Name: GUMMALA HARI NAIDU
Regd.No: 721128805329
College: Dr.Lankapalli BullayyaCollege

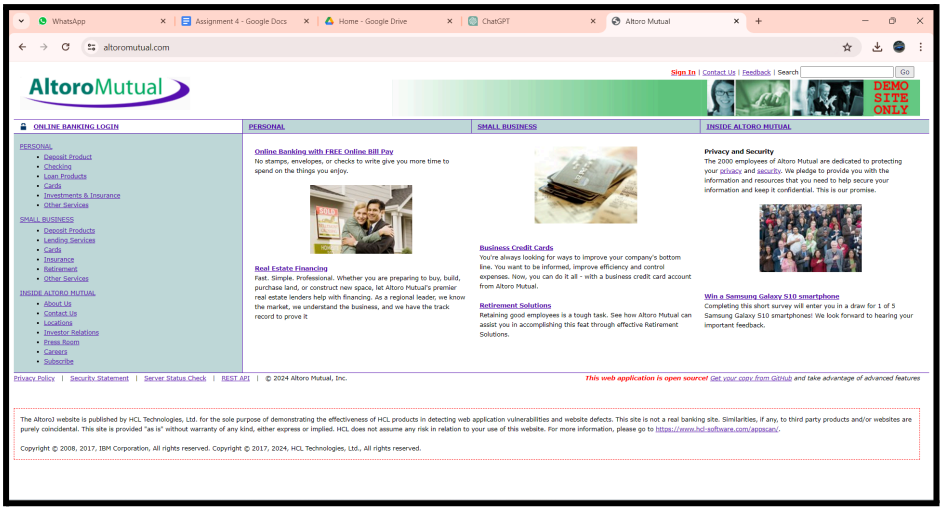
Assignment 4

Step 1: OWASP Top 10 Vulnerabilities Overview

1. Injection: Injection flaws occur when untrusted data is sent to an interpreter as part of a command or query. This can result in an attacker executing unintended commands or accessing unauthorized data.
2. Broken Authentication: This vulnerability arises when an application does not correctly manage authentication and session management, allowing attackers to compromise passwords, keys, or session tokens.
3. Sensitive Data Exposure: This occurs when an application fails to adequately protect sensitive data, such as financial information or personal identifiers. Attackers can exploit this vulnerability to access such data.
4. XML External Entities (XXE): XXE vulnerabilities arise when an application parses XML input from untrusted sources. Attackers can exploit this to access sensitive data, execute remote code, or perform denial of service attacks.
5. Broken Access Control: This vulnerability occurs when restrictions on what authenticated users can do are not properly enforced. Attackers can exploit this to access unauthorized functionality or data.
6. Security Misconfiguration: Security misconfigurations happen when an application is not securely configured, leaving it vulnerable to attack. This could include default configurations, open cloud storage, or unnecessary features enabled.
7. Cross-Site Scripting (XSS): XSS vulnerabilities occur when an application includes untrusted data in a web page without proper validation or escaping. Attackers can exploit this to execute scripts in the victim's browser, potentially stealing cookies or other sensitive information.
8. Insecure Deserialization: Insecure deserialization vulnerabilities arise when untrusted data is used to abuse the logic of an application, leading to remote code execution or other attacks.
9. Using Components with Known Vulnerabilities: This occurs when an application uses components (such as libraries or frameworks) with known vulnerabilities. Attackers can exploit these vulnerabilities to compromise the application.

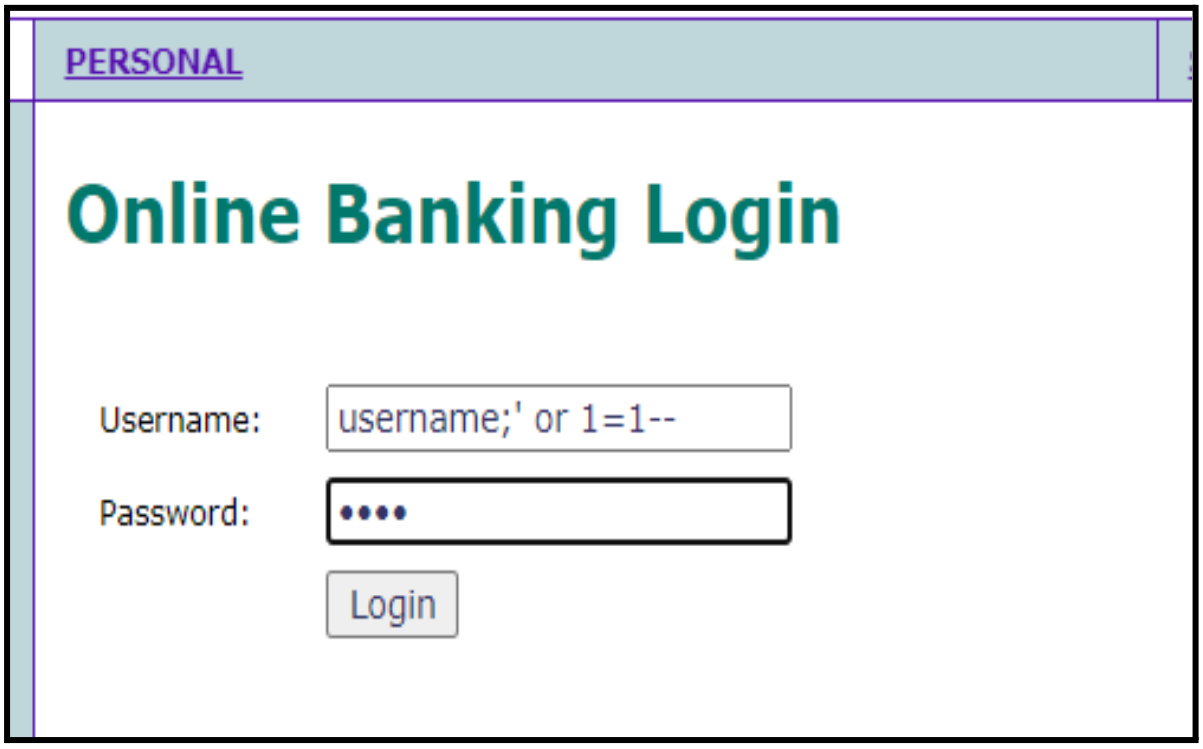
10. Insufficient Logging and Monitoring: When an application lacks sufficient logging and monitoring, it becomes difficult to detect and respond to security incidents. Attackers can exploit this to maintain persistence in the system or to cover their tracks.

Step 2: Altro Mutual Website Analysis

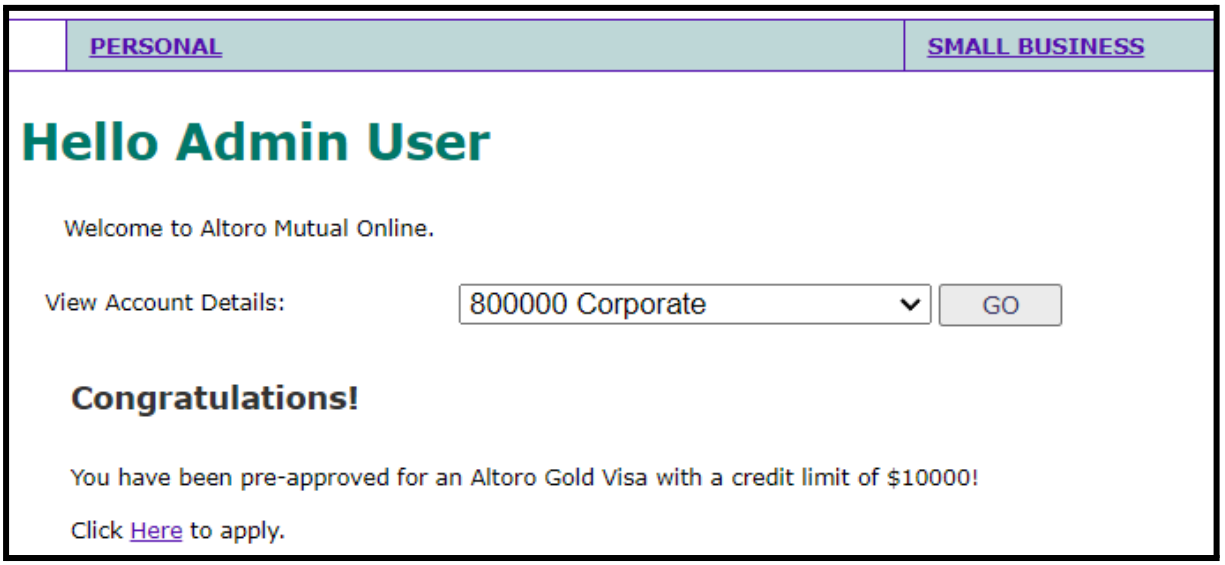


Altoro Mutual is a subsidiary of Altoro, a multi-state holding company located in the heart of Massachusetts. Altoro Mutual offers a broad range of commercial, private, retail and mortgage banking services to small- and middle-market businesses and individuals.

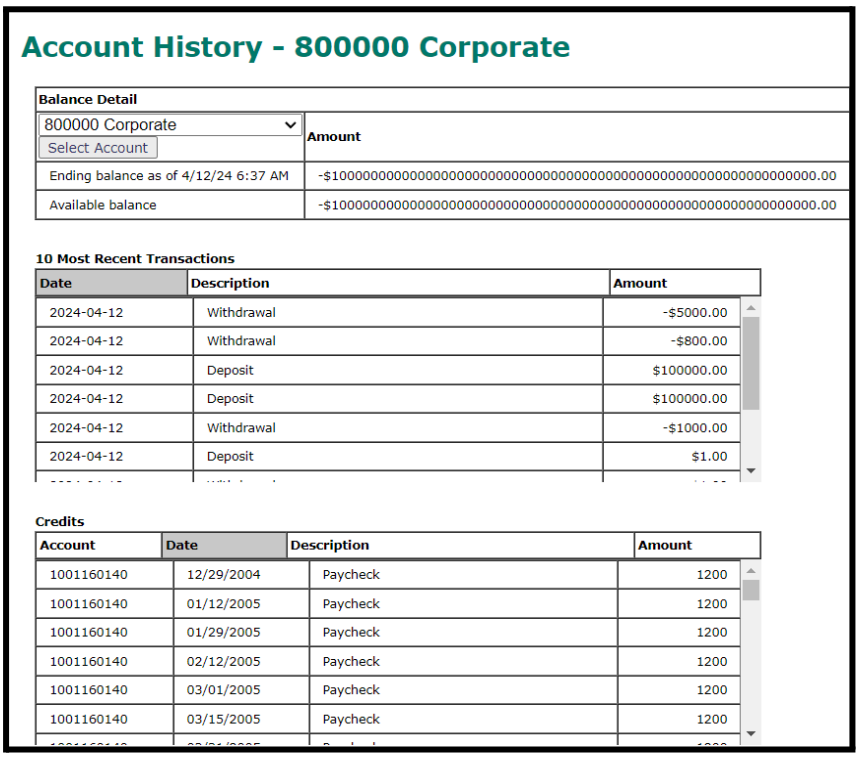
Step 3: Vulnerability Identification Report:



Yet we used invalid login credentials but it allowed to log in



This vulnerability is termed as Broken Authentication

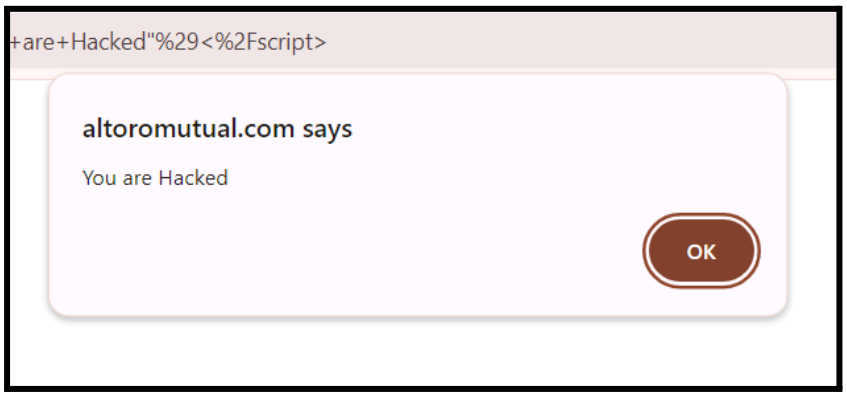


We also got access to account data, this vulnerability is called Sensitive Data Exposure.

Step 4: Vulnerability Exploitation Demonstration

For this demonstration we inject a script to altoro mutual site

It is - ***<script>alert("You are hacked")</script>***



This is termed as cross-site scripting .

Step 5: Mitigation Strategy Proposal:

Sensitive Data Exposure:

- Data Encryption: Encrypt sensitive data both at rest (stored data) and in transit (data transmitted over networks). Use strong encryption algorithms and ensure that encryption keys are managed securely.
- Secure Data Storage: Store sensitive data securely, following industry best practices and compliance standards. This may include using secure databases, encrypting data fields, and protecting data backups. Regularly review and audit data storage mechanisms to ensure compliance with security policies and standards.
- Input Validation and Sanitization: Implement robust input validation and data sanitization techniques to prevent injection attacks, such as SQL injection, NoSQL injection, or XSS (Cross-Site Scripting) attacks. Use parameterized queries and prepared statements to mitigate SQL injection vulnerabilities.

Broken Authentication:

- Session Management:
    - Implement secure session management practices, including session timeouts, session regeneration after authentication, and secure cookie attributes (e.g., HttpOnly, Secure).
    - Use secure protocols (e.g., HTTPS) to encrypt session data during transmission.
- Credential Handling:
    - Follow secure password storage practices, such as hashing passwords with strong and industry-standard hashing algorithms (e.g., bcrypt, Argon2).
    - Avoid storing sensitive authentication credentials (e.g., passwords) in clear text or weakly encrypted formats.

- User Education and Awareness:
    - Educate users about best practices for creating and managing secure passwords, recognizing phishing attempts, and protecting their accounts from unauthorized access.