Vulnerability Assessment & Penetration Testing (VAPT) Report

Target Application: Testfire Bank (http://testfire.net)

Prepared By:
Shivam Chilkewar
Red Team & VAPT Security Researcher
Email : chilkewarshivam@gmail.com
Phone : +91 94206 14292

Organization / Project:
Independent Security Assessment (For Academic & Training Purposes)

Date of Assessment: 02 October 2025
Report Version: v1.0

Tools Used:
Burp Suite , Nmap , SQLmap , Dirsearch , Nikto , Postman , BloodHound , Mimikatz , Custom Python VAPT Framework

---

Confidentiality Notice

This document contains results of a controlled security assessment performed on the Testfire Bank application for learning and demonstration purposes.
Unauthorized distribution or misuse of the information contained herein is prohibited.

📜 Table of Contents

3. Methodology

The Vulnerability Assessment and Penetration Testing (VAPT) for the Testfire Bank web application was conducted using a hybrid approach, combining both automated scanning and manual exploitation techniques.
The goal was to identify, validate, and document vulnerabilities that could potentially compromise the confidentiality, integrity, or availability of the application and its associated components.

The overall process followed the standard PTES (Penetration Testing Execution Standard) and OWASP Testing Guide v4 framework, ensuring comprehensive coverage across all security layers.

3.1 Reconnaissance

The reconnaissance phase involved gathering publicly available information about the target application to identify exposed components, technologies, and potential entry points.
Activities included:

- Identifying technologies using Wappalyzer and WhatWeb
- Discovering accessible endpoints using dirsearch
- Collecting metadata and application behavior through manual browsing and Burp Suite proxy capture

---

3.2 Scanning and Enumeration

In this phase, the application was scanned using automated tools to detect common vulnerabilities and misconfigurations.

- Nmap was used for port and service discovery.
- Nikto was used to identify known server misconfigurations.
- SQLmap, Burp Suite Scanner, and Dirsearch were utilized to identify input validation and injection points.

3.3 Vulnerability Analysis

All gathered information was analyzed to pinpoint genuine security weaknesses.
Automated tool results were manually verified to remove false positives.
Potential issues were mapped against the OWASP Top 10 2021 vulnerabilities to ensure coverage of critical web application threats.

3.4 Exploitation

This phase focused on safely exploiting identified vulnerabilities to confirm their impact and feasibility of compromise.
Controlled proof-of-concept (PoC) attacks were conducted for:

- SQL Injection
- Cross-Site Scripting (XSS)
- Insecure Direct Object Reference (IDOR)
- Broken Authentication
- Information Disclosure

No destructive testing (e.g., Denial of Service) was performed during this assessment.

3.5 Post-Exploitation

Once exploitation was successful, post-exploitation activities were carried out in a controlled environment to demonstrate possible escalation paths.
Examples include:

- Accessing restricted user data through IDOR
- Demonstrating privilege escalation using default credentials
- Validating data extraction through SQLi

All activities were limited to the authorized test scope.

3.6 Reporting and Recommendations

All verified vulnerabilities were documented in this report along with:

- Severity ratings based on CVSS v3.1
- Business impact analysis
- Proof-of-Concept evidence (screenshots, payloads)
- Actionable remediation recommendations

The report aims to help stakeholders understand security risks and prioritize mitigation steps effectively.

Summary:

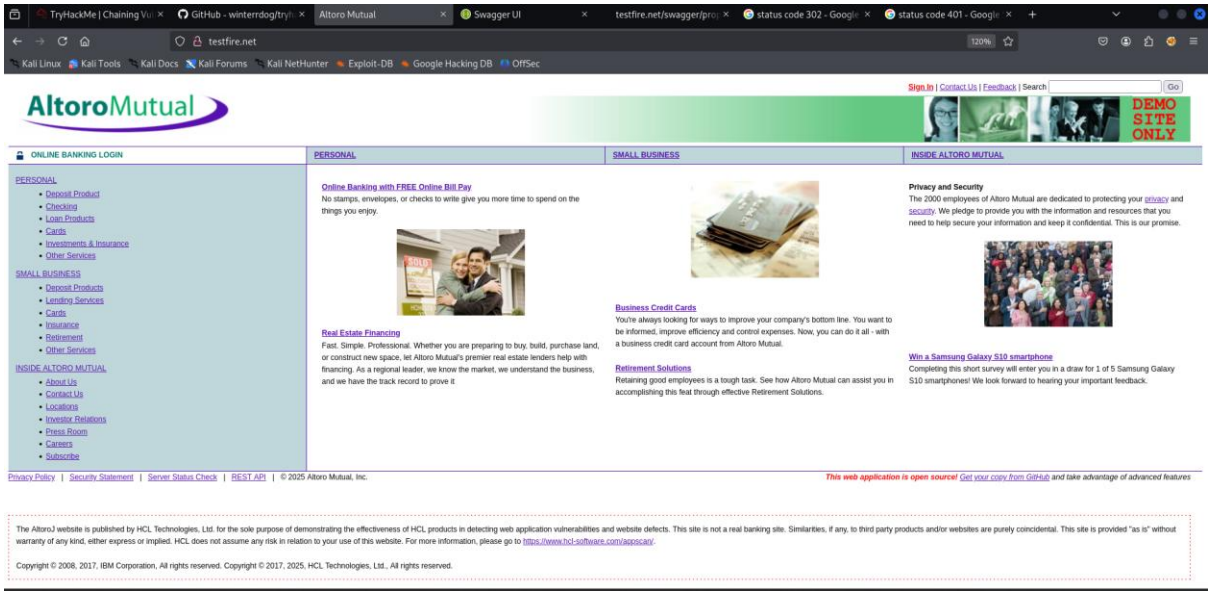
This methodology ensures a comprehensive, repeatable, and risk-focused approach to web application security testing, providing accurate and actionable results for improving the security posture of the Testfire Bank platform.

Vulnerability Assessment & Penetration Testing (VAPT) Report

Target: http://testfire.net/

Date:02/10/2025

Home page of target:



Vulnerability Details

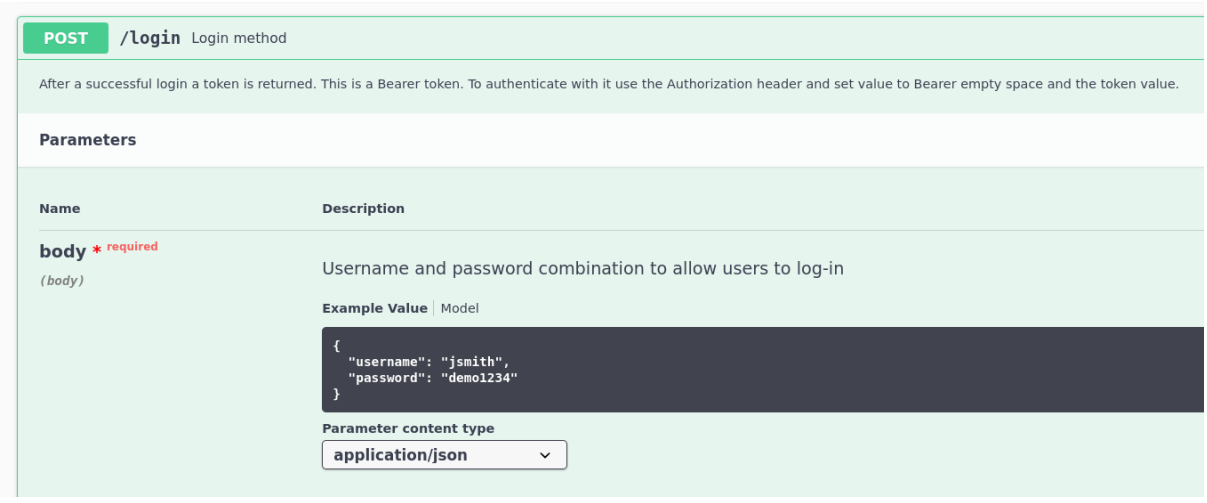
1. **Information Disclosure** or **Sensitive Data Exposure** (Specifically via a poorly secured **REST API** endpoint on the homepage).
    a. **Severity: High or Critical.**

- It is **High** because a successful attack grants direct access to valid user credentials.
- It could be **Critical** if those credentials belong to an administrator, a service account, or are reused on other high-value systems.

b. Impact:

- **Direct Compromise:** The immediate impact is the **compromise of the jsmith user account**, allowing an attacker to log in and act as that user.
- **Horizontal Escalation:** If jsmith has access to sensitive data (customer records, financial data, intellectual property), this data is exposed.
- **Vertical Escalation:** If the jsmith account is a service account or an administrator, the attacker could gain full control over the application or even the underlying server infrastructure.
- **Reputation Damage & Regulatory Fines:** If customer data is breached, the organization faces potential regulatory non-compliance (e.g., GDPR, HIPAA) and severe reputational harm.
- **Wider API Attack Surface:** The presence of the API URL itself on the homepage suggests poor security hygiene, potentially exposing other unauthenticated or weakly protected endpoints.

Their is Rest API URl is present in the home page after getting into it got jsmith credentials.



Logged in as jsmith

PERSONAL | SMALL BUSINESS

Hello John Smith

Welcome to Altoro Mutual Online.

View Account Details: 800002 Savings GO

Congratulations!

You have been pre-approved for an Altoro Gold Visa with a credit limit of $10000!

Click Here to apply.

2. **Use of Default/Weak Credentials** (Often categorized under **Authentication Flaws** or **Inadequate Credential Management**).
    a. **Severity: Critical**.
        - This is the highest level of severity because it grants an unauthenticated attacker **immediate, full administrative access** to the system. There is no need for brute-forcing, exploiting a flaw, or performing complex steps.

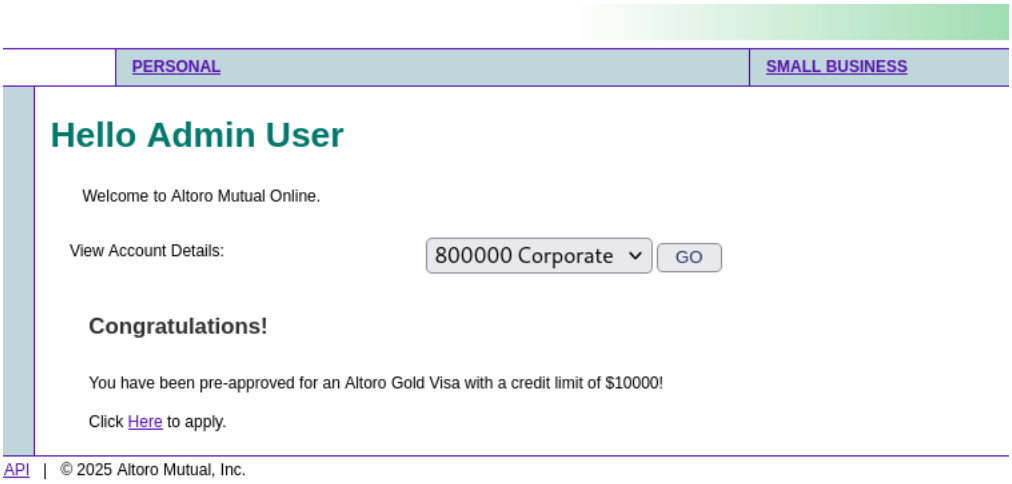
    b. **Impact:**

        - **Total System Compromise:** An attacker now has the highest level of access possible, allowing them to:
            - **Modify/Delete Data:** Access, change, or destroy any application or customer data.
            - **System Takeover:** Install malicious software, change configurations, or create backdoors.
            - **Lateral Movement:** Use the compromised system as a launchpad to attack other systems on the network.
        - Regulatory Non-Compliance: This type of failure almost guarantees a violation of data protection laws (like GDPR, HIPAA, etc.) if customer data is involved.
        - **Reputation Damage:** A successful attack leading to data loss or system downtime will cause severe reputational harm and potential legal liability.

    tried credentials:

    Username:admin & Password:admin

    Got admin panel

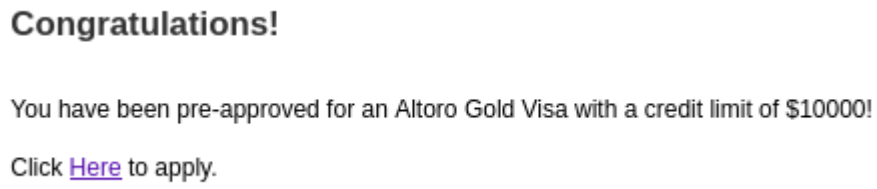


3. **Broken Authentication and Authorization** (Specifically, a flaw in the **Re-authentication/Step-up Authentication** mechanism).

   This is sometimes called a **Logic Flaw** because the application's logic for verifying the password check is fundamentally broken.

   a. **Severity: High**.
      - It is High because an attacker (or a user with malicious intent) who has already gained session access can bypass a critical security control (password re-entry) to perform a sensitive financial action. It directly compromises the **integrity** of the user's account and the financial transaction.

   b. **Impact:**
      - **Financial Fraud and Account Takeover:** An attacker who has merely hijacked an existing, active session (e.g., through XSS, session fixation, or a previously stolen session cookie) can instantly perform sensitive actions like increasing the credit limit, transferring funds, changing contact information, or adding new payment recipients—all without ever knowing the actual password.
      - **Bypassing Security Controls:** The intended "step-up authentication" (re-entering the password) is a control designed to confirm the user's identity before high-risk actions. Bypassing it renders this critical security measure useless.
      - **Reputational and Trust Damage:** If users suffer financial losses because of this flaw, the institution's reputation and user trust will be severely damaged.



After clicking apply button for Altoro Gold Visa we have to input password for authentication in this we can add any random text for authentication.

Altoro Mutual Gold Visa Application

No application is needed. To approve your new $10000 Altoro Mutual Gold Visa with an 7.9% APR simply enter your password below.

Password: ••••

Submit

```
POST /bank/ccApply HTTP/1.1
Host: testfire.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:143.0)
Gecko/20100101 Firefox/143.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Origin: http://testfire.net
Connection: keep-alive
Referer: http://testfire.net/bank/apply.jsp
Cookie: JSESSIONID=5A63C46273575D0FCB5650892l3BD9DF; AltoroAccounts=
"ODAwMDAyflNhdmluZ3N+MS44NDQ2NzQ0MDc4MDA1NDUzRTE5fDgwMDAwM35DaGVja2luZ3
42LjkyMjMzNzIwMzkwMDIyOUUyMHw0NTM5MDgyMDM5Mzk2Mjg4fkNyZWRpdCBDYXJkfjEwM
C40Mnw="
Upgrade-Insecure-Requests: 1
Priority: u=0, i

passwd=test&Submit=Submit
```

```
1  HTTP/1.1 200 OK
2  Server: Apache-Coyote/1.1
3  Content-Type: text/html;charset=ISO-8859-1
4  Content-Length: 5361
5  Date: Thu, 02 Oct 2025 10:19:23 GMT
6
7
8
9
10
11
12
13
14 <!-- BEGIN HEADER -->
15 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
16
17 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" >
18
19
20
21     <head>
```

Altoro Mutual Gold Visa Application

Your new Altoro Mutual Gold VISA with a $10000 and 7.9% APR will be sent in the mail.

4. **Cross-Site Scripting (XSS)**.
    a. **Severity: Medium** to **High**.

        It is generally considered High because an attacker can completely compromise the integrity and confidentiality of the victim's session.

    b. **Impact:**

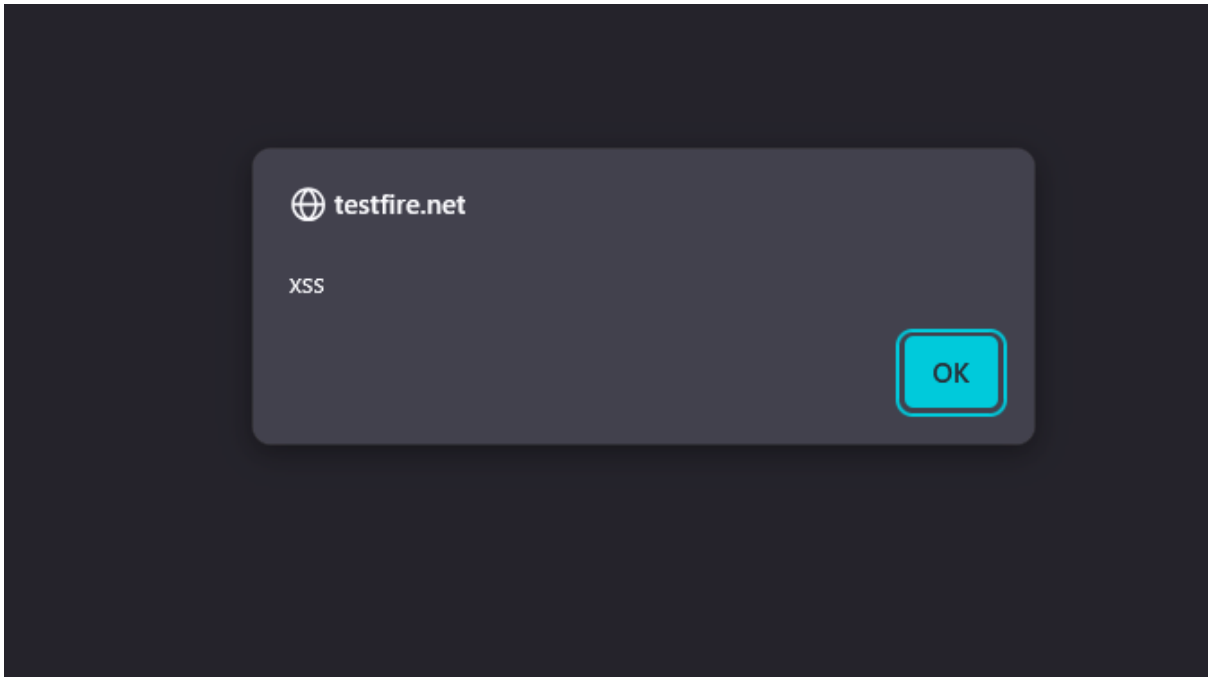
- **Session Hijacking/Account Takeover:** The attacker can steal the victim's **session cookies** (which prove they are logged in) and use them to hijack the victim's account.
- **Data Theft:** Attackers can use JavaScript to read sensitive information displayed on the page (e.g., credit card numbers, personal data) and send it to their own server.
- **Malicious Redirection:** The attacker can secretly redirect the victim to a malicious website (e.g., a phishing page) to trick them into entering credentials.
- **Defacement and Reputation Damage:** The attacker can alter the appearance of the compromised page to display inappropriate or malicious content, damaging the website's credibility and trust.
- **Keylogging:** In advanced scenarios, an attacker can use JavaScript to record all the victim's keystrokes while they are on the compromised page.

Script:

```
<script>alert(“xss”)</script>
```

Input field:

Pop up:


testfire.net
XSS
OK

5. Cross-Site Scripting (XSS).

a. Severity: Medium to High.

It is generally considered High because an attacker can completely compromise the integrity and confidentiality of the victim's session.

b. Impact:

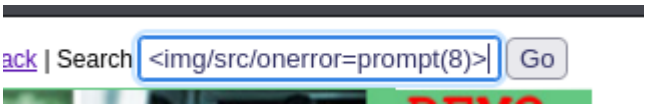
- **Session Hijacking/Account Takeover:** The attacker can steal the victim's **session cookies** (which prove they are logged in) and use them to hijack the victim's account.
- **Data Theft:** Attackers can use JavaScript to read sensitive information displayed on the page (e.g., credit card numbers, personal data) and send it to their own server.
- **Malicious Redirection:** The attacker can secretly redirect the victim to a malicious website (e.g., a phishing page) to trick them into entering credentials.
- **Defacement and Reputation Damage:** The attacker can alter the appearance of the compromised page to display inappropriate or malicious content, damaging the website's credibility and trust.
- **Keylogging:** In advanced scenarios, an attacker can use JavaScript to record all the victim's keystrokes while they are on the compromised page.

Script:

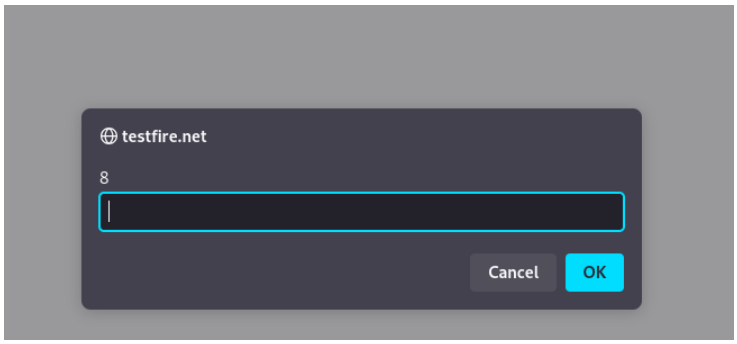
```
<img/src/onerror=prompt(8)>
```

Location: rest api .

Input field:



Pop up:



6. Cross-Site Scripting (XSS).

a. Severity: Medium to High.

It is generally considered High because an attacker can completely compromise the integrity and confidentiality of the victim's session.

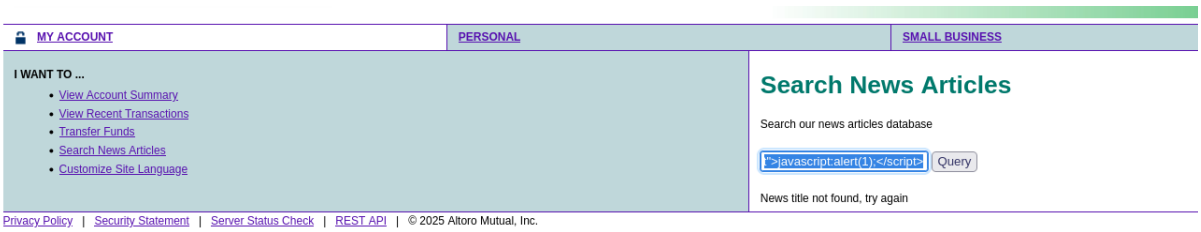
b. Impact:

- **Session Hijacking/Account Takeover:** The attacker can steal the victim's **session cookies** (which prove they are logged in) and use them to hijack the victim's account.
- **Data Theft:** Attackers can use JavaScript to read sensitive information displayed on the page (e.g., credit card numbers, personal data) and send it to their own server.
- **Malicious Redirection:** The attacker can secretly redirect the victim to a malicious website (e.g., a phishing page) to trick them into entering credentials.
- **Defacement and Reputation Damage:** The attacker can alter the appearance of the compromised page to display inappropriate or malicious content, damaging the website's credibility and trust.
- **Keylogging:** In advanced scenarios, an attacker can use JavaScript to record all the victim's keystrokes while they are on the compromised page.

Script:

```
<script\x20type="text/javascript">javascript:alert(1);</script>
```

Location: http://testfire.net/bank/queryxpath.jsp?content=queryxpath.jsp



7. Cross-Site Scripting (XSS).

a. Severity: Medium to High.

It is generally considered High because an attacker can completely compromise the integrity and confidentiality of the victim's session.

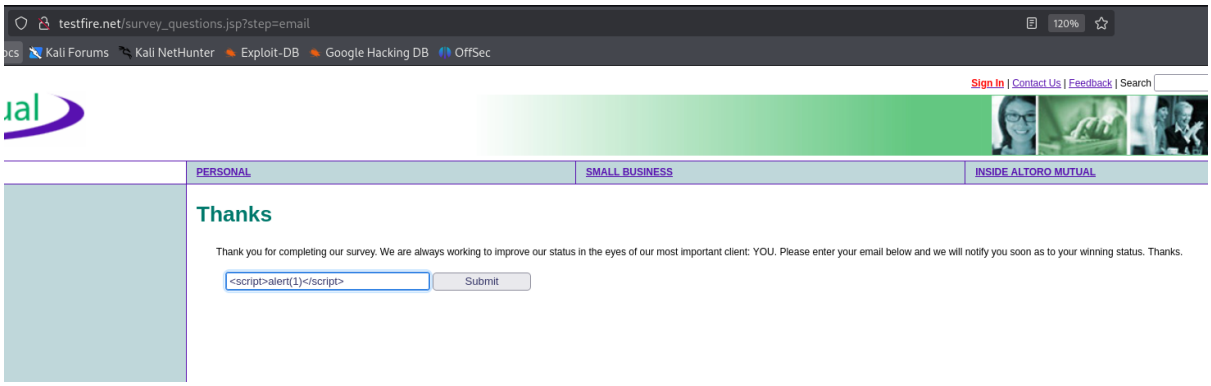
b. Impact:

- **Session Hijacking/Account Takeover:** The attacker can steal the victim's **session cookies** (which prove they are logged in) and use them to hijack the victim's account.
- **Data Theft:** Attackers can use JavaScript to read sensitive information displayed on the page (e.g., credit card numbers, personal data) and send it to their own server.
- **Malicious Redirection:** The attacker can secretly redirect the victim to a malicious website (e.g., a phishing page) to trick them into entering credentials.
- **Defacement and Reputation Damage:** The attacker can alter the appearance of the compromised page to display inappropriate or malicious content, damaging the website's credibility and trust.
- **Keylogging:** In advanced scenarios, an attacker can use JavaScript to record all the victim's keystrokes while they are on the compromised page.

Script:

```
<script>alert(1)</script>
```

Location: http://testfire.net/survey_questions.jsp?step=email

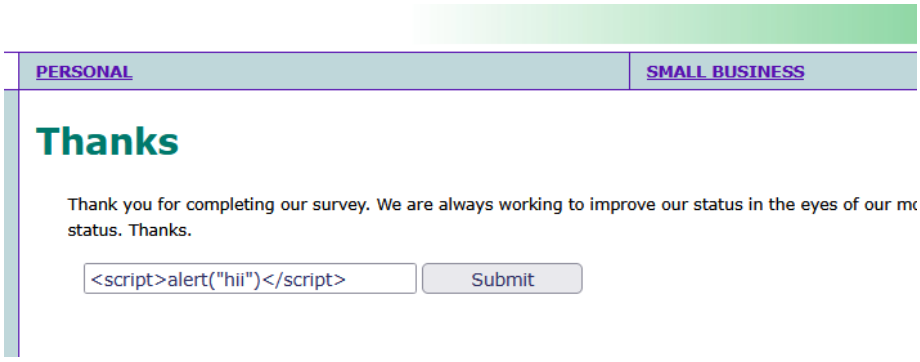


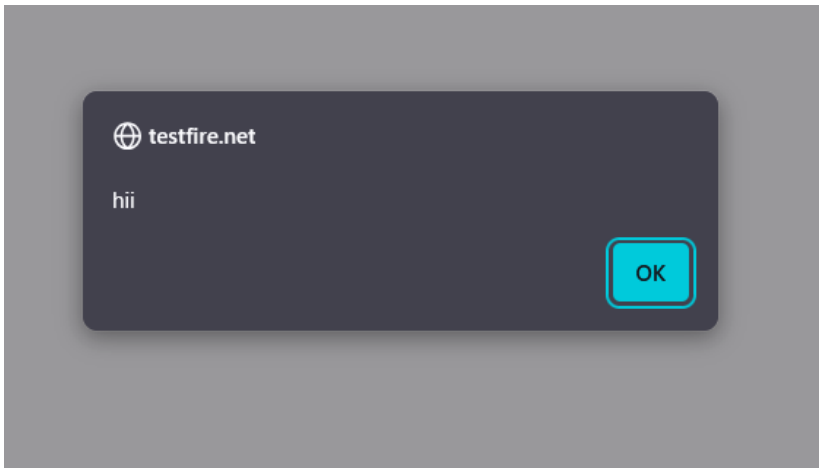
8. Cross-Site Scripting (XSS).

URL: http://testfire.net/survey_questions.jsp?step=email

Script:

```
<script>alert(“hi”)</script>
```





9. **Improper Neutralization of Input during Web Page Generation** (often categorized as a **Missing Function Level Access Control** or **Improper Input Validation** that leads to **Email Spoofing**).

   The core technical failure is that the server-side logic accepts and uses a sender email address provided by the client (you) without verifying if that email address actually belongs to the authenticated user or is even a legitimate address.

   a. **Severity: High**.

      While it doesn't grant direct access to customer accounts, it allows the bank's own system to be weaponized for external attacks, severely compromising trust.

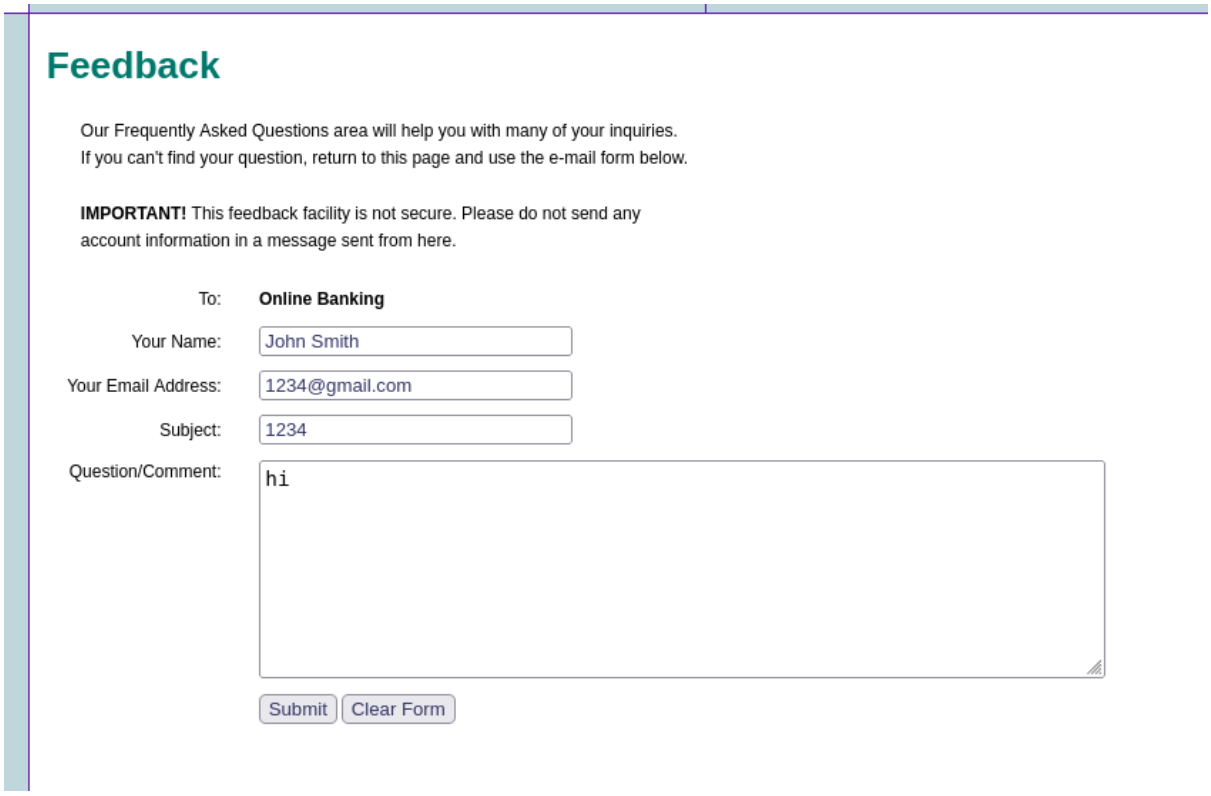
   b. **Impact:**

      Phishing & Fraud: An attacker can send convincing, malicious emails to *any* bank customer or employee, making it appear as though the email originated from a trusted source (e.g., *security@bank.com* or *ceo@bank.com*). Because the email appears to come directly from the bank's system, it bypasses many spam filters and carries high credibility.

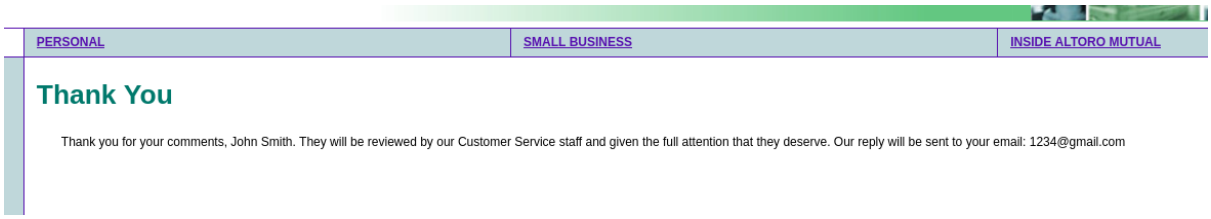
      - **Reputation Damage:** The bank's email infrastructure could be blacklisted by email providers if it's used to send spam or phishing emails, damaging the bank's reputation and impairing legitimate communication.
      - **Impersonation:** The attacker can impersonate bank officials to commit fraud, gather sensitive information, or trick recipients into clicking malicious links.

   Address: http://testfire.net/feedback.jsp

   Email is not verifying for sender.



Here I can change the senders email also.



In feedback I can change to any sender email and any emailer name .

10.SQL Injection (SQLi).

This vulnerability occurs when user-supplied data is concatenated into a SQL query without proper sanitization or handling, allowing an attacker to modify the query's logic.

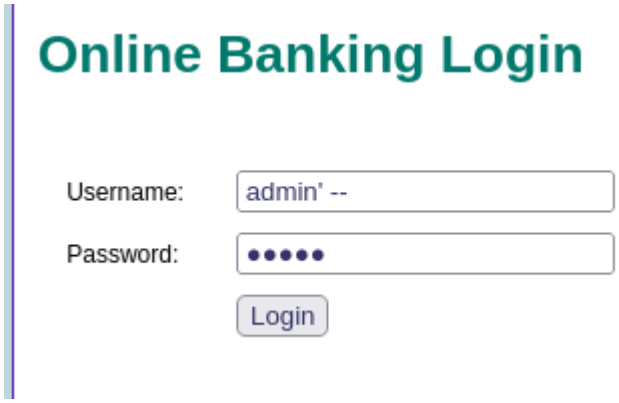
a. Severity: Critical.

This is the highest level of severity. SQLi often allows for complete and immediate compromise of the application's database, which is the repository for all sensitive customer and financial data.

b. Impact:

- **Total Data Compromise (Confidentiality):** The attacker can read, dump, and exfiltrate the entire contents of the database. This includes:
  - Customer Personally Identifiable Information (PII) like names, addresses, phone numbers.
  - Financial data, transaction records, and potentially even partial or full account numbers.
  - Usernames, password hashes, and security tokens for all users, including administrators.
- **Data Integrity Violation:** The attacker can modify or delete data, such as changing transaction records, altering balances, or destroying critical system configurations.
- **Authentication Bypass:** The attacker can bypass the login mechanism to log in as any user, including an administrator, gaining full administrative control over the application.
- **Denial of Service (DoS):** An attacker could potentially crash or lock the database, causing the banking website to become unavailable.

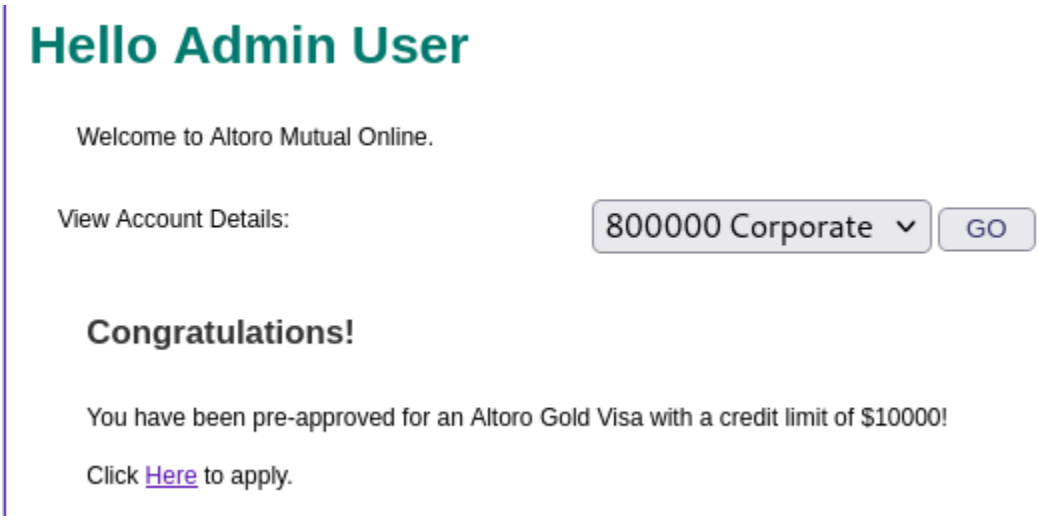
Sql injection on login page:



Used payload:

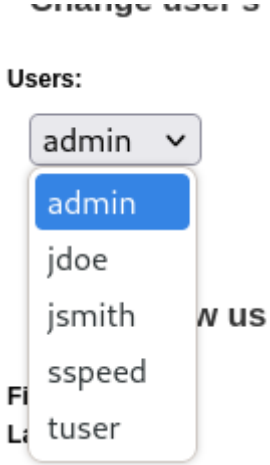
Username:admin’ –

Password:any characters

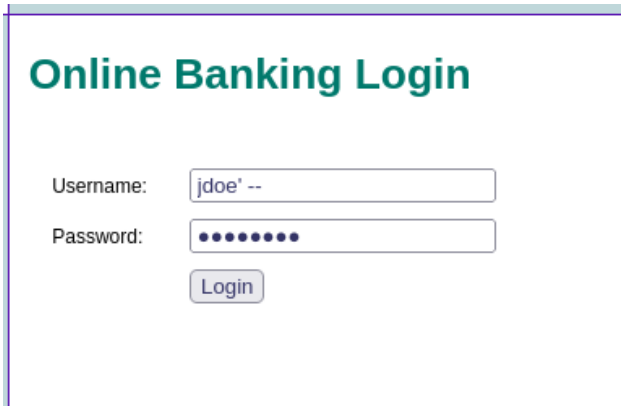


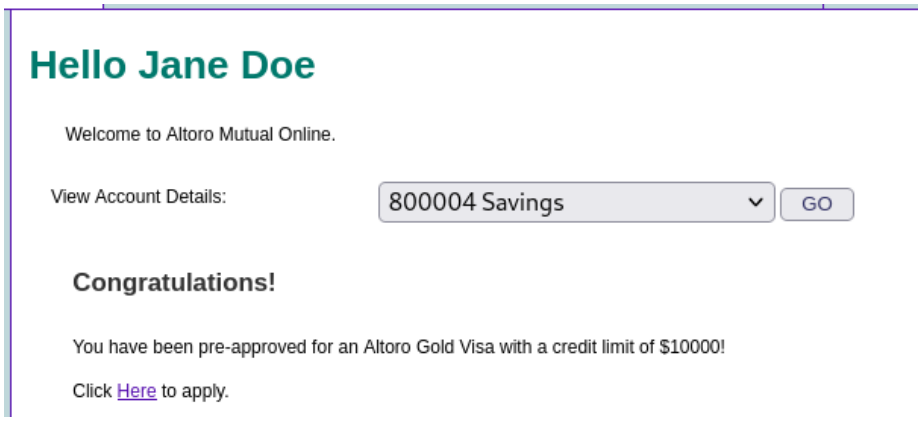
Logged in as admin

Got all users on the website:

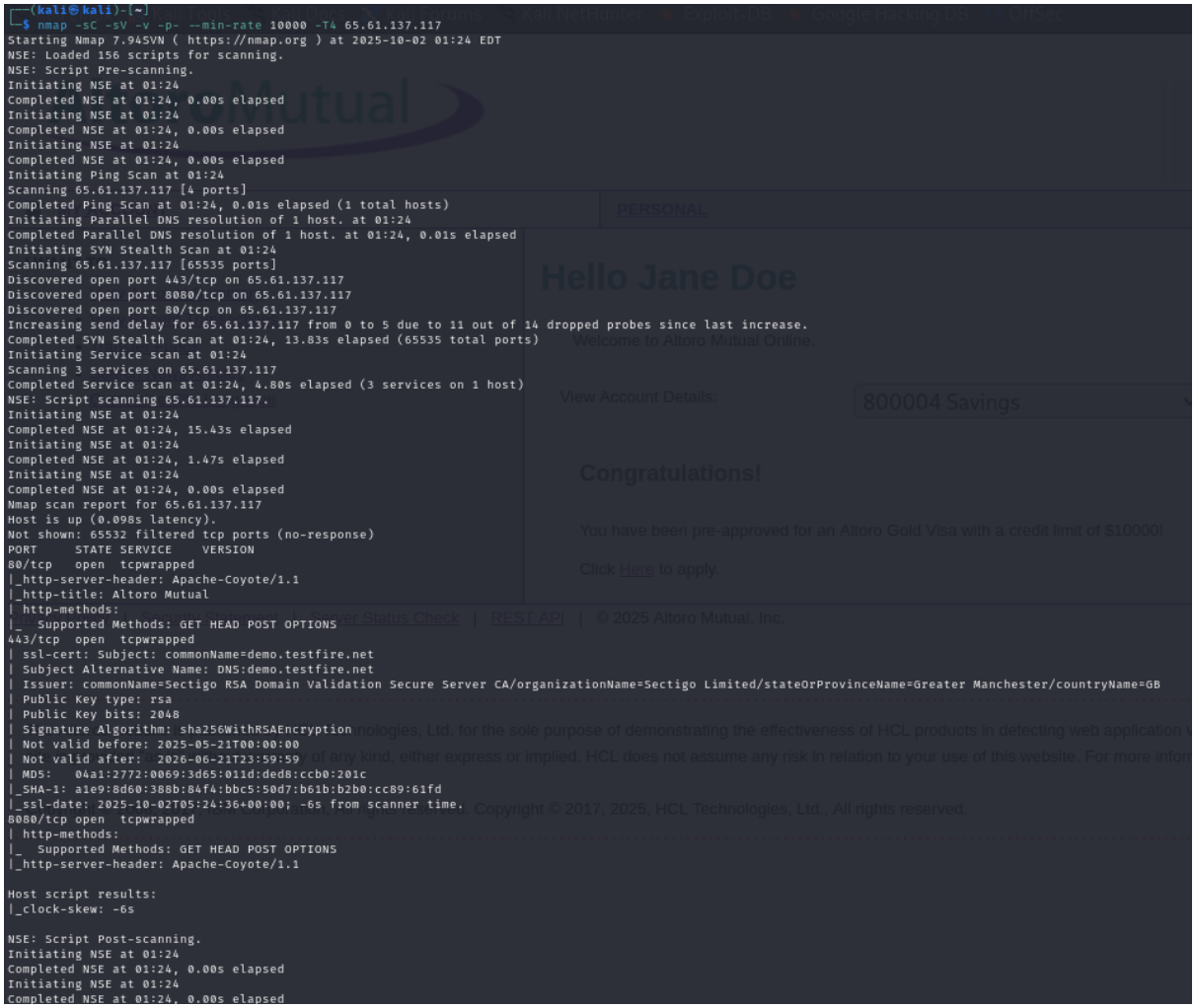


Also tried sql injection for other users:




Hello Jane Doe
Welcome to Altoro Mutual Online.
View Account Details:
800004 Savings
GO
Congratulations!
You have been pre-approved for an Altoro Gold Visa with a credit limit of $10000!
Click Here to apply.

11.Nmap scan for the target.

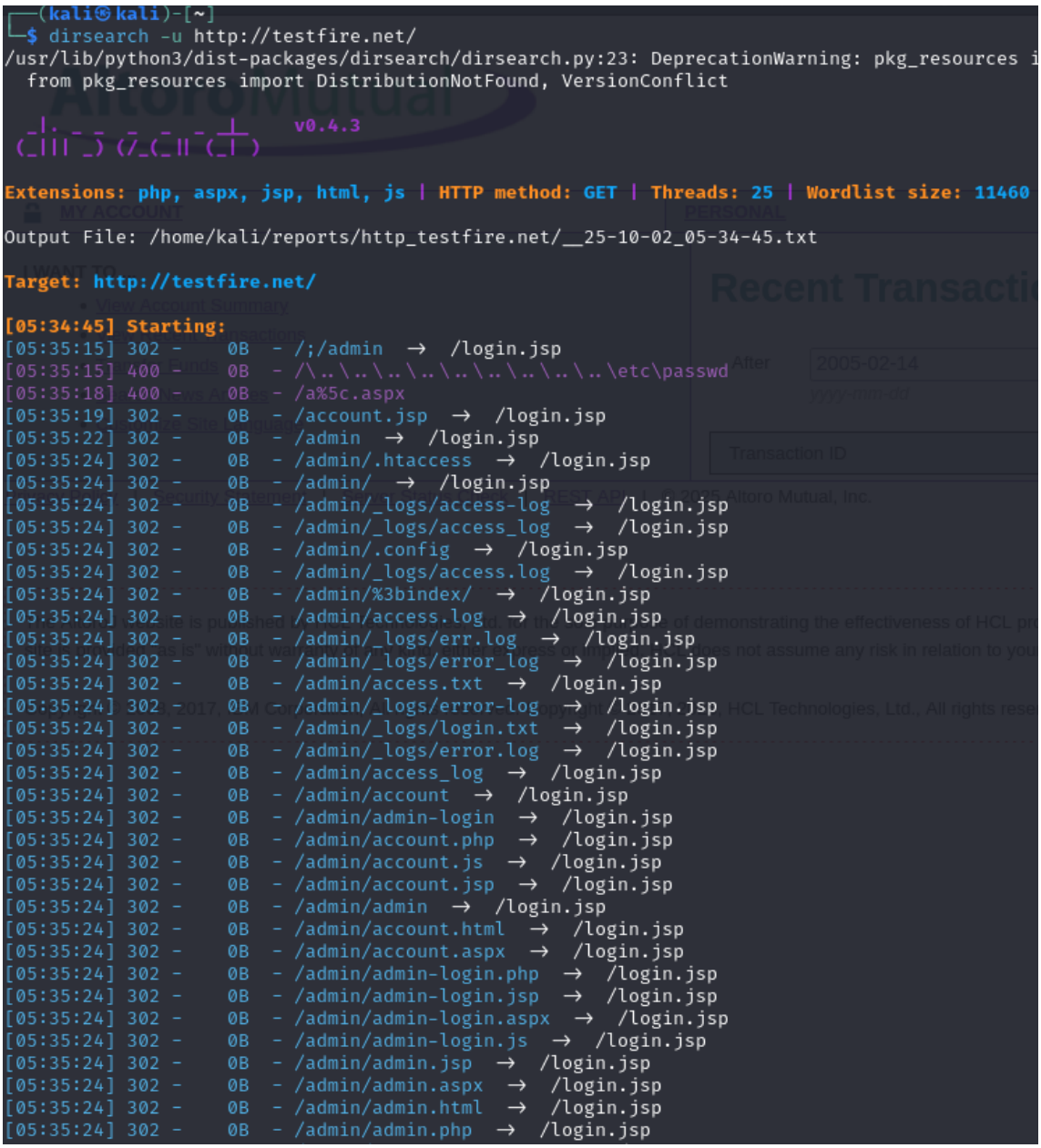


12.Directory enumeration on target

Using dirsearch tool in terminal.

Dirsearch tool command for enumeratrion:

dirsearch -u http:// testfire.net/



```
[05:35:26] 302 -    0B  - /admin/login.js  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/manage.asp  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/mysql/index.php  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/phpMyAdmin  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/mysql2/index.php  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/logs/error_log  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/manage/admin.asp  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/logs/errors.log  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/mysql/  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/phpmyadmin/  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/logs/access.log  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/logs/access-log  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/phpmyadmin/index.php  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/manage  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/phpMyAdmin/index.php  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/phpmyadmin2/index.php  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/pMA/  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/pma/index.php  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/tiny_mce  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/tinymce  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/upload.php  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/signin  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/private/logs  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/secure/logon.jsp  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/portalcollect.php?f=http://xxx&t=js  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/uploads.php  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/phpMyAdmin/  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/views/ajax/autocomplete/user/a  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/user_count.txt  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/web/  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/PMA/index.php  →  /login.jsp
[05:35:26] 302 -    0B  - /admin/pma/  →  /login.jsp
[05:35:27] 302 -    0B  - /admin/scripts/fckeditor  →  /login.jsp
[05:35:27] 302 -    0B  - /admin/release  →  /login.jsp
[05:35:27] 302 -    0B  - /admin/sysadmin/  →  /login.jsp
[05:35:27] 302 -    0B  - /admin/sqladmin/  →  /login.jsp
[05:35:27] 302 -    0B  - /admin/sxd/  →  /login.jsp
[05:35:27] 302 -    0B  - /admin/pol_log.txt  →  /login.jsp
[05:35:27] 302 -    0B  - /admin;/  →  /login.jsp
[05:35:45] 401 -   35B  - /api/application.wadl
[05:36:16] 200 -    8KB - /feedback.jsp
[05:36:27] 302 -    0B  - /images  →  /images/
[05:36:39] 200 -    8KB - /login.jsp
[05:36:41] 302 -    0B  - /logout.jsp  →  index.jsp
[05:37:14] 302 -    0B  - /pr  →  /pr/
[05:37:24] 200 -    7KB - /search.jsp
[05:37:38] 302 -    0B  - /static  →  /static/
[05:37:42] 302 -    0B  - /swagger  →  /swagger/
[05:37:42] 200 -    1KB - /swagger/index.html
```

13.Insecure Direct Object Reference (IDOR).

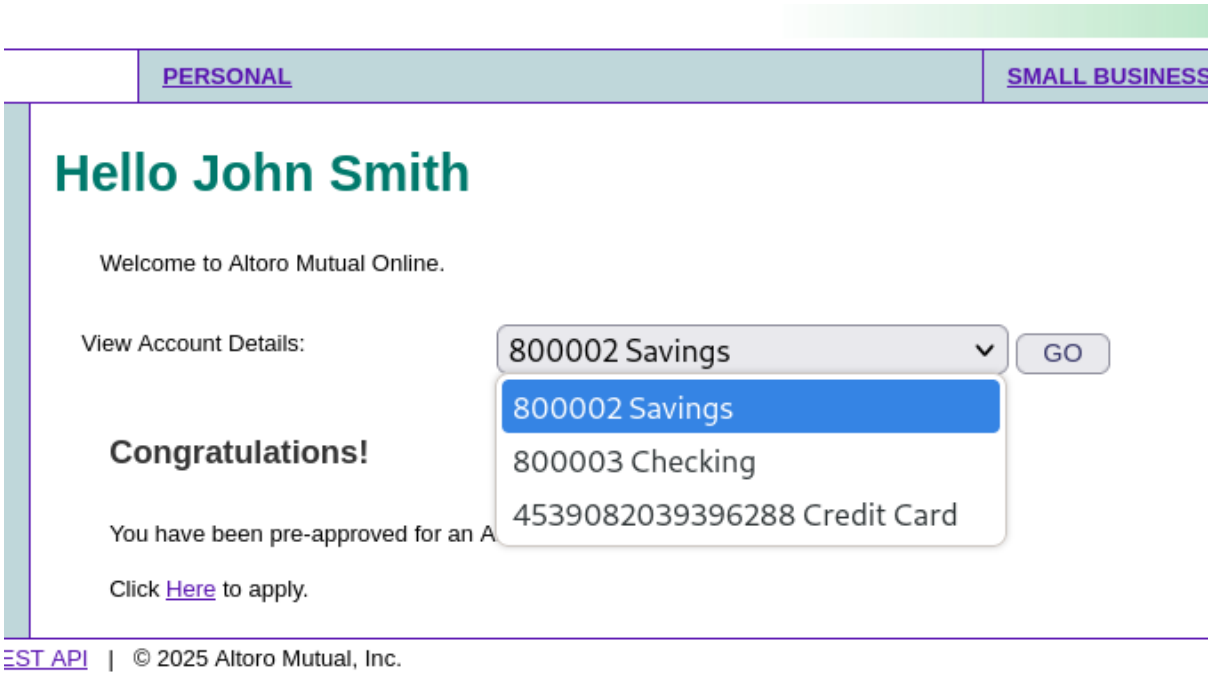
This occurs when an application exposes a direct reference to an internal implementation object (like an account number or user ID) and does not perform sufficient **authorization checks** to confirm that the *currently logged-in user* is allowed to access or modify that specific object.

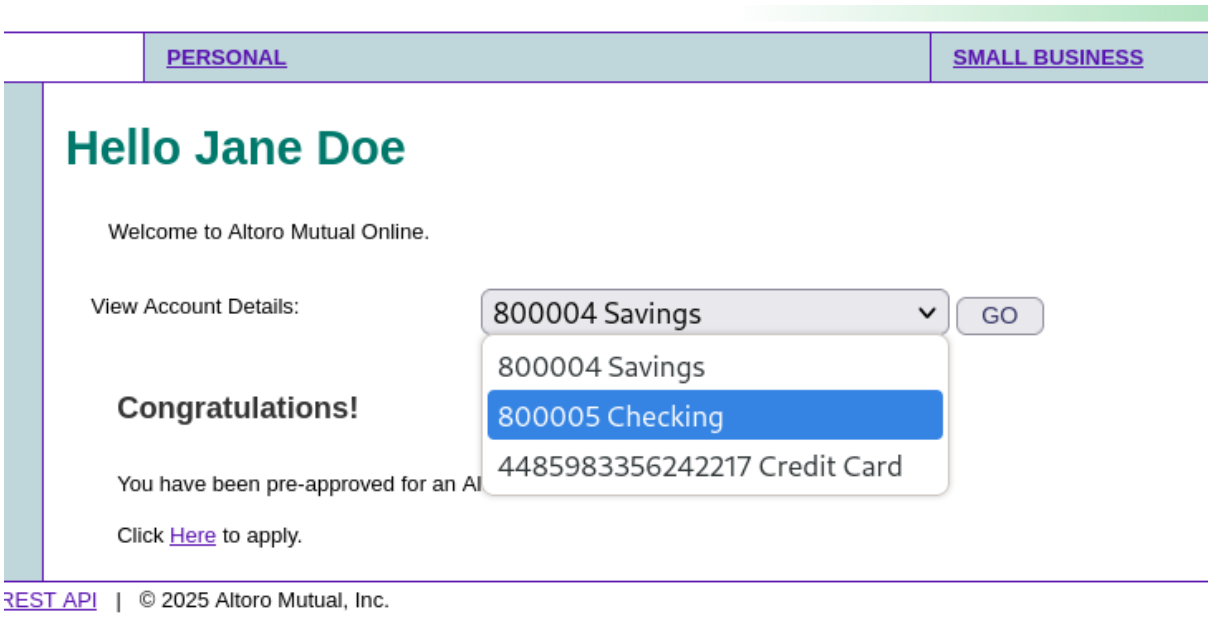
- **Severity: Critical.**
  This is the highest level of severity. In a banking context, an IDOR that allows fund transfers is equivalent to having a universal key to move money between accounts, leading to massive and immediate financial fraud.

- **Impact:**

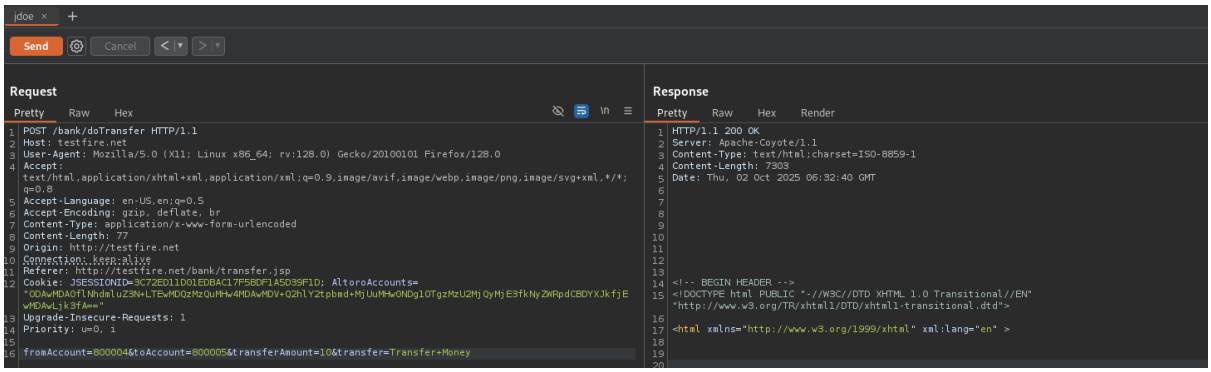
  - **Financial Theft and Fraud (Confidentiality & Integrity):** The attacker can transfer funds from *any* other customer's account to their own, leading to widespread financial loss for customers and the bank.
  - **Total Data Compromise (Confidentiality):** If the IDOR applies to other objects, the attacker can view or download any other customer's personal data, transaction history, and account balances.
  - **Regulatory Fines & Lawsuits:** The bank will face massive regulatory fines (e.g., for non-compliance with data protection laws) and certain lawsuits from affected customers.
  - **Catastrophic Reputation Damage:** News of this flaw would instantly destroy public trust and potentially lead to a bank run or mass account closures.

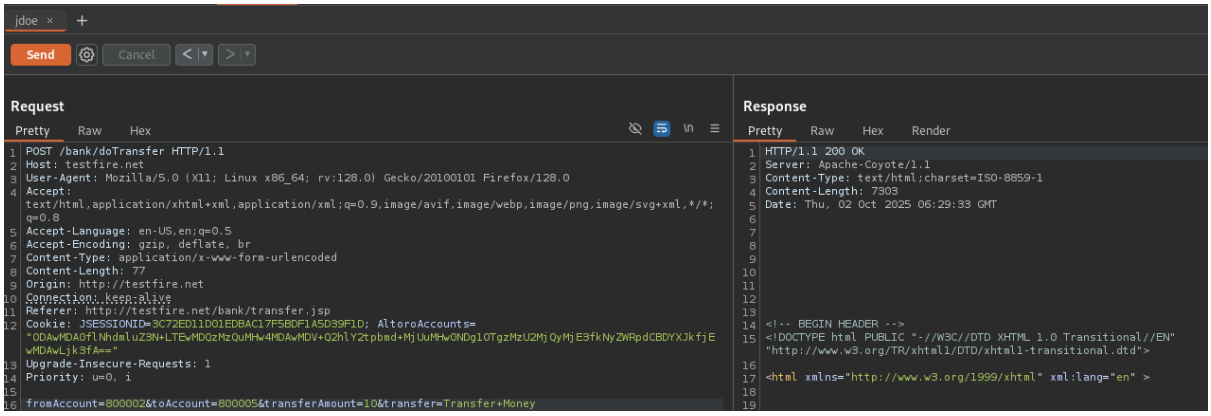
Here I can transfer any unknown user’s account money to my account or to any account .So this is very critical vulnerability present here.


PERSONAL
SMALL BUSINESS
Hello John Smith
Welcome to Altoro Mutual Online.
View Account Details:
800002 Savings
GO
800002 Savings
800003 Checking
4539082039396288 Credit Card
Congratulations!
You have been pre-approved for an A
Click Here to apply.
EST API | © 2025 Altoro Mutual, Inc.


PERSONAL
SMALL BUSINESS
Hello Jane Doe
Welcome to Altoro Mutual Online.
View Account Details:
800004 Savings
GO
800004 Savings
800005 Checking
4485983356242217 Credit Card
Congratulations!
You have been pre-approved for an Al
Click Here to apply.
REST API | © 2025 Altoro Mutual, Inc.

This intercepted request is of jdoe transfering his 10 rs from ac 800004 to 800005 but I changed the from ac to 800002 .800002 ac is of another user named john smith.so jdoe used to get 10 rs using IDOR from john smith account in unauthorized way.





Prepared by: Shivam Chilkewar | Red Team & VAPT Security Researcher

Email: chilkewarshivam@gmail.com

Phone: +91 94206 14292

LinkedIn: linkedin.com/in/shivamchilkewar

Vulnerability Assessment & Penetration Testing (VAPT) Report – Testfire Bank