**Web Application**

**Vulnerability Assessment**

**and**

**Penetration Testing Report**

**For**

**https://demo.testfire.ne**

**From Rahul Gupta**

**Assessment Date:25-02-2022**

**Table of Contents**

**1.INJECTION**

1.1 SQL INJECTION…………………………………….………3

1.2 HTML INJECTION……………………………….…………6

**2.CROSS SITE SCRITPING(XSS)**

2.1 XSS REFELCTED………………………………...………….9

2.2 XSS COOKIES………………………………………………10

**3.SECURITY MISCONFIGURATION**………….…………………12

**4. BACK BUTTON BROWSING**……………………………………14

**5.SENSITIVE DATA EXPOSURE** …………………………………16

**6. MISSING OF HST HEADER**…………………………………….18

**1 INJECTION**

* 1. **SQL INJECTION**

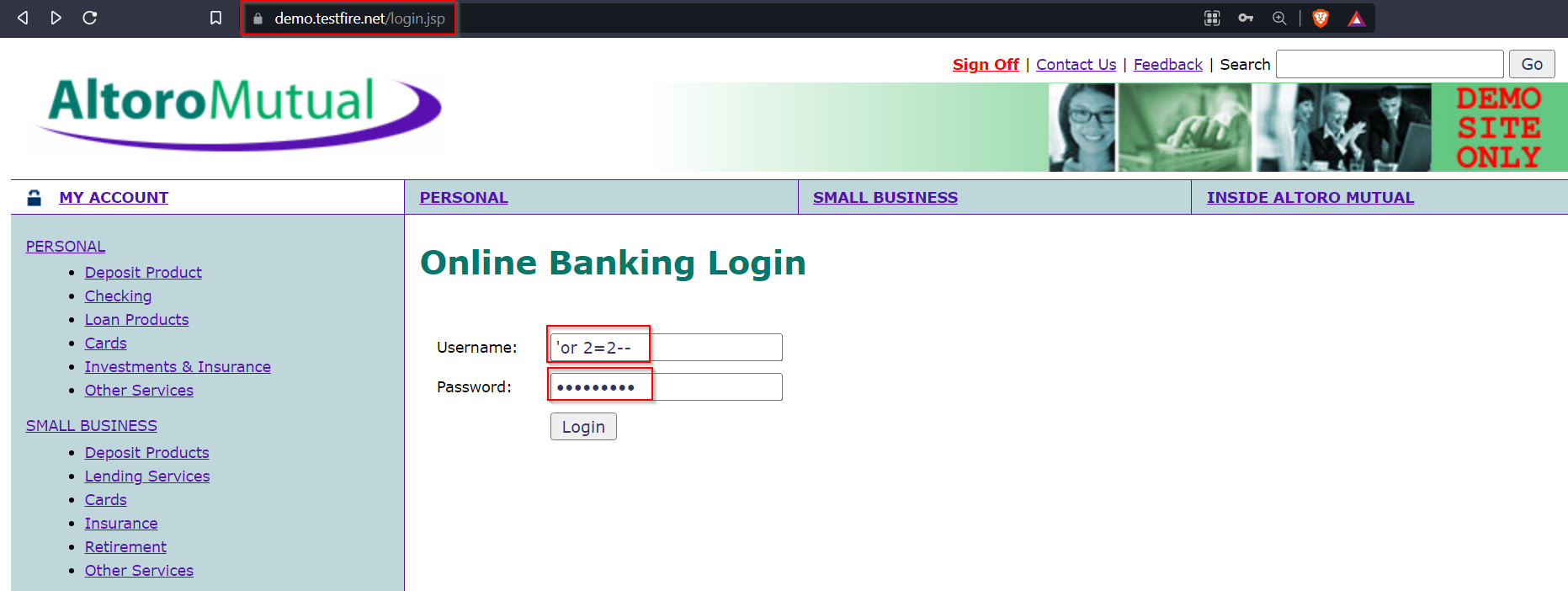
infected url: <https://demo.testfire.net/login.jsp>

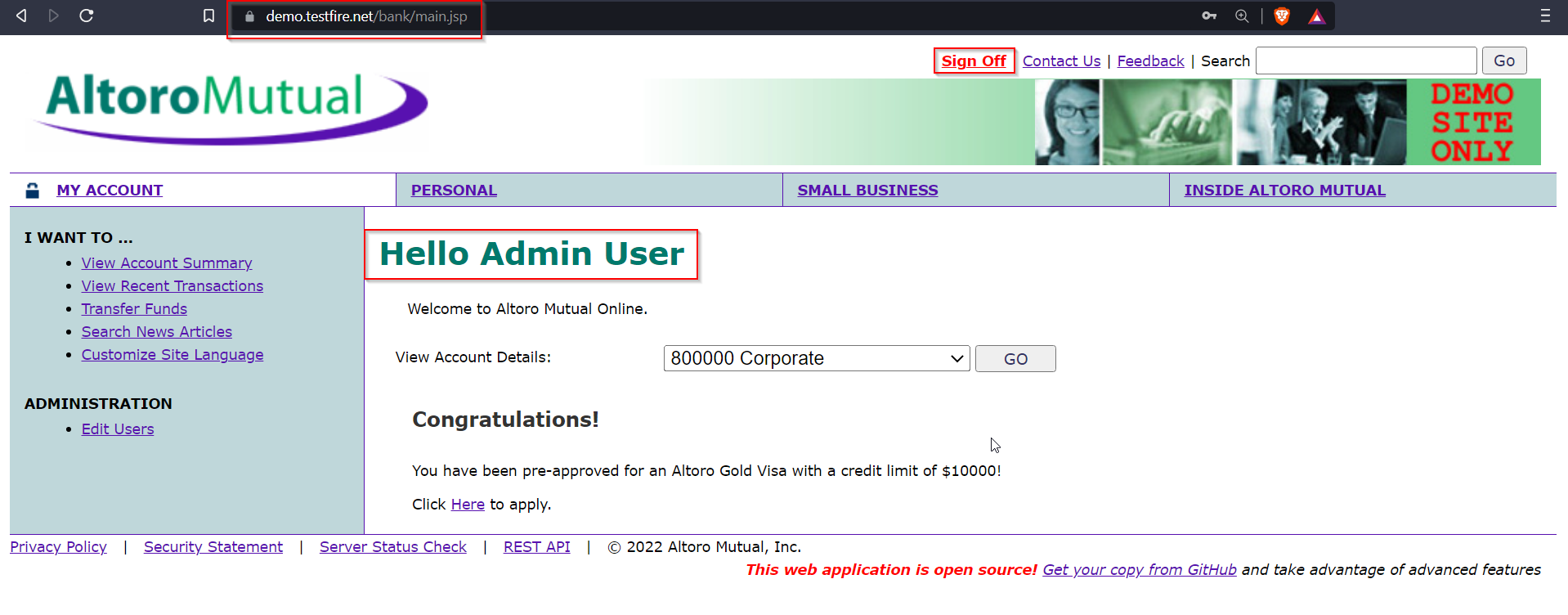
payload used: ‘ ‘

This specifies attacker can perform SQl injection

**SQL injection part:2**

Infected url: <https://demo.testfire.net/login.jsp>

payload used: ‘or 2=2--



using above payload attacker can get access to **ADMIN** account

**Impact:**

1. Attacker can log in application as admin can use sensitive
2. Attacker can make changes in system and can make changes in database
3. Attacker will have access to whole application can leak application data
4. Attackers can use SQL injections to alter or add new data to the accessed database.

**Mitigation’s**

1.sql injection can be prevented by using parameterized queries

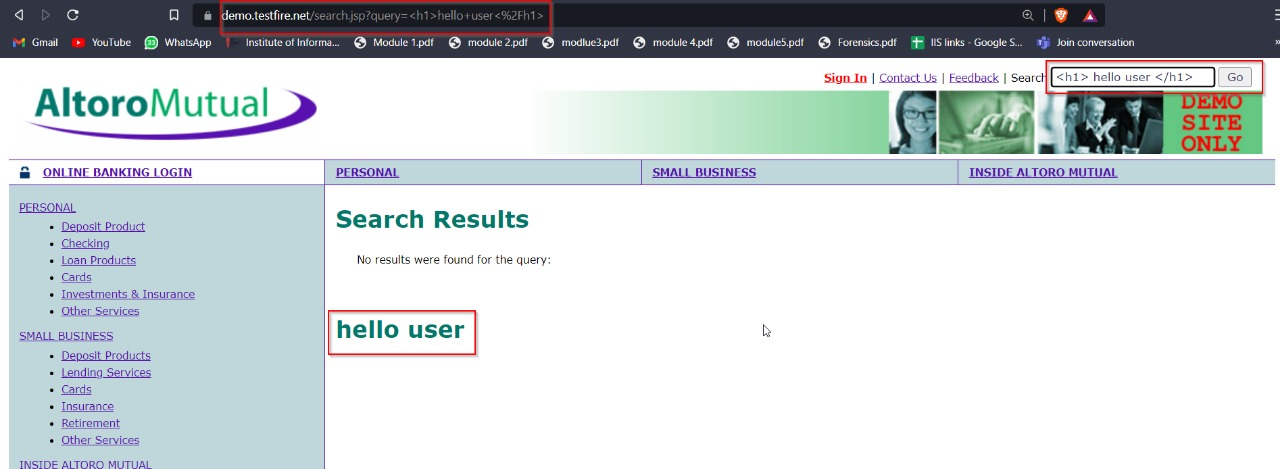
Parameterized SQL queries allow you to place parameters in an SQL query instead of a constant value. A parameter takes a value only when the query is executed, which allows the query to be reused with different values and for different purposes.

2. **Input validation**

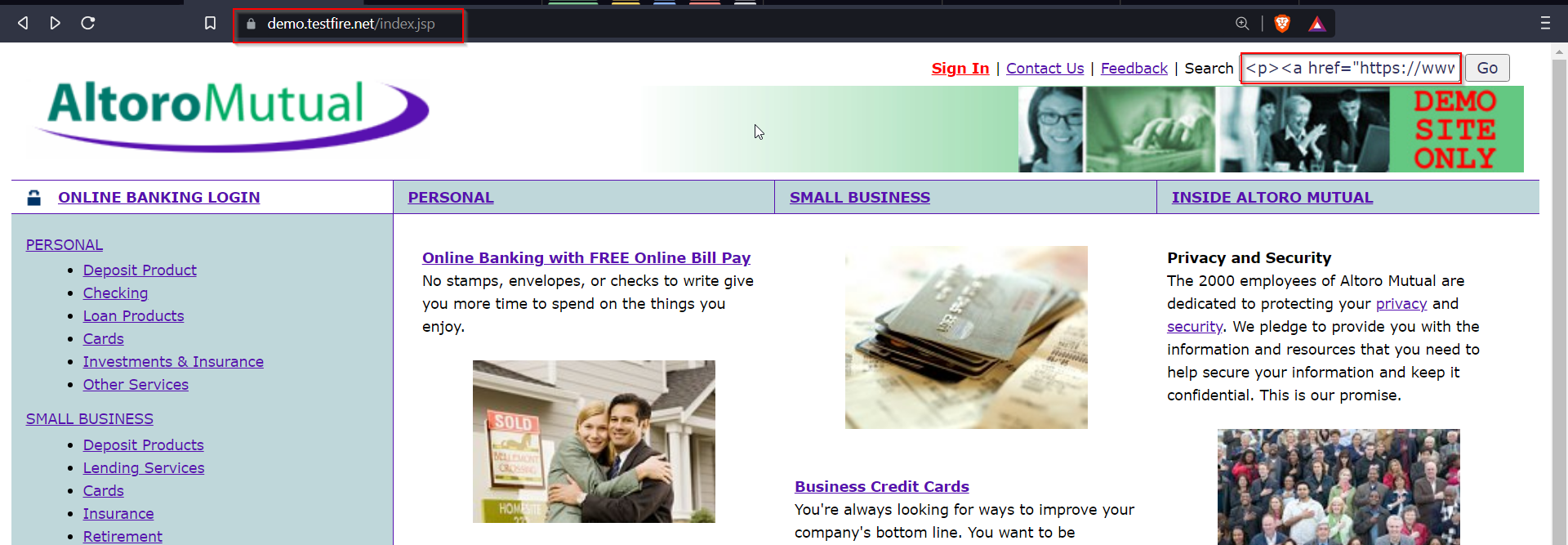
It is the first step of checking the type and content of data supplied by a user or application. Improper input validation is a major factor in many web security vulnerabilities SQL injection.

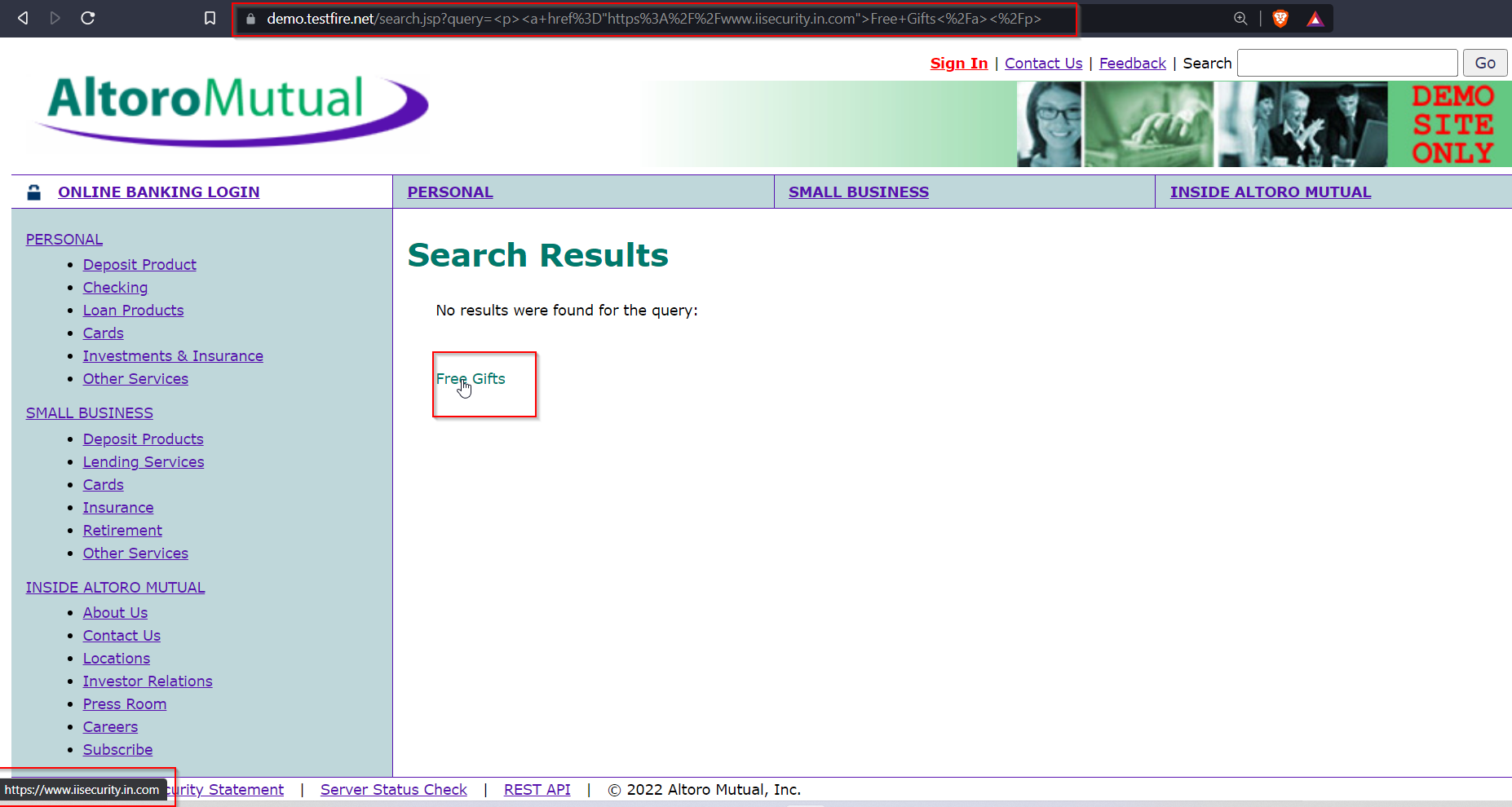
**1.2 HTML INJECTION**

Infected url: <https://demo.testfire.net/index.jsp>

Payload we used: <h1> hello user </h1>

Injecting url in html field

Payload: <p><a href="https://www.iisecurity.in.com">Free Gifts</a></p>



**Impact of HTML Injection:**

1.It can allow an attacker to modify the page.

2.The attacker discovers injection vulnerability and decides to use an HTML injection attack.

3.Attacker crafts malicious links, including his injected HTML content, and sends it to a user via email.

**Mitigations**

1.Every input should be checked if it contains any script code or any HTML code.

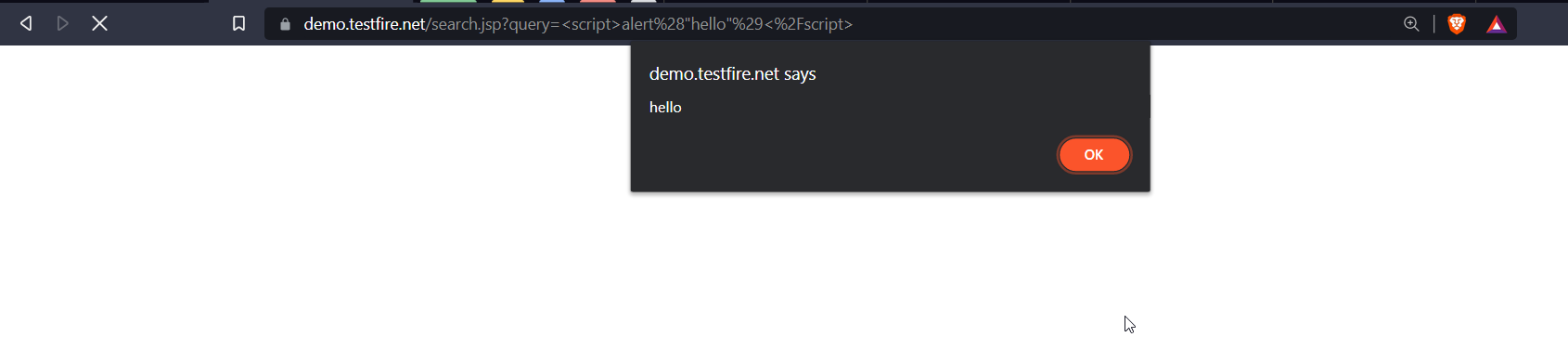
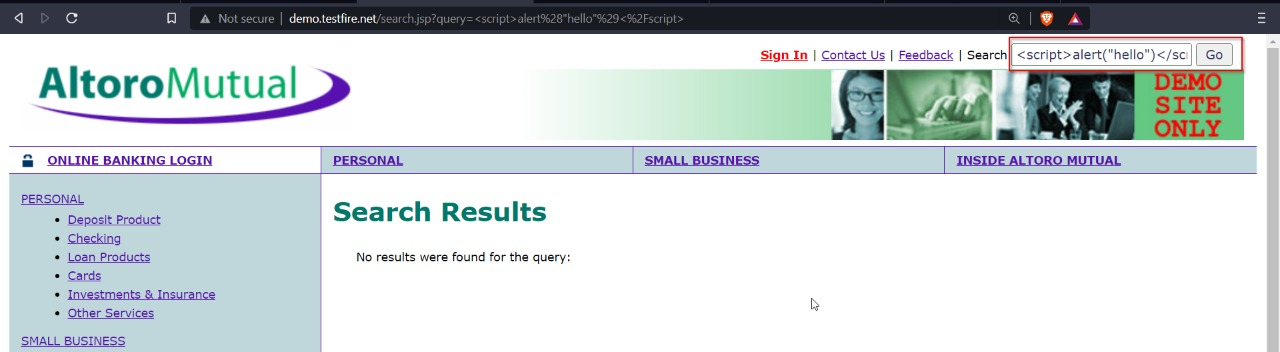
2.Application should check, if the code contains any special script or HTML brackets – <script></script>, <html></html>

1. **CROSS SITE SCRIPTING**

**2.1 cross site scripting (reflected)**

**Infected url**: <https://demo.testfire.net/index.jsp>

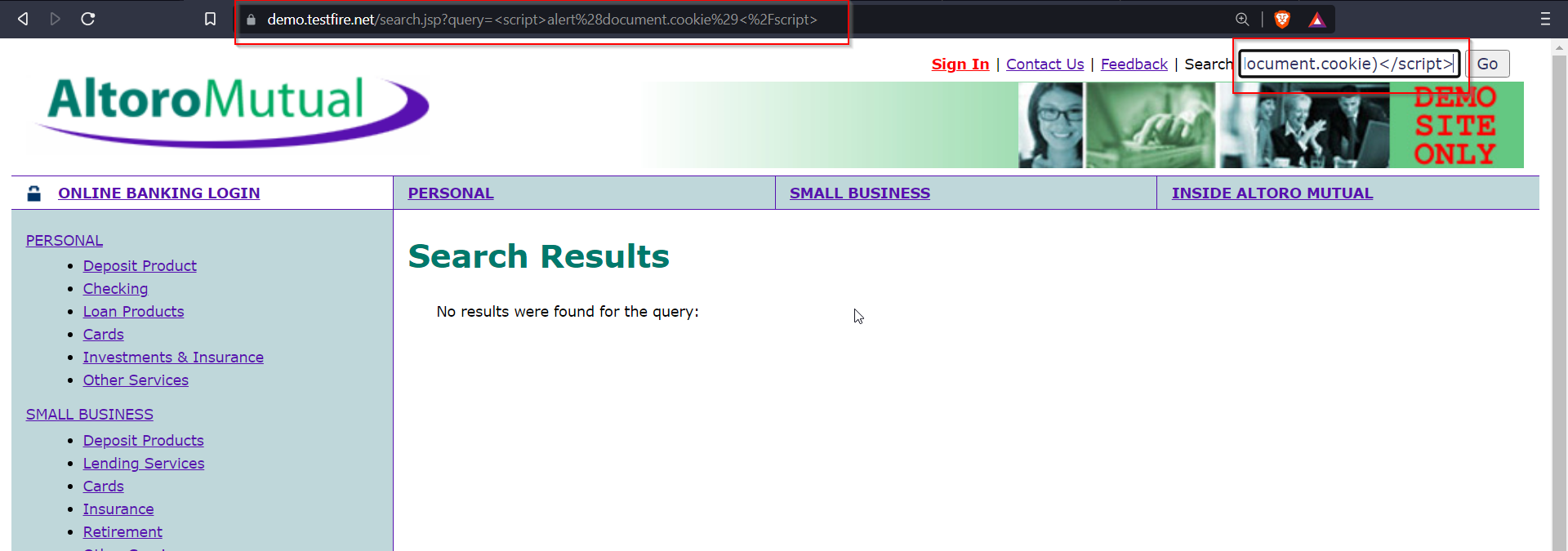
**payload used:** <script>alert(“hello”) </script>

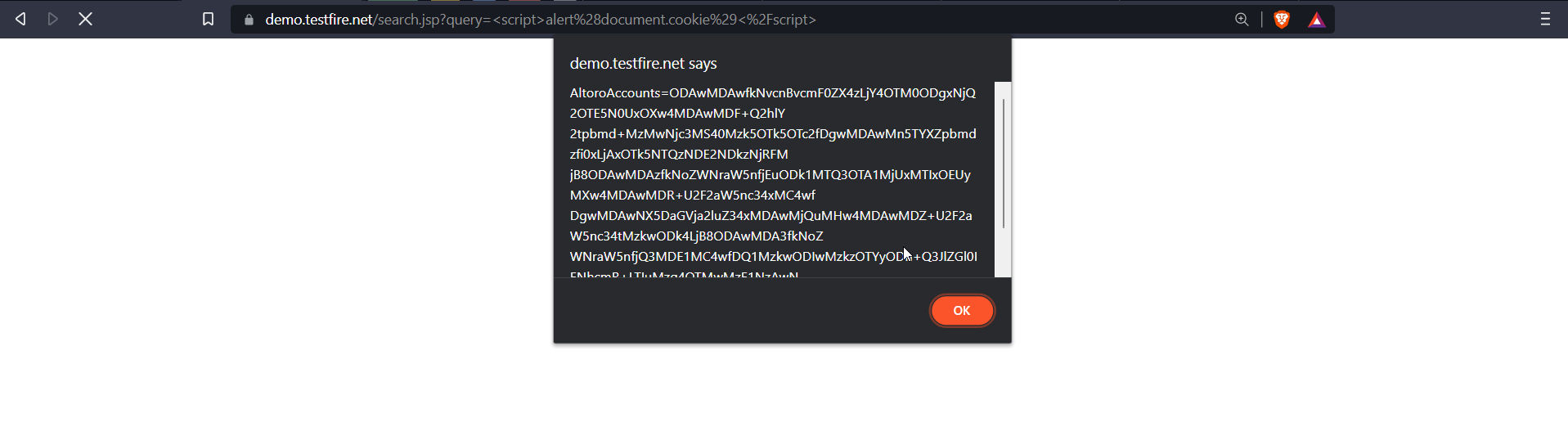


Attacker can use script in text fields

**2.2 Xss to get user cookies**

**Infected url**: <https://demo.testfire.net/index.jsp>

**Payload:** <script>alert(document.cookie)</script>



**Impact of cross site scripting**

**1.Data Leakage**

If attacker get user cookie user data will be in risk. Cookies are small files that websites send to user device that the sites then use to monitor user and remember certain information about user

**2.Session hijacking**

If the server doesn't set the HttpOnly attribute in session cookies, then malicious scripts can get at your session ID. An example of a cross-site scripting attack to execute session hijacking would be when an attacker sends out emails with a special link to a known, trusted website.

**Mitigation**

**1. Sanitizing Input**

Reflected and stored cross-site scripting can be sanitized on the server-side and there are multiple ways of doing it.

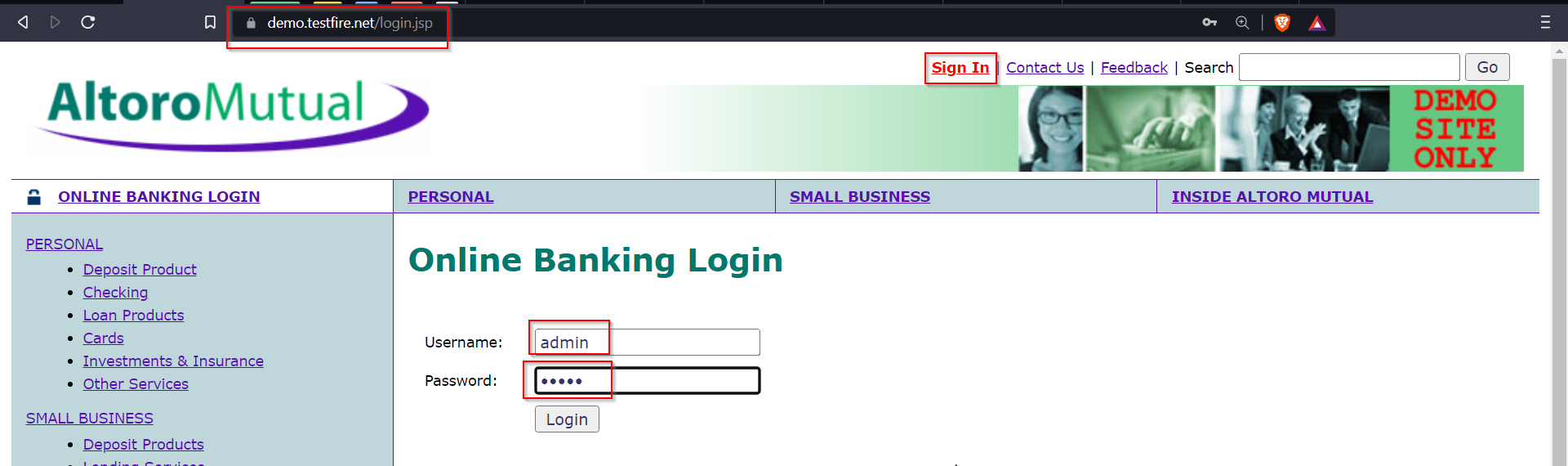
**2.Blacklisting**

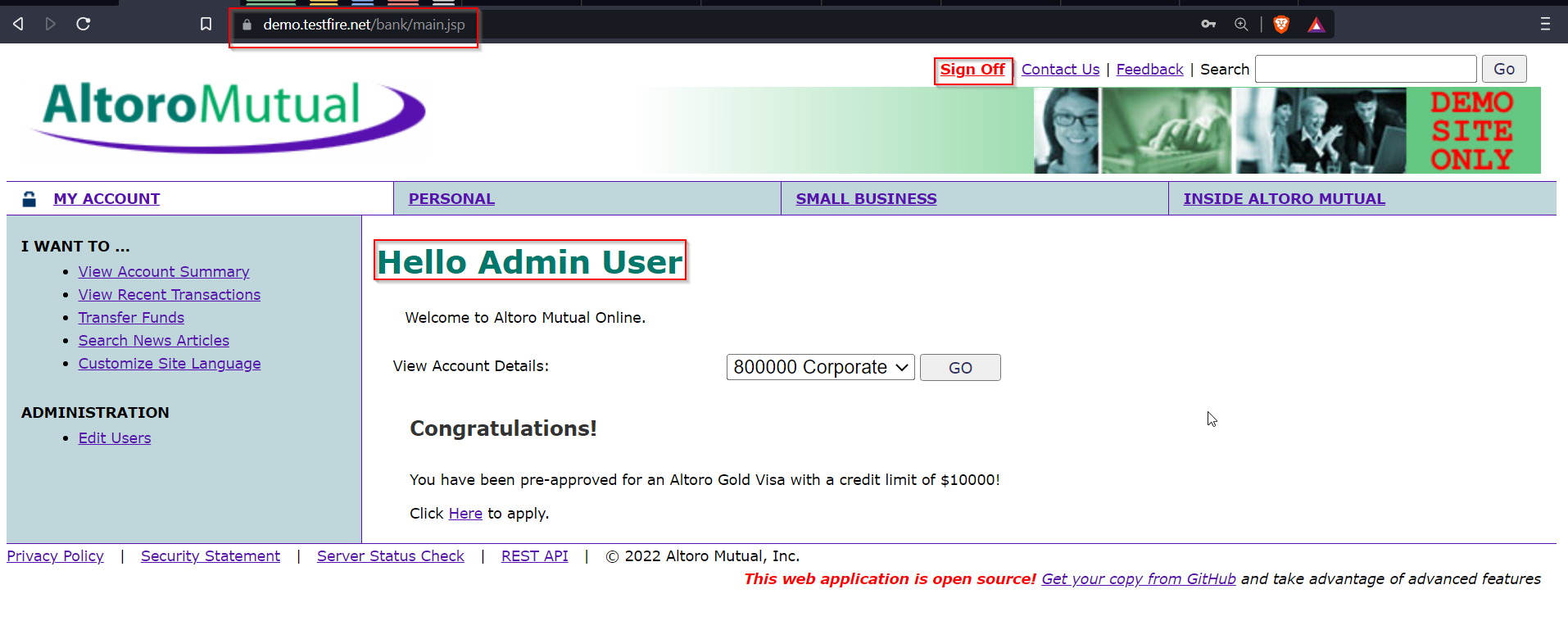
Blacklisting characters that are unsafe won’t really work out in the long run since some malicious user might figure out some bypass for it as it usually happens. What you need to do is whitelist what is allowed.

**3.Security Misconfiguration**

**Infected url**: <https://demo.testfire.net/login.jsp>

**Payload:** admin admin





Attacker can use default credential to access this application

**Impact**

**1.Access to unauthorized features**

Once attacker have access to admin portal whole application will compromise. every user’s data will be access to attacker

**2.Data Misuse**

As attack have access, they can use this data to impersonate. Ask for Ransome, post in social media.

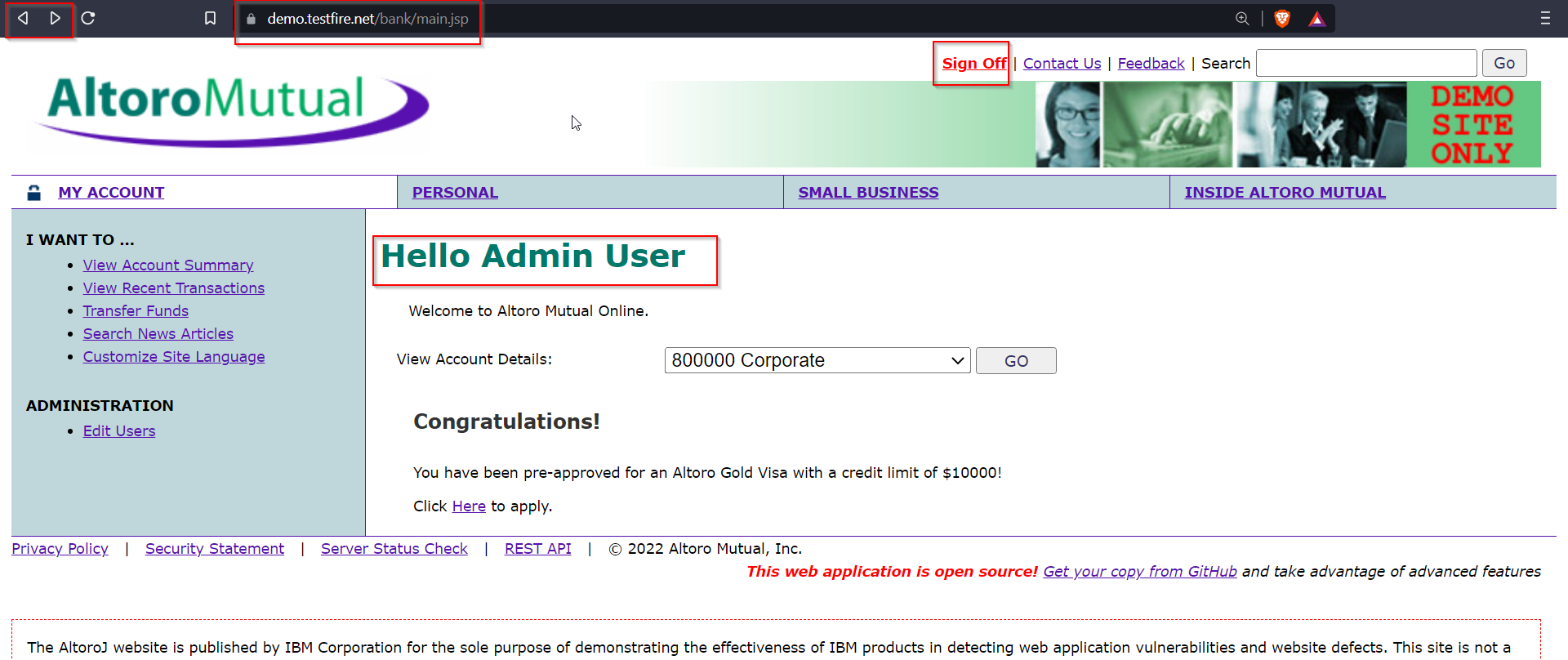
**3.Access to database**

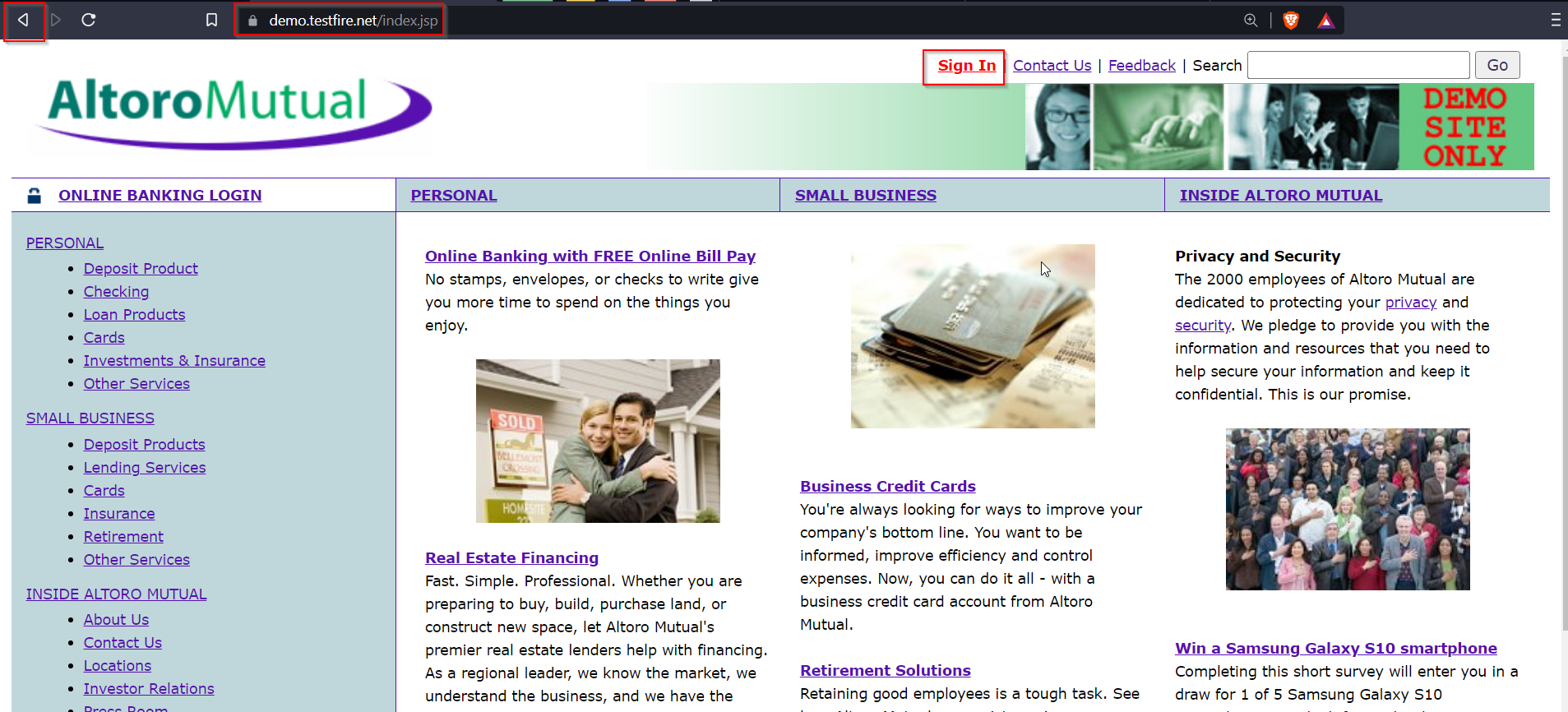
Attacker can Alter, update or delete database

**Mitigation**

1. Remove default credentials
2. Implement password policy
3. Use multifactor authentication for login
4. Components should be always updated

**4.Back button browsing**

<https://demo.testfire.net/index.jsp> 



Application is vulnerable to Back Button Browsing

**Impact**

1.Attacker can make use of back button and navigate to the previous pages visited by the user

2. If attacker click back button after user have logout, browser automatically resubmit the request with all the information

3. Attacker can have full access to user account and they can use data

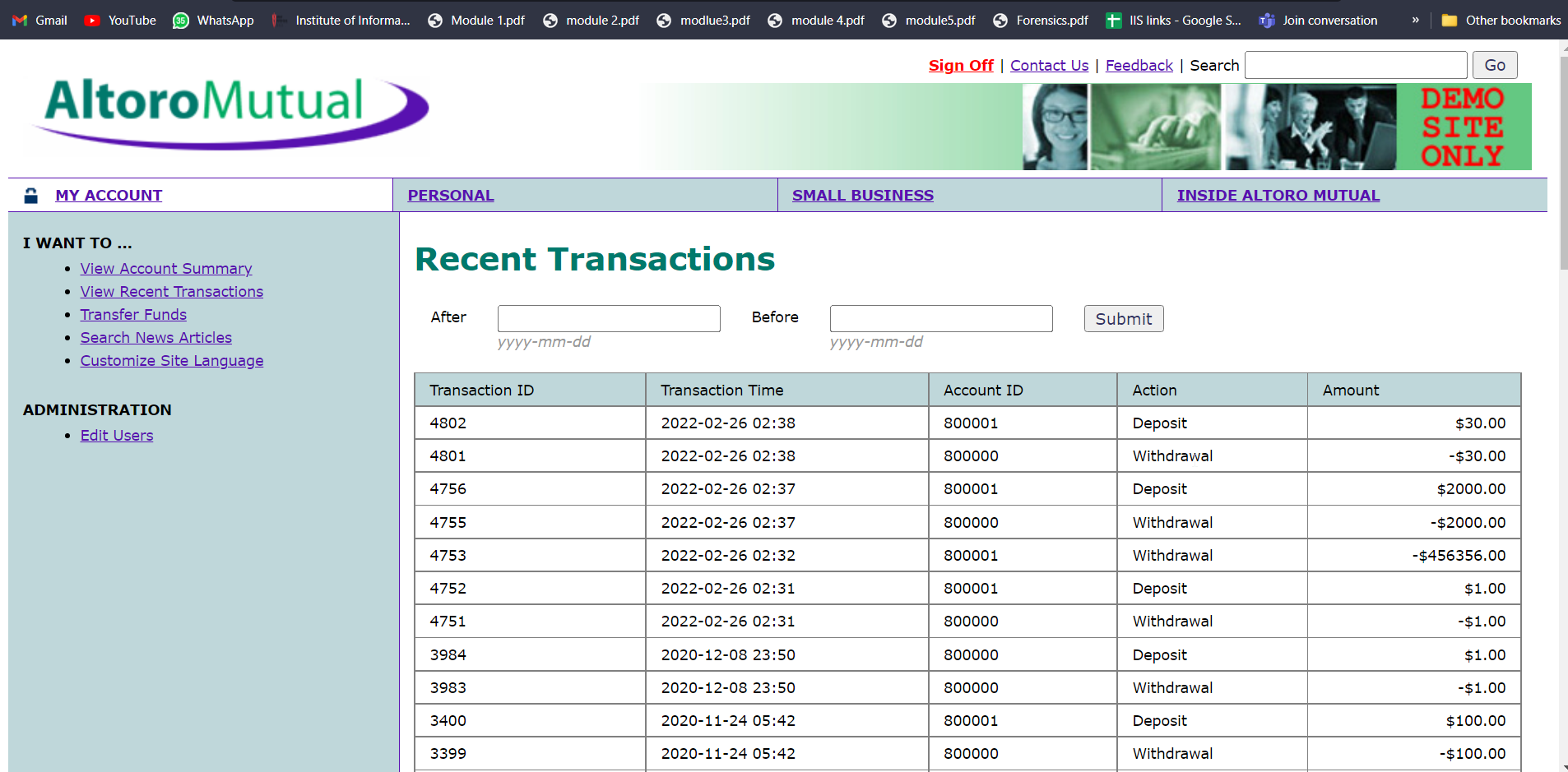
**Mitigation**

1.Implement session management correctly

2. session should be destroyed as user logout from application

**5.Sensitive Data Exposure**

**Infected url:** <https://demo.testfire.net/bank/transaction.jsp>



As attacker have access to admins account, all the transaction data is accessible to cackler

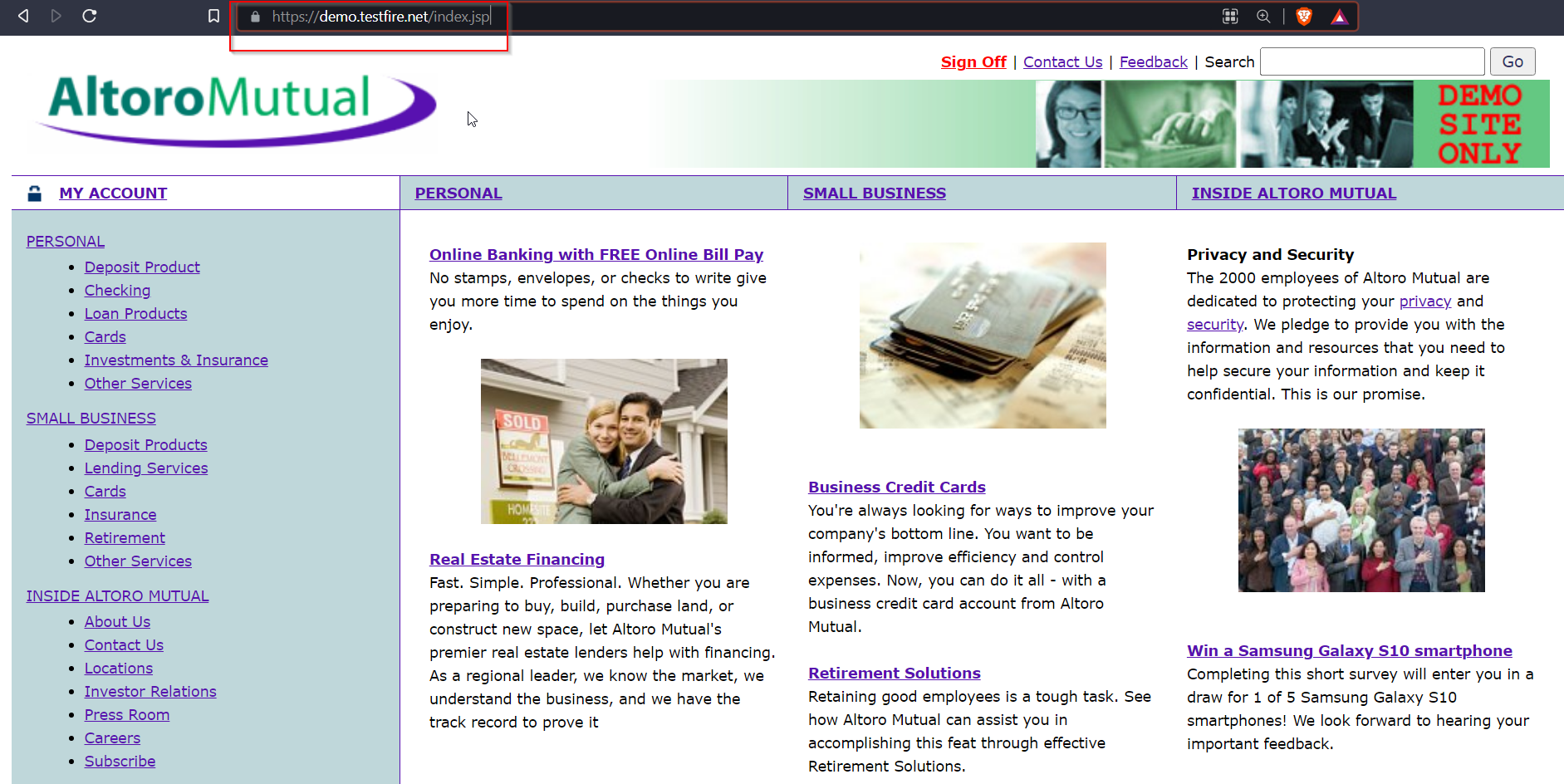
**Impact:**

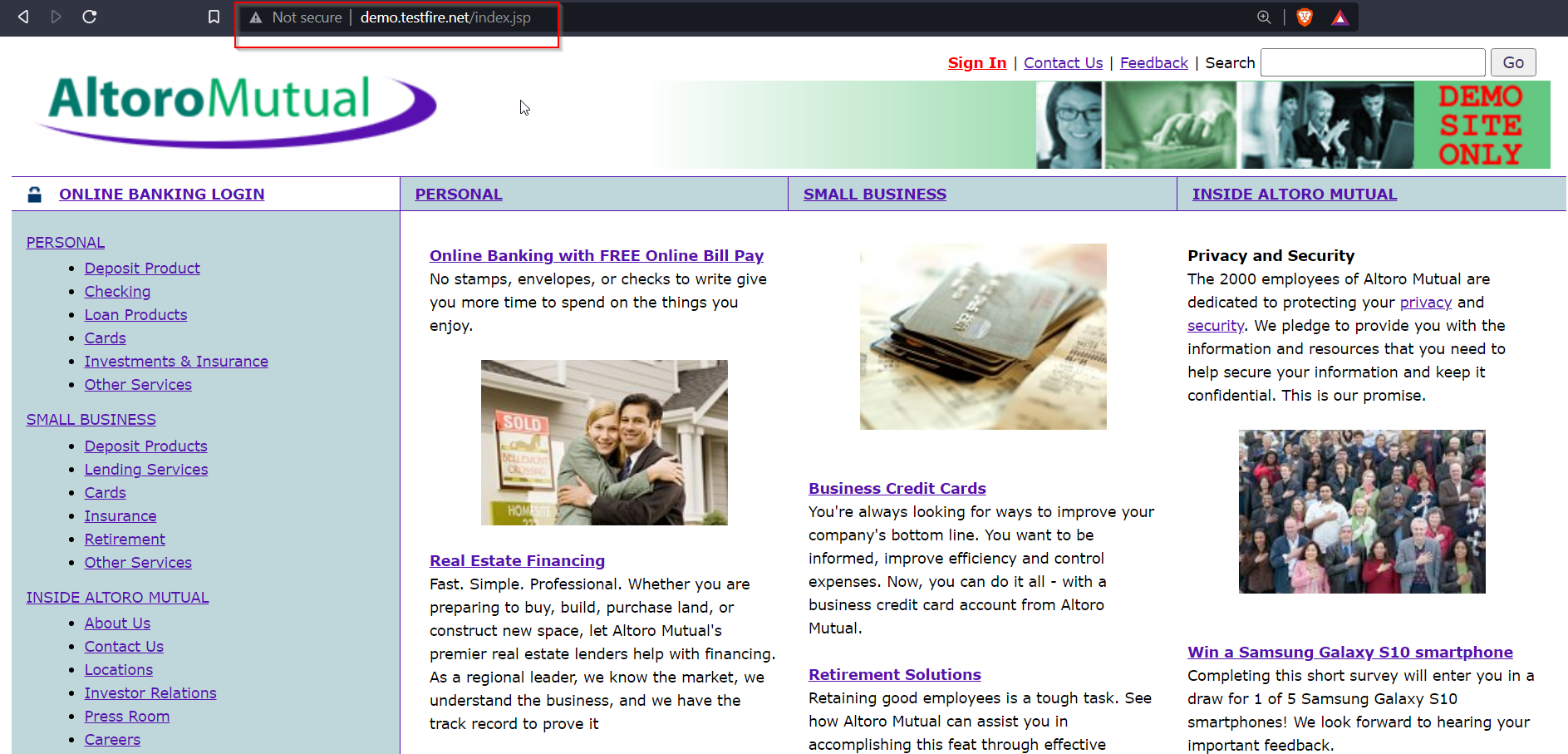
1. Missing or poor encryption is one of the most common vulnerabilities that lead to the exposure of sensitive data.
2. Security exposure occurs because application is not using encryption technique on password

**Mitigation**

1. Ensure you encrypt all sensitive data at rest
2. Store passwords with strong salted hashing functions. Make use of key derivation functions like bcrypt, Argon2, scrypt or PBKDF2.
3. Disable web server directory listing and ensure file metadata is not present in web

**6.Missing of HTTP strict Transport Security (HSTS)**





Attacker is able to transfer website from https to http

**Impact**

1. If website works on http also, then attackers can capture all the data that is transmitting over network
2. http doesn’t use encryption when transmitting data
3. if user is doing any payment attacker can perform MIMT attack and can intercept all the data

**Mitigation**

1. application should use HTTP Strict Transport Security
2. this header make user to use application only on https , as https is secure all the data in transits is encrypted no one can alter data
3. HSTS header can help protect application against protocol downgrade attack, such as Poodle attack