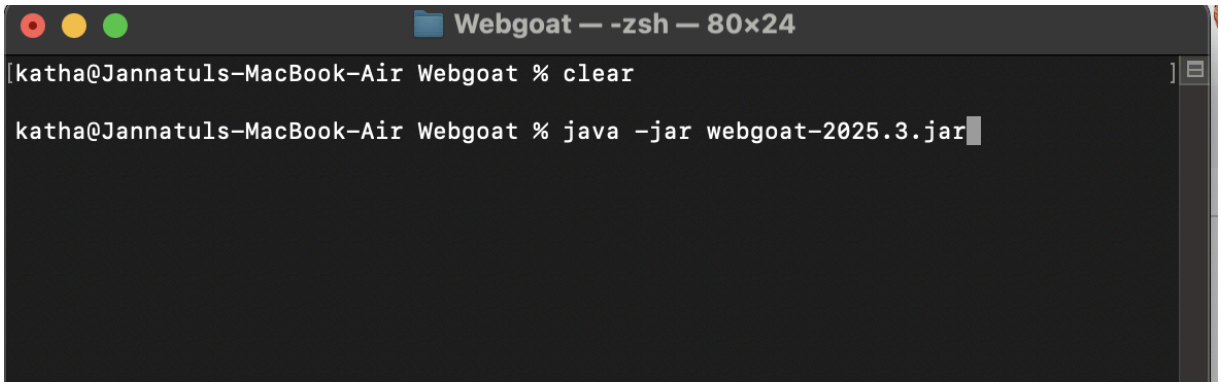
WebGoat is an intentionally insecure web application maintained by **OWASP**, designed to teach web application security lessons by intentionally including common vulnerabilities such as **SQL Injection, XSS, and CSRF.**

To download and set up WebGoat, follow these steps:

1. **Install Java (JDK):**
   Ensure Java 11 or later is installed on your system.

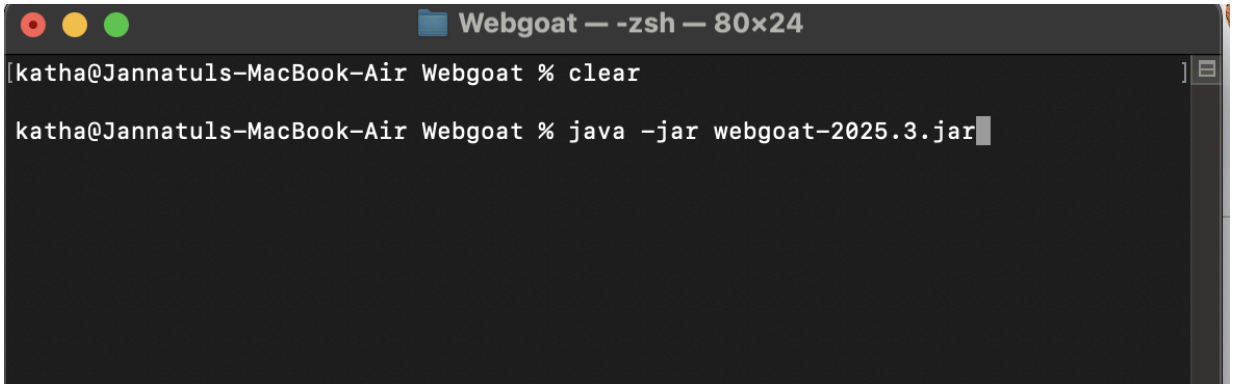
2. **Download WebGoat JAR File:**
   Visit the official OWASP GitHub page or go to https://owasp.org/www-project-webgoat/ and download the latest .jar file.

3. **Run WebGoat:**
   Open a terminal or command prompt and navigate to the folder where the JAR file is located. Then run:



4. **Access WebGoat in Browser:**

   Once running, open your browser and go to: http://localhost:8080/WebGoat



5. Login and Start Learning:

Use default credentials (e.g., guest/guest or register a new user) to log in and access various security lessons.

6. Login to the WebGoat Site:

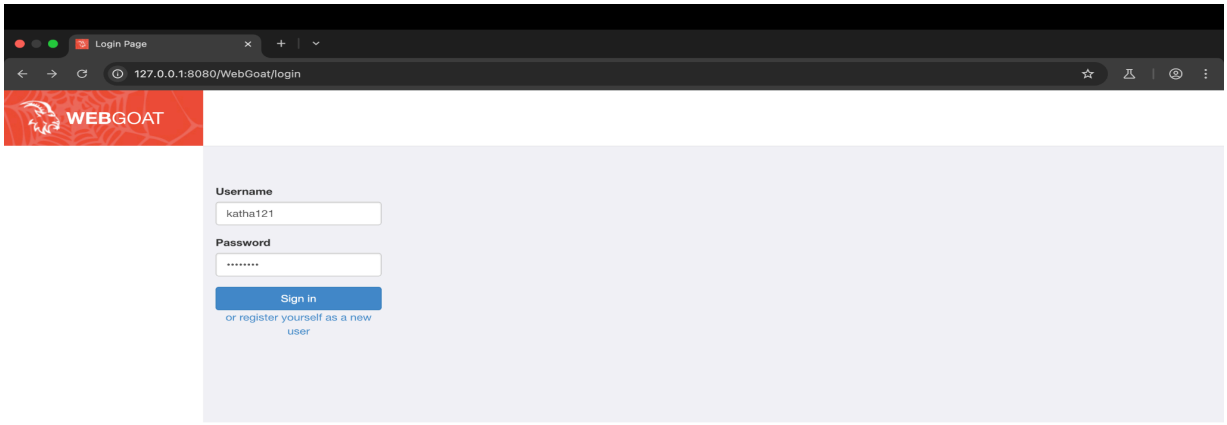


Fig-1: Login to the site

Scan List of Vulnerabilities by using ZAP:

❖ List of Vulnerabilities:

By Automated Scan found a number of vulnerabilities -

- Absence of Anti-CSRF Tokens
- Content Security Policy (CSP) Header Not Set
- Missing Anti -clickjacking header
- Cookie without same site Attribute
- X-content-Type-Options Header Missing and so on.

❖ Attachment of Vulnerabilities found:

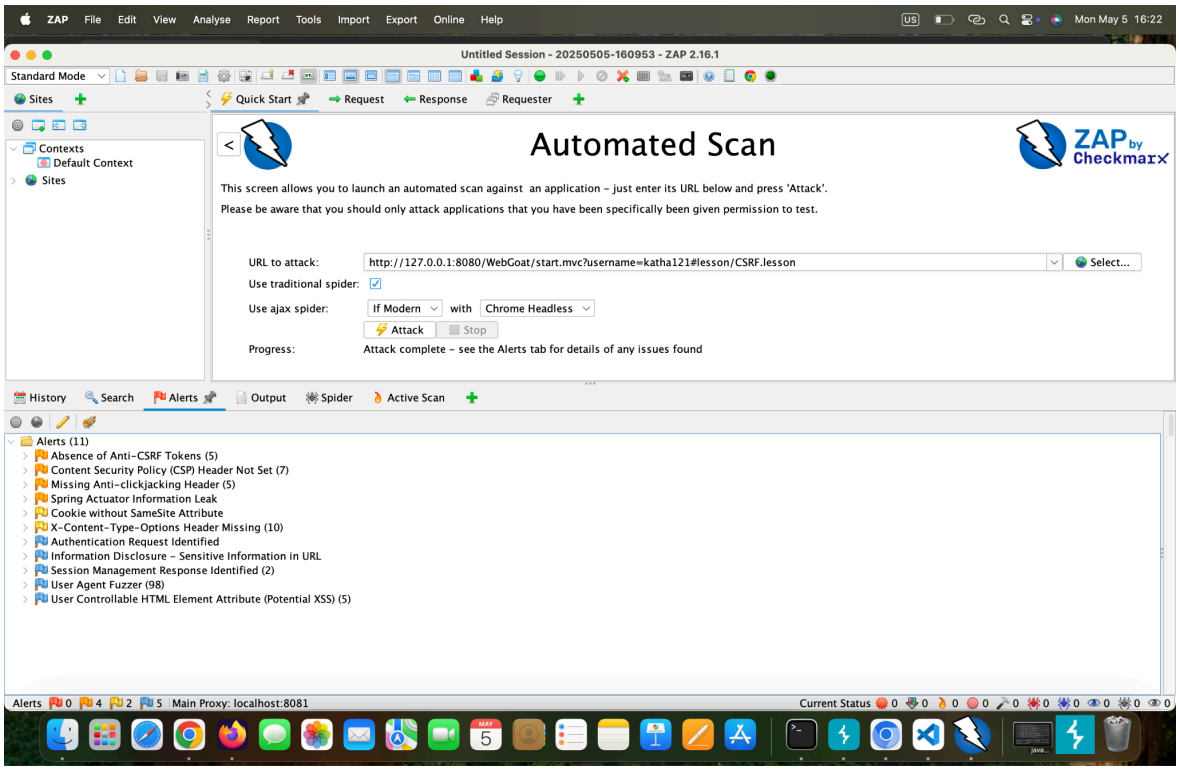


Fig -2 : Vulnerabilities List

This Report contains only SQL injection and Anti-CSRF tokens Scan:

❖ Scenario -1: Perform SQL Injection:

Step -1: Update Name field and hit submit . No specific results matched -

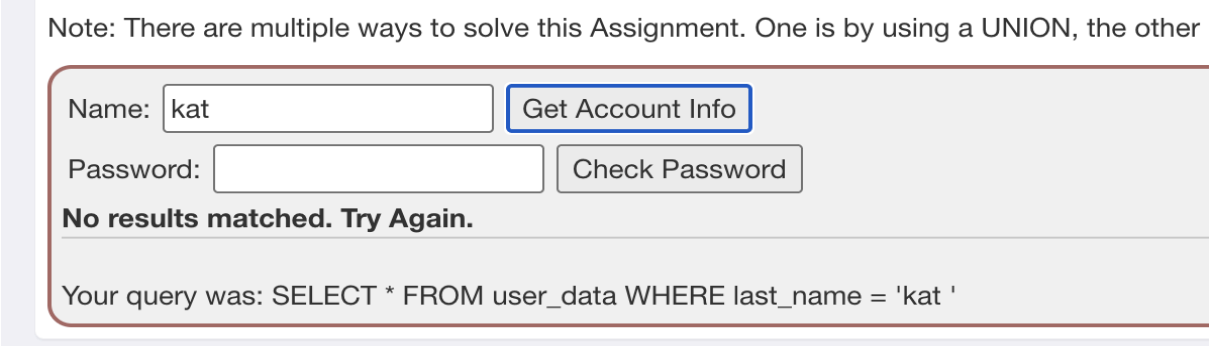


Fig -3: Results of After update Name Field

Step -2: To get data about the mapping table need to use an SQL query and will get a list where all of the information is stored about Database.

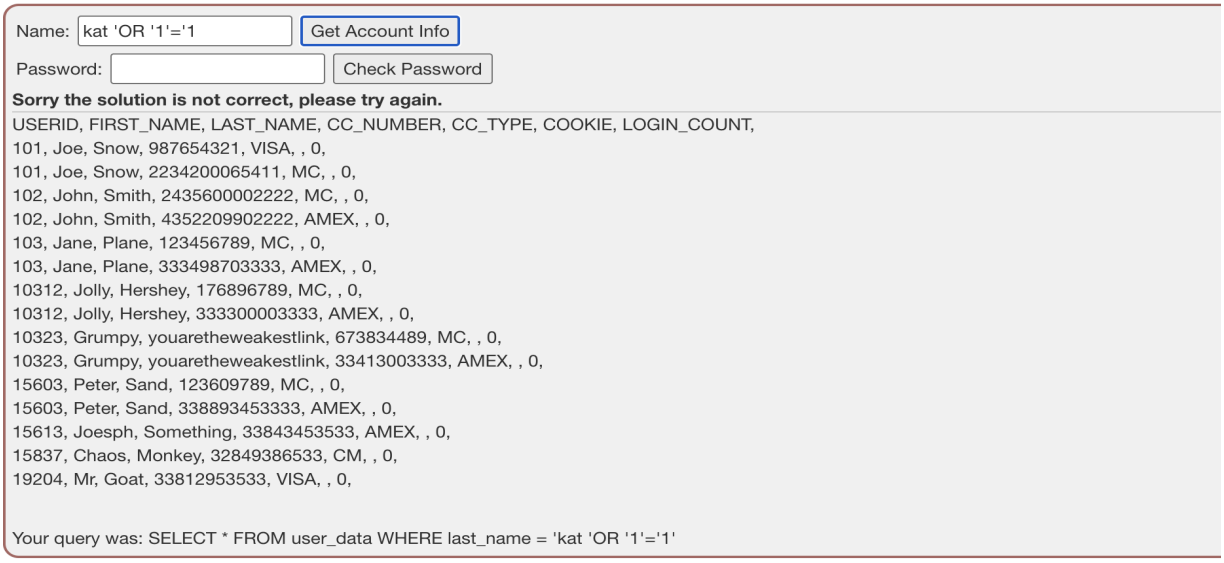


Fig - 4 : Lost of DB information

❖ List of Anti - CSRF Tokens Attachment:

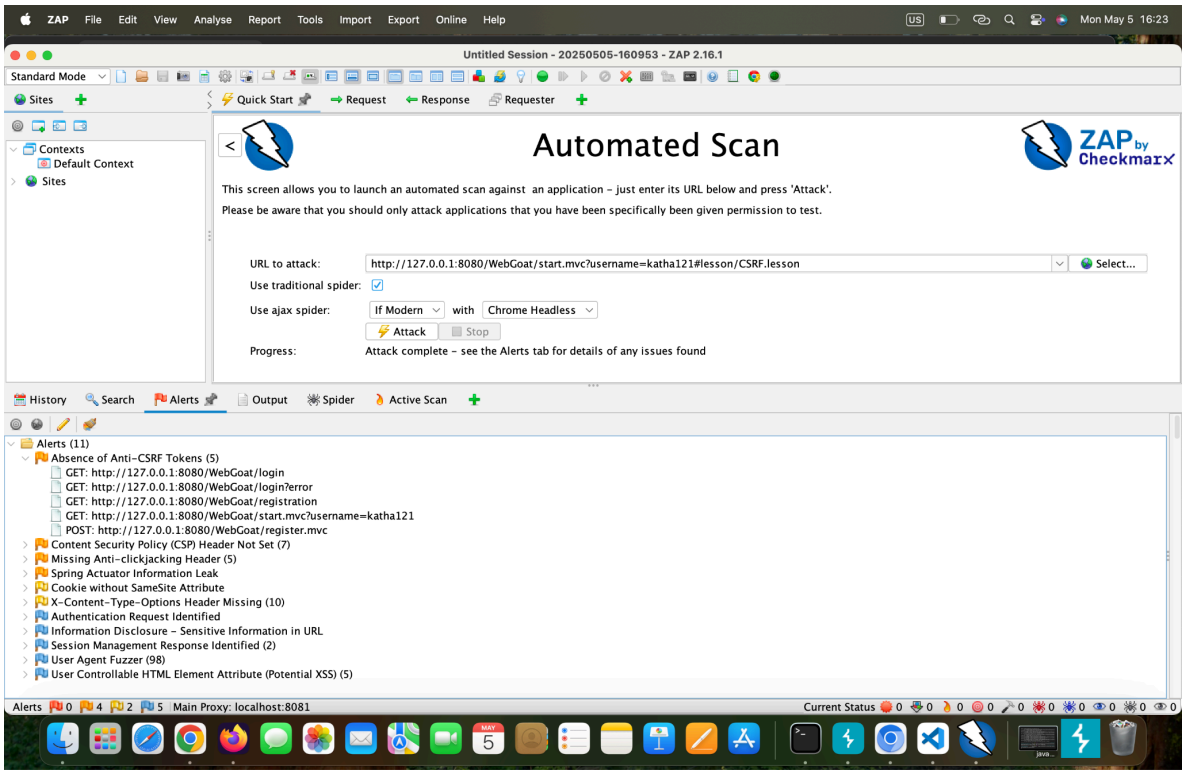


Fig - 5: Absence of Anti CSRF Tokens Lists

❖ Perform CSRF on WebGoat Site: Absence of Anti - CSRF Tokens :

☐ Scenario -1 : Basic Get CSRF Exercise

Step -1: Navigate to the Webgoat site
Step -2: Login to the site
Step -3: Navigate to the Cross - Site Request Forgeries
Step - 4: Go to the Tab - 3 to perform **Basic Get CSRF Exercise**
Step -5: Then Go to the **Burp Suite Tool** and Navigate to the **Proxy Tab**
Step -6: Turn on **Intercept** and Navigate to the URL to test **CSRF**
Step-7: Turn Of **Intercept** and Navigate to the URL and check Responses - http://127.0.0.1:8080/WebGoat/csrf/basic-get-flag
Step-8: According To the Response Create a HTML File For CSRF Token
Step -9: Upload The HTML file into the form and Hit into the browser
Step-10: After Successful insertion demo.html file a success response must be shown which contains Flag Number
Step -11: Then Go to the WebGoat CSRF part and update Confirm Flag field
Step-12: A Confirmation Message will be shown after successful submission of Flag Value

🖇Attachments of every step:

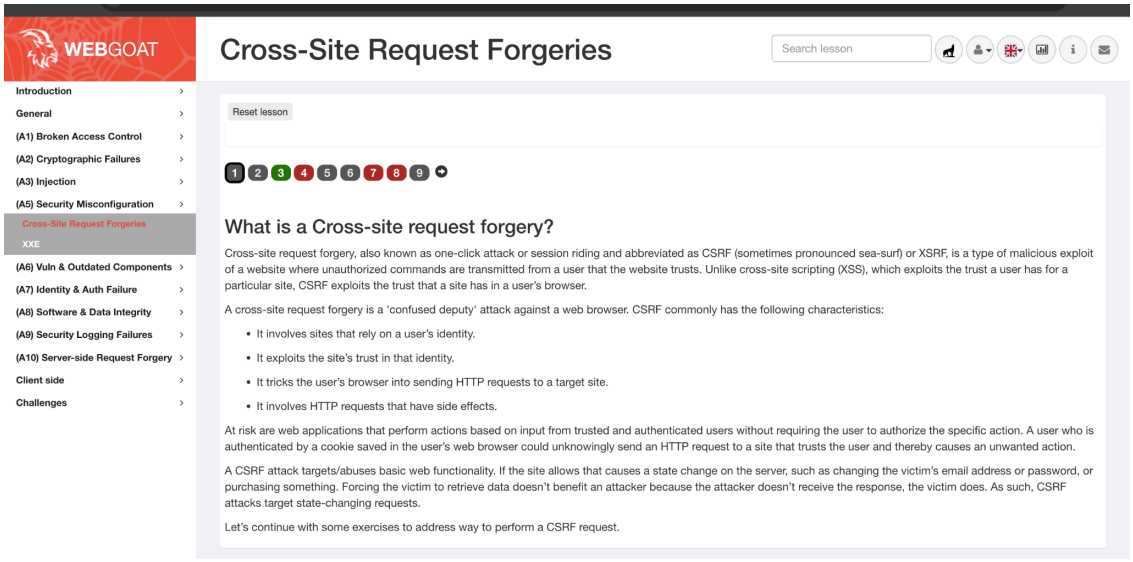


Fig-6: Step -3

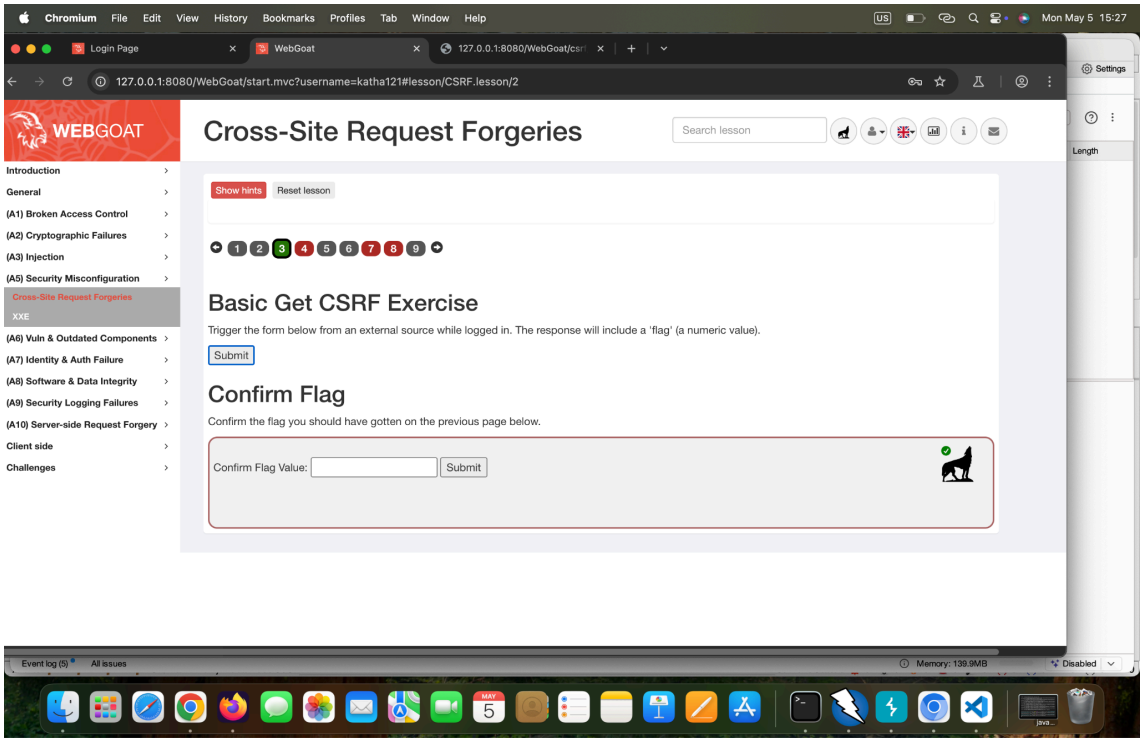


Fig -7: Step 4

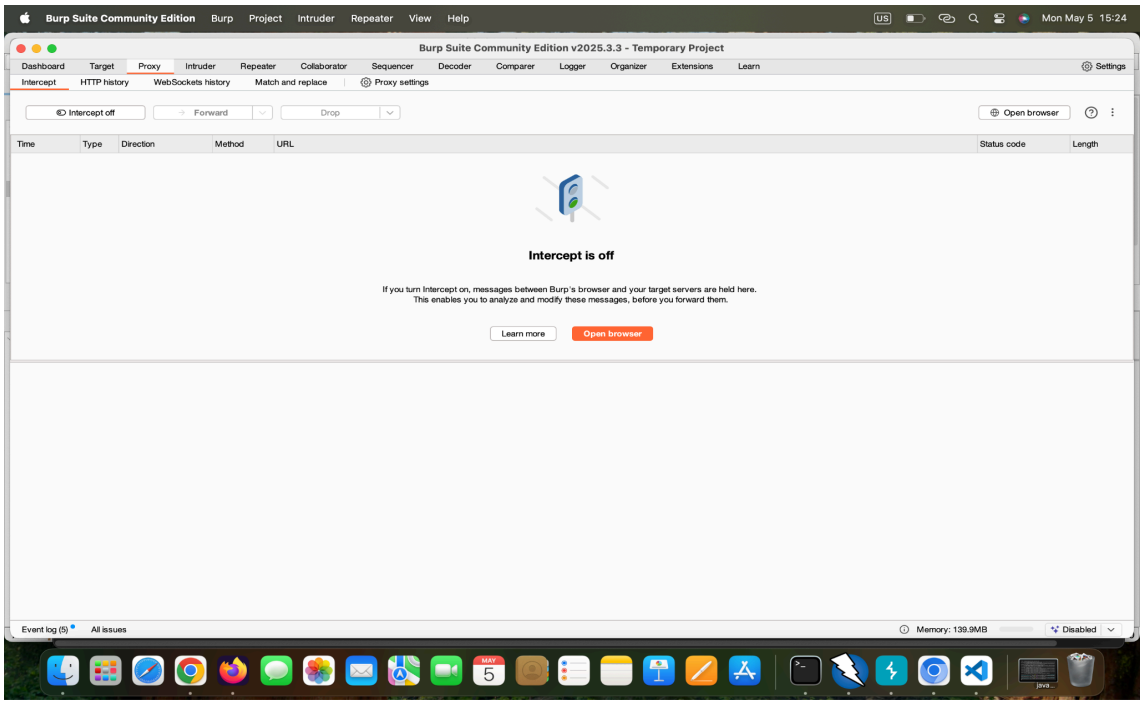


Fig -8: Step -6

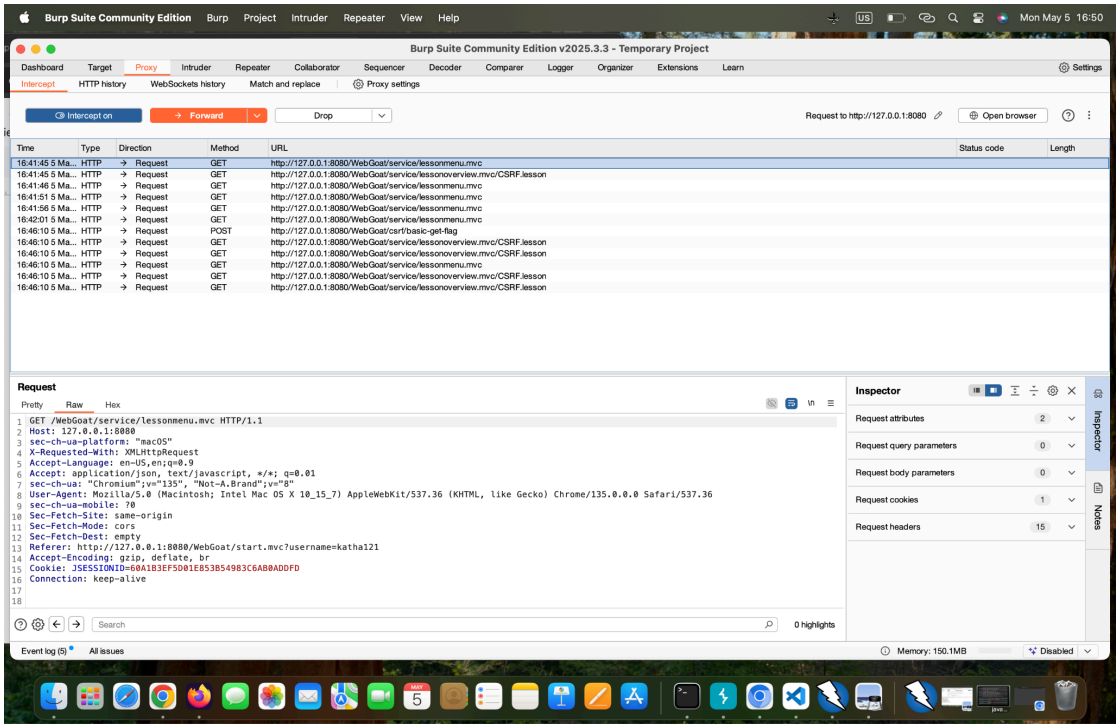


Fig -9: Step -7

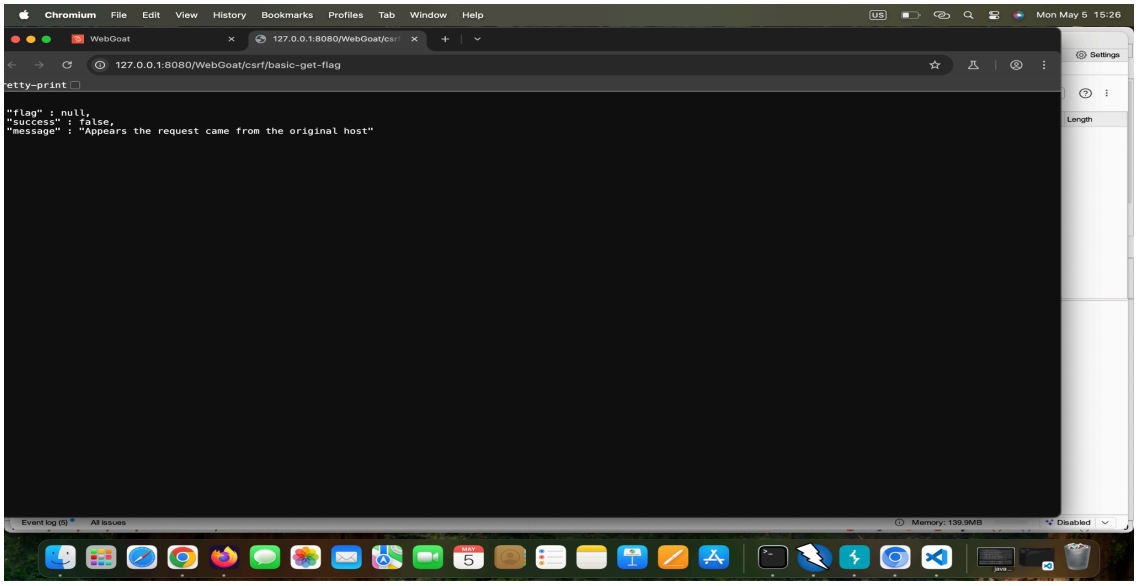


Fig -10: Step -8

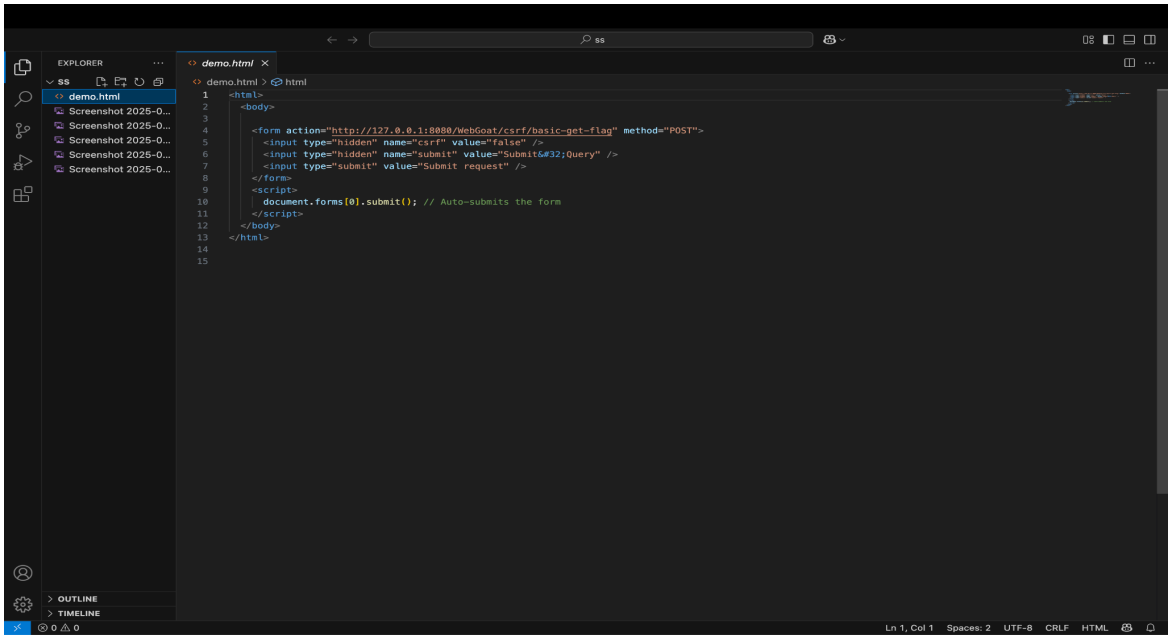


Fig -11: Step -11

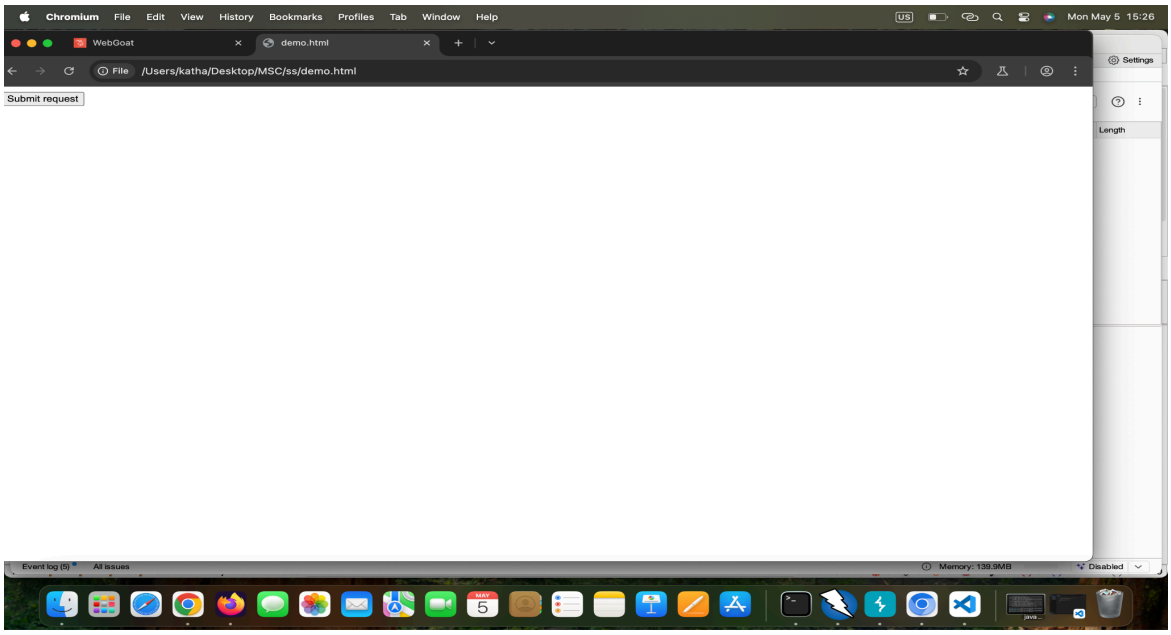


Fig -12: Step -11

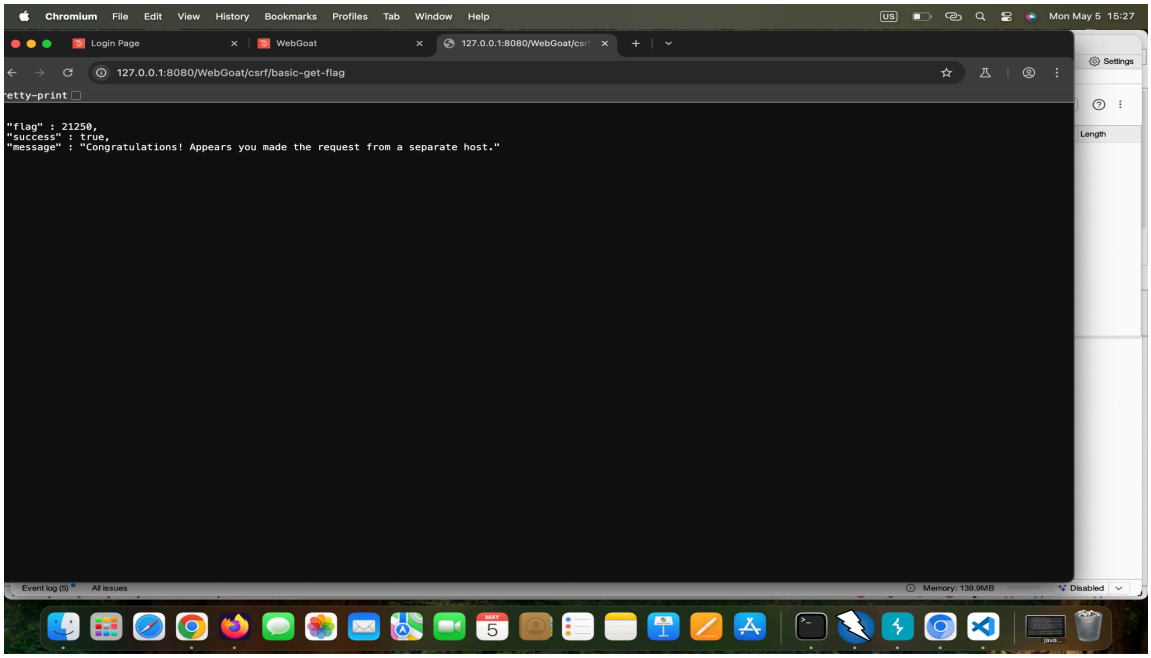


Fig -13: Step -11

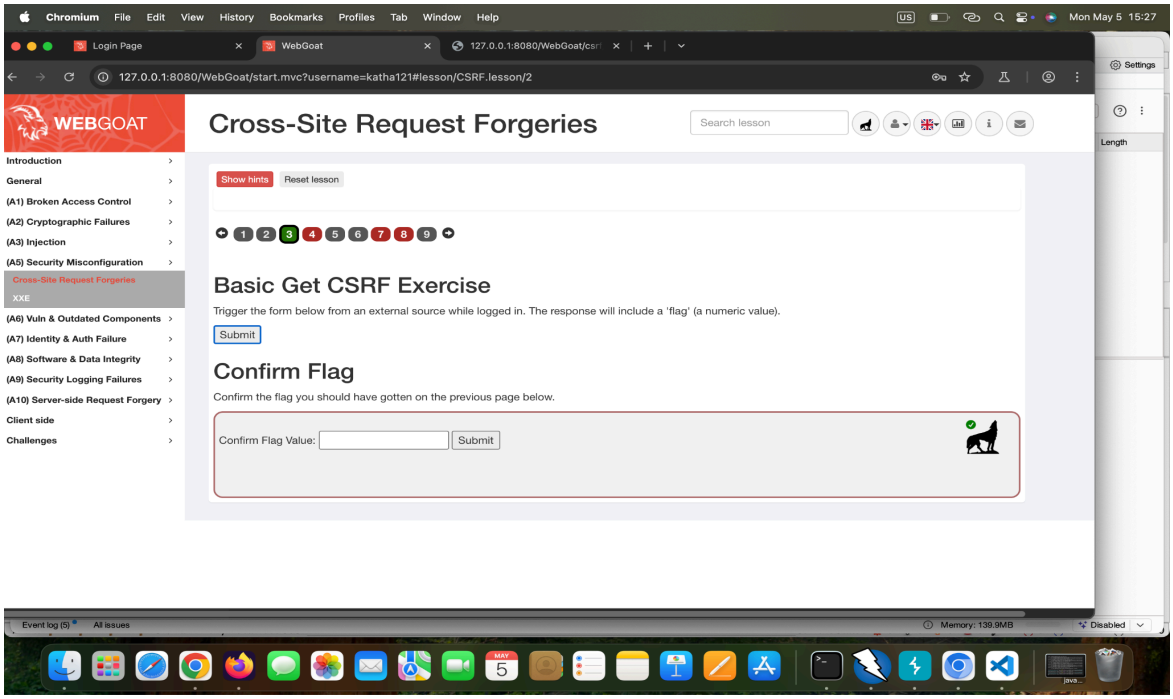


Fig -14: Step -12

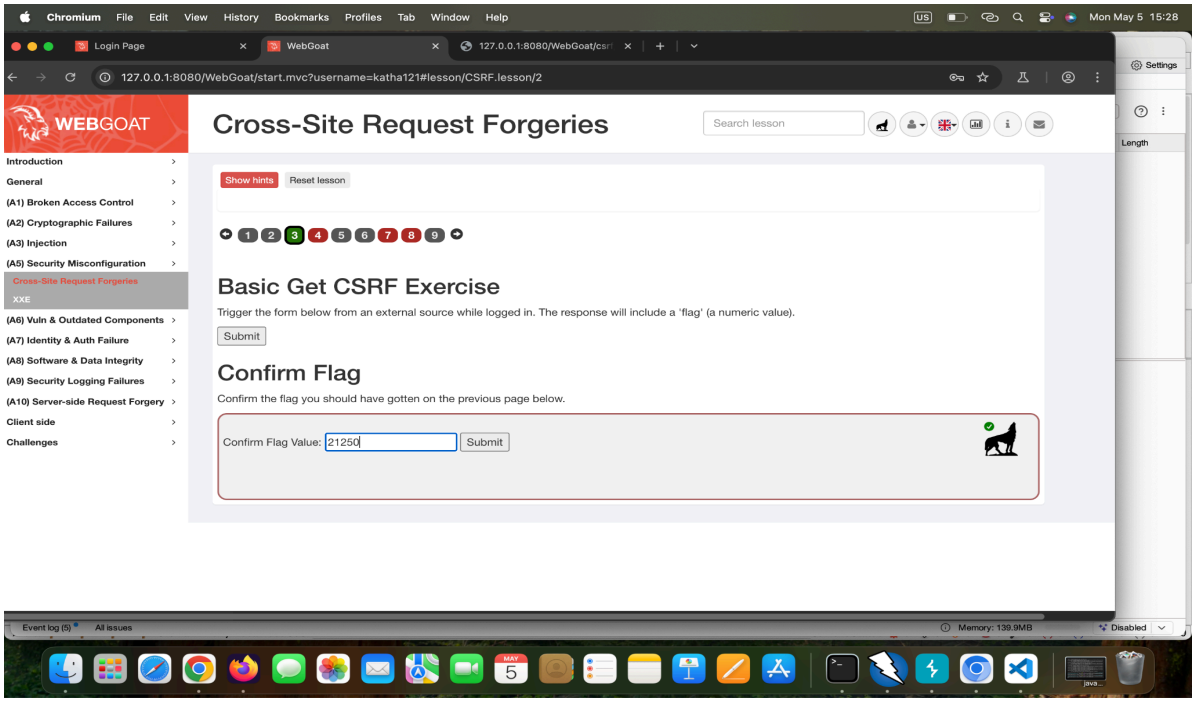


Fig -15: Step -12

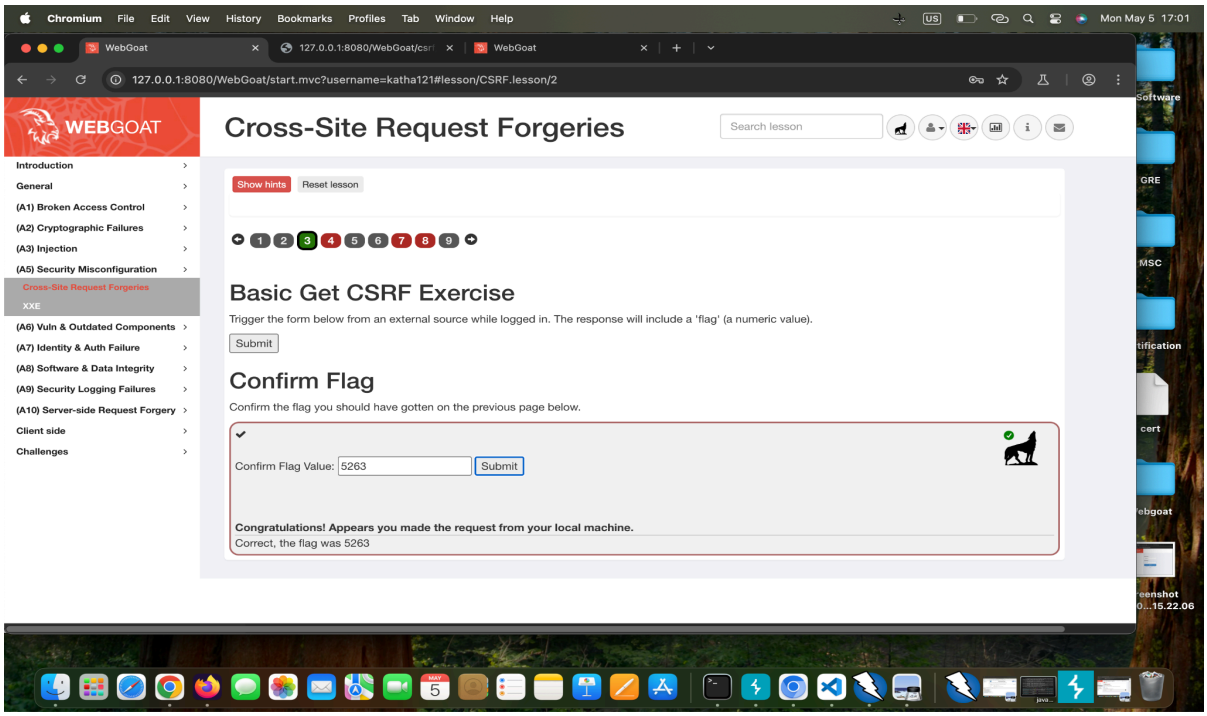


Fig -16: Step -12

❖ Scenario -2: Post a review on someone else's behalf

Step-1: Add two fields as - **Add Review and Stars.** Then Submit the review
Step-2: Check **Status, PayLoad, Response** from **Network Tab**
Step-3: According to the **PayLoad** and **Response** create a html file -
Step-4: Upload the html file. And check responses and status which must be shown in an accurate way. Where a successful message is also shown on the UI.

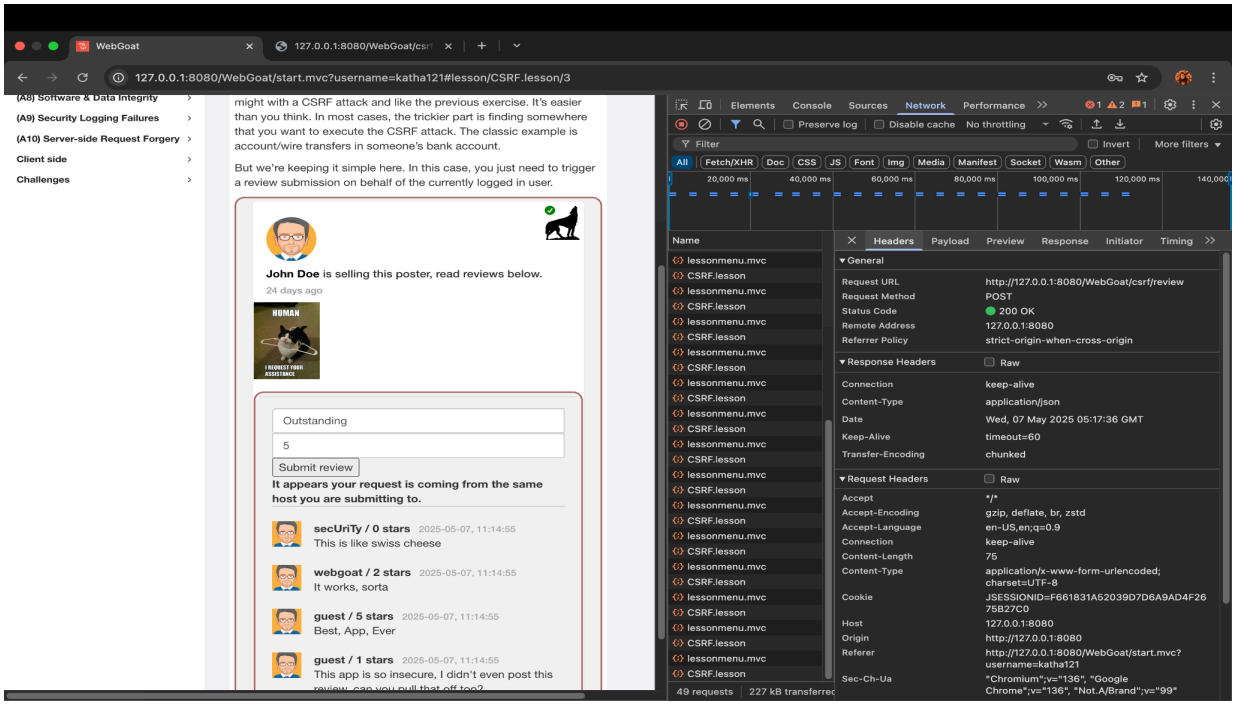


Fig - 17: Add Two fields

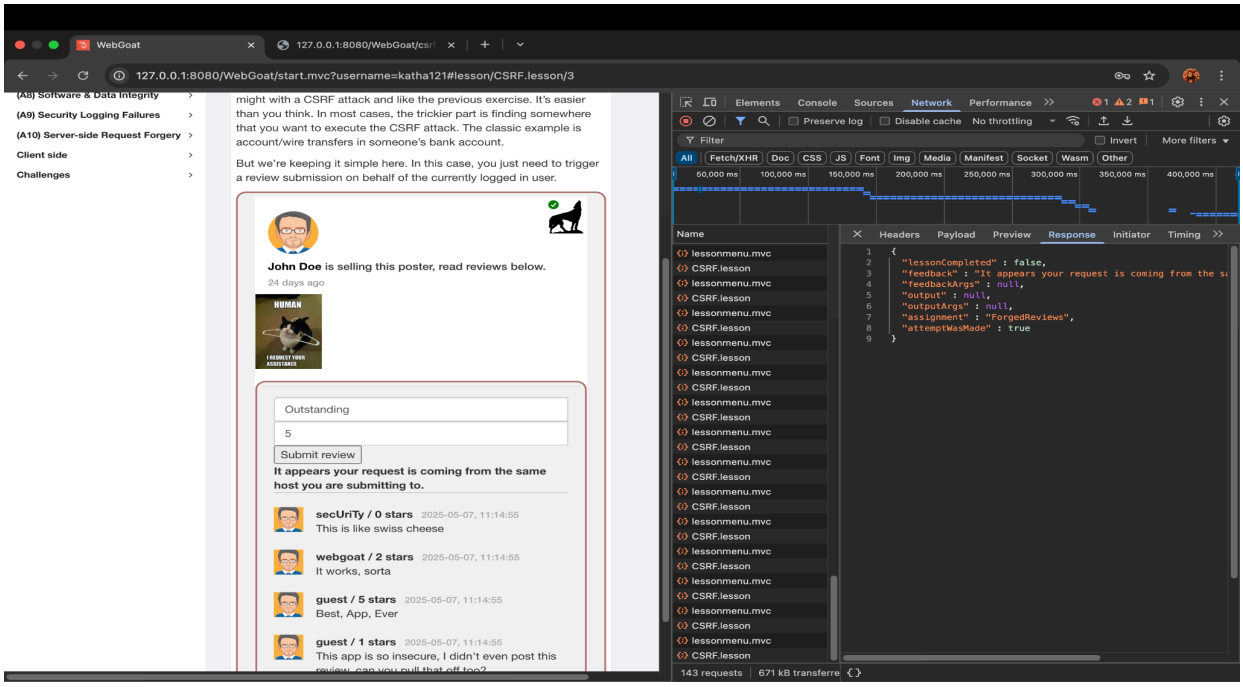


Fig - 18 : Check Network Tab

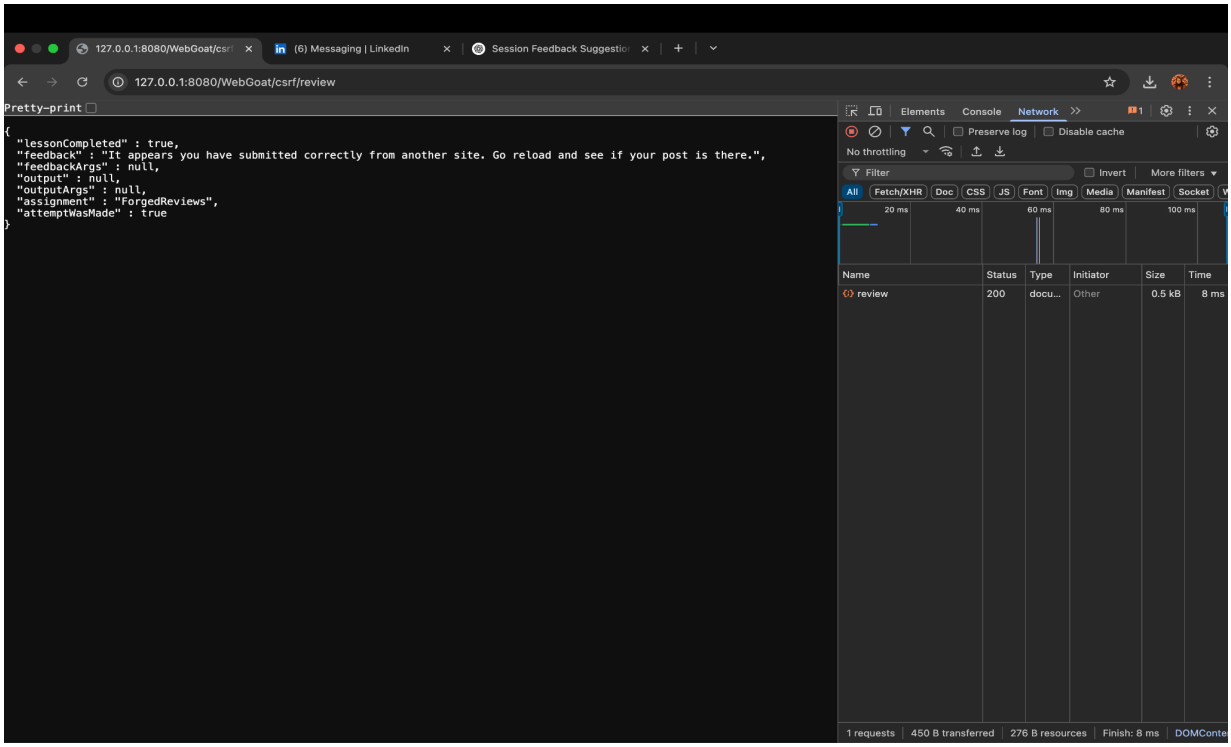


Fig - 19 : Check Status and Payloads

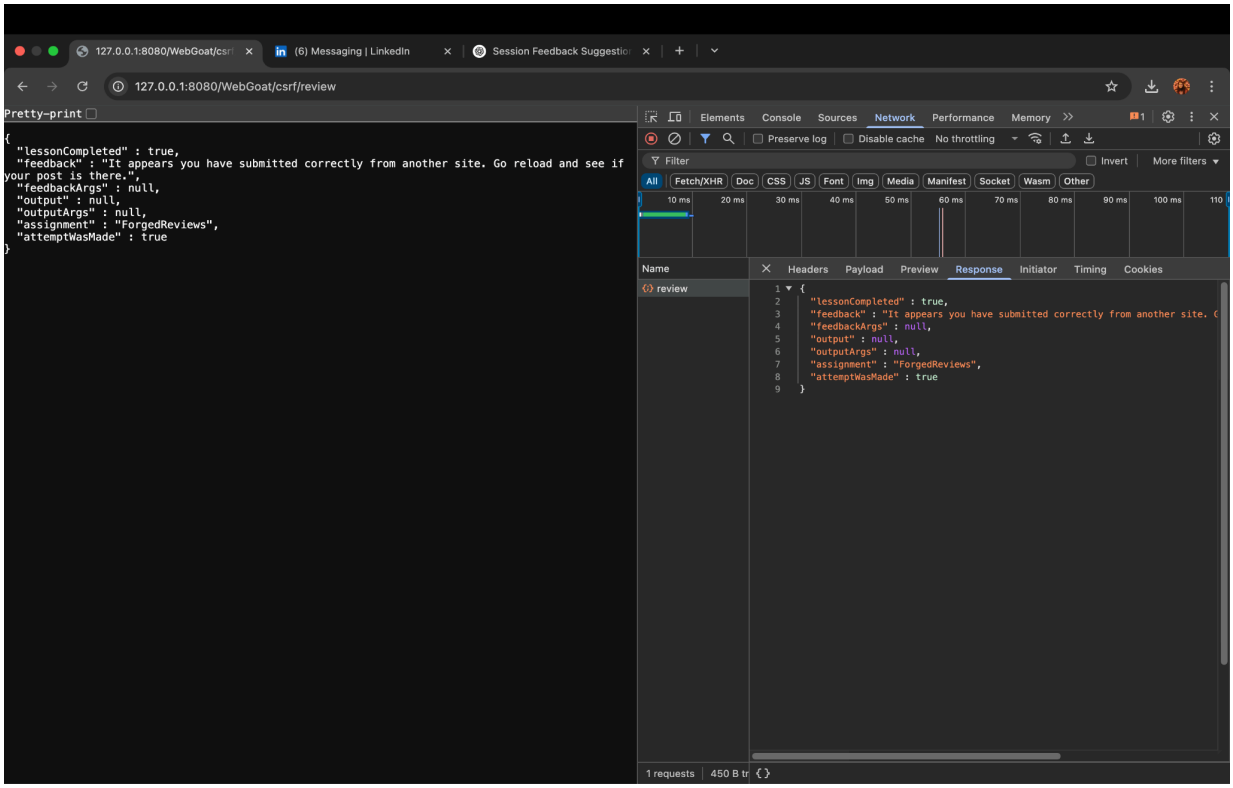


Fig - 20 : Check Responses

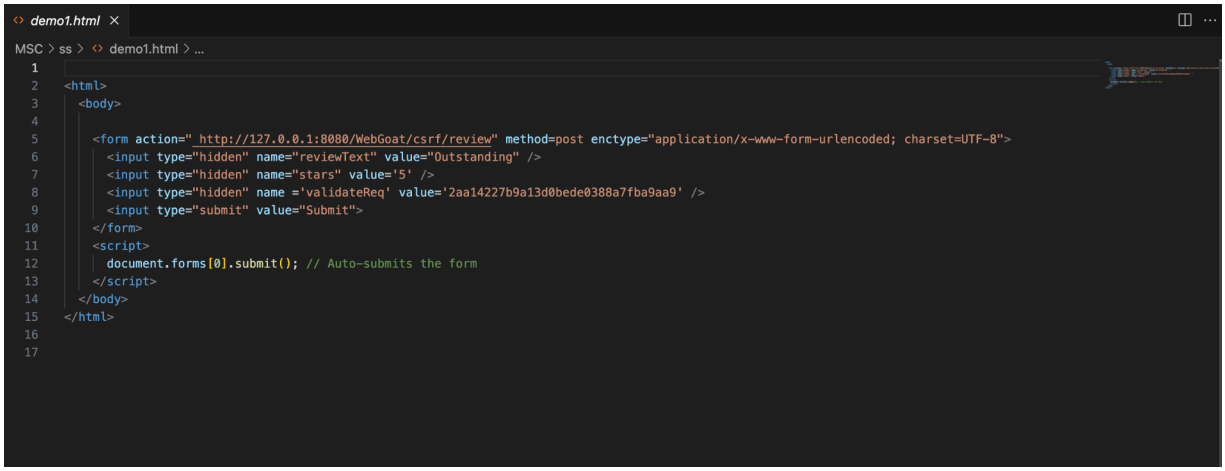


Fig - 21 : Html file

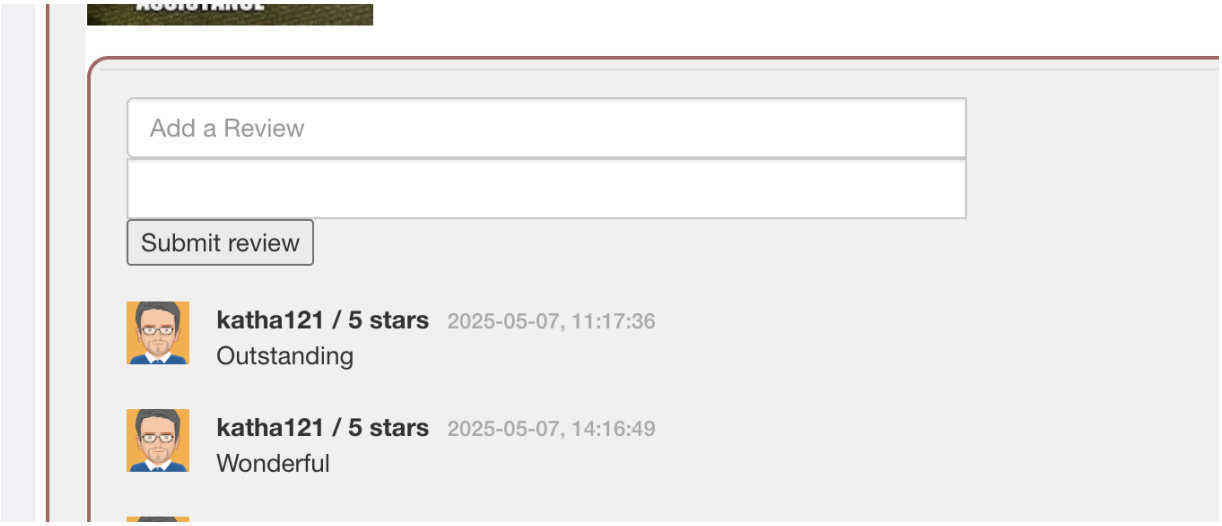


Fig - 22 : Final Result

❖ Scenario -3: CSRF and content-type

Step - 1: Check Content Type; For example - **json** or **plain text** etc.
Step -2: Fill the required Field and check the reponses, headers, status and payloads.
Step -3: According to the payload create a .html file and upload to the server.
Step - 4: Check UI to be sure that the content is uploaded accurately or not.
Step - 4: From Response we get a flag number which will be used to update the confirmed flag file and flag number should work and a successful message must be displayed.

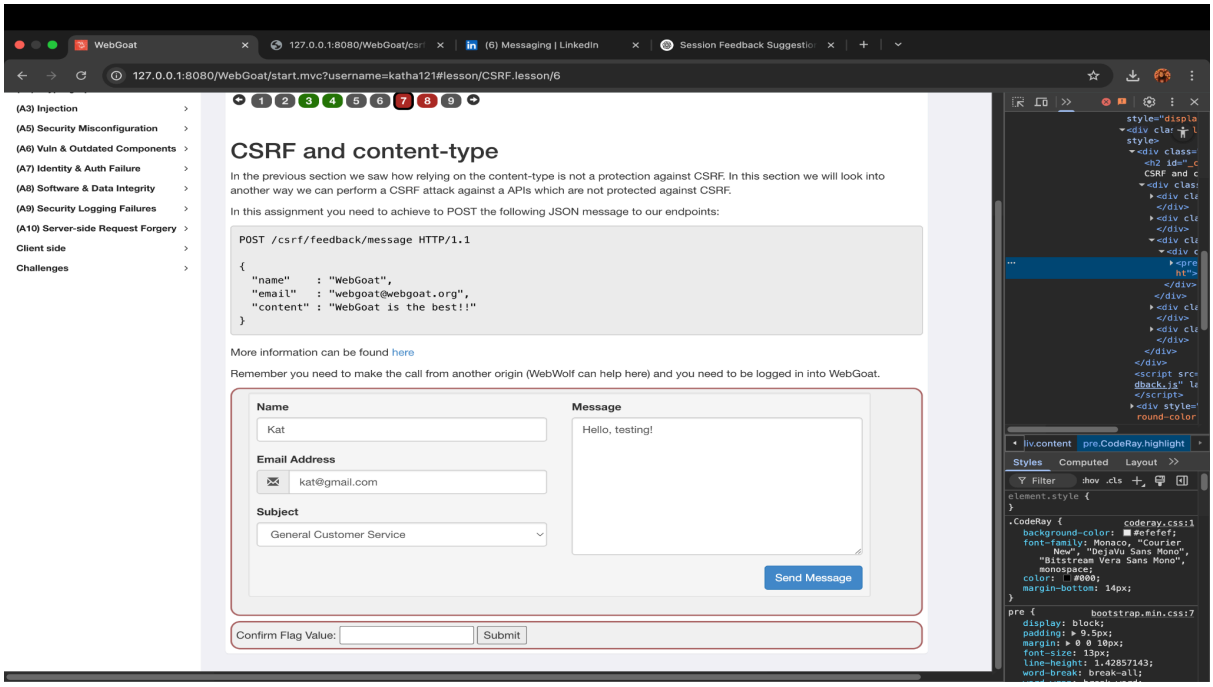


Fig - 23 : CSRF and Content Type

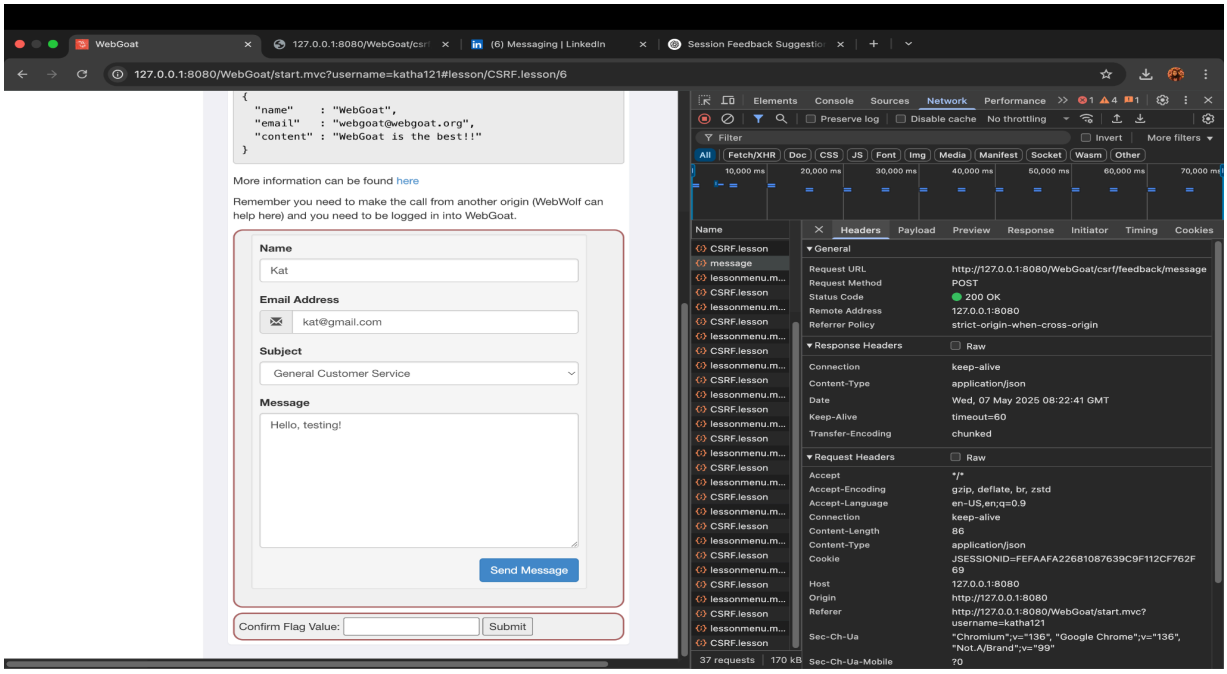


Fig - 24 : check Headers

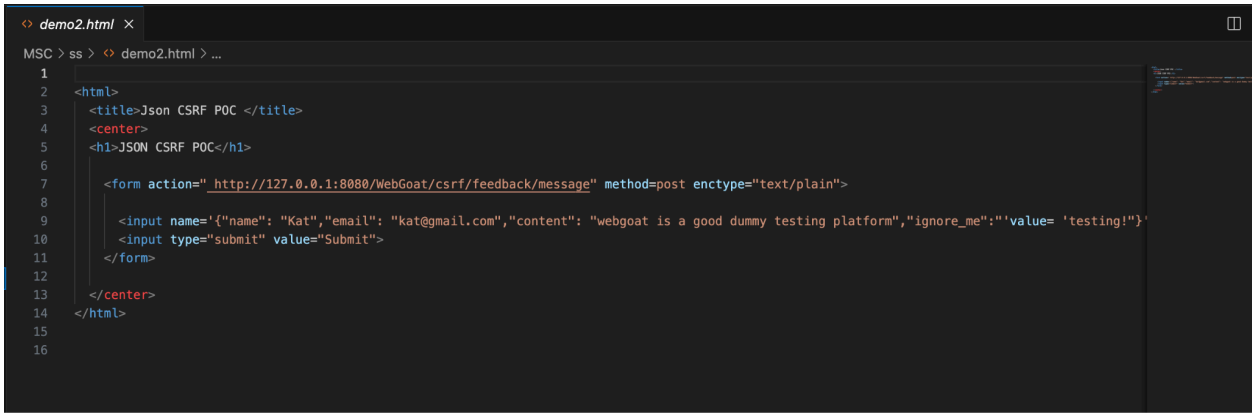


Fig - 25 : JSON CSRF POC.html

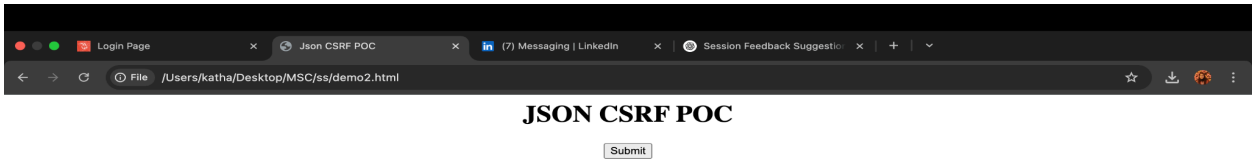


Fig - 26 : Successfully Updated File

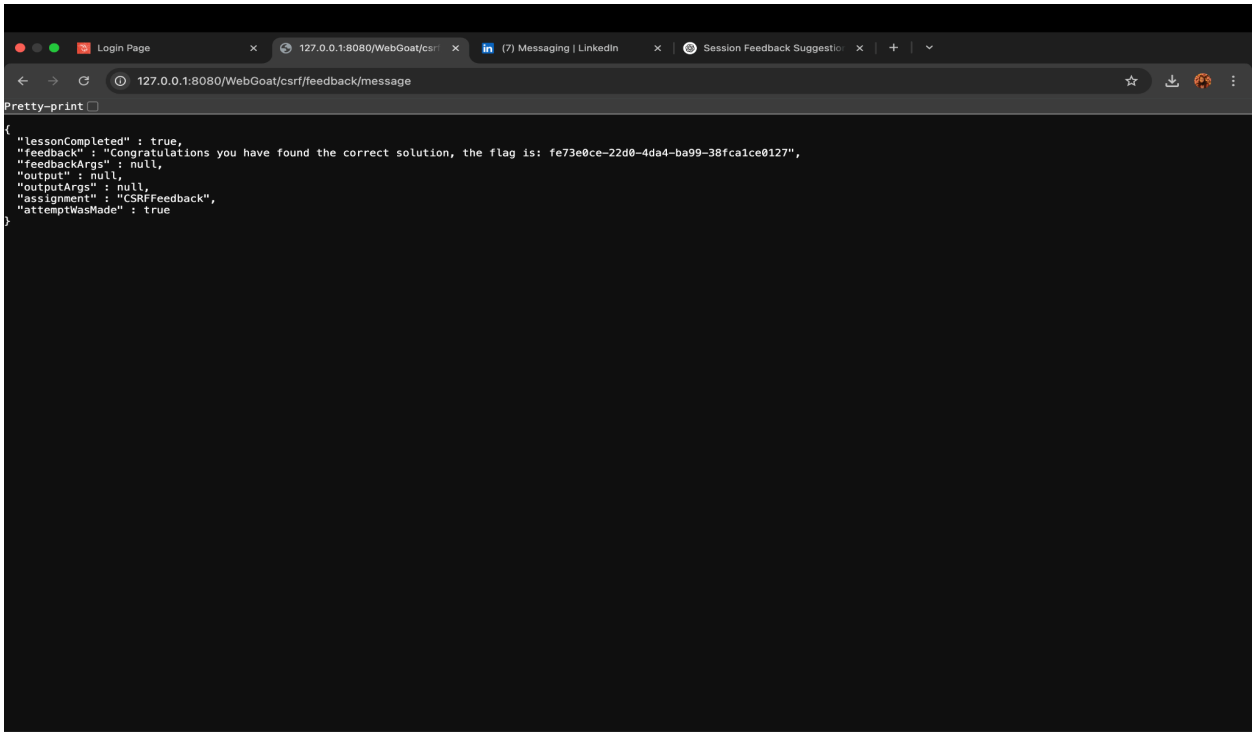


Fig - 27 : Response Check

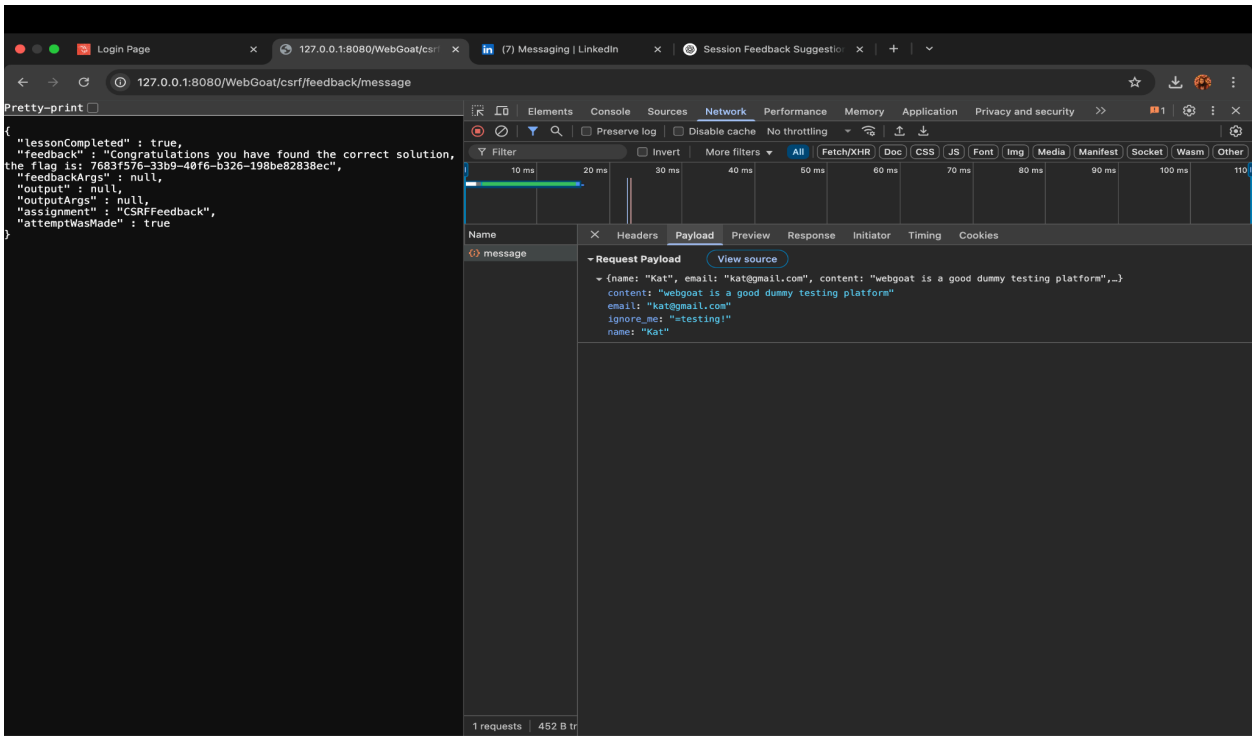


Fig - 28 : Payload Check

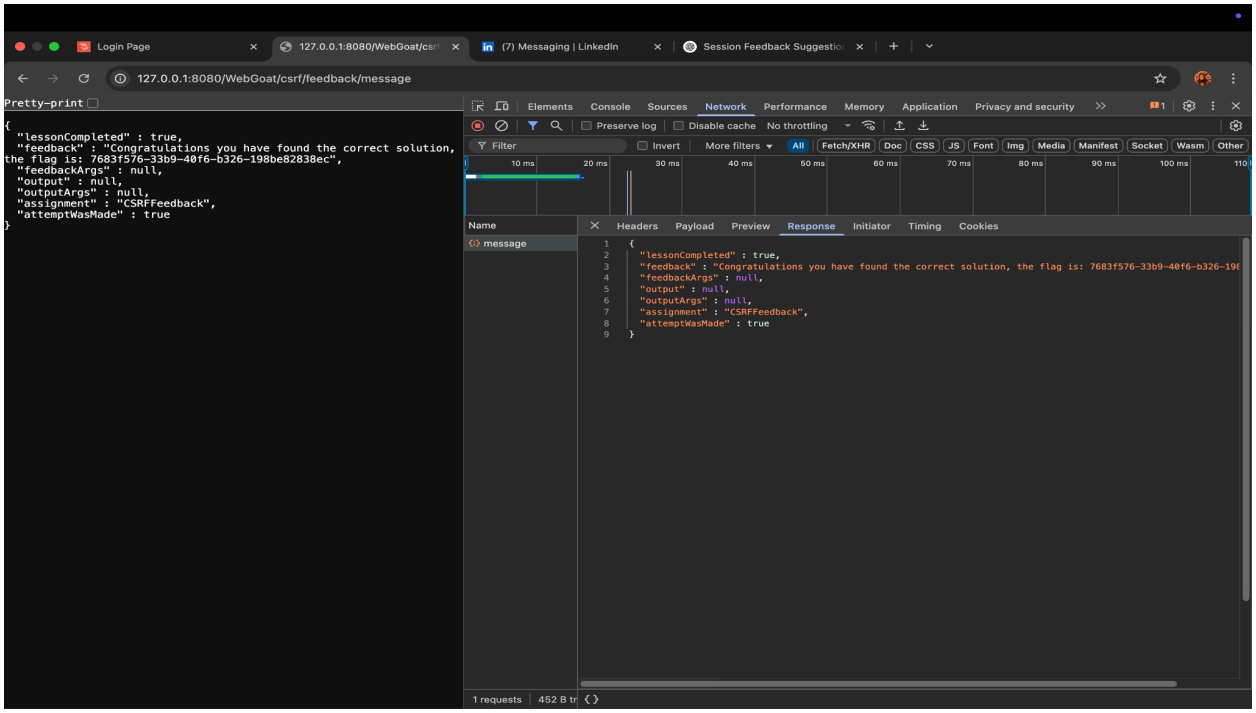


Fig - 29 : Response Check

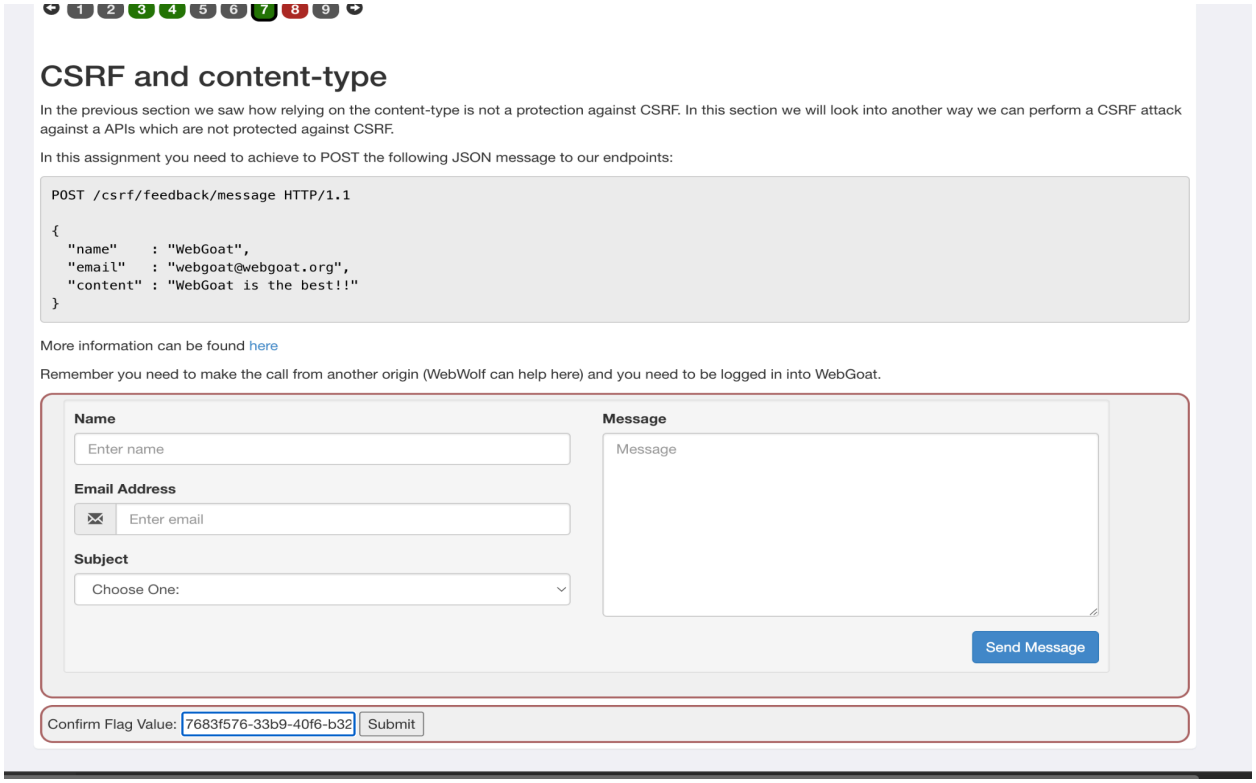


Fig - 30 : Update Flag Value

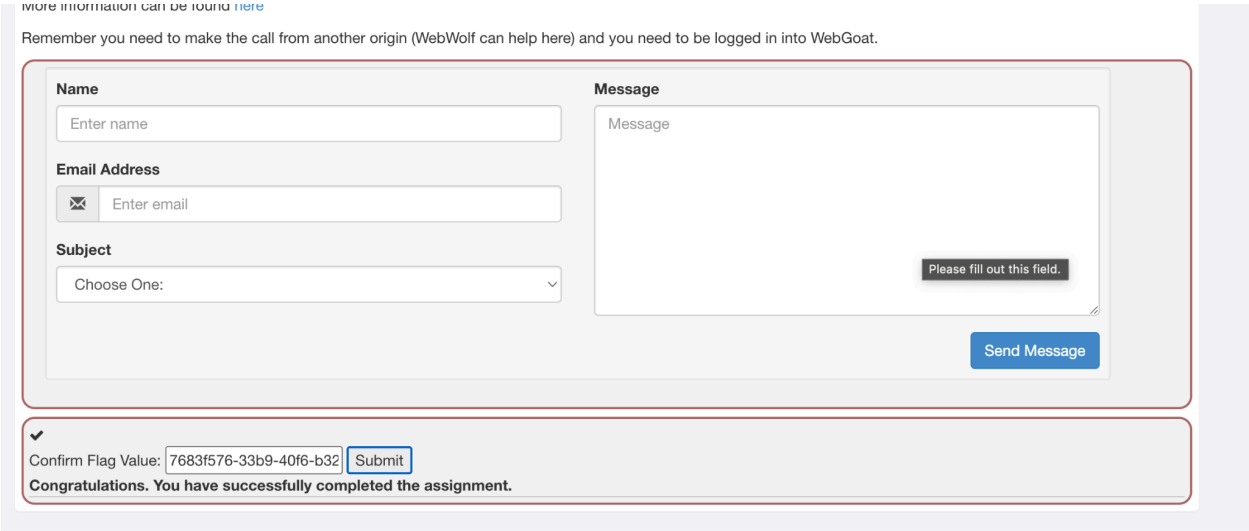


Fig - 31 : Final Result

Conclusion

The WebGoat exercises on **Cross-Site Request Forgery (CSRF)** and **SQL Injection** provided hands-on experience with two of the most common and critical web application vulnerabilities. Through practical exploitation and analysis, I gained a deeper understanding of how attackers can manipulate user sessions and query execution to compromise data integrity and confidentiality.

The CSRF exercise demonstrated how unauthorized commands can be submitted on behalf of authenticated users without their knowledge, emphasizing the importance of anti-CSRF tokens and proper session validation. In contrast, the SQL Injection tasks illustrated how unvalidated user input can be used to manipulate backend SQL queries, highlighting the need for parameterized queries and input sanitization.

These labs reinforced the significance of secure coding practices, the necessity of input validation, and the role of security testing in the software development lifecycle. Overall, this assignment deepened my appreciation for secure application development and the proactive steps required to mitigate such vulnerabilities.

🔧Part - 3 : Penetration Testing Report: WebGoat Application

Target Application: WebGoat

Test Type: Web Application Security Assessment

Tester: Jannatul Ferdous Katha

Course: Application Security Design and Management

Course Code: MCS-1102

Date: 11.05.2025

Contents

Executive Summary

This report presents the results of a penetration test conducted on the WebGoat application, a deliberately insecure platform designed for educational purposes. The test aimed to identify common vulnerabilities including SQL Injection and Cross-Site Request Forgery (CSRF), assess the security posture of the application, and provide recommendations for remediation.

Scope of Work

Coverage

- **Assessment Focus:** WebGoat modules related to authentication, data handling, and session management.
- **Attack Vectors:** SQL Injection, CSRF, and other OWASP Top 10 vulnerabilities.

Target Descriptions

- **Target System:** WebGoat (v8)
- **Environment:** Isolated virtual lab setup

Assumptions / Constraints

- The application is a controlled, intentionally vulnerable environment.
- No real user data was involved.
- Limited to WebGoat modules only; not extended to other network components.

Methodology

The test was done according to penetration testing best practices. The fow from start to finish is listed below -

Pre-Engagement

- Environment setup (WebGoat, OWASP ZAP, browser tools)
- Defined testing goals and scope

Penetration Testing

- Manual exploitation and automated scans using OWASP ZAP
- Vulnerability validation and documentation

Post-Engagement (On-Demand)

- Discussion of findings
- Support for remediation and validation of fixes

Risk Factors

- Limited scope excludes full application stack
- Educational environment differs from real-world systems

Criticality Definitions

Severity	Description
High	Exploits that can compromise data or system integrity with minimal effort
Medium	Vulnerabilities requiring some level of access or user interaction
Low	Minor risks or those mitigated by existing controls
Informational	Observations that do not present immediate risk

---

Summary of Findings

The following charts group discovered vulnerabilities by OWASP vulnerability Type and overall estimated severity

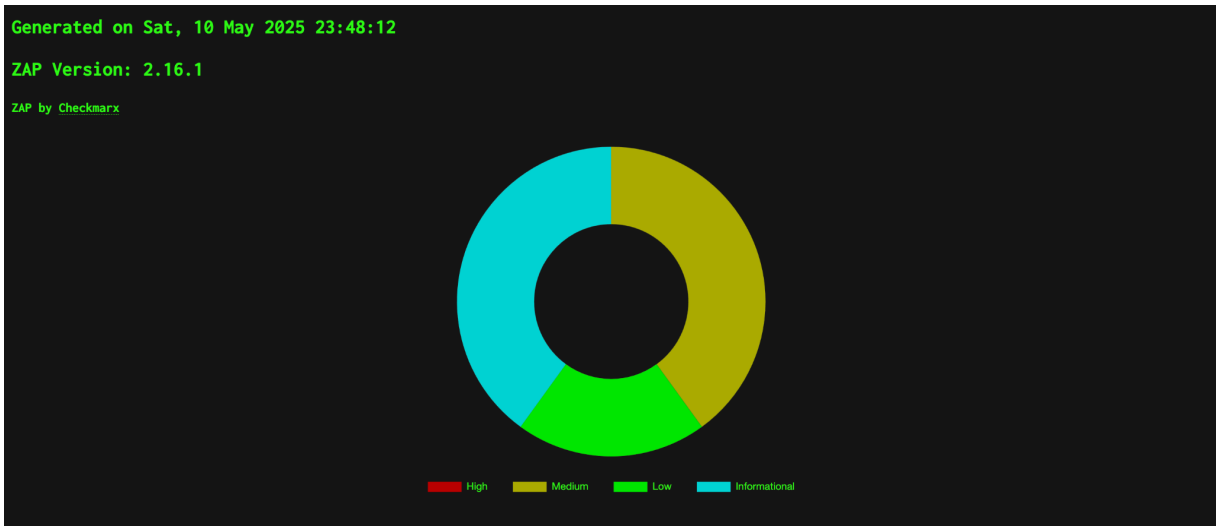


Fig: Risk Analysis Chart

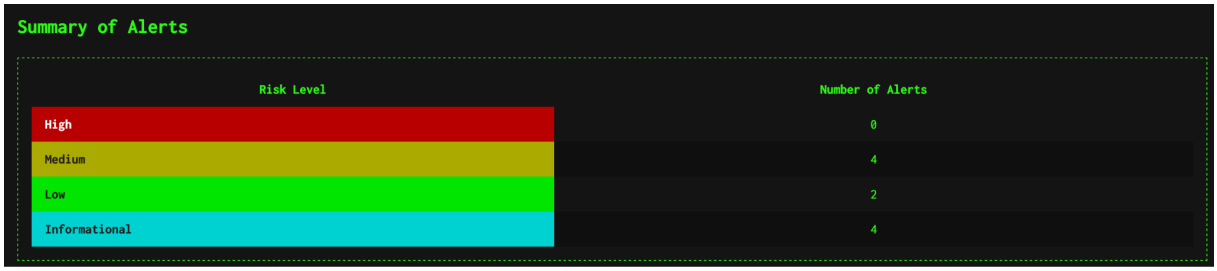


Fig: Summary of Alerts

Fig: List of Alerts with Risk level and No of Instances

Vulnerability Impact:

By Using ZAP found few vulnerabilities of WebGoat such as -

- Absence of Anti-CSRF Tokens
- Content Security Policy(CSP)
- Missing Anti-clickjacking header
- Authentication Required Identified and so on.

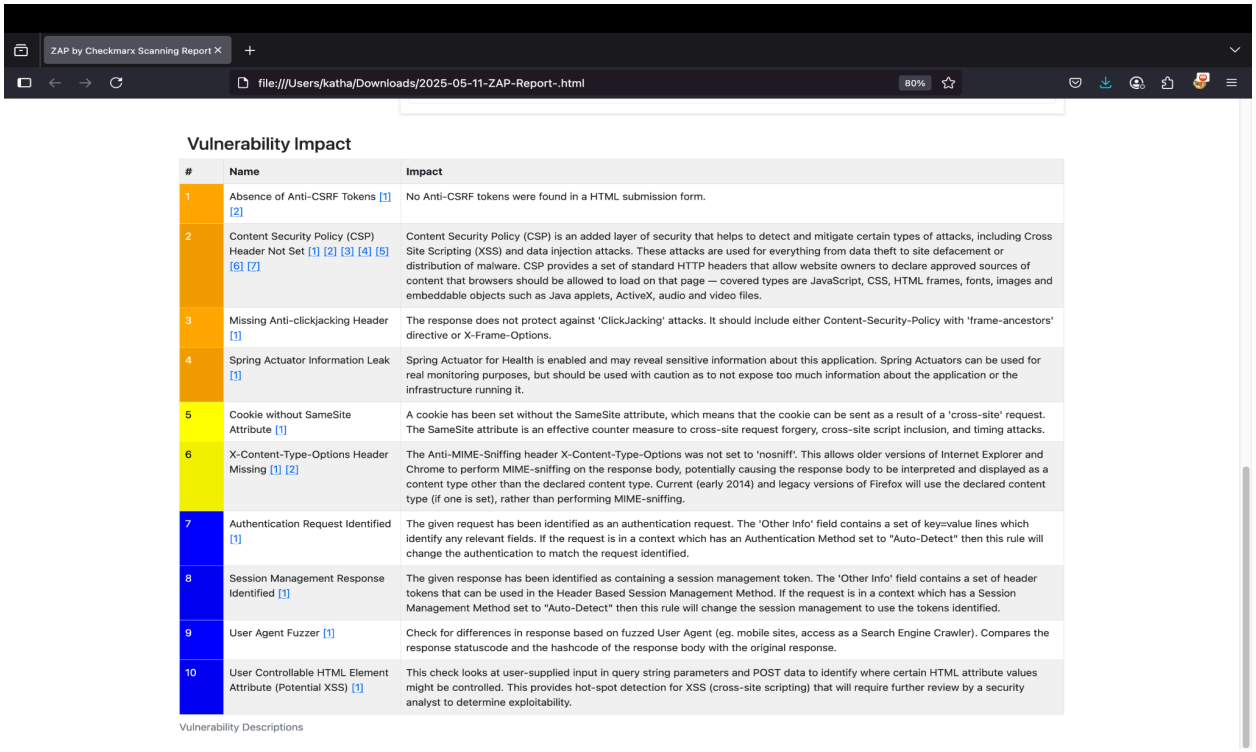


Fig: Vulnerability Impact

Analysis

- SQL Injection vulnerabilities were identified that allowed bypassing authentication and accessing restricted data.

- CSRF vulnerabilities were found in user settings modules, enabling unauthorized state changes.

General Risk Profile

- High-risk vulnerabilities were present due to lack of input validation and session protection mechanisms.
- The application's architecture supports demonstrative exploitation, underscoring the importance of secure coding practices.

Summary of Recommendations

- Implement **parameterized queries** and input validation to mitigate SQL Injection.
- Introduce **anti-CSRF tokens**, and enforce **same-site cookies**.
- Ensure secure session management and proper authentication mechanisms.
- Regularly perform code reviews and security testing.

Post-Test Remediation

- Guidance provided to secure vulnerable modules
- Follow-up tests can be scheduled upon request

Terms

This report is confidential and intended solely for educational evaluation. The findings are based on a simulated test environment and may not represent real-world application threats unless similarly configured.

Appendix 1 – Finding Details

Finding: SQL Injection

- **Module:** SQL Injection (Classic)
- **Payload:** ' OR '1'='1
- **Impact:** Login bypass, data extraction
- **Recommendation:** Use parameterized statements
- **Attachments:**

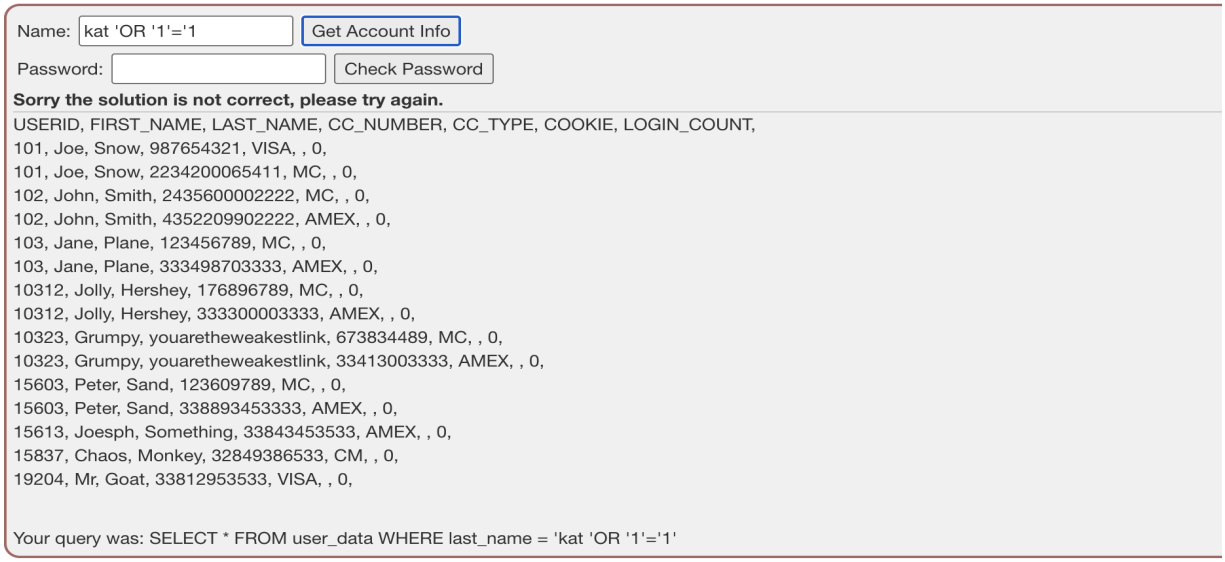


Fig: SQL injection with payload

Finding: CSRF

- **Module:** CSRF attack demonstration
- **Technique:** Malicious form auto-submission
- **Impact:** Unauthorized user changes
- **Recommendation:** Implement CSRF protection tokens
- **Proof of concept (POC)**

Scenario: An attacker crafts a malicious HTML page and tricks a logged-in user into visiting it and shares a fake review, generates flag value and also can change the content type such as Application/json to plain Text.

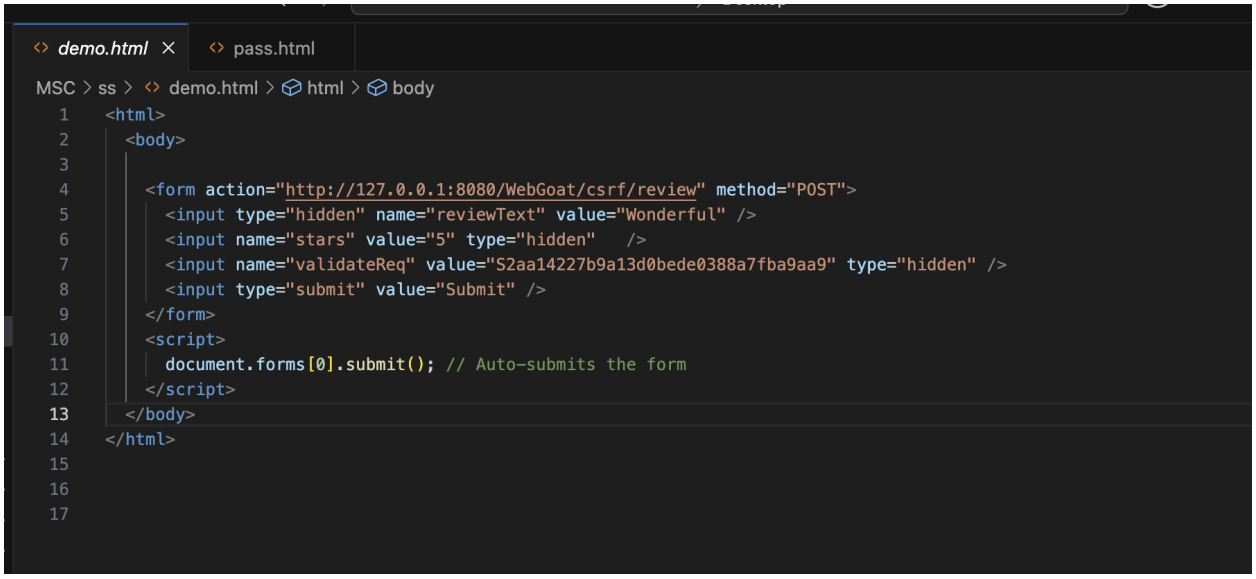


Fig - 1: Fake Review html file.

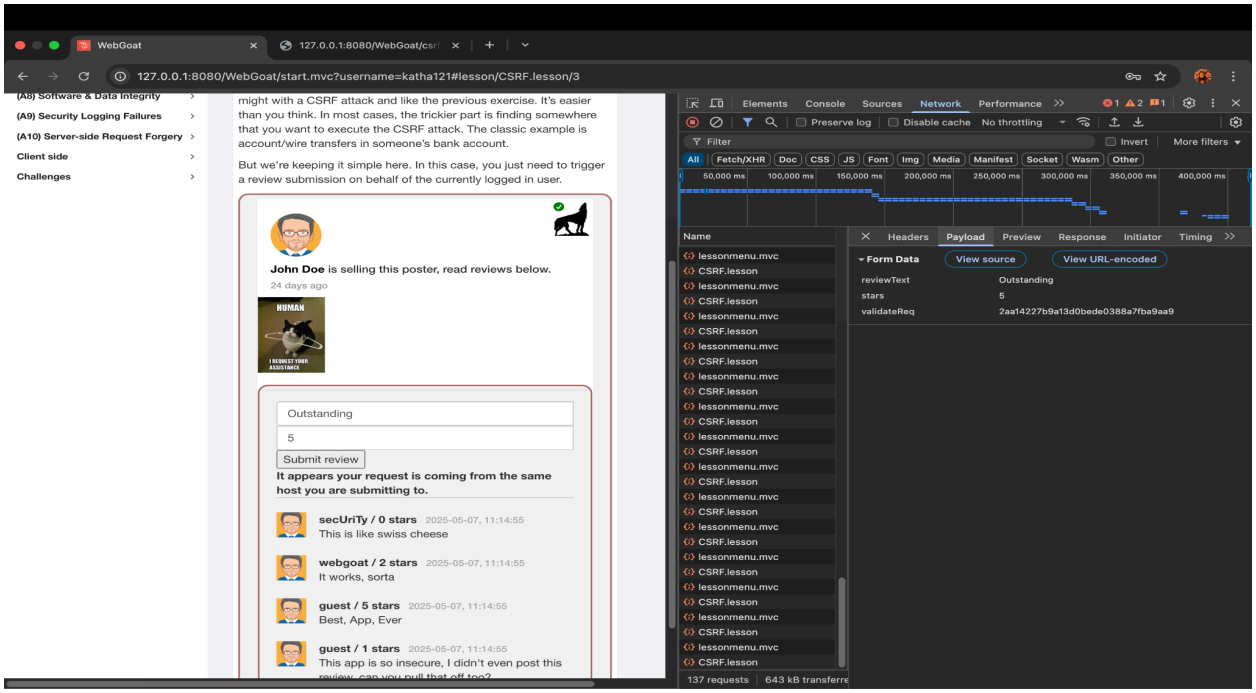


Fig - 2: Review text update

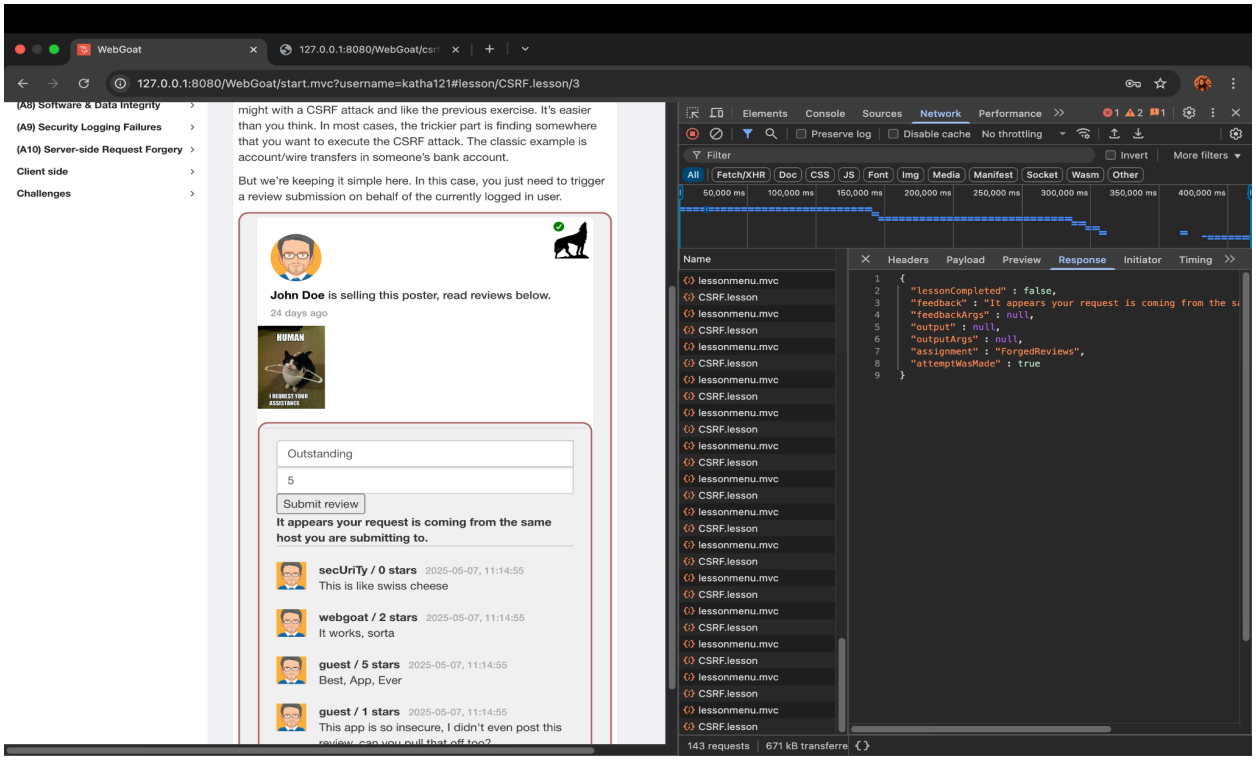


Fig - 3: Payload for fake review

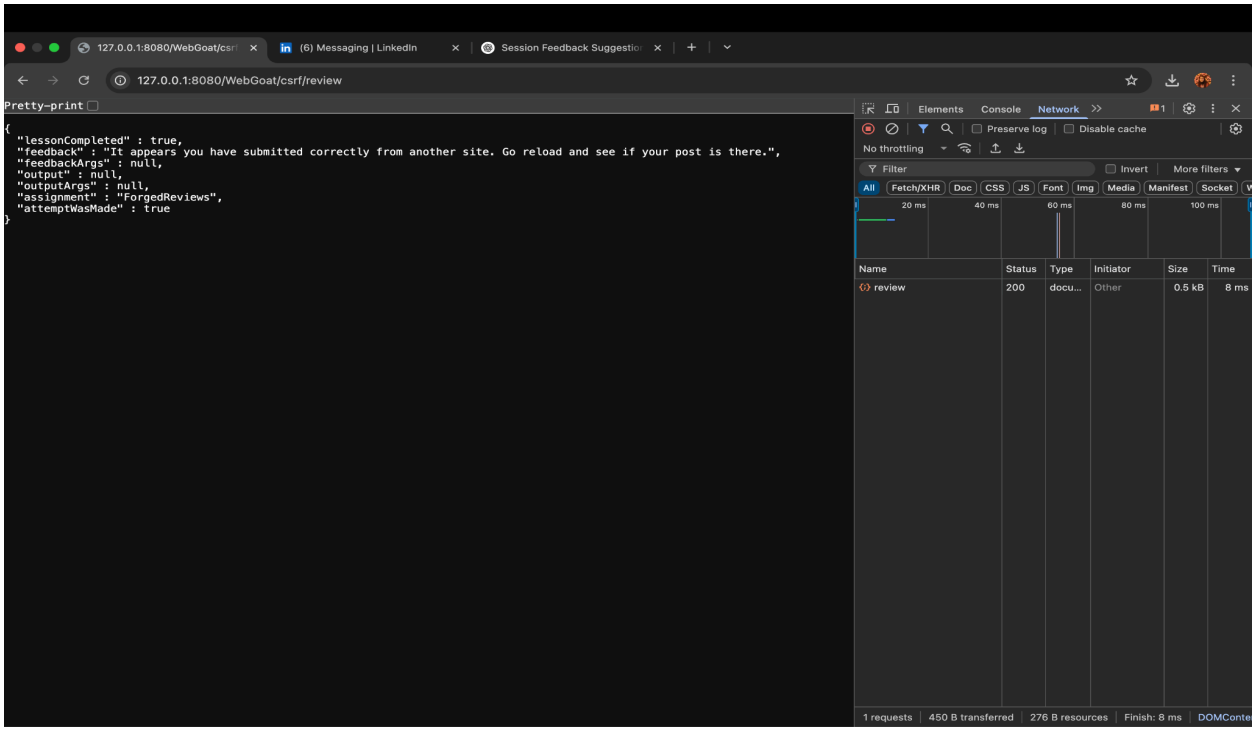


Fig - 4: Response for Fake Review

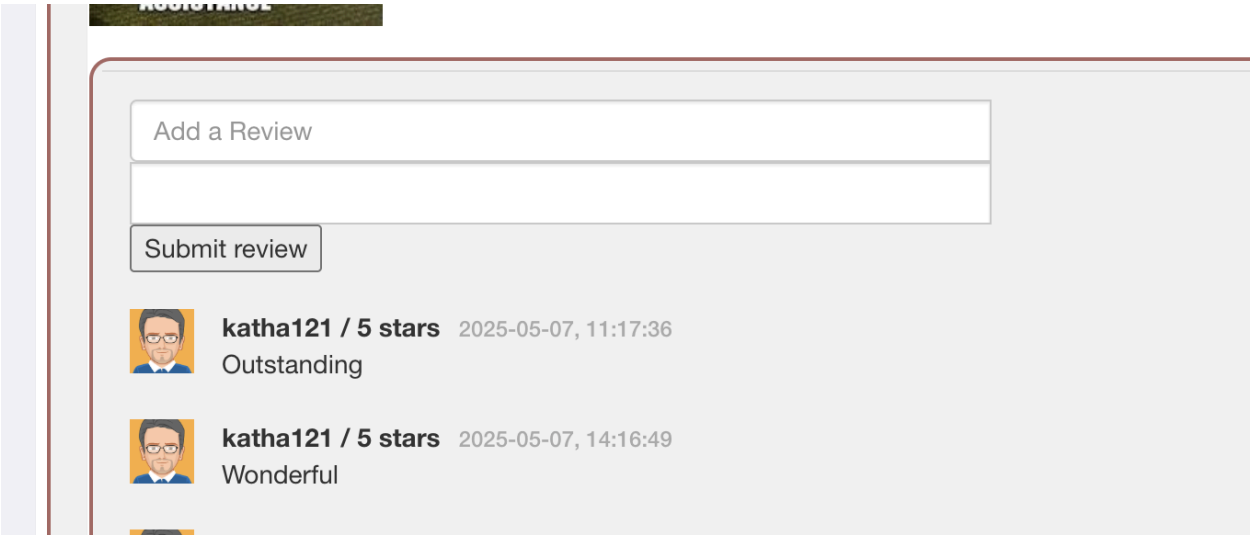


Fig - 5: Final Result For Fake Review

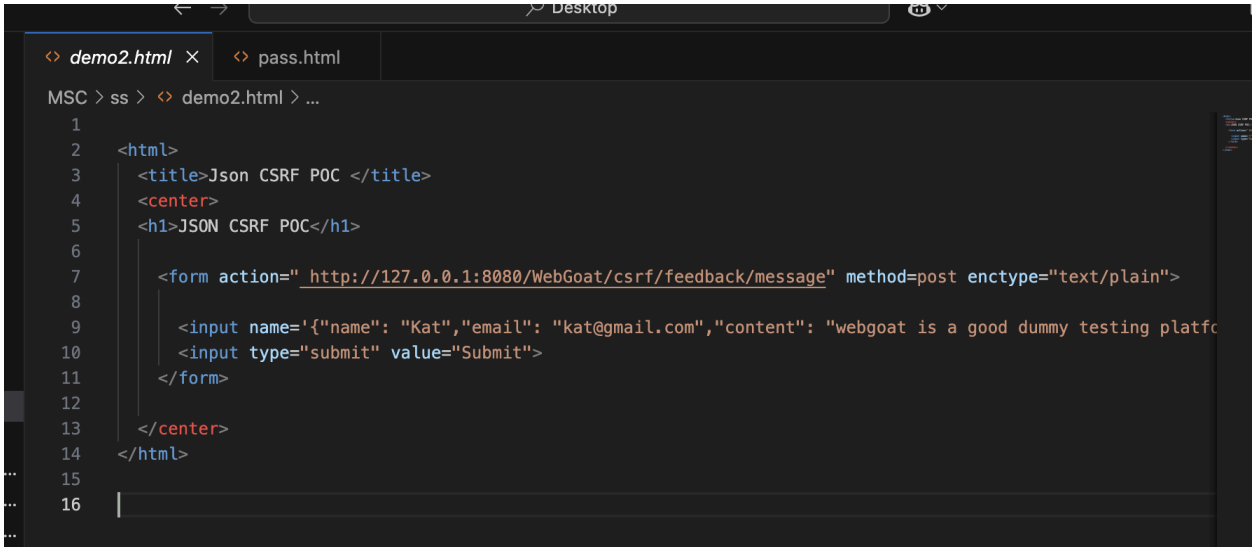


Fig - 6: Content type break html file

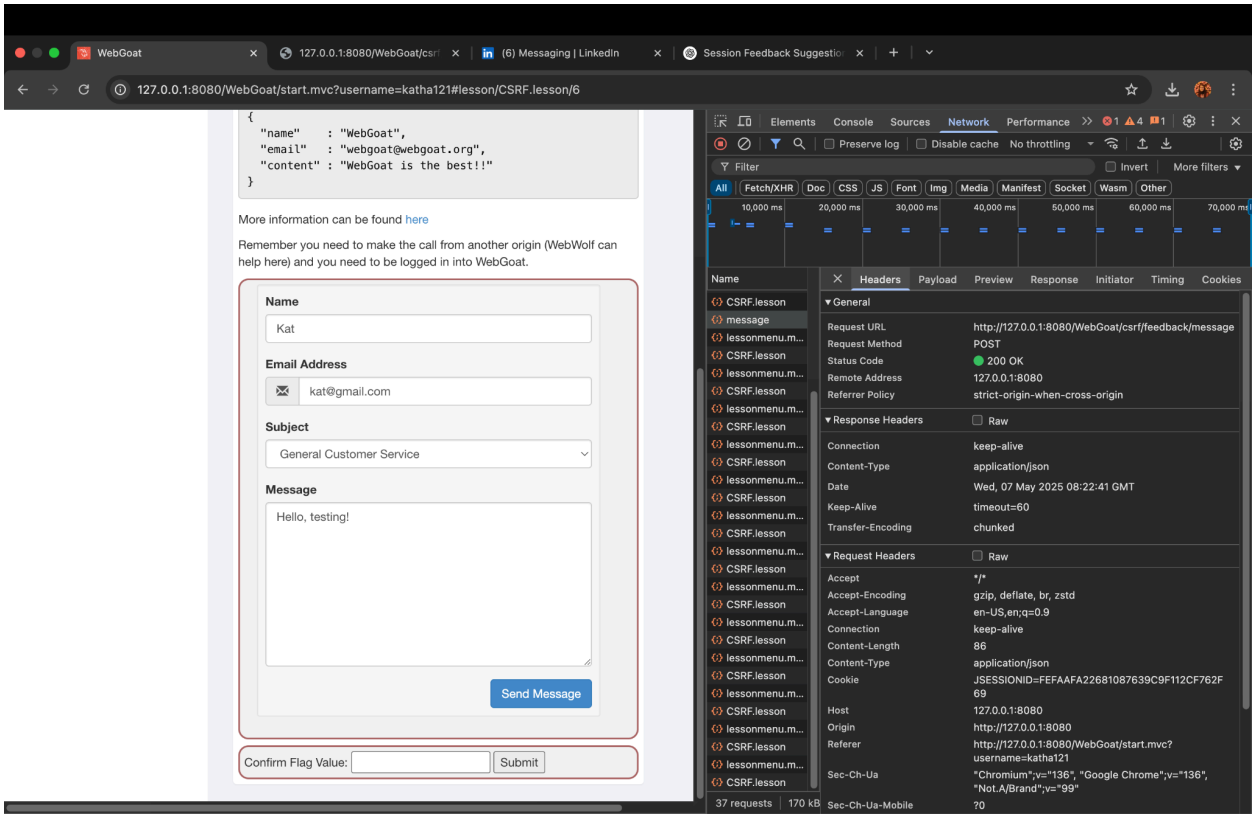


Fig - 7: Updated Content type

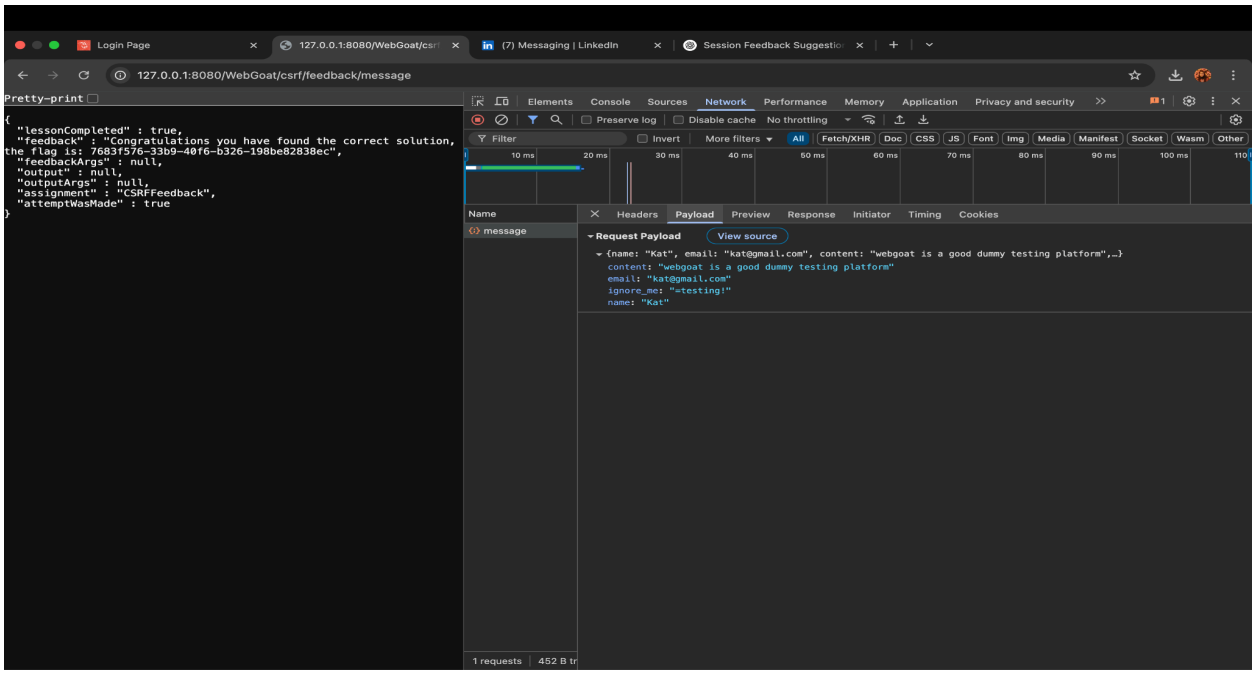


Fig 8 - Updated Payload

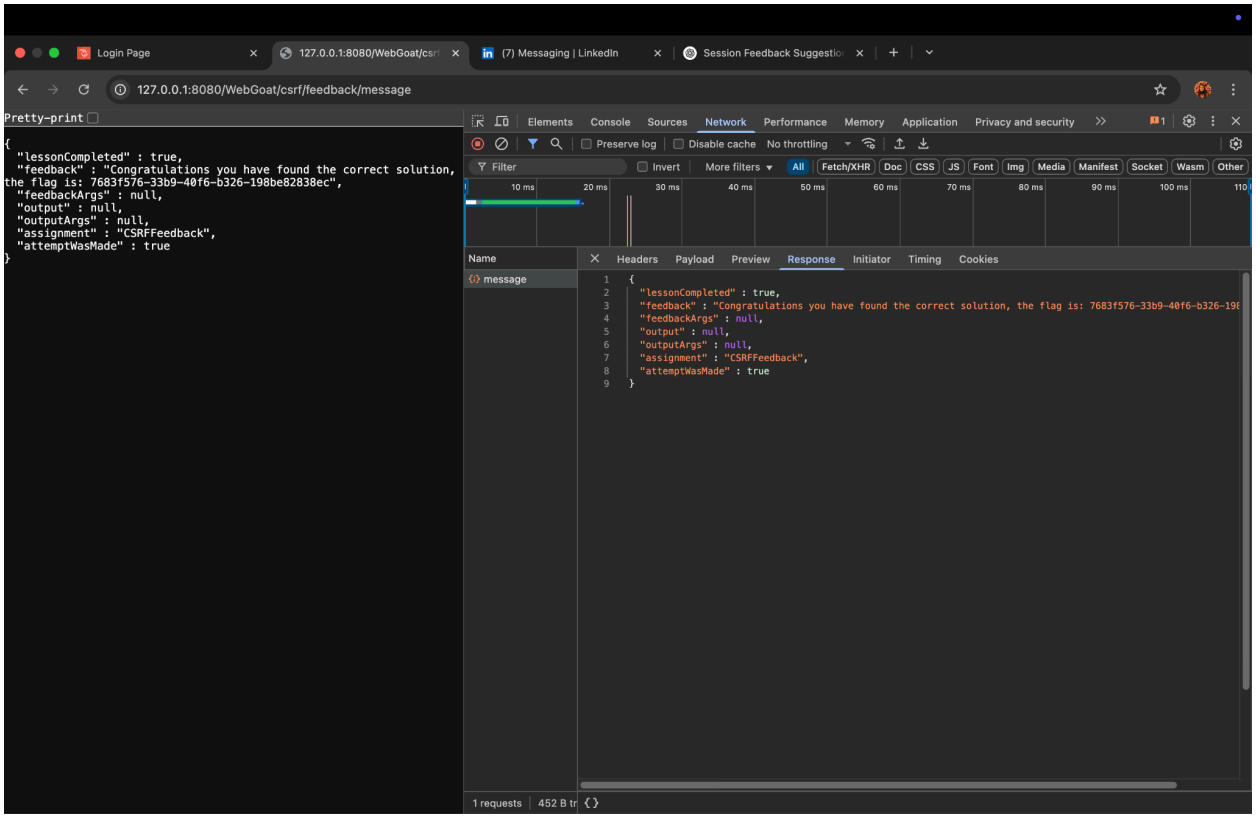


Fig 9 : Flag Value for updated content type

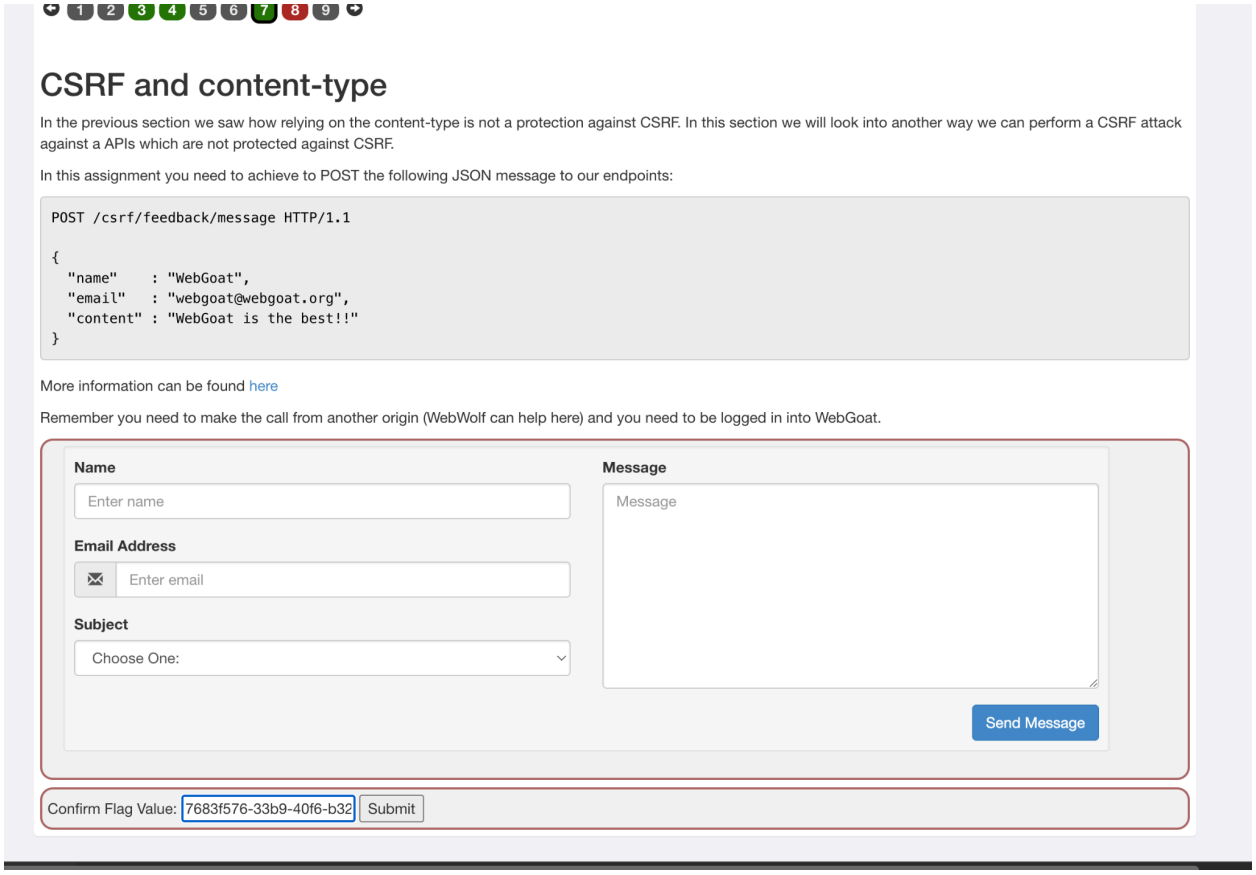


Fig:10 Flag Value Submission

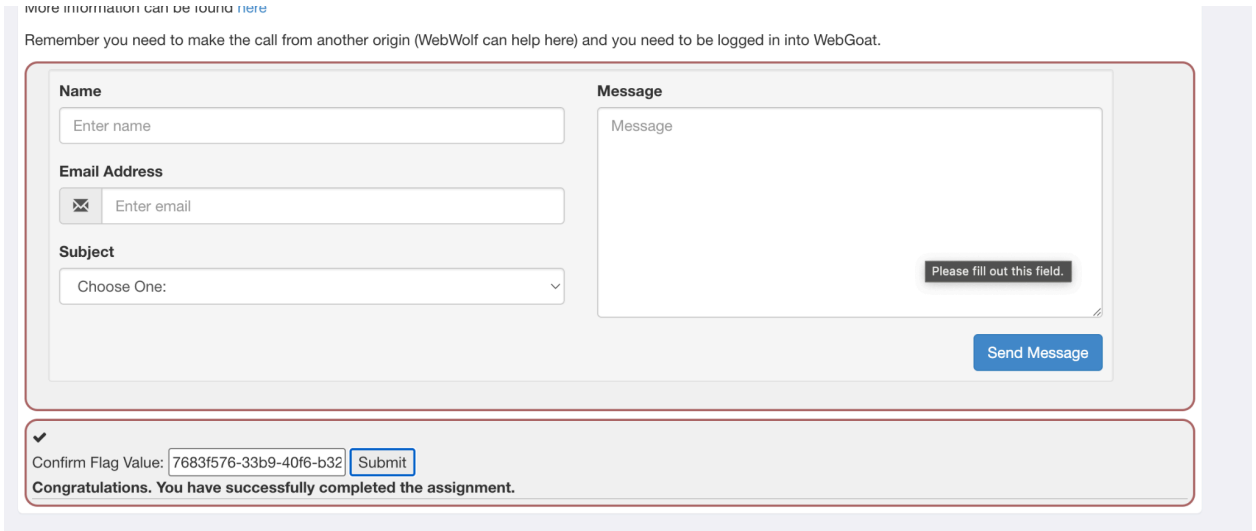


Fig -11 : Successfully updated flag value.

Observation:

Upon visiting the attacker's page, the form auto-submits a request to WebGoat, successfully executing a state-changing action on behalf of the authenticated user — without the user's interaction or consent.

Impact

- Unauthorized actions may be executed using a logged-in user's session.
- Critical account information could be altered.
- Possibility of privilege escalation or data manipulation.

Recommendation

To mitigate CSRF attacks, the following measures should be implemented:

1. **Anti-CSRF Tokens**: Generate and validate unique tokens for every form submission and state-changing HTTP request.
2. **SameSite Cookies**: Configure session cookies with the SameSite=Strict or Lax attribute.
3. **User Confirmation**: For sensitive actions, implement a user re-authentication or confirmation step.
4. **Use Security Frameworks**: Leverage modern frameworks that provide built-in CSRF protection mechanisms (e.g., Spring Security, Django, etc.).

References

- OWASP CSRF Guide: https://owasp.org/www-community/attacks/csrf
- CWE-352: https://cwe.mitre.org/data/definitions/352.html
- OWASP Top 10 2021: https://owasp.org/Top10/

Conclusion

The absence of CSRF protection in WebGoat presents a serious vulnerability. Proper implementation of anti-CSRF mechanisms is essential to prevent unauthorized actions initiated from external domains. Addressing this issue will significantly strengthen the application's security posture.
