ASSIGNMENT -4

➢ OSWAP TOP 10 VULNERABILITIES OVERVEIW:

The OWASP (Open Web Application Security Project) Top 10 is a widely recognized list of the top 10 most critical web application security risks. These vulnerabilities represent some of the most common and impactful threats to web application security. Here's an overview of the OWASP Top 10 vulnerabilities:

- **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. The attacker's malicious data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

- **Broken Authentication**: This vulnerability occurs when web applications improperly authenticate users, manage session IDs, and handle passwords. Attackers can exploit these flaws to steal user credentials, session tokens, or take over user accounts.

- **Sensitive Data Exposure**: When web applications fail to properly protect sensitive data, such as financial information, personal data, or credentials, it can be exposed to attackers. This vulnerability can lead to identity theft, fraud, or unauthorized access.

- **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 denial of service attacks.

- **Broken Access Control**: Broken access control vulnerabilities arise when restrictions on what authenticated users are allowed to do are not properly enforced. Attackers can exploit these flaws to gain unauthorized access to functionalities and data.

- **Security Misconfiguration**: Security misconfiguration occurs when security settings are not implemented correctly, leaving vulnerabilities that attackers can exploit. Common examples include default configurations, open cloud storage, and verbose error messages.

- **Cross-Site Scripting (XSS):** XSS vulnerabilities arise when an application includes untrusted data in a web page without proper validation or escaping. Attackers can use this vulnerability to execute malicious scripts in the context of a victim's browser.

- **Insecure Deserialization**: Insecure deserialization vulnerabilities occur when an application deserializes untrusted data without proper validation. Attackers can exploit this vulnerability to execute arbitrary code, tamper with data, or conduct denial of service attacks.

- **Using Components with Known Vulnerabilities**: When developers use components, such as libraries or frameworks, with known vulnerabilities, attackers can exploit these vulnerabilities to compromise the application. It is important to keep components up to date and to patch known vulnerabilities.

- **Insufficient Logging and Monitoring**: Insufficient logging and monitoring make it difficult to detect and respond to security incidents. Attackers can exploit this by conducting attacks without being detected, leading to further compromise.

By addressing these vulnerabilities, organizations can improve the security posture of their web applications and reduce the risk of exploitation by attackers.

These vulnerabilities is crucial for several reasons in Real Time :

• Protecting User Data: Web applications often handle sensitive user information. Addressing vulnerabilities such as injection attacks, broken authentication, and sensitive data exposure helps safeguard this data from unauthorized access or theft.

• Maintaining Trust: Security breaches can severely damage an organization's reputation and erode user trust. By proactively addressing vulnerabilities, organizations demonstrate their commitment to protecting user data and maintaining a secure online environment.

• Legal and Regulatory Compliance: Many industries are subject to regulations regarding data protection and privacy, such as GDPR (General Data Protection Regulation) or HIPAA (Health Insurance Portability and Accountability Act). Failure to address vulnerabilities and protect user data may result in legal consequences and financial penalties.

• Preventing Financial Loss: Security breaches can lead to financial losses resulting from theft, fraud, or legal liabilities. Investing in security measures to mitigate vulnerabilities can help prevent these losses and protect the organization's financial interests.

Altro Mutual website Analysis :

To analyze the Altro Mutual website from a cybersecurity perspective, I would typically perform a detailed review of its architecture, functionality, and implementation. Since I can't access the specific website, I'll provide a general overview of the key areas I would focus on:

1. Secure Communication : check if the website uses HTTPS to encrypt data transmitted between the user's browser and the server. This helps protect against eavesdropping and data tampering.

2. Authentication and Access Control: Evaluate the website's authentication mechanisms to ensure they are secure and that access control is properly enforced. This includes checking for strong password policies, multi-factor authentication (MFA), and proper session management.

3. Input Validation: Ensure that the website properly validates all user inputs to prevent injection attacks, such as SQL injection and cross-site scripting (XSS).

4. Sensitive Data Handling: Verify that sensitive data, such as passwords and personal information, is properly encrypted both at rest and in transit. Also, ensure that data is only accessible to authorized users.

5. Security Headers: Check if the website implements security headers, such as Content Security Policy (CSP), X-Frame-Options, and X-XSS-Protection, to protect against various types of attacks.

6. Software and Patch Management : Ensure that the website is running on updated software and that security patches are applied promptly to protect against known vulnerabilities.

7. Secure File Uploads : If the website allows file uploads, ensure that proper validation and security measures are in place to prevent malicious files from being uploaded.

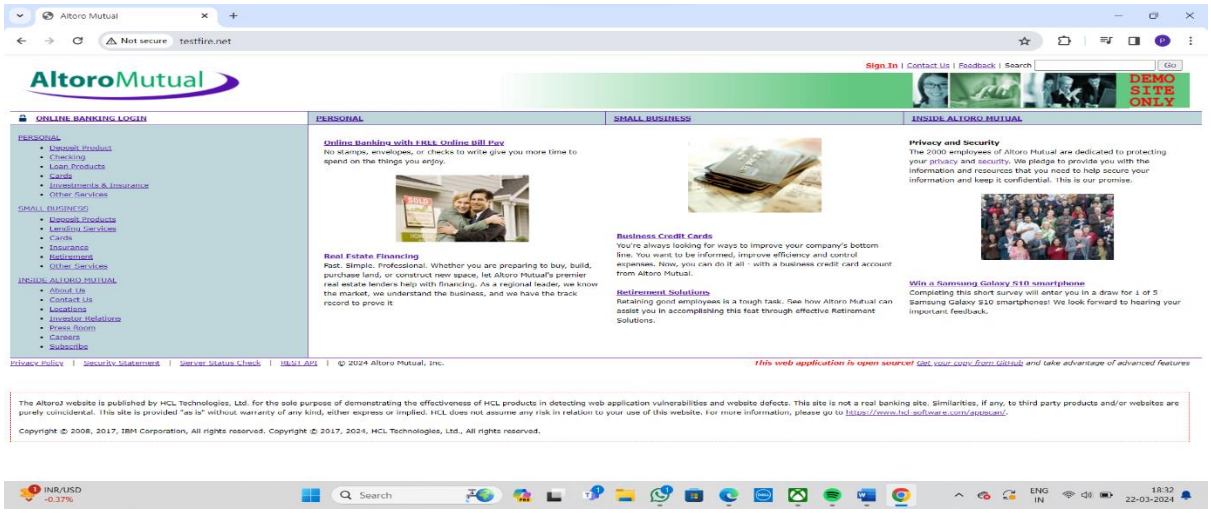
8. Error Handling: Check if the website handles errors gracefully without exposing sensitive information or internal details that could be exploited by attackers.

9. Secure APIs: If the website uses APIs, ensure that they are properly secured and that access is restricted to authorized users only.

10. Regular Security Audits and Testing: Conduct regular security audits and penetration testing to identify and address vulnerabilities in the website.

It's important to note that cybersecurity is an ongoing process, and websites should be regularly reviewed and updated to mitigate new threats and vulnerabilities.

➢ ALTORO MUTUAL WEBSITE INTERACE :

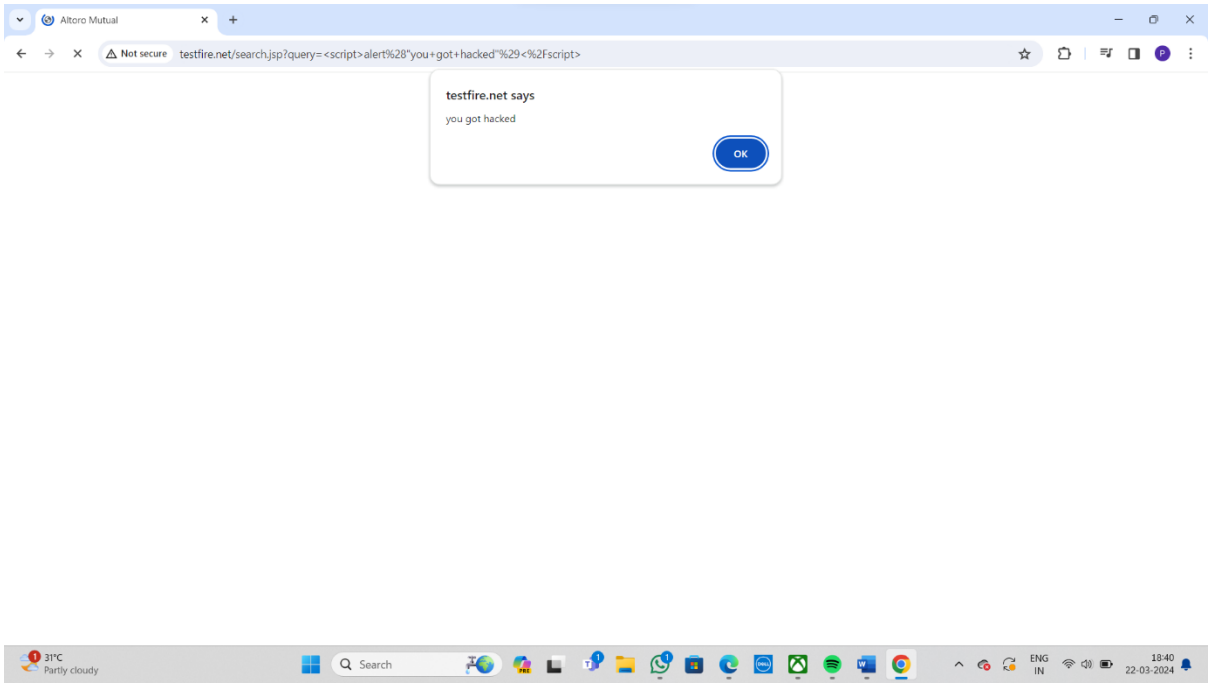


Identifying vulnerabilities such as Cross site scripting (XSS),SQL injection, insecure authentication mechanisms, insecure direct object references, etc., based on the OWASP Top 10 list within Altro Mutual's website structure:

➢ ***Cross site scripting (XSS) :***

Cross-Site Scripting (XSS) is a vulnerability where attackers inject malicious scripts into web pages viewed by other users. This allows them to steal information, manipulate content, or perform unauthorized actions on behalf of the user, compromising web application security.

We enter a script into the search bar and press enter then you got popup like this <SCRIPT>ALERT(YOU GOT HACKED)</SCRIPT>

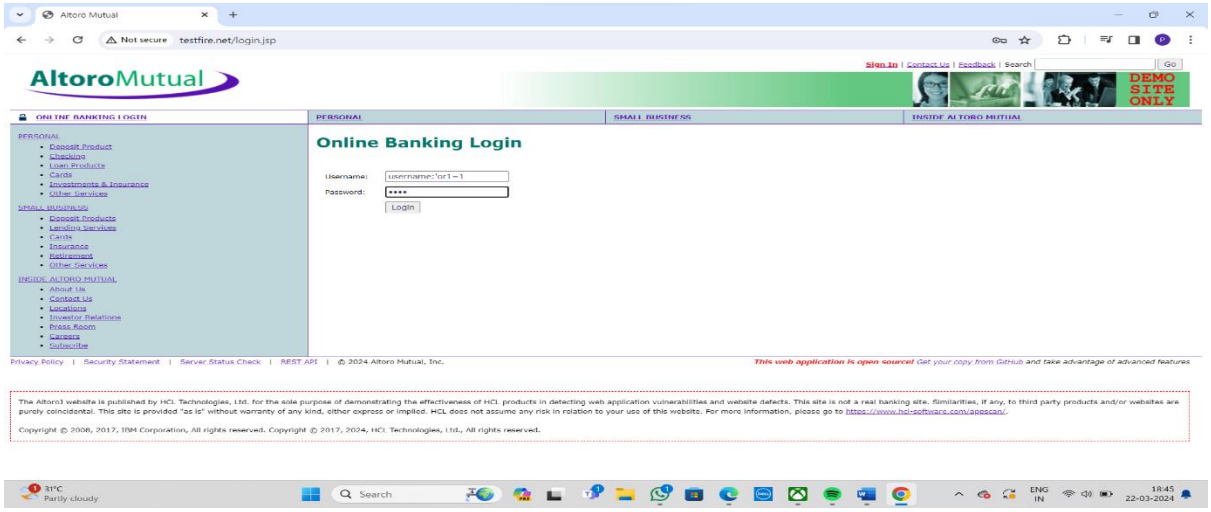


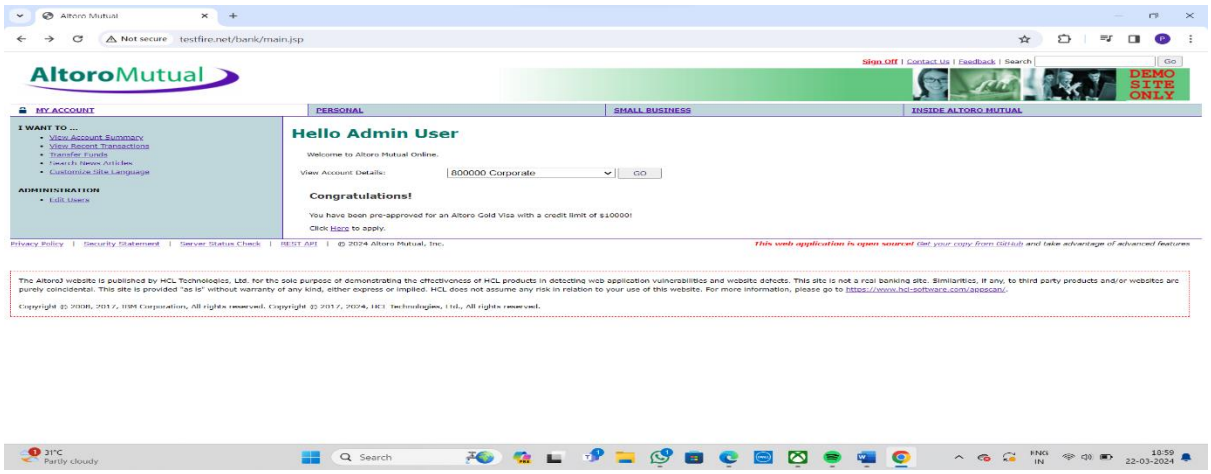
➢ ***Insecure Authentication Mechanisms:***

1. Insecure authentication mechanisms can include weak password policies, lack of multifactor authentication (MFA), improper session management, or storage of passwords in plaintext or weakly hashed formats.
2. Potential Impact: Attackers can exploit weak authentication mechanisms to gain unauthorized access to user accounts, leading to data breaches, identity theft, and unauthorized transactions.
3. Example: Altro Mutual's login process may lack MFA or enforce weak password policies, making it susceptible to brute force attacks or credential stuffing.

➢ WHEN WE ENTER CREDENTIAL DATA LIKE “USERNAME; ’ OR 1=1” & PASSWORD “1234” ϖ THEN PRESS ENTER AND YOU SIGN INTO ACCOUNT

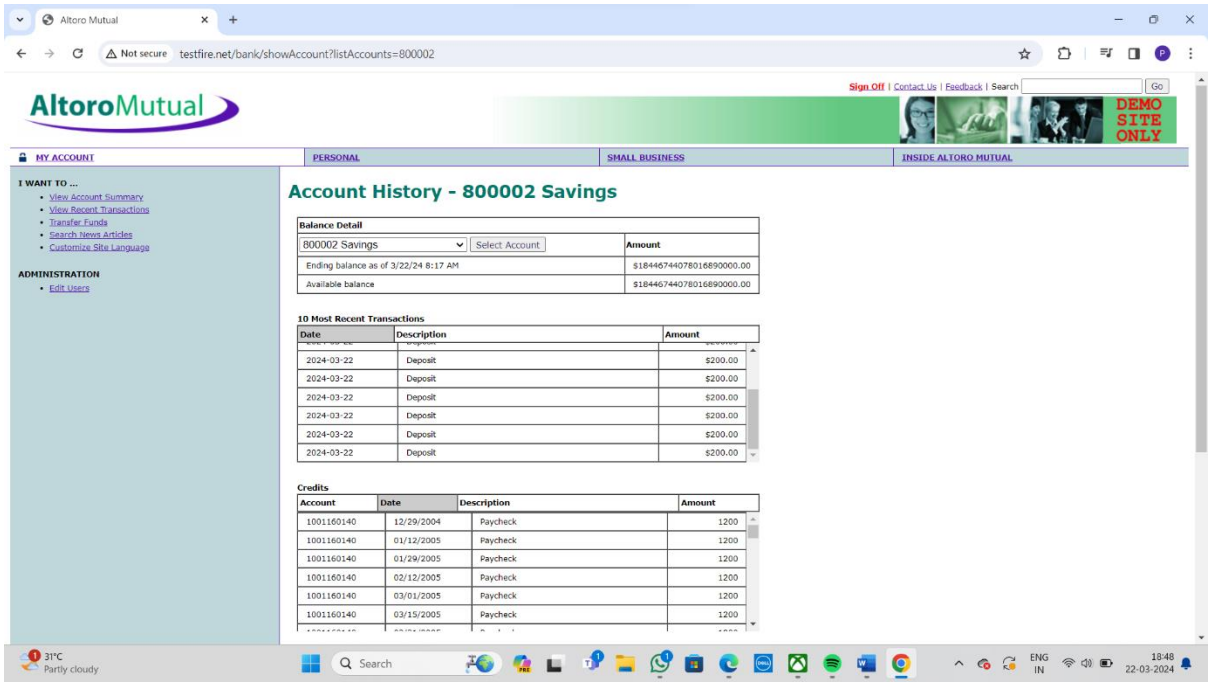
➢ WHEN SOME USER CAN SET MFA(MULTIFACTOR AUTHENTICATION) TO THEIR ACCOUT

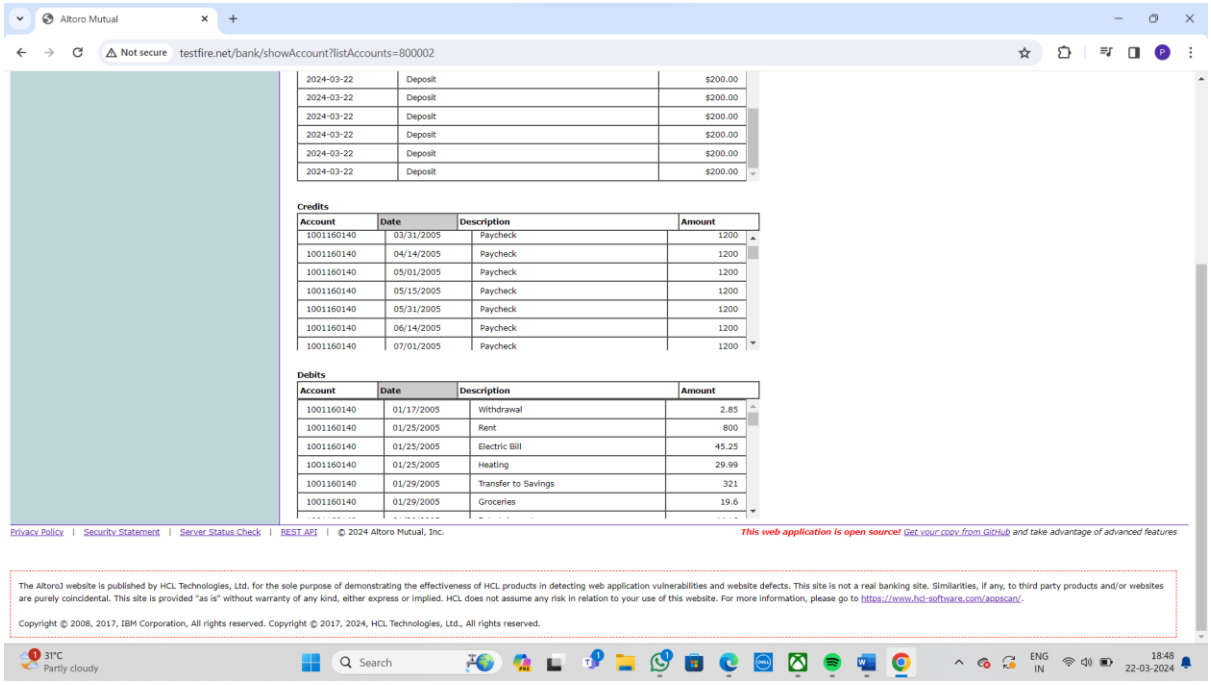




➢ SQL injection :

- SQL injection vulnerabilities may exist in the website's input forms, search functionalities, or any other interaction that involves user supplied data being passed to a database query without proper validation or sanitization.
- *Potential Impact*: Attackers can manipulate SQL queries to gain unauthorized access to the database, extract sensitive information, modify or delete data, and even execute arbitrary commands





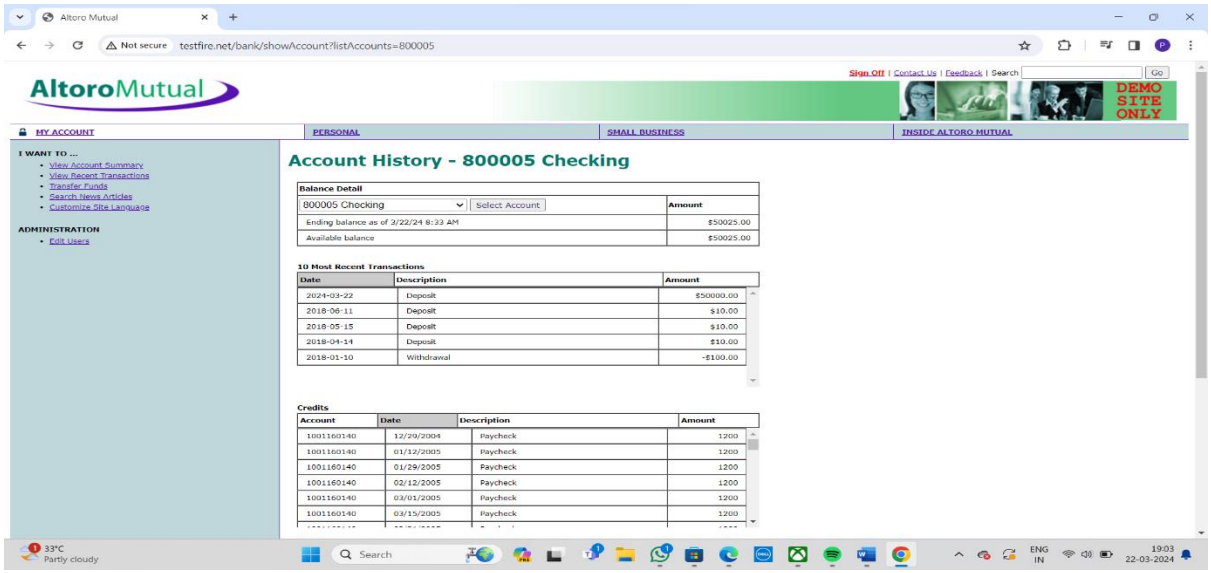
➢ **Insecure Direct Object References (IDOR):**

• IDOR vulnerabilities occur when an application exposes internal implementation details, such as database keys or file paths, directly to users without proper access controls.

• Potential Impact: Attackers can manipulate object references to access unauthorized data or perform actions beyond their privileges, leading to data exposure or modification.

• Example: A URL parameter that directly references a user's account ID might not be adequately protected, allowing attackers to change the parameter to access other users' data.

WHEN WE CHANGE PARAMETER SEARCH BAR “800005” TO “800006” IF THE INPUT IS VALID THEN ACCESS TO THE OTER USER DATA



Sensitive Data Exposure:

➢ Data exposure vulnerabilities may occur when sensitive information, such as user credentials, financial data, or personal details, is not adequately protected in transit or at rest

Vulnerability Identification Report:

➢ Vulnerability Identification Report for Altro Mutual's Website

❖ Website Structure and Functionality Overview:

(1) Homepage

(2) Account Management

(3) Transaction Pages

(4) Support Pages

(5) Login Page

(6) Search Functionality

❖ Identified Vulnerabilities :

- SQL Injection:

Explanation: Input fields such as search forms or login pages may be vulnerable to SQL injection if user input is not properly validated or sanitized before being included in database queries.

Exploitation: Attackers can inject malicious SQL code into input fields to manipulate database queries, potentially gaining unauthorized access to sensitive data or executing arbitrary commands.

Impact: Data breaches, unauthorized access to customer accounts, manipulation of financial records, and reputational damage to Altro Mutual.

- Cross Site Scripting (XSS):

Explanation: User generated content areas such as comment sections or feedback forms may be vulnerable to XSS attacks if input is not properly encoded before being displayed to other users.

Exploitation: Attackers can inject malicious scripts into web pages, leading to session hijacking, cookie theft, or redirection to malicious sites.

Impact: Compromised user accounts, defacement of the website, distribution of malware to users, and loss of customer trust.

- Insecure Authentication Mechanisms:

Explanation: Weak password policies, lack of multifactor authentication (MFA), or improper session management may expose Altro Mutual to unauthorized access to user accounts.

Exploitation: Attackers can exploit weak authentication mechanisms to gain unauthorized access to user accounts, leading to data breaches or unauthorized transactions.

Impact: Compromised user accounts, financial loss due to fraudulent transactions, reputational damage to Altro Mutual.

- Insecure Direct Object References (IDOR):

Explanation: Exposing internal implementation details such as database keys or file paths directly to users without proper access controls can lead to IDOR vulnerabilities.

Exploitation: Attackers can manipulate object references to access unauthorized data or perform actions beyond their privileges.

Impact: Unauthorized access to sensitive data, manipulation of account information, and loss of data integrity.

❖ Recommendations for Mitigation:

SQL Injection:

- Implement input validation and parameterized queries to prevent SQL injection attacks.
- Use prepared statements or stored procedures to sanitize user input before executing database queries.

Cross Site Scripting (XSS):

- Validate and sanitize user input to prevent XSS attacks.
- Implement output encoding to ensure that user generated content is properly escaped before being displayed.

Insecure Authentication Mechanisms:

- Enforce strong password policies, including requirements for length, complexity, and expiration.
- Implement multifactor authentication (MFA) to add an extra layer of security.
- Use secure session management practices, such as session timeouts and secure cookies

Insecure Direct Object References (IDOR):

- Implement proper access controls to restrict access to sensitive resources.
- Avoid exposing internal identifiers such as database keys or file paths directly to users.
- Use indirect references or access controls to protect sensitive data.

❖ Mitigation Measures:

a. SQL Injection:
>>Implement strict input validation and parameterized queries to prevent SQL injection attacks. o Conduct regular security audits and code reviews to identify and remediate any potential vulnerabilities.

b. Cross Site Scripting (XSS):
>> Implement output encoding to sanitize user input and prevent XSS attacks.
>>Utilize Content Security Policy (CSP) headers to restrict the execution of inline scripts and mitigate the impact of XSS attacks.

c. Insecure Authentication Mechanisms:
>> Enforce strong password policies, including minimum length, complexity, and expiration requirements.
>> Implement multifactor authentication (MFA) to add an extra layer of security and protect against credential based attacks.
>> Use secure session management practices, such as session timeouts and secure cookies, to prevent unauthorized access.

d. Insecure Direct Object References (IDOR):
>> Implement proper access controls to restrict access to sensitive resources.