ASSIGNMENT4

1. OSWAP TOP 10 VULNERABILITIES OVERVEIW:

The OWASP (Open Web Application Security Project) Top 10 vulnerabilities list is a widely recognized document that outlines the most critical security risks to web applications. Here's an overview of the OWASP Top 10 vulnerabilities and their potential impact on web application security:

- **Injection:** Injection flaws occur when untrusted data is sent to an interpreter as part of a command or query. This can lead to unauthorized access, data loss, or data manipulation. Common types include SQL injection, LDAP injection, and OS command injection.

- **Broken Authentication**: This vulnerability arises when authentication mechanisms are not implemented correctly, allowing attackers to compromise user accounts, access sensitive data, or impersonate users. Examples include weak password policies, session fixation, and insecure password storage.

- **Sensitive Data Exposure**: Failure to properly protect sensitive data such as financial information, personal details, or authentication credentials can result in data breaches, identity theft, and financial losses. This vulnerability often stems from inadequate encryption, weak authentication, or insufficient access controls.

- **XML External Entities (XXE):** XXE vulnerabilities occur when XML input processing is not properly configured, allowing attackers to exploit XML parsers to access sensitive files, execute arbitrary code, or perform denialofservice attacks.

- **Broken Access Control**: Insecure access controls enable unauthorized users to view or modify restricted resources, bypass authentication mechanisms, or escalate privileges. This can lead to data breaches, unauthorized data manipulation, and exposure of sensitive information.

- **Security Misconfiguration**: This vulnerability arises from improper configuration of security controls, server settings, or application components, leaving systems vulnerable to various attacks such as unauthorized access, data breaches, or denialofservice attacks.

- **CrossSite Scripting (XSS):** XSS vulnerabilities allow attackers to inject malicious scripts into web pages viewed by other users. This can lead to

session hijacking, data theft, defacement of websites, or distribution of malware.

- **Insecure Deserialization**: Insecure deserialization vulnerabilities occur when untrusted data is deserialized by an application, leading to remote code execution, data tampering, or denialofservice attacks.

- **Using Components with Known Vulnerabilities**: Failure to update or patch thirdparty components, libraries, or frameworks can expose web applications to known vulnerabilities, allowing attackers to exploit these weaknesses to gain unauthorized access, execute arbitrary code, or steal sensitive data.

- **Insufficient Logging & Monitoring**: Inadequate logging and monitoring make it difficult to detect and respond to security incidents effectively. Without comprehensive logs and monitoring mechanisms, attackers can carry out malicious activities undetected, resulting in prolonged breaches, data exfiltration, or system compromise.

Addressing these vulnerabilities is crucial for maintaining the security of web applications. Failure to do so can have severe consequences, including:

- ✓ Data breaches leading to financial losses, regulatory fines, and reputational damage.
- ✓ Compromise of user accounts leading to identity theft, fraud, or unauthorized access to sensitive information.
- ✓ Disruption of services due to denialofservice attacks or unauthorized data manipulation.
- ✓ Legal and regulatory consequences for noncompliance with data protection laws and industry regulations.
- ✓ Damage to brand reputation and loss of customer trust.

The potential impact of these vulnerabilities on web application security can be severe, including data breaches, financial loss, reputational damage, legal consequences, and disruption of services. Addressing these vulnerabilities is essential to prevent exploitation by attackers and safeguard the confidentiality, integrity, and availability of web applications. This involves implementing secure coding practices, adopting security controls and defenses, conducting regular security assessments and audits, and staying informed about emerging threats and best practices in web application security. By proactively addressing these vulnerabilities, organizations can mitigate risks, protect sensitive data, and maintain the trust and confidence of users and stakeholders.

- ➢ These vulnerabilities is crucial for several reasons:

- **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.

2. **Altro Mutual Website Analysis:**

   - **Website Structure Overview**:

   Identify the main sections of the website, such as homepage, login page, account management, transaction pages, support pages, etc.

   Analyze the underlying architecture, including the use of frameworks, libraries, and serverside technologies.

   - **Functionality Assessment**:

   Review the functionality of each component, such as user authentication, authorization, session management, data input forms, database interactions, file uploads, etc.

   Evaluate the implementation of security features, such as encryption, secure communication protocols (HTTPS), and input validation.

   - **Potential Areas of Vulnerability**:

   Conduct a systematic review of the OWASP Top 10 vulnerabilities and assess how each applies to Altro Mutual's website.

   Identify specific areas where these vulnerabilities may exist within the website's structure and functionality.

   - **Detailed Vulnerability Analysis**:

   For each potential vulnerability, provide a detailed explanation of how it could manifest within Altro Mutual's website.

   Analyze the potential impact of exploitation, considering the sensitivity of data, the potential for financial loss, and the reputational damage to Altro Mutual.

   - **Recommendations for Mitigation**:

   Propose specific mitigation strategies for addressing each identified vulnerability, tailored to Altro Mutual's website and technology stack.

   Prioritize recommendations based on the severity of vulnerabilities and the resources available for mitigation efforts.

- **Best Practices Implementation**:

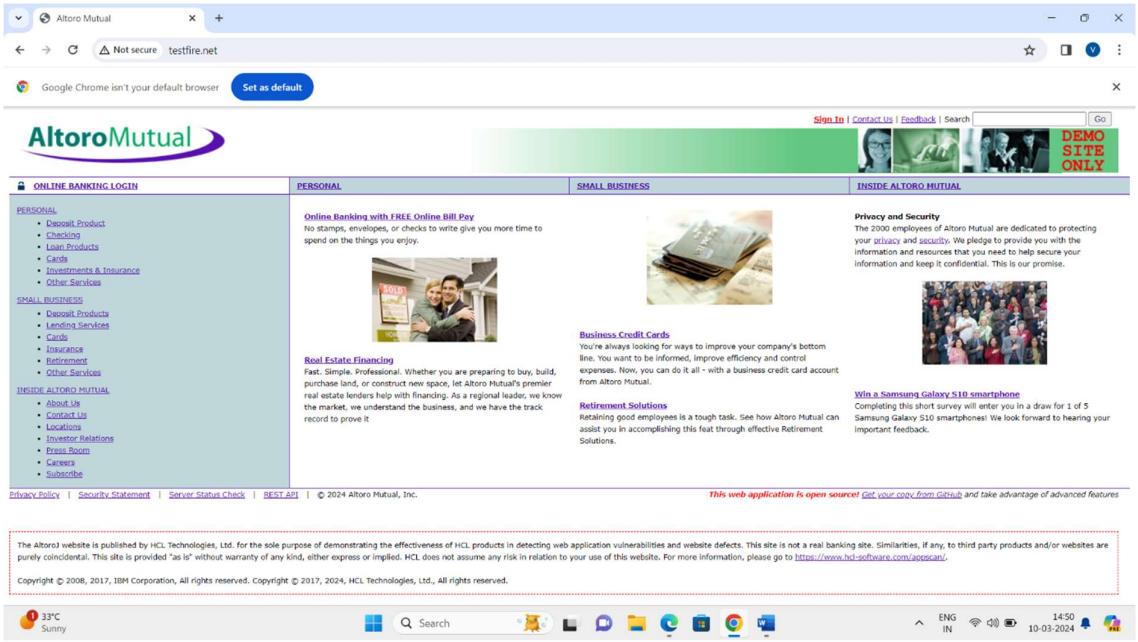
Recommend the adoption of best practices in web application security, such as secure coding guidelines, regular security training for developers, and the use of automated security testing tools.

- **Continuous Monitoring and Improvement**:

Emphasize the importance of ongoing monitoring and improvement of Altro Mutual's website security posture.

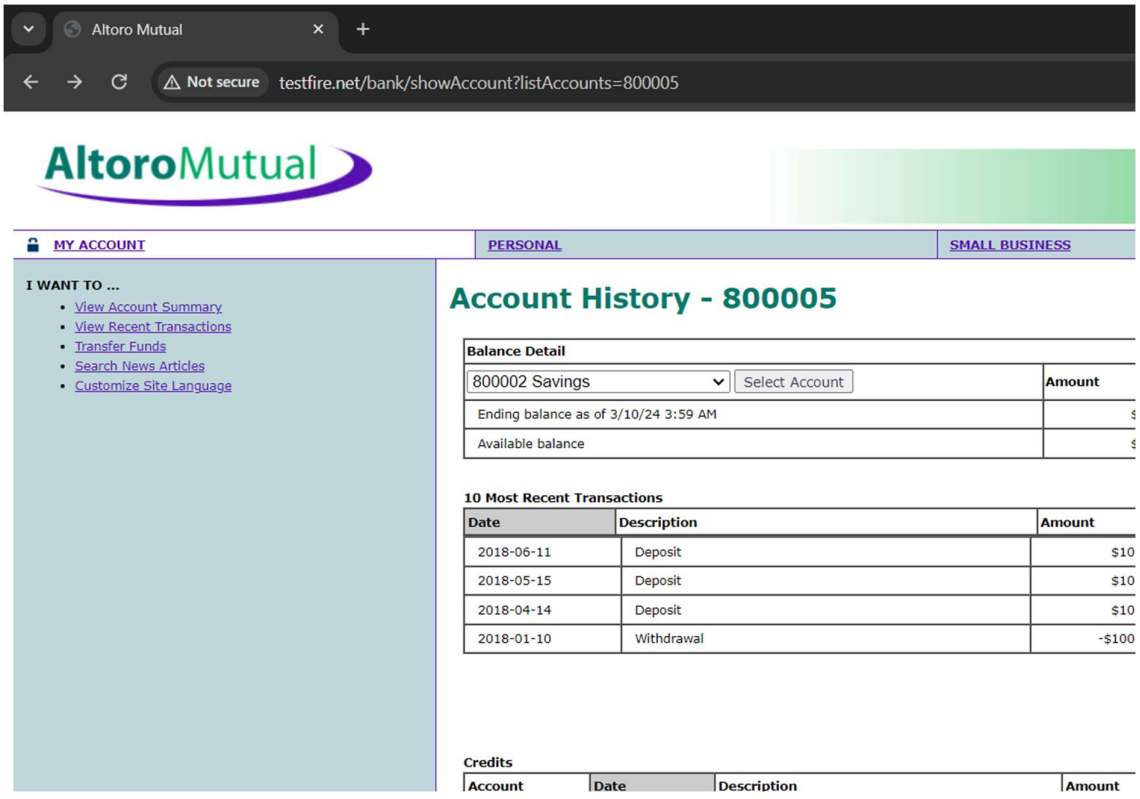
Recommend the establishment of incident response procedures and regular security assessments to identify and address emerging threats.

- ✓ ALTORO MUTUAL WEBSITE INTERACE



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

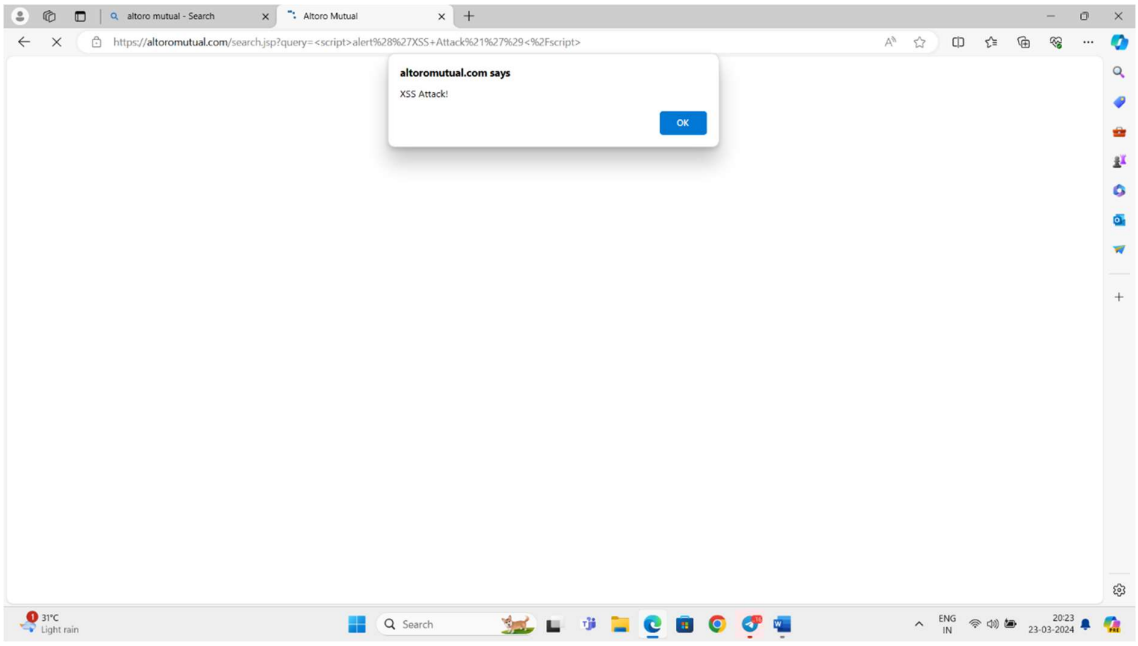
- **SQL Injection:**
  - SQL injection vulnerabilities may exist in the website's input forms, search functionalities, or any other interaction that involves usersupplied data being passed to a database query without proper validation or sanitization.
  - Potential Impact: Attackers can manipulate SQL queries to gain unauthorized access to the database, extract sensitive information, modify or delete data, and even execute arbitrary commands.
  - Example: The login form or search feature may be susceptible to SQL injection if input validation and parameterized queries are not implemente



- **CrossSite Scripting (XSS):**
  - XSS vulnerabilities may arise in the website's input fields, comment sections, or any other usergenerated content areas where untrusted data is displayed without proper encoding or validation.
  - Potential Impact: Attackers can inject malicious scripts into web pages viewed by other users, leading to session hijacking, cookie theft, defacement of the website, or redirection to malicious sites.
  - Example: A feedback form that doesn't properly sanitize user input may be vulnerable to XSS attacks.

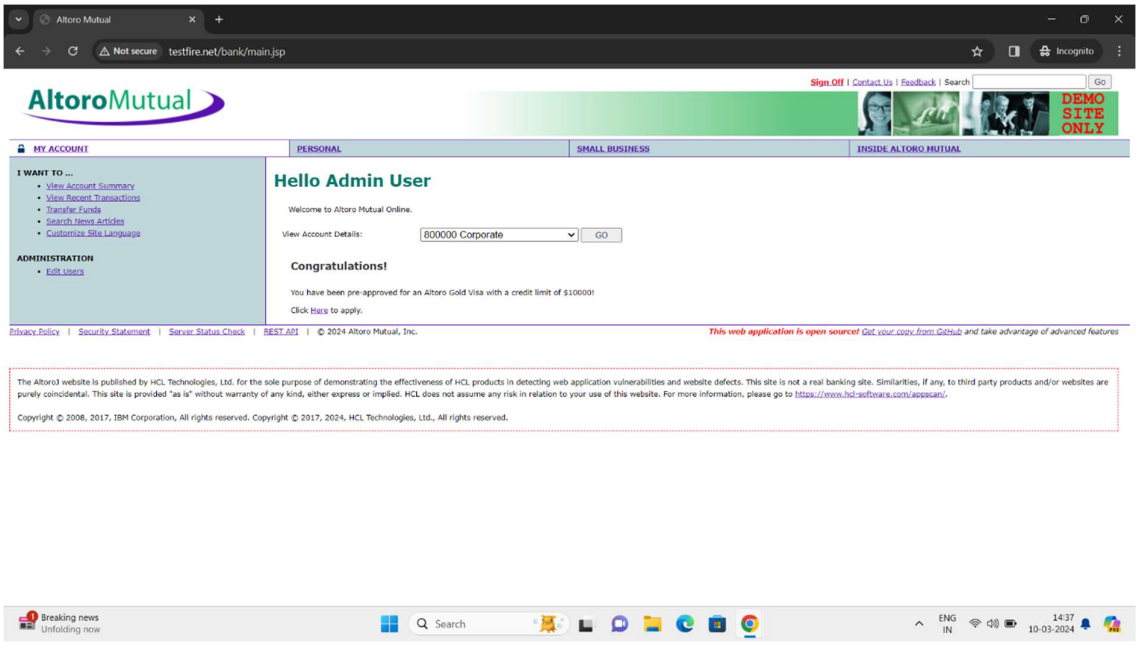
We enter a script into the search bar and press enter then you got popup like this

<SCRIPT>ALERT(YOU GOT HACKED)</SCRIPT>



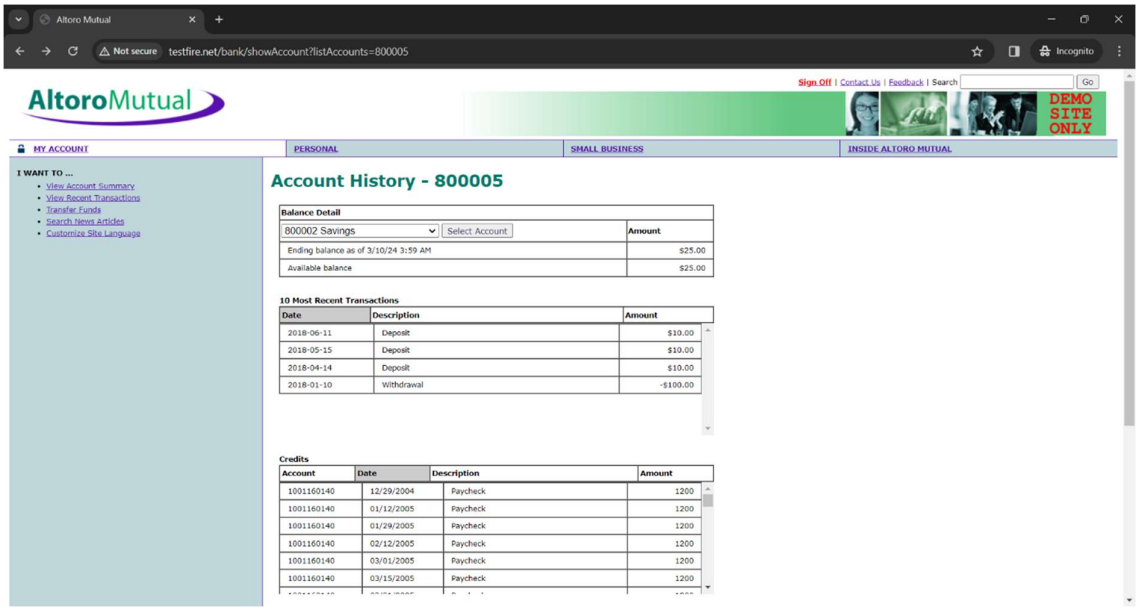
- **Insecure Authentication Mechanisms**:
  - Insecure authentication mechanisms can include weak password policies, lack of multifactor authentication (MFA), improper session management, or storage of passwords in plaintext or weakly hashed formats.
  - Potential Impact: Attackers can exploit weak authentication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

- WHEN WE ENTER CREDENTIAL DATA LIKE “USERNAME; ’ OR 1=1” & PASSWORD “1234”
- THEN PRESS ENTER AND YOU SIGN INTO ACCOUNT
- WHEN SOME USER CAN SET MFA(MULTIFACTOR AUTHENTICATION) TO THEIR ACCOUT



- **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.
  - Potential Impact: Attackers can intercept sensitive data, leading to identity theft, financial loss, or unauthorized access to accounts.
  - Example: Altro Mutual's website may transmit sensitive data over HTTP instead of HTTPS, making it vulnerable to interception.

3. **Vulnerability Identification Report:**
   - Vulnerability Identification Report for Altro Mutual's Website

1. Website Structure and Functionality Overview:
   (1) Homepage
   (2) Account Management
   (3) Transaction Pages
   (4) Support Pages
   (5) Login Page
   (6) Search Functionality

2. **Identified Vulnerabilities:**

a. 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.

b. CrossSite Scripting (XSS):

- Explanation: Usergenerated 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.

c. 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.

d. 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.

3. **Recommendations for Mitigation:**

a. 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.

b. CrossSite Scripting (XSS):

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

c. 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.

d. 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.

4. **Vulnerability Exploitation Demonstration**

- **SQL Injection Exploitation**:

✓ Scenario: Suppose Altro Mutual's website has a search feature allowing users to search for transactions by entering a transaction ID.

✓ Exploitation: An attacker can manipulate the input field to inject malicious SQL code, such as appending ' OR 1=1 to the end of the transaction ID. This would cause the query to return all transactions instead of just the specified one.

✓ ProofofConcept (PoC):
✓
➢ **Input: ' OR 1=1**
➢ **SQL Query: SELECT * FROM transactions WHERE transaction_id = '' OR 1=1'**
✓ Result: All transactions are returned.
✓

✓ Impact: The attacker gains unauthorized access to sensitive transaction data, potentially including confidential customer information.

- **CrossSite Scripting (XSS) Exploitation:**

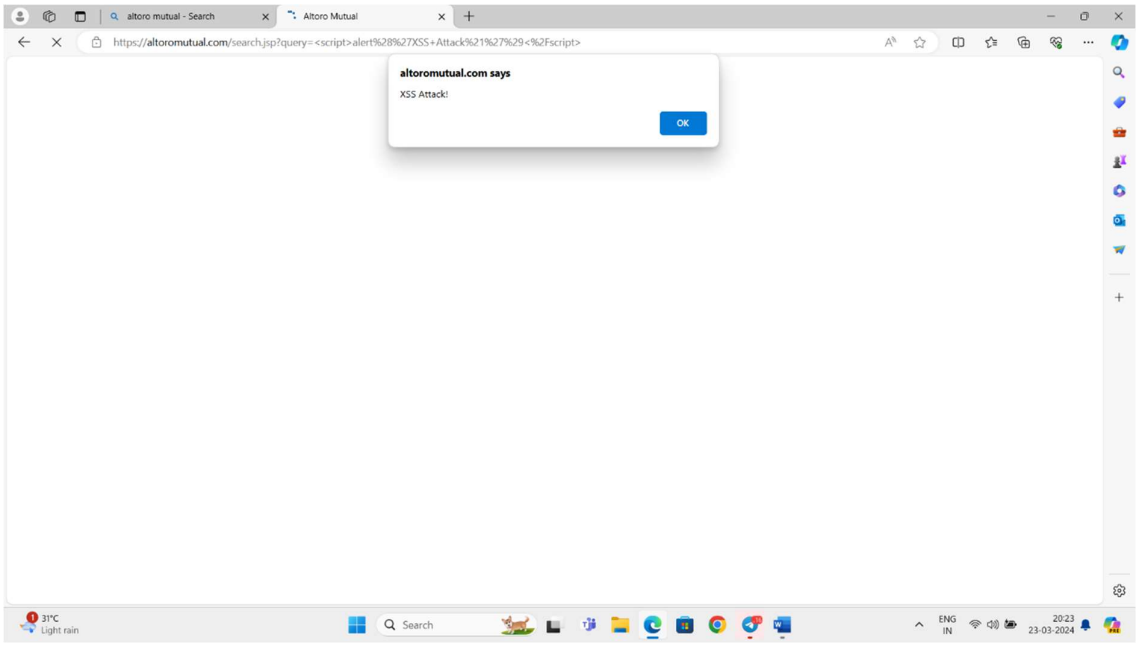
✓ Scenario: Altro Mutual's website includes a feedback form where users can submit comments.

✓ Exploitation: An attacker submits a comment containing malicious JavaScript code. When another user views the comment, the code executes in their browser.

✓ ProofofConcept (PoC):

➢ **Malicious Comment: <script>alert('XSS Attack!')</script>**

✓ Impact: When a user views the comment, the JavaScript code executes, displaying an alert box with the message "XSS Attack!" This demonstrates how an attacker can execute arbitrary scripts in users' browsers.



- **Insecure Authentication Mechanisms Exploitation:**

✓ Scenario: Altro Mutual's website allows users to log in using a username and password.

✓ Exploitation: An attacker attempts to brute force user passwords using automated tools.

✓ ProofofConcept (PoC):

➢ Attack: Automated tool attempts to guess user passwords using a list of commonly used passwords or dictionary words.

➢ Impact: If weak passwords are used, the attacker may successfully gain unauthorized access to user accounts, potentially compromising sensitive financial information.

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

✓ Scenario: Altro Mutual's website allows users to view their account details by specifying their account ID in the URL.

✓ Exploitation: An attacker modifies the account ID in the URL to access another user's account without proper authorization.

✓ ProofofConcept (PoC):

➢ **URL: https://altromutual.com/account?id=12345 (legitimate)**

- **Modified URL: https://altromutual.com/account?id=54321 (attackermodified)**

- ✓ Impact: The attacker gains unauthorized access to another user's account, potentially viewing sensitive financial information or making unauthorized transactions.

5. **Mitigation Strategy Proposal:**

Mitigation Strategy Proposal for Altro Mutual's Website

1. Prioritization of HighRisk Vulnerabilities:

The mitigation strategy will prioritize addressing the highrisk vulnerabilities identified in the vulnerability identification report. These vulnerabilities include SQL injection, CrossSite Scripting (XSS), Insecure Authentication Mechanisms, and Insecure Direct Object References (IDOR). Addressing these vulnerabilities will significantly reduce the risk of data breaches, unauthorized access, and other security incidents.

2. Mitigation Recommendations:

a. 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.

b. CrossSite Scripting (XSS):
Validate and sanitize user input to prevent XSS attacks.
Implement output encoding to ensure that usergenerated content is properly escaped before being displayed.

c. 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.

d. 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.

3. Implementation Timeline:

The mitigation strategy will be implemented in phases, with highrisk vulnerabilities addressed first. The timeline for implementation will be as follows:

Phase 1 (Immediate): Address SQL injection and XSS vulnerabilities by implementing input validation, parameterized queries, and output encoding.

Phase 2 (12 weeks): Enhance authentication mechanisms by enforcing strong password policies and implementing MFA.

Phase 3 (24 weeks): Implement access controls and secure direct object references to mitigate IDOR vulnerabilities.

4. Security Training and Awareness:

Provide security training and awareness sessions for developers and staff members to ensure understanding of best practices in web application security. Regular updates and reminders about security policies and procedures will be communicated to all relevant personnel.

5. Ongoing Monitoring and Review:

Implement continuous monitoring and regular security reviews to detect and address any new vulnerabilities or emerging threats. This includes conducting periodic penetration testing, code reviews, and security assessments to maintain the security posture of Altro Mutual's website.