Security Assessment Report for e-Store

Word Count: 2,885

01/04/2025

Donna Naadu Botchway
ROBERT GORDON UNIVERSITY

Table of Contents

Table of Figures

1.0. Executive Summary

This report evaluates e-store, an e-commerce website's security vulnerabilities, defences, and legal, ethical, social, and professional (LESP) implications.

The offensive security assessment identified critical vulnerabilities: SQL Injection (A03:2021), Broken Access Control (A01:2021), and Cross-Site Scripting (A07:2021), allowing unauthorized access to accounts and sensitive data (EC-Council, no date). Also, a Business Logic Flaw through Parameter Tampering caused financial fraud, and inadequate server-side sanitization (MITRE, 2024b; Feta, 2021).

The defensive measures recommended enforcing HTTPS for secure communication, and, configuring strict cookie settings to mitigate XSS and CSRF risks. Security headers were configured to block iframe embedding and prevent clickjacking (OWASP, 2023). Prepared statements, output encoding, and input sanitization strengthened defences against SQL Injection and XSS attacks.

To align with LESP standards, the e-Store must comply with regulations like GDPR and PCI DSS while emphasizing transparency, customer privacy and accessibility (BigCommerce, 2023).

In conclusion, while vulnerabilities remain, the defences taken and adherence to LESP standards will improve the e-store's security and compliance with relevant regulations.

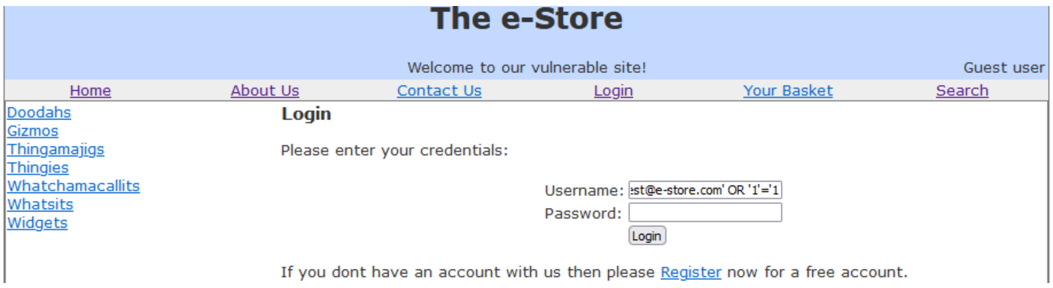
2.0. Task 1: Offensive Security

Challenge	Vulnerability Type	How It Happened	Impact	Solution
Challenge1: Login As Test	SQL Injection (A03:2021)	Malicious SQL query in login form bypassed authentication.	Unauthorized access to user accounts.	Use prepared statements, validate input.
Challenge4: Find Hidden Content	Broken Access Control (A01:2021)	Users bypassed authentication and accessed admin panel.	Exposure of privileged data.	Implement strict access control checks.
Challenge6: Level 1 XSS	Reflected XSS (A07:2021)	Malicious script injected into search field.	Site defacement.	Sanitize input, use CSP.
Challenge7: Level 2 XSS	Stored XSS (A07:2021)	Script stored via user registration form executed on login.	Site defacement.	Sanitize input before storing, use encoding.
Challenge8: Access Someone's Basket	Insecure Direct Object Reference (A01:2021)	Manipulated basket ID to access another user's basket.	Unauthorized data access.	Verify permissions, use complex IDs.
Challenge9: Get the Store to Owe	Parameter Tampering (A01:2021)	Altered item quantity to negative, causing store to owe money.	Financial loss.	Validate inputs, secure business logic.
Challenge10: Change Password via GET	Broken Access Control (A01:2021) & Sensitive Data Exposure (A04:2021)	System allowed method change from POST to GET, exposing passwords in URLs.	Sensitive data exposure.	Enforce HTTP method validation, encrypt data.
Challenge11: Conquer AES Encryption	Cross-Site Scripting (A07:2021)	Bypassed AES encryption via client-side manipulation.	Site defacement.	Sanitize inputs and outputs.
Challenge12: Conquer AES & Append Table	SQL Injection (A03:2021)	Used error detail and union query to access sensitive database info.	Exposure of database tables.	Use prepared statements, hide error details.

Figure 1: Offensive Security Test Summary

2.1. Challenge 1: Login As Test

The offensive testing revealed an SQL Injection (SQLi) in the login, allowing unauthorized access. Classified as A03:2021-Injection in OWASP's Top 10, this flaw is present in 94% of tested applications (OWASP, 2021a). The "view page source" of the login form revealed no input validation allowing malicious SQL code to manipulate authentication logic. As shown below, entering test@estore.com' OR '1'='1 as the username with the password blank introduced tautology ('1'='1), forcing authentication to always return true (EC-Council, no date).



The e-Store

Welcome to our vulnerable site! Guest user

Home | About Us | Contact Us | Login | Your Basket | Search

Doodahs
Gizmos
Thingamajigs
Thingies
Whatchamacallits
Whatsits
Widgets

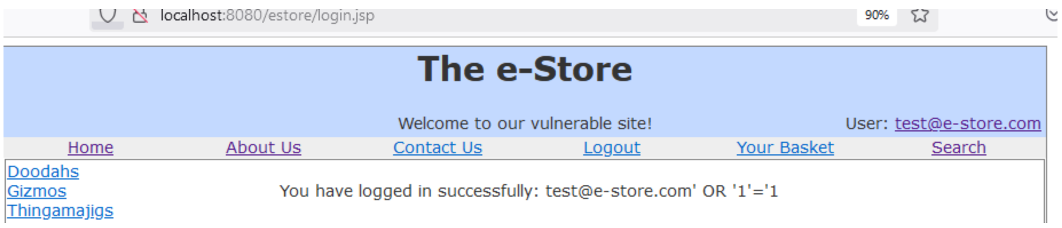
Login

Please enter your credentials:

Username: st@e-store.com' OR '1'='1
Password:
Login

If you dont have an account with us then please Register now for a free account.

Figure 2: Login as test via SQLi.



localhost:8080/estore/login.jsp 90%

The e-Store

Welcome to our vulnerable site! User: test@e-store.com

Home | About Us | Contact Us | Logout | Your Basket | Search

Doodahs
Gizmos
Thingamajigs

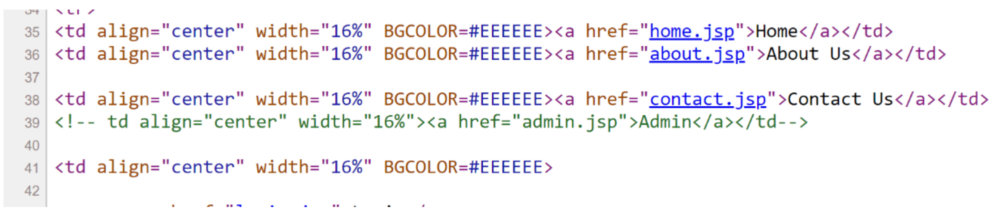
You have logged in successfully: test@e-store.com' OR '1'='1

Figure 3: Successful login as test via SQLi.

Further testing revealed that using the admin's email with same payload granted administrator access, posing a high security risk. Unauthorized access to accounts confirmed the absence of input validation and parameterized queries. To prevent exploitation, the e-Store must use prepared statements and layered input validation (PortSwigger, no date).

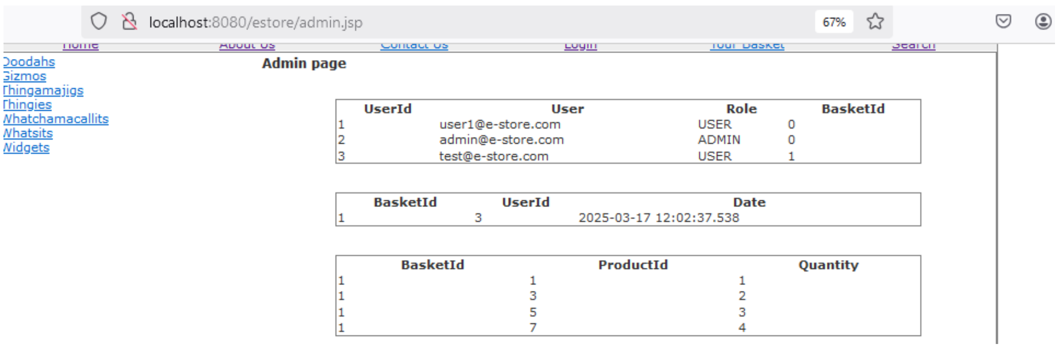
2.2. Challenge 4: Find Hidden Content

The test revealed a Broken Access Control (A01:2021) allowing users bypass restrictions and access admin interface without authentication due to improper access controls. A hidden link led to admin interface, exposing privileged data and violating the principle of least privilege (Figures 4 and 5) (OWASP, 2021b). To mitigate this, the e-Store must enforce "deny by default" access controls and validate permissions on every request (OWASP, 2021b).



```
<td align="center" width="16%" BGCOLOR=#EEEEEE><a href="home.jsp">Home</a></td>
<td align="center" width="16%" BGCOLOR=#EEEEEE><a href="about.jsp">About Us</a></td>

<td align="center" width="16%" BGCOLOR=#EEEEEE><a href="contact.jsp">Contact Us</a></td>
<!-- td align="center" width="16%"><a href="admin.jsp">Admin</a></td-->

<td align="center" width="16%" BGCOLOR=#EEEEEE>
```

Figure 4: Hidden admin page link.



localhost:8080/estore/admin.jsp

Admin page

UserId	User	Role	BasketId
1	user1@e-store.com	USER	0
2	admin@e-store.com	ADMIN	0
3	test@e-store.com	USER	1

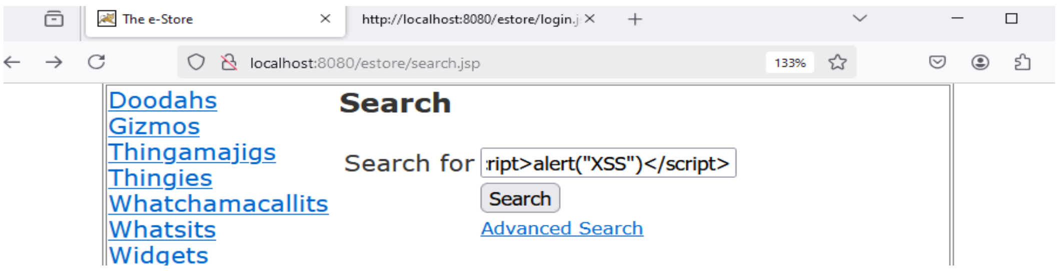
BasketId	UserId	Date
1	3	2025-03-17 12:02:37.538

BasketId	ProductId	Quantity
1	1	1
1	3	2
1	5	3
1	7	4

Figure 5: Access admin interface.

2.3. Challenge 6: Level 1 XSS

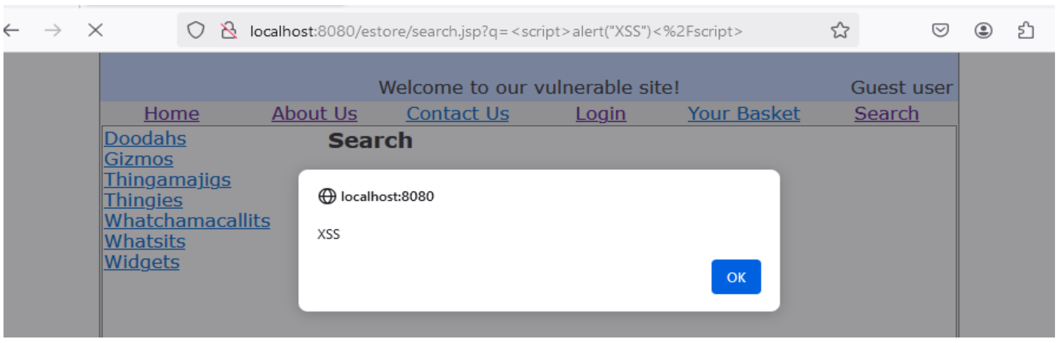
A Reflective Cross-Site Scripting (A07:2021 - XSS) in the search functionality, allow malicious scripts to execute in the users' browser (OWASP 2021c). Testing with a simple <i>hello</i> rendered italics, confirming no input sanitization. The payload (<script>alert("XSS")</script>) was then injected, triggering a popup alert (Figures 6 and 7).



localhost:8080/estore/search.jsp

Search for ript>alert("XSS")</script>

Figure 6: Level1-XSS in search page.



localhost:8080/estore/search.jsp?q=<script>alert("XSS")<%2Fscript>

localhost:8080

XSS

Figure 7 : Level1-XSS pop up.

This demonstrated arbitrary JavaScript execution resulting in site defacement. According to OWASP (2021c), prevention requires proper input sanitization, and output encoding.

2.4. Challenge 7: Level 2 XSS

A Stored Cross-Site Scripting (A07:2021 - XSS) on the registration page, allow malicious scripts to be permanently stored and executed across multiple sessions (OWASP 2021c). Figure 8 shows injecting <script>alert("XSS")</script> into the email field in a valid email format succeeded, as the input was not sanitized before being stored in the database.

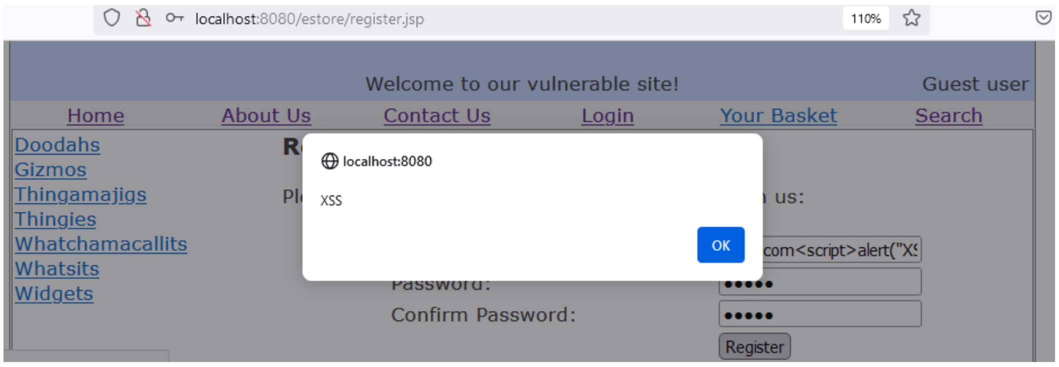
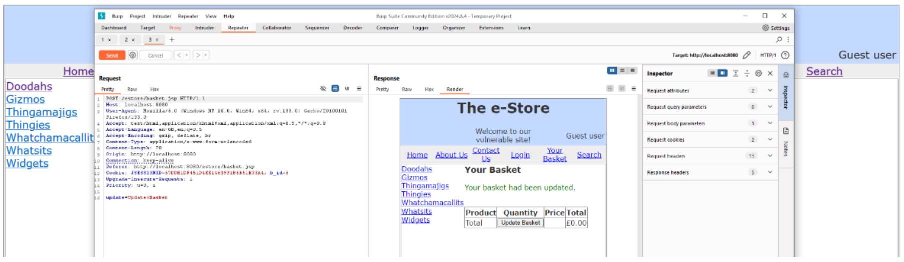


Figure 8: Level2-XSS pop up.

When the user logged in, the script executed, triggering a popup alert with every interaction. Unlike the reflective XSS, stored XSS persists, posing greater risks. Input validation is needed to prevent this(OWASP 2021c).

2.5. Challenge 8: Access someone's basket.

An Insecure Direct Object Reference (A01:2021; CWE-639) on the basket page, allow attackers to access or modify objects by manipulating unique identifiers (MITRE, 2024a). Burp Suite was used to intercept and analyse the basket ID in the cookie (Figure 9).



```
POST /estore/basket.jsp HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:133.0) Gecko/20100101 Firefox/133.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 20
Origin: http://localhost:8080
Connection: keep-alive
Referer: http://localhost:8080/estore/basket.jsp
Cookie: JSESSIONID=57DDB1DF451D4EE1469371B5E51E33A4; b_id=5
Upgrade-Insecure-Requests: 1
Priority: u=0, i

update=Update+Basket
```

Figure 9: Capture basket page traffic via burp suite.

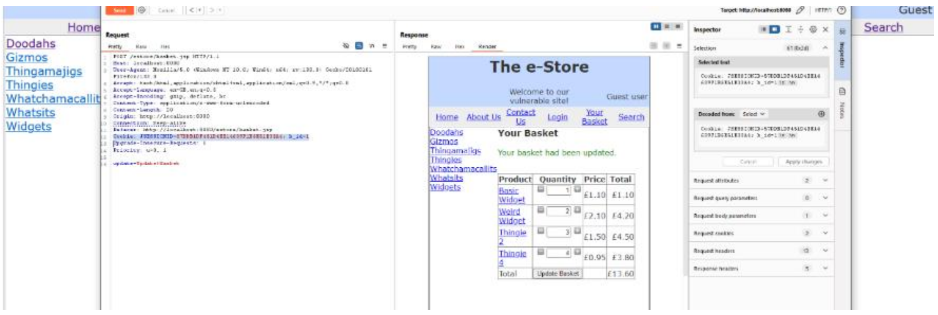
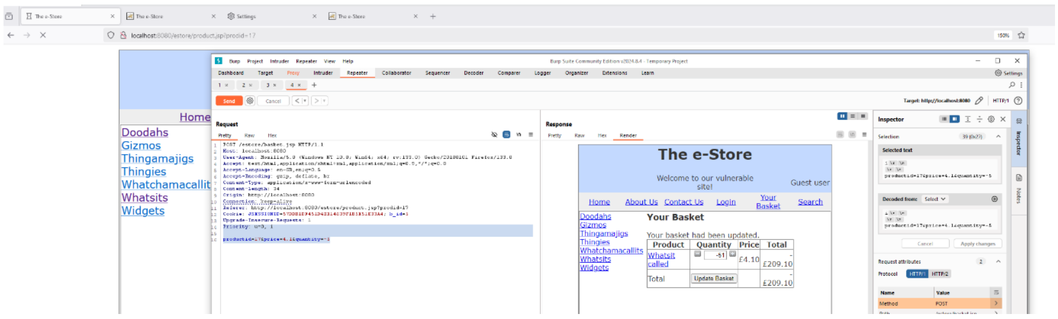


Figure 10: Successfully accessed another's basket.

Finding basket IDs from the admin page without authorization helped exploitation. By modifying the basket ID, attackers can access and alter another user's basket (Figure 10). To mitigate this, the e-Store must verify user permissions for every request and use complex, unpredictable identifiers to obscure resource references (GeeksforGeeks, 2023).

2.6. Challenge 9: Get the store to owe.

Parameter tampering flaw (A01:2021) in the basket functionality, allow attackers to manipulate business logic workflow (MITRE, 2024b; Feta, 2021). Using Burp Suite, traffic was intercepted, and the quantity was changed to a negative value (Figure 11), leading to potential monetary loss.


Burp Suite Community Edition v2024.8.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Send Cancel
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render
Inspector

```
POST /estore/basket.jsp HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:133.0) Gecko/20100101 Firefox/133.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 34
Origin: http://localhost:8080
Connection: keep-alive
Referer: http://localhost:8080/estore/product.jsp?prodid=17
Cookie: JSESSIONID=57DDB1DF451D4331483F1B5E51E33A4; b_id=1
Upgrade-Insecure-Requests: 1
Priority: u=0, i

productid=17&price=4.1&quantity=-5
```

The e-Store

Welcome to our vulnerable site! Guest user

Home About Us Contact Us Login Your Basket Search

Doodahs
Gizmos
Thingamajigs
Thingies
Whatchamacallits
Whatsits
Widgets

Your Basket

Your basket had been updated.

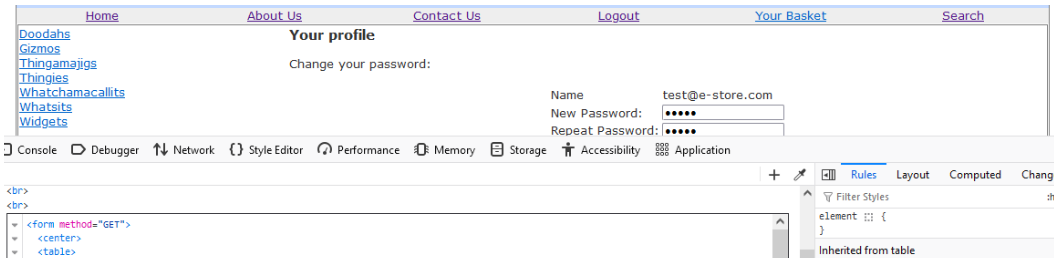
Product	Quantity	Price	Total
Whatsit called	-51	£4.10	-£209.10
Total	Update Basket		-£209.10

Figure 11:Store owes user due to negative quantity.

The flaw arose from inadequate consideration of potential threats and misuse cases in business logic design. Prevention requires, a thorough understanding of the business processes and regular threat modelling to cover all exploit scenarios (OWASP, no date).

2.7. Challenge 10: Change password via GET.

Also, password change via GET method was possible, using Developer Tools, as shown in Figure 13. This modification exposes the password in the URL (Figure 14).


Home About Us Contact Us Logout Your Basket Search
Doodahs
Gizmos
Thingamajigs
Thingies
Whatchamacallits
Whatsits
Widgets
Console Debugger Network Style Editor Performance Memory Storage Accessibility Application
Rules Layout Computed Chang
Filter Styles
Inherited from table

Your profile

Change your password:

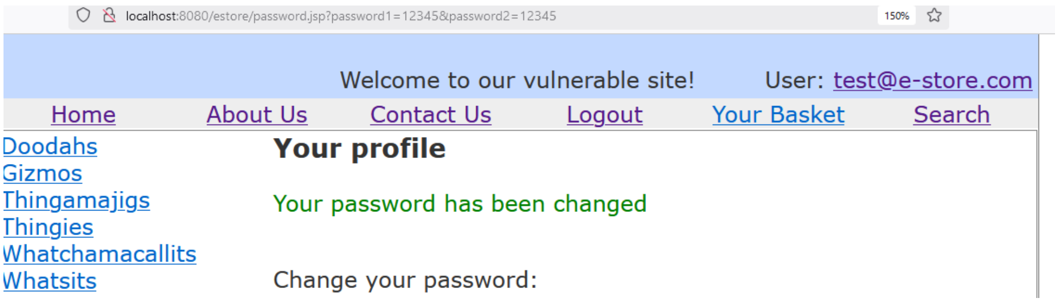
Name	test@e-store.com
New Password:	•••••
Repeat Password:	•••••

```
<br>
<br>
<form method="GET">
  <center>
  <table>
```

```
element {
}
```

Figure 12:Change POST to GET

According to CWE-598, passwords in URLs create a Sensitive Data Exposure risk, as they can be intercepted or stored insecurely in logs or browser history (MITRE, 2024c).



localhost:8080/estore/password.jsp?password1=12345&password2=12345

Welcome to our vulnerable site! User: test@e-store.com

Your profile

Your password has been changed

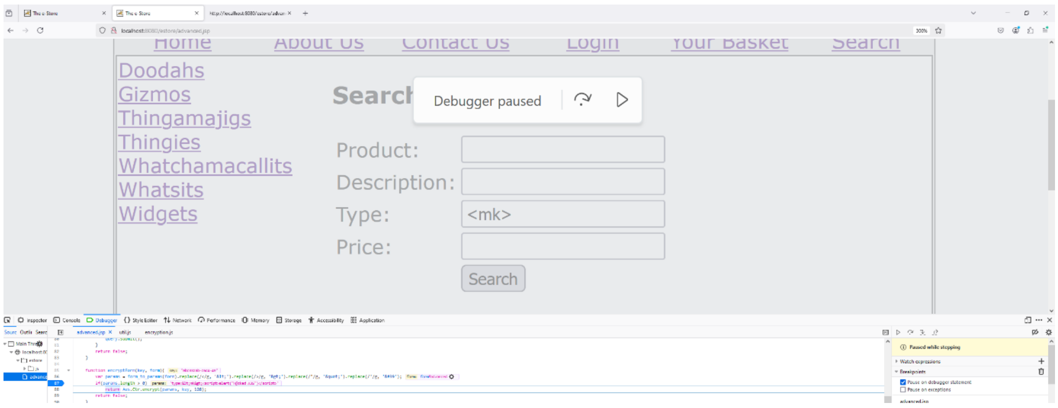
Change your password:

Figure 13: Password changed via GET.

To mitigate this, server must enforce secure methods such as POST, with validation to ensure requests are untampered (TuringSecure, no date). Multi-Factor Authentication(MFA), such as OTP or an email link, should be implemented for password changes.

2.8. Challenge 11: Conquer AES encryption and display a popup.

The "Advanced Search" page had applied HTML encoding to sanitize special characters, followed by AES encryption. However, by setting a breakpoint after the HTML encoding and modifying the params value in the console, the sanitization was bypassed as shown below.



Search

Product:

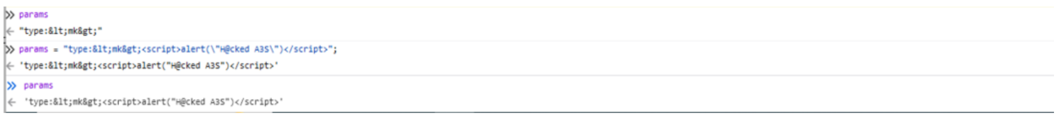
Description:

Type: <mk>

Price:

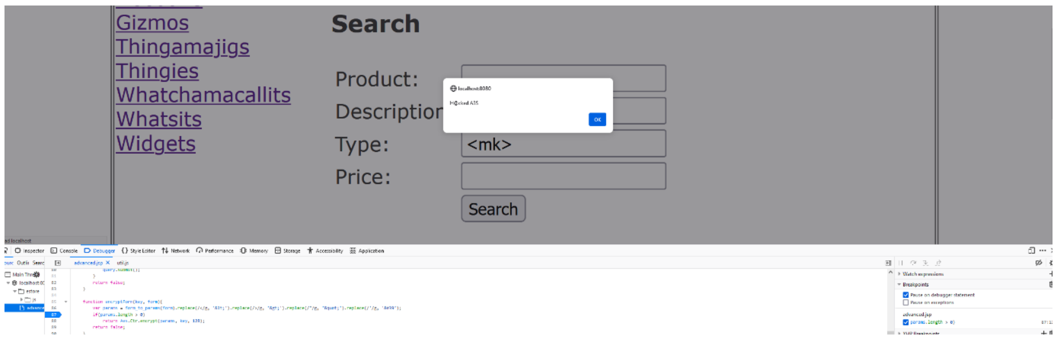
```
        }
        return false;
    }

    function encryptForm(key, form){
        var params = form_to_params(form).replace(/</g, '&lt;').replace(/>/g, '&gt;').replace(/"/g, '&quot;').replace(/'/g, '&#39');
        if(params.length > 0)
            return Aes.Ctr.encrypt(params, key, 128);
        return false;
    }
```

Figure 14: Introduce a breakpoint.



```
>> params
← "type:&lt;mk&gt;"
>> params = "type:&lt;mk&gt;<script>alert(\"H@cked A3S\")</script>";
← 'type:&lt;mk&gt;<script>alert("H@cked A3S")</script>'
>> params
← 'type:&lt;mk&gt;<script>alert("H@cked A3S")</script>'
```

Figure 15: AES-payload.



Search

Product:

Description

Type: <mk>

Price:

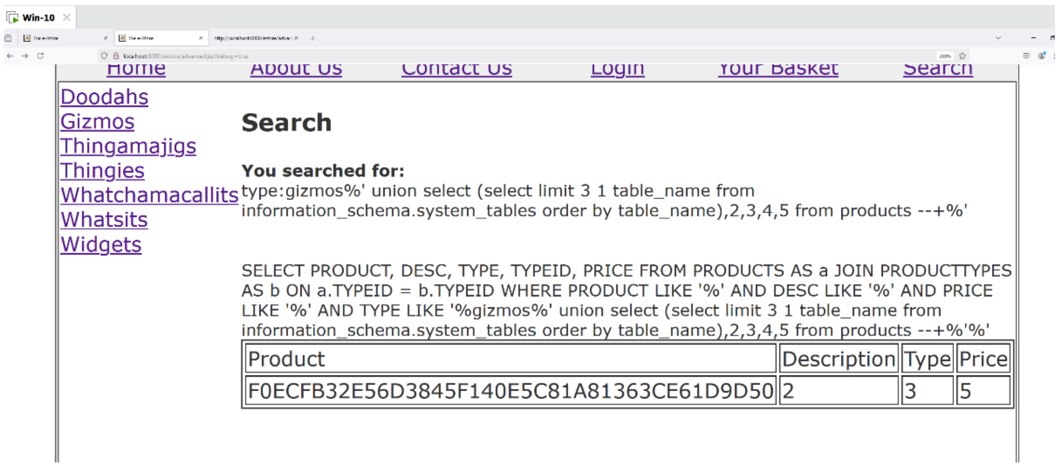
H@cked A3S

Figure 16: AES conquered and XSS successful.

The payload in Figure 15 successfully executed confirming that while client-side input sanitization was applied, it was not adequately enforced on the server side. To mitigate this, both input validation and output encoding should be strictly implemented on the server side (OWASP, 2021c).

2.9. Challenge 12: Conquer AES encryption and append a list of the table names.

Exploiting improper handling of user input, an SQLi was successful by modifying the params value to include a union SQL query. It fetched table names from the database and appended them to the existing view shown in Figure 17 (EC-Council, no date).



Search

You searched for:
type:gizmos%' union select (select limit 3 1 table_name from information_schema.system_tables order by table_name),2,3,4,5 from products --+%'

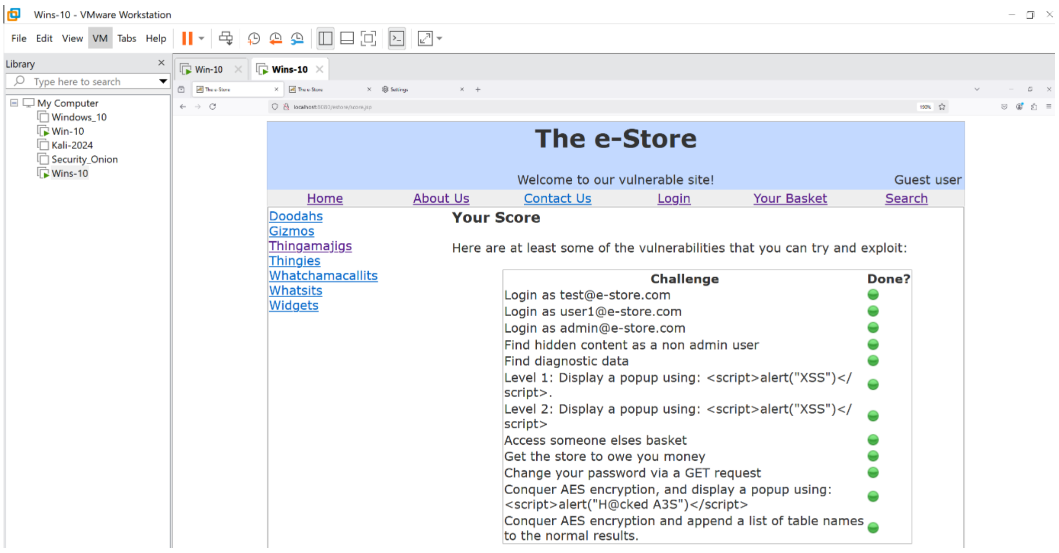
```
SELECT PRODUCT, DESC, TYPE, TYPEID, PRICE FROM PRODUCTS AS a JOIN PRODUCTTYPES AS b ON a.TYPEID = b.TYPEID WHERE PRODUCT LIKE '%' AND DESC LIKE '%' AND PRICE LIKE '%' AND TYPE LIKE '%gizmos%' union select (select limit 3 1 table_name from information_schema.system_tables order by table_name),2,3,4,5 from products --+%'%'
```

Product	Description	Type	Price
F0ECFB32E56D3845F140E5C81A81363CE61D9D50	2	3	5

Figure 17: Append table name.

This allowed the attacker to extract sensitive database information due the application directly incorporated user input into SQL queries without using prepared statements. The recommended mitigation is to implement parameterized queries and disable detailed error messages (OWASP, 2021a).

Here is the video demonstration:[Link to video]



Your Score

Here are at least some of the vulnerabilities that you can try and exploit:

Challenge	Done?
Login as test@e-store.com	●
Login as user1@e-store.com	●
Login as admin@e-store.com	●
Find hidden content as a non admin user	●
Find diagnostic data	●
Level 1: Display a popup using: <script>alert("XSS")</script>.	●
Level 2: Display a popup using: <script>alert("XSS")</script>	●
Access someone elses basket	●
Get the store to owe you money	●
Change your password via a GET request	●
Conquer AES encryption, and display a popup using: <script>alert("H@cked A3S")</script>	●
Conquer AES encryption and append a list of table names to the normal results.	●

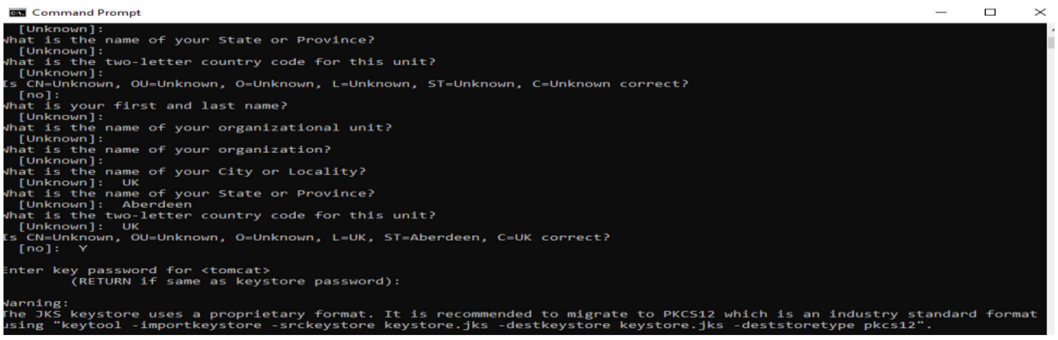
Figure 18: Completed ScoreBoard

3.0. Task 2: Defensive Security

3.1.0. Tomcat Security

3.1.1. Security measure 1: Enforcing HTTPS for Secure Communication

Encrypting data in transit prevents MITM attacks and protects user credentials. Without HTTPS, sensitive data can be intercepted and modified, while browsers flag HTTP sites as "Not Secure," reducing trust (Apache Tomcat, 2025).

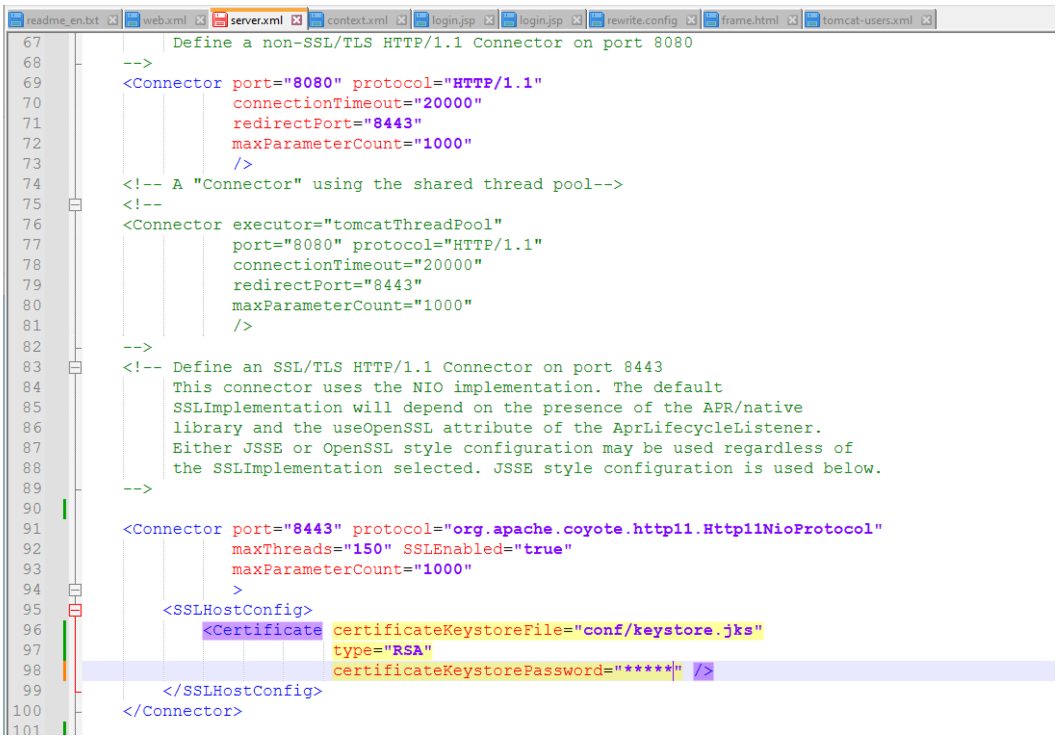


```
Command Prompt
  [Unknown]:
What is the name of your State or Province?
  [Unknown]:
What is the two-letter country code for this unit?
  [Unknown]:
Is CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown correct?
  [no]:
What is your first and last name?
  [Unknown]:
What is the name of your organizational unit?
  [Unknown]:
What is the name of your organization?
  [Unknown]:
What is the name of your City or Locality?
  [Unknown]:  UK
What is the name of your State or Province?
  [Unknown]:  Aberdeen
What is the two-letter country code for this unit?
  [Unknown]:  UK
Is CN=Unknown, OU=Unknown, O=Unknown, L=UK, ST=Aberdeen, C=UK correct?
  [no]:  Y

Enter key password for <tomcat>
        (RETURN if same as keystore password):

Warning:
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is an industry standard format
using "keytool -importkeystore -srckeystore keystore.jks -destkeystore keystore.jks -deststoretype pkcs12".
```

Figure 19:SSL-Certificate via Keytool

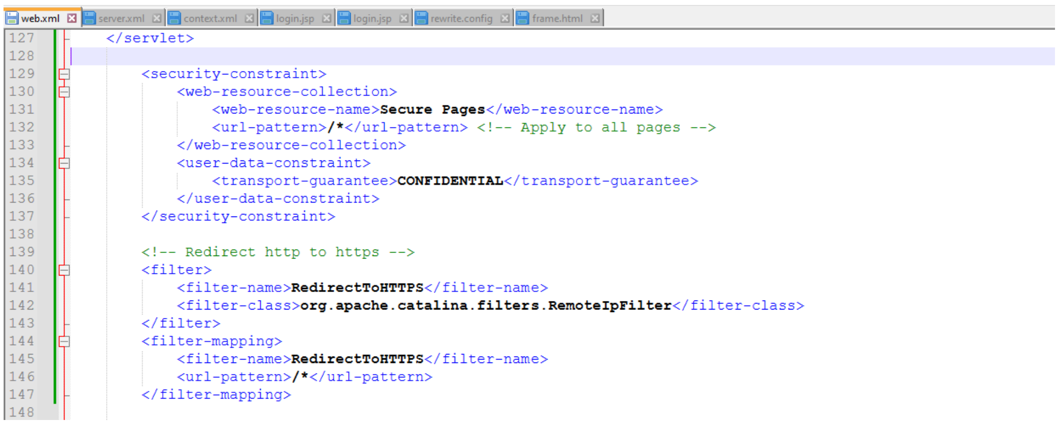
An SSL certificate was generated using Keytool (Figure19). A connector in server.xml configured HTTPS on port 8443 (Figure20), and a security constraint in web.xml redirected HTTP traffic from port 8080 to 8443 with CONFIDENTIAL transport guarantee (Figure21; Apache Tomcat, 2025).

After implementation, accessing the e-store via http://localhost:8080 automatically redirected to http://localhost:8443, verified using the Kali terminal (Figure22).As this uses a self-signed certificate, browsers issued a warning. For production, it is advisable to use a trusted Certificate Authority(CA) like Let's Encrypt, DigiCert, or GlobalSign.



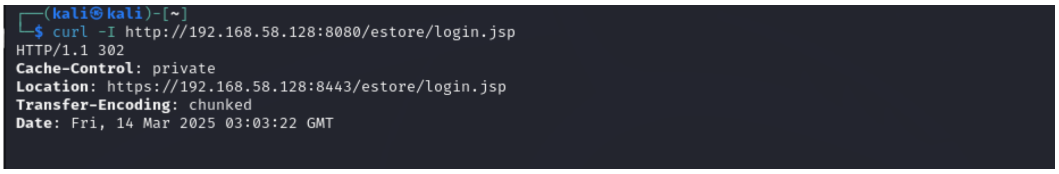
```
         Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
    -->
    <Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443"
               maxParameterCount="1000"
               />
    <!-- A "Connector" using the shared thread pool-->
    <!--
    <Connector executor="tomcatThreadPool"
               port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443"
               maxParameterCount="1000"
               />
    -->
    <!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443
         This connector uses the NIO implementation. The default
         SSLImplementation will depend on the presence of the APR/native
         library and the useOpenSSL attribute of the AprLifecycleListener.
         Either JSSE or OpenSSL style configuration may be used regardless of
         the SSLImplementation selected. JSSE style configuration is used below.
    -->

    <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
               maxThreads="150" SSLEnabled="true"
               maxParameterCount="1000"
               >
        <SSLHostConfig>
            <Certificate certificateKeystoreFile="conf/keystore.jks"
                         type="RSA"
                         certificateKeystorePassword="*****" />
        </SSLHostConfig>
    </Connector>
```

Figure 20: Server.xml-Configure HTTPS port 8443.



```
    </servlet>

        <security-constraint>
            <web-resource-collection>
                <web-resource-name>Secure Pages</web-resource-name>
                <url-pattern>/*</url-pattern> <!-- Apply to all pages -->
            </web-resource-collection>
            <user-data-constraint>
                <transport-guarantee>CONFIDENTIAL</transport-guarantee>
            </user-data-constraint>
        </security-constraint>

        <!-- Redirect http to https -->
        <filter>
            <filter-name>RedirectToHTTPS</filter-name>
            <filter-class>org.apache.catalina.filters.RemoteIpFilter</filter-class>
        </filter>
        <filter-mapping>
            <filter-name>RedirectToHTTPS</filter-name>
            <url-pattern>/*</url-pattern>
        </filter-mapping>
```

Figure 21: Web.xml-Configure HTTP Redirect

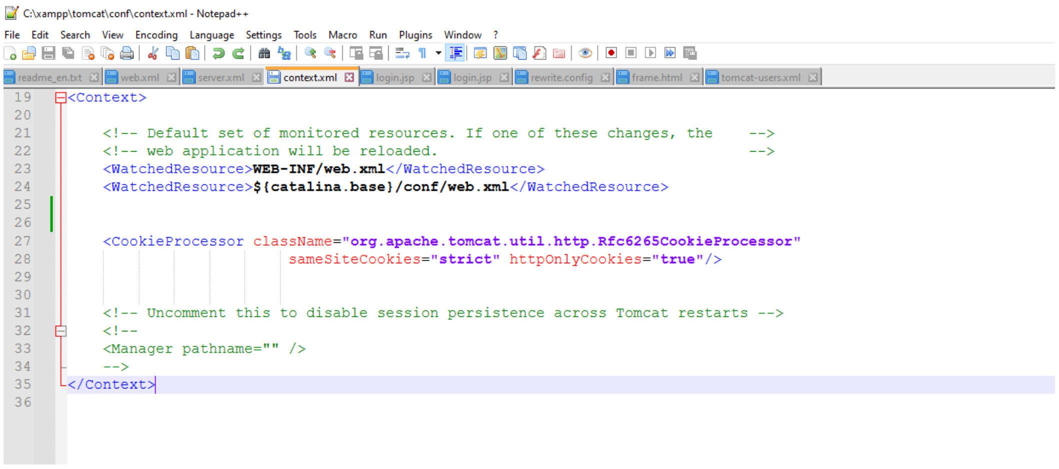


```
┌──(kali㉿kali)-[~]
└─$ curl -I http://192.168.58.128:8080/estore/login.jsp
HTTP/1.1 302
Cache-Control: private
Location: https://192.168.58.128:8443/estore/login.jsp
Transfer-Encoding: chunked
Date: Fri, 14 Mar 2025 03:03:22 GMT
```

Figure 22: Kali showing HTTP-redirect to HTTPS.

3.1.2. Security measure 2: Securing Cookies and HTTP Headers

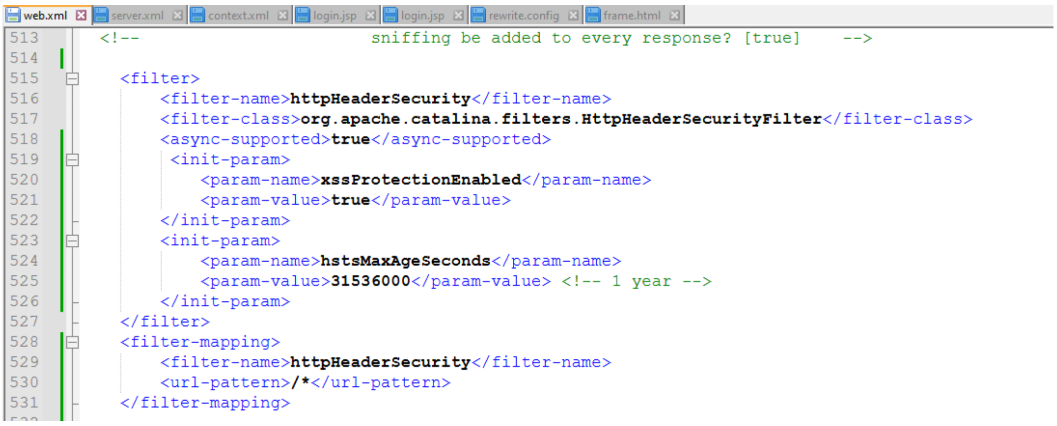
Cookies and security headers protect user sessions from XSS, CSRF, and clickjacking attacks. Without them, attackers can hijack sessions, inject malicious scripts, or trick users into performing unintended actions. According to OWASP, Cookie attributes can limit the impact of an XSS attack. Seen in Figure23, the context.xml was configured with CookieProcessor with sameSiteCookies="strict" and httpOnlyCookies="true" to prevent cookies access by scripts or block cross-site requests (Apache Tomcat, 2025).


C:\xampp\tomcat\conf\context.xml - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
readme_en.txt
web.xml
server.xml
context.xml
login.jsp
login.jsp
rewrite.config
frame.html
tomcat-users.xml

```
<Context>

    <!-- Default set of monitored resources. If one of these changes, the    -->
    <!-- web application will be reloaded.                                   -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>
    <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>


    <CookieProcessor className="org.apache.tomcat.util.http.Rfc6265CookieProcessor"
                     sameSiteCookies="strict" httpOnlyCookies="true"/>


    <!-- Uncomment this to disable session persistence across Tomcat restarts -->
    <!--
    <Manager pathname="" />
    -->
</Context>
```

Figure 23: Context.xml-Configure CookieProcessor

In Figure 24, web.xml was updated with the HttpHeaderSecurityFilter to enforce Strict-Transport-Security (HSTS), and X-Content-Type-Options headers, to force HTTPS, prevent MIME-type sniffing and block web-based attacks (Apache Tomcat, 2025; Mozilla Developer Network, no date).

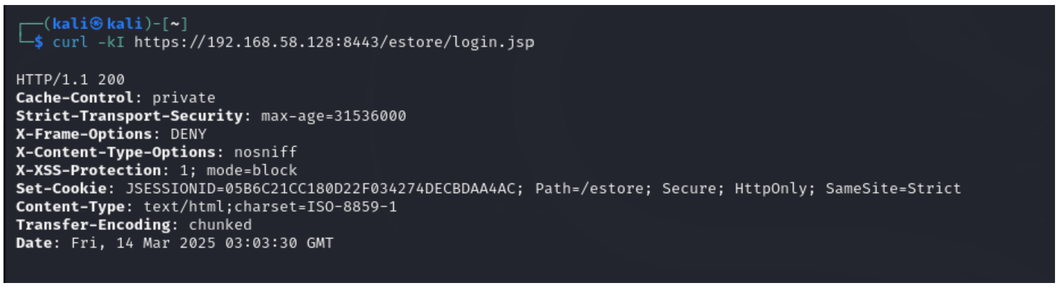


```xml
<!--                          sniffing be added to every response? [true]    -->

  <filter>
      <filter-name>httpHeaderSecurity</filter-name>
      <filter-class>org.apache.catalina.filters.HttpHeaderSecurityFilter</filter-class>
      <async-supported>true</async-supported>
       <init-param>
          <param-name>xssProtectionEnabled</param-name>
          <param-value>true</param-value>
      </init-param>
      <init-param>
          <param-name>hstsMaxAgeSeconds</param-name>
          <param-value>31536000</param-value> <!-- 1 year -->
      </init-param>
  </filter>
  <filter-mapping>
      <filter-name>httpHeaderSecurity</filter-name>
      <url-pattern>/*</url-pattern>
  </filter-mapping>
```

Figure 24: Web.xml-Configure Headers

Since the e-store currently does not use external logins or payment gateways,SameSite=Strict is used. However, if OAuth or payment integrations are added, Lax would be needed to allow secure cross-site requests (Mozilla Developer Network, no date).

Testing (Figure 25), confirmed that headers were correctly sent and cookies restricted as configured, securing user sessions. Also, attempts to embed the login page in an iframe were blocked due to the X-Frame-Options:DENY header, confirming effective clickjacking protection.

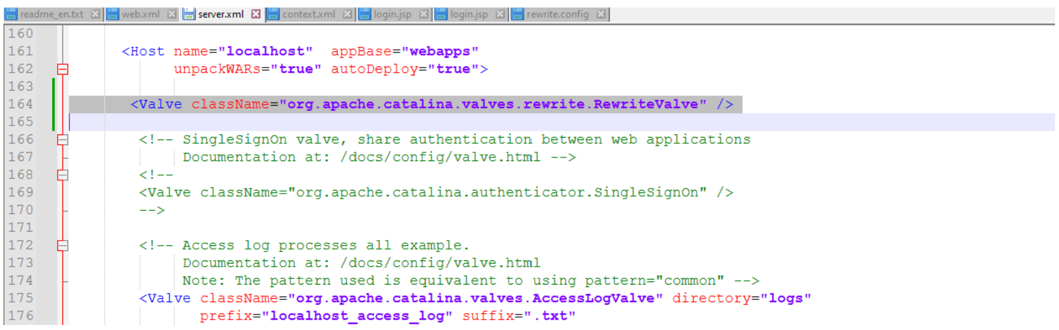


```
┌──(kali㉿kali)-[~]
└─$ curl -kI https://192.168.58.128:8443/estore/login.jsp

HTTP/1.1 200
Cache-Control: private
Strict-Transport-Security: max-age=31536000
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Set-Cookie: JSESSIONID=05B6C21CC180D22F034274DECBDAA4AC; Path=/estore; Secure; HttpOnly; SameSite=Strict
Content-Type: text/html;charset=ISO-8859-1
Transfer-Encoding: chunked
Date: Fri, 14 Mar 2025 03:03:30 GMT
```

Figure 25: Kali showing headers.

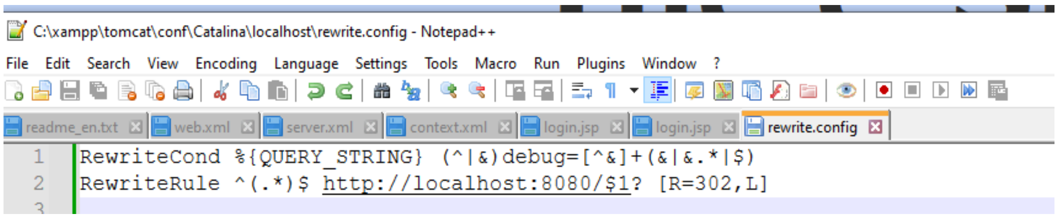
3.1.3. Security measure 3: Prevent Debug Page Access

Restricting access to debug pages is needed to prevent exposing sensitive debug information, which attackers could exploit. Figures below, show the RewriteValve configured in server.xml, and a rewrite.config file created in the tomcat\conf\Catalina\localhost directory to block requests containing the debug=true query parameter (Apache Tomcat, 2025; Apache HTTP Server, no date). The file was saved as "All Files" to avoid a .txt extension.



```xml

      <Host name="localhost"  appBase="webapps"
            unpackWARs="true" autoDeploy="true">

       <Valve className="org.apache.catalina.valves.rewrite.RewriteValve" />

        <!-- SingleSignOn valve, share authentication between web applications
             Documentation at: /docs/config/valve.html -->
        <!--
        <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
        -->

        <!-- Access log processes all example.
             Documentation at: /docs/config/valve.html
             Note: The pattern used is equivalent to using pattern="common" -->
        <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
               prefix="localhost_access_log" suffix=".txt"
```

Figure 26: Sever.xml-Configure RewriteValve

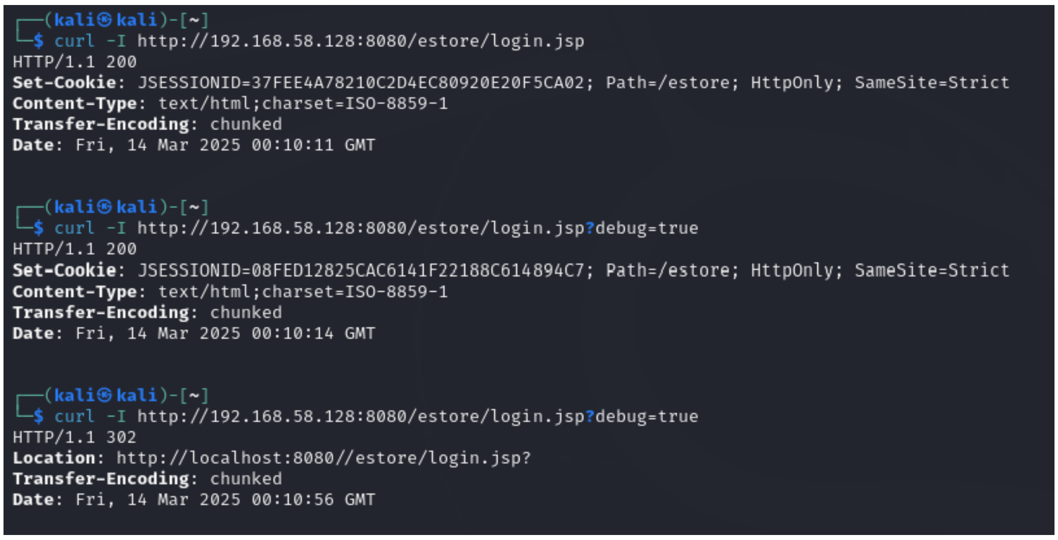


```
RewriteCond %{QUERY_STRING} (^|&)debug=[^&]+(&|&.*|$)
RewriteRule ^(.*)$ http://localhost:8080/$1? [R=302,L]
```

Figure 27: Rewrite.config file.

The configuration ensures that requests with debug=true are redirected to the base URL, even if there are additional parameters, preventing access to debug pages.

Testing in Kali confirmed the restriction, ensuring debug pages remain secure and reducing the risk of exploitation (Figure28).



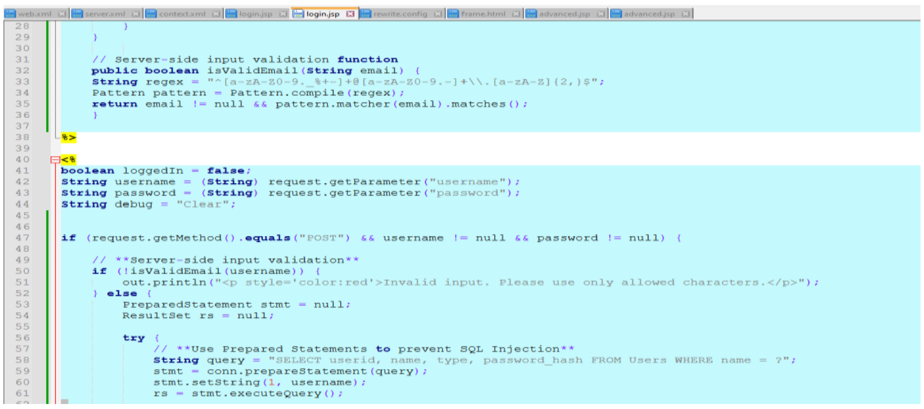
```
┌──(kali㉿kali)-[~]
└─$ curl -I http://192.168.58.128:8080/estore/login.jsp
HTTP/1.1 200
Set-Cookie: JSESSIONID=37FEE4A78210C2D4EC80920E20F5CA02; Path=/estore; HttpOnly; SameSite=Strict
Content-Type: text/html;charset=ISO-8859-1
Transfer-Encoding: chunked
Date: Fri, 14 Mar 2025 00:10:11 GMT


┌──(kali㉿kali)-[~]
└─$ curl -I http://192.168.58.128:8080/estore/login.jsp?debug=true
HTTP/1.1 200
Set-Cookie: JSESSIONID=08FED12825CAC6141F22188C614894C7; Path=/estore; HttpOnly; SameSite=Strict
Content-Type: text/html;charset=ISO-8859-1
Transfer-Encoding: chunked
Date: Fri, 14 Mar 2025 00:10:14 GMT


┌──(kali㉿kali)-[~]
└─$ curl -I http://192.168.58.128:8080/estore/login.jsp?debug=true
HTTP/1.1 302
Location: http://localhost:8080//estore/login.jsp?
Transfer-Encoding: chunked
Date: Fri, 14 Mar 2025 00:10:56 GMT
```

Figure 28: Kali confirms debug-restriction.

3.2.0. Source code security.

3.2.1. SQL-Injection Mitigation

As recommended by OWASP, preventing SQLi requires input validation and prepared statements. Without these, attackers can manipulate queries by injecting malicious code, risking unauthorized access or data loss (eSecurity Planet, 2023).

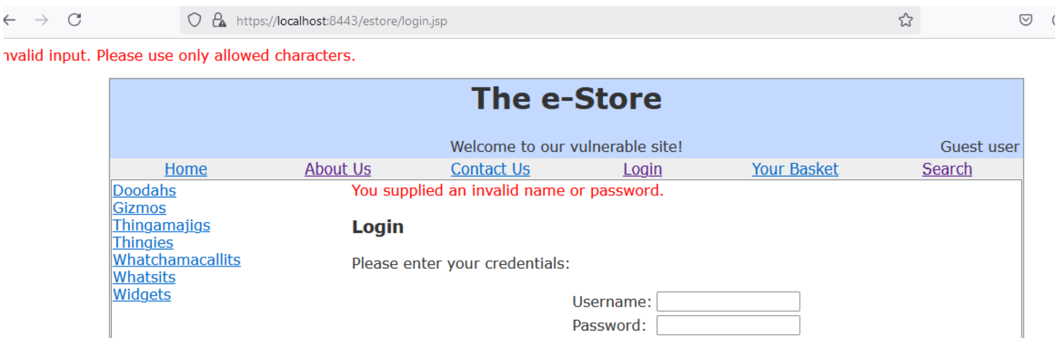


```
            }
        }

        // Server-side input validation function
        public boolean isValidEmail(String email) {
        String regex = "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$";
        Pattern pattern = Pattern.compile(regex);
        return email != null && pattern.matcher(email).matches();
        }

%>

<%
boolean loggedIn = false;
String username = (String) request.getParameter("username");
String password = (String) request.getParameter("password");
String debug = "Clear";


if (request.getMethod().equals("POST") && username != null && password != null) {

        // **Server-side input validation**
        if (!isValidEmail(username)) {
            out.println("<p style='color:red'>Invalid input. Please use only allowed characters.</p>");
        } else {
            PreparedStatement stmt = null;
            ResultSet rs = null;

            try {
                // **Use Prepared Statements to prevent SQL Injection**
                String query = "SELECT userid, name, type, password_hash FROM Users WHERE name = ?";
                stmt = conn.prepareStatement(query);
                stmt.setString(1, username);
                rs = stmt.executeQuery();
```

Figure 29: SQLi Code Mitigation

To address the vulnerability in the login.jsp, server-side validation and prepared statement was implemented. Instead of directly inserting user inputs into queries, placeholders were used to securely bind values (Figure29). This ensures that inputs are treated as data rather than part of the query, preventing query manipulation.

Testing with the same payload from Challenge1, confirmed that the login page now blocks SQLi attempts. This prevents attackers from bypassing authentication and accessing sensitive data (EC-Council, no date).



https://localhost:8443/estore/login.jsp

nvalid input. Please use only allowed characters.

The e-Store

Welcome to our vulnerable site! Guest user

Home | About Us | Contact Us | Login | Your Basket | Search

Doodahs
Gizmos
Thingamajigs
Thingies
Whatchamacallits
Whatsits
Widgets

You supplied an invalid name or password.

Login

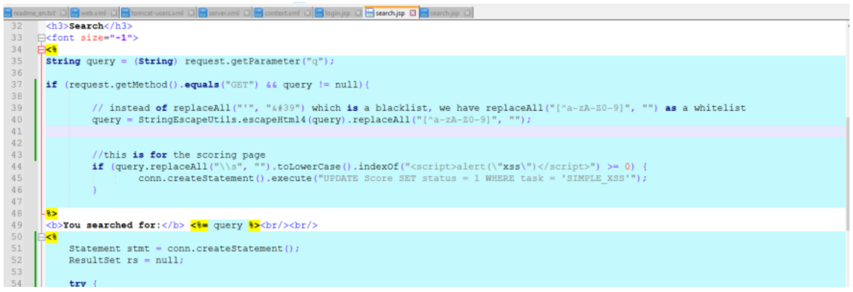
Please enter your credentials:

Username:
Password:

Figure 30: SQLi Prevented

3.2.2. XSS Mitigation

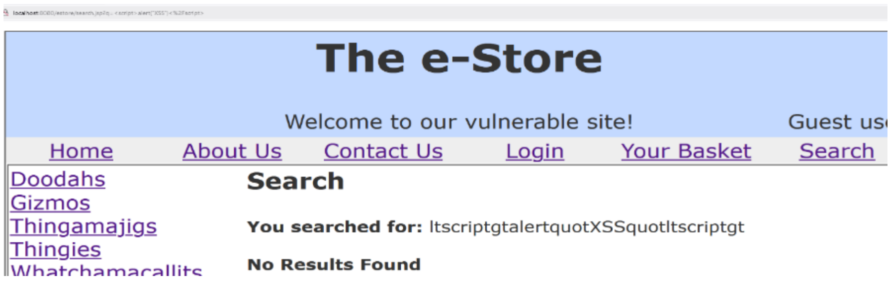
XSS in e-Store's search and register pages due to improper input handling, allows attackers to inject malicious scripts for site defacement or data theft. As recommended by OWASP, preventing XSS requires input validation and output encoding. Implementing input sanitization, an allowlist was enforced over a deny list,permitting only alphanumeric characters and blocking harmful scripts at submission (Figure 31).



```
<h3>Search</h3>
<font size="-1">
<%
String query = (String) request.getParameter("q");

if (request.getMethod().equals("GET") && query != null){

        // instead of replaceAll("'", "&#39") which is a blacklist, we have replaceAll("[^a-zA-Z0-9]", "") as a whitelist
        query = StringEscapeUtils.escapeHtml4(query).replaceAll("[^a-zA-Z0-9]", "");


        //this is for the scoring page
        if (query.replaceAll("\\s", "").toLowerCase().indexOf("<script>alert(\"xss\")</script>") >= 0) {
                conn.createStatement().execute("UPDATE Score SET status = 1 WHERE task = 'SIMPLE_XSS'");
        }

%>
<b>You searched for:</b> <%= query %><br/><br/>
<%
    Statement stmt = conn.createStatement();
    ResultSet rs = null;

    try {
```

Figure 31: XSS Code Mitigation

Also, output encoding using StringEscapeUtils.escapeHtml4() was applied. This converts special characters like < and > into plain text rather than being executed as code. Any attempt to inject scripts is neutralized. Testing with the same XSS payload from challenge 6, confirmed that the search function safely handles user input (Figure 32). These security measures prevent injection, protecting users from potential exploits.



localhost:8080/estore/search.jsp?q...<script>alert("XSS")<%2Fscript>

The e-Store

Welcome to our vulnerable site! Guest us

Home | About Us | Contact Us | Login | Your Basket | Search

Doodahs
Gizmos
Thingamajigs
Thingies
Whatchamacallits

Search

You searched for: ltscriptgtalertquotXSSquotltscriptgt

No Results Found

Figure 32: XSS Prevented

4.0. Task 3: Legal, Ethical, Social and Professional (LESP) Issues

4.1.0. Legal Issues

4.1.1. Data Protection and Privacy

The e-Store must follow data protection laws requiring security measures that protect user data, like the General Data Protection Regulation (GDPR, EU) or the California Consumer Privacy Act (CCPA, US). That is obtaining user consent for data collection and providing users with the right to access and delete their data (Shopify, 2023; BigCommerce, 2023). Meta faced a $1.3 billion fine (2023) and Amazon an $877 million fine (2021) for GDPR violations related to customer data transfers and privacy (Hill and Sharma, 2025). These cases reveal the legal and financial impact of violating data protection regulations.

The e-Store's security flaws, like SQL injection and an unprotected admin page, risk data breaches and violate GDPR's security-by-design principle. These flaws could result in similar hefty fines and reputational damage if not addressed. (California Attorney General, 2023; GDPR.eu, 2023). Also, consumer protection laws like the Consumer Rights Act require clear product information and transparent refund policies (FSB, 2023; LegalGPS, 2023). The flaws expose customers to identity theft violating these requirements. The e-Store must ensure transparency regarding customer data handling to be compliant.

4.1.2. Payment Security & PCI DSS Compliance

The Payment Card Industry Data Security Standard (PCI DSS) requires secure handling of payment transactions to protect customer data (BigCommerce, 2023). In 2008, Heartland Payment Systems suffered a loss of $150 million because of poor security practices, exposing millions of credit card numbers via SQL injection (Secureworks, 2012). This incident shows that compliance alone is not enough without specific-business implementation.

The e-Store's flaws like debug code, insecure object references, and SQL injections risk exposing payment data, violating PCI DSS and standards like NIST CSF and ISO 27001. These flaws increase the likelihood of unauthorized access and legal consequences. The e-Store must implement secure payment processing, require strong passwords, and comply with PCI DSS to protect customer data.

4.2.0. Ethical Issues

4.2.1. Customer Privacy and Data Security

The e-Store must prioritize customer privacy by securing sensitive data, such as personal and payment information. Customers are exposed to significant risks by vulnerabilities like account takeovers (OWASP, 2023). Illustrating the result of negligence, the 2018 British Airways data breach compromised 400,000 customers' financial data due to inadequate security (Tidy, 2020). The e-Store can protect user trust and avoid such breaches by implementing measures like input validation and secure authentication(eCommerce Fastlane, 2023).

4.2.2. Transparency, Fair Trade and Responsible Disclosure

The e-Store must prioritize transparency and fair trade to build customer trust and protect its reputation. Addressing security flaws, such as an unprotected admin page, promptly is needed, as undisclosed vulnerabilities compromise transparency and violates responsible disclosure principles (ISO/IEC 29147) (Hinrich Foundation, 2023). In January 2025, UK lawmakers criticized Shein for evading questions about its cotton supply chain amid concerns of forced labour involving China's Uyghur minority, highlighting the consequences of lacking transparency (Hui, 2025).

Fair trade practices require sourcing from suppliers who respect workers' rights and ensure safe working conditions (Airboxr, 2023), avoiding forced labor connections as like Shein's case. Additionally, providing clear and honest product descriptions and pricing to foster customer trust and responsible business practices (BigCommerce, 2023; Hinrich Foundation, 2023).

4.3.0. Social Issues

4.3.1. Security Awareness

The e-Store must educate users on safe online practices, such as recognizing phishing attempts and using multi-factor authentication (MFA) (eCommerce Consulting Firm, 2023). Security notifications reinforce customer confidence by ensuring users feel safe and valued. Amazon offers real-time alerts for suspicious activities, while Shopify provides phishing protection guidance and MFA recommendations to enhance account security (Amazon Help, no date; Shopify Help Center, no date). Prioritizing inclusivity through Web Content Accessibility Guidelines (WCAG) compliance ensures access for users with disabilities and prevents risks like frauds or data theft (eCommerce Consulting Firm, 2023).

4.3.2. Sustainability

The e-Store should reduce its carbon footprint through eco-friendly packaging and optimized logistics (Creatuity, 2023). It should prioritize ethical sourcing by selecting sustainable suppliers (eCommerce Consulting Firm, 2023). Platforms like Temu have faced backlash for unsafe products, environmental concerns, and exploitative labor practices, while brands like Allbirds set positive examples with natural materials and transparent supply chains (Business & Human Rights Resource Centre, 2024; Webiators, 2023). Partnering with non-profits and supporting social initiatives further enhances the e-Store's community impact and reputation (Hinrich Foundation, 2023).

Social responsibility is no longer just a nice-to-have; it is a must-have for eCommerce companies like the e-Store to align its operations with values that positively impact society while achieving profits (Zen Media, 2023).

4.4.0. Professional Issues

4.4.1. Secure Software Development and Accessibility

The e-Store must follow ISO/IEC 27001 and OWASP Top10 standards for secure software development. These standards emphasize secure coding, input validation, and regular vulnerability testing to prevent SQL injection and XSS flaws (BigCommerce, 2023). Neglecting these measures leads to breaches like the 2018 Ticketmaster incident. A third-party vulnerability exposed customer data, showing the risks of inadequate security practices (Twingate, 2024).

These standards also support accessibility features like screen readers. XSS vulnerabilities can affect these features and put users with disabilities at risk. Companies like Amazon adhere to WCAG guidelines and offer accessible features like voice-controlled shopping via Alexa (Amazon Help, no date; W3C, 2023). The e-Store can build trust, inclusivity, and a safe shopping experience by considering security and accessibility.

4.4.2. Continuous Improvement

Professionalism is commitment to transparency and continuous improvement in the e-Store's operations. That is, conducting audits to identify vulnerabilities, updating security measures regularly, and offering bug-bounty programs for ethical vulnerability reporting (BigCommerce, 2023). Hosted on HackerOne, Shopify's bug bounty program has rewarded ethical hackers for identifying security flaws, preventing breaches, and enhancing platform security (Shopify, 2025).

The e-store's operational efficiency and customer trust are improved by complying with ISO 9001 for quality management (Hinrich Foundation, 2023). The e-Store can maintain credibility and security by adopting such measures.

5.0. Conclusion

The e-Store's security assessment found vulnerabilities like SQL Injection, Broken Access Control, and Cross-Site Scripting that threaten user data and business integrity (OWASP, 2023). The defence recommends strict cookie settings, input validation, HTTPS, and security headers. Also, aligning with LESP standards ensures compliance with GDPR and ethical practices (Hinrich Foundation, 2023). While challenges remain, these measures significantly improve the e-Store's security posture and resilience against web threats.

6.0. References

Amazon Help (no date).Help and customer service. [online]. Available from: https://www.amazon.com/gp/help/customer/display.html?nodeId=GTR9XC96UBSLN5WZ (Accessed: 29 March 2025).

Apache HTTP Server (no date). Module mod_rewrite Documentation. [online]. Available from: https://httpd.apache.org/docs/current/mod/mod_rewrite.html?form=MG0AV3 (Accessed: 14 March 2025).

Apache Tomcat (2025). HTTP Header Security Filter Configuration. [online]. Available from: https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter (Accessed: 14 March 2025).

Apache Tomcat (2025). Rewrite Documentation for Tomcat 9.0. [online]. Available from: https://tomcat.apache.org/tomcat-9.0-doc/rewrite.html?form=MG0AV3 (Accessed: 14 March 2025).

Apache Tomcat (2025). SSL Configuration HOW-TO. [online]. Available from: https://tomcat.apache.org/tomcat-9.0-doc/ssl-howto.html (Accessed: 14 March 2025).

BigCommerce (2023). E-commerce compliance: Legal requirements and best practices. [online]. Available from: https://www.bigcommerce.com/articles/ecommerce/compliance/ (Accessed: 5 March 2025).

BigCommerce (2023). Online Business Laws. [online]. Available from: https://www.bigcommerce.com/blog/online-business-laws/ (Accessed: 5 March 2025).

Business & Human Rights Resource Centre (2024). 2024 Ethical Fashion Report reveals Temu and Shein fall short in addressing labour issues. [online]. Available from: https://www.business-humanrights.org/en/latest-news/2024-ethical-fashion-report-reveals-temu-and-shein-fall-short-in-addressing-labour-issues/ (Accessed 29 March 2025).

Creatuity (2023). The importance of social responsibility in e-commerce. [online]. Available from: https://creatuity.com/insights/the-importance-of-social-responsibility-in-ecommerce (Accessed: 5 March 2025).

EC-Council, (no date).Web Application Attacks and Countermeasures. Module 07. [Restricted access]. Available from: https://campusmoodle.rgu.ac.uk/pluginfile.php/7200640/mod_resource/content/1/Web%20Application%20Attacks%20and%20Countermeasures.pdf (Accessed: 29 March 2025)

eCommerce Consulting Firm (2023). Embracing social responsibility: How e-commerce is making a difference. [online]. Available from: https://www.ecommerceconsultingfirm.com/embracing-social-responsibility-how-ecommerce-is-making-a-difference/ (Accessed: 5 March 2025).

eCommerce Fastlane (2023). Ethical e-commerce practices. [online]. Available from: https://ecommercefastlane.com/ethical-ecommerce-practices/ (Accessed: 5 March 2025).

eSecurity Planet (2023). How to Prevent SQL Injection Attacks. [online]. Available from: https://www.esecurityplanet.com/threats/how-to-prevent-sql-injection-attacks/ (Accessed: 14 March 2025).

Federation Of Small Businesses (FSB) (2023). 14 E-commerce laws and legal requirements for online businesses. [online]. Available from: https://www.fsb.org.uk/resources-page/14-e-commerce-laws-and-legal-requirements-for-online-businesses.html (Accessed: 5 March 2025).

Feta, S. (2021). Parameter Tampering Vulnerability Using 3 Different Approaches. [online]. Available from: https://www.cobalt.io/blog/parameter-tampering-vulnerability-using-3-different-approaches (Accessed: 30 March 2025).

GDPR.EU (2023). What is GDPR? [online]. Available from: https://gdpr.eu/what-is-gdpr/ (Accessed: 5 March 2025).

GeeksforGeeks (2023) Insecure Direct Object Reference (IDOR) Vulnerability. [online]. Available from: https://www.geeksforgeeks.org/insecure-direct-object-reference-idor-vulnerability/ (Accessed: 16 March 2025).

Hill, M. and Sharma, S. (2025) The biggest data breach fines, penalties, and settlements so far. CSO Online. [online]. Available from: https://www.csoonline.com/article/567531/the-biggest-data-breach-fines-penalties-and-settlements-so-far.html#:~:text=Hit%20with%20a%20$%201.3%20billion,Regulation%20(GDPR)%20in%20Europe. (Accessed: 29 March 2025).

Hinrich Foundation (2023). Standards matter in digital transformation and e-commerce. [online]. Available from: https://www.hinrichfoundation.com/research/article/tech/standards-matter-in-digital-transformation-and-ecommerce/ (Accessed: 5 March 2025).

Hui, S. (2025). UK Parliament to question Shein and Temu on workers' rights. Associated Press. Updated 6:24 PM GMT, January 7, 2025. [online]. Available from: https://apnews.com/article/uk-parliament-shein-temu-a24fa3a6511cde7fe6b4763d693b986b (Accessed: 29 March 2025).

LEGALGPS (2023). Legal Considerations for eCommerce Sites. [online]. Available from: https://www.legalgps.com/ecommerce-agreements/blog/legal-considerations-for-ecommerce-sites (Accessed: 5 March 2025).

MITRE (2024a) CWE-639: Authorization Bypass Through User-Controlled Key. [online]. Available from: https://cwe.mitre.org/data/definitions/639.html (Accessed: 16 March 2025).

MITRE (2024b) CWE-840: Business Logic Errors. [online]. Available from: https://cwe.mitre.org/data/definitions/840.html (Accessed: 16 March 2025).

MITRE (2024c) CWE-598: Query Strings in URLs. [online]. Available from: https://cwe.mitre.org/data/definitions/598.html (Accessed: 16 March 2025).

Mozilla Developer Network (no date). Guide to Cookies. [online]. Available from: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Cookies (Accessed: 14 March 2025).

Mozilla Developer Network (no date). X-Content-Type-Options Header Documentation. [online]. Available from: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/X-Content-Type-Options (Accessed: 14 March 2025).

Mozilla Developer Network (no date). X-Frame-Options Header Documentation. [online]. Available from: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/X-Frame-Options (Accessed: 14 March 2025).

OWASP (2021a) A03:2021 – Injection. [online]. Available from: https://owasp.org/Top10/A03_2021-Injection/ (Accessed: 16 March 2025).

OWASP (2021b) A01:2021 – Broken Access Control. [online]. Available from: https://owasp.org/Top10/A01_2021-Broken_Access_Control/ (Accessed: 16 March 2025).

OWASP (2021c) Cross-Site Scripting (XSS). [online]. Available from: https://owasp.org/www-community/attacks/xss/ (Accessed: 16 March 2025).

OWASP (no date) Insecure Direct Object Reference Prevention Cheat Sheet. [online]. Available from: https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html (Accessed: 16 March 2025).

OWASP (no date) SQL Injection. [online]. Available from: https://owasp.org/www-community/attacks/SQL_Injection (Accessed: 16 March 2025).

OWASP (2023). OWASP Top Ten Web Application Security Risks. [online]. Available from: https://owasp.org/www-project-top-ten/ (Accessed: 5 March 2025).

PortSwigger (no date) SQL injection. [online]. Available from: https://portswigger.net/web-security/sql-injection#how-to-prevent-sql-injection (Accessed: 16 March 2025).

PTC (2025). Arbortext Publishing Engine Installation and Configuration Guide. [online]. Available from: https://support.ptc.com/help/arbortext/r8.2.1.0/en/index.html#page/PE/peinstall/pe2095.html (Accessed: 14 March 2025).

Secureworks (2012). A famous data security breach & PCI case study: Four years later. [online]. Available from: https://www.secureworks.com/blog/general-pci-compliance-data-security-case-study-heartland (Accessed: 29 March 2025).

Shopify Help Center (no date). Account security best practices. [online]. Available from: https://help.shopify.com/en/manual/privacy-and-security/account-security/account-security-best-practices (Accessed: 29 March 2025).

Shopify (2023). eCommerce Laws. [online]. Available from: https://www.shopify.com/blog/ecommerce-laws (Accessed: 5 March 2025).

Shopify (2025). Shopify Bug Bounty Program. [online]. Available from: https://hackerone.com/shopify?type=team (Accessed: 29 March 2025).

Tidy, J. (2020). British Airways fined £20m over data breach. BBC News, 16 October. [online]. Available from: https://www.bbc.com/news/technology-54568784 (Accessed: 29 March 2025).

TuringSecure (no date) Password submitted using GET method. [online]. Available from: https://turingsecure.com/knowledge-base/issues/password-submitted-using-get-method/ (Accessed: 16 March 2025).

Twingate (2024). What happened in the Ticketmaster data breach? [online]. Available from: https://www.twingate.com/blog/tips/ticketmaster-data-breach (Accessed: 29 March 2025).

W3C (2023). Web Content Accessibility Guidelines (WCAG). [online]. Available from: https://www.w3.org/WAI/standards-guidelines/wcag/ (Accessed: 5 March 2025).

Webiators (2023). Patagonia and Allbirds: Champions of Ethical Practices in E-Commerce. [online]. Available from: https://webiators.com/patagonia-and-allbirds-champions-of-ethical-practices-in-e-commerce/ (Accessed: 29 March 2025).

Zen Media (2023). E-commerce trend for 2021: An increase in socially responsible businesses. [online]. Available from: https://zenmedia.com/blog/ecommerce-trend-for-2021-an-increase-in-socially-responsible-businesses/ (Accessed: 5 March 2025).