ALTORO-MUTUAL-VULNERABILITY-ANALYSIS-REPORT

KRISHNA SATHVIKA GANNI
4TH YEAR,B.TECH
RCEE

STEPS:

Step 1: OWASP Top 10 Vulnerabilities Overview:

1. **Broken Access Control:** Flaws in authentication and access control can allow unauthorized access to sensitive data or systems. Using Infrastructure as Code (IaC) tools can help detect configuration errors leading to access control failures.
2. **Cryptographic Failures:** Mistakes like hardcoded passwords or weak encryption methods can expose sensitive data. Scanning for hardcoded secrets and ensuring proper encryption can mitigate these risks.
3. **Injection:** Attackers exploit vulnerabilities in web applications to inject malicious code, like SQL injection or Cross-Site Scripting (XSS). Application security testing can help detect these flaws.
4. **Insecure Design:** Focuses on fundamental design flaws rather than implementation issues. Secure design practices, developer training, and threat modeling are essential to prevent such vulnerabilities.
5. **Security Misconfiguration:** Errors in server, framework, or cloud infrastructure configurations can lead to breaches. Regularly hardening configurations and scanning for misconfigurations are crucial for mitigation.
6. **Vulnerable and Outdated Components:** Third-party libraries and components can introduce vulnerabilities. Building a Software Bill of Materials (SBoM) and using tools for vulnerability management can help track and mitigate risks.
7. **Identification and Authentication Failures:** Weaknesses in user identification and authorization processes can be exploited. Secure coding practices and tools for detecting credential stuffing and brute force attacks are essential for protection.
8. **Software and Data Integrity Failures:** Attackers can exploit the

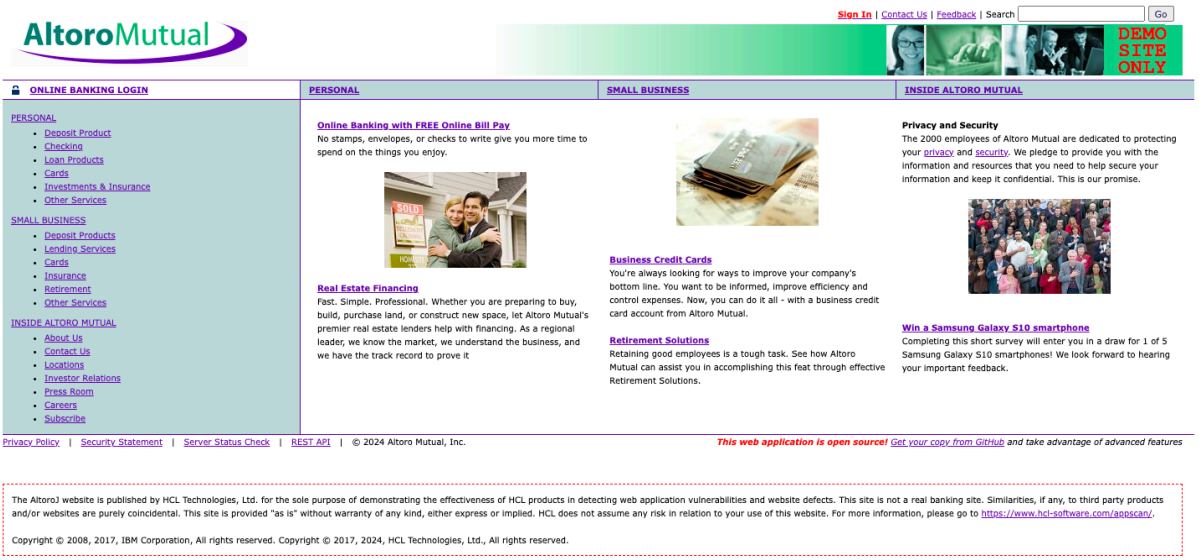
software build process to inject malicious code or steal secrets. Ensuring the security of the build process and components used is crucial for mitigating this threat.

9. **Security Logging and Monitoring Failures:** Adequate logging and monitoring are essential for detecting and responding to security breaches. Regular verification of logging and alerting processes is necessary for effective incident response.
10. **Server-Side Request Forgery (SSRF):** Attackers can manipulate web applications to send requests to unintended destinations. Mitigation involves sanitizing user input and inspecting request responses to prevent SSRF attacks.

The potential impact of these vulnerabilities on web application security and the importance of addressing them to prevent exploitation by attackers:

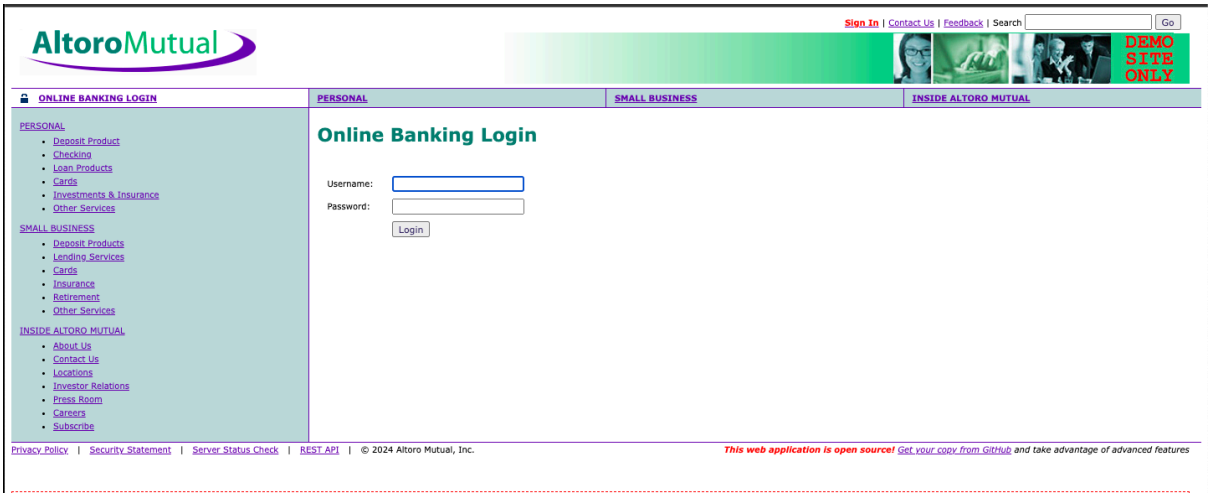
1. **Broken Access Control:** Unauthorized access can lead to data breaches and financial loss.
2. **Cryptographic Failures:** Weak encryption can expose sensitive data, leading to theft and reputational damage.
3. **Injection:** Attacks can manipulate data, steal information, and compromise systems.
4. **Insecure Design:** Fundamental flaws can result in widespread vulnerabilities, causing data breaches and system downtime.
5. **Security Misconfiguration:** Misconfigurations can lead to unauthorized access and compliance violations.
6. **Vulnerable and Outdated Components:** Exploiting vulnerabilities can lead to system compromise and reputational damage.
7. **Identification and Authentication Failures:** Weak authentication can result in unauthorized access and fraud.
8. **Software and Data Integrity Failures**: Compromised software integrity can lead to malware distribution and financial losses.
9. **Security Logging and Monitoring Failures:** Inadequate monitoring can delay detection of security incidents, allowing attackers to cause further damage.
10. **Server-Side Request Forgery (SSRF):** Exploitation can lead to unauthorized access and data exfiltration.

Step 2: Altro Mutual Website Analysis:

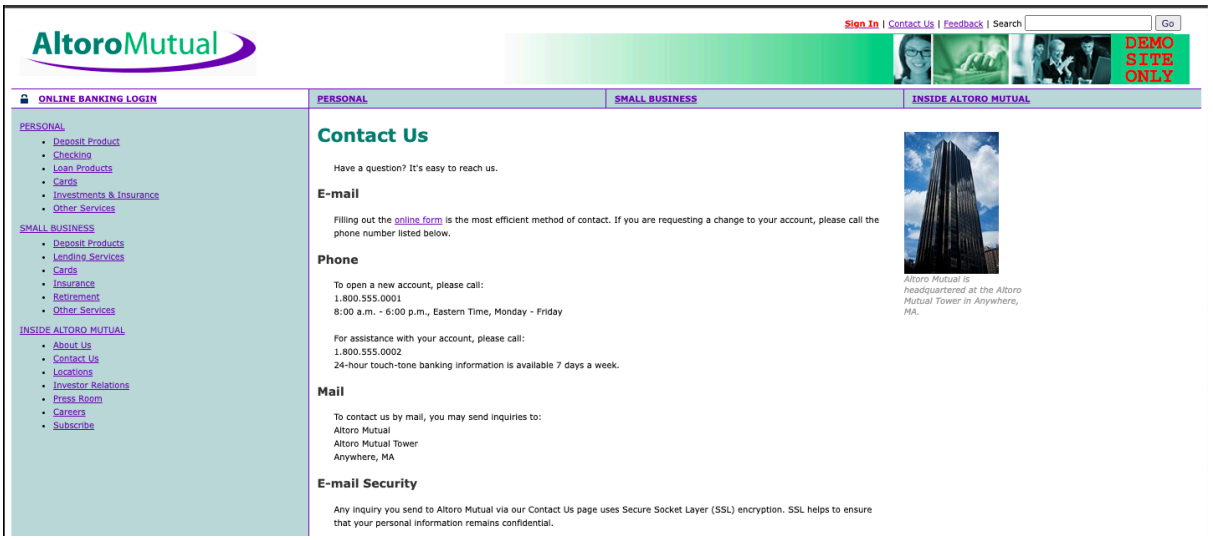


It's the main page of the Altoro Mutual website. It features "feedback," "contact," and "sign in" options. The search feature should be added after logging in and deleted in order to facilitate component searches on the website. An attack using "cross-site scripting" could be possible.
There are several categories, including personal, small company, and inside Alotoro Mutual, as seen in the graphic below. The left side column lists the sub-divisions for each category.

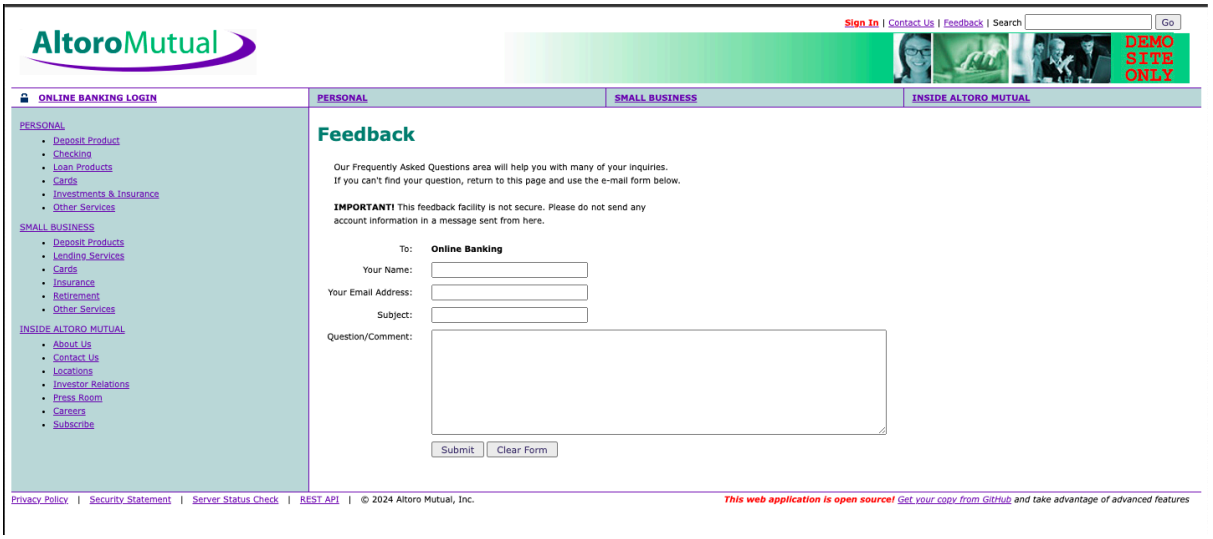
login page:



Contact page:



Feedback form:

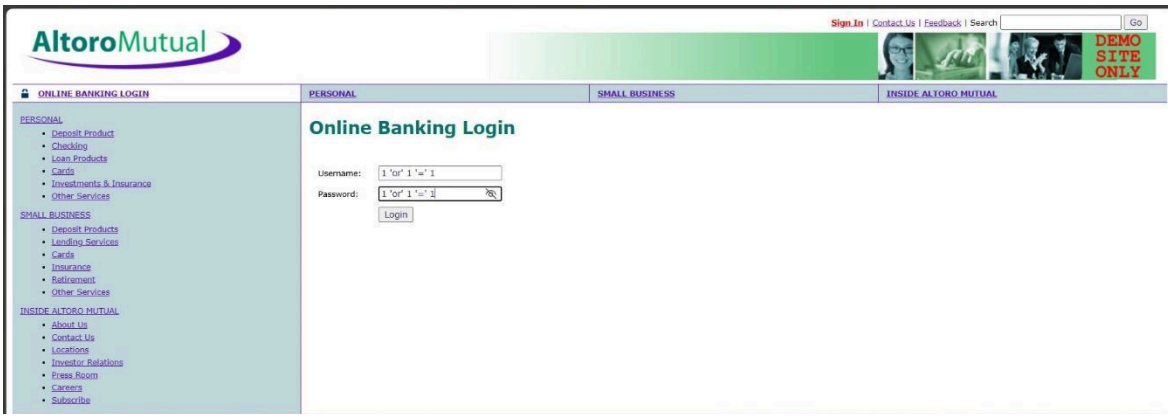


Step 3: Vulnerability Identification Report:

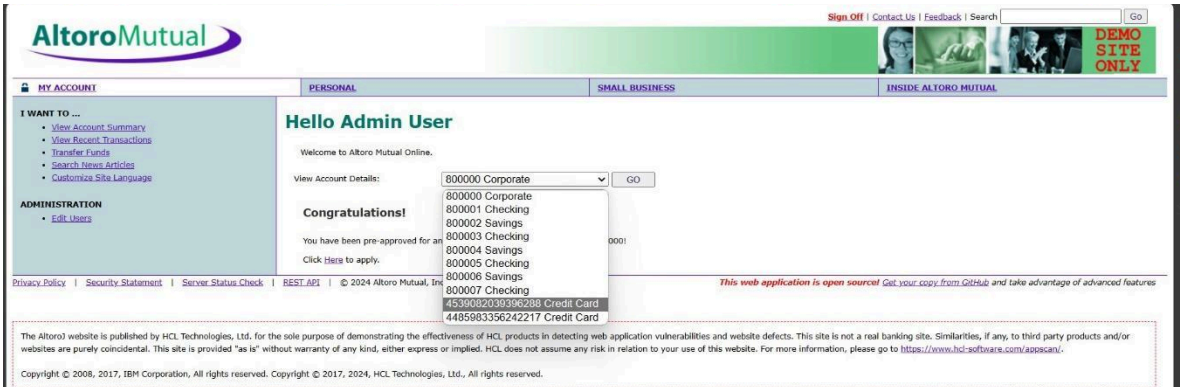
➔ The report includes a detailed description of Altro Mutual's website structure and functionality, including potential areas of vulnerability.

Step 4: Vulnerability Exploitation Demonstration:

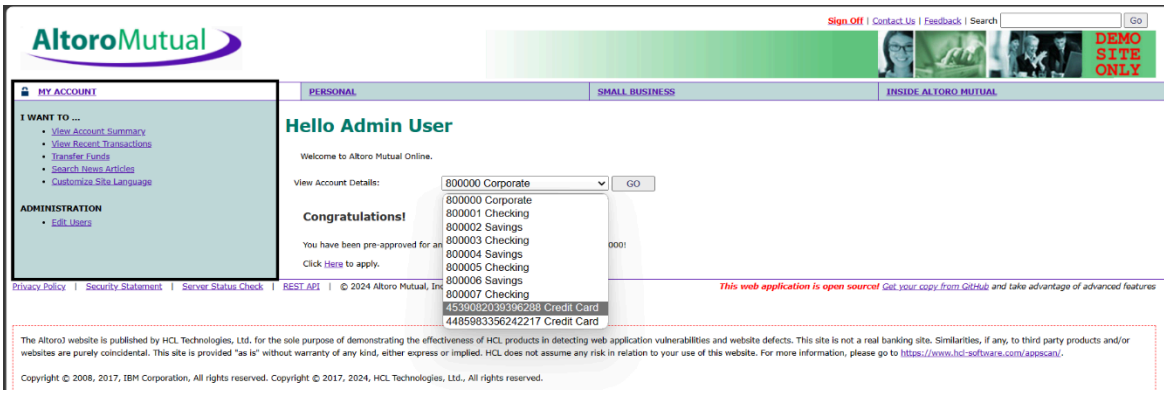
Here, we'll see if we can log in to the website. If so, we have a vulnerability known as "Broken Authorization," which can lead to other vulnerabilities like Broken Access Control and Sensitive Data Exposure.



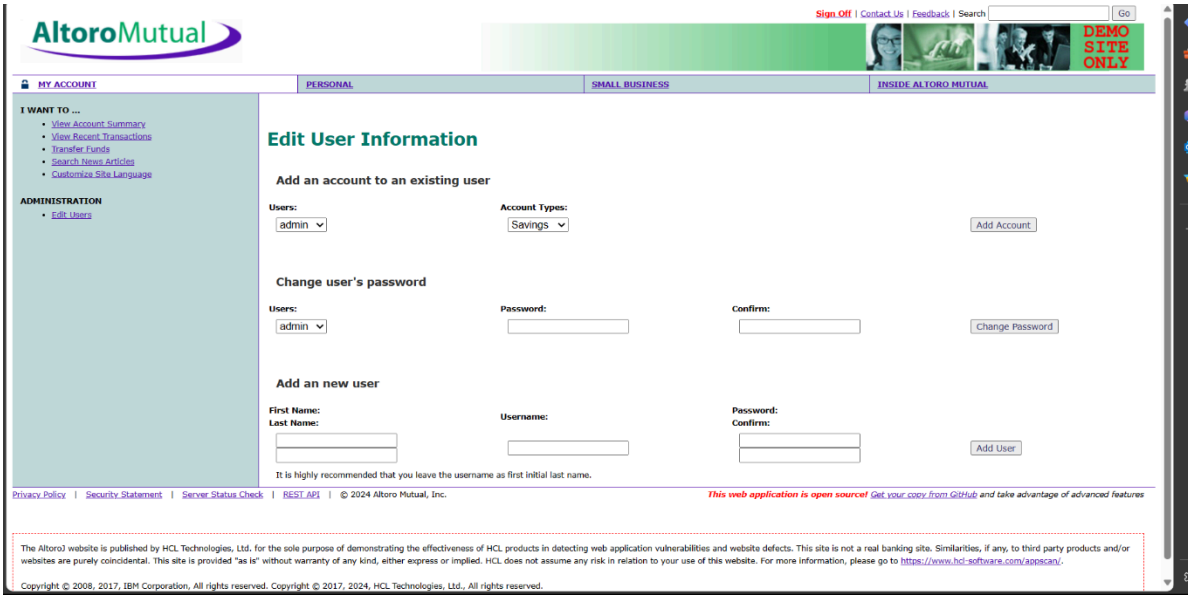
Here, we logged in as admin and used the payload 1 "or" 1 "=" 1 as the login and password. Other users are visible to us. We have the ability to add users, modify their passwords, and manage their accounts.



Users of the bank can view their account details here. The content pages for fund transfers, account summaries, and recent transactions are shown in the image below in the highlighted column.

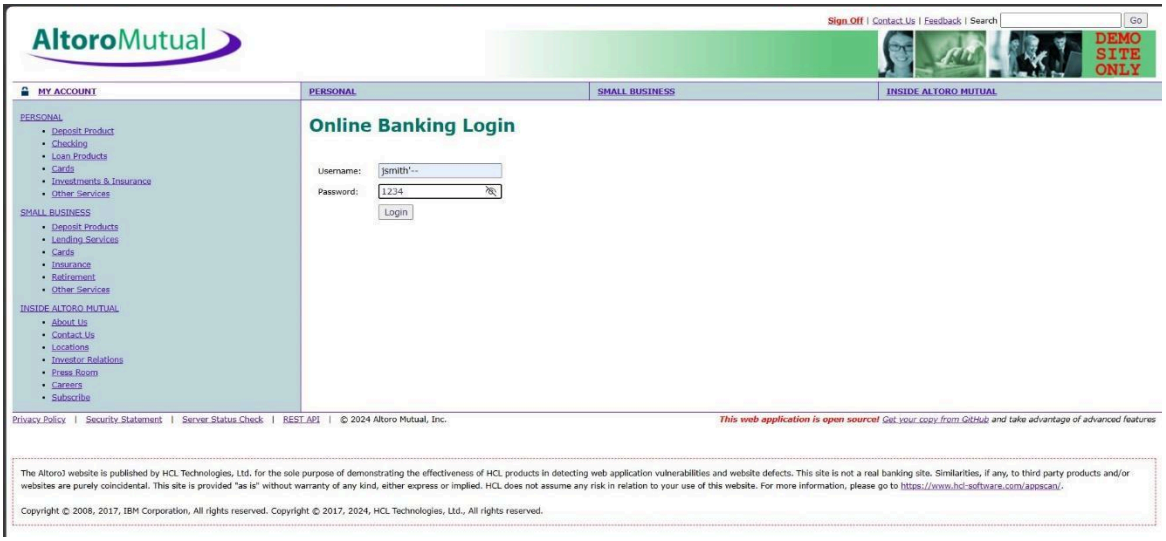


We are able to edit users and their details, as seen in the image above. These are depicted in the graphic below.



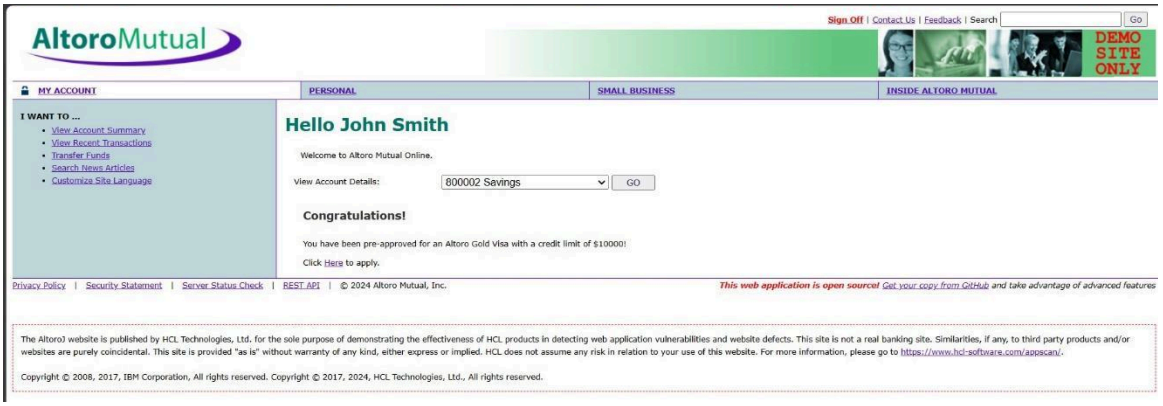
We obtained users from above. We have access to users' data and can reset their password.

We will now verify the user's authorization.

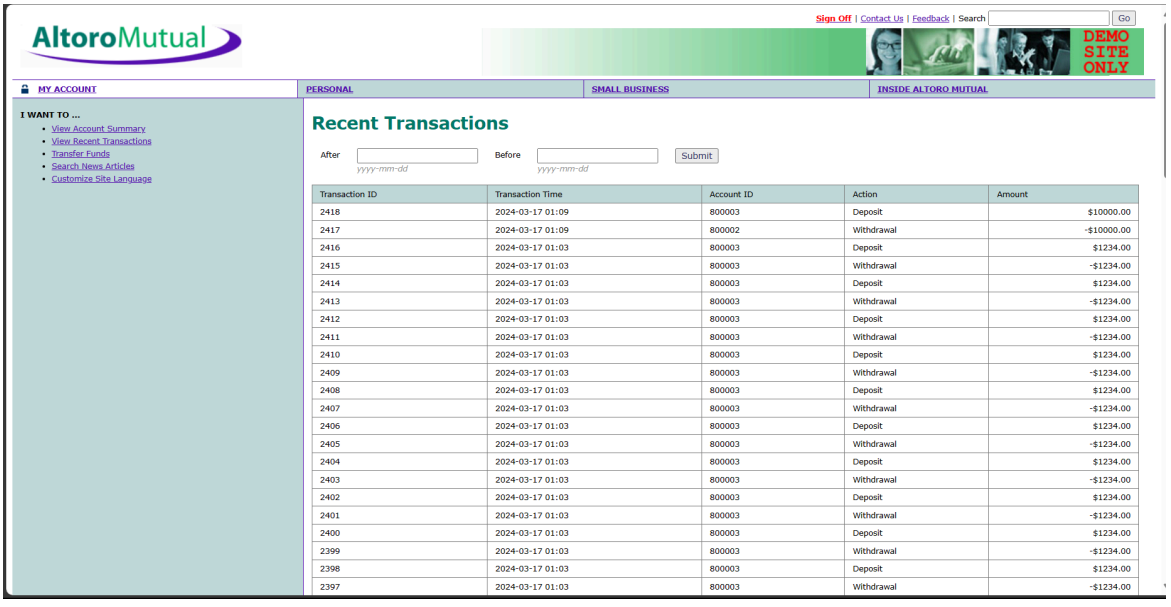


"Jsmith" is among the users that we discovered. However, we are utilising a payload to conflate the permission code and are using the password "1234" because we do not know the password.

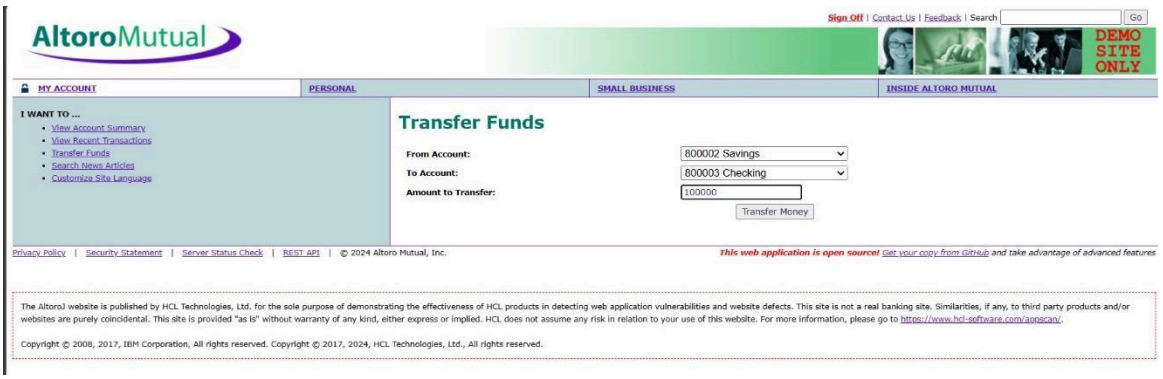
The real username, "jsmith," is attached with '--' as the payload.



We have access to user transactions as well as their transaction history.



Transferring funds:



➔ Students can demonstrate how each identified vulnerability could be exploited using proof-of-concept attacks or simulation tools.

➔ For example, they could demonstrate how SQL injection attacks can be used to extract sensitive information from the database or how cross-site scripting (XSS) attacks can be used to execute malicious scripts in users' browsers.

Step 5: Mitigation Strategy Proposal:

1. **Broken Access Control:**
    a. Implement proper authentication and authorization mechanisms.
    b. Enforce least privilege access.
    c. Regularly audit and review access controls.
2. **Cryptographic Failures:**
    a. Use strong encryption algorithms and secure storage methods.
    b. Avoid hardcoded passwords and keys.
    c. Regularly update cryptographic libraries and protocols.
3. **Injection:**
    a. Use parameterized queries and input validation to prevent SQL injection.

    b. Employ output encoding to mitigate Cross-Site Scripting (XSS).
    c. Implement strict OS command filtering to prevent command injection.
4. **Insecure Design:**
    a. Conduct thorough threat modeling and security reviews during the design phase.
    b. Follow secure coding practices and principles. Implement secure design patterns and controls.
5. **Security Misconfiguration:**
    a. Follow secure configuration guidelines for servers, frameworks, and cloud services.
    b. Regularly audit and update configurations.
    c. Utilize automated tools for configuration scanning and validation.
6. **Vulnerable and Outdated Components:**
    a. Maintain an inventory of third-party components and libraries. Regularly update and patch components to address known vulnerabilities.
    b. Use Software Bill of Materials (SBoM) to track dependencies and vulnerabilities.
7. **Identification and Authentication Failures**:
    a. Implement multi-factor authentication (MFA) where possible.
    b. Enforce strong password policies and account lockout mechanisms.
    c. Monitor for abnormal login activities and brute force attempts.
8. **Software and Data Integrity Failures:**
    a. Implement code signing and verification mechanisms.
    b. Employ secure build and deployment pipelines.
    c. Regularly scan code for vulnerabilities and malicious code.
9. **Security Logging and Monitoring Failures:**
    a. Implement comprehensive logging across systems and applications.
    b. Set up alerting mechanisms for suspicious activities.
    c. Regularly review logs and conduct incident response drills.
10. **Server-Side Request Forgery (SSRF):**
    a. Validate and sanitize user-supplied URLs to prevent SSRF.
    b. Use whitelists to restrict allowed destinations for outbound requests.
    c. Implement rate limiting and request validation mechanisms.

Step 6: Documenting the Exploit Process:

➔ Document the exploit process, including the commands used, the output received, and any challenges encountered.