SECURITY TESTING - ASSESSMENT 1

Guided By

Mr SUMIT THIGALE

EFFIGO GLOBAL

Submitted By

SHAHIR BILAGI

PRODUCT ENGINEERING INTERN

EFFIGO GLOBAL

Security Testing

Security testing identifies vulnerabilities, threats, and weaknesses in a system to ensure data integrity, confidentiality, and availability. It helps in preventing unauthorised access, cyber-attacks, and data breaches.

Some of the Basic Features

1. **Vulnerability Assessment:** Identifies security loopholes in applications and networks.
2. **Penetration Testing (Ethical Hacking):** Simulates real-world cyber-attacks to test security defences.
3. **Authentication & Authorization Testing:** Ensures only authorized users can access the system.
4. **SQL Injection and XSS Testing:** Prevents SQL Injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF).
5. **API Security Testing:** Ensures APIs do not expose sensitive data or allow unauthorized actions.

Importance of Security Testing

1. Prevent Cyber Attacks.
2. Ensure Data Protection.
3. Reduce Costs and save time.
4. Maintains Customer Trust & Business Reputation.

Vulnerability Testing

Vulnerability testing is a scanning process that identifies security weaknesses in a system, application, or network. It automates the detection of known vulnerabilities but does not exploit them. Some of the key features are:

1. Uses automated tools like Nessus, OpenVAS, and Qualys.
2. Generates reports listing vulnerabilities along with risk levels.

Penetration Testing

Penetration testing is an offensive security approach where ethical hackers simulate real-world attacks to exploit vulnerabilities. It is a manual process combined with automation to assess the actual security risk. Some key Features are:

1. Includes black-box (external), white-box (internal), and gray-box testing.
2. Uses tools like Metasploit, Burp Suite, Kali Linux, and Nmap.

The key Difference Between Vulnerability and Penetration Testing is:

Feature	Vulnerability Testing	Penetration Testing
Purpose	Detect vulnerabilities	Exploit vulnerabilities
Method	Automated scanning	Manual + automated testing
Risk Assessment	Identifies security risks	Determines actual impact
Tools Used	Nessus, OpenVAS, Qualys	Metasploit, Burp Suite, Nmap
Depth of Testing	Surface-level detection	Deep security analysis

Feature	Vulnerability Testing	Penetration Testing
Exploitation	No	Yes
Use Case	Routine security assessment	Real-world attack simulation

Comparison: Manual Security Testing vs. Security Testing Tools:

Feature	Manual Security Testing	Security Testing Tools
Approach	Performed by security experts	Automated scanning & reporting
Efficiency	Time-consuming, requires expertise	Fast & efficient
Customization	Can adapt to complex attack scenarios	Limited to predefined vulnerabilities
Threat Detection	Detects unknown and zero-day attacks	Detects known vulnerabilities
Exploitation	Can simulate real attacks	Mostly detects, does not exploit
Examples	Manual SQL Injection, XSS, CSRF testing	Nessus, Burp Suite, OWASP ZAP

CIA Triad: The Foundation of Information Security

1. **Confidentially (C)**
   - Ensures that only authorized users can access sensitive data.
   - Prevents data leaks, unauthorized access, and identity theft.
   - Implemented through encryption, authentication, and access control.
   - Example: Implementing Multi-Factor Authentication (MFA) for login security.
2. **Integrity (I)**
   - Ensures that data remains unchanged and unaltered during transmission or storage.
   - Prevents modification, corruption, or tampering by attackers.
   - Example: Blockchain technology for tamper-proof data storage.
3. **Availability (A)**
   - Ensures that data and systems are accessible whenever needed.
   - Implemented using redundancy, backups, and failover systems.
   - Example: Cloud-based disaster recovery solutions for business continuity.
   - Example: Load balancing to prevent website downtime.

Threads

A thread is the smallest unit of execution within a process. It is a lightweight sub-process that shares resources (memory, data, files) with other threads within the same process. Threads allow a program to execute multiple tasks simultaneously (concurrent execution).

1. **Intentional Threads:** These are security risks that are intentionally introduced by malicious actors to compromise a system. Attackers deliberately exploit weaknesses to gain unauthorized access, steal data, or disrupt operations.
    - Example: **SQL Injection:** Injecting malicious SQL queries to manipulate databases.
2. **Unintentional Threads:** These vulnerabilities arise unintentionally due to poor coding practices, misconfigurations, or system design flaws. They are not created with malicious intent but can still be exploited.
    - Example: **Poor Access Control:** Users gaining higher privileges due to misconfigured roles.
    - Example: I**nsecure APIs:** Exposing sensitive data due to weak authentication.

Input Validation and Output Encoding

Input Validation: Input validation is the process of checking whether user input meets predefined criteria before being processed by the system.

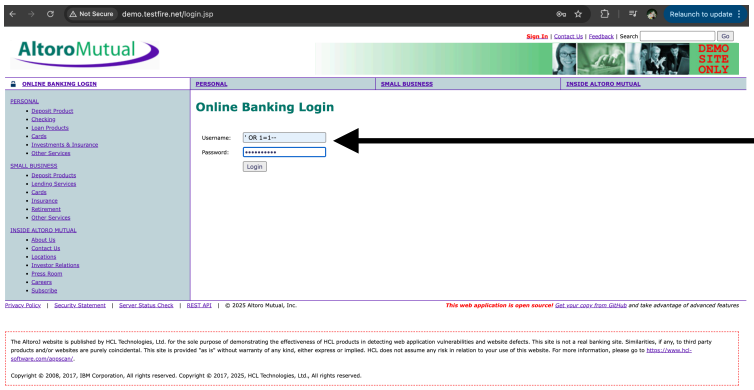
Output Encoding: Output encoding is converting potentially dangerous characters into a safe format before displaying them on a webpage or passing them to another system.

Client Side Validation versus Server Side Validation

Feature	Client-Side Validation	Server-Side Validation
Location	Happens in the user's browser before data is sent to the server.	Happens on the server after data is received from the client.
Technology Used	JavaScript, HTML5 attributes (e.g., required, pattern), frameworks	Backend languages like Python (Flask/Django), Node.js, Java, PHP, etc.
Speed	Faster as validation occurs on the client without a network request.	Slower since it requires communication with the server.
Security	Less secure because it can be bypassed by disabling JavaScript or manipulating requests.	More secure as it ensures all data is validated before processing.
Error Handling	Provides immediate feedback to users without a page refresh.	Ensures invalid or malicious data doesn’t enter the database.
Reliability	Can be bypassed by users.	Cannot be bypassed, making it essential for security.
Use Case	Used for improving user experience, such as formatting checks (e.g., valid email, password strength).	Used to enforce business rules, prevent SQL injection, and ensure data integrity.

Vulnerability Testing Examples

SQL Injections: SQL Injection (SQLi) is a web security vulnerability that allows an attacker to interfere with an application's database queries. It occurs when an attacker manipulates input fields to execute SQL commands, gaining unauthorized access to sensitive data, modifying records, or even compromising the entire database.



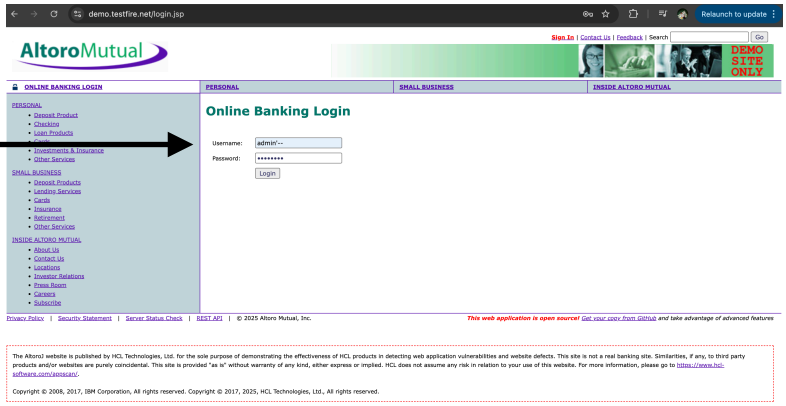
' Closes any existing string in the SQL query.

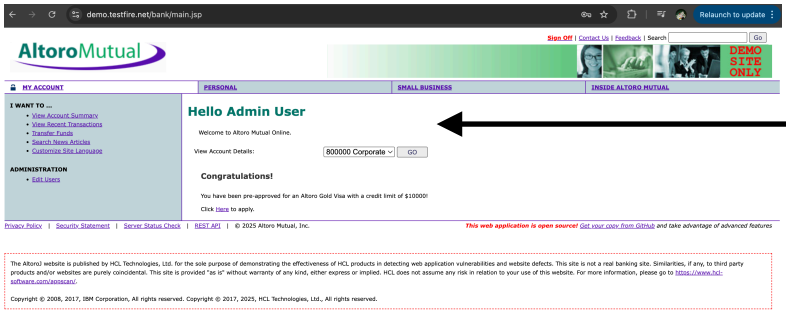
OR 1=1 Always evaluates to TRUE, making the condition always valid.

-- Comments on the rest of the SQL query, preventing syntax errors.

admin' Closes the username field and allows injection.

-- Comments out the password





Login Successful in admin

Impact of SQL Injection:

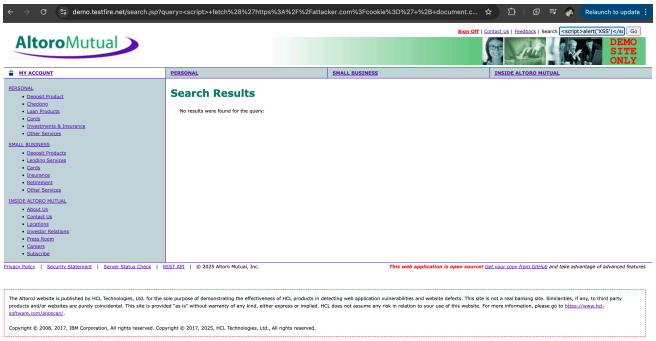
- Unauthorized access to sensitive data.
- Data modification or deletion.
- Potential full database compromise.

Cross-Site Scripting (XSS): It is a security vulnerability that allows attackers to inject malicious scripts (typically JavaScript) into web pages viewed by other users. This can be used to steal sensitive data and deface websites. How XSS works:

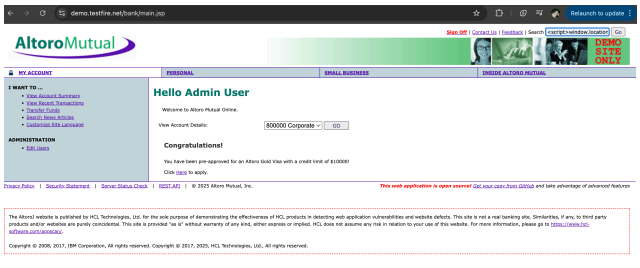
- An attacker injects malicious JavaScript into a website's input field.
- The website unknowingly stores or reflects this script.
- When another user visits the affected page, the script executes in their browser.

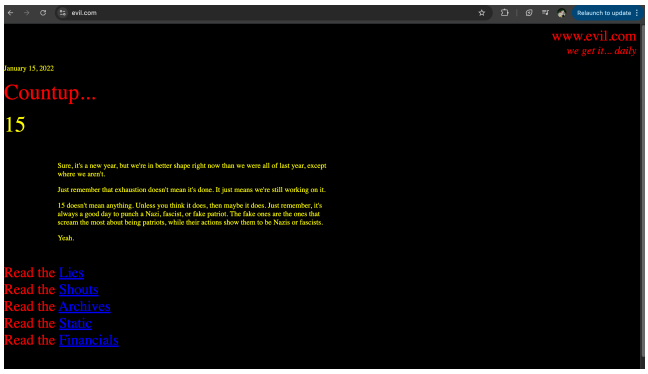
Types of XSS,

1. **Stored XSS:** The script is permanently stored in the database and executed when users load the affected page. Example: Injecting a malicious script in a comment section of a blog.
2. **Reflected XSS:** The malicious script is included in a URL or form submission and immediately executed without being stored. Example: A phishing email with a link containing an injected script.
3. **DOM-Based XSS:** The attack occurs entirely on the client side by modifying the Document Object Model (DOM). Example: A JavaScript function dynamically inserting unvalidated user input into the page.



JavaScript: <script>alert('XSS')</script> (Reflected XSS)

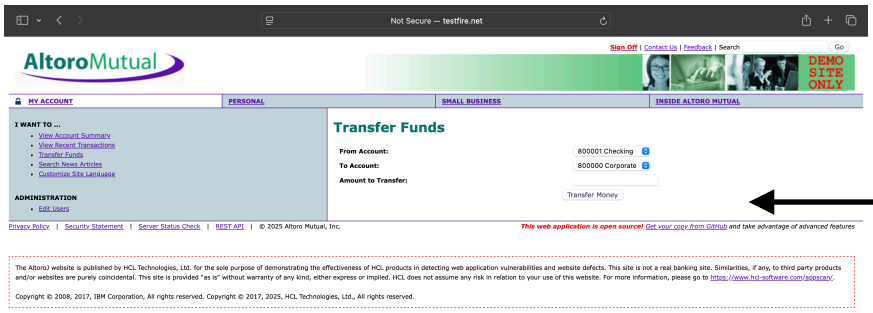




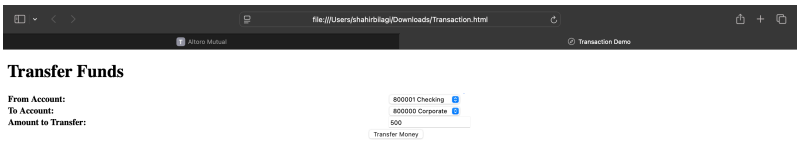
JavaScript: <script>window.location='https://evil.com'</script> (Reflected XSS)

CSRF (Cross-Site Request Forgery): It is a type of web security vulnerability where an attacker tricks a user into making unintended requests to a trusted website where they are authenticated. How CSRF Works

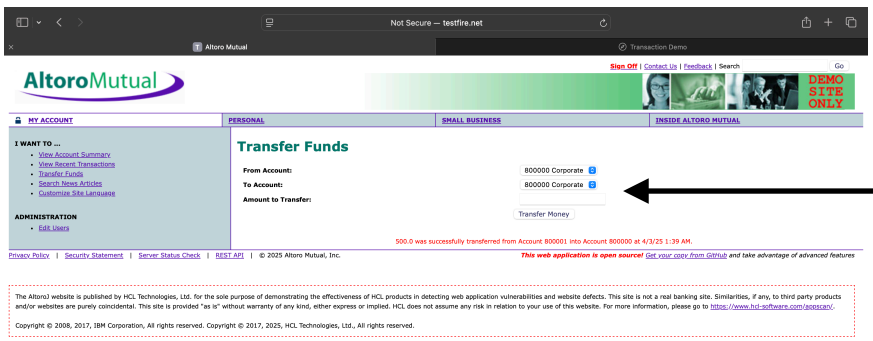
- User Authentication: The user logs into a website, and the site sets an authentication session (e.g., via cookies).
- Attacker's Trick: The attacker crafts a malicious request (e.g., transferring money) and embeds it in a malicious site.
- User Interaction: If the user, while logged in, visits the attacker's page or clicks a link, the request is automatically sent to the target website using the user's session cookies.
- Unintended Action: The target website processes the request as if it came from the authenticated user, leading to unauthorized actions.



User logs into a website, and the site sets an authentication session



Attacker crafts a malicious request



The target website processes the request