**NAME :** SHAIK JASMIN

**ASSIGNMENT : 5**

**COLLEGE :** ANNAMACHARYA INSTITUTE OF TECHNOLOGY & SCIENCES, TIRUPATI

**BRANCH :** CSE(CIC)

**E-MAIL:** [jasminshaik523@gmail.com](mailto:jasminshaik523@gmail.com)

Overview of major vulnerabilitiesin OWASP 10

**• Top 10 OWASP Vulnerabilities and Their Potential Consequences:**

* **Broken Access Control (A01:2021): The most serious vulnerability, *Broken Access Control,* is related to poorly managed access controls. It is important to ensure that users only have access to authorized resources and functions.**
* **Impact to Business: Inadequate control of access controls can lead to unauthorized entry, security breaches and compromise of user accounts, which can lead to financial, reputational and legal issues. too bad consequences.**
* **Solution: Implement strong permissions, role-based permissions and thorough testing to prevent unauthorized access.**
* **Cryptographic Failures (A02:2021):** Formerly known as *"Sensitive Data Exposure,"* this classification highlights cryptographic issues. Encryption errors can lead to disclosure of sensitive information or system compromise.
* **Business impact:** Encryption errors can expose confidential information such as customer information or payment information. Such violations can undermine trust, damage brand credibility and lead to legal consequences.
* **Solution:** Use strong encryption algorithms, secure key management and regular security audits.
* **Injection (A03:2021):** Injection vulnerabilities remain a major concern. 94% of applications were tested for some type of injection. This classification also includes cross-site scripting (XSS).
* **Business impact:** Additional vulnerabilities (such as SQL injection or command injection) allow attackers to corrupt data, execute unauthorized code, or gain unauthorized access. These actions can disrupt services, compromise data integrity, and harm business operations.
* **Solution:** validate and sanitize user inputs, use parameterized queries, and avoid executing dynamic code.
* **Insecure Design (A04:2021):** This category is a recent addition to the list and highlights design weaknesses. Threat modeling, secure design patterns, and reference architectures are key to mitigating this risk.
* **Business impact:** Poorly designed systems can contain vulnerabilities that are difficult to patch in the future. These vulnerabilities can lead to costly remediation, security breaches, and delays in project completion.
* **Solution:** Perform threat modeling, follow secure design principles, and involve security experts from the very beginning of the development phase.
* **Security Misconfiguration (A05:2021):** Bugs are on the rise as customizable software becomes more common. Ninety percent of applications were tested for various types of bugs.
* **Business impact:** Bugs expose sensitive information, weaken security measures, and create opportunities for attackers. These can lead to data breaches, service interruptions, and breaches.
* **Solution:** Evaluate settings regularly, adhere to the highest security standards, and automate configuration evaluation.
* **Vulnerable and Outdated Components (A06:2021):** This category, formerly known as *"Parts with Known Vulnerabilities"*, is updated from the previous ranking.
* **Business Impact:** Using outdated or sensitive components can create vulnerabilities. Violations caused by uninstalled libraries can result in financial setbacks, legal liability, and reputational damage.
* **Solution:** Monitor dependencies, apply patches quickly, and use tools to identify vulnerable components.
* **Identification and Authentication Failures (A07:2021):** The category previously labeled *"Broken Authentication"* now includes CWEs related to authentication errors. Adopting standard frameworks helps reduce this risk.
* **Business Impact:** Inadequate authentication methods can allow unauthorized access, which can lead to data breaches, compromised accounts, and financial fraud.
* **Solution:** Implement multi-factor authentication, ensure strong password guidelines. and conduct regular security assessments.

Altoro Mutual Website Analysis

* **Analysis of the Altoro Mutual website:**
* A review of the Altoro Mutual website indicates that “Oxytis Forensics” has conducted a comprehensive security assessment. This assessment included penetration testing guided by the OWASP Top 10, a recognized document on developer awareness and web application security.
* The findings highlighted several vulnerabilities that could potentially lead to unauthorized system use or data breaches.
* In addition, Altoro Mutual emphasizes its commitment to protecting customer privacy and security through financial services and customer data protection measures. Also referenced is a GitHub repository where Altoro Mutual, Inc. conducted a detailed *“Vulnerability Assessment and Penetration Test”* (VAPT) using several web application security testing methods.
* Vulnerabilities such as SQL injection, cross-site scripting (XSS), insecure authentication mechanisms, unprotected direct object references and more were identified on the Altro Mutual Website.

Figure 1: Altoro Mutual Website's Home Page

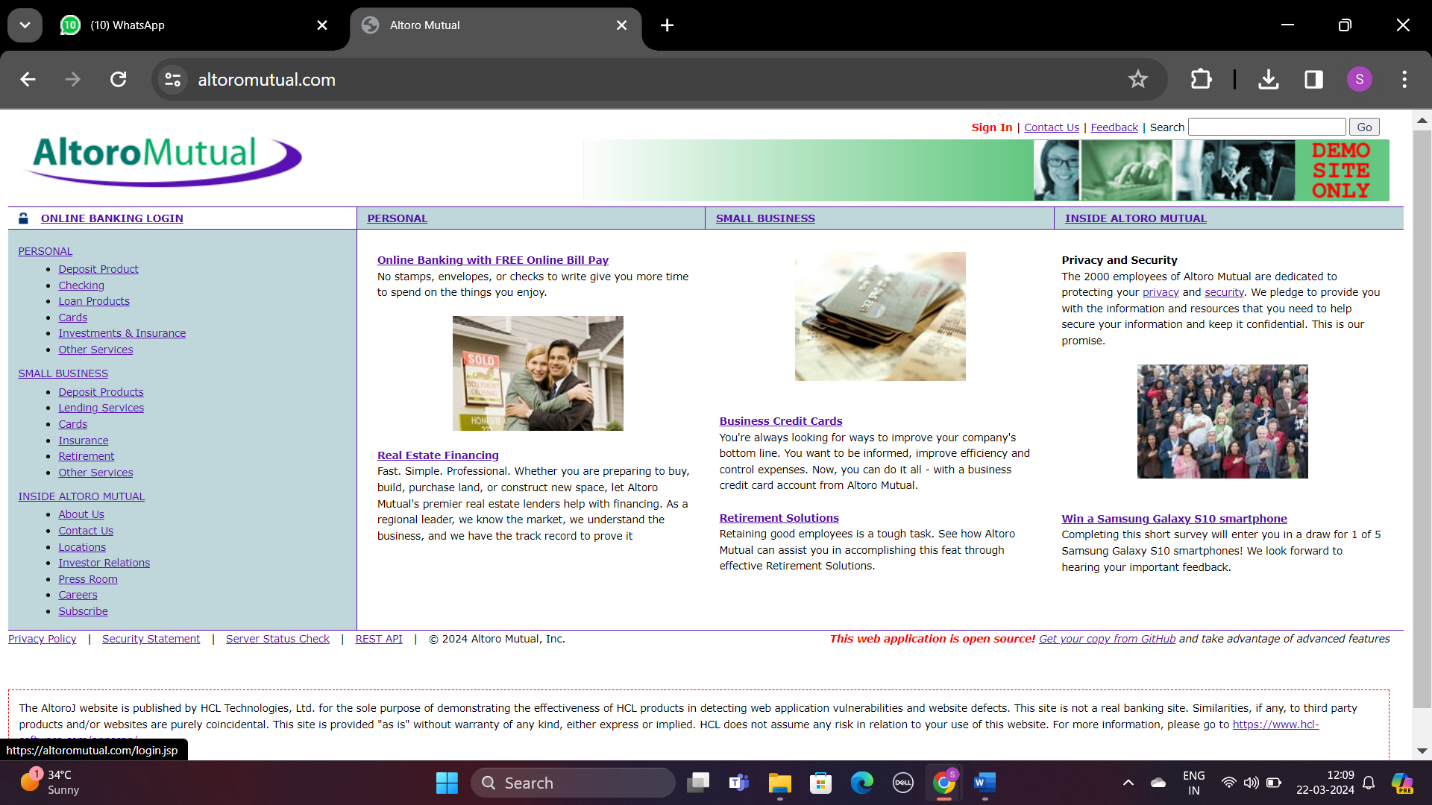


Figure 2: Login Page

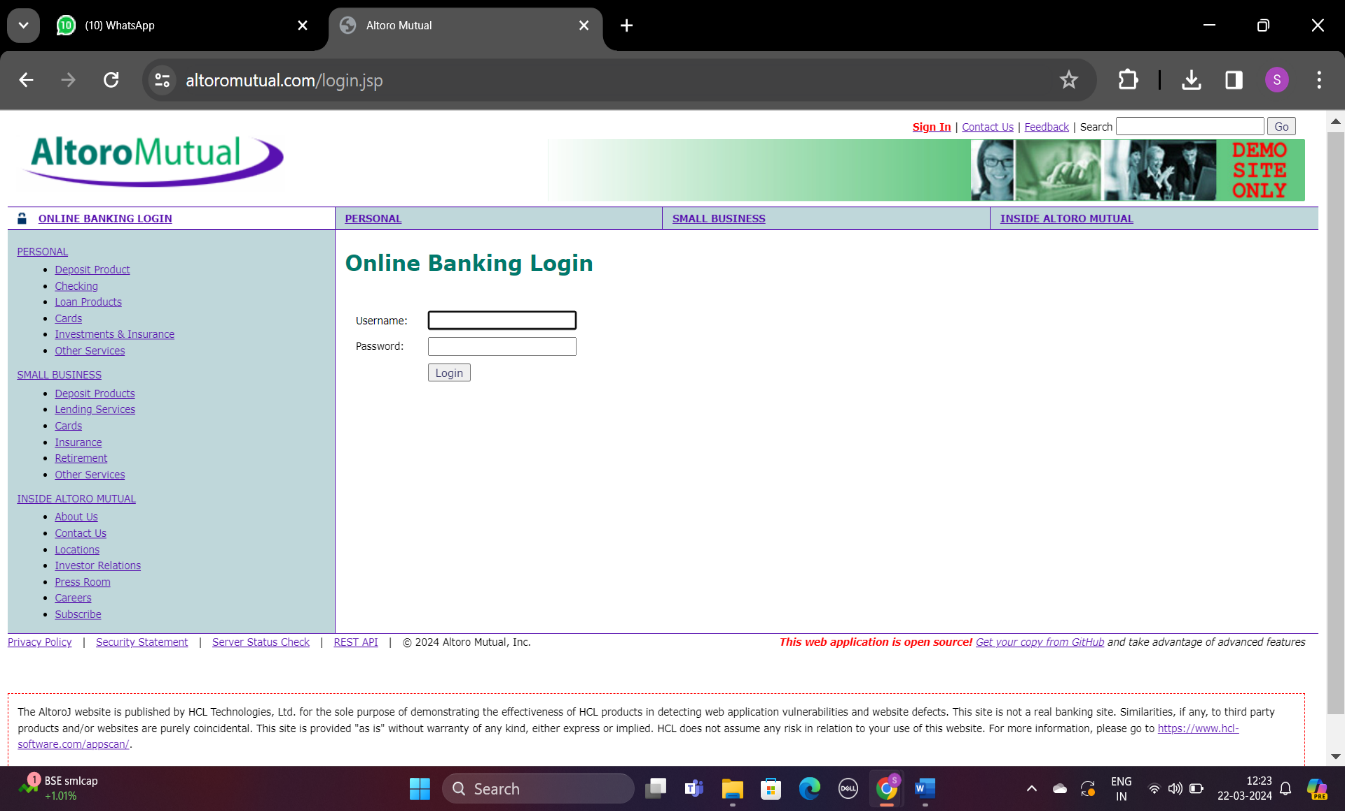


Figure 3: Contact Information

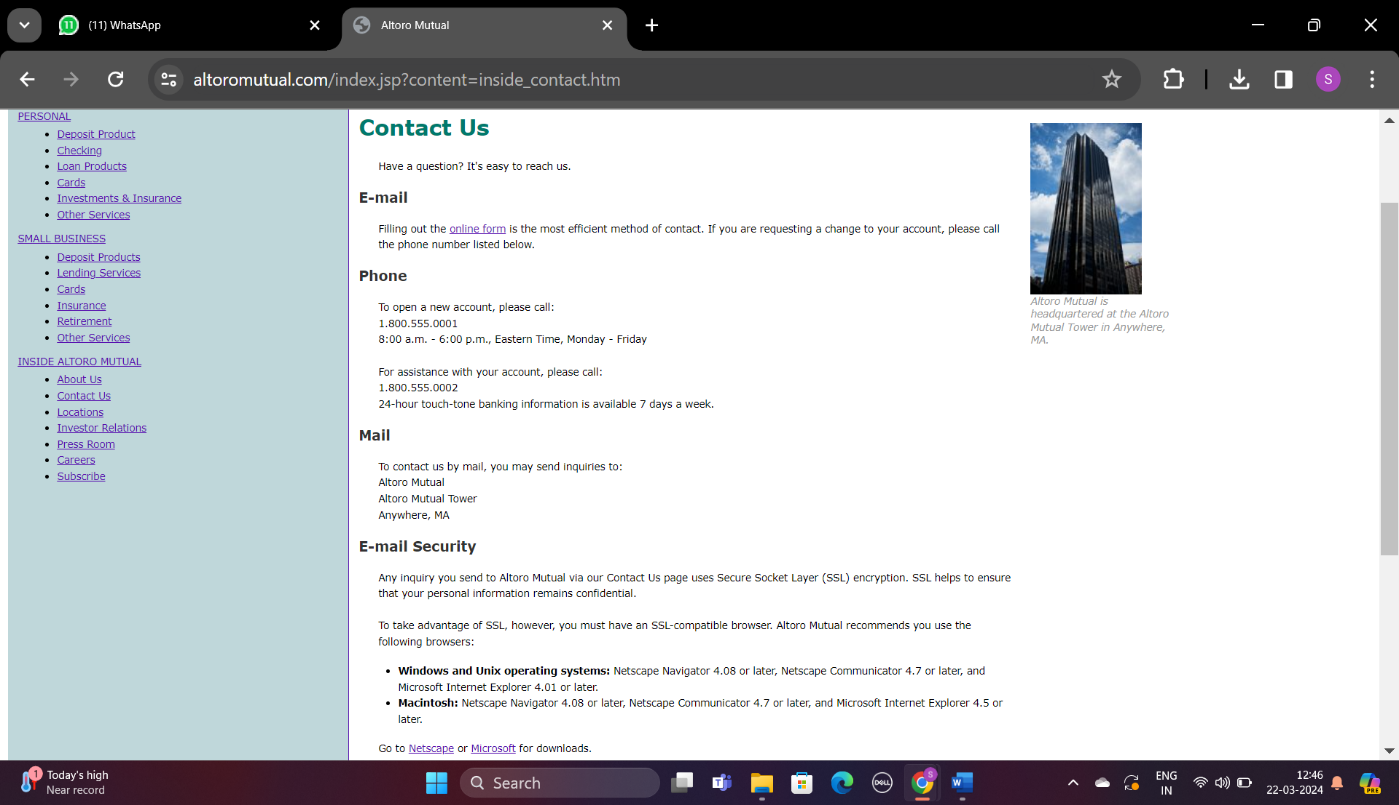
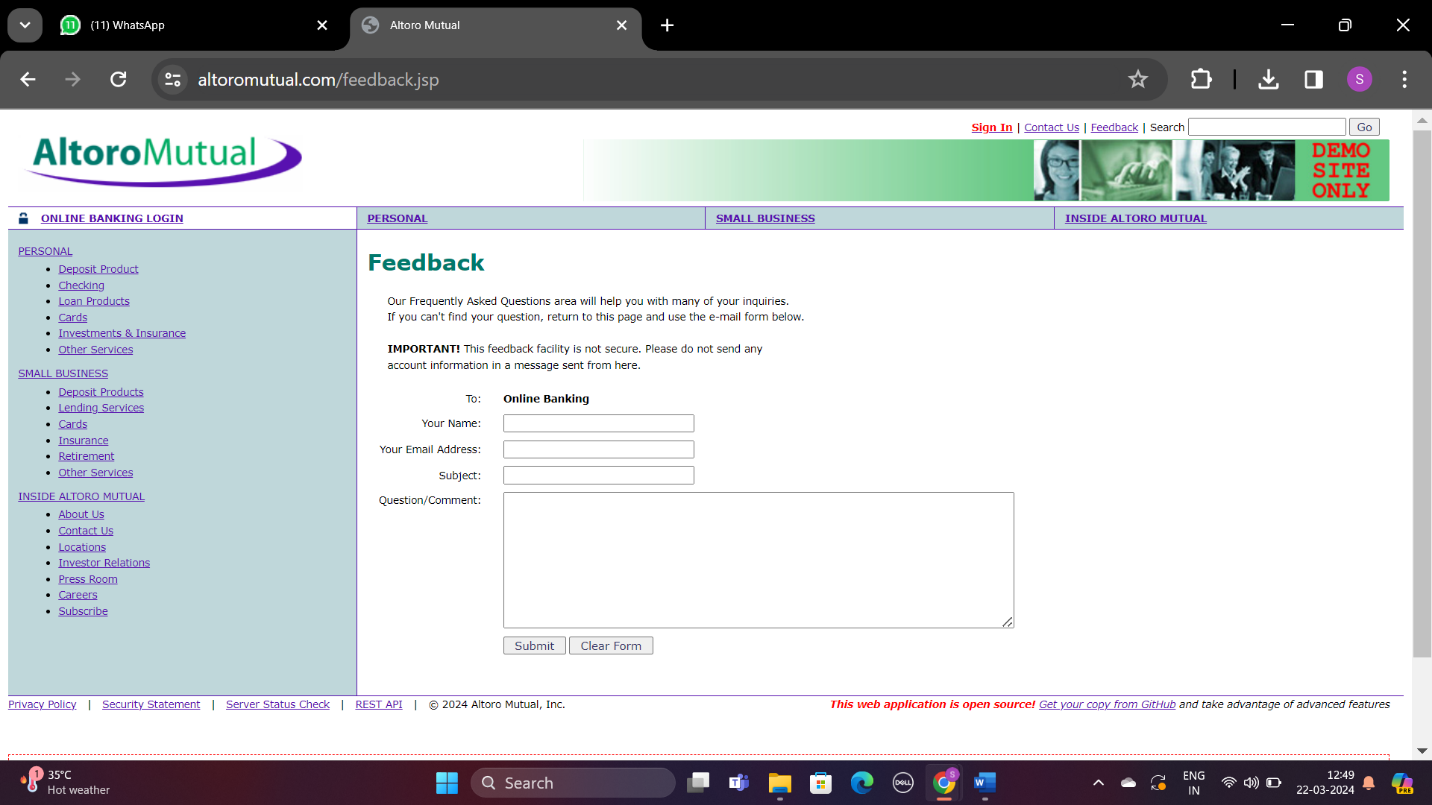


Figure 4: Feedback Page

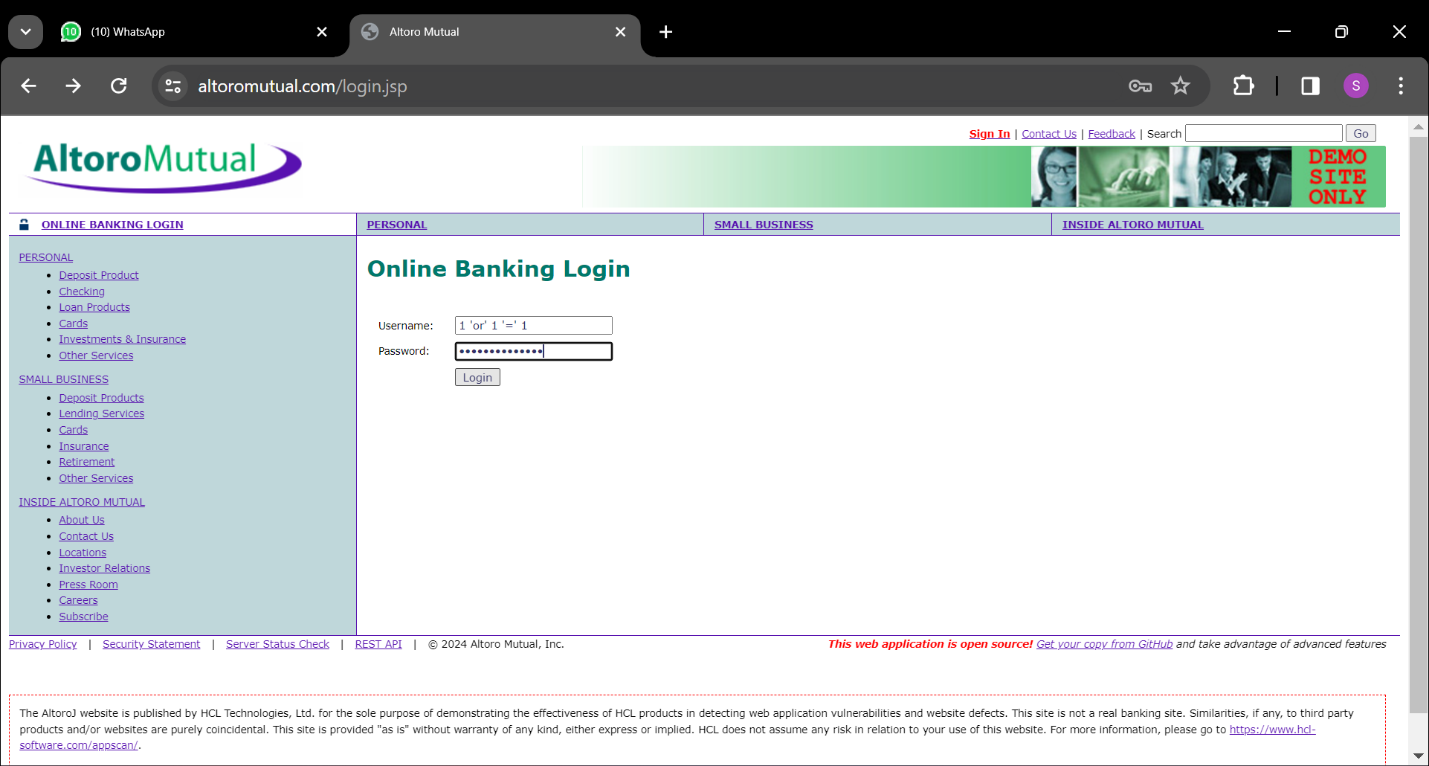


Vulnerability Detection and Exploitation

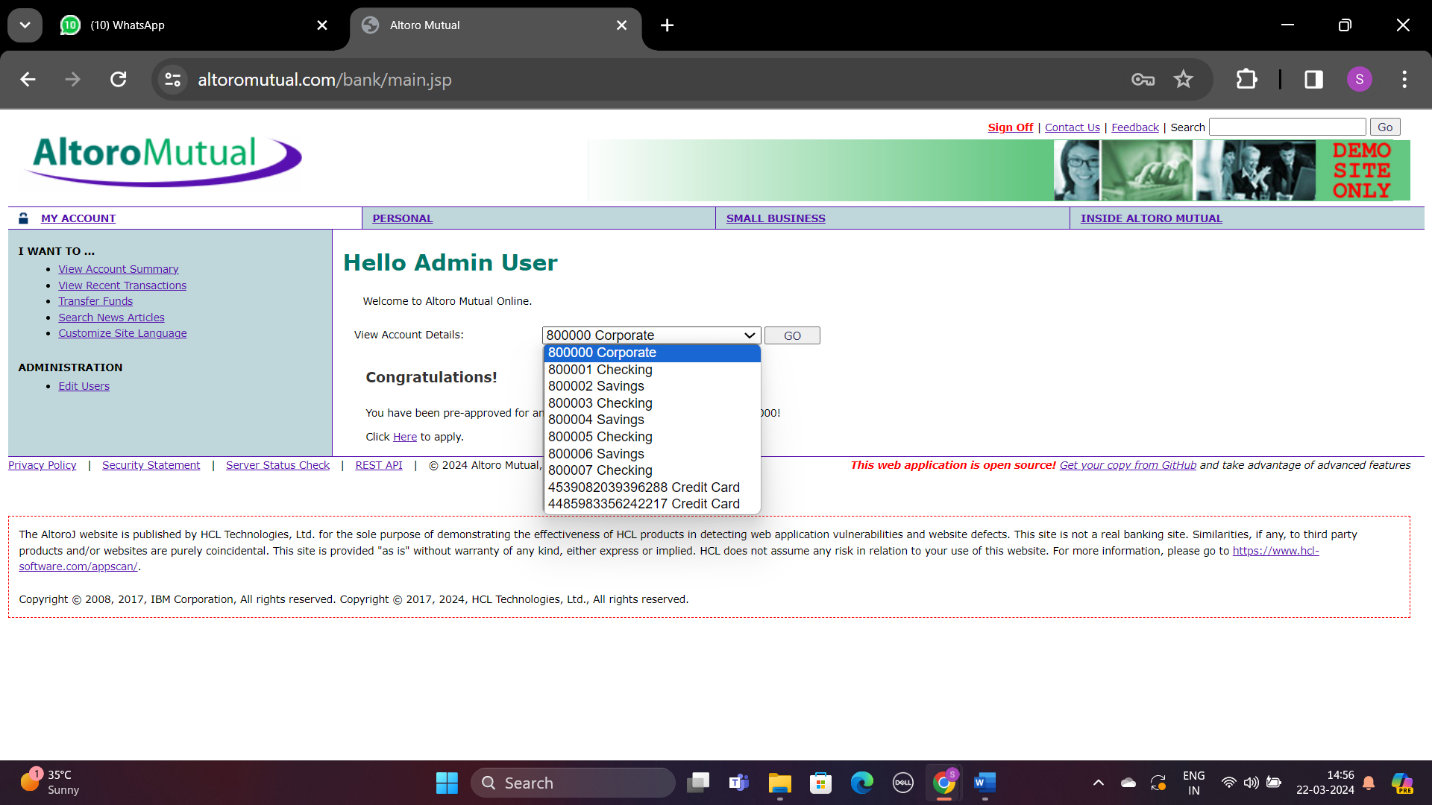
* **Vulnerability Detection and Vulnerability Exploitation Report:**
* Altoro Mutual's website is designed to provide a wide range of banking services that include personal and business banking, investment options, insurance options and various loan offers.
* The platform includes secure online banking features, detailed information about the bank's heritage, ways to contact the institution and various measures to ensure privacy and security. In addition, the site offers practical tools for online invoicing and real estate financial solutions.
* The design of the site aims to improve communication with these offers so that customers can track their finances, apply for loans and receive financial advice. Despite its focus on user engagement, the site remains committed to security and promises to protect user data and privacy with effective safeguards.
* The site's security posture exposes several potential vulnerabilities:
* Cross-Site Scripting (XSS): XSS errors detected may allow malicious script injection into websites affecting other users.
* Insecure file inclusions: Potential file inclusions could allow an attacker to inject files into a server, causing unauthorized access or data leakage.
* Open a directory: Visible folders can expose application plans and sensitive files.
* Misconfigured permissions: Misconfigured permissions can allow unauthorized access to files in certain folders, compromising security and privacy.
* Strong permissions must be implemented to avoid these vulnerabilities, security protocols, including comprehensive input inspection, secure output handling, and precise configuration of files and folders. Ongoing security assessments and adherence to web application development best practices are also important to mitigate these security risks.
* **Vulnerability Check:**

**Broken Authorization**

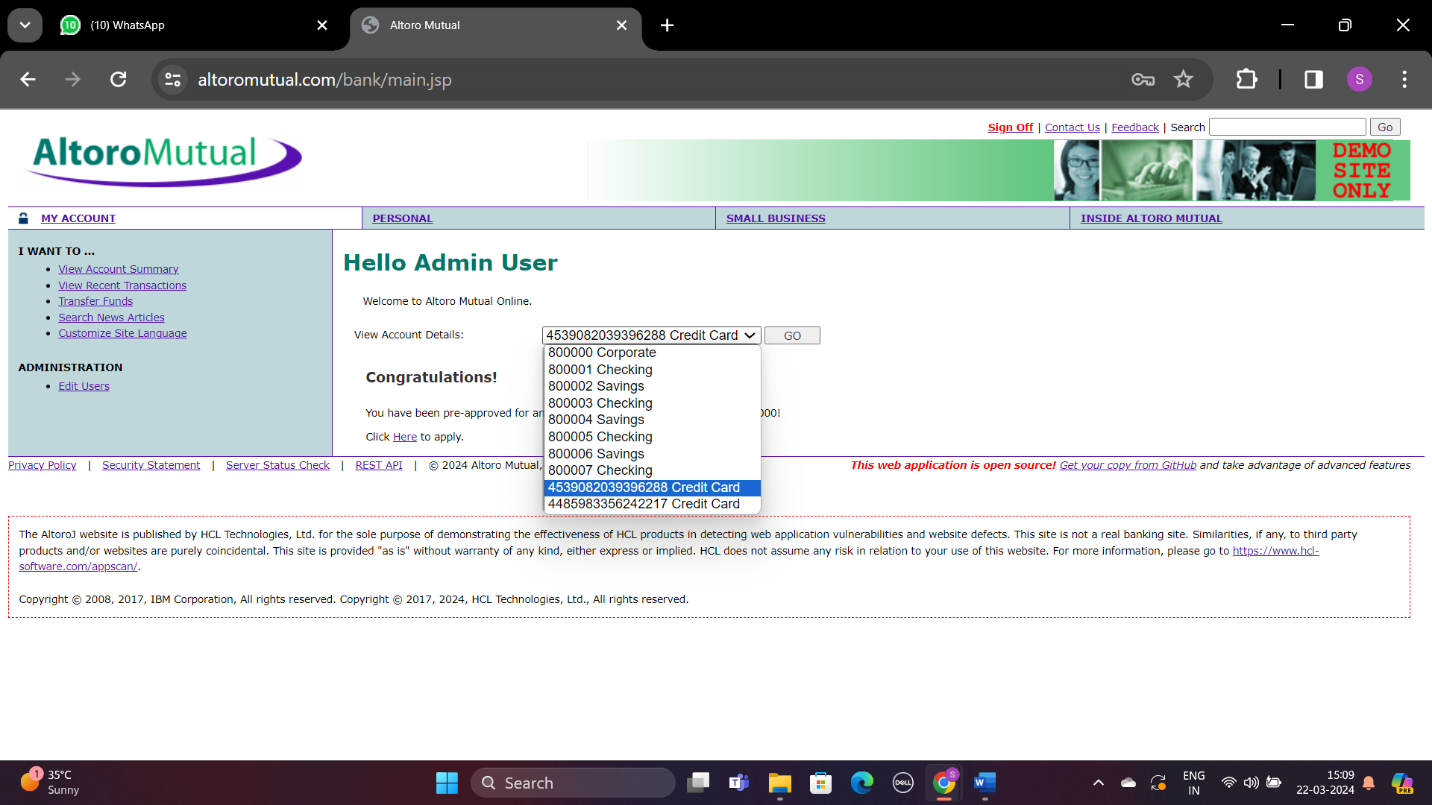
* Here we check if we can log in to the website. When we can login, the vulnerability is called *"Broken Authorization"*. This can lead to other vulnerabilities such as Sensitive Data Exposure and Broken Access Control.



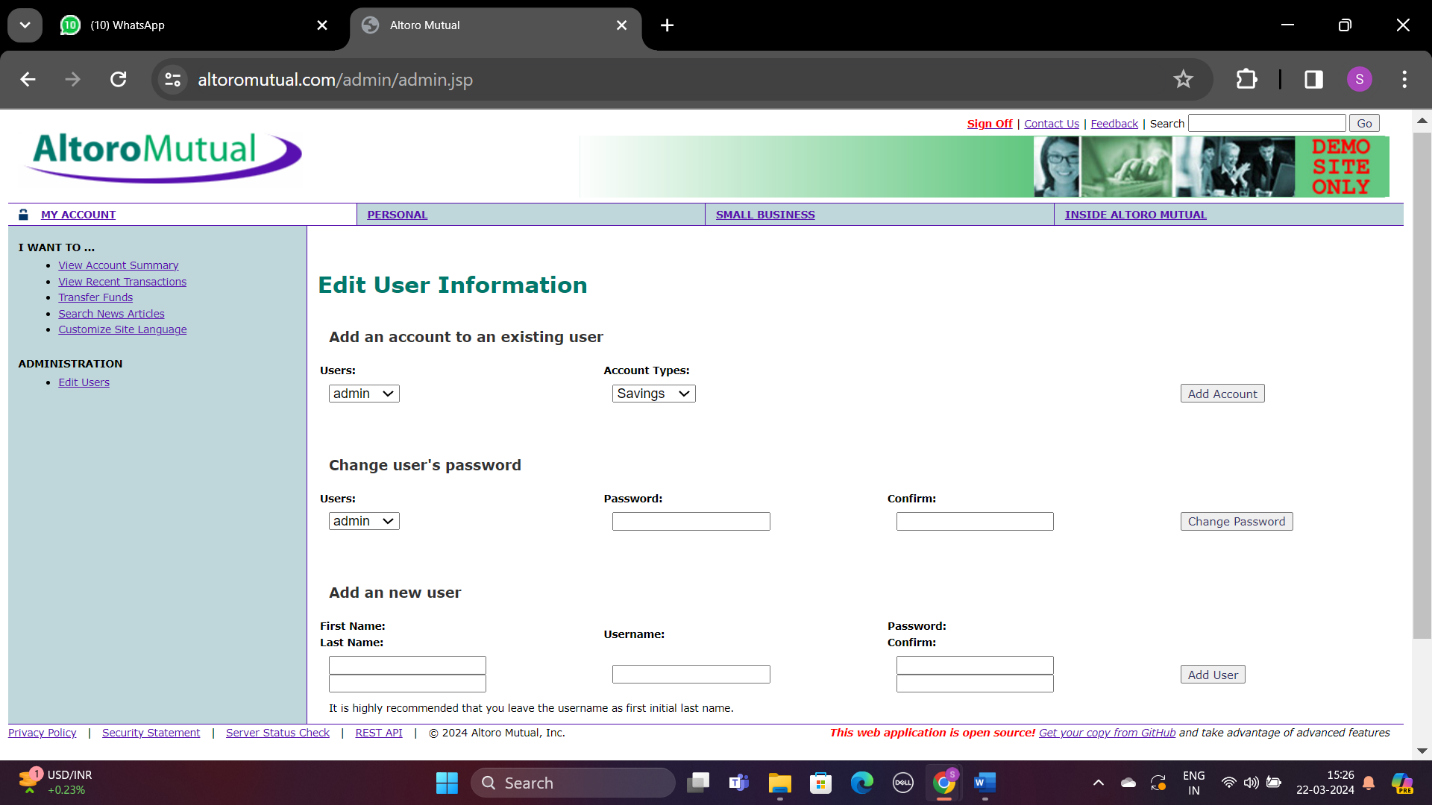
* + Here we used payload 1 ‘or’ 1 ‘=’ 1 as username and password and login as administrator. We can see other users. We can add users, change their passwords and manage their accounts.



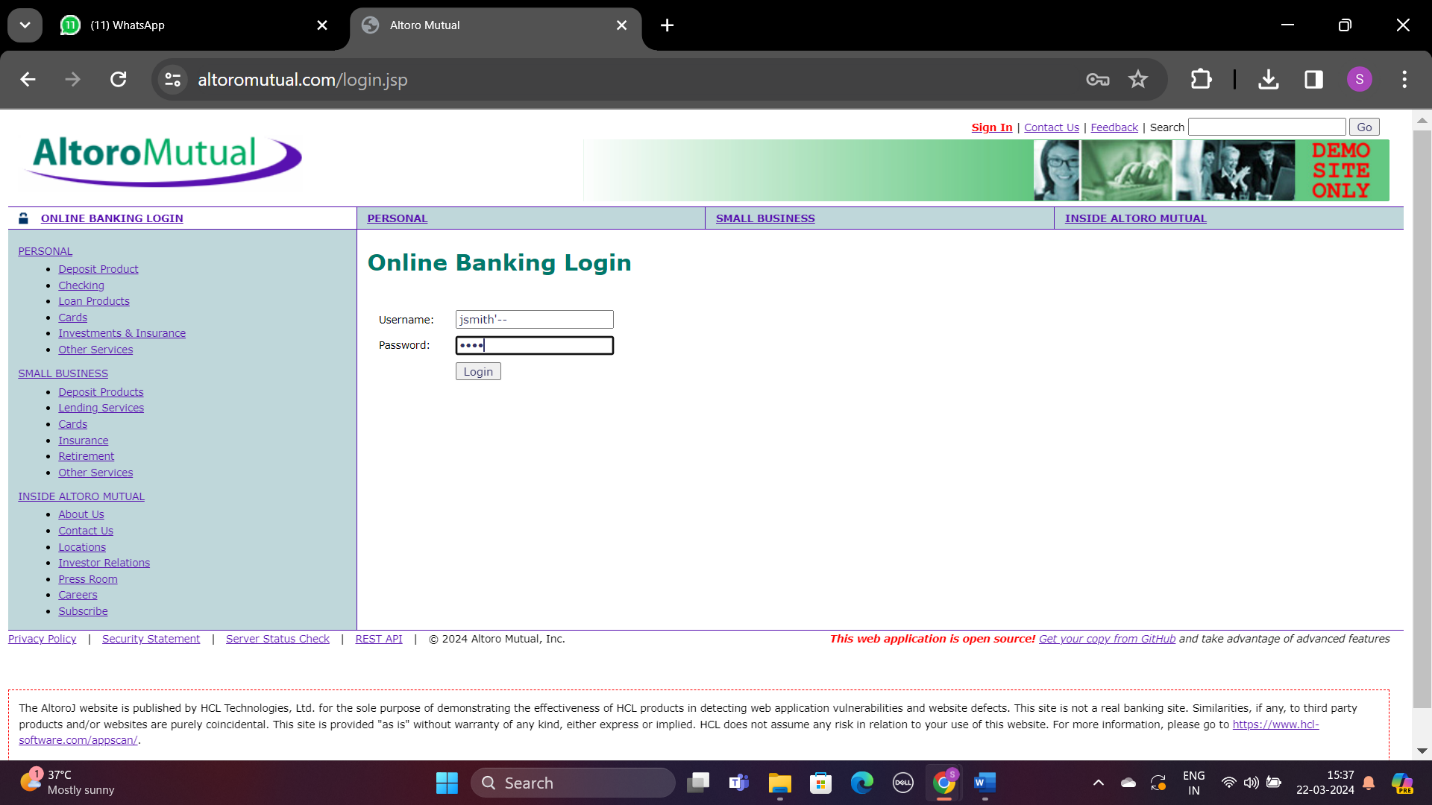
* + Here we can see the account information of bank users. The column highlighted in the image above shows the content pages for Months Transfer, Account Summary and Recent Transactions.



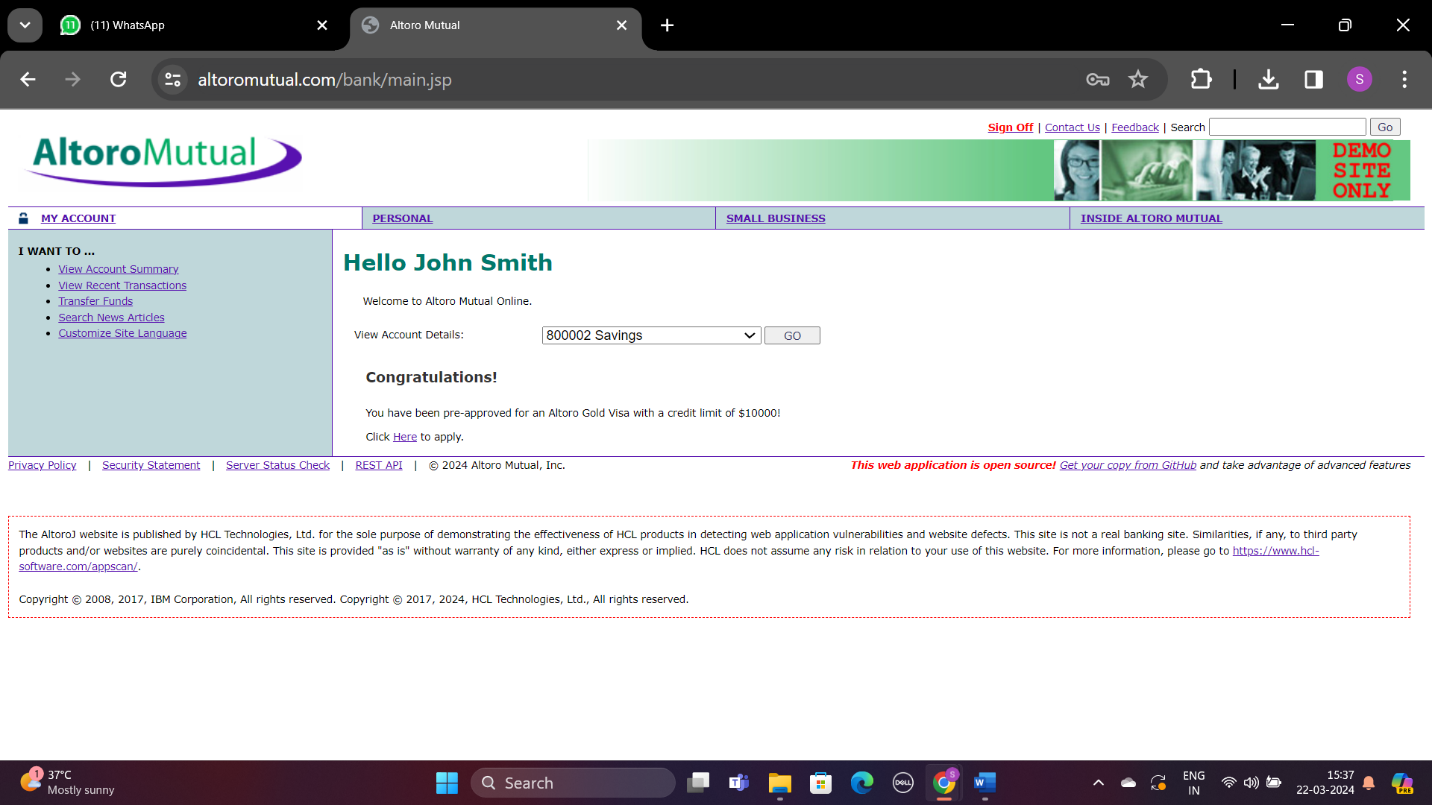
* + The above image shows us that we can edit users and their information.



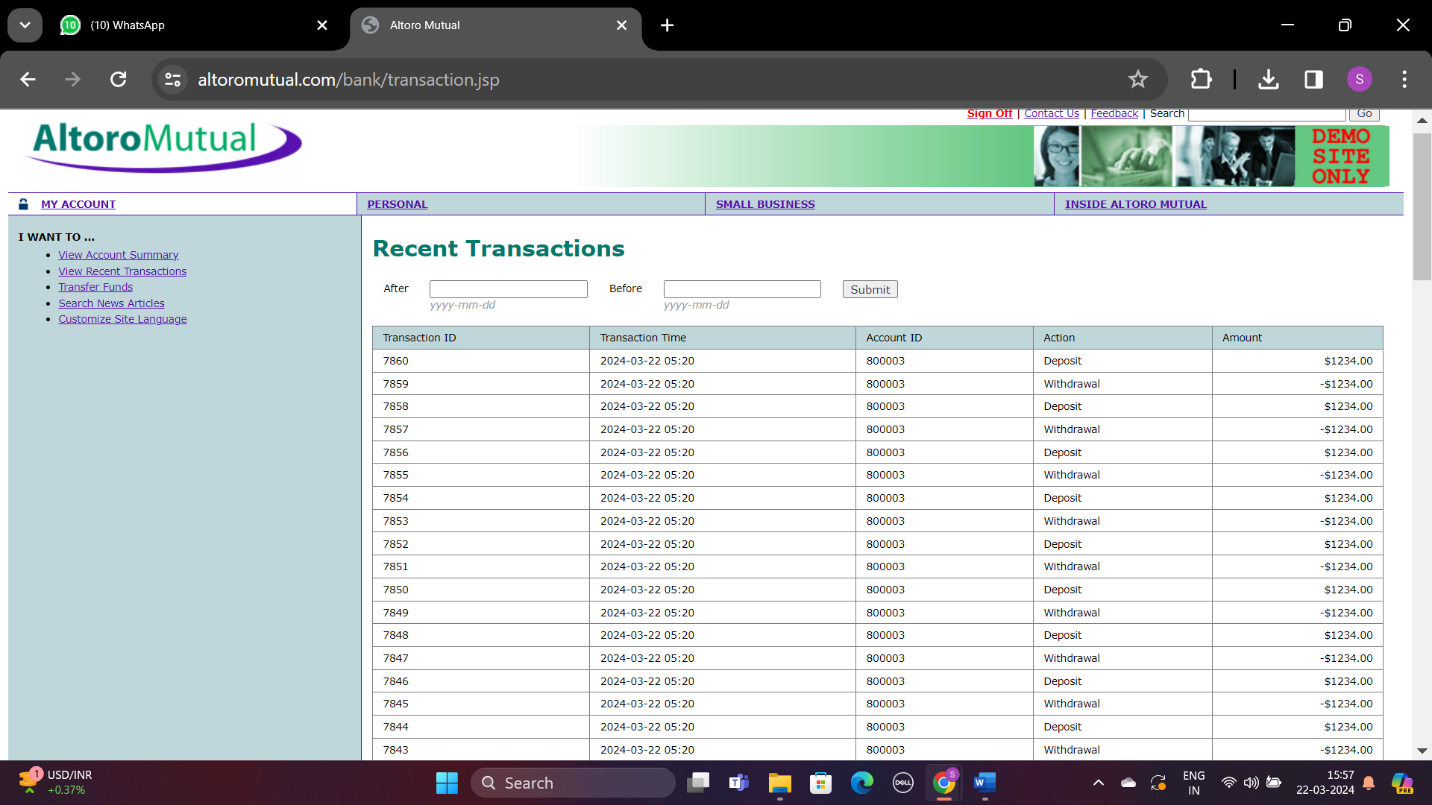
* + Based on the information we receive, we can change user’s passwords and gain access to their information.
* Next, let's look at user authorization levels.



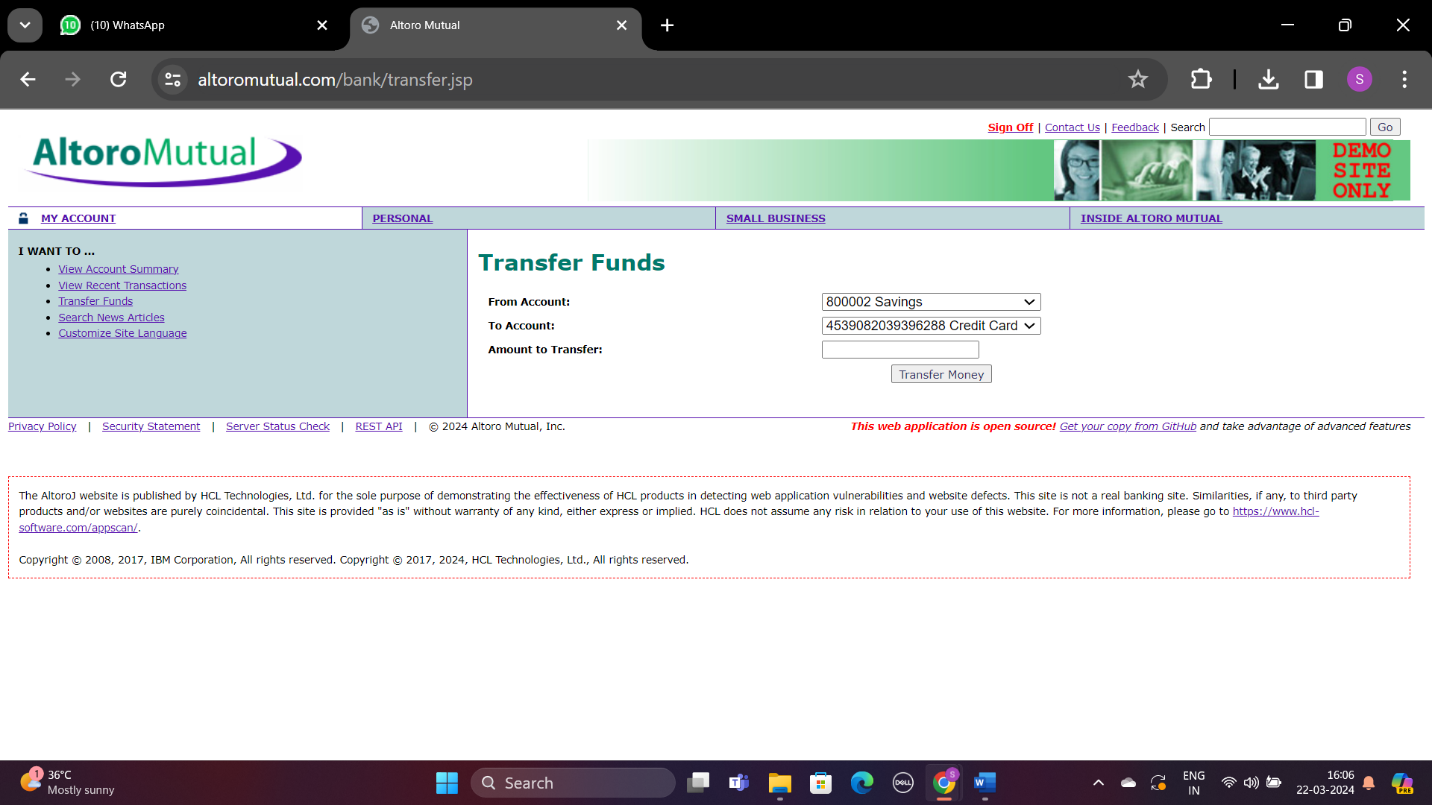
* + One of the users we found was "jsmith". But we don't know the password. To bypass this, we are trying to use "1234" as the password and using a payload that interferes with the authentication code. We create the payload by appending ‘-- to username which is “jsmith” here.



* + The image above shows a successful login with the username John Smith. This allows us to review his transaction history and manage money transfers to other accounts, as shown below.



* + We can track the financial activities of the users, including transaction history.



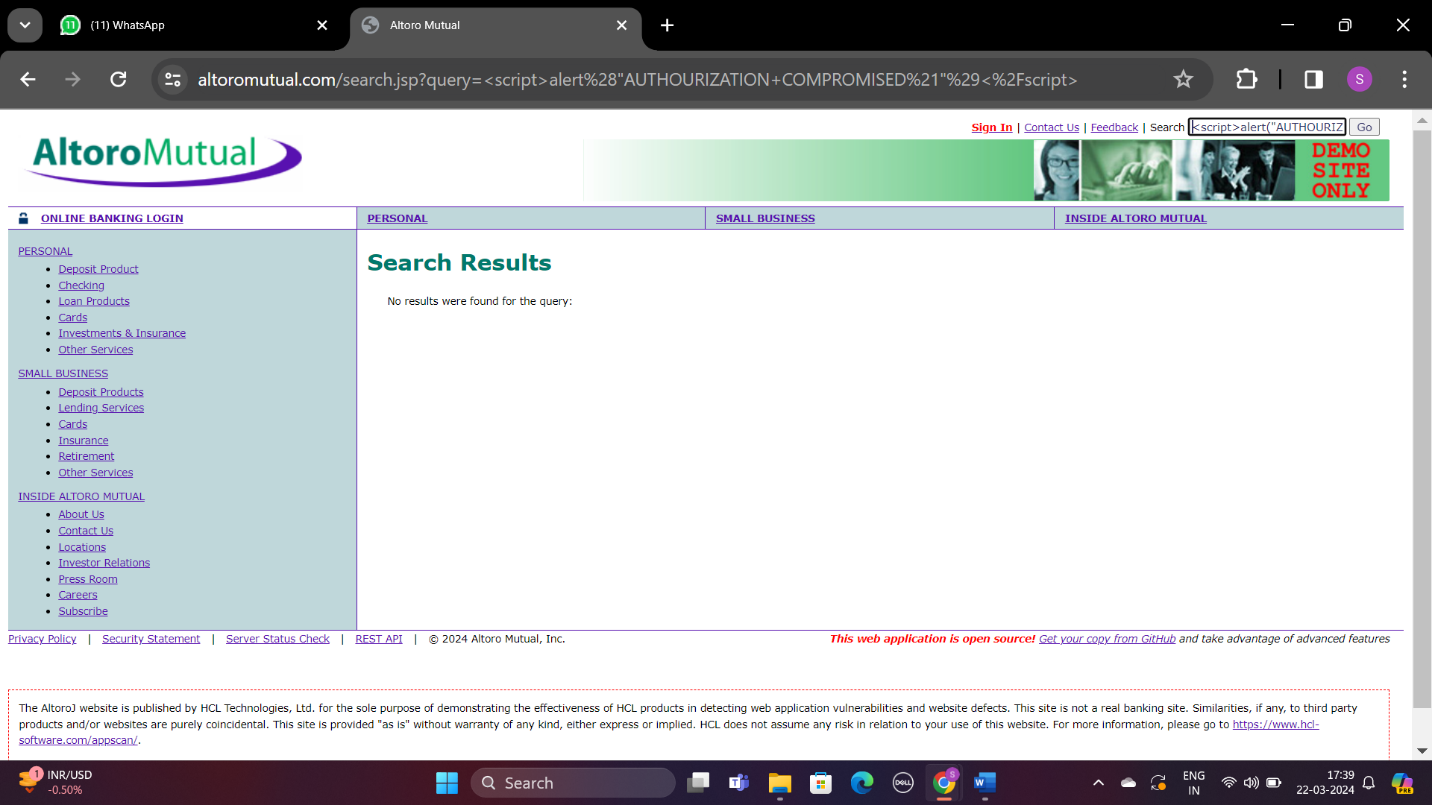
* + The above image illustrates user’s (jsmith) transferring funds to other accounts.
* It was found that there are vulnerabilities in the website such as Broken Authorization and Broken Access Control.

**Mitigations**

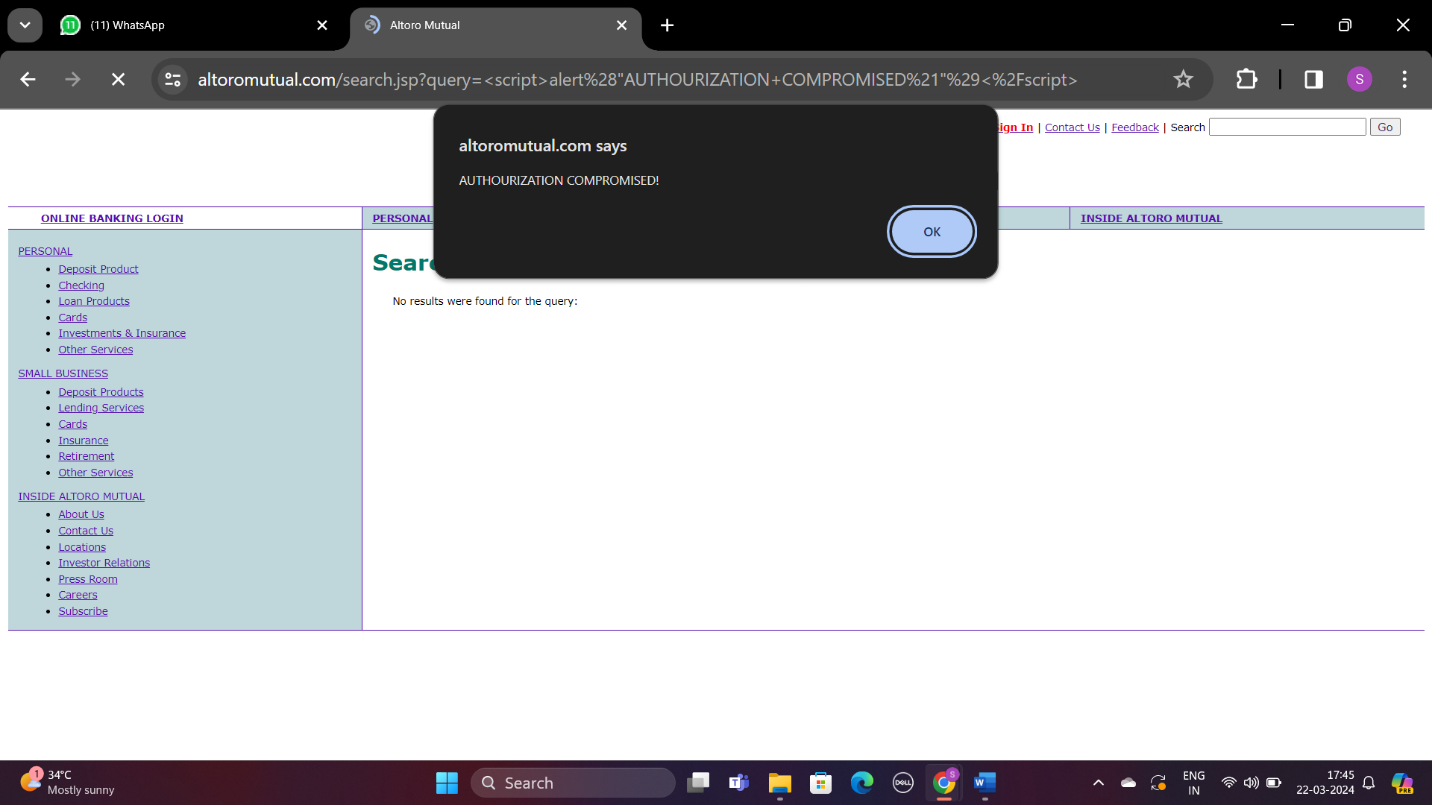
* **Configure Strong Authentication Protocols:**
  + Implement strong authentication, such as multifactor authentication (MFA), to limit system access to authenticated users. It uses advanced password protection techniques, such as salt hashing, to protect user information.
* **Implements the concept of Limited Privileges:**
  + Give users the maximum access they need for their work. Monitor and adjust permissions to ensure compliance with the principle of least privilege.
* **Enable Role-Based Access Control (RBAC):**
  + Use RBAC to define user actions and permissions to provide clear authority over what allows to do each job.
* **Utilize function attributes - Advanced Access Control (ABAC):**
  + Makes access decisions based on factors such as user characteristics, environmental characteristics, and resource characteristics. ABAC conditions and policies will change to adjust access.
* **Enable Access Rights Lists (ACLs) and add whitelists:**
  + Use ACLs to specify the access rights of users or groups to specific resources. Use a whitelist to restrict actions to only those that are explicitly allowed.
* **Secure Session Management:**
  + Securely manage session credentials from creation to confirmation using HTTPS and only HTTP cookies to protect session data from unauthorized tampering.
* **Run regularly Security Audits and Code Reviews :**
  + Conduct regular security audits, such as penetration tests and code reviews, to find and fix access control weaknesses using both automated and manual inspection methods.
* **Ensure proper error handling:**
  + Avoid exposing sensitive data information in error messages that can help exploit vulnerabilities. Use non-specific error messages for users while keeping detailed logs for administrators.
* **Secure APIs and Web Services:**
  + Secure APIs and web services with proper authentication and authorization using methods such as OAuth or JWT tokens for secure API interactions.
* **Access Event Monitoring and Audit:**
  + Monitor and record access events to identify and investigate unauthorized or suspicious activity. Regularly check usage logs and audits for potential security breaches.

**Cross-Site Scripting (XSS)**

* Now we are going to test Altoro Mutual's website for cross-site scripting (XSS) vulnerabilities. By adding malicious scripts to a website's search bar, we can determine whether it is vulnerable to XSS attacks based on how it reacts to the script.



* **Result Obtained:**



**Mitigations**

* **Enable input validation and sanitization:**
  + Enable on the server side that all user input is validated and sanitized in the expected format and does not contain malicious code.
* **Enable Content Protection Policy (CSP):**
  + Use CSP to determine which. resources that the browser must execute or load, which reduces the risk of running malicious scripts.
* **Practice encoding your output:**
  + Before displaying user-generated content in an HTML context, encode it so that executable code does not handle it in browsers.
* **Set cookies for HTTPOnly and protected flags:**
  + Use the HttpOnly flag in cookies to prevent access to client-side scripts, which helps protect against session identity theft via XSS.
* **Use frames with XSS Security:**
  + Regularly update framework dependencies to take advantage of the latest security features and fixes.
* **Use context-sensitive output prevention:**
  + Use output coding tailored to the data display context, such as HTML, JavaScript, or CSS for targeted XSS prevention.
* **Conduct regular security training:**
  + Educate developers about secure coding practices and XSS threats, promote the use of secure HTML templates, and avoid mixing untrusted information with HTML.
* **Use a security analyzer:**
  + Use static analysis and web vulnerability scanning tools to detect code and XSS errors in web applications. Conduct continuous security assessments and penetration tests to proactively combat XSS risks.
* **Follow the Secure Development Life Cycle (SDLC):**
  + Integrate security-focused coding standards and best practices throughout the development process to reduce the likelihood of XSS vulnerabilities.