

## **AltoroJ security report**

Youssef Hussein Mahmoud	20206111
Ahmed El Sayed Moein	20206120
Ahmed Sami Darwish	20206136
Abdulrahman Emad Kamel	20206148
Youssef Ahmed Abdulghafar	20206152

## Findings list

<b>ID</b>	<b>Name</b>	<b>Severity</b>	<b>After retest severity</b>
<b>ID01</b>	Default Credentials in the <b>login page</b> to the admin panel	<b>High</b>	<b>Resolved</b>
<b>ID02</b>	SQL Injection in the <b>login page</b>	<b>High</b>	<b>Resolved</b>
<b>ID03</b>	Local File Inclusion (LFI) to WEB-INF init logs in the <b>home page</b>	<b>High</b>	
<b>ID04</b>	SQL Injection in <b>bank apply</b> to gold visa	<b>High</b>	<b>Resolved</b>
<b>ID05</b>	Insecure direct object reference (IDOR) in <b>show account history</b>	<b>High</b>	
<b>ID06</b>	Improper Access Control leading to admin pages	<b>High</b>	<b>Resolved</b>
<b>ID07</b>	Cross-Site Request Forgery (CSRF) in <b>transfer funds</b>	<b>High</b>	<b>Resolved</b>
<b>ID08</b>	Cross-Site Request Forgery (CSRF) in <b>adding a new user by admin</b>	<b>High</b>	<b>Resolved</b>
<b>ID09</b>	Cross-Site Request Forgery (CSRF) in <b>Adding Account to an existing user by admin</b>	<b>High</b>	<b>Resolved</b>

<b>ID10</b>	Cross-Site Request Forgery (CSRF) in <b>change user's password by admin</b>	<b>High</b>	<b>Resolved</b>
<b>ID11</b>	HTML Injection in the <b>home page</b>	<b>Medium</b>	<b>Resolved</b>
<b>ID12</b>	Cross-Site Scripting (XSS) in <b>home page search</b>	<b>Medium</b>	<b>Resolved</b>
<b>ID13</b>	Reflected Cross-Site Scripting (RXSS) in <b>bank search articles</b>	<b>Medium</b>	<b>Resolved</b>
<b>ID14</b>	Reflected Cross-Site Scripting (RXSS) in <b>customize language</b>	<b>Medium</b>	<b>Resolved</b>
<b>ID15</b>	Post-Based Cross-Site Scripting (PXSS) in <b>feedback</b>	<b>Medium</b>	<b>Resolved</b>
<b>ID16</b>	URL Redirection in <b>customize language</b>	<b>Medium</b>	<b>Resolved</b>
<b>ID17</b>	Clickjacking vulnerability	<b>Medium</b>	<b>Resolved</b>
<b>ID18</b>	HTML Injection in <b>home page search</b>	<b>Medium</b>	<b>Resolved</b>
<b>ID19</b>	Information disclosure (Sensitive data exposure) in <b>login.jsp code</b>	<b>Low</b>	<b>Resolved</b>

## **Finding details (test and retest)**

**Name:** ID01: Default Credentials in the **Login Page** to the admin panel

**Test Severity: Critical**

**Test Score: 9.1**

### **Description:**

Default credentials were discovered in the login page for accessing the admin panel. This vulnerability allows an attacker to access the admin panel using pre-configured or default usernames and passwords, potentially granting unauthorized access to sensitive administrative functions and data.

### **Impact:**

The impact of this vulnerability is severe. Successful exploitation could lead to unauthorized access to sensitive information, manipulation of system configurations, or even complete compromise of the application and underlying infrastructure.

### **Recommendations:**

- Immediately change the default credentials to strong, unique passwords that adhere to password complexity requirements.
- Implement multi-factor authentication (MFA) to add an extra layer of security to the login process and mitigate the risk of credential-based attacks.
- Regularly rotate passwords and ensure that they are not shared or reused across multiple accounts or systems.
- Conduct security assessments to identify and remediate default credential vulnerabilities, including thorough review of configuration files and settings.
- Educate administrators and users about the importance of using strong, unique passwords and avoiding default or common credentials.
- Monitor login attempts and enforce account lockout policies to prevent brute-force attacks against login credentials.

---

**Name:** ID02: SQL Injection in the **Login Page**

**Test Severity: Critical**

**Test Score: 9.3**

**Description:**

SQL Injection vulnerability was discovered in the login page. This vulnerability allows an attacker to manipulate the SQL queries executed by the application, potentially bypassing authentication mechanisms, accessing unauthorized data, or executing arbitrary SQL commands.

**Impact:**

The impact of this vulnerability is severe. Successful exploitation could lead to unauthorized access to sensitive information, such as user credentials or other confidential data stored in the database, as well as potential data manipulation or deletion.

**Recommendations:**

- Implement parameterized queries or prepared statements to prevent SQL injection attacks by properly escaping user input before executing SQL queries.
- Utilize stored procedures or ORM frameworks to abstract database interactions and minimize direct SQL query execution.
- Enforce principle of least privilege by restricting database access and permissions to only necessary operations and resources.
- Regularly update and patch the application and database management system (DBMS) to address known vulnerabilities and improve security posture.
- Conduct thorough security assessments, including penetration testing and code reviews, to identify and remediate SQL injection vulnerabilities.
- Implement web application firewalls (WAFs) or runtime application self-protection (RASP) solutions to provide additional layers of defense against SQL injection and other injection-based attacks.

**Name:** ID03: Local File Inclusion (LFI) to WEB-INF Init Logs in **Home Page**

**Test Severity: High**

**Test Score: 8.6**

**Description:**

Local File Inclusion (LFI) vulnerability was discovered in the home page. This vulnerability allows an attacker to include arbitrary files located within the WEB-INF directory, such as initialization logs or configuration files, which may contain sensitive information.

**Impact:**

The impact of this vulnerability is severe. Successful exploitation could lead to unauthorized access to sensitive information, including credentials, session tokens, or other confidential data stored in initialization logs or configuration files.

**Recommendations:**

- Review and restrict access permissions to sensitive files and directories, such as WEB-INF, to prevent unauthorized access.
  - Implement input validation and output encoding to sanitize user-supplied input and prevent directory traversal attacks.
  - Configure the web server or application container to disallow direct access to files within the WEB-INF directory.
  - Monitor file system access and log activities to detect and respond to potential exploitation attempts.
  - Regularly review and update file system permissions and configurations to minimize the attack surface and mitigate potential vulnerabilities.
- 

**Name:** ID04: SQL Injection in **Bank Apply to Gold Visa**

**Test Severity: Critical**

**Test Score: 9.2**

**Description:**

SQL Injection vulnerability was discovered in the bank apply functionality for the Gold Visa card. This vulnerability allows an attacker to manipulate SQL

queries executed by the application when processing requests related to applying for the Gold Visa card, potentially gaining unauthorized access to sensitive data or performing malicious actions on the database.

### **Impact:**

The impact of this vulnerability is severe. Successful exploitation could lead to unauthorized access to sensitive information stored in the database, including personally identifiable information (PII), financial records, or other confidential data, as well as potential data manipulation or deletion.

### **Recommendations:**

- Implement parameterized queries or prepared statements to prevent SQL injection attacks by properly escaping user input before executing SQL queries.
- Utilize ORM frameworks or stored procedures to abstract database interactions and minimize direct SQL query execution.
- Enforce principle of least privilege by restricting database access and permissions to only necessary operations and resources.
- Regularly update and patch the application and database management system (DBMS) to address known vulnerabilities and improve security posture.
- Conduct thorough security assessments, including penetration testing and code reviews, to identify and remediate SQL injection vulnerabilities.

---

**Name:** ID05: Insecure Direct Object Reference (IDOR) in **Show Account History**

**Test Severity:** Critical

**Test Score:** 9.3

### **Description:**

Insecure Direct Object Reference (IDOR) vulnerability was discovered in the "show account history" functionality. This vulnerability allows an attacker to manipulate input parameters or directly access resources, such as account records or transactions belonging to other users, which are not properly protected or authorized.

**Impact:**

The impact of this vulnerability is severe. Successful exploitation could lead to unauthorized access to sensitive information, including account balances, transaction details, or personally identifiable information (PII) of other users, as well as potential data manipulation or deletion.

**Recommendations:**

- Implement proper access controls and authorization mechanisms to restrict access to sensitive resources based on user privileges and ownership.
- Utilize indirect object references or opaque identifiers to prevent direct access to internal data structures or database records.
- Implement role-based access control (RBAC) or attribute-based access control (ABAC) to enforce fine-grained access policies and minimize the risk of IDOR vulnerabilities.
- Regularly review and update access control configurations and policies to ensure they align with security requirements and best practices.
- Conduct thorough security assessments, including code reviews and penetration testing, to identify and remediate IDOR vulnerabilities and other authorization issues.
- Implement logging and monitoring to detect and respond to unauthorized access attempts or suspicious activities related to sensitive resources.

---

**Name:** ID06: Improper Access Control Leading to **Admin Pages**

**Test Severity:** Critical

**Test Score:** 9.3

**Description:**

Improper Access Control vulnerability was discovered, allowing unauthorized users to access admin pages or functionalities that should be restricted to privileged users. This vulnerability may arise due to insufficient access controls, misconfigured permissions, or lack of proper authentication mechanisms.

**Impact:**

The impact of this vulnerability is severe. Successful exploitation could lead to unauthorized access to sensitive administrative functionalities, potentially allowing attackers to manipulate system configurations, access confidential information, or perform malicious actions that compromise the security and integrity of the application.

**Recommendations:**

- Review and update access control configurations to enforce proper authentication and authorization mechanisms, ensuring that only authenticated and authorized users can access admin pages or functionalities.
  - Implement role-based access control (RBAC) or attribute-based access control (ABAC) to define and enforce fine-grained access policies based on user roles, privileges, and attributes.
  - Conduct thorough security assessments, including access control reviews and penetration testing, to identify and remediate improper access control vulnerabilities and misconfigurations.
  - Implement multi-factor authentication (MFA) and session management controls to enhance the security of authentication processes and prevent unauthorized access to sensitive functionalities.
- 

**Name:** ID07: Cross-Site Request Forgery (CSRF) in **Transfer Funds**

**Test Severity:** High

**Test Score:** 8.6

**Description:**

Cross-Site Request Forgery (CSRF) vulnerability was discovered in the transfer funds functionality. This vulnerability allows an attacker to trick authenticated users into executing unauthorized transactions, such as transferring funds or making changes to account settings, by exploiting the trust established between the user's browser and the application.

**Impact:**

The impact of this vulnerability is severe. Successful exploitation could lead to unauthorized fund transfers, financial loss, or manipulation of account

balances, compromising the integrity and security of the user's financial transactions.

## **Recommendations:**

- Implement CSRF protection mechanisms, such as anti-CSRF tokens or same-site cookie attributes, to prevent unauthorized requests from being executed by authenticated users.
  - Validate and verify the integrity of incoming requests, including checking the origin and referer headers, to detect and block potential CSRF attacks.
  - Utilize state-changing HTTP methods, such as POST, for sensitive operations like fund transfers, and enforce proper authorization checks to ensure that only authorized users can perform such actions.
  - Conduct security reviews and penetration testing regularly to identify and address CSRF vulnerabilities and other potential security risks.
  - Implement additional security measures, such as Content Security Policy (CSP) headers or security headers, to mitigate the risk of CSRF attacks and protect against other web-based threats.
- 

**Name:** ID08: Cross-Site Request Forgery (CSRF) in **Adding a New User by Admin**

**Test Severity:** Medium

**Test Score:** 6.9

### **Description:**

Cross-Site Request Forgery (CSRF) vulnerability was discovered in the functionality for adding a new user by an admin. This vulnerability allows an attacker to trick the admin user into unintentionally creating a new user account with arbitrary privileges or permissions by exploiting the admin's session and authority within the application.

### **Impact:**

The impact of this vulnerability is severe. Successful exploitation could lead to unauthorized creation of user accounts with elevated privileges, such as administrative roles, compromising the security and integrity of the application's user management system.

## **Recommendations:**

- Implement CSRF protection mechanisms, such as anti-CSRF tokens or same-site cookie attributes, to prevent unauthorized requests from being executed by authenticated users.
  - Validate and verify the integrity of incoming requests, including checking the origin and referer headers, to detect and block potential CSRF attacks.
  - Utilize state-changing HTTP methods, such as POST, for sensitive operations like adding new users, and enforce proper authorization checks to ensure that only authorized users can perform such actions.
  - Implement additional confirmation mechanisms, such as CAPTCHA challenges or multi-step verification processes, for critical operations to mitigate the risk of CSRF attacks and prevent unauthorized actions.
  - Implement additional security measures, such as Content Security Policy (CSP) headers or security headers, to mitigate the risk of CSRF attacks and protect against other web-based threats.
- 

**Name:** ID09: Cross-Site Request Forgery (CSRF) in **Adding Account to an Existing User by Admin**

**Test Severity: Critical**

**Test Score: 9.3**

### **Description:**

Cross-Site Request Forgery (CSRF) vulnerability was discovered in the functionality for adding an account to an existing user by an admin. This vulnerability allows an attacker to trick the admin user into unintentionally adding an account to an existing user with arbitrary permissions or privileges by exploiting the admin's session and authority within the application.

### **Impact:**

The impact of this vulnerability is severe. Successful exploitation could lead to unauthorized addition of accounts to existing users with elevated privileges, such as administrative roles, compromising the security and integrity of the application's user management system.

## **Recommendations:**

- Implement CSRF protection mechanisms, such as anti-CSRF tokens or same-site cookie attributes, to prevent unauthorized requests from being executed by authenticated users.
  - Utilize state-changing HTTP methods, such as POST, for sensitive operations like adding accounts to existing users, and enforce proper authorization checks to ensure that only authorized users can perform such actions.
  - Implement additional confirmation mechanisms, such as CAPTCHA challenges or multi-step verification processes, for critical operations to mitigate the risk of CSRF attacks and prevent unauthorized actions.
  - Implement additional security measures, such as Content Security Policy (CSP) headers or security headers, to mitigate the risk of CSRF attacks and protect against other web-based threats.
- 

**Name:** ID10: Cross-Site Request Forgery (CSRF) in **Change User's Password by Admin**

**Test Severity:** High

**Test Score:** 8.2

**Description:**

This vulnerability allows an attacker to forge a request to change a user's password by exploiting the admin privileges without the user's consent. CSRF attacks can trick an authenticated admin into executing unintended actions, such as changing a user's password, by disguising malicious requests as legitimate ones.

**Impact:**

The impact of this vulnerability is severe; successful exploitation allows an attacker to change a user's password without their knowledge or consent, compromising their account security and potentially leading to unauthorized access to sensitive information or actions.

**Recommendations:**

- Implement CSRF protection mechanisms such as unique tokens for each request.

- Require re-authentication for sensitive actions, particularly those with elevated privileges.
  - Validate the origin of requests to ensure they originate from trusted sources.
  - Enforce strict access controls to limit the actions that can be performed by admin users.
  - Ensure that users can only modify their own passwords through secure, authenticated channels.
- 

**Name:** ID11: HTML Injection in the **Home Page**

**Test Severity:** Medium

**Test Score:** 4.8

**Description:**

HTML Injection vulnerability was discovered on the home page of the application. This vulnerability allows an attacker to inject malicious HTML code into the home page, potentially leading to various attacks such as cross-site scripting (XSS), phishing, or defacement of the page.

**Impact:**

The impact of this vulnerability is moderate. Successful exploitation could lead to unauthorized code execution within users' browsers, compromising their session data, stealing credentials, or redirecting them to malicious websites.

**Recommendations:**

- Regularly sanitize and validate user input to prevent the injection of malicious HTML code.
- Implement Content Security Policy (CSP) headers to restrict the sources of content that can be loaded on the page.
- Encode user-supplied content to HTML entities to prevent it from being interpreted as HTML code.
- Conduct security reviews and penetration testing regularly to identify and address vulnerabilities proactively.

- Educate developers about secure coding practices, particularly regarding input validation and output encoding.
- 

**Name:** ID12: Cross-Site Scripting (XSS) in **Home Page Search**

**Test Severity:** High

**Test Score:** 8.3

**Description:**

Cross-Site Scripting (XSS) vulnerability was discovered in the search functionality of the home page. This vulnerability allows an attacker to inject malicious scripts into the search input field, which can be executed within the context of other users' browsers when they perform a search.

**Impact:**

The impact of this vulnerability is severe. Successful exploitation could lead to various attacks, including session hijacking, credential theft, defacement of the page, or the execution of arbitrary actions on behalf of the user.

**Recommendations:**

- Implement input validation and output encoding to sanitize user-supplied input and prevent the injection of malicious scripts.
  - Use appropriate encoding mechanisms (such as HTML entity encoding or JavaScript escaping) to neutralize user input before rendering it in the browser.
  - Apply Content Security Policy (CSP) headers to restrict the sources of scripts that can be executed on the page.
  - Conduct security reviews and penetration testing regularly to identify and address XSS vulnerabilities.
  - Educate developers about secure coding practices, particularly regarding input validation and output encoding, and provide training on the risks associated with XSS attacks.
-

**Name:** ID13: Reflected Cross-Site Scripting (RXSS) in **Bank Search Articles**

**Test Severity:** High

**Test Score:** 8.8

**Description:**

Reflected Cross-Site Scripting (RXSS) vulnerability was discovered in the bank search articles feature. This vulnerability allows an attacker to inject malicious scripts into the URL parameters used for searching articles, which get reflected back in the response, potentially leading to script execution within the context of other users' browsers.

**Impact:**

The impact of this vulnerability is severe. Successful exploitation could lead to various attacks, including session hijacking, credential theft, defacement of the page, or the execution of arbitrary actions on behalf of the user.

**Recommendations:**

- Implement input validation and output encoding to sanitize user-supplied input and prevent the injection of malicious scripts.
- Use appropriate encoding mechanisms (such as URL encoding) to neutralize user input before using it in dynamic content.
- Apply Content Security Policy (CSP) headers to restrict the sources of scripts that can be executed on the page.
- Conduct security reviews and penetration testing regularly to identify and address RXSS vulnerabilities.
- Educate developers about secure coding practices, particularly regarding input validation and output encoding, and provide training on the risks associated with RXSS attacks.

---

**Name:** ID14: Reflected Cross-Site Scripting (RXSS) in **Customize Language**

**Test Severity:** High

**Test Score:** 8.5

**Description:**

A Reflected Cross-Site Scripting (RXSS) vulnerability was discovered in the customize language feature. This vulnerability allows an attacker to inject malicious scripts into the parameters used for customizing language settings, which get reflected back in the response, potentially leading to script execution within the context of other users' browsers.

**Impact:**

The impact of this vulnerability is severe. Successful exploitation could lead to various attacks, including session hijacking, credential theft, defacement of the page, or the execution of arbitrary actions on behalf of the user.

**Recommendations:**

- Implement input validation and output encoding to sanitize user-supplied input and prevent the injection of malicious scripts.
- Use appropriate encoding mechanisms (such as HTML entity encoding) to neutralize user input before using it in dynamic content.
- Apply Content Security Policy (CSP) headers to restrict the sources of scripts that can be executed on the page.
- Conduct security reviews and penetration testing regularly to identify and address RXSS vulnerabilities.
- Educate developers about secure coding practices, particularly regarding input validation and output encoding, and provide training on the risks associated with RXSS attacks.

---

**Name:** ID15: Post-Based Cross-Site Scripting (PXSS) in **Feedback**

**Test Severity: High**

**Test Score: 8.5**

**Description:**

Post-Based Cross-Site Scripting (PXSS) vulnerability was discovered in the feedback feature. This vulnerability allows an attacker to inject malicious scripts into the feedback submission form, which get stored and reflected in the response, potentially leading to script execution within the context of other users' browsers.

**Impact:**

The impact of this vulnerability is severe. Successful exploitation could lead to various attacks, including session hijacking, credential theft, defacement of the page, or executing arbitrary actions on behalf of the user.

**Recommendations:**

- Implement input validation and output encoding to sanitize user-supplied input and prevent the injection of malicious scripts.
  - Use appropriate encoding mechanisms (such as HTML entity encoding) to neutralize user input before storing and rendering it in dynamic content.
  - Apply strict content security policies to prevent the execution of untrusted scripts.
  - Implement rate limiting and CAPTCHA mechanisms to mitigate automated attacks on feedback submission forms.
  - Conduct regular security reviews and penetration testing to identify and address PXSS vulnerabilities.
  - Educate developers about secure coding practices, particularly regarding input validation and output encoding, and provide training on the risks associated with PXSS attacks.
- 

**Name:** ID16: URL Redirection in **Customize Language**

**Test Severity:** High

**Test Score:** 8.3

**Description:**

URL Redirection vulnerability was discovered in the customize language feature. This vulnerability allows an attacker to manipulate the URL parameters for customizing language settings to redirect users to malicious websites or phishing pages.

**Impact:**

The impact of this vulnerability is moderate. Successful exploitation could lead to unauthorized redirection of users to malicious websites, potentially resulting in phishing attacks, the theft of sensitive information, or the installation of malware.

## **Recommendations:**

- Implement whitelist-based validation of redirection URLs to restrict redirection to trusted domains.
  - Avoid passing user-controlled input directly into the redirection URL.
  - Use a safe redirection method that does not rely on user-supplied input.
  - Educate users about the risks associated with unexpected redirects and advise them to verify URLs before proceeding.
  - Monitor and log redirection attempts to detect and mitigate suspicious activities.
  - Conduct regular security reviews and penetration testing to identify and address URL redirection vulnerabilities.
  - Stay informed about the latest security best practices and consider implementing security headers like Content Security Policy (CSP) to mitigate the risk of such vulnerabilities.
- 

**Name:** ID17: Clickjacking Vulnerability

**Test Severity:** Medium

**Test Score:** 5.7

### **Description:**

A clickjacking vulnerability was discovered in the application. This vulnerability allows an attacker to embed the application in a malicious website or iframe, tricking users into performing unintended actions by clicking on seemingly harmless elements that are layered on top of the application.

### **Impact:**

The impact of this vulnerability is severe. Successful exploitation could lead to various attacks, including unauthorized actions performed by the user without their knowledge, such as clicking on disguised buttons or links that perform malicious actions within the application.

## **Recommendations:**

- Implement X-Frame-Options HTTP header with the value "DENY" or "SAMEORIGIN" to prevent the application from being embedded in iframes on other domains.
  - Utilize the Content-Security-Policy (CSP) frame-ancestors directive to restrict which domains can embed the application in iframes.
  - Implement a frame-busting script to prevent the application from being framed by other websites.
  - Educate users about the risks of clickjacking and advise them to be cautious when interacting with unfamiliar or suspicious elements on web pages.
  - Regularly review and update security headers and policies to mitigate emerging threats and vulnerabilities.
  - Conduct security reviews and penetration testing regularly to identify and address clickjacking vulnerabilities.
  - Stay informed about the latest security best practices and consider implementing additional security measures, such as JavaScript-based protection mechanisms, to mitigate the risk of clickjacking attacks further.
- 

**Name:** ID18: HTML Injection in **Home Page Search**

**Test Severity:** Low

**Test Score:** 2.1

### **Description:**

HTML Injection vulnerability was discovered in the search functionality of the home page. This vulnerability allows an attacker to inject arbitrary HTML code into the search input field, which may be reflected in other users' browsers, potentially leading to various attacks such as Cross-Site Scripting (XSS), phishing, or defacement of the page.

**Impact:**

The impact of this vulnerability is moderate. Successful exploitation could lead to unauthorized execution of HTML or JavaScript code within users' browsers, potentially compromising their session data, stealing credentials, or redirecting them to malicious websites.

**Recommendations:**

- Implement strict input validation to sanitize user-supplied input and prevent the injection of HTML code.
- Encode user-generated content and dynamically generated HTML to prevent it from being interpreted as HTML code by the browser.
- Utilize appropriate output encoding mechanisms (such as HTML entity encoding) to neutralize user input before rendering it in the browser.
- Implement Content Security Policy (CSP) headers to restrict the sources of content that can be loaded on the page.
- Conduct regular security reviews and penetration testing to identify and address HTML injection vulnerabilities.
- Educate developers about secure coding practices, particularly regarding input validation and output encoding, and provide training on the risks associated with HTML injection attacks.

---

**Name:** ID19: Information Disclosure (Sensitive Data Exposure) in **login.jsp**  
**Code**

**Test Severity:** High

**Test Score:** 8.5

**Description:**

Information Disclosure (Sensitive Data Exposure) vulnerability was discovered in the code of the login.jsp file. This vulnerability may expose sensitive information, such as plaintext passwords, session tokens, or other confidential data, directly in the source code or through improper error handling.

## **Impact:**

The impact of this vulnerability is severe. Successful exploitation could lead to unauthorized access to sensitive information, compromising user accounts, and privacy, and potentially leading to identity theft or unauthorized access to protected resources.

## **Recommendations:**

- Review the code of login.jsp and ensure that sensitive information is not directly exposed or stored in plaintext.
  - Implement secure password storage mechanisms, such as hashing with salt, to protect user passwords from exposure during a breach.
  - Utilize secure communication protocols (e.g., HTTPS) to encrypt sensitive data transmitted between the client and server.
  - Implement proper error handling and response mechanisms to prevent the leaking of sensitive information in error messages or logs.
  - Regularly review and update the codebase to address potential vulnerabilities and improve security practices.
-

## Finding scenarios

### ID01

- Finding test steps
  - Go to the login page in doLogin function
  - Use Burp Suite to intercept the request with the following payload(default credentials): **uid=admin&passw=admin&btnSubmit=login**

Burp Suite Community Edition v2024.3.1.4 - Temporary Project

Target: http://localhost:8081

Request

```
POST /Gradle__altoromutual_war/doLogin HTTP/1.1
Host: localhost:8081
User-Agent: curl/8.2.1
Accept: */*
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 37
uid=admin&passw=admin&btnSubmit=login
```

Response

```
HTTP/1.1 302
Set-Cookie: JSESSIONID=FDBADE6628AAB609FFCA88D01CE2B3E; Path=/Gradle__altoromutual_war;
HTTP/1.1 200
Set-Cookie: AltoroAccounts=0DAW0DAwF1MvCvBvnfBZx41LjzOTQ300M2MUU3fDgwMDAwMX50aGVja2luZ345MzgyMC40Nhw=
Location: /Gradle__altoromutual_war/bank/main.jsp
Content-Length: 0
Date: Wed, 08 May 2024 16:42:50 GMT
Connection: close

```

- Whoohoo! It logged in to the admin pages

Altoro Mutual

localhost:8081/Gradle\_\_altoromutual\_war/bank/main.jsp

AltoroMutual

Sign Off | Contact Us | Feedback | Search | Go

MY ACCOUNT | PERSONAL | SMALL BUSINESS | INSIDE ALTORO MUTUAL

I WANT TO ...

- View Account Summary
- View Recent Transactions
- Transfer Funds
- Search News Articles
- Customize Site Language

ADMINISTRATION

- Edit Users

Hello Admin User

Welcome to Altoro Mutual Online.

View Account Details: 800000 Corporate GO

Congratulations!

You have been pre-approved for an Altoro Gold Visa with a credit limit of \$100000!

Click Here to apply.

This web application is open source! Get your copy from [GitHub](#) and take advantage of advanced features

The Altoro website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/aposcan/>.

Copyright © 2008, 2017, IBM Corporation. All rights reserved. Copyright © 2017, 2024, HCL Technologies, Ltd., All rights reserved.

- Fixing steps
  - Go to LoginServlet in the source code
  - if username equals to password it will be false

The screenshot shows the Altorox J3.2 IDE interface. On the left, there's a project tree with various Java files like admin.jsp, transaction.jsp, transfer.jsp, TransferServlet.java, and AdminLoginServlet.java. The AdminLoginServlet.java file is open in the main editor, showing Java code for handling login requests. A specific line of code is highlighted: `else if(password == username){` This indicates a security vulnerability where the password and username are being compared directly. Below the editor, the Tomcat 8.5.100 log window shows several INFO messages from the Catalina startup host configuration.

```

10-May-2024 23:46:28.182 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig
10-May-2024 23:46:28.198 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig
10-May-2024 23:46:28.198 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig
10-May-2024 23:46:28.271 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig
10-May-2024 23:46:28.271 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig
10-May-2024 23:46:28.277 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig
10-May-2024 23:46:28.277 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig
10-May-2024 23:46:28.285 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

```

- It is a simple solution but it is must prevent from the admins from adding such default credentials
- Finding RE-TEST steps
  - Log in with these credentials and will fail 😊

## ID02

- Finding test steps

- Go to the login page and try to make SQL injection on username input

- Use BurbSuite to intercept the request with the payload:  
uid=admin'+OR+1%3d1+---+-&passw=admin&btnSubmit=login

- Accessed to main pages and set cookies for user identity
- Fixing steps
  - Go to DBUtilis in isValidUser function
  - The input username and password are not sanitized !!

```

    /**
 * Authenticate user
 * @param user user name
 * @param password password
 * @return true if valid user, false otherwise
 * @throws SQLException
 */
5 usages ▲ apivkine@ca.ibm.com <apivkine@ca.ibm.com>
public static boolean isValidUser(String user, String password) throws SQLException{
    if (user == null || password == null || user.trim().length() == 0 || password.trim().length() == 0)
        return false;

    Connection connection = getConnection();
    Statement statement = connection.createStatement();

    ResultSet resultSet = statement.executeQuery("SELECT COUNT(*)FROM PEOPLE WHERE USER_ID = ?");
    if (resultSet.next()){
        if (resultSet.getInt( columnIndex: 1 ) > 0)
            return true;
    }
    return false;
}

/**
 * Get user information
 * @param username
 * @return user information
 * @throws SQLException
 */

```

- Implement prepared statement to prevent SQL injection

```

    /**
 * Authenticate user
 * @param user user name
 * @param password password
 * @return true if valid user, false otherwise
 * @throws SQLException
 */
5 usages ▲ apivkine@ca.ibm.com <apivkine@ca.ibm.com>
public static boolean isValidUser(String user, String password) throws SQLException{
    if (user == null || password == null || user.trim().length() == 0 || password.trim().length() == 0)
        return false;

    Connection connection = getConnection();
    Statement statement = connection.createStatement();

    ResultSet resultSet = statement.executeQuery("SELECT COUNT(*)FROM PEOPLE WHERE USER_ID = ?");
    if (resultSet.next()){
        if (resultSet.getInt( columnIndex: 1 ) > 0)
            return true;
    }
    return false;
}

/**
 * Get user information
 * @param username
 * @return user information
 * @throws SQLException
 */

```

- Finding RE-TEST steps

- Go to login page with same payload

The AltoroJ website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/appscan/>.

Copyright © 2008-2017 IBM Corporation. All rights reserved. Copyright © 2017-2024 HCL Technologies Ltd. All rights reserved.

- Use BurbSuite with the same payload on the same request

- Prevented the user from proceeding to protected routes!!

The screenshot shows a web browser window for 'Altoro Mutual' at 'localhost:8081/Gradle\_\_altoromutual\_war/login.jsp'. The page has a header with the Altoro Mutual logo, navigation links for 'Sign In', 'Contact Us', 'Feedback', and a search bar. A banner on the right says 'DEMO SITE ONLY'. The main content area is titled 'Online Banking Login' and displays a red error message: 'Login Failed: We're sorry, but this username or password was not found in our system. Please try again.' Below the message are input fields for 'Username' and 'Password', and a 'Login' button. On the left sidebar, there are three sections: 'PERSONAL' (Deposit Product, Checking, Loan Products, Cards, Investments & Insurance, Other Services), 'SMALL BUSINESS' (Deposit Products, Lending Services, Cards, Insurance, Retirement, Other Services), and 'INSIDE ALTORO MUTUAL' (About Us, Contact Us, Locations, Investor Relations, Press Room, Careers, Subscribe). At the bottom, there are links for 'Privacy Policy', 'Security Statement', 'Server Status Check', 'REST API', and copyright information: 'Copyright © 2008-2017 IBM Corporation. All rights reserved. Copyright © 2017-2024 HCL Technologies Ltd. All rights reserved.' A note at the bottom right states: 'This web application is open source! Get your copy from GitHub and take advantage of advanced features'.

## ID03

- Finding test steps
  - Go to index and route to any subroute

- change the content parameter to “..../WEB-INF/init.log”

The AltoroJ website is published by HCL Technologies, Ltd, for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/appscan/>.

Copyright © 2008-2017 IBM Corporation. All rights reserved. Copyright © 2017-2024 HCL Technologies, Ltd. All rights reserved.

- It gets the log file from the server ✗
- Use BurbSuite to intercept the request

- Fixing steps

-

- Finding RE-TEST steps

## ID04

- Finding test steps
  - Go to credit card gold visa apply
  - Use BurbSuite to intercept the request with payload:passwd=testpassword'+OR+1=1---&Submit=Submit

The screenshot shows the Burp Suite interface with the following details:

**Request:**

```

1 POST /Gradle__altermutual_war/bank/ccApply HTTP/1.1
2 Host: localhost:8081
3 User-Agent: curl/8.2.1
4 Accept: */*
5 Connection: close
6 Cookie: JSESSIONID=96ADFBF4604FCF82D22FD37974090474; AltoroAccounts=
7 ODAwMDAwIjNvcnBcmF0ZX41lJizOTQ30DM2MU3FDgWMDawMX5DaGVja2luZ345MzgycMC40NHw=
8 Content-Length: 46
9 Content-Type: application/x-www-form-urlencoded
10 passwd=testpassword'+OR+1=1---&Submit=Submit
  
```

**Response:**

```

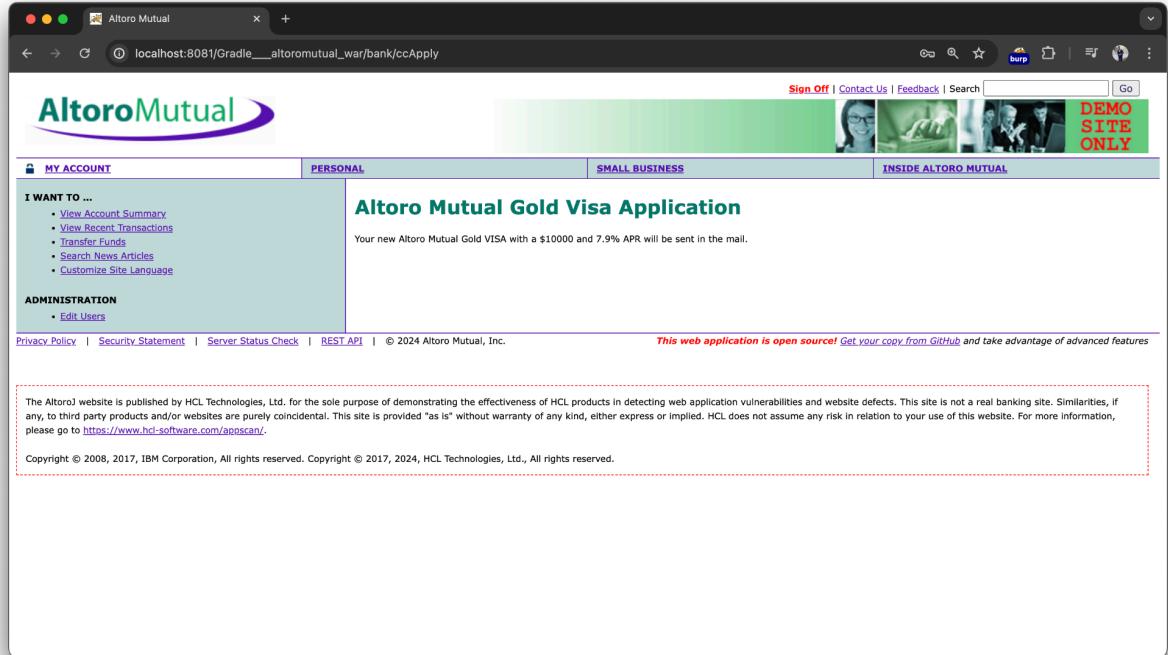
1 HTTP/1.1 500
2 Content-Type: text/html; charset=ISO-8859-1
3 Content-Length: 6106
4 Date: Thu, 09 May 2024 18:12:14 GMT
5 Connection: close
6
7
8
9
10
11
12
13
14 <!-- BEGIN HEADER -->
15 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
16 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
17 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" >
18
19
20
21 <head>
22   <title>
23     Altoro Mutual
24   </title>
25   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /
26   <link href="/Gradle__altermutual_war/style.css" rel="stylesheet" type
27 =="text/css" />
28 </head>
29 <body style="margin-top:5px;">
30
31 <div id="header" style="margin-bottom:5px; width: 99%;">
32   <form id="frmSearch" method="get" action="/Gradle__altermutual_war/search.jsp">
33     <table width="100%" border="0" cellpadding="0" cellspacing="0">
34       <tr>
35         <td rowspan="2">
36           <a id="HoverLink1" href="/Gradle__altermutual_war/index.jsp" >
  
```

**Inspector** panel on the right shows the following sections:

- Request attributes: 2
- Request query parameters: 0
- Request body parameters: 2
- Request cookies: 2
- Request headers: 7
- Response headers: 4

Bottom status bar: 6,245 bytes | 67 millis | Memory: 209.3MB

- SQL injection was detected and proceeded to the requested functionality



- Fixing steps
  - Go to CCApplyServlet

```

dminServlet.java AdminLoginServlet.java LoginServlet.java DBUtil.java CCApplyServlet.java
10 usages ▲ apkv +1
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException {
    // do nothing with credit card application
    // redirect to success page

    try {
        String passwd = request.getParameter("passwd");
        User user = (User)(request.getSession()).getAttribute(ServletUtil.SESSION_ATTR_USER);

        //correct password entered
        if (DBUtil.isValidUser(user.getUsername(), passwd.trim())) {
            RequestDispatcher dispatcher = request.getRequestDispatcher("/bank/applysuecess.jsp");
            dispatcher.forward(request, response);
        }
        //incorrect password entered
        request.getSession().setAttribute("loginError", "Login Failed: We're sorry, but this user");
        RequestDispatcher dispatcher = request.getRequestDispatcher("/bank/apply.jsp");
        dispatcher.forward(request, response);
    }
}

```

Tomcat Localhost Log

10-May-2024 17:35:29.351 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 17:35:29.344 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 17:35:29.344 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 17:35:29.409 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 17:35:29.409 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 17:35:29.415 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 17:35:29.421 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

- This servlet uses the same isValidUser function in DBUtils that has resolved SQL injection vulnerability in ID02
- Finding RE-TEST steps
  - Use BurbSuite to intercept the request with the same payload

Burp Suite Community Edition v2024.3.1.4 - Temporary Project

Target: http://localhost:8081

**Request**

```
Pretty Raw Hex
1 POST /Gradle__altoromutual_war/bank/ccApply HTTP/1.1
2 Host: localhost:8081
3 User-Agent: curl/8.2.1
4 Accept: */*
5 Connection: close
6 Cookie: JSESSIONID=F990221B091F6F045B8B25E123C29039; AltoroAccounts=0DAwMDAwTkvcnBvcmFzX41LjZzOT30DE2MUU3fDgMDAwMX50aGVja2luZ345MzgyNS40NHw=; Content-Type: application/x-www-form-urlencoded
7 Content-Length: 46
8
9
10 passwd=testpassword'+OR#1=1---&Submit=Submit
```

**Response**

```
Pretty Raw Hex Render
93
94
95
96
97
98
99
```

**Inspector**

Selected text: `\t \t \t Login Failed: We're sorry, but this username or password was not found in our system. Please try again.`

- Blocked from applying for Gold Visa credit card 😊

Altoro Mutual

Sign Off | Contact Us | Feedback | Search | Go

MY ACCOUNT PERSONAL SMALL BUSINESS INSIDE ALTORO MUTUAL

I WANT TO ...

- View Account Summary
- View Recent Transactions
- Transfer Funds
- Search News Articles
- Customize Site Language

ADMINISTRATION

- Edit Users

Altoro Mutual Gold Visa Application

No application is needed. To approve your new \$10000 Altoro Mutual Gold Visa with an 7.9% APR simply enter your password below.

Login Failed: We're sorry, but this username or password was not found in our system. Please try again.

Password:

Privacy Policy | Security Statement | Server Status Check | REST API | © 2024 Altoro Mutual, Inc.

This web application is open source! Get your copy from GitHub and take advantage of advanced features

The AltoroJ website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/appscan/>.

Copyright © 2008, 2017, IBM Corporation, All rights reserved. Copyright © 2017, 2024, HCL Technologies, Ltd., All rights reserved.

## ID05

- Finding test steps
  - Go sign in as a normal user
  - Go to account summary
  - Change the route to “bank/showAccount?listAccounts=800000”

The screenshot shows a web browser window for the Altoro Mutual website. The URL is `localhost:8081/Gradle_..._altoromutual_war/bank/showAccount?listAccounts=800000`. The page has a green header with the Altoro Mutual logo and navigation links for Sign Off, Contact Us, Feedback, and Search. A banner on the right says "DEMO SITE ONLY". The main content area is titled "Account History - 800000". It includes a "Balance Detail" table with one row for "800002 Savings" showing an ending balance of \$52394781.61 and an available balance of \$52394781.61. Below it is a "10 Most Recent Transactions" table with six rows of deposit and withdrawal entries. At the bottom is a "Credits" table with five rows of paycheck entries for account 1001160140.

Date	Description	Amount
2024-05-09	Deposit	\$1.00
2024-05-09	Withdrawal	-\$1.00
2024-05-09	Deposit	\$1.00
2024-05-09	Withdrawal	-\$1.00
2024-05-09	Deposit	\$1.00
2024-05-09	Withdrawal	-\$1.00

Account	Date	Description	Amount
1001160140	12/29/2004	Paycheck	1200
1001160140	01/12/2005	Paycheck	1200
1001160140	01/29/2005	Paycheck	1200
1001160140	02/12/2005	Paycheck	1200

- Accessed admin bank account 😢
- Fixing steps
  -
- Finding RE-TEST steps

## ID06

- Finding test steps

- Sign in as a normal user

**Altoro Mutual**

**Online Banking Login**

Username:

Password:

**PERSONAL**

- Deposit Product
- Checking
- Loan Products
- Cards
- Investments & Insurance
- Other Services

**SMALL BUSINESS**

- Deposit Products
- Lending Services
- Cards
- Insurance
- Retirement
- Other Services

**INSIDE ALTORO MUTUAL**

- About Us
- Contact Us
- Locations
- Investor Relations
- Press Room
- Careers
- Subscribe

Privacy Policy | Security Statement | Server Status Check | REST API | © 2024 Altoro Mutual, Inc.

This web application is open source! Get your copy from GitHub and take advantage of advanced features

The AltoroJ website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/appscan/>.

Copyright © 2008-2017, IBM Corporation. All rights reserved. Copyright © 2017-2024, HCL Technologies, Ltd. All rights reserved.

- route to admin page

**Hello John Smith**

Welcome to Altoro Mutual Online.

View Account Details:  GO

**Congratulations!**

You have been pre-approved for an Altoro Gold Visa with a credit limit of \$10000!

Click [Here](#) to apply.

**MY ACCOUNT**

- View Account Summary
- View Recent Transactions
- Transfer Funds
- Search News Articles
- Customize Site Language

Privacy Policy | Security Statement | Server Status Check | REST API | © 2024 Altoro Mutual, Inc.

This web application is open source! Get your copy from GitHub and take advantage of advanced features

The AltoroJ website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/appscan/>.

Copyright © 2008, 2017, IBM Corporation, All rights reserved. Copyright © 2017, 2024, HCL Technologies, Ltd., All rights reserved.

- improper access control accessed user to admin pages

The screenshot shows a web browser window for the 'Altoro Mutual' website, specifically the 'admin/admin.jsp' page. The header includes the Altoro Mutual logo, navigation links for 'Sign Off', 'Contact Us', 'Feedback', and a search bar. A banner at the top right says 'DEMO SITE ONLY'. The main content area is titled 'Edit User Information'.

**Add an account to an existing user:**

- Users: admin (selected)
- Account Types: Savings (selected)
- Add Account button

**Change user's password:**

- Users: admin (selected)
- Password and Confirm fields
- Change Password button

**Add a new user:**

- First Name and Last Name input fields
- Username and Password fields
- Confirm fields
- Add User button

A note at the bottom of the form area states: "It is highly recommended that you leave the username as first initial last name."

At the very bottom of the page, there are links for 'Privacy Policy', 'Security Statement', 'Server Status Check', 'REST API', and copyright information: '© 2024 Altoro Mutual, Inc.' and 'This web application is open source! Get your copy from GitHub and take advantage of advanced features'.

- Fixing steps
  - Go to admin.jsp in the admin subroutine
  - Edit the access of the content to admin roles only if not, it redirects to index.jsp

The screenshot shows the AltOrOJ IDE interface. The top navigation bar includes tabs for FeedbackServlet.java, queryxpath.jsp, login.jsp, search.jsp, AdminServlet.java, admin.jsp (which is currently selected), feedbackReview.jsp, and membertoc.jspf. The left sidebar displays project files like pf, FeedbackServlet.java, queryxpath.jsp, login.jsp, search.jsp, AdminServlet.java, admin.jsp, feedbackReview.jsp, membertoc.jspf, and a file named div#Wrapper. The main editor area contains Java code for the admin.jsp page, which includes JSP tags, imports for ServletUtil and User, and a JavaScript function for password confirmation. Below the editor is a Services panel showing Tomcat 8.5.100. The bottom right corner shows the terminal with the command "AltOrOJ > WebContent > admin > admin.jsp" and the output "28:52 (121 chars, 1 line break) LF ISO-8859-1 Tab\*". The bottom left corner shows the status bar with "AltOrOJ" and "ISO-8859-1".

```
22 %>
23 <jsp:include page="/header.jspf"/>
24
25 <div id="wrapper" style="...>
26   <jsp:include page="/bank/membertoc.jspf" />
27   <!-- User checker user = (User) request.getSession().getAttribute("user"); -->
28   <if (checker.getRole() == User.Role.Admin){%>
29     <td valign="top" colspan="3" class="bb">
30       <%@page import="com.ibm.security.appscan.altoromutual.util.ServletUtil"%>
31       <%@page import="com.ibm.security.appscan.altoromutual.model.User" %>
32
33       <%
34         String[] users = ServletUtil.getBankUsers();
35       <%
36
37       <script language="javascript">
38
39         function confirmpass(myform)
40         {
41           if (myform.password1.value.length && (myform.password1.value==myform.password2.value))
42             div#Wrapper
```

Services All Services Tomcat 8.5.100

Tomcat Localhost Log Tomcat Catalina Log

Tomcat 8.5.100 [local] Gradle : altoromutual

10-May-2024 23:02:19.108 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig 10-May-2024 23:02:19.120 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig 10-May-2024 23:02:19.120 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig 10-May-2024 23:02:19.185 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig 10-May-2024 23:02:19.185 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig 10-May-2024 23:02:19.191 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig 10-May-2024 23:02:19.191 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig 10-May-2024 23:02:19.198 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

The screenshot shows the AltoroJ IDE interface with the following details:

- Top Bar:** Shows tabs for FeedbackServlet.java, queryxpath.jsp, login.jsp, search.jsp, AdminServlet.java, admin.jsp (selected), feedbackReview.jsp, and membertoc.jspf.
- Code Editor:** Displays the Java code for `admin.jsp`. The code includes JSP tags like `<td colspan="4">`, `<tr>`, and `</table>`, as well as a conditional block with `<%> else response.sendRedirect("../index.jsp");%>`.
- Bottom Left:** Shows the project structure under `div#Wrapper`, listing Services, All Services, and Tomcat 8.5.100.
- Bottom Center:** Shows the Tomcat Localhost Log and Tomcat Catalina Log. The Tomcat Catalina Log is expanded, showing multiple entries from May 10, 2024, at 23:02:19, such as INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.

- Finding RE-TEST steps
    - Sign in as a normal user

- o go to admin page

**Altoro Mutual**

**PERSONAL**

- Deposit Product
- Checking
- Loan Products
- Cards
- Investments & Insurance
- Other Services

**PERSONAL**

**Online Banking with FREE Online Bill Pay**

No stamps, envelopes, or checks to write give you more time to spend on the things you enjoy.

**Real Estate Financing**

Fast, Simple, Professional. Whether you are preparing to buy, build, purchase, or construct new space, let Altoro Mutual's premier real estate lenders help with financing. As a regional leader, we know the market, we understand the business, and we have the track record to prove it.

**SMALL BUSINESS**

- Deposit Products
- Lending Services
- Cards
- Insurance
- Retirement
- Other Services

**INSIDE ALTORO MUTUAL**

- About Us
- Contact Us
- Locations
- Investor Relations
- Press Room
- Careers
- Subscribe

**INSIDE ALTORO MUTUAL**

**Business Credit Cards**

You're always looking for ways to improve your company's bottom line. You want to be informed, improve efficiency and control expenses. Now, you can do it all - with a business credit card account from Altoro Mutual.

**Retirement Solutions**

Retaining good employees is a tough task. See how Altoro Mutual can assist you in accomplishing this feat through effective Retirement Solutions.

**DEMO SITE ONLY**

**Privacy and Security**

The 2000 employees of Altoro Mutual are dedicated to protecting your [privacy](#) and [security](#). We pledge to provide you with the information and resources that you need to help secure your information and keep it confidential. This is our promise.

**Win a Samsung Galaxy S10 smartphone**

Completing this short survey will enter you in a draw for 1 of 5 Samsung Galaxy S10 smartphones! We look forward to hearing your important feedback.

**This web application is open source!** Get your copy from [GitHub](#) and take advantage of advanced features

The AltoroJ website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/appscan/>.

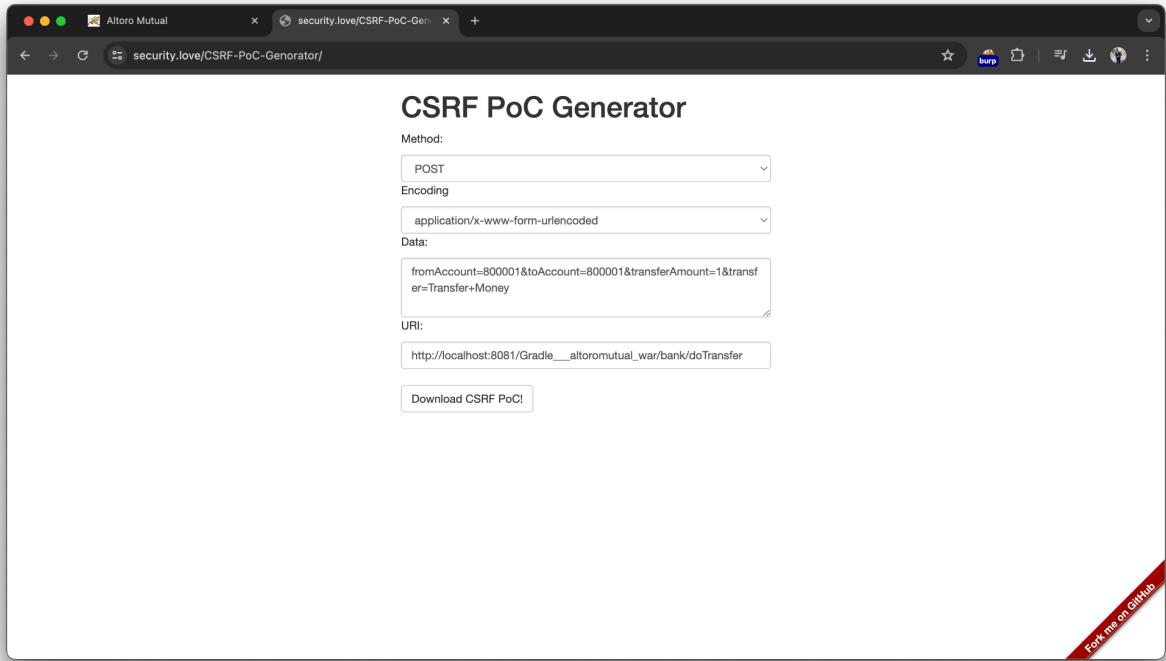
Copyright © 2008, 2017, IBM Corporation, All rights reserved. Copyright © 2017, 2024, HCL Technologies, Ltd., All rights reserved.

- o Routed to index 😊

## ID07

- Finding test steps

- Let's prepare a CSRF PoC for transfer funds functionality 😈



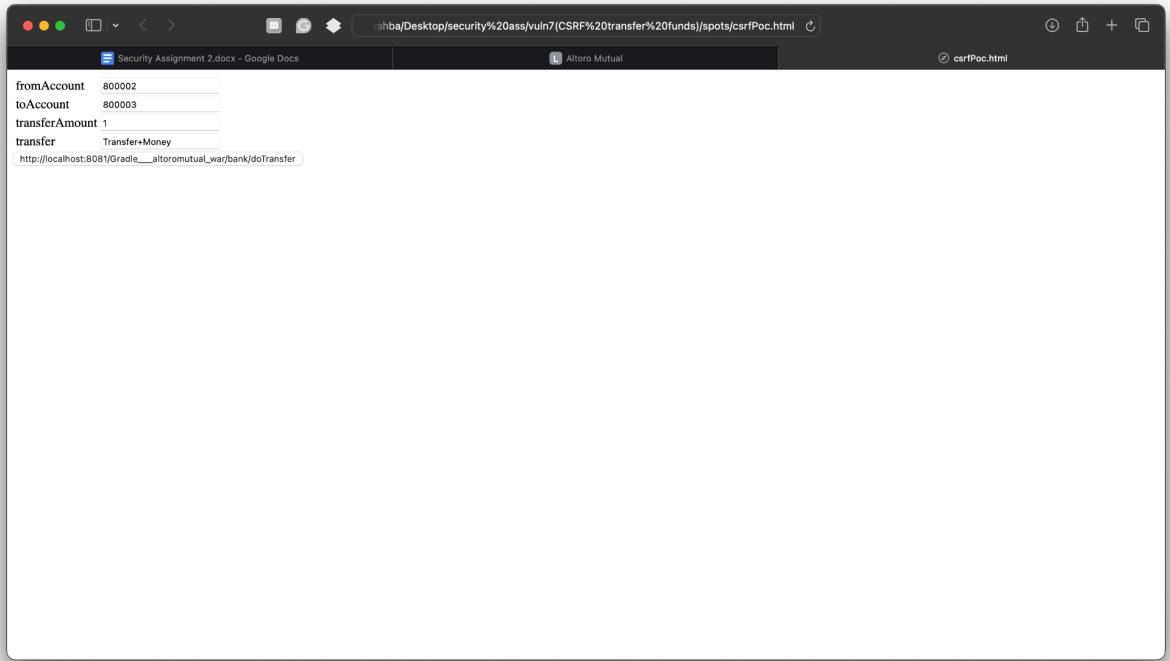
```

<html>
<form enctype="application/x-www-form-urlencoded" method="POST"
action="http://localhost:8081/Gradle__altoromutual_war/bank/doTransfer">
<table>
<tr>
<td>fromAccount</td>
<td><input type="text" value="800002" name="fromAccount"></td>
</tr>
<tr>
<td>toAccount</td>
<td><input type="text" value="800003" name="toAccount"></td>
</tr>
<tr>
<td>transferAmount</td>
<td><input type="text" value="1" name="transferAmount"></td>
</tr>
<tr>
<td>transfer</td>
<td><input type="text" value="Transfer+Money" name="transfer"></td>
</tr>
</table><input type="submit" value="http://localhost:8081/Gradle__altoromutual_war/bank/
doTransfer">
</form>

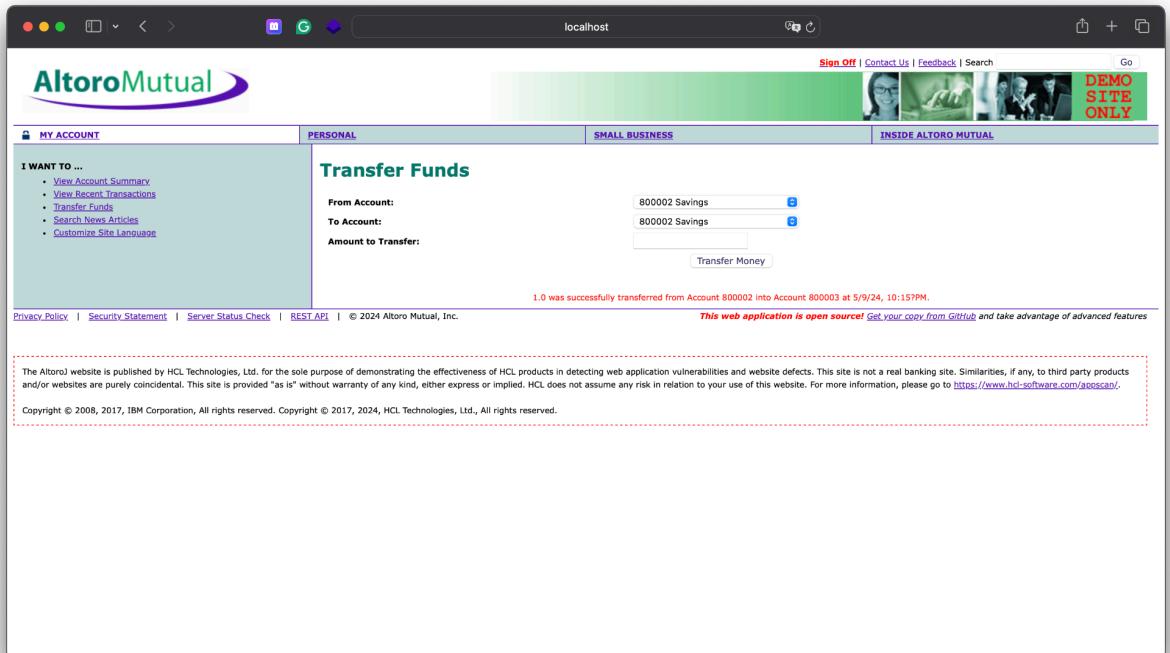
```

The code editor shows the generated HTML code for the CSRF payload. The file is named "csrfPoc.html". The code consists of an HTML form with a POST method, an action of "http://localhost:8081/Gradle\_\_altoromutual\_war/bank/doTransfer", and a table with four rows. The first row contains "fromAccount" and its value "800002". The second row contains "toAccount" and its value "800003". The third row contains "transferAmount" and its value "1". The fourth row contains "transfer" and its value "Transfer+Money". A submit button at the end points to the same URL.

- Open the generated poc in the browser



- Go and press the button
- Ooooh! executed the transfer between the accounts specified 😢



- Fixing steps

- add csrf token in the admin.jsp and in the submit form using hidden input

```

<jsp:include page="/header.jspf"/>
<div id="wrapper" style="...>
<jsp:include page="membertoc.jspf"/>
<td valign="top" colspan="3" class="bb">
<%@page import="com.ibm.security.appscan.altoromutual.model.Account"%>
<%
String csrfToken = java.util.UUID.randomUUID().toString();
session.setAttribute("csrf", csrfToken);
%>
<%
com.ibm.security.appscan.altoromutual.model.User user = (com.ibm.security.appscan.altoromutual.model.User) session.getAttribute("user");
%>

<script type="text/javascript">

function confirmInput(myform) {
    var dbt=document.getElementById("fromAccount").value;
    var cdt=document.getElementById("toAccount").value;
}

```

```

</select>
</td>
</tr>
<tr>
<td><strong> Amount to Transfer:</strong></td>
<td><input type="text" id="transferAmount" name="transferAmount"></td>
<input type="hidden" name="csrfToken" value="<%>${csrfToken}<%>">
</tr>
<tr>
<td colspan="2" align="center"><input type="submit" name="transfer" value="Transfer Money" />
</td>
</tr>
<tr>
<td colspan="2" align="center">
<span id="ctl0__ctl0_Content_Main_postResp" align="center"><span style='...>${(request.get
<span id="soapResp" name="soapResp" align="center" />

```

- check-in admin servlet if the csrf token in session is equal to the token that came in the request then execute else redirect to login and release

the session

The screenshot shows the AltoroJ IDE interface. On the left, the Project tree displays several servlet classes like AccountViewServlet, AdminLoginServlet, AdminServlet, etc., under the servlet package. Below them are DBUtil, OperationsUtil, and ServletUtil. Under WebContent [main], there's an admin folder containing an admin.jsp file. The main editor window shows the Java code for TransferServlet.java. The code handles a POST request, checks if the user is logged in, retrieves account IDs and transfer amount from parameters, generates a CSRF token, and performs a transfer operation using OperationsUtil. The Tomcat 8.5.100 log window at the bottom shows deployment logs for 'altoromutual.war' and Catalina startup logs. The status bar at the bottom right indicates the log contains 73 lines.

```
if(!ServletUtil.isLoggedIn(request)){
    response.sendRedirect( "Login.jsp");
    return ;
}

String accountIdString = request.getParameter( "fromAccount");
long creditActId = Long.parseLong(request.getParameter( "toAccount"));
double amount = Double.parseDouble(request.getParameter( "transferAmount"));

String reqToken = request.getParameter( "csrf");
HttpSession session = request.getSession();
if (reqToken == session.getAttribute( "csrf")) {
    String message = OperationsUtil.doServletTransfer(request,creditActId,accountIdString,amount);

    RequestDispatcher dispatcher = request.getRequestDispatcher( "transfer.jsp");
    request.setAttribute( "message", message);
    dispatcher.forward(request, response);
} else {
    response.sendRedirect( "Login.jsp");
}
```

- Finding RE-TEST steps
  - Go to the HTML poc generated
  - the user will be logged out and redirected to login

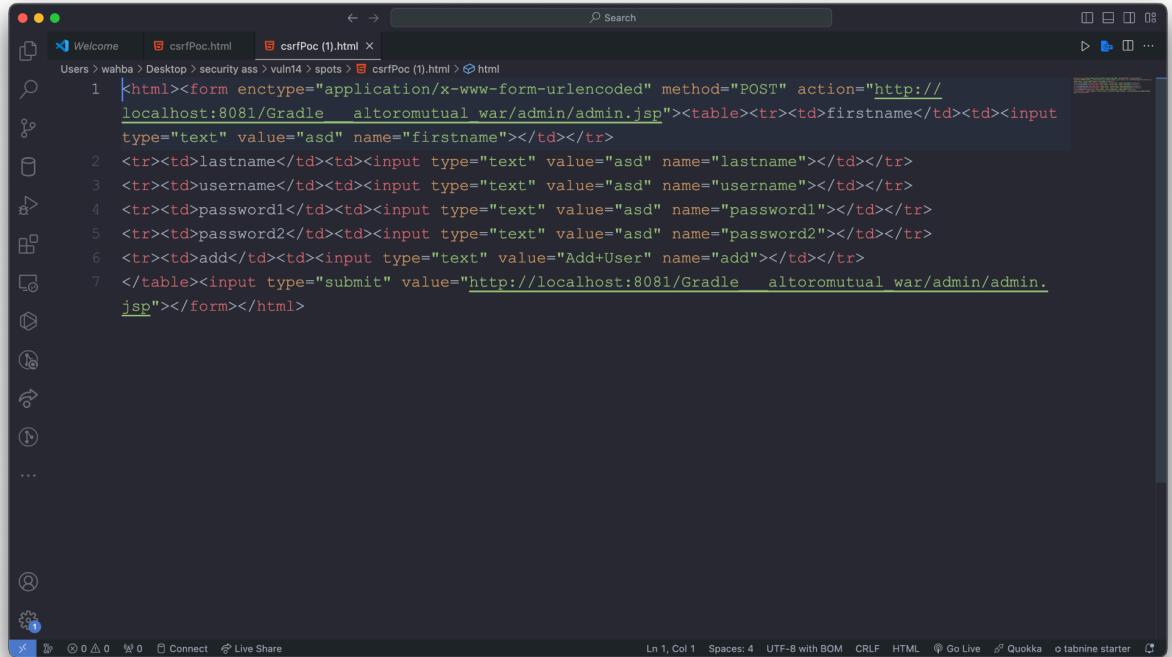
The screenshot shows the Altoro Mutual Online Banking Login page. The header includes a Google Docs link, a logo, and navigation links for Sign In, Contact Us, Feedback, and Search. The main content area has tabs for PERSONAL and SMALL BUSINESS, both showing a list of services. The PERSONAL tab includes Deposit Product, Checking, Loan Products, Cards, Investments & Insurance, and Other Services. The SMALL BUSINESS tab includes Deposit Products, Lending Services, Cards, Insurance, Retirement, and Other Services. A sidebar on the left lists INSIDE ALTORO MUTUAL links such as About Us, Contact Us, Locations, Investor Relations, Press Room, Careers, and Subscribe. At the bottom, there are links for Privacy Policy, Security Statement, Server Status Check, REST API, and a copyright notice for 2008-2017 IBM Corporation. A note at the bottom right states: "This web application is open source! Get your copy from GitHub and take advantage of advanced features".

- Finding RE-TEST steps

## ID08

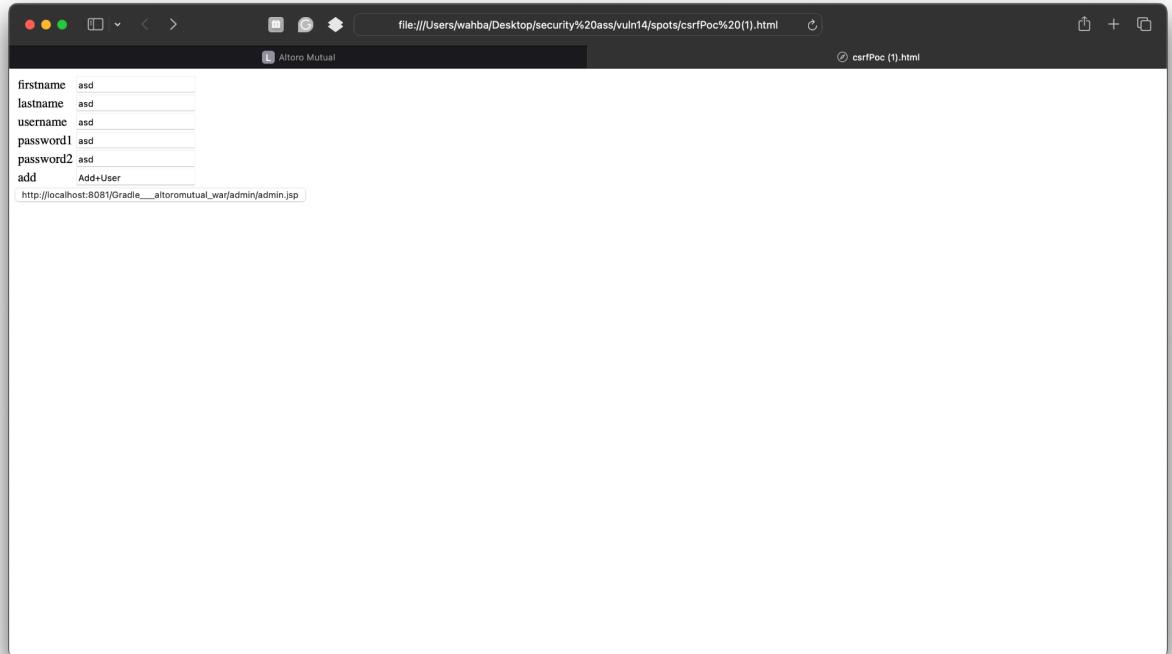
- Finding test steps
  - Let's prepare a CSRF PoC for add user functionality 😈

The screenshot shows a web browser window with three tabs: 'Altoro Mutual', 'security.love/CSRF-PoC-Gen', and 'Altoro Mutual'. The active tab displays the 'CSRF PoC Generator' tool. The interface includes fields for Method (set to POST), Encoding (set to application/x-www-form-urlencoded), Data (containing 'firstname=asd&lastname=asd&username=asd&password1=asd&password2=asd&addUser'), and URI (set to 'http://localhost:8081/Gradle\_\_altermutual\_war/admin/admin.jsp'). A 'Download CSRF PoC!' button is at the bottom. A red diagonal banner in the bottom right corner says 'Fork me on GitHub'.



```
1 <html><form enctype="application/x-www-form-urlencoded" method="POST" action="http://localhost:8081/Gradle__altermutual_war/admin/admin.jsp"><table><tr><td>firstname</td><td><input type="text" value="asd" name="firstname"></td></tr>
2 <tr><td>lastname</td><td><input type="text" value="asd" name="lastname"></td></tr>
3 <tr><td>username</td><td><input type="text" value="asd" name="username"></td></tr>
4 <tr><td>password1</td><td><input type="text" value="asd" name="password1"></td></tr>
5 <tr><td>password2</td><td><input type="text" value="asd" name="password2"></td></tr>
6 <tr><td>add</td><td><input type="text" value="Add+User" name="add"></td></tr>
7 </table><input type="submit" value="http://localhost:8081/Gradle__altermutual_war/admin/admin.jsp"/></form></html>
```

- Open the generated poc in the browser



- Go and press the button

- Added new User!!

The screenshot shows the 'Edit User Information' page of the Altoro Mutual demo site. The interface is divided into sections: 'I WANT TO ...' (with links to Account Summary, Recent Transactions, Transfer Funds, News Articles, and Site Language), 'ADMINISTRATION' (with a link to Edit Users), and the main 'Edit User Information' section.

**Edit User Information**

**Add an account to an existing user**

Users: admin Account Types: Savings Add Account

**Change user's password**

Users: admin Password: Confirm: Change Password

**Add a new user**

First Name: Last Name: Username: Password: Confirm: Add User

A note at the bottom of the form area states: "It is highly recommended that you leave the username as first initial last name."

At the bottom of the page, there is a footer with links to Privacy Policy, Security Statement, Server Status Check, REST API, and © 2024 Altoro Mutual, Inc. It also includes a note: "This web application is open source! Get your copy from GitHub and take advantage of advanced features". Below this, a red dotted box contains a disclaimer about the website being a demonstration and not a real banking site.

- Fixing steps
  - add csrf token in the admin.jsp and in the submit form using hidden input

```

<div id="wrapper" style="...>
<jsp:include page="/bank/membertoc.jspf"/>
<% User checkerUser = (User) request.getSession().getAttribute("user"); %>
<%if(checkerUser.getRole() == User.Role.Admin){%>
<td valign="top" colspan="3" class="bb">
<%@page import="com.ibm.security.appscan.altoromutual.util.ServletUtil"%>
<%@ page import="com.ibm.security.appscan.altoromutual.model.User" %>

<%
String[] users = ServletUtil.getBankUsers();
String csrfToken = java.util.UUID.randomUUID().toString();
session.setAttribute("csrf", csrfToken);
%>
<script language="javascript">
function confirmpass(myform)
{
    if (myform.password1.value.length && (myform.password1.value==myform.password2.value))
    {
        ...
    }
}
</script>

```

Services All Services Tomcat 8.5.100

Tomcat Localhost Log Tomcat Catalina Log

10-May-2024 23:26:43.115 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 23:26:43.127 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 23:26:43.128 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 23:26:43.204 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 23:26:43.205 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 23:26:43.211 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 23:26:43.211 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 23:26:43.219 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

```

<td>
<input type="text" name="username">
</td>
<td>
<input type="password" name="password1">
<br>
<input type="password" name="password2">
</td>
<td>
<input type="hidden" name="csrfToken" value=<%=csrfToken%>>
<input type="submit" name="add" value="Add User">

```

Services All Services Tomcat 8.5.100

Tomcat Localhost Log Tomcat Catalina Log

10-May-2024 23:26:43.115 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 23:26:43.127 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 23:26:43.128 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 23:26:43.204 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 23:26:43.205 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 23:26:43.211 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 23:26:43.211 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

10-May-2024 23:26:43.219 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

- check-in admin servlet if the csrf token in session is equal to the token that came in the request then execute else redirect to login and release

the session

The screenshot shows the AltoroJ 3.2 IDE interface. The left pane displays a project structure for 'AltoroJ' with various servlet classes like AdminServlet, CCApplyServlet, FeedbackServlet, LoginServlet, RedirectServlet, SubscribeServlet, SurveyServlet, TransferServlet, and utility classes DBUtil and OperationsUtil. The right pane shows the code for AdminServlet.java, specifically the doPost method. The code handles a POST request, checks for a csrf token, and adds a user if the account type is valid. Below the code editor is a log viewer for Tomcat 8.5.100 showing startup logs. The bottom status bar indicates the code has 145 characters and 2 line breaks.

```
/*
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String message = null;
    String reqToken = request.getParameter("csrf");
    HttpSession session = request.getSession();
    if (reqToken == session.getAttribute("csrf")) {
        //add account
        if (request.getRequestURL().toString().endsWith("addAccount")) {
            String username = request.getParameter("username");
            String acctType = request.getParameter("acctType");
            if (username == null || acctType == null || username.trim().length() == 0 || acctType == null) {
                message = "An error has occurred. Please try again later.";
            } else {
                String error = DBUtil.addAccount(username, acctType);
                if (error != null)
                    message = error;
            }
        }
    }
}
```

- Finding RE-TEST steps
  - Go to the HTML poc generated

The screenshot shows a browser window for 'Altoro Mutual' with a URL of 'localhost:8081/Gradle...\_altoromutual\_war/admin/admin.jsp'. The page displays an 'Edit User Information' form. On the right, the Network tab of the developer tools is open, showing a list of requests. One request, 'admin.jsp', has its 'Form Data' expanded, revealing fields for 'firstname', 'lastname', 'username', 'password1', 'password2', and 'csrftoken'. The csrftoken value is highlighted in blue. The developer tools also show a timeline of network requests and a list of 6 requests with a total transfer size of 9.9 kB.

- the user will be logged out and redirected to login

**PERSONAL**

- Deposit Product
- Checking
- Loan Products
- Cards
- Investments & Insurance
- Other Services

**SMALL BUSINESS**

- Deposit Products
- Lending Services
- Cards
- Insurance
- Retirement
- Other Services

**INSIDE ALTORO MUTUAL**

- About Us
- Contact Us
- Locations
- Investor Relations
- Press Room
- Careers
- Subscribe

[Privacy Policy](#) | [Security Statement](#) | [Server Status Check](#) | [REST API](#) | © 2024 Altoro Mutual, Inc.

*This web application is open source! Get your copy from [GitHub](#) and take advantage of advanced features.*

The AltoroJ website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/appscan/>.

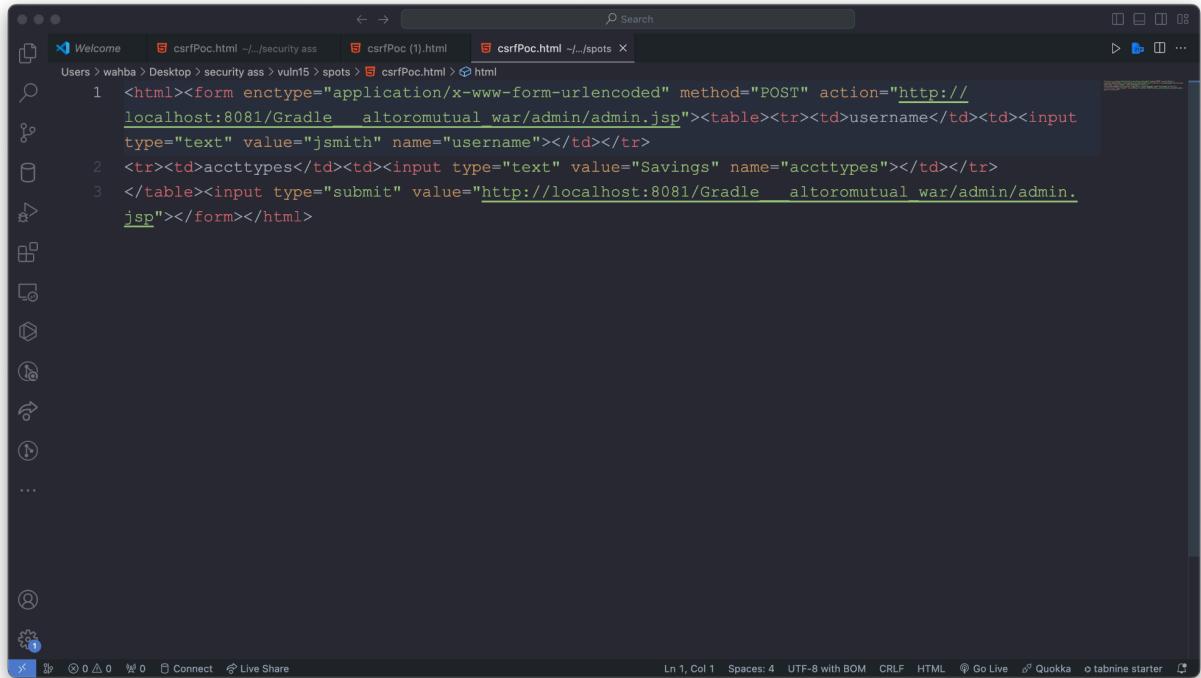
Copyright © 2008, 2017, IBM Corporation, All rights reserved. Copyright © 2017, 2024, HCL Technologies, Ltd., All rights reserved.

**DEMO SITE ONLY**

## **ID09**

- Finding test steps

- Let's prepare a CSRF PoC for add account to userfunctionality 😈

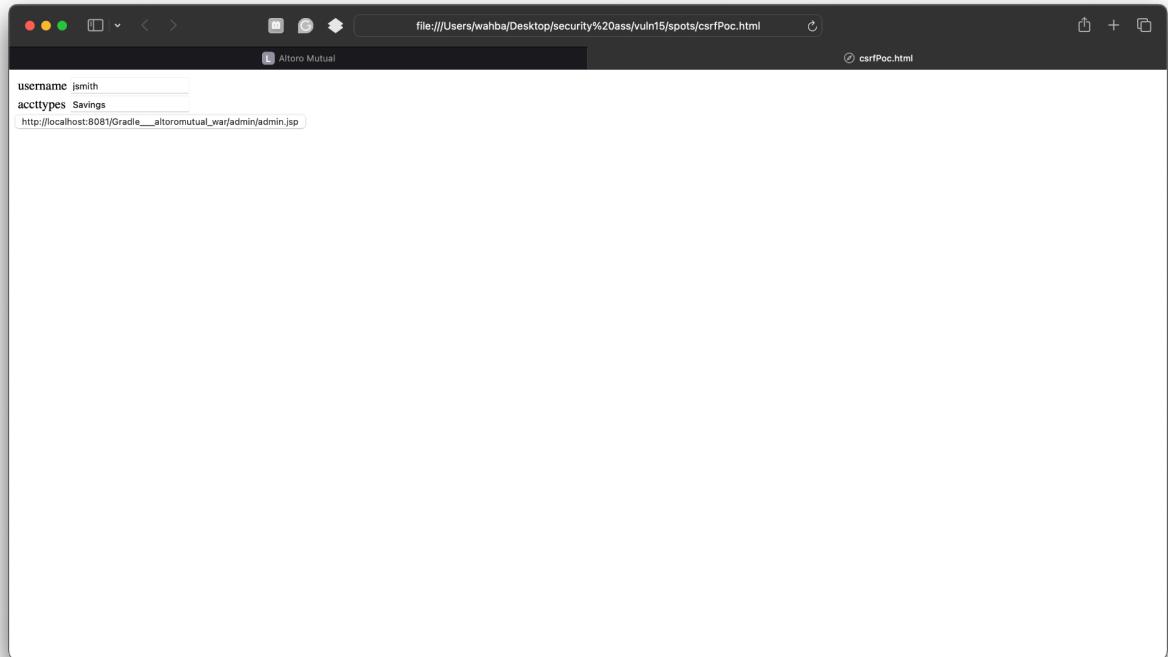


The screenshot shows a code editor with several tabs open. The active tab contains a POST request for a JSP page. The URL is `http://localhost:8081/Gradle__altermutual_war/admin/admin.jsp`. The form has two fields: `username` (value: `jsmith`) and `acctypes` (value: `Savings`). A `submit` button is present.

```

1 <html><form enctype="application/x-www-form-urlencoded" method="POST" action="http://
localhost:8081/Gradle__altermutual_war/admin/admin.jsp"><table><tr><td>username</td><td><input
type="text" value="jsmith" name="username"></td></tr>
2 <tr><td>acctypes</td><td><input type="text" value="Savings" name="acctypes"></td></tr>
3 </table><input type="submit" value="http://localhost:8081/Gradle__altermutual_war/admin/admin.
jsp"></form></html>
```

- Open the generated poc in the browser



- Go and press the button

- Added an account to user

**Edit User Information**

Add an account to an existing user

Users:  Account Types:

Change user's password

Users:  Password:  Confirm:

Add a new user

First Name:  Username:  Password:  Confirm:

It is highly recommended that you leave the username as first initial last name.

[Privacy Policy](#) | [Security Statement](#) | [Server Status Check](#) | [REST API](#) | © 2024 Altoro Mutual, Inc. *This web application is open source! Get your copy from [GitHub](#) and take advantage of advanced features*

The AltoroJ website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/appscan/>.

- Fixing steps
  - add csrf token in the admin.jsp and in the submit form using hidden input

```


<jsp:include page="/bank/membertoc.jspf"/>
    <% User checkerUser = (User) request.getSession().getAttribute("user"); %>
    <%if(checkerUser.getRole() == User.Role.Admin){%>
        <%@page import="com.ibm.security.appscan.altoromutual.util.ServletUtil"%>
        <%@ page import="com.ibm.security.appscan.altoromutual.model.User" %>
    <%>
        String[] users = ServletUtil.getBankUsers();
        String csrfToken = java.util.UUID.randomUUID().toString();
        session.setAttribute("csrf", csrfToken);
    <%>
        <script language="javascript">
            function confirmpass(myform)
            {
                if (myform.password1.value.length && (myform.password1.value==myform.password2.value))
                    ...
            }
        </script>
    </div#Wrapper> <td.bb> <script> confirmpass()


```

```

        <option value="<%>%><=%><=%>"><=%><=%></option>
    </select>
</td>
<td>
    <Select name="acctypes">
        <option Value="Checking">Checking</option>
        <option Value="Savings" Selected>Savings</option>
        <option Value="IRA">IRA</option>
    </Select></td>
<td>
    <input type="hidden" name="csrfToken" value="<%>${csrfToken}%>">
    <input type="submit" value="Add Account">
</td>
</tr>
</form>

```

- check-in admin servlet if the csrf token in session is equal to the token that came in the request then execute else redirect to login and release

the session

The screenshot shows the AltroJ 3.2 IDE interface. On the left is a project tree with various servlet files like AdminServlet, CCApplyServlet, FeedbackServlet, LoginServlet, RedirectServlet, SubscribeServlet, SurveyServlet, TransferServlet, and several utility classes. The main editor window displays AdminServlet.java, specifically the doPost method. The code handles a POST request, checks for a csrf token, and adds a user if the account type is valid. The Tomcat Localhost Log tab at the bottom shows deployment and startup logs for Tomcat 8.5.100. The status bar indicates the code has 145 characters and 2 line breaks.

```
/*
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String message = null;
    String reqToken = request.getParameter("csrf");
    HttpSession session = request.getSession();
    if (reqToken == session.getAttribute("csrf")) {
        //add account
        if (request.getRequestURL().toString().endsWith("addAccount")) {
            String username = request.getParameter("username");
            String acctType = request.getParameter("acctypes");
            if (username == null || acctType == null || username.trim().length() == 0 || acctType == null) {
                message = "An error has occurred. Please try again later.";
            } else {
                String error = DBUtil.addAccount(username, acctType);
                if (error != null)
                    message = error;
            }
        }
    }
}
```

- Finding RE-TEST steps
  - Go to the HTML poc generated

The screenshot shows a browser window for Altoro Mutual. The URL is localhost:8081/Gradle....\_altoromutual\_war/admin/admin.jsp. The page displays the 'Edit User Information' form. The Network tab in the developer tools shows a list of requests, including admin.jsp, logo.gif, header\_pic.jpg, pf\_lock.gif, and gradient.jpg. The Headers and Payload sections show form data being sent, including 'username: sspeed', 'acctypes: Savings', and 'csrftoken: ec959582-1d3d-4dd0-8732-1fd175b1c218'. The developer tools also show a warning about non-unique IDs for the 'username' field.

- the user will be logged out and redirected to login

**ONLINE BANKING LOGIN**

**PERSONAL**

- Deposit Product
- Checking
- Loan Products
- Cards
- Investments & Insurance
- Other Services

**SMALL BUSINESS**

- Deposit Products
- Lending Services
- Cards
- Insurance
- Retirement
- Other Services

**INSIDE ALTORO MUTUAL**

- About Us
- Contact Us
- Locations
- Investor Relations
- Press Room
- Careers
- Subscribe

**Online Banking Login**

Username:

Password:

This web application is open source! Get your copy from [GitHub](#) and take advantage of advanced features.

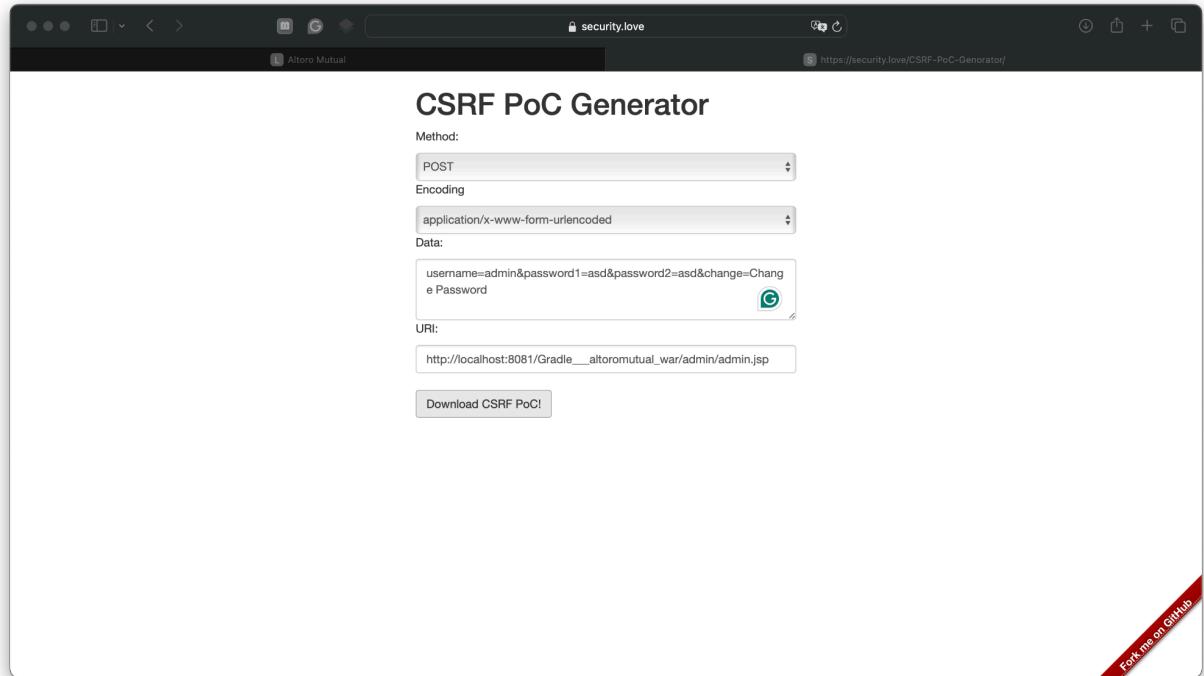
The AltoroJ website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/appscan/>.

Copyright © 2008, 2017, IBM Corporation, All rights reserved. Copyright © 2017, 2024, HCL Technologies, Ltd., All rights reserved.

## ID10

- Finding test steps

- Let's prepare a CSRF PoC for change password fo user functionality 😈



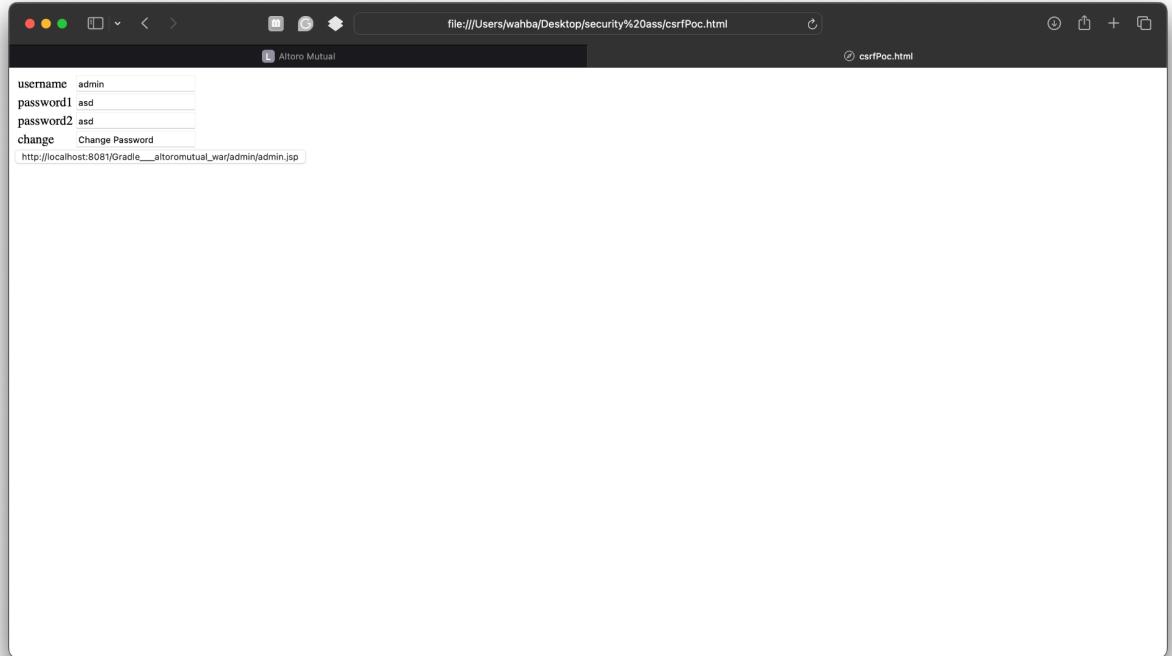
```

<html><form enctype="application/x-www-form-urlencoded" method="POST" action="http://localhost:8081/Gradle__altoromutual_war/admin/admin.jsp"><table><tr><td>username</td><td><input type="text" value="admin" name="username"></td></tr>
<tr><td>password1</td><td><input type="text" value="asd" name="password1"></td></tr>
<tr><td>password2</td><td><input type="text" value="asd" name="password2"></td></tr>
<tr><td>change</td><td><input type="text" value="Change Password" name="change"></td></tr>
</table><input type="submit" value="http://localhost:8081/Gradle__altoromutual_war/admin/admin.jsp"></form></html>

```

The screenshot shows a Quokka browser window with a dark theme. The title bar says "Welcome" and the tab is "csrfPoc.html". The content area displays the generated HTML code for a CSRF exploit. The code is a POST form with the action URL set to "http://localhost:8081/Gradle\_\_altoromutual\_war/admin/admin.jsp". The form contains three text input fields ("username", "password1", "password2") and one text input field for "change". The "change" field has the value "Change Password". Below the form is a submit button with the value "http://localhost:8081/Gradle\_\_altoromutual\_war/admin/admin.jsp". The status bar at the bottom of the browser shows "Ln 1, Col 1" and other standard browser status indicators.

- Open the generated poc in the browser



- Go and press the button
- Password for user changed

**Edit User Information**

Add an account to an existing user

Users: admin Account Types: Savings

Change user's password

Users: admin Password: Confirm: Change Password

Add an new user

First Name: Username: Password: Confirm:

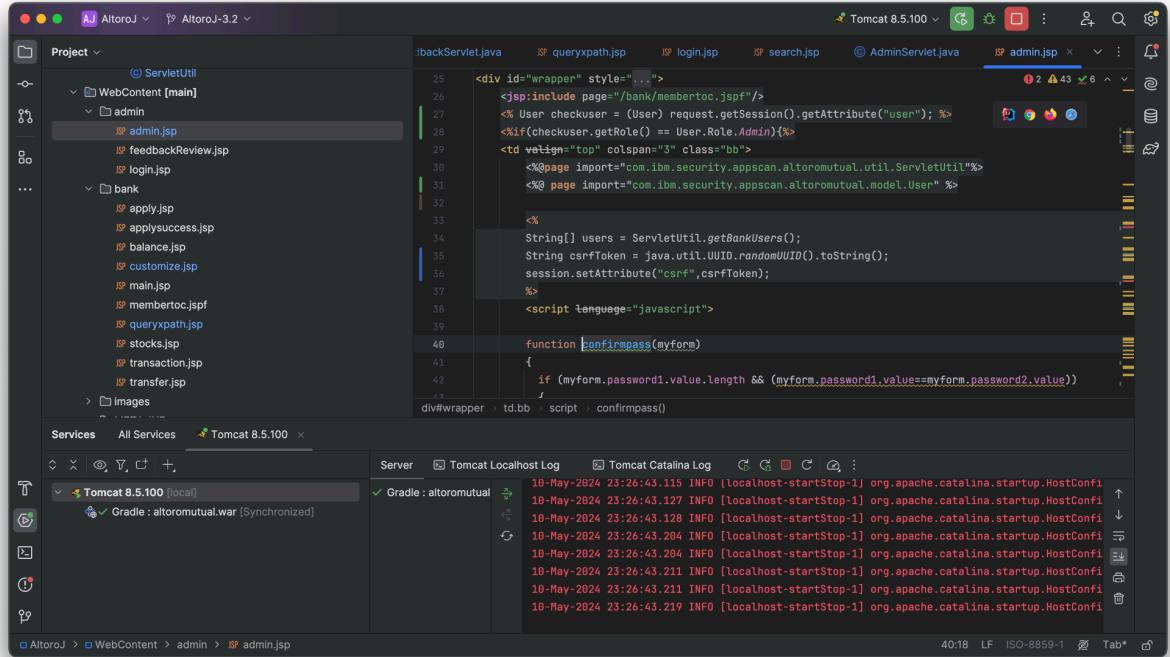
It is highly recommended that you leave the username as first initial last name.

Privacy Policy | Security Statement | Server Status Check | REST API | © 2024 Altoro Mutual, Inc. This web application is open source! Get your copy from GitHub and take advantage of advanced features

The Altoro3 website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/appscan/>.

- Fixing steps

- add csrf token in the admin.jsp and in the submit form using hidden input

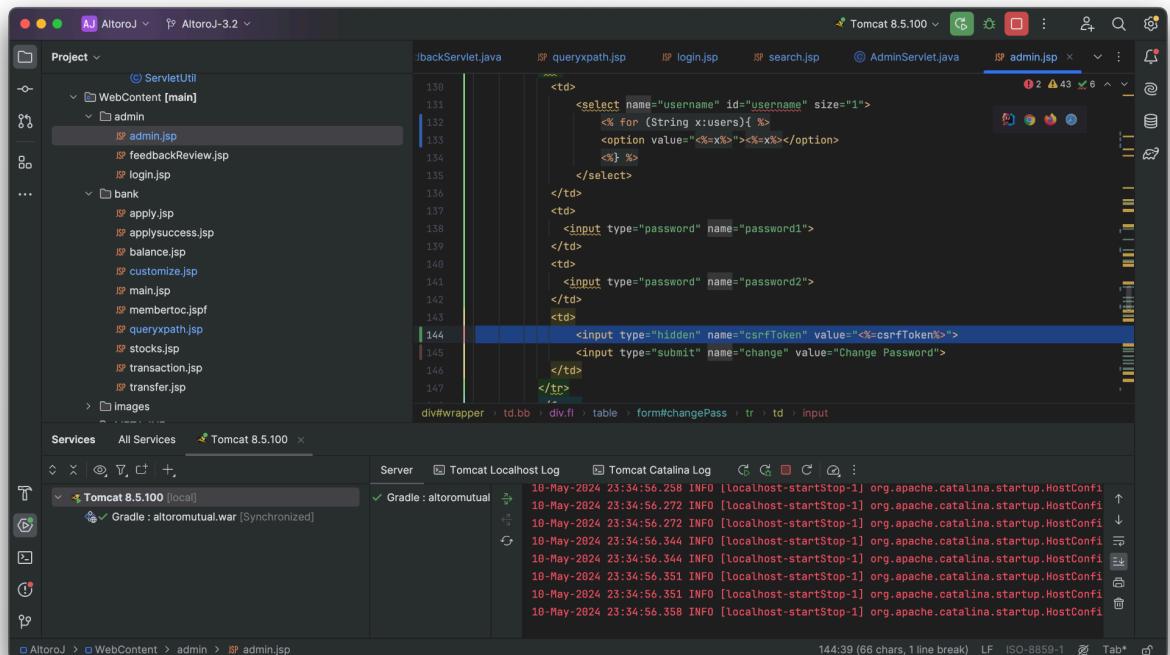


```

<div id="wrapper" style="...>
    <jsp:include page="/bank/membertoc.jspf"/>
    <% User checkuser = (User) request.getSession().getAttribute("user"); %>
    <%if(CheckUser.getRole() == User.Role.Admin){%>
        <%@page import="com.ibm.security.appscan.altoromutual.util.ServletUtil"%>
        <%@ page import="com.ibm.security.appscan.altoromutual.model.User" %>

        <%
        String[] users = ServletUtil.getBankUsers();
        String csrfToken = java.util.UUID.randomUUID().toString();
        session.setAttribute("csrf", csrfToken);
        %>
        <script language="javascript">
            function confirmPass(myForm)
            {
                if (myForm.password1.value.length && (myForm.password1.value==myForm.password2.value))
                    ...
            }
        </script>
    </td>

```



```

<td>
    <select name="username" id="username" size="1">
        <% for (String x:users){ %>
            <option value="<%=x%>"><%=x%></option>
        <% } %>
    </select>
</td>
<td>
    <input type="password" name="password1">
</td>
<td>
    <input type="password" name="password2">
</td>
<td>
    <input type="hidden" name="csrfToken" value="<%-csrfToken%>">
    <input type="submit" name="change" value="Change Password">
</td>
</tr>

```

- check-in admin servlet if the csrf token in session is equal to the token that came in the request then execute else redirect to login and release

the session

The screenshot shows the AltoroJ IDE interface. The left pane displays a project structure for 'AltoroJ' with various servlet classes like AdminServlet, CCApplyServlet, FeedbackServlet, LoginServlet, RedirectServlet, SubscribeServlet, SurveyServlet, TransferServlet, and utility classes DBUtil and OperationsUtil. The right pane shows the code for AdminServlet.java, specifically the doPost method. The code handles a POST request, checks for a csrf token, and adds a user if the account type is valid. Below the code editor is a log window for 'Tomcat Localhost Log' and 'Tomcat Catalina Log', showing deployment and startup messages. The bottom status bar indicates the code has 145 characters and 2 line breaks.

```
/*
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String message = null;
    String reqToken = request.getParameter("csrf");
    HttpSession session = request.getSession();
    if (reqToken == session.getAttribute("csrf")) {
        //add account
        if (request.getRequestURL().toString().endsWith("addAccount")) {
            String username = request.getParameter("username");
            String acctType = request.getParameter("acctType");
            if (username == null || acctType == null || username.trim().length() == 0 || acctType == null) {
                message = "An error has occurred. Please try again later.";
            } else {
                String error = DBUtil.addAccount(username, acctType);
                if (error != null)
                    message = error;
            }
        }
    }
}
```

- Finding RE-TEST steps
  - Go to the HTML poc generated

The screenshot shows a browser window for 'Altoro Mutual' at 'localhost:8081'. The page displays user information editing forms for existing users and new users. On the right, the developer tools Network tab is open, showing a timeline of requests. A specific request to 'admin.jsp' is selected, showing form data being sent with parameters: 'username: admin', 'password1: asd', 'password2: asd', and 'csrfToken: ddae2ebd-026a-4d63-8d70-95710b93b3a4'. The Network tab also shows other files like 'style.css', 'header\_pic.jpg', 'pf\_lock.gif', and 'gradient.jpg'. The bottom status bar indicates 6 requests and 10.1 kB transferred.

- the user will be logged out and redirected to login

**PERSONAL**

- Deposit Product
- Checking
- Loan Products
- Cards
- Investments & Insurance
- Other Services

**SMALL BUSINESS**

- Lending Products
- Lending Services
- Cards
- Insurance
- Retirement
- Other Services

**INSIDE ALTORO MUTUAL**

- About Us
- Contact Us
- Locations
- Investor Relations
- Press Room
- Careers
- Subscribe

[Privacy Policy](#) | [Security Statement](#) | [Server Status Check](#) | [REST API](#) | © 2024 Altoro Mutual, Inc.

*This web application is open source! Get your copy from [GitHub](#) and take advantage of advanced features*

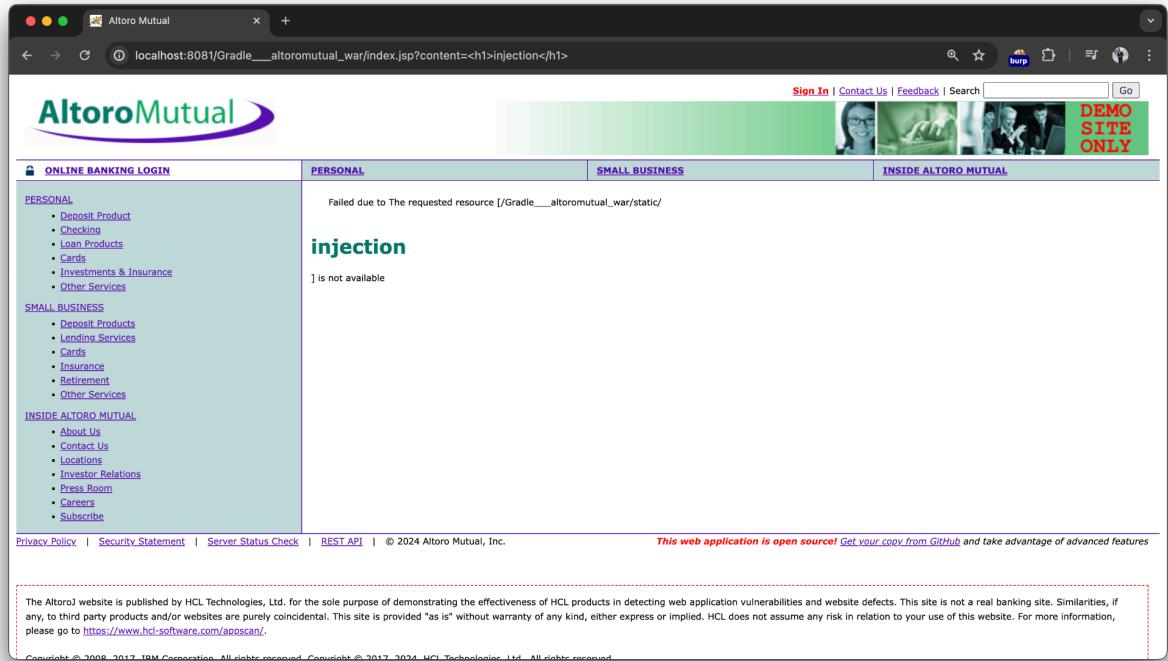
The AltoroJ website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/appscan/>.

Copyright © 2008, 2017, IBM Corporation, All rights reserved. Copyright © 2017, 2024, HCL Technologies, Ltd., All rights reserved.

## **ID11**

- Finding test steps
  - In the index screens(not protected), While routing to different pages content parameter is passed.

- let's inject any HTML code to it



- HTML injection detected 😞
  - Fixing steps
    - Go to index.jsp file in the source code
    - content parameter not sanitized

The screenshot shows the AltoroJ IDE interface. The top navigation bar includes tabs for 'AltoroJ' and 'AltoroJ-3.2'. The main workspace has a 'Project' view on the left listing various JSP files and configuration files like 'app.properties', 'ibm-web-ext.xml', and 'web.xml'. The central editor window displays the content of 'index.jsp'. The code includes JSP tags and Java code, such as:

```
<div id="wrapper" style="...>
<jsp:include page="toc.jsp"/>
<td valign="top" colspan="3" class="bb">
<%
java.lang.String content = request.getParameter("content");
if (content == null)
    content = "default.htm";
else
    content = request.getParameter("content");%>
    if (ServletUtil.isAppPropertyTrue("advancedStaticPageProcessing")){
        String path = request.getSession().getServletContext().getRealPath("/static");
    %>
    <% try { %>
        <%>
```

The bottom section shows the 'Services' view, which lists 'All Services' and 'Tomcat 8.5.100'. The 'Tomcat 8.5.100' service is expanded, showing its logs. The 'Localhost Log' tab displays several INFO messages from the Catalina startup process, such as:

```
18-May-2024 17:35:29.331 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig
18-May-2024 17:35:29.344 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig
18-May-2024 17:35:29.344 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig
18-May-2024 17:35:29.469 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig
18-May-2024 17:35:29.469 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig
18-May-2024 17:35:29.415 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig
18-May-2024 17:35:29.415 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig
18-May-2024 17:35:29.421 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig
```

- Go to ServletUtil to implement the sanitize function for inputs

The screenshot shows the Eclipse IDE interface with the following details:

- Project View (left):** Shows the project structure with files like `app.properties`, `index.jsp`, `ServletUtil.java`, `login.jsp`, `web.xml`, `status_check.jsp`, and `AdminServlet.java`.
- Code Editor (right):** Displays the `ServletUtil.java` file. The cursor is at line 410. The code handles exception handling, logs errors using Log4AltoroJ, prints stack traces, and initializes Swagger properties. It includes a `sanitizeInput` static method for replacing special characters.

- Use the sanitizeInput function implemented to sanitize content parameter against HTML injection

The screenshot shows the Eclipse IDE interface with several open files:

- Project View:** Shows the project structure with files like app.properties, ibm-web-ext.xml, init.log, web.xml, disclaimer.htm, feedback.jsp, feedbacksuccess.jsp, footer.jspf, header.jspf, high\_yield\_investments.htm, index.jsp, login.jsp, notfound.jsp, retirement.htm, search.jsp, status\_check.jsp, style.css, subscribe.jsp, subscribe.swf, survey\_questions.jsp, toc.jspf, .classpath, .gitignore, .project, build.gradle, gradlew, gradlew.bat, and LICENSE.
- Editor View:** The current file is `index.jsp`, which contains JSP code. It includes logic to handle content parameters and invoke `ServletUtil.sanitizeInput`. It also contains a block of code for advanced static page processing involving command-line execution via `Runtime.getRuntime().exec`.
- Tomcat 8.5.100 View:** Shows the running Tomcat instance.
- Search View:** A search bar at the top right.

- Finding RE-TEST steps

- Let's try the same injection on the same parameter in the index.jsp page

The screenshot shows a web browser window for 'Altoro Mutual' at the URL `localhost:8081/Gradle___altoromutual_war/index.jsp?content=<h1>injection</h1>`. The page displays a navigation bar with links for 'Sign In', 'Contact Us', 'Feedback', and a search bar. A banner on the right side of the header area says 'DEMO SITE ONLY'. The main content area has three tabs: 'ONLINE BANKING LOGIN', 'PERSONAL', and 'SMALL BUSINESS'. The 'PERSONAL' tab is active, showing a list of services: Deposit Product, Checking, Loan Products, Cards, Investments & Insurance, and Other Services. The 'SMALL BUSINESS' tab shows a list: Deposit Products, Lending Services, Cards, Insurance, Retirement, and Other Services. The 'INSIDE ALTORO MUTUAL' tab lists: About Us, Contact Us, Locations, Investor Relations, Press Room, Careers, and Subscribe. A message in the center of the page states: 'Failed due to The requested resource [/Gradle\_\_\_altoromutual\_war/static/] is not available'. At the bottom of the page, there is a note: 'This web application is open source! Get your copy from GitHub and take advantage of advanced features'. Below this, a red dashed box contains a copyright notice: 'The AltoroJ website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/appscan/>'. Another small note at the very bottom says: 'Copyright © 2008-2017 IBM Corporation. All rights reserved. Copyright © 2017-2024 HCL Technologies Ltd. All rights reserved.'

- Resolved 😊

## ID12

- Finding test steps

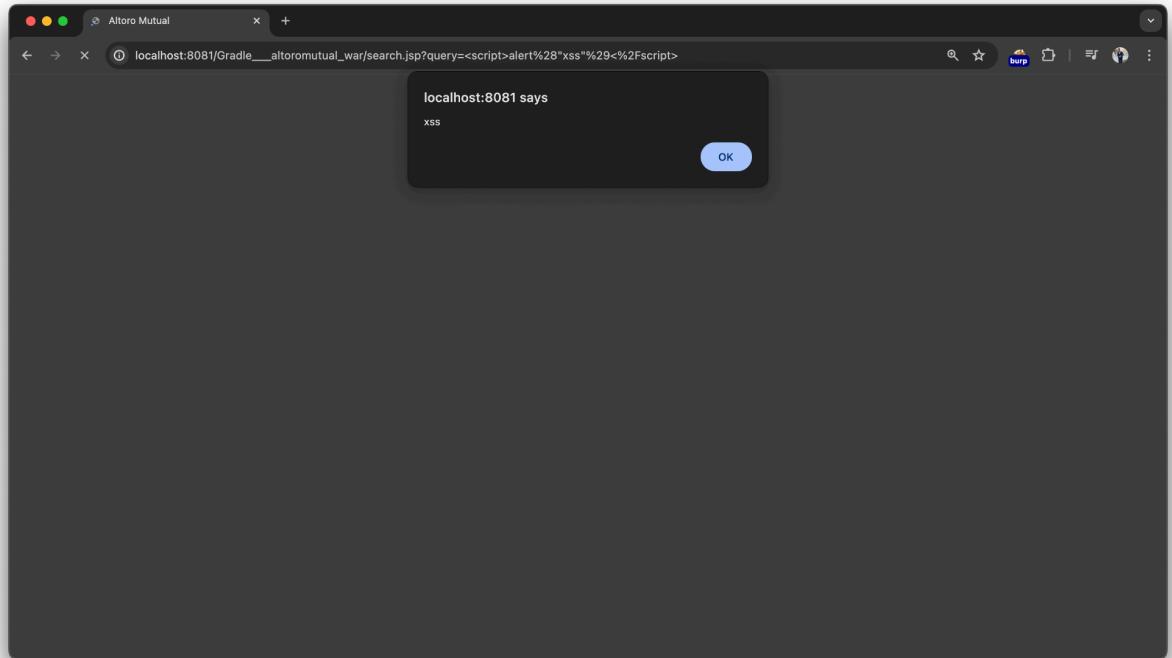
- In the index page go to the search in top right of the page

The screenshot shows a web browser window for 'Altoro Mutual' at localhost:8081. The search bar contains the query: <script>alert("xss")</script>. The search results page displays a 'Search Results' section with the message: 'No results were found for the query:'. Below this, there is a note: 'This web application is open source! Get your copy from GitHub and take advantage of advanced features'. At the bottom, a red box highlights a copyright notice: 'The AltoroJ website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/appscan/>. Copyright © 2008-2017 IBM Corporation. All rights reserved. Copyright © 2017-2024 HCL Technologies Ltd. All rights reserved.'

- Try to write a JS script with: <script>alert("xss")</script>

The screenshot shows a web browser window for 'Altoro Mutual' at localhost:8081. The search bar contains the query: <script>alert("xss")</script>. The main content area shows various service sections like Online Banking, Personal, Small Business, and Inside Altoro Mutual. Each section has a list of links. The 'PERSONAL' section includes a 'Real Estate Financing' image featuring a couple in front of a house. The 'SMALL BUSINESS' section includes a 'Business Credit Cards' image showing several credit cards. The 'INSIDE ALTORO MUTUAL' section includes a 'Retirement Solutions' image showing a group of people. A red box highlights the same copyright notice as the previous screenshot.

- XSS detected and executed the alert function



- Fixing steps
    - Go to search.jsp function in the source code
    - the query parameter is not sanitized or encoded

The screenshot shows the Eclipse IDE interface with the AltoroJ project open. The left sidebar displays the project structure, including files like app.properties, web.xml, and various JSP files. The main editor area shows the content of search.jsp, which includes JSP code for searching a database and displaying results. The status bar at the bottom indicates the file is 32:41 (48 chars, 1 line break) long.

```
<%>
<jsp:include page="header.jspf"/>

<div id="wrapper" style="...>
    <jsp:include page="toc.jspf"/>
    <td valign="top" colspan="3" class="bb">
        <@page import="com.ibm.security.appscan.altoromutual.util.ServletUtil"%>
        ...
        <%
            String query = request.getParameter("query");
            String[] results = null;
            if (query != null && query.trim().length() > 0)
                results = ServletUtil.searchSite(query, request.getSession().getServletContext().getRealPath("..."));
        %>
        ...
        <div class="fl" style="...>
            <h1>Search Results</h1>
            <p>No results were found for the query:<br /><br />
                <%= query %>
            </p>
        </div>
    </td>
</div>
<jsp:include page="footer.jspf"/>
```

- Use the same sanitize function implemented before in ServletUtil

```

22 <%>
23 <jsp:include page="header.jspf"/>
24 <div id="wrapper" style="...>
25 <jsp:include page="toc.jspf"/>
26 <td valign="top" colspan="3" class="bb">
27 <%@page import="com.ibm.security.appscan.altoromutual.util.ServletUtil"%>
28 <%
29 String query = ServletUtil.sanitizeInput(request.getParameter("query"));
30 String[] results = null;
31 if (query != null && query.trim().length()>0)
32     results = ServletUtil.searchSite(query, request.getSession().getServletContext().getRealPath("root"))
33 <%
34 <div class="fl" style="...>
35 <div class="fr" style="...>
36 </div>
37 </div>
38 </td>
39 </tr>

```

Tomcat Localhost Log

```

10-May-2024 18:58:47.058 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployWAR: Deploying web application archive /opt/tomcat/webapps/altoromutual.war
10-May-2024 18:58:47.073 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployWAR: Deployment of web application archive /opt/tomcat/webapps/altoromutual.war has finished in 1,617 ms
10-May-2024 18:58:47.073 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployWAR: Deploying web application archive /opt/tomcat/webapps/altoromutual.war
10-May-2024 18:58:47.141 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployWAR: Deployment of web application archive /opt/tomcat/webapps/altoromutual.war has finished in 1,617 ms
10-May-2024 18:58:47.141 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployWAR: Deploying web application archive /opt/tomcat/webapps/altoromutual.war
10-May-2024 18:58:47.147 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployWAR: Deployment of web application archive /opt/tomcat/webapps/altoromutual.war has finished in 1,617 ms
10-May-2024 18:58:47.147 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployWAR: Deploying web application archive /opt/tomcat/webapps/altoromutual.war
10-May-2024 18:58:47.155 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployWAR: Deployment of web application archive /opt/tomcat/webapps/altoromutual.war has finished in 1,617 ms

```

- Finding RE-TEST steps
  - Let's try the same script in the input search

Altoro Mutual

Sign In | Contact Us | Feedback | Search | Go

Search Results

No results were found for the query:

<script>alert("xss")</script>

[Privacy Policy](#) | [Security Statement](#) | [Server Status Check](#) | [REST API](#) | © 2024 Altoro Mutual, Inc.

This web application is open source! Get your copy from [GitHub](#) and take advantage of advanced features

The AltoroJ website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/appscan/>.

Copyright © 2008-2017 IBM Corporation. All rights reserved. Copyright © 2017-2024 HCL Technologies, Ltd. All rights reserved.

- Encoded input and resolved XSS 😊

## ID13

- Finding test steps
  - Sign in in the site
  - Go to search articles route

The screenshot shows a web browser window for the 'Altoro Mutual' demo site. The URL in the address bar is `localhost:8081/Gradle__altoromutual_war/bank/queryxpath.jsp?content=queryxpath.jsp&query=asd`. The page has a green header bar with the 'AltoroMutual' logo and navigation links for 'Sign Off', 'Contact Us', 'Feedback', and a search bar. A banner on the right says 'DEMO SITE ONLY'. The main content area has tabs for 'MY ACCOUNT', 'PERSONAL', 'SMALL BUSINESS', and 'INSIDE ALTORO MUTUAL'. On the left, there's a sidebar with 'I WANT TO ...' and a list of links: 'View Account Summary', 'View Recent Transactions', 'Transfer Funds', 'Search News Articles', and 'Customize Site Language'. Below that is an 'ADMINISTRATION' section with a 'Edit Users' link. The central part of the page is titled 'Search News Articles' and contains a search form with a text input containing 'asd' and a 'Query' button. A message below the form says 'News title not found, try again'. At the bottom, there's a footer with links for 'Privacy Policy', 'Security Statement', 'Server Status Check', 'REST API', and copyright information: 'Copyright © 2008, 2017, IBM Corporation, All rights reserved. Copyright © 2017, 2024, HCL Technologies, Ltd., All rights reserved.' A note in the footer states: 'The AltoroJ website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/aposcan/>'.

- Let's inject a script in the search input

The screenshot shows a web browser window for 'Altoro Mutual' at [localhost:8081/Gradle\\_\\_alotoromutual\\_war/bank/queryxpath.jsp?content=queryxpath.jsp&query=asd](http://localhost:8081/Gradle__alotoromutual_war/bank/queryxpath.jsp?content=queryxpath.jsp&query=asd). The page displays a search form with the placeholder 'Search News Articles'. A user has injected the payload `<script>alert('reflectedxss')` into the search input field. The search results show a single entry: 'News title not found, try again'. The status bar at the bottom right indicates 'This web application is open source! Get your copy from GitHub and take advantage of advanced features'.

- The script is executed RXXS 😱

The screenshot shows a web browser window for 'Altoro Mutual' at [localhost:8081 says](http://localhost:8081/Gradle__alotoromutual_war/bank/queryxpath.jsp?content=queryxpath.jsp&query=asd). A modal dialog box titled 'localhost:8081 says' contains the text 'xss' and an 'OK' button. This indicates that the injected JavaScript code was successfully executed and alerted the user.

- Use BurbSuite to intercept the request

Burp Suite Community Edition v2024.3.1.4 - Temporary Project

Target: http://localhost:8081 | HTTP/1.1

**Requests**

```
1 GET /Graal__altoromutual_war/bank/queryxpath.jsp?content=queryxpath.jsp&query=asdh2k3f3Cscript%3Ealert%28%22xss%22%29%3C%2Fscript%3E
```

**Response**

```
</td>
</li>
</span>
</td>
<!-- MEMBER_TOC_END -->
<td valign="top" colspan="3" class="bb">


### Search News Articles


<form id="QueryPath" method="get" action="http://localhost:8081/Graal__altoromutual_war/bank/queryxpath.jsp">
    Search our news articles database
    <br />
    <br />
    <input type="hidden" id="content" name="content" value="queryxpath.jsp"/>
    <input type="text" id="query" name="query" width="450" value="<xss>">
    <input type="submit" width="75" id="Button1" value="Query">
</form>
    alert("xss")
</script>
<br />
<input type="submit" width="75" id="Button1" value="Query">
<br />
<br />
    News title not found, try again
</div>
</td>


```

**Inspector**

Request attributes: 2 Request parameters: 2 Request body parameters: 0 Request cookies: 2 Request headers: 5 Response headers: 4

6,570 bytes | 11 millis | Memory: 319.3MB

- Fixing steps
  - Go to querxpath.jsp in bank route
  - The query input is not sanitized

AltoroJ 3.2.2

Tomcat 8.5.100

AdminLoginServlet.java customize.jsp feedback.jsp web.xml FeedbackServlet.java queryxpath.jsp User.java AdminServlet.java

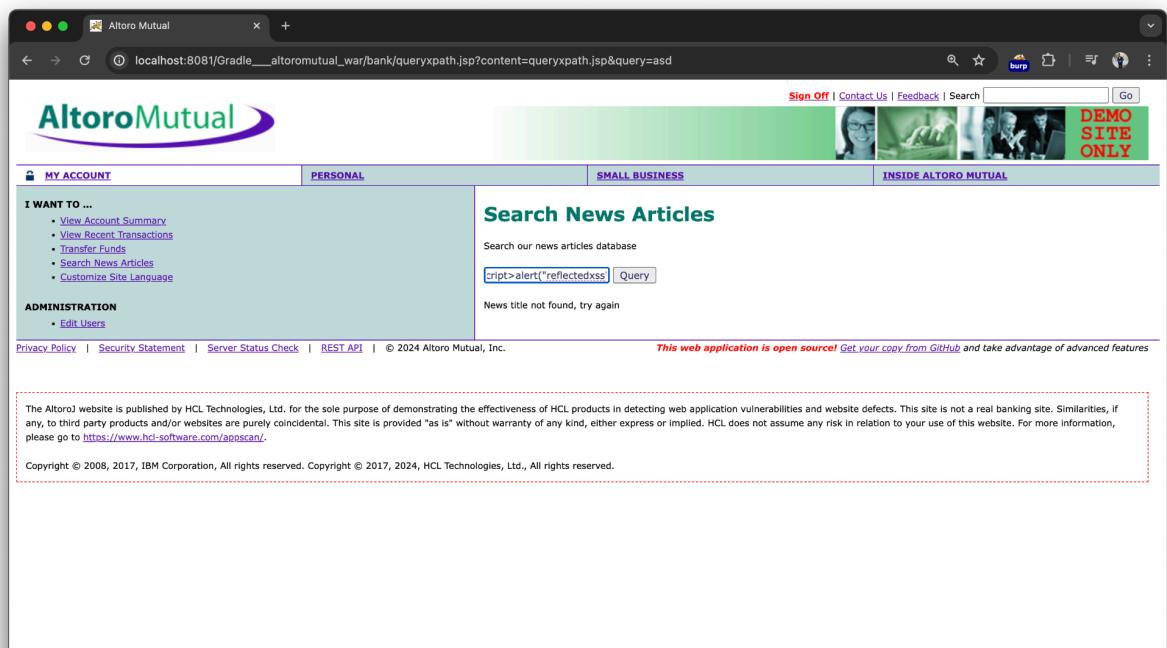
```
30 <div class="fl" style="width: 99%;>
31     <h3>Search News Articles</h3>
32     <form id="QueryPath" method="get" action="<%request.getRequestURL()%>">
33         Search our news articles database
34         <br /><br />
35         <input type="hidden" id="content" name="content" value="queryxpath.jsp"/>
36         <input type="text" id="query" name="query" width="450" value="<%=(request.getParameter("query")==null)? "Enter title (e.g. Watchfire") : request.getParameter("query")%>">
37         <input type="submit" width="75" id="Button1" value="Query">
38         <br /><br />
39
40         if (request.getParameter("query") != null) {
41             String[] results = ServletUtil.searchArticles(request.getParameter("query"), request.getSession().getServletContext().getRealPath("/pr/Docs.xml"));
42             if (results == null)
43                 out.println("News title not found, try again");
44             else {
45                 out.println("Found news title:<br/>");
46                 for(String result:results)
47                     out.println(result+"<br/>");
48             }
49         }
50     <%>
51
52     </form>
53 </div>
54
55
56
57 <div# wrapper > <td>
```

AltoroJ > WebContent > bank > queryxpath.jsp

54:13 LF ISO-8859-1 Tab\*

- Use the same sanitize function implemented before in ServletUtil

- Finding RE-TEST steps
    - Let's do the same steps and test the script in the HTML



- Resolved RXSS 😊

The screenshot shows a web browser window for 'Altoro Mutual' at localhost:8081. The page displays a navigation bar with links for 'Sign Off', 'Contact Us', 'Feedback', 'Search', and 'Go'. A banner on the right side of the header reads 'DEMO SITE ONLY'. Below the header, there are four small profile pictures. The main content area features a sidebar with sections for 'MY ACCOUNT', 'PERSONAL', 'SMALL BUSINESS', and 'INSIDE ALTORO MUTUAL'. Under 'MY ACCOUNT', there's a list titled 'I WANT TO ...' with items like 'View Account Summary', 'View Recent Transactions', 'Transfer Funds', 'Search News Articles', and 'Customize Site Language'. Under 'ADMINISTRATION', there's a link to 'Edit Users'. The central part of the page is titled 'Search News Articles' and contains a search input field with the value 'asd"><script>alert("xss")'. Below the search field, a message says 'News title not found, try again'. At the bottom of the page, there's a footer with links for 'Privacy Policy', 'Security Statement', 'Server Status Check', 'REST API', and copyright information: 'Copyright © 2008, 2017, IBM Corporation, All rights reserved. Copyright © 2017, 2024, HCL Technologies, Ltd., All rights reserved.' A red dashed box highlights the search input field.

## ID14

- Finding test steps

- Go login in to the site and go to Customize Language page

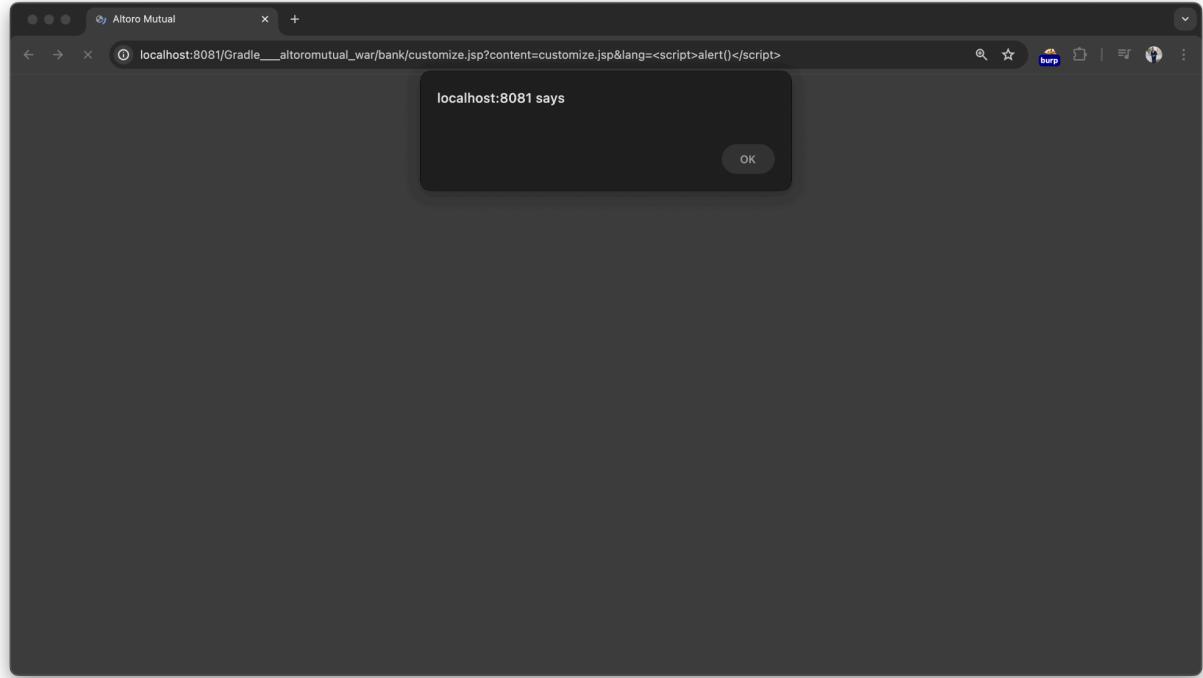
The screenshot shows the Altoro Mutual website's 'Customize Site Language' page. The URL is `localhost:8081/Gradle__altermutual_war/bank/customize.jsp?content=customize.jsp&lang=<script>alert()</script>`. The page displays a banner for 'DEMO SITE ONLY'. On the left, there's a sidebar with 'I WANT TO ...' and 'ADMINISTRATION' sections. The main content area shows the title 'Customize Site Language' and a note about changing the language. A red box highlights the injected script in the response body:

```
<script>alert()</script>
```

- Use BurbSuite to intercept the request with injected script in lang parameter

The screenshot shows the Burp Suite interface with the request and response tabs selected. The request shows a GET request to the 'customize.jsp' page with a 'lang' parameter containing the injected script. The response tab shows the modified HTML where the injected script has been executed, resulting in an alert box. The 'Inspector' panel on the right shows the modified HTML structure.

- It injected the script in the web page



- Fixing steps
  - Go to customize.jsp file inside the bank route
  - We found non non-sanitized lang parameter passed

```

    ...
    response.sendRedirect(content);
}

%>

<h1>Customize Site Language</h1>

<form method="post">
<p>
  Current Language: <%=(request.getParameter("lang")==null)?"":request.getParameter("lang")%>
</p>

<p>
  You can change the language setting by choosing:
</p>
<p>
  <a id="HyperLink1" href=".//customize.jsp?content=customize.jsp&lang=international">International</a>
  <a id="HyperLink2" href=".//customize.jsp?content=customize.jsp&lang=english">English</a>
</p>

```

The screenshot shows the IDE interface with the 'customize.jsp' file selected in the project tree. The code editor displays JSP and Java混用的代码。在 JSP 部分，`lang` 参数直接用于 `href` 属性中，没有进行适当的转义或验证。

- Use the sanitizeInput function in ServletUtil implemented before

```

36     response.sendRedirect(content);
37 }
38 }
39 }
40 ...
41 <h1>Customize Site Language</h1>
42 ...
43 <form method="post">
44 <p>
45 Current Language: <%=(request.getParameter("lang")==null)?"": ServletUtil.sanitizeInput(request.getParameter("lang"))%>
46 </p>
47 <p>
48 You can change the language setting by choosing:
49 </p>
50 <a id="HyperLink1" href=".customize.jsp?content=customize.jsp&lang=international">International</a>
51 <a id="HyperLink2" href=".customize.jsp?content=customize.jsp&lang=english">English</a>
52 </p>
53 </form>
54 </div>
55 </td>
56 </div>
57 <jsp:include page="/footer.jspf"/>
58 ...
59 ...
60 ...
61 ...

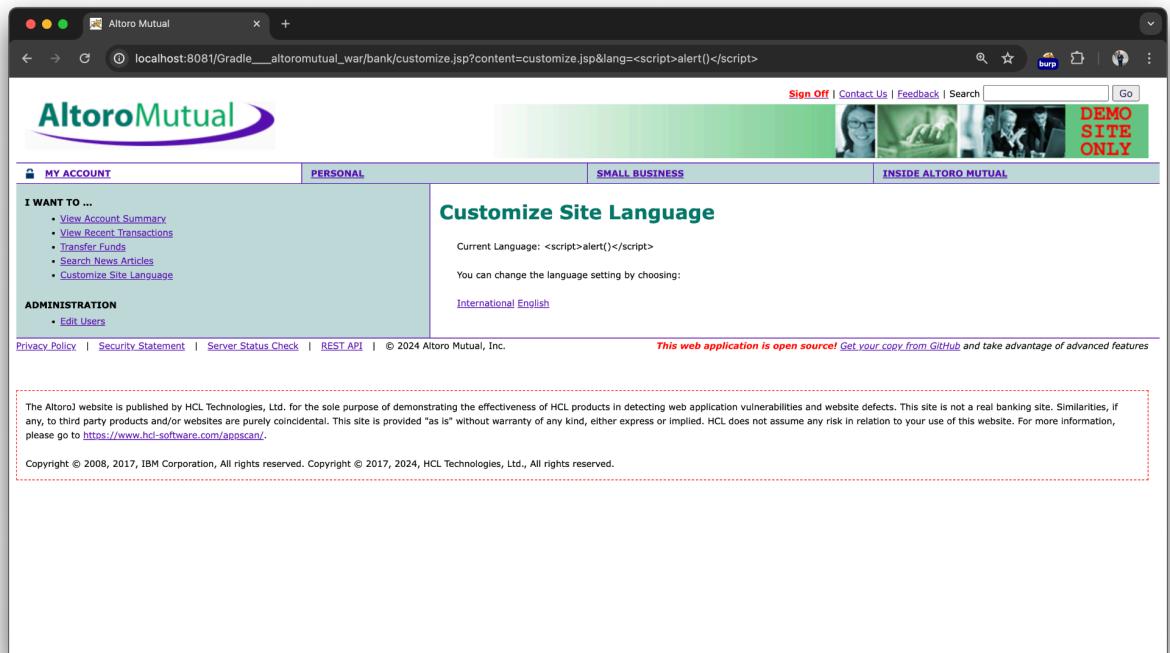
```

div#wrapper : td.bb : div.fl : form : p

AltoroJ > WebContent > bank > customize.jsp

50:15 LF ISO-8859-1 Tab\*

- Finding RE-TEST steps
  - Test the web page on the site manually first



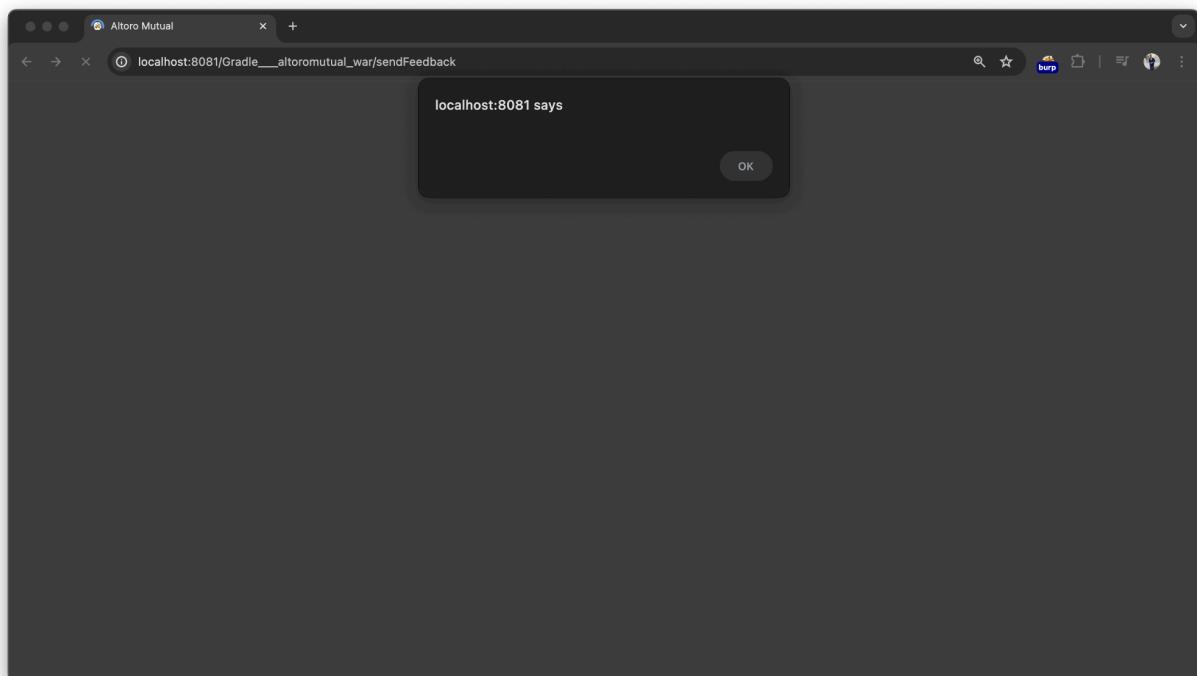
- Resolved RXSS 😊

## ID15

- Finding test steps
  - Go to feedback page and try inject a script in any input

The screenshot shows a web browser window for 'Altoro Mutual' at 'localhost:8081/Gradle\_\_altoromutual\_war/feedback.jsp'. The page has a navigation bar with links like 'Sign Off', 'Contact Us', 'Feedback', and a search bar. A banner at the top right says 'DEMO SITE ONLY'. The main content area is titled 'Feedback' and contains a form with fields for 'Your Name', 'Your Email Address', 'Subject', and a large 'Question/Comment' text area. In the 'Your Name' field, the value is '<script>alert()</script>'. The 'Question/Comment' area also contains the value 'asdf'. Below the form are 'Submit' and 'Clear Form' buttons. At the bottom of the page, there's a note about the site being a demo and a link to GitHub.

- The script is injected and executed after submit



- Fixing steps
  - Go to FeedbackServlet file in source code
  - The name input name is not sanitized or encoded

```

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    //the feedback is not actually submitted
    if (request.getParameter("comments") == null) {
        response.sendRedirect("index.jsp");
        return;
    }

    String name = request.getParameter("name");

    if (name != null){
        request.setAttribute("message_feedback", name);
        String email = request.getParameter("email_addr");
        String subject = request.getParameter("subject");
        String comments = request.getParameter("comments");
        //store feedback in the DB - display their feedback once submitted

        String feedbackId = OperationsUtil.sendFeedback(name, email, subject, comments);
        if (feedbackId != null) {
            request.setAttribute("feedback_id", feedbackId);
        }
    } else {
        request.removeAttribute("name");
    }

    RequestDispatcher dispatcher = request.getRequestDispatcher("/feedbacksuccess.jsp");
    dispatcher.forward(request, response);
}
}

```

- Use the sanitizeInput function implemented before in the ServletUtil

```

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    //the feedback is not actually submitted
    if (request.getParameter("comments") == null) {
        response.sendRedirect("index.jsp");
        return;
    }

    String name = Servletutil.sanitizeInput(request.getParameter("name"));

    if (name != null){
        request.setAttribute("message_feedback", name);
        String email = request.getParameter("email_addr");
        String subject = request.getParameter("subject");
        String comments = request.getParameter("comments");
        //store feedback in the DB - display their feedback once submitted

        String feedbackId = OperationsUtil.sendFeedback(name, email, subject, comments);
        if (feedbackId != null) {
            request.setAttribute("feedback_id", feedbackId);
        }
    } else {
        request.removeAttribute("name");
    }

    RequestDispatcher dispatcher = request.getRequestDispatcher("/feedbacksuccess.jsp");
    dispatcher.forward(request, response);
}
}

```

- Finding RE-TEST steps

- Let's retest the same script on the webpage

The screenshot shows a web browser window for 'Altoro Mutual' at 'localhost:8081/Gradle\_\_altermutual\_war/feedback.jsp'. The page has a header with 'Sign In | Contact Us | Feedback | Search [Go]' and a 'DEMO SITE ONLY' banner. The main content area is titled 'Feedback'. On the left, there's a sidebar with 'ONLINE BANKING LOGIN' and links for 'PERSONAL' (Deposit Product, Checking, Loan Products, Cards, Investments & Insurance, Other Services), 'SMALL BUSINESS' (Deposit Products, Lending Services, Cards, Insurance, Retirement, Other Services), and 'INSIDE ALTORO MUTUAL' (About Us, Contact Us, Locations, Investor Relations, Press Room, Careers, Subscribe). The right side contains a form for sending feedback. The 'To:' field contains '**Online Banking**'. The 'Your Name:' field contains '<script>alert()</script>'. The 'Your Email Address:' and 'Subject:' fields are empty. The 'Question/Comment:' text area is empty. Below the form are 'Submit' and 'Clear Form' buttons. At the bottom, there are links for 'Privacy Policy', 'Security Statement', 'Server Status Check', 'REST API', and copyright information: '© 2024 Altoro Mutual, Inc.' and 'This web application is open source! Get your copy from GitHub and take advantage of advanced features'. A note at the bottom states: 'The AltoroJ website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/appscan/>'.

- It's gone perfectly

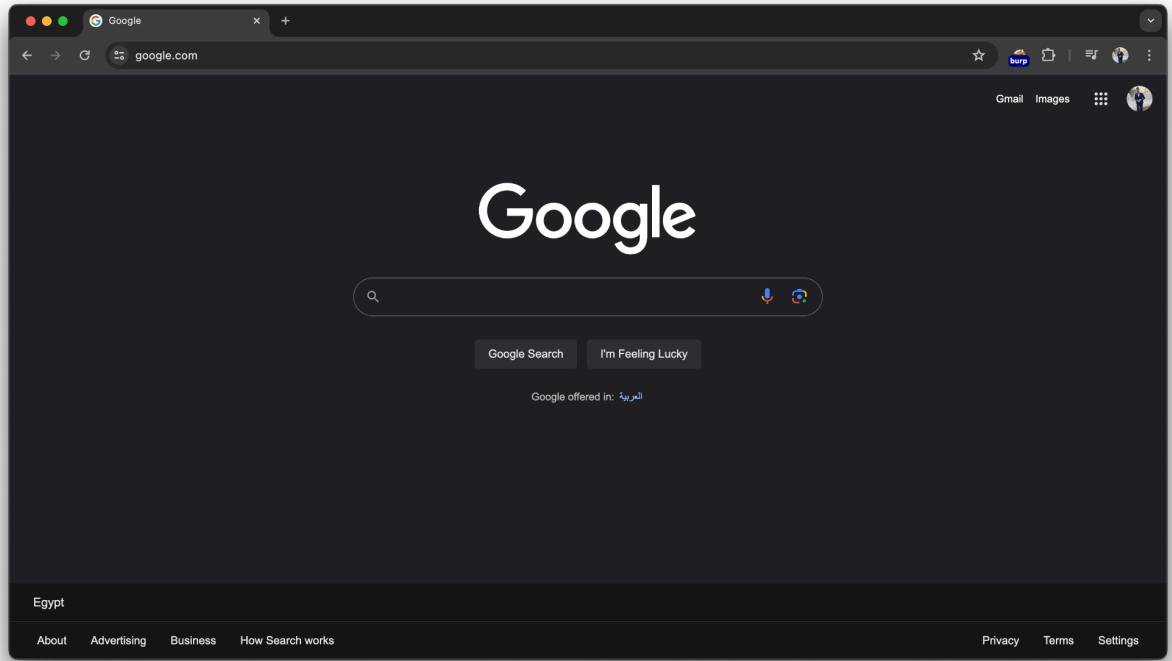
The screenshot shows a web browser window for 'Altoro Mutual' at 'localhost:8081/Gradle\_\_altermutual\_war/sendFeedback'. The layout is identical to the previous screenshot, with the 'Feedback' page. The right side now displays a 'Thank You' message: 'Thank you for your comments, <script>alert()</script>. They will be reviewed by our Customer Service staff and given the full attention that they deserve. However, the email you gave is incorrect () and you will not receive a response.' Below this message, the 'Question/Comment:' text area is empty. The rest of the page, including the sidebar and footer, remains the same as the previous screenshot.

**ID16**

- Finding test steps
  - Go and sign in
  - Go to customize language route
  - Try to change the lang parameter with any link in our case (<https://google.com>)

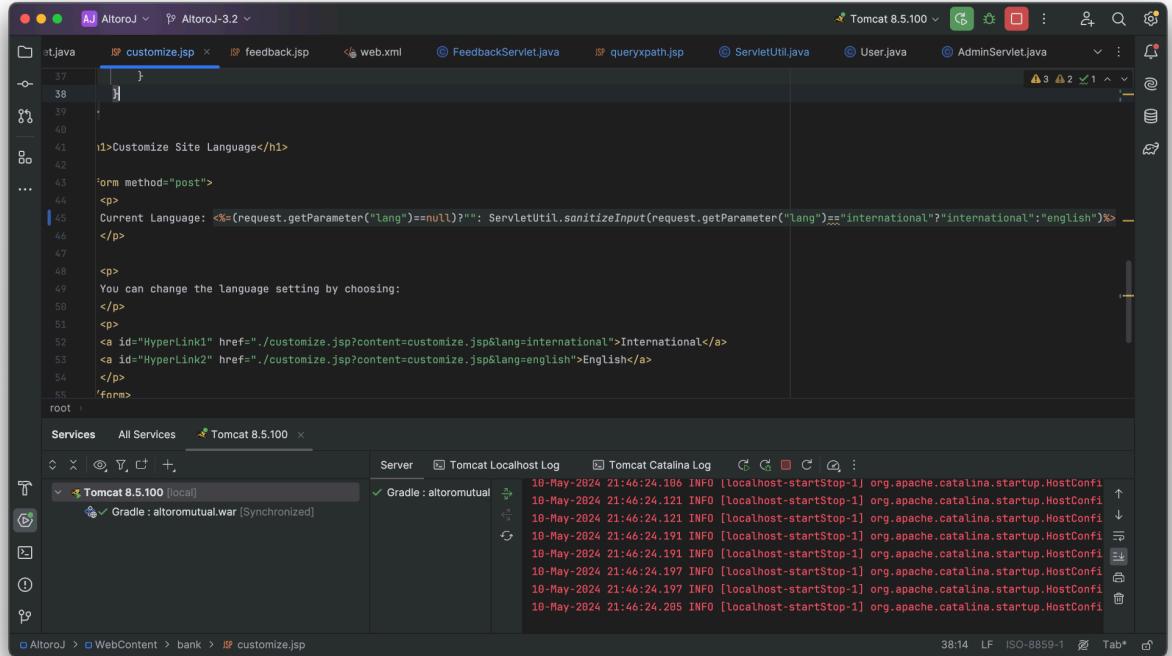
The Altoro Mutual website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/aposcan/>.

- The site is redirected to the specified link 😞



- Fixing steps
  - Go to customize.jsp file in bank route
  - We want to prevent redirection so escape or sanitize input, additionally we want to restrict the languages to be either “english” or

“international” as written in the code

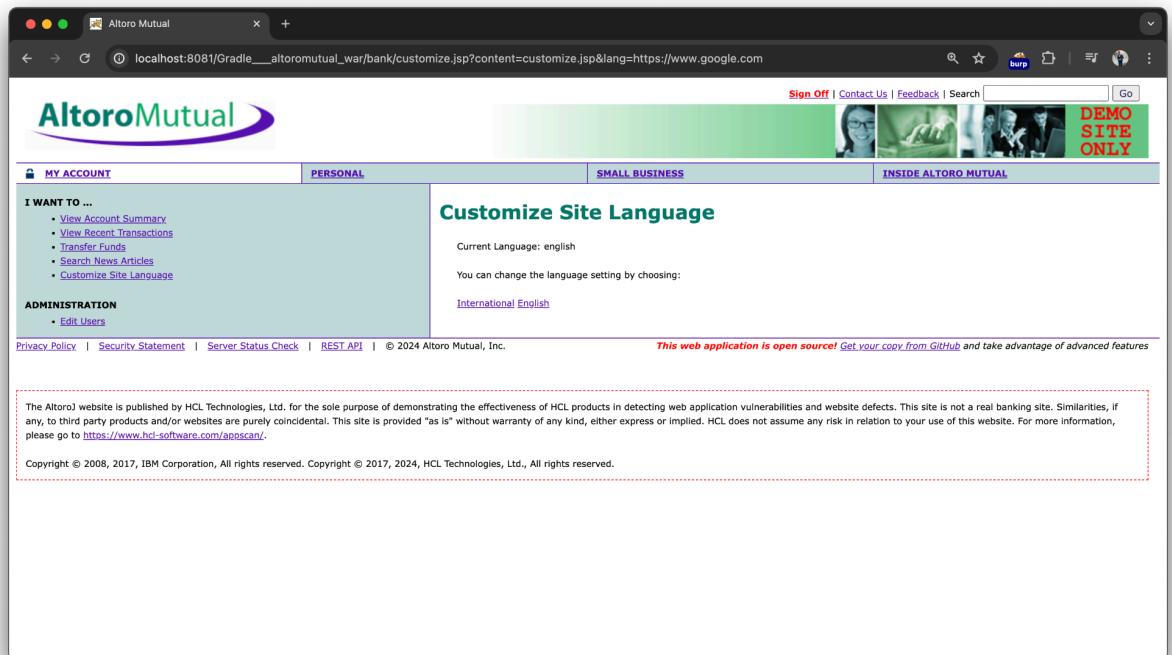


The screenshot shows the AltoroJ IDE interface. On the left, there's a code editor with Java code for a JSP file named 'customize.jsp'. The code includes a conditional statement based on the 'lang' parameter. On the right, there's a 'Services' panel showing 'Tomcat 8.5.100 [local]' and a 'Server' panel showing the 'Tomcat Localhost Log' and 'Tomcat Catalina Log'. The catalina log displays several INFO messages from May 2024, including startup logs for HostConfig and HostContext.

```
37 }  
38 }  
39 }  
40 }  
41 <h1>Customize Site Language</h1>  
42 ...  
43 <form method="post">  
44 <p>Current Language: <%=(request.getParameter("lang")==null)?:"": ServletUtil.sanitizeInput(request.getParameter("lang"))=="international"? "international": "english"%>  
45 </p>  
46 <p>  
47 You can change the language setting by choosing:  
48 </p>  
49 <p>  
50 <a id="HyperLink1" href=". /customize.jsp?content=customize.jsp&lang=international">International</a>  
51 <a id="HyperLink2" href=". /customize.jsp?content=customize.jsp&lang=english">English</a>  
52 </p>  
53 </form>  
54  
root
```

Services All Services Tomcat 8.5.100  
Tomcat Localhost Log Tomcat Catalina Log  
Tomcat 8.5.100 [local] Gradle : altoromutual  
Gradle : altoromutual.war [Synchronized]  
10-May-2024 21:46:24.106 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig  
10-May-2024 21:46:24.121 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig  
10-May-2024 21:46:24.121 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig  
10-May-2024 21:46:24.191 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig  
10-May-2024 21:46:24.191 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig  
10-May-2024 21:46:24.197 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig  
10-May-2024 21:46:24.197 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig  
10-May-2024 21:46:24.285 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig

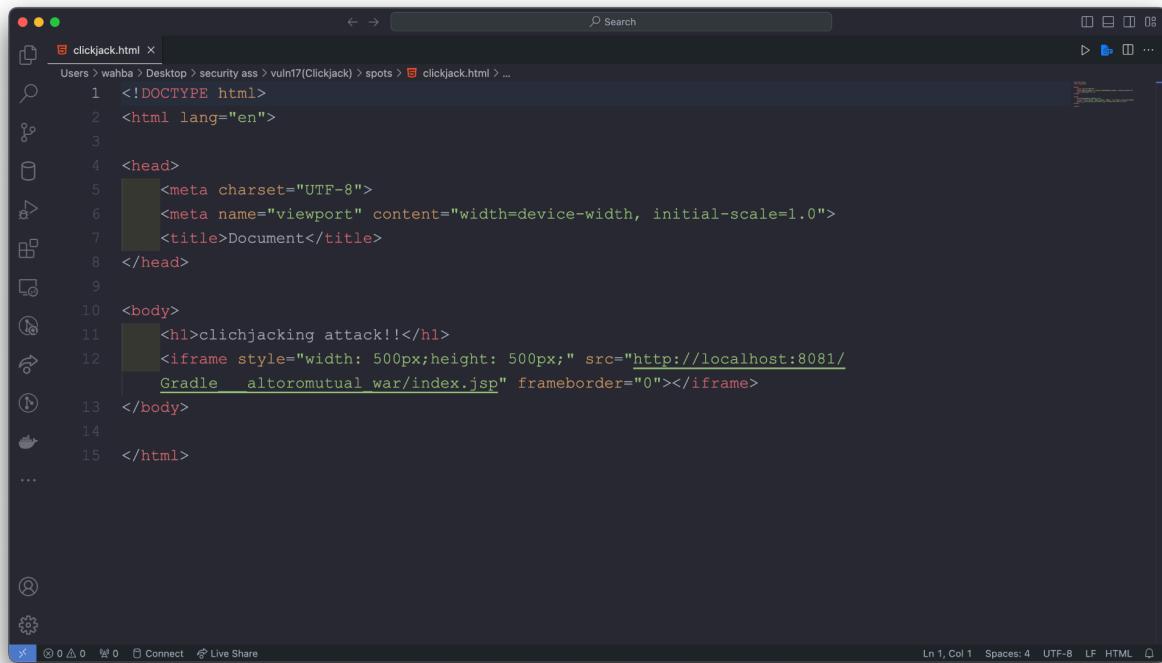
- Finding RE-TEST steps
  - Go and make the same request with the same payload of the link



- Redirection is resolved and default value is english as shown 

## ID17

- Finding test steps
  - Tried to get the base URL of the web page and inject it in an iframe in an HTML file to see if it causes the issue or not

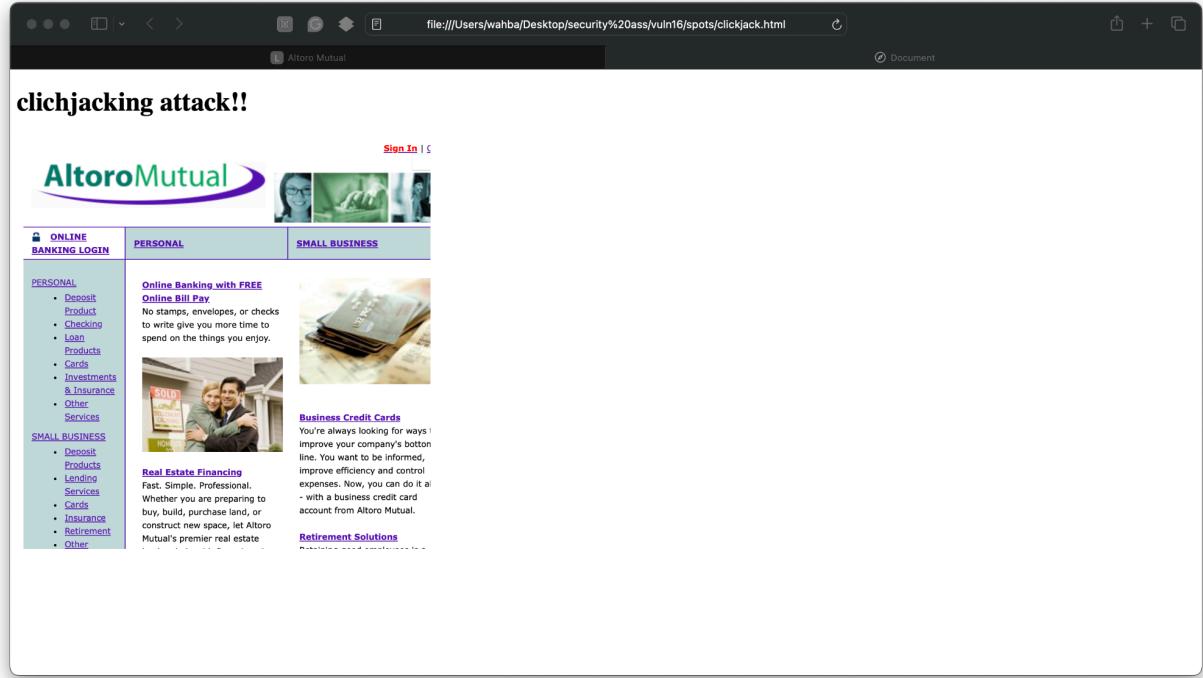


The screenshot shows a code editor window with the file 'clickjack.html' open. The code is as follows:

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9
10 <body>
11   <h1>clickjacking attack!!</h1>
12   <iframe style="width: 500px; height: 500px;" src="http://localhost:8081/
13     Gradle    altoromutual war/index.jsp" frameborder="0"></iframe>
14 </body>
15 </html>
...
```

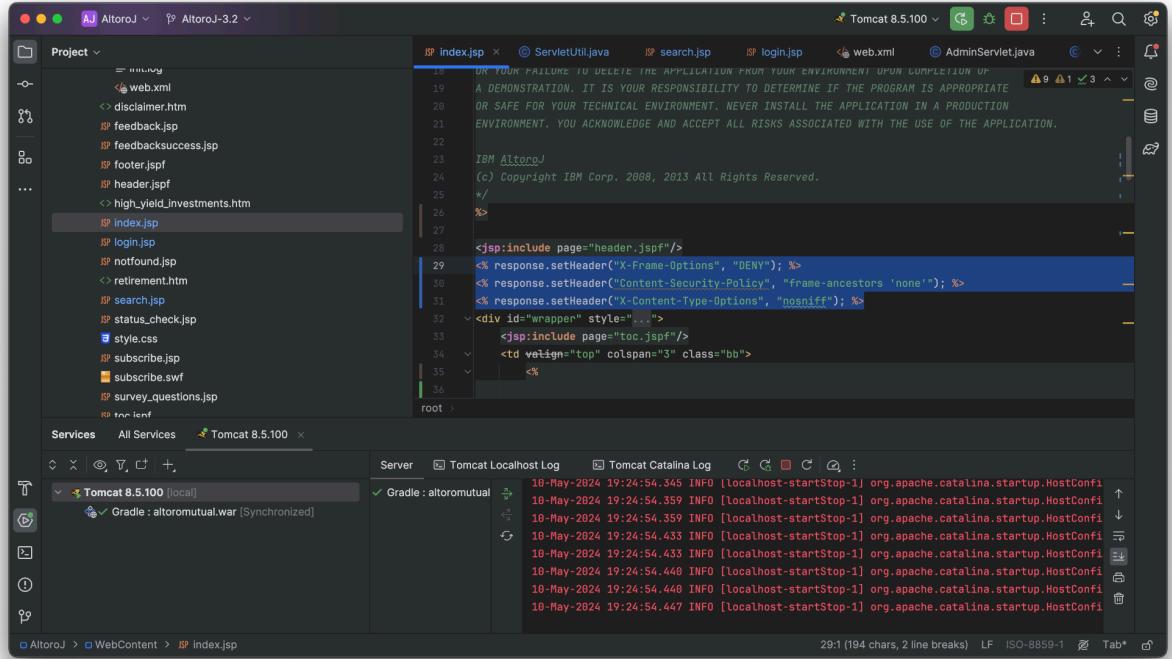
The code includes an `iframe` tag with a source URL pointing to a local host port 8081. The file path in the sidebar indicates it's located in a folder named 'vuln17(Clickjack) > spots'.

- Opened this HTML file in the browser and found it accessible



- Fixing steps
  - To mitigate clickjacking vulnerabilities we should set some response headers like `X-Frame-Options`, `Content-Security-Policy`, `X-Content-Type-Options`

- To avoid sniffing and clickjacking



```

16 OR YOUR FAILURE TO DELETE THE APPLICATION FROM YOUR ENVIRONMENT UPON COMPLETION OF
17 A DEMONSTRATION. IT IS YOUR RESPONSIBILITY TO DETERMINE IF THE PROGRAM IS APPROPRIATE
18 OR SAFE FOR YOUR TECHNICAL ENVIRONMENT. NEVER INSTALL THE APPLICATION IN A PRODUCTION
19 ENVIRONMENT. YOU ACKNOWLEDGE AND ACCEPT ALL RISKS ASSOCIATED WITH THE USE OF THE APPLICATION.
20
21 IBM AltoroJ
22 (c) Copyright IBM Corp. 2008, 2013 All Rights Reserved.
23 */
24 <%
25 %>
26 <jsp:include page="header.jspf"/>
27 <% response.setHeader("X-Frame-Options", "DENY"); %>
28 <% response.setHeader("Content-Security-Policy", "frame-ancestors 'none'"); %>
29 <% response.setHeader("X-Content-Type-Options", "nosniff"); %>
30 <div id="wrappern" style="...">
31   <jsp:include page="toc.jspf"/>
32   <td vAlign="top" colspan="3" class="bb">
33     <%
34       ...
35     <%
36   </div>

```

- additionally we can put the headers in each servlet to make all pages protected against this issue
- `response.setHeader("X-Frame-Options", "DENY");`
- `response.setHeader("Content-Security-Policy", "frame-ancestors 'none'");`
- `response.setHeader("X-Content-Type-Options", "nosniff");`
- Finding RE-TEST steps

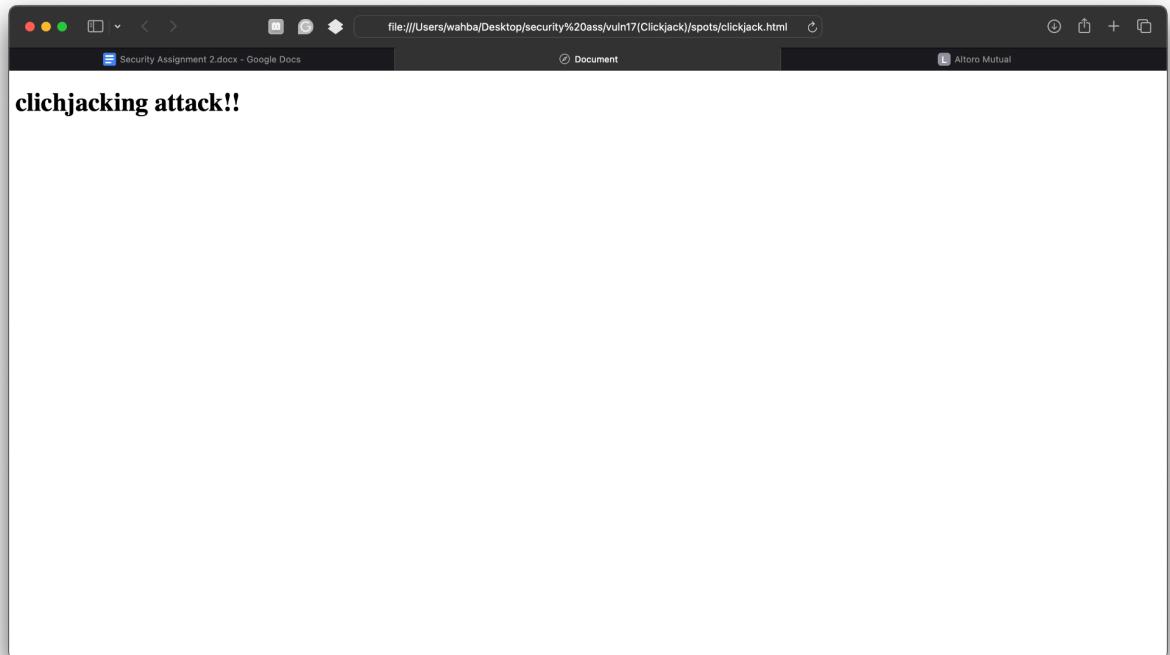
- Ensure that these headers are set in response headers lets intercept the request on BurbSuit

```

HTTP/1.1 200
Set-Cookie: JSESSIONID=B470E5E1B882C5FF2B9D766E89E06568; Path=/Gradle_____altermutual_war; HttpOnly
X-Frame-Options: DENY
Content-Security-Policy: frame-ancestors 'none'
Content-Type: text/html; charset=ISO-8859-1
Date: Fri, 10 May 2024 16:26:33 GMT
Connection: close
Content-Length: 9692
<!-- BEGIN HEADER -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" >
<head>
<title>
Altoro Mutual
</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<link href="/Gradle_____altermutual_war/style.css" rel="stylesheet" type="text/css" />
</head>

```

- All right! let's check the HTML file we created



- Resolved Clickjacking issue 😊

## ID18

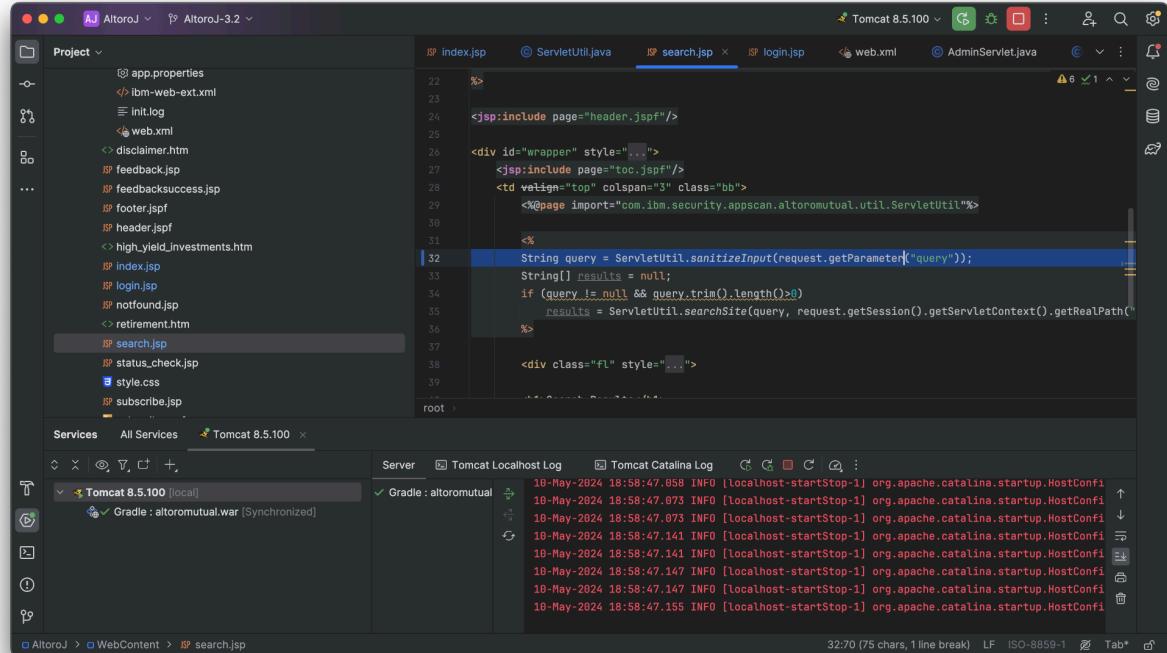
- Finding test steps
  - In the index page go to the search in the top right of the page

The screenshot shows a web browser window for 'Altoro Mutual' at 'localhost:8081/Gradle\_\_altoromutual\_war/search.jsp?query=<script>alert%28\*xss%29<%2Fscript>'. The search bar contains the injected script. The search results page displays a green banner with three small images and the text 'DEMO SITE ONLY'. Below the banner, there are four main navigation categories: PERSONAL, SMALL BUSINESS, and INSIDE ALTORO MUTUAL, each with a list of links. The central content area is titled 'Search Results' and contains the message 'No results were found for the query:'. At the bottom of the page, there is a note: 'The AltoroJ website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/aposcan/>. Copyright © 2009-2017 IBM Corporation. All rights reserved. Copyright © 2017-2024 HCL Technologies Ltd. All rights reserved.'

- Try to write HTML code in search input: <h1>HTML INJECTION</h1>

The screenshot shows a web browser window for 'Altoro Mutual' at 'localhost:8081/Gradle\_\_altoromutual\_war/search.jsp?query=<h1>HTML+INJECTION<h1%2F>'. The search bar contains the injected HTML. The search results page displays a green banner with three small images and the text 'DEMO SITE ONLY'. Below the banner, there are four main navigation categories: PERSONAL, SMALL BUSINESS, and INSIDE ALTORO MUTUAL, each with a list of links. The central content area is titled 'Search Results' and contains the message 'No results were found for the query:'. Below the search results, the word 'HTML INJECTION' is displayed in large, bold, teal letters. At the bottom of the page, there is a note: 'The AltoroJ website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/aposcan/>. Copyright © 2009-2017 IBM Corporation. All rights reserved. Copyright © 2017-2024 HCL Technologies Ltd. All rights reserved.'

- Injected HTML in the web page
- Fixing steps
  - Go to search.jsp to query parameter
  - Found that we sanitized the input before in ID12 



```

%>
<jsp:include page="header.jspf"/>
<div id="wrapper" style="...>
<jsp:include page="toc.jspf"/>
<td valign="top" colspan="3" class="bb">
<%@page import="com.ibm.security.appscan.altoromutual.util.ServletUtil"%>
<%
String query = ServletUtil.sanitizeInput(request.getParameter("query"));
String[] results = null;
if (query != null && query.trim().length()>0)
    results = ServletUtil.searchSite(query, request.getSession().getServletContext().getRealPath("root"));
%>
<div class="fl" style="...>

```

- Finding RE-TEST steps

- Write the same HTML code in the input search

The screenshot shows a web application interface for 'Altoro Mutual'. The URL in the address bar is `localhost:8081/Gradle__altermutual_war/search.jsp?query=<h1>HTML+INJECTION<%2Fh1>`. The page title is 'Altoro Mutual'. The top navigation bar includes links for 'Sign In', 'Contact Us', 'Feedback', a search bar, and a 'Go' button. A banner on the right side says 'DEMO SITE ONLY'. The main content area is titled 'Search Results' and shows the query `<h1>HTML INJECTION</h1>`. Below this, it says 'No results were found for the query:'. On the left sidebar, there are three sections: 'PERSONAL' (Deposit Product, Checking, Loan Products, Cards, Investments & Insurance, Other Services), 'SMALL BUSINESS' (Deposit Products, Lending Services, Cards, Insurance, Retirement, Other Services), and 'INSIDE ALTORO MUTUAL' (About Us, Contact Us, Locations, Investor Relations, Press Room, Careers, Subscribe). At the bottom of the page, there are links for 'Privacy Policy', 'Security Statement', 'Server Status Check', 'REST API', and copyright information: 'Copyright © 2008-2017 IBM Corporation. All rights reserved. Copyright © 2017-2024 HCL Technologies Ltd. All rights reserved.' A note at the bottom right says 'This web application is open source! Get your copy from GitHub and take advantage of advanced features'.

- sanitized input, so resolved issue 😊

## ID19

- Finding test steps
  - Open **login page**

- Using Burp Suite to intercept the request

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. In the 'Request' pane, a GET request is shown:

```

1 GET /Grade_alteromutual_war/login.jsp? HTTP/1.1
2 Host: localhost:8081
3 User-Agent: curl/8.2.1
4 Accept: */*
5 Connection: close
6
7

```

In the 'Response' pane, the server's response is displayed as a JSP file. A specific comment block is highlighted in the 'Selected text' pane:

```

<!-- To get the latest admin login, please contact SiteOps at
415-555-6159 -->

```

The 'Inspector' pane on the right shows the selected text and various request/response details.

- Sensitive data exposure in the code was left in a comment by the developer that can be used by the attacker
- Fixing steps
  - Inspect login.jsp file in the project source code

- Found the sensitive data in line 33

```

n.jsp  <> disclaimer.htm  <> inside_jobs.htm  gradle-wrapper.properties  index.jsp  login.jsp  ...
23
24 <jsp:include page="header.jspf"/>
25
26 <div id="wrapper" style="...">
27   <jsp:include page="/toc.jspf"/>
28   <td valign="top" colspan="3" class="bb">
29     <div class="fl" style="...">
30       <h1>Online Banking Login</h1>
31
32       <!-- To get the latest admin login, please contact SiteOps at 415-555-6159 -->
33       <p><span id="_ctl0__ctl0_Content_Main_message" style="...">
34         java.lang.String error = (String)request.getSession(true).getAttribute("loginError");
35
36         if (error != null && error.trim().length() > 0){
37           request.getSession().removeAttribute("loginError");
38           out.print(error);
39         }
40
41   </div>
42 </td>
43 </div>

```

- Remove the comment

```

n.jsp  <> disclaimer.htm  <> inside_jobs.htm  gradle-wrapper.properties  index.jsp  login.jsp  ...
23
24 <jsp:include page="header.jspf"/>
25
26 <div id="wrapper" style="...">
27   <jsp:include page="/toc.jspf"/>
28   <td valign="top" colspan="3" class="bb">
29     <div class="fl" style="...">
30       <h1>Online Banking Login</h1>
31
32       <p><span id="_ctl0__ctl0_Content_Main_message" style="...">
33         java.lang.String error = (String)request.getSession(true).getAttribute("loginError");
34
35         if (error != null && error.trim().length() > 0){
36           request.getSession().removeAttribute("loginError");
37           out.print(error);
38         }
39
40   </div>
41 </td>
42 </div>

```

- Finding RE-TEST steps
  - Open login page

- Use Burp Suite to intercept the request

The screenshot shows the Burp Suite interface with the following details:

- Request:**

```

1 GET /oracle_alteromutual_war/login.jsp HTTP/1.1
2 Host: localhost:8081
3 User-Agent: curl/8.2.1
4 Accept: */*
5 Connection: close
6
7

```
- Response:**

```

Subscribe
</a>
</li>
</ul>
</td>
<!-- TOC END -->
<td valign="top" colspan="3" class="bb">
<div class="fl" style="width: 99%;>
<h1>
    Online Banking Login
</h1>
<p>
    <span id="ctl0__ctl0_Content_Main_message" style="color: #FF0066; font-size: 12pt; font-weight: bold;">
        </span>
    </p>
<form action="doLogin" method="post" name="login" id="login"
    onsubmit="return (confirmInput(login));">
    <table>
        <tr>
            <td>
                Username:
            </td>
            <td>
                <input type="text" id="uid" name="uid" value="" style="width: 150px;">
            </td>
        </tr>
        <tr>
            <td>
                Password:
            </td>
            <td>
                <input type="password" id="pwd" name="pwd" value="" style="width: 150px;">
            </td>
        </tr>
    </table>
    <input type="submit" value="Log In" style="width: 150px; height: 30px; margin-top: 10px;">
</form>

```
- Inspector Tab:** Shows various request and response details like attributes, query parameters, body parameters, cookies, headers, and response headers.
- Bottom Status:** Event log (2), All issues, 8,986 bytes | 4 millis, Memory: 188.3MB

- Sensitive data is gone 😊