

## Assignment 4

CHITTALA SWATHI

B. TECH

20ME1A4612

RCEE

### Step 1: OWASP Top 10 Vulnerabilities Overview:

1. **Injection:** Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. Attackers can exploit these flaws to execute malicious commands or access unauthorized data.
2. **Broken Authentication:** Broken authentication vulnerabilities occur when attackers can compromise passwords, keys, or session tokens, allowing them to assume the identity of other users or bypass access controls.
3. **Sensitive Data Exposure:** This vulnerability involves the exposure of sensitive data, such as financial information, healthcare records, or personally identifiable information (PII), either due to insecure storage, transmission, or inadequate protection mechanisms.
4. **XML External Entities (XXE):** XXE vulnerabilities occur when an application processes XML input containing a reference to an external entity. Attackers can exploit this to disclose confidential data, execute remote code, or perform server-side request forgery (SSRF) attacks.
5. **Broken Access Control:** Broken access control vulnerabilities allow attackers to bypass restrictions on accessing resources they shouldn't have access to. This could include unauthorized access to data or functionality within the application.
6. **Security Misconfiguration:** Security misconfiguration vulnerabilities arise when security settings are implemented incorrectly, leaving the application vulnerable to various attacks. Examples include default configurations, incomplete or outdated security configurations, and unnecessary services enabled.
7. **Cross-Site Scripting (XSS):** XSS vulnerabilities occur when an application includes untrusted data in a web page without proper validation or escaping, allowing attackers to execute malicious scripts in the context of the victim's browser.
8. **Insecure Deserialization:** Insecure deserialization vulnerabilities occur when untrusted data is deserialized by an application. Attackers can exploit this to execute arbitrary code, perform denial-of-service attacks, or tamper with serialized objects.
9. **Using Components with Known Vulnerabilities:** This vulnerability arises when an application uses third-party components, libraries, or frameworks with known security vulnerabilities. Attackers can exploit these vulnerabilities to compromise the application's security.

10. **Insufficient Logging and Monitoring:** Insufficient logging and monitoring make it difficult for organizations to detect and respond to security incidents effectively. This includes inadequate logging of security-relevant events, lack of monitoring for suspicious activities, and insufficient incident response capabilities.

### **Potential Impact of These vulnerabilities on web applications:**

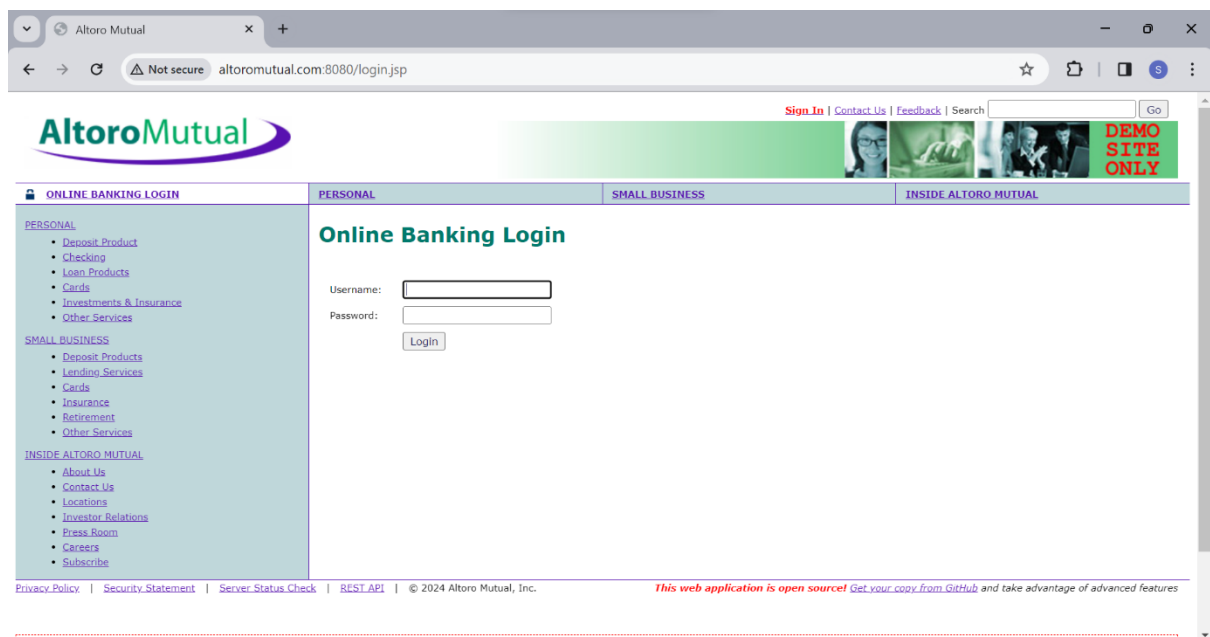
1. **Injection:** Injection vulnerabilities, such as SQL injection, can allow attackers to execute arbitrary SQL commands against the application's database. This could lead to unauthorized access to sensitive data, data manipulation, or even complete database compromise.
2. **Broken Authentication:** If attackers can compromise user authentication mechanisms, they can impersonate legitimate users, gaining unauthorized access to sensitive data or performing actions on behalf of those users. This could result in data breaches, financial fraud, or loss of trust from customers.
3. **Sensitive Data Exposure:** Exposure of sensitive data, such as credit card numbers or personal health information, can lead to identity theft, financial fraud, regulatory penalties, and damage to an organization's reputation and trustworthiness.
4. **XML External Entities (XXE):** XXE vulnerabilities can allow attackers to read arbitrary files, perform SSRF attacks, or execute arbitrary code on the server. This could result in data leakage, server compromise, or further exploitation of internal systems.
5. **Broken Access Control:** Failure to properly enforce access controls can allow attackers to access unauthorized resources or perform actions beyond their privileges. This could lead to unauthorized data access, privilege escalation, or unauthorized modification of critical data.
6. **Security Misconfiguration:** Security misconfigurations can expose sensitive information, provide unnecessary access to resources, or leave security mechanisms ineffective. Attackers can exploit these misconfigurations to gain unauthorized access, escalate privileges, or conduct other malicious activities.
7. **Cross-Site Scripting (XSS):** XSS vulnerabilities can enable attackers to inject malicious scripts into web pages viewed by other users. This could lead to session hijacking, defacement of web pages, theft of sensitive cookies, or phishing attacks against users.
8. **Insecure Deserialization:** Insecure deserialization vulnerabilities can allow attackers to execute arbitrary code, perform denial-of-service attacks, or tamper with serialized

objects. This could lead to server compromise, data manipulation, or disruption of service availability.

9. **Using Components with Known Vulnerabilities:** If an application uses components with known vulnerabilities, attackers can exploit these vulnerabilities to compromise the application's security. This could result in unauthorized access, data breaches, or compromise of the entire system.
10. **Insufficient Logging and Monitoring:** Without proper logging and monitoring, organizations may not be able to detect and respond to security incidents effectively. This could result in delayed detection of attacks, prolonged unauthorized access, and increased damage caused by successful attacks.

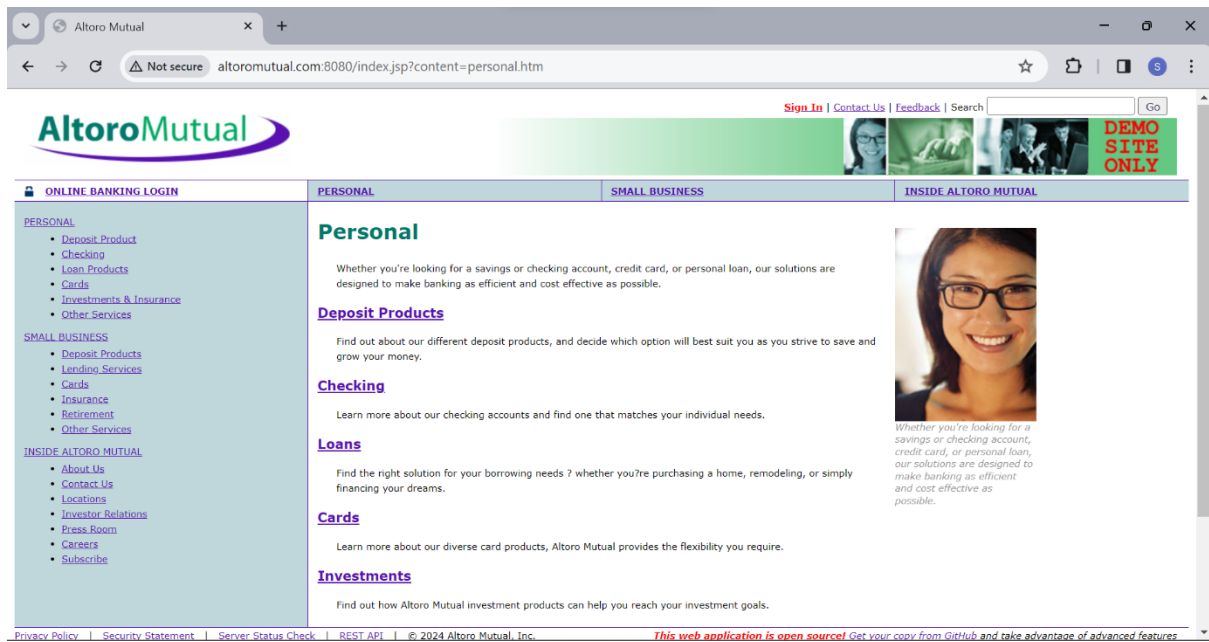
## Step 2: Altro Mutual Website Analysis:

The Altro Mutual website consists of “Login page” which consists of Username and Password and it has “Contact Us” and “Feedback” and also search bar which is used for searching.



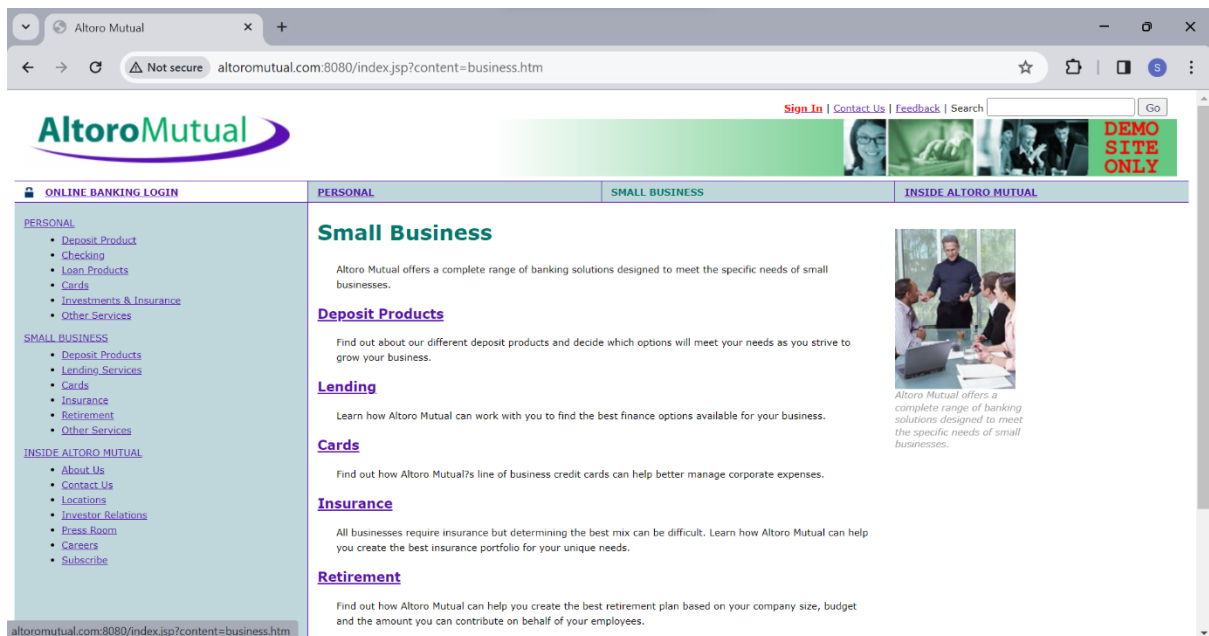
1. This website consists of Personal page where it has

- Deposit Products
- Checking
- Loans
- Cards
- Investments



2. It also consists of Small Business where it has

- Deposit Products
- Lending
- Cards
- Insurance
- Retirement



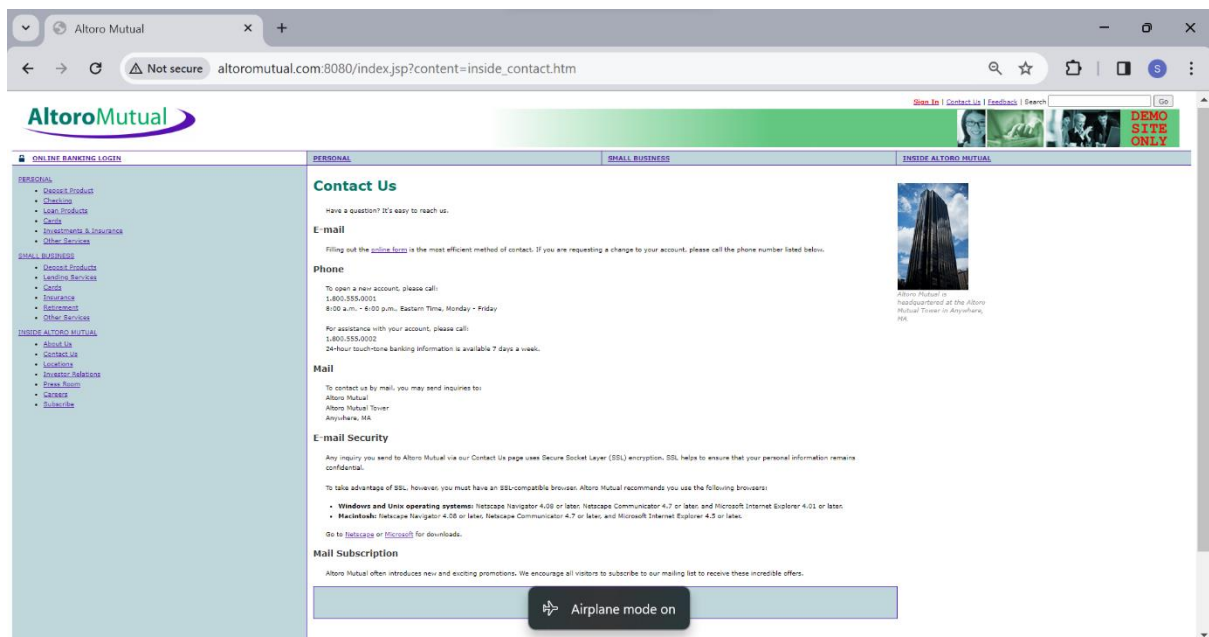
3. It has INSIDE ALTRO MUTUAL where it has

- Part of a corporate Family details

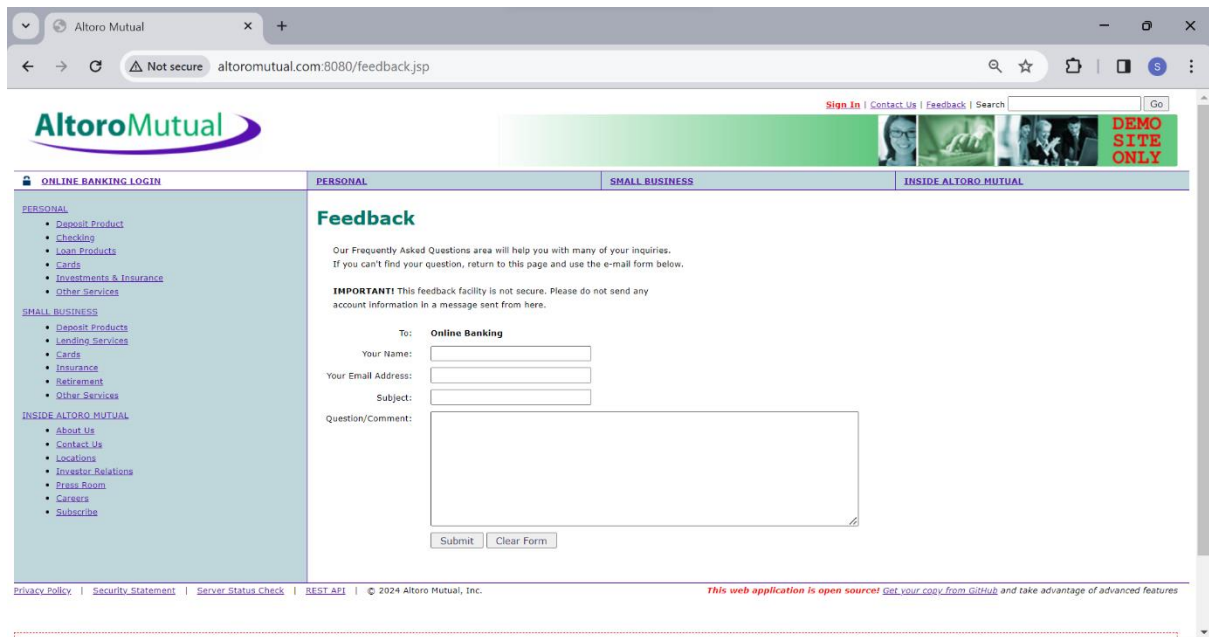


This website consists of Contact us form and feedback form as follow:

## Contact Us Form:



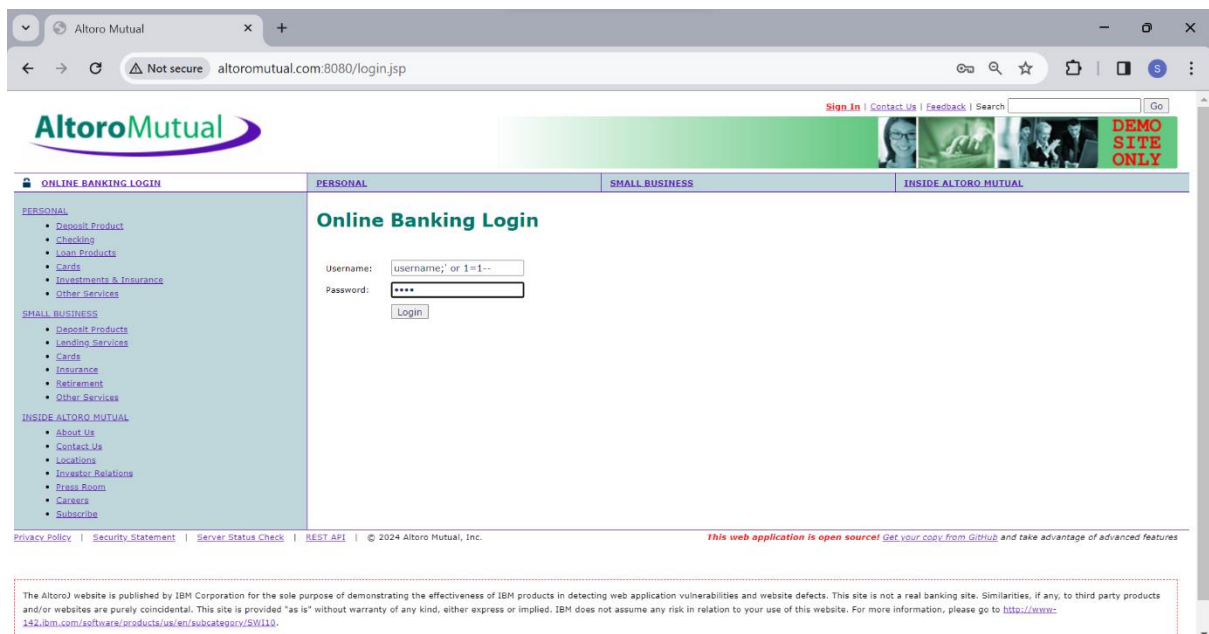
## Feedback Form:



The screenshot shows a web browser window with the URL `altoromutual.com:8080/feedback.jsp`. The page features the Altoro Mutual logo and a navigation bar with links for [Sign In](#), [Contact Us](#), [Feedback](#), and a search bar. A "DEMO SITE ONLY" banner is visible in the top right. The main content area is titled "Feedback" and includes a message: "Our Frequently Asked Questions area will help you with many of your inquiries. If you can't find your question, return to this page and use the e-mail form below." Below this, a warning states: "IMPORTANT! This feedback facility is not secure. Please do not send any account information in a message sent from here." The form fields include "To: Online Banking", "Your Name:", "Your Email Address:", "Subject:", and a large "Question/Comment:" text area. "Submit" and "Clear Form" buttons are at the bottom. A footer contains links for [Privacy Policy](#), [Security Statement](#), [Server Status Check](#), [REST API](#), and copyright information for 2024 Altoro Mutual, Inc. A note at the bottom right mentions: "This web application is open source! Get your copy from GitHub and take advantage of advanced features."

## Step 3: Vulnerability Identification Report:

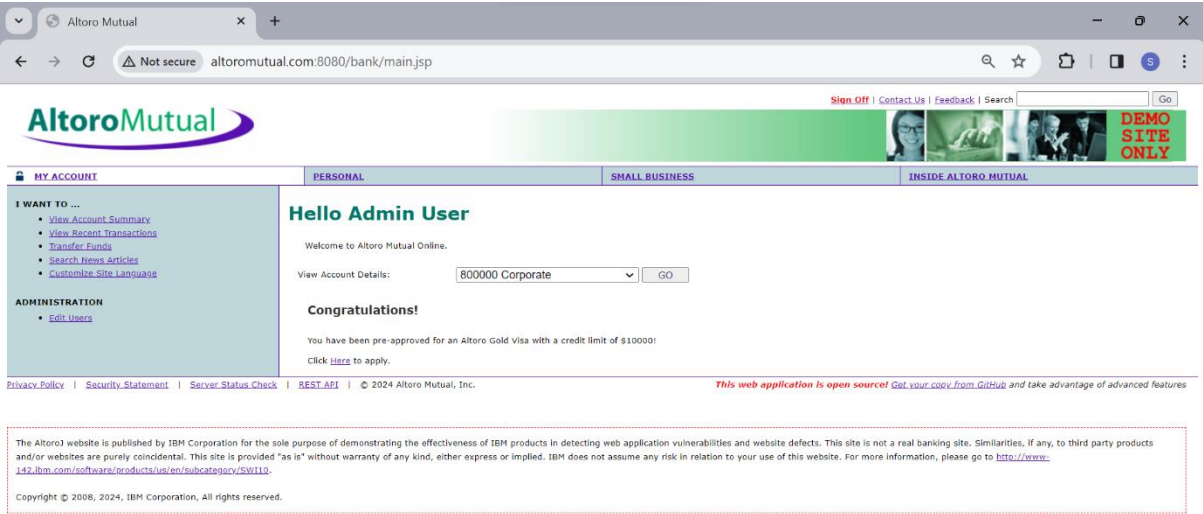
We will assess whether login functionality on a website is successful. If login is possible, it indicates a potential vulnerability known as "Broken Authorization," which can further expose the system to risks such as Sensitive Data Exposure and Broken Access Control.



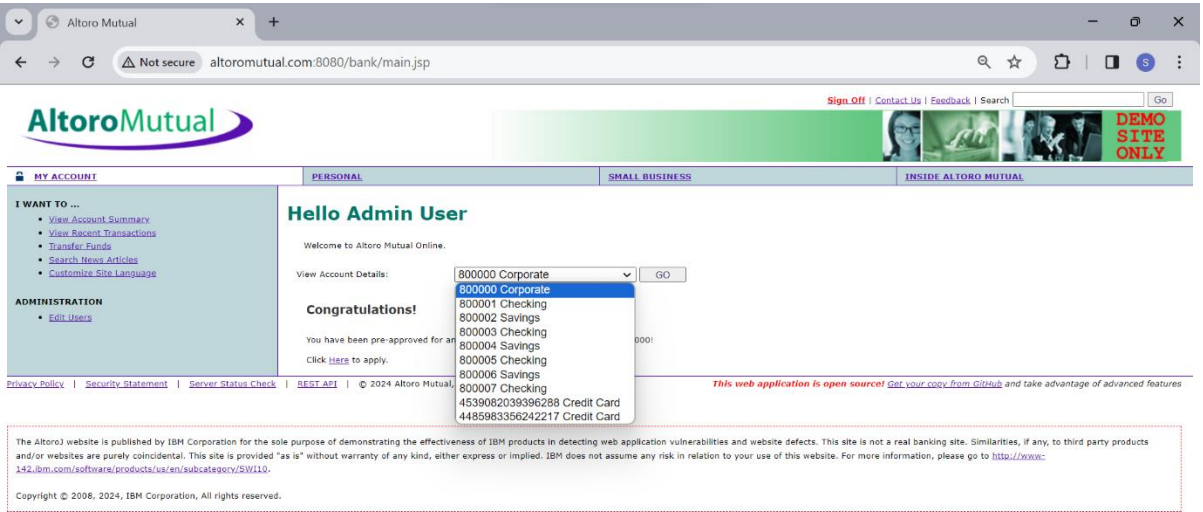
The screenshot shows a web browser window with the URL `altoromutual.com:8080/login.jsp`. The page features the Altoro Mutual logo and a navigation bar with links for [Sign In](#), [Contact Us](#), [Feedback](#), and a search bar. A "DEMO SITE ONLY" banner is visible in the top right. The main content area is titled "Online Banking Login" and includes fields for "Username:" (containing the text `username;' or 1=1--`) and "Password:" (containing the text `1234`). A "Login" button is located below the password field. The left sidebar contains a menu with categories: 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 footer contains links for [Privacy Policy](#), [Security Statement](#), [Server Status Check](#), [REST API](#), and copyright information for 2024 Altoro Mutual, Inc. A note at the bottom right mentions: "This web application is open source! Get your copy from GitHub and take advantage of advanced features." A disclaimer at the bottom states: "The AltoroJ website is published by IBM Corporation for the sole purpose of demonstrating the effectiveness of IBM 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. IBM does not assume any risk in relation to your use of this website. For more information, please go to <http://www-192.ibm.com/software/products/us/en/subcategory/SW110>."

Here we enter username as **username;' or 1=1**—and password as **1234** and we can access another person accounts and we can modify it.

This page this after login successfully:



Here we can check the persons details:





We can also edit the users information such as users name and account type details and etc.. :

**AltoroMutual**

Sign Off | Contact Us | Feedback | Search

**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

### 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:  Last 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

We can also access recent Transactions:

**AltoroMutual**

Sign Off | Contact Us | Feedback | Search

**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

### Recent Transactions

After  Before

yyyy-mm-dd yyyy-mm-dd

Transaction ID	Transaction Time	Account ID	Action	Amount
2363	2024-03-22 05:26	800001	Deposit	\$543.00
2362	2024-03-22 05:26	800000	Withdrawal	-\$543.00
2361	2024-03-22 05:23	800001	Deposit	\$134.00
2360	2024-03-22 05:23	800000	Withdrawal	-\$134.00
2359	2024-03-22 04:57	800001	Deposit	\$1235.00
2358	2024-03-22 04:57	800000	Withdrawal	-\$1235.00
2357	2024-03-22 04:52	800001	Deposit	\$678.00
2356	2024-03-22 04:52	800000	Withdrawal	-\$678.00
2355	2020-11-24 05:15	800002	Deposit	\$1.00
2354	2020-11-24 05:15	800003	Withdrawal	-\$1.00
2353	2020-11-24 05:15	4539082039396288	Cash Advance Fee	\$2.50
2350	2020-11-24 05:15	800002	Deposit	\$1.00
2349	2020-11-24 05:15	4539082039396288	Cash Advance	\$1.00
2352	2020-11-24 05:15	800002	Deposit	\$1.00
2351	2020-11-24 05:15	800002	Withdrawal	-\$1.00
2348	2020-11-24 05:15	800002	Deposit	\$1.00
2347	2020-11-24 05:15	800002	Withdrawal	-\$1.00



We can transfer funds also:

Altoro Mutual

Sign Off | Contact Us | Feedback | Search

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

**Transfer Funds**

From Account: 800000 Corporate

To Account: 800000 Corporate

Amount to Transfer: 800003124

Transfer Money

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 Altoro website is published by IBM Corporation for the sole purpose of demonstrating the effectiveness of IBM 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. IBM does not assume any risk in relation to your use of this website. For more information, please go to <http://www-142.ibm.com/software/products/us/en/subcategory/SW110>.

Copyright © 2008, 2024, IBM Corporation, All rights reserved.

## Mitigations:

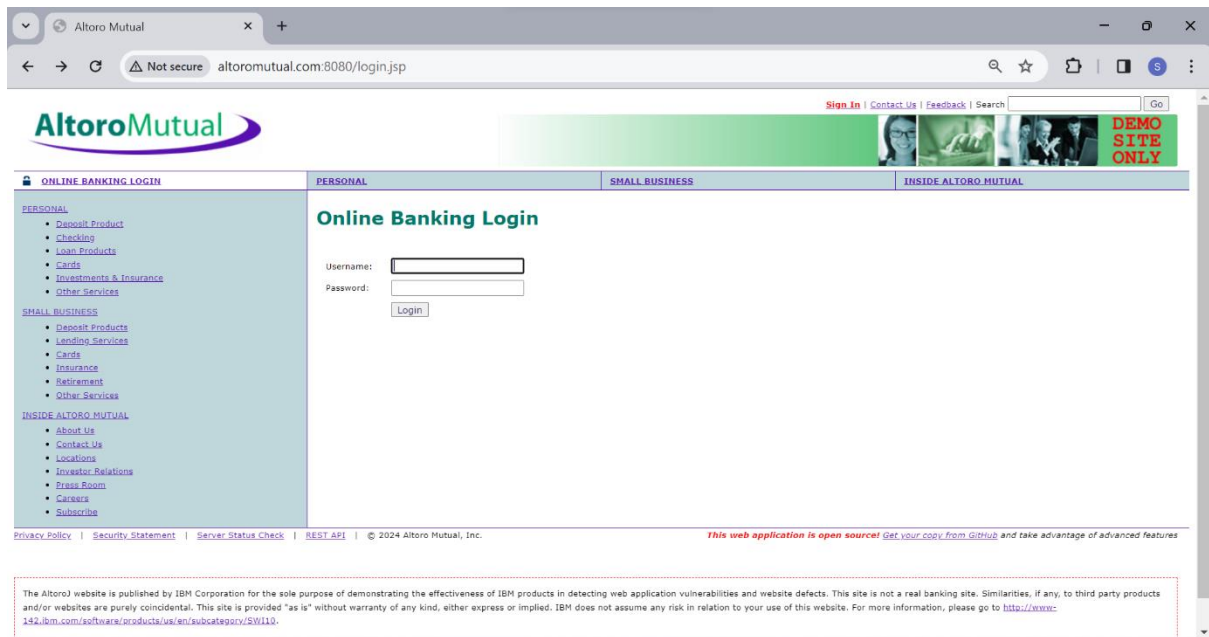
Utilize Strong Authentication Methods:

- Employ robust authentication mechanisms like multi-factor authentication (MFA) to ensure only authorized users can access the system. Store passwords securely using techniques such as hashing with salts to prevent unauthorized access to user credentials.
- Enforce Principle of Least Privilege:
  - Assign minimal access levels required for each user or role in the system. Regularly review and update permissions to align with the principle of least privilege, ensuring users have access only to necessary resources and functionalities.
- Implement Role-Based Access Control (RBAC):
  - Define and enforce user roles and permissions using RBAC, allowing granular control over user actions within the system. Ensure roles are well-defined and regularly reviewed to maintain security.
- Implement Attribute-Based Access Control (ABAC):
  - Base access control decisions on various attributes such as user characteristics or resource properties using ABAC. Implement dynamic access control policies to adapt to changing conditions.
- Employ Access Control Lists (ACLs) and Whitelisting:

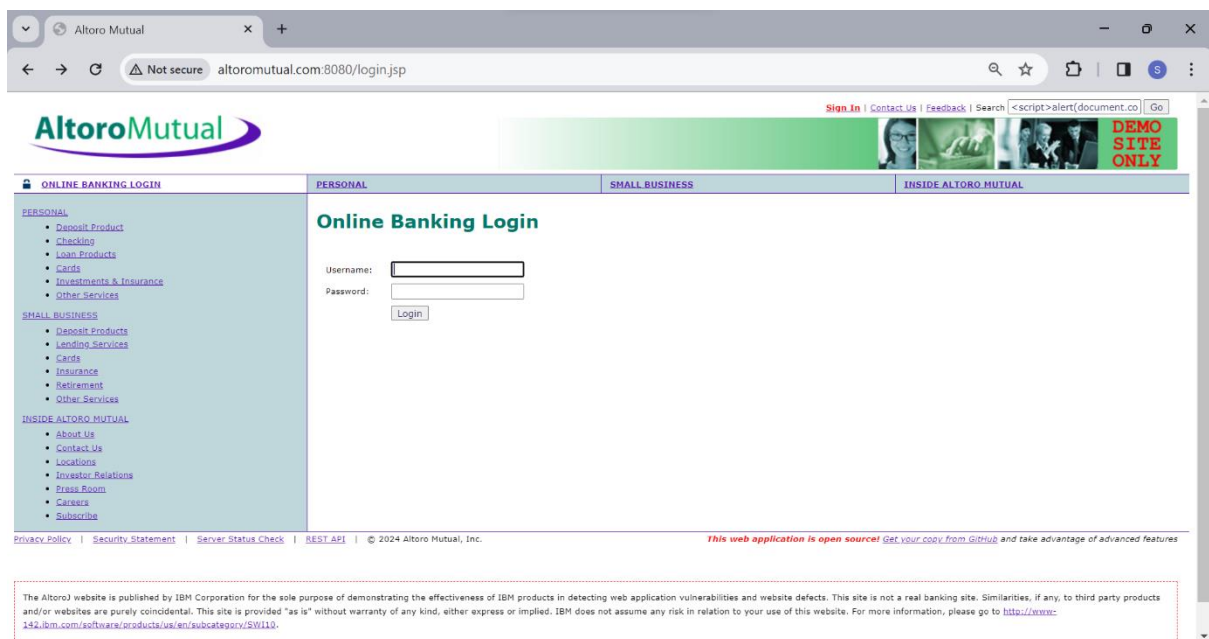
- Utilize ACLs to manage resource access for specific users or groups. Implement whitelisting to specify allowed actions or entities, restricting access to approved entities or actions only.
- Secure Session Management:
  - Generate, transmit, and validate session tokens securely. Use HTTPS and HTTP-only cookies to protect session data from unauthorized access or tampering.
- Conduct Regular Security Testing and Code Reviews:
  - Perform frequent security assessments, including penetration testing and code reviews, to identify and mitigate access control vulnerabilities. Utilize automated tools and manual techniques to uncover security flaws.
- Implement Proper Error Handling:
  - Avoid exposing sensitive information in error messages that could aid attackers. Provide generic error messages to users and log detailed error information for system administrators.
- Secure APIs and Web Services:
  - Implement robust authentication and authorization mechanisms for APIs and web services. Utilize technologies such as OAuth or JWT tokens to authenticate and authorize requests securely.
- Monitor and Audit Access Control Events:
  - Implement logging and monitoring systems to track access control events and detect suspicious activities. Regularly review access logs and audit trails to identify and respond to potential security incidents.

## **Cross site scripting**

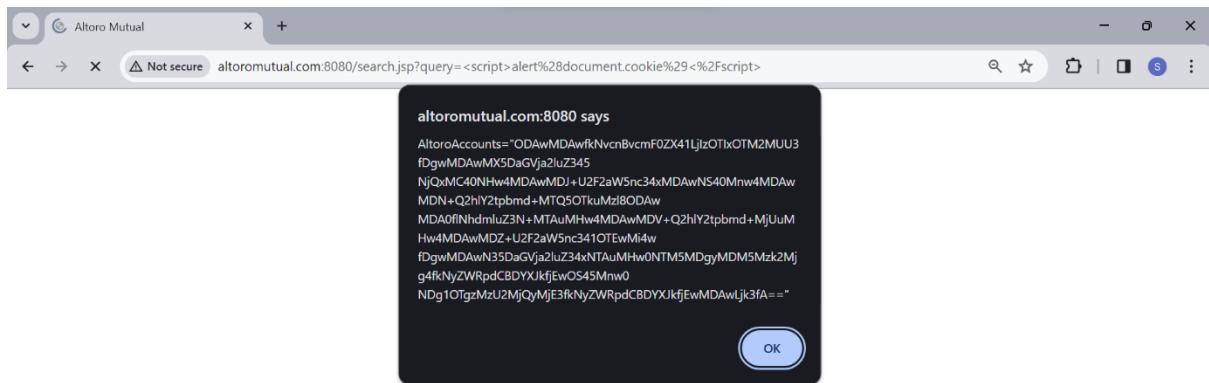
Now we are going to perform cross site scripting on the Altoro Mutual website.



In a search bar write a script: `<script>alert(document.cookie)</script>`



It shows the alert



---

## Mitigations:

- Input Validation and Sanitization:
  - Validate and sanitize all user input server-side to ensure conformity to expected formats and to eliminate malicious code.
- Content Security Policy (CSP):
  - Deploy a Content Security Policy (CSP) to dictate browser behavior regarding script execution and resource loading, reducing the impact of injected scripts.
- Output Encoding:
  - Encode user-generated content before rendering in HTML context to prevent browsers from interpreting it as executable code.
- HTTP Only and Secure Cookies:
  - Enable the **HttpOnly** flag on cookies to prevent client-side script access, thus safeguarding session tokens against XSS attacks.
- Frameworks with Built-in XSS Protection:
  - Maintain updated framework dependencies to leverage the latest security enhancements and patches.
- Contextual Output Escaping:
  - Apply output encoding tailored to the data's rendering context (e.g., HTML, JavaScript, CSS) for precise protection against XSS attacks.
- Regular Security Training and Awareness:

- Educate developers on secure coding practices and XSS vulnerabilities' risks. Advocate for safe HTML templating methods and discourage concatenation of untrusted data with HTML markup.
- Static and Dynamic Analysis Tools:
  - Employ static code analysis tools and web vulnerability scanners to detect XSS vulnerabilities. Conduct periodic security assessments and penetration tests to identify and address XSS risks promptly.
- Secure Development Lifecycle (SDLC):
  - Enforce security-centric coding guidelines and best practices throughout the development process to minimize XSS vulnerabilities.