**Vulnerability Report**

**Web Application Security Testing**

Website: <http://testphp.vulnweb.com> and <http://testphp.vulnweb.com>

---

**Vulnerability 1: Cross-Site Scripting (XSS)**

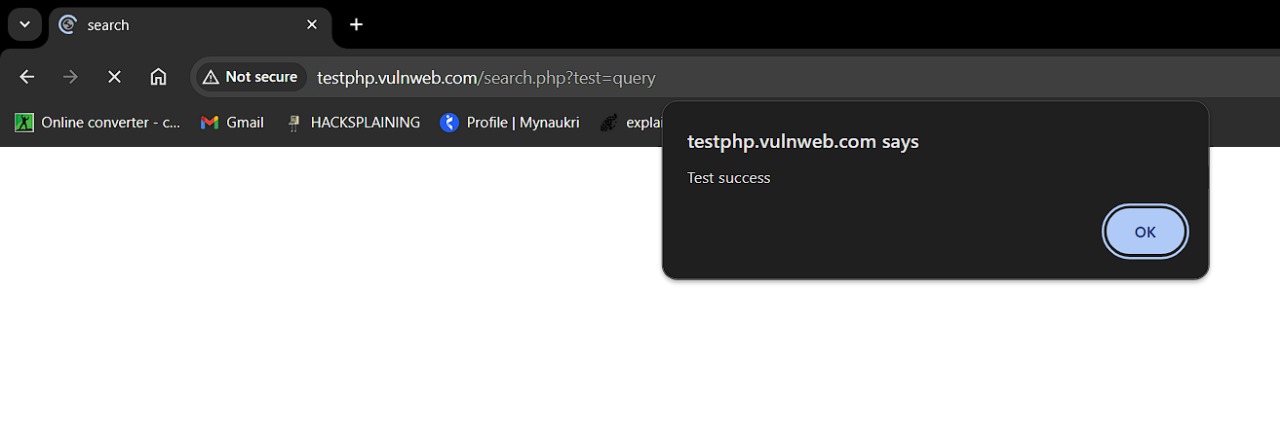
Description: XSS is a common vulnerability where attackers inject malicious code into input fields of a website. Upon successful injection, the website may execute the injected code, leading to various exploits.

Exploit:

- Payload: `<script>alert("Hacked")</script>`

- Upon injection and execution, a pop-up window displaying "Hacked" was observed, indicating the presence of XSS vulnerability.

Screenshot:



**Vulnerability 2: Request Manipulation**

Description: Request manipulation vulnerability involves attackers intercepting and modifying requests sent from the user to the server, potentially altering sensitive data or transactions.

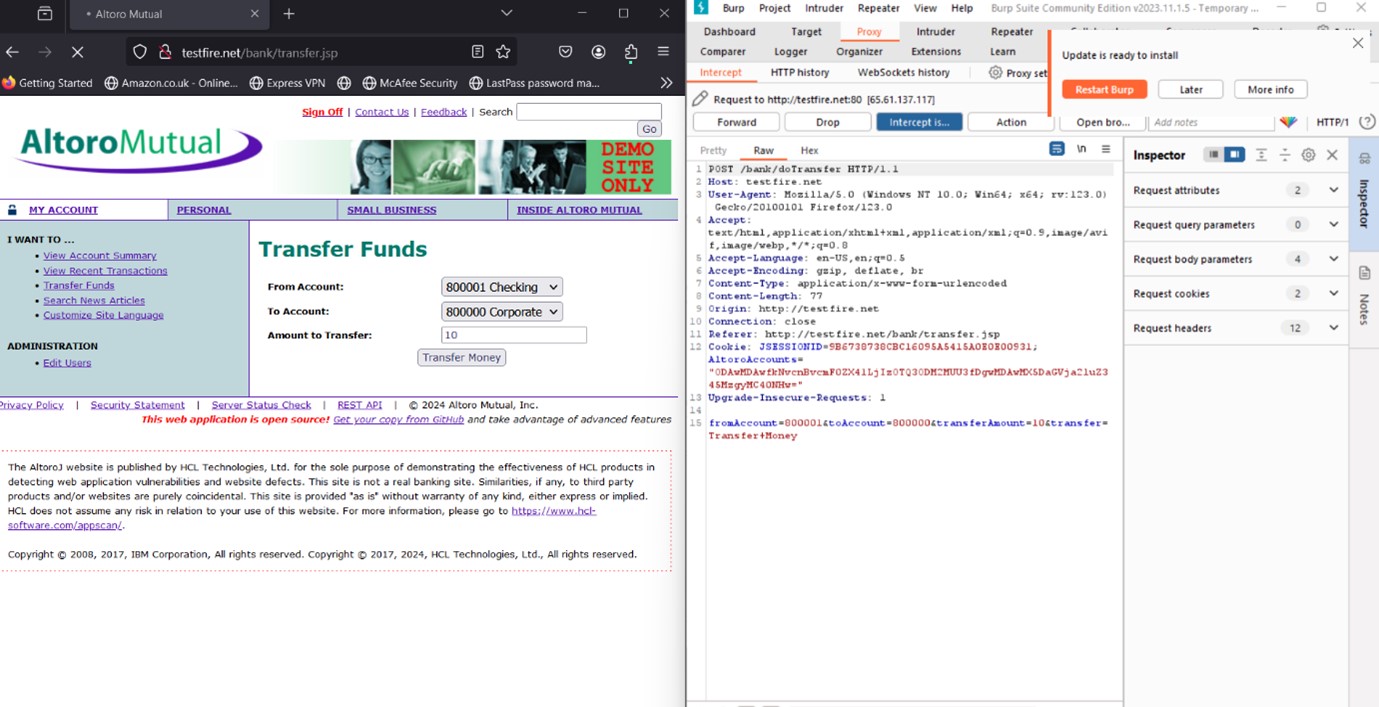
Exploit:

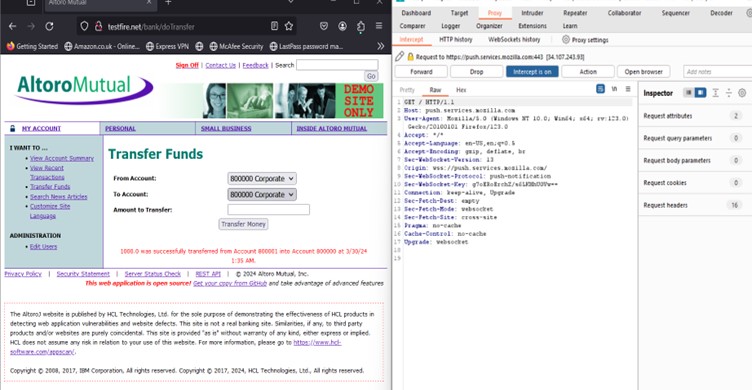
- Attacker intercepted a transaction request where the user intended to send $10.

- Attacker manipulated the request using Burp Suite, changing the transaction amount from $10 to $1000.

- As a result, $1000 was successfully debited from the user's account instead of $10.

Screenshots:





**Vulnerability 3: SQL Injection**

Description: SQL injection is a prevalent vulnerability where attackers input malicious SQL code into input fields, targeting websites that interact with databases like MySQL or Oracle. Successful exploitation can lead to unauthorized access to data or functionalities.

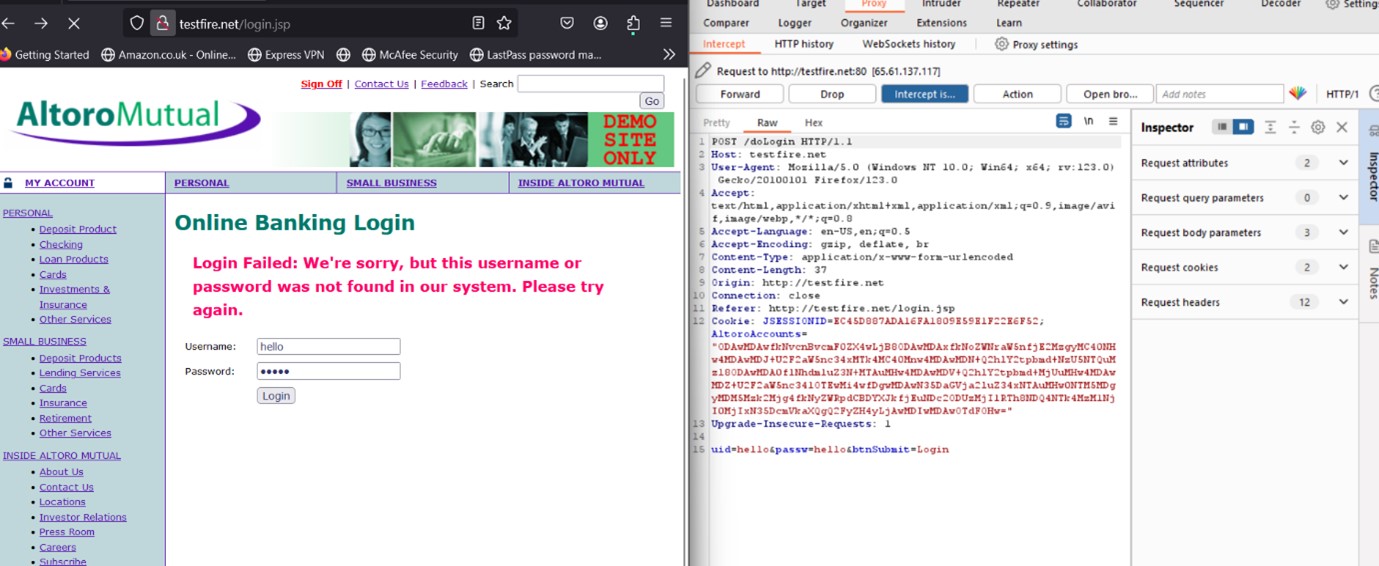
Exploit:

- Attacker identified a login page on testfire.net susceptible to SQL injection.

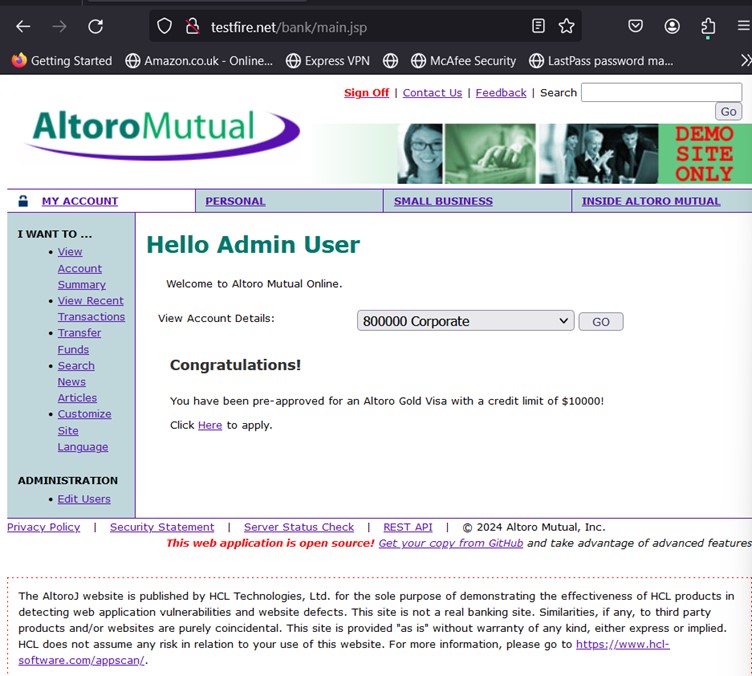
- Injected a malicious SQL query into the username or password field: `' OR 1=1--`

- Upon execution, the attacker bypassed the login page and gained unauthorized access to the admin account.

Screenshots:







---

Conclusion:

The conducted vulnerability tests on <http://testphp.vulnweb.com> <http://testphp.vulnweb.com> revealed critical security flaws, including Cross-Site Scripting (XSS), request manipulation, and SQL injection. These vulnerabilities pose significant risks to the confidentiality, integrity, and availability of the web application.

**Recommendations for Mitigation**:

1. Implement input validation and output encoding to prevent XSS attacks.

2. Utilize secure coding practices and implement server-side validation to mitigate request manipulation vulnerabilities.

3. Employ parameterized queries or ORM frameworks to prevent SQL injection attacks.

4. Regularly update and patch web application components to address known vulnerabilities.

By addressing these vulnerabilities and implementing robust security measures, the overall security posture of the web application can be significantly improved, reducing the risk of exploitation and unauthorized access.