AI for Cyber Security with IBM Qradar

Name: Prince Levin Panditi Email: prince.21bce9648@vitapstudent.ac.in

1. Target Website: Let us assume you have a basic web page with a search bar where users can search for products. The search query is displayed on the search results page without proper sanitization. Here I have taken demo.testfire.net

Vulnerability: We'll test for a reflected XSS vulnerability, where an attacker injects malicious code that is executed in the victim's browser when the search results are displayed.

Steps:

Go to demo.testfire.net

Identify Input Point:

Identify the input point where user data is displayed without proper validation or sanitization. In this case, it is the search query displayed on the search results page.

Test for Vulnerability:

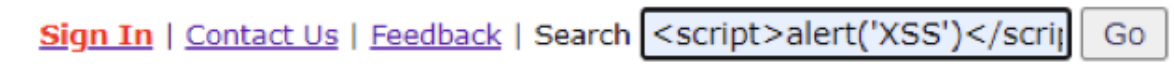
Search for a product using the search bar and enter a payload that triggers an XSS attack. For example, use the following payload:

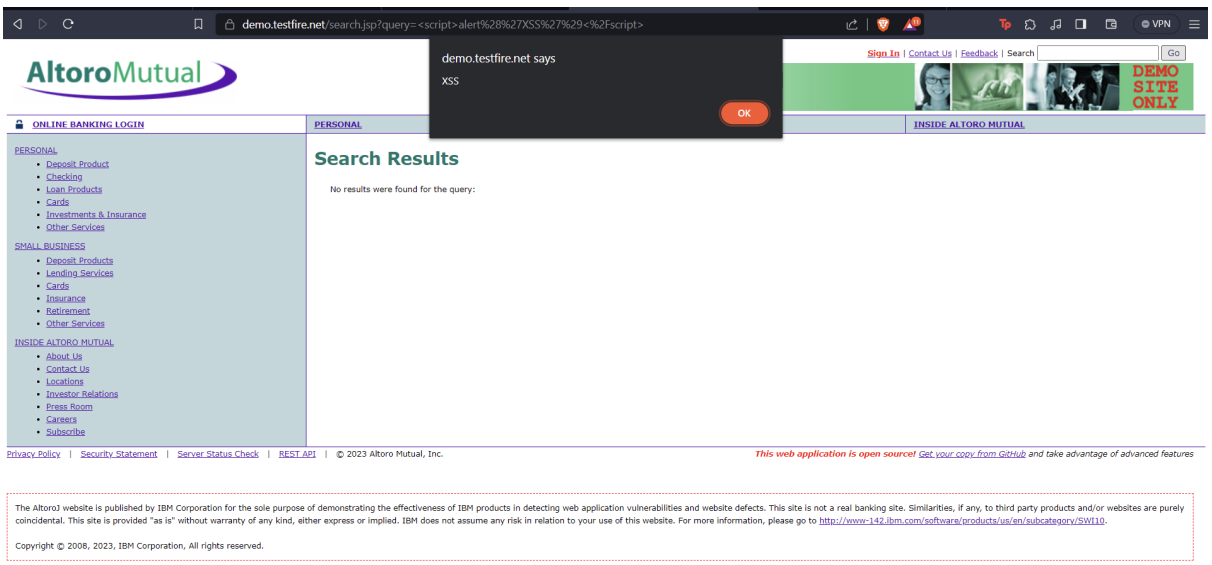
Run this code in the search box:

<script>alert('XSS')</script>

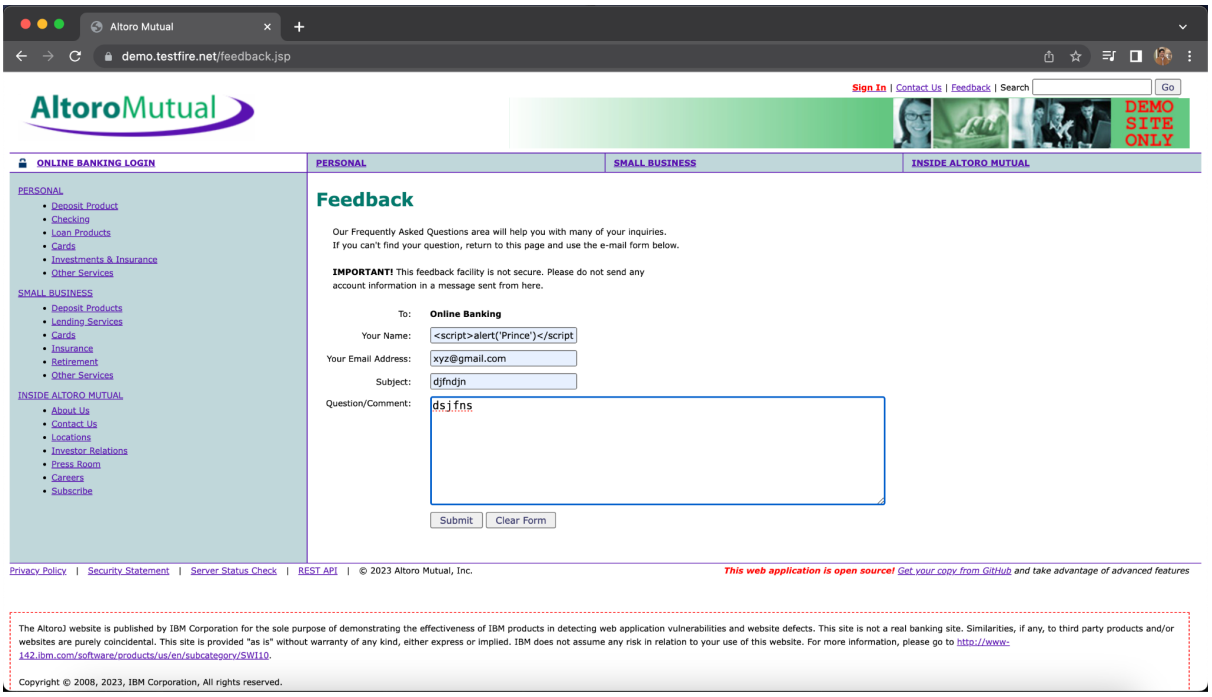
If the payload is executed as JavaScript code on the search results page and an alert box pops up, the vulnerability is present.

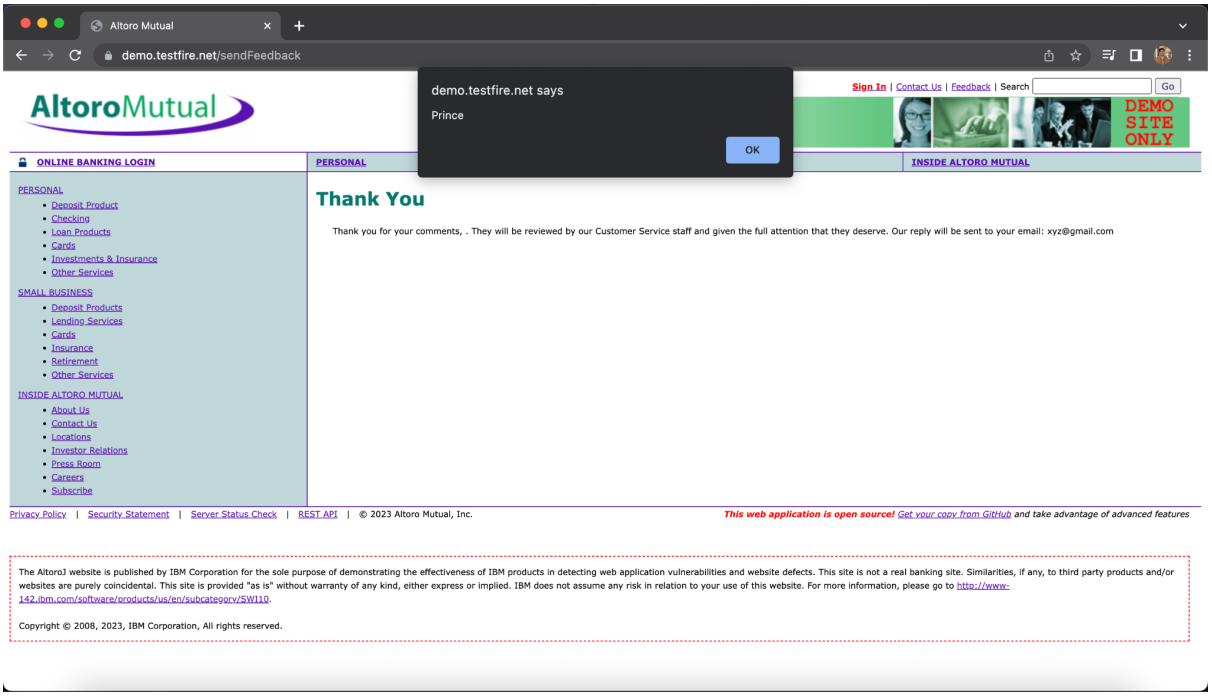
Click on GO





Fill the feedback form, in the place of the name inject a Javascript code. If the code works then it is vulnerable to Reflected XSS.





Mitigation:

To mitigate XSS vulnerabilities, the website should properly validate and sanitize user input before displaying it on the page. In this case, the search query should be properly encoded or sanitized to prevent the execution of malicious scripts.

1. Target Website: Let us assume you have a simple e-commerce website with a search bar that allows users to search for products by entering keywords. Here I have taken demo.testfire.net

Vulnerability: We'll test for a basic SQL Injection vulnerability, where an attacker manipulates the input to execute unauthorized SQL queries on the database.

Steps:

Identify Input Point:

Find an input point where user data is used in an SQL query. In this case, it's the search bar.

Test for Vulnerability:

Enter a single-quote character (') in the search bar as input. If the website returns an error message that indicates a database error, there is a possibility of a vulnerability.

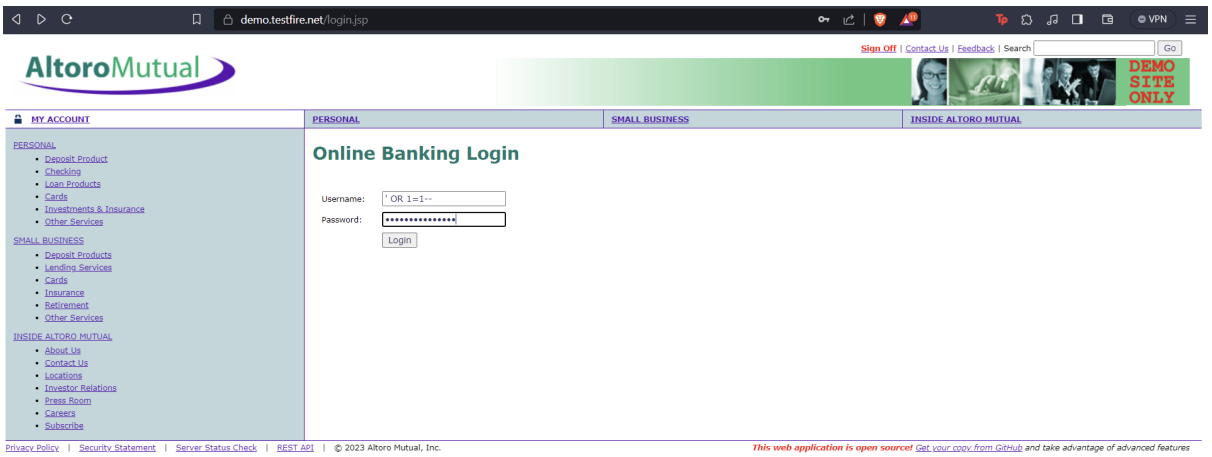
Inject SQL Code:

Try injecting a simple SQL payload to manipulate the query. For example, use the following payload to make the query always true, which could potentially return all products in the database:

Inject this SQL Query inside username and give any random password.

' OR 1=1--

If the search results now display all products or show different behavior, the vulnerability might be present.



It is showing that the login is successful.

Mitigation:

To mitigate SQL Injection vulnerabilities, websites should use parameterized queries or prepared statements to handle user input. Parameterized queries ensure that user input is treated as data and not executable SQL code.