

VAPT REPORT

**Altoro Mutual
(Dummy Website)**

Date: 27th May 2025

**Prepared by
Aniket Tiwari**

Table of Contents

Web Application Penetration Testing.....	1
1.1 Summary of Findings.....	3
Vulnerability Details	4
2.1 Sql injection.....	4
2.2 Cross site scripting.....	7
2.3 IDOR.....	9
2.4 Brute force.....	13
2.5 Clear-text submission of password.....	18
2.6 Strict transport security not enforced	20
2.7 Email Address Exposed	21

1.1 Summary of Findings

The Web Vulnerability Assessment and Penetration Testing (VAPT) conducted for the target organization yielded important findings and insights. This summary provides an overview of the key results obtained during the assessment.

It was observed that the application was exposed to a total of **10 security vulnerabilities** during the given assessment tenure with 1 as Critical, 1 as High 2 as Medium, C 6 as Low Severity vulnerabilities.

S. No.	Name	Severity
1.	SQL injection	Critical
2.	Cross site scripting (XSS)	High
3.	IDOR (Insecure Direct Object Reference)	High
4.	Login brute force	High
5.	Clear-text submission of password	High
6.	Strict transport security not enforced	low
7.	Email Address Exposed	low

Vulnerability Details

In this section we will give details of the observed vulnerabilities in the website.

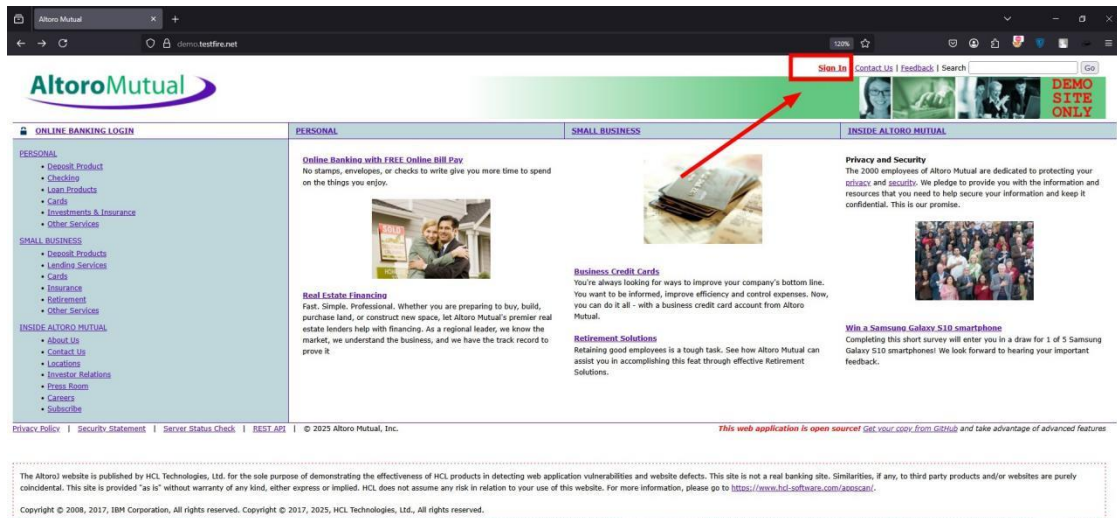
2.1 SQL injection in login form

Parameter	Description
Severity	Critical
Description	An SQL Injection vulnerability was identified in the login form of the application. The input fields fail to properly validate or sanitize user input, allowing direct manipulation of the underlying SQL query used for authentication.
Vulnerable parameter	
Affected URL	https://demo.testfire.net/search.jsp
Threat	<ul style="list-style-type: none">• Full Admin Access: The attacker can access the admin dashboard and all sensitive data.• Data Breach: Potential exposure of user data, internal records, and application settings.• Privilege Escalation: Unauthorized actions like user management, configuration changes, and more.• Compliance Risk: Violates data protection laws like GDPR, potentially resulting in legal penalties.
Tools used	Manual

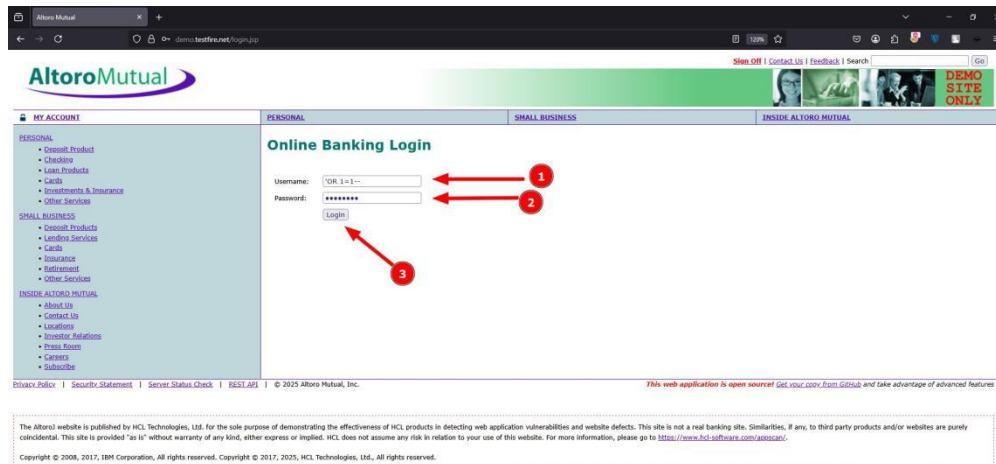
SQL Injection is a web vulnerability where attackers insert malicious SQL code into input fields to manipulate the database. It can allow unauthorized access, data theft, or even full control of the database. For example, logging in without a password by tricking the system with special input.

Steps to Reproduce

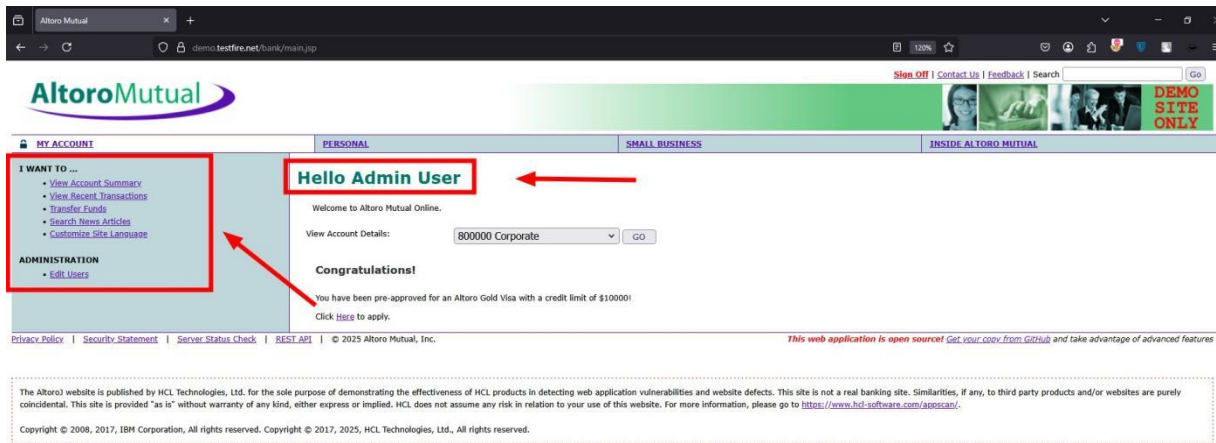
1. Access the URL, click on sign in button <https://demo.testfire.net>



2. Enter this payload **OR 1=1--** in the username, and random password. click login



3. Observe you logged in an admin account with all privileges and permissions of admin



Recommended Remediation:

1. Use Parameterized Queries / Prepared Statements:

- Avoid directly including user input in SQL queries.
- Use secure database APIs that separate data from code.

2. Input Validation:

- Allow only expected characters or formats.
- Block or escape dangerous inputs.

3. Implement Web Application Firewall (WAF):

- Add another layer of defense by detecting and blocking suspicious requests.

4. Regular Security Testing:

- Perform code reviews, vulnerability scans, and penetration tests.

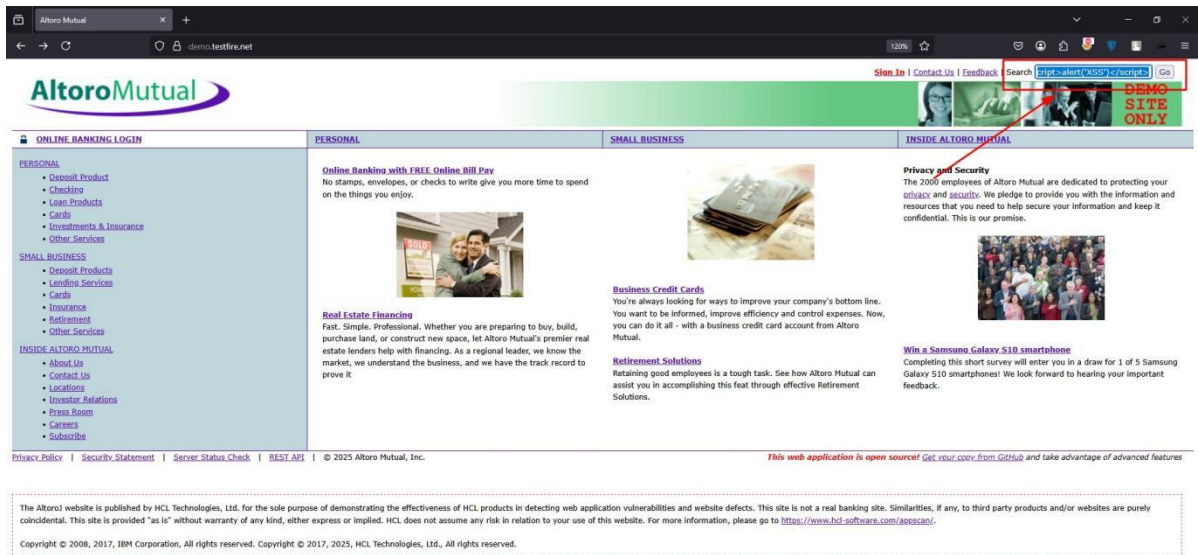
2.2 Cross site scripting (Reflected XSS)

Parameter	Description
Severity	High
Description	A reflected Cross-Site Scripting (XSS) vulnerability was identified in the search functionality of the web application
Vulnerable parameter	query
Affected URL	https://demo.testfire.net/search.jsp
Threat	<p>An attacker can exploit this vulnerability by tricking users into clicking a malicious link. Potential impacts include:</p> <ul style="list-style-type: none"> • Session Hijacking: If the victim is logged in, the attacker may steal session cookies. • Credential Theft: Fake login forms can be injected to steal usernames and passwords. • Phishing Attacks: Users may be redirected or shown fake content to trick them into revealing sensitive information. • Browser Exploits: Malicious scripts could load additional payloads to exploit browser or plugin vulnerabilities. • Reputation Damage: Trust in the application may be reduced if users are exploited via this vulnerability.
Tools used	Manual

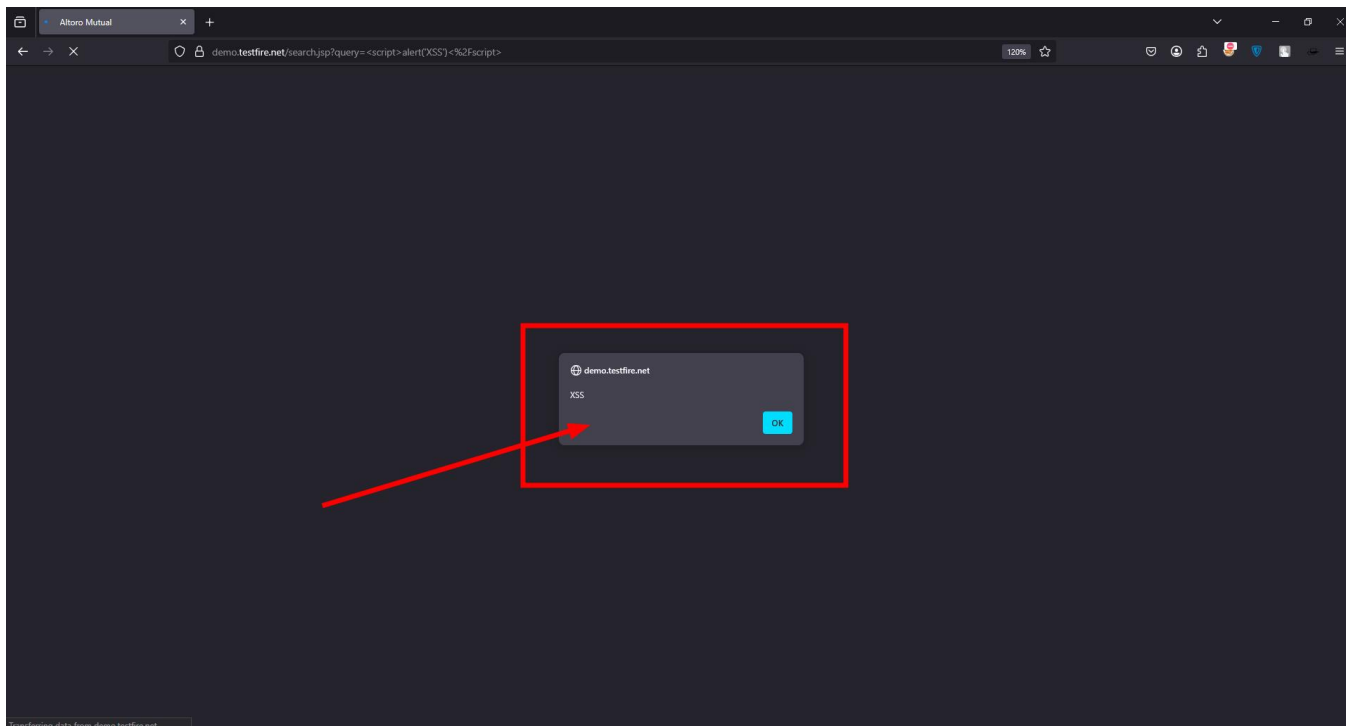
Cross-Site Scripting (XSS) is a web vulnerability where attackers inject malicious scripts into websites. When other users visit the page or click a crafted link, the script runs in their browser. This can lead to stolen data, fake pages, or actions done without the user's consent.

Steps to Reproduce

1. Access given URL <https://demo.testfire.net>
2. On the top right corner: paste the payload in search bar and click go `<script>alert('XSS')</script>`



3. Observe an alert box is triggered, demonstrating successful JavaScript execution.



Recommended Remediation:

Input Validation & Output Encoding:

- Sanitize all user inputs.
- Encode output to neutralize special characters (e.g., `<`, `>`, `"`, `'`) before rendering it in the browser.
- Use frameworks or libraries that auto-escape HTML (e.g., JSTL `<c:out>` in JSP).

Content Security Policy (CSP):

- Implement CSP headers to restrict the execution of inline scripts and mitigate XSS impact.

Use Secure Development Practices:

- Avoid directly reflecting unsanitized user input in the HTML response.
- Conduct regular security testing and code reviews.

2.3 Insecure Direct object Reference (IDOR)

Parameter	Description
Severity	High
Description	An IDOR vulnerability was identified in the account details page of the application. The system exposes sensitive bank account information based solely on an <code>account number</code> passed in the URL, without properly verifying the authenticated user's authorization.
Vulnerable parameter	listAccounts
Affected URL	<code>https://demo.testfire.net/bank/showAccount?listAccounts=800002</code>

Threat	<p>This allows any logged-in user to access other users' account details simply by changing the account number in the URL.</p> <ul style="list-style-type: none"> • Data Exposure: Unauthorized access to sensitive personal and financial data. • Privacy Violation: Direct breach of user confidentiality. • Regulatory Risk: Non-compliance with data protection laws such as GDPR and PCI-DSS. • Potential for Abuse: Attackers can loop through account numbers and scrape user data at scale.
Tools used	Manual

IDOR (Insecure Direct Object Reference) is a vulnerability where attackers access unauthorized data by modifying a reference (like an ID) in the URL or request. If the app doesn't properly check permissions, anyone can view or manipulate other users' data. For example, changing `account=123` to `account=124` may expose someone else's information.

Steps to Reproduce

1. Access given URL <https://demo.testfire.net/login.jsp>
2. login with credentials jsmith:demo1234

Altoro Mutual

Sign In | Contact Us | Feedback | Search

ONLINE BANKING LOGIN

PERSONAL

- Deposit Product
- Checking
- Loan Products
- Cards
- Investments & Insurance
- Other Services

SMALL BUSINESS

- Deposit Products
- Lending Services
- Cards
- Insurance
- Retirement
- Other Services

INSIDE ALTORO MUTUAL

- About Us
- Contact Us
- Locations
- Investor Relations
- Press Room
- Careers
- Subscribe

Online Banking Login

Username:

Password:

Login

Privacy Policy | Security Statement | Server Status Check | REST API | © 2025 Altoro Mutual, Inc.

This web application is open source! Get your copy from GitHub and take advantage of advanced features.

The Altoro website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/apscan/>.

Copyright © 2008, 2017, IBM Corporation, All rights reserved. Copyright © 2017, 2025, HCL Technologies, Ltd., All rights reserved.

3. click on Go to view your account details. Notice your account number is 800002

Altoro Mutual

Sign Out | Contact Us | Feedback | Search

MY ACCOUNT

PERSONAL

SMALL BUSINESS

INSIDE ALTORO MUTUAL

Hello John Smith

Welcome to Altoro Mutual Online.

View Account Details:

Congratulations!

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

Click [Here](#) to apply.

Privacy Policy | Security Statement | Server Status Check | REST API | © 2025 Altoro Mutual, Inc.

This web application is open source! Get your copy from GitHub and take advantage of advanced features.

The Altoro website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/apscan/>.

Copyright © 2008, 2017, IBM Corporation, All rights reserved. Copyright © 2017, 2025, HCL Technologies, Ltd., All rights reserved.

4. It shows only your account details.

AltoroMutual

MY ACCOUNT PERSONAL SMALL BUSINESS INSIDE ALTORO MUTUAL

I WANT TO ...

- View Account Summary
- View Recent Transactions
- Transfer Funds
- Search News Articles
- Customize Site Language

Account History - 800002 Savings

Balance Detail

800002 Savings	Select Account	Amount
Ending balance as of 4/5/25 11:16 PM		-\$619125889.58
Available balance		-\$619125889.58

10 Most Recent Transactions

Date	Description	Amount
2025-04-05	Withdrawal	-\$5.00
2025-04-05	Withdrawal	-\$5.00
2025-04-05	Deposit	\$500.00
2025-04-05	Withdrawal	-\$500.00
2025-04-05	Withdrawal	-\$500.00
2025-04-05	Deposit	\$1000.00

Credits

Account	Date	Description	Amount
1001160140	12/29/2004	Paycheck	1200
1001160140	01/12/2005	Paycheck	1200
1001160140	01/29/2005	Paycheck	1200
1001160140	02/12/2005	Paycheck	1200
1001160140	03/01/2005	Paycheck	1200
1001160140	03/15/2005	Paycheck	1200

Debits

Account	Date	Description	Amount
---------	------	-------------	--------

5. change 2 to 0 in the address bar to see other account details, and press enter.

AltoroMutual

MY ACCOUNT PERSONAL SMALL BUSINESS INSIDE ALTORO MUTUAL

I WANT TO ...

- View Account Summary
- View Recent Transactions
- Transfer Funds
- Search News Articles
- Customize Site Language

Account History - 800000 Savings

Balance Detail

800000 Savings	Select Account	Amount
Ending balance as of 4/5/25 11:16 PM		-\$619125889.58
Available balance		-\$619125889.58

10 Most Recent Transactions

Date	Description	Amount
2025-04-05	Withdrawal	-\$5.00
2025-04-05	Withdrawal	-\$5.00
2025-04-05	Deposit	\$500.00
2025-04-05	Withdrawal	-\$500.00
2025-04-05	Withdrawal	-\$500.00
2025-04-05	Deposit	\$1000.00

Credits

Account	Date	Description	Amount
1001160140	12/29/2004	Paycheck	1200
1001160140	01/12/2005	Paycheck	1200
1001160140	01/29/2005	Paycheck	1200
1001160140	02/12/2005	Paycheck	1200
1001160140	03/01/2005	Paycheck	1200
1001160140	03/15/2005	Paycheck	1200

Debits

Account	Date	Description	Amount
---------	------	-------------	--------

6. observe you get details of other user account.

[illegible]

Recommended Remediation:

1. Access Control Checks:

- 0 Ensure the backend verifies that the requesting user has permission to access the requested resource.

2. Use Indirect References:

- 0 Avoid exposing internal identifiers (e.g., account numbers) directly in URLs. Use mapped or tokenized values instead.

3. Audit and Logging:

- 0 Log access to sensitive endpoints and flag unusual access patterns.

4. Security Testing:

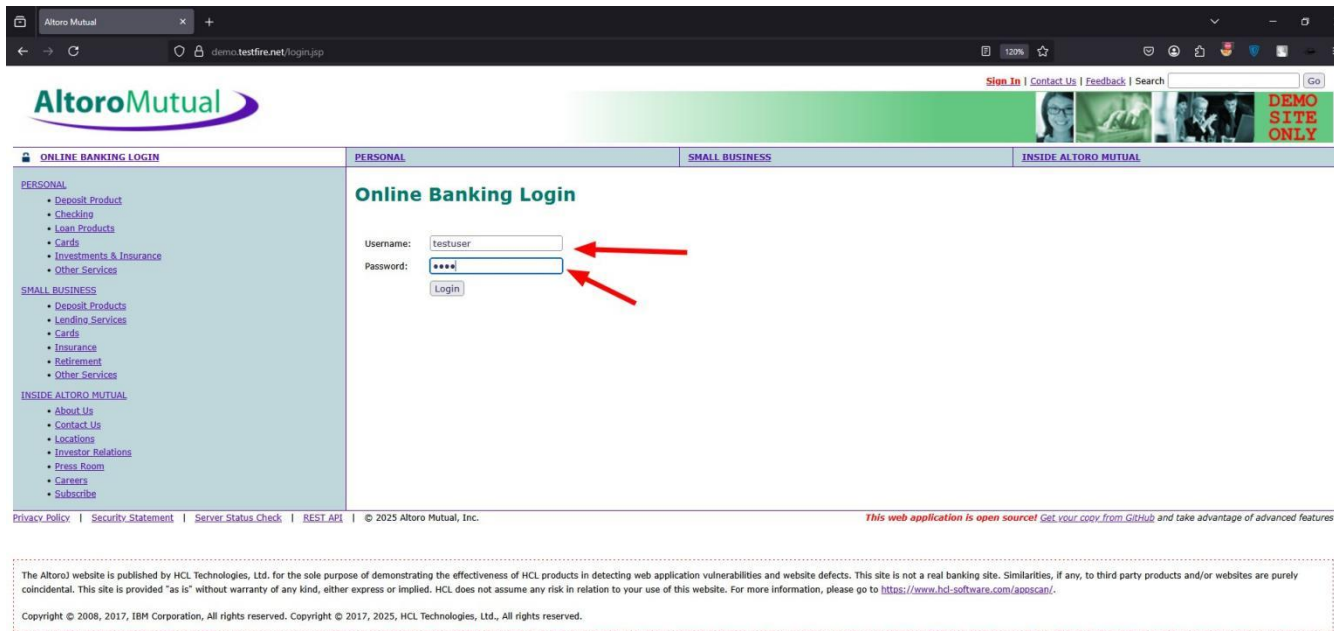
- 0 Perform authorization testing for all endpoints handling sensitive data.

2.4 Brute Force Login Vulnerability

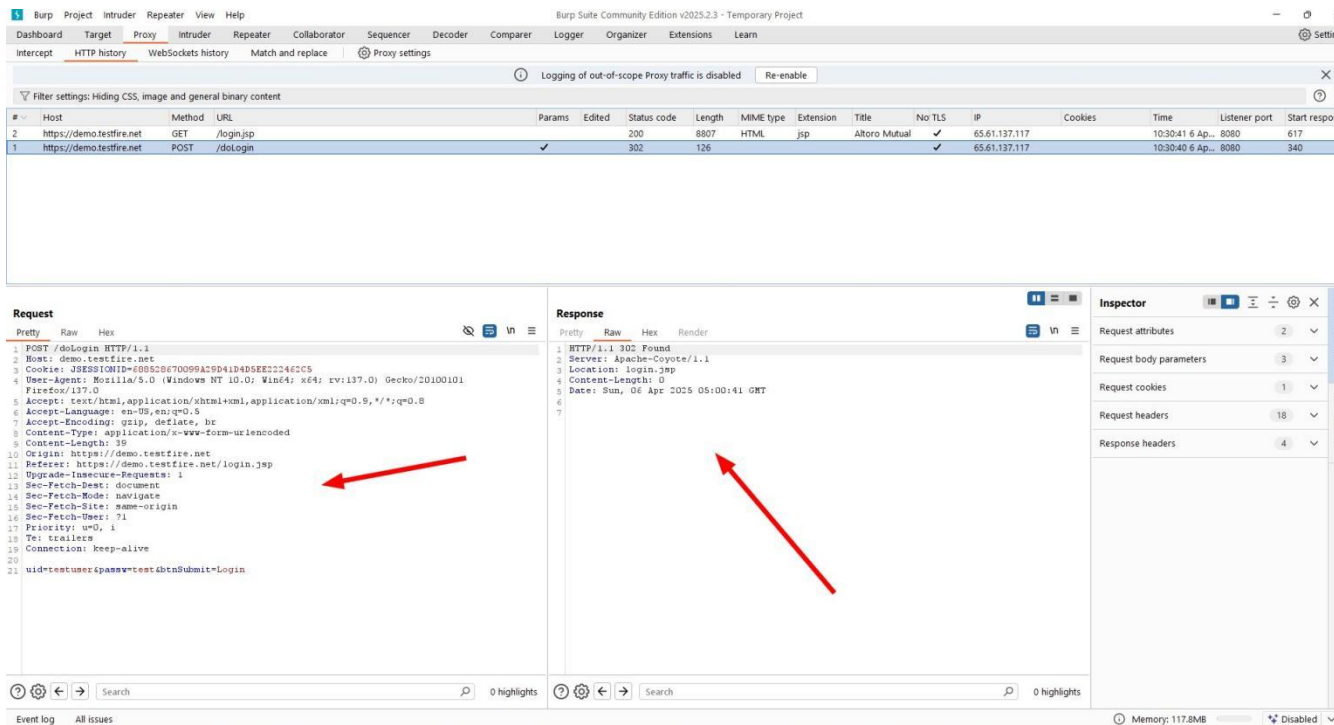
Parameter	Description
Severity	High
Description	The login functionality of the application is vulnerable to brute force attacks . This allows an attacker to automate password-guessing attempts and potentially gain unauthorized access to user accounts.
Vulnerable parameter	-
Affected URL	https://demo.testfire.net/login.jsp
Threat	<ul style="list-style-type: none"> • Unauthorized Access: Attacker can take over user accounts, including admin accounts. • Data Breach: Gaining access to sensitive user data and actions within the application. • Service Abuse: Can lead to impersonation, financial fraud, or internal system misuse. • Reputation Damage & Compliance Risk: May violate data protection regulations such as GDPR or PCI-DSS.
Tools used	Burp suite

Steps to Reproduce

1. Access the URL <https://demo.testfire.net/login.jsp>
2. fill a random credential in username and password.



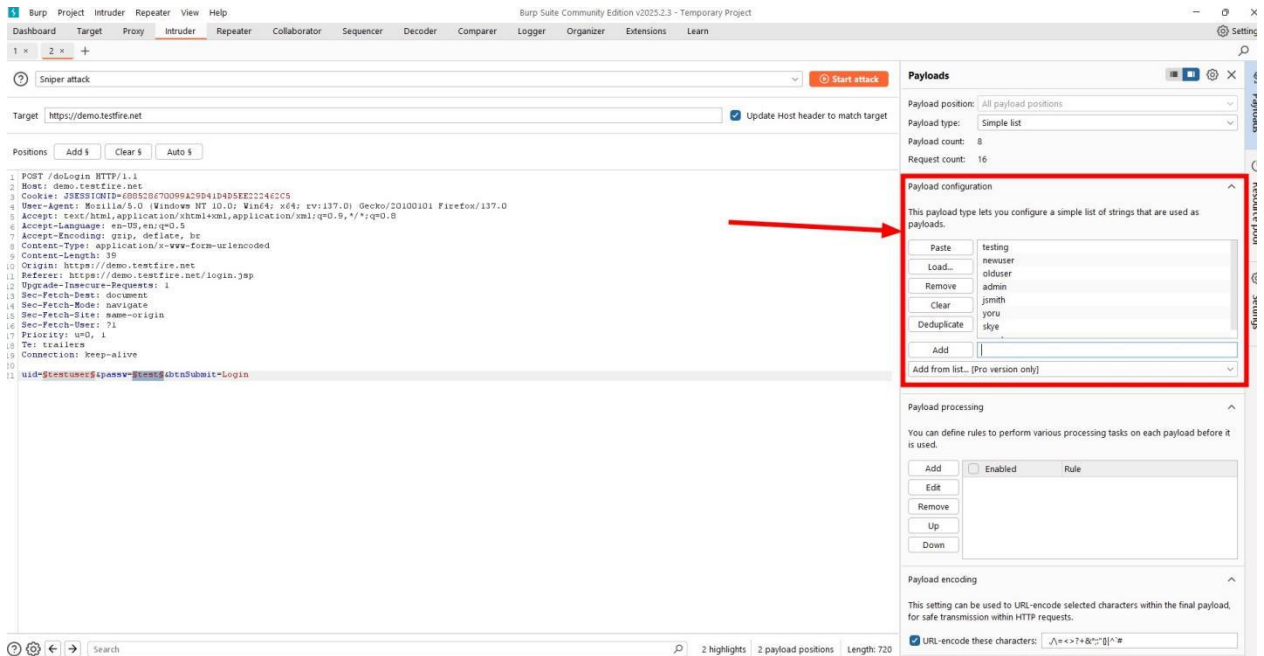
3. Open burp suite, proxy the login request
4. Your request will failed but check your burp suite for request details.



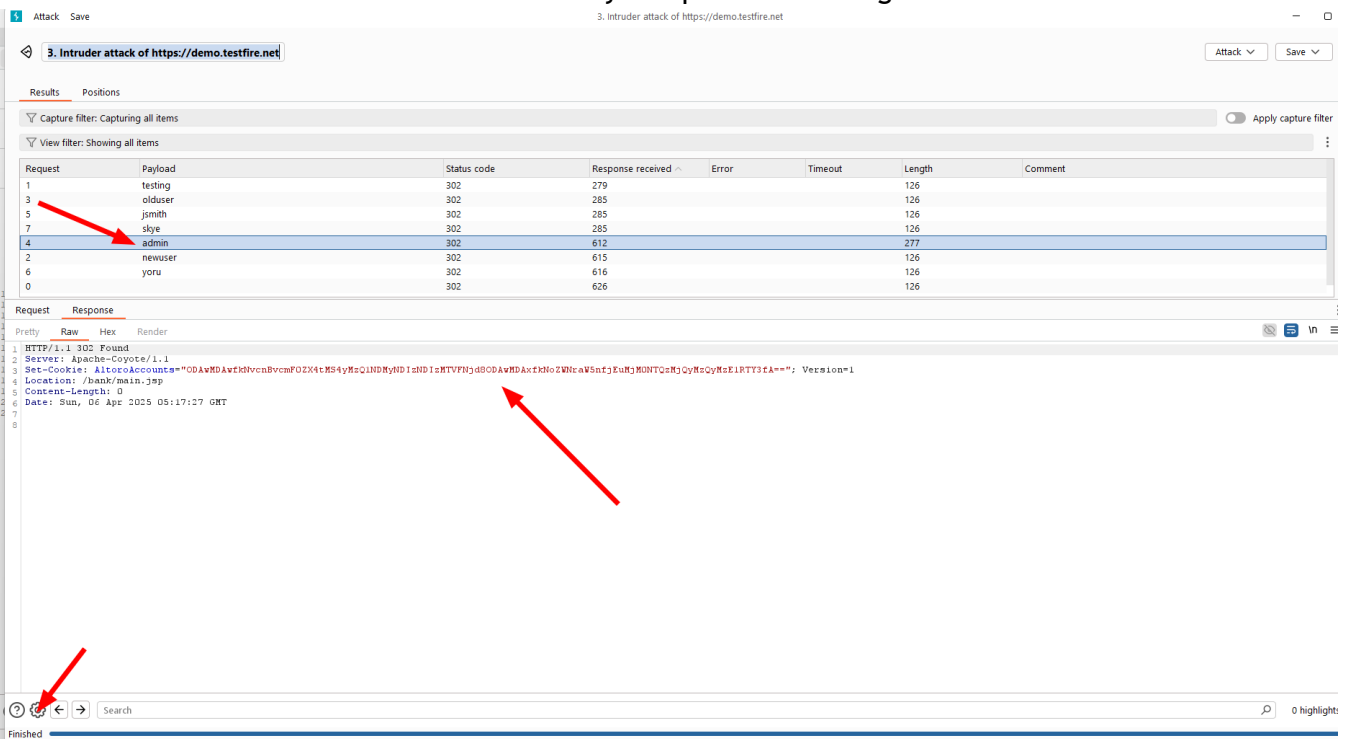
5. sent this request to intruder > by right click > send to intruder

6. Add both the uid and passw values as placeholder and select battering ram attack.

7. Add your list to brute force in payload configuration, and click on start attack.



8. observe the bruteforce is successfully completed and we got our credentials



Recommended Remediation:

1. **Rate Limiting:** Restrict the number of login attempts per IP or user.
2. **Account Lockout / Delay:** Temporarily lock or delay logins after several failed attempts.

3. **CAPTCHA:** Introduce CAPTCHA after a few failed login attempts.
4. **Monitoring & Alerts:** Log brute force attempts and notify admins of suspicious login behavior.

2.5 Clear-text submission of password

Parameter	Description
Severity	High
Description	<p>It was identified that the application transmits user credentials – specifically passwords – in cleartext over the network.</p> <p>In some environments, especially with weak or misconfigured SSL/TLS, attackers can intercept these credentials using tools like Wireshark, MITM proxies, or ARP spoofing techniques.</p>
Vulnerable parameter	-
Affected URL	https://demo.testfire.net/login.jsp
Threat	<ul style="list-style-type: none"> • Credential Theft: Attackers intercepting network traffic could steal login credentials. • Unauthorized Access: Compromised credentials can lead to account takeovers and data breaches. • Regulatory Non-Compliance: Violates security standards like OWASP Top 10 (A2:2021) and may breach GDPR, PCI-DSS, etc. • Increased Attack Surface: Cleartext credentials may be stored in browser caches, proxies, or server logs.
Tools used	Burp suite

Steps to Reproduce:-

1. Go to URL <https://demo.testfire.net/login.jsp>
2. use burp suite to proxy and check the login request.
3. fill the login form and click go
4. check the post request in burp suite

Request

Pretty Raw Hex

```

1 POST /doLogin HTTP/1.1
2 Host: demo.testfire.net
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:137.0) Gecko/20100101 Firefox/137.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 41
9 Origin: http://demo.testfire.net
10 Connection: close
11 Referer: http://demo.testfire.net/login.jsp
12 Cookie: JSESSIONID=564E400ECC29ECA268818FB064987495
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15
16 uid=testuser&passw=secret&btnSubmit=Login

```


Response

Pretty Raw Hex Render

```

1 HTTP/1.1 302 Found
2 Server: Apache-Coyote/1.1
3 Location: login.jsp
4 Content-Length: 0
5 Date: Thu, 10 Apr 2025 04:49:03 GMT
6 Connection: close
7
8

```



- Observe the password was found in **plain text** in the request body without any encryption or hashing.

Recommended Remediation:

- Always Use HTTPS:** Ensure that **all** application components enforce HTTPS with valid certificates and HSTS headers.
- Encrypt Sensitive Fields:** Use **client-side encryption** or tokens for transmitting sensitive data like passwords.
- Implement Secure Headers:** Use `Strict-Transport-Security` to prevent protocol downgrading.
- Perform TLS Hardening:** Disable weak cipher suites, enforce TLS 1.2+ only.
- Avoid Logging Sensitive Data:** Ensure credentials are never logged in server or proxy logs.

2.6 Strict transport security not enforced

Parameter	Description
Severity	Low
Description	The web application does not enforce HTTP Strict Transport Security (HSTS) headers. HSTS is a critical security mechanism that tells browsers to only connect to the website over HTTPS,

	preventing protocol downgrade attacks and cookie hijacking. Without HSTS, attackers can potentially intercept or manipulate traffic by tricking users into accessing the site via HTTP.
Vulnerable parameter	-
Affected URL	https://demo.testfire.net
Threat	<ul style="list-style-type: none"> • Man-in-the-Middle (MITM) Attacks: Attackers can intercept or alter traffic before redirection. • Session/Cookie Hijacking: Sensitive information like session cookies can be exposed if forced onto HTTP. • Downgrade Attacks: Attackers can force connections to HTTP using tools like SSLStrip.
Tools used	Burp suite

Steps to Reproduce:-

1. Accessed the site at <https://demo.testfire.net>
2. Inspected response headers in burp proxy – **no Strict-Transport-Security header present**

Request	Response
<div> <div>Pretty Raw Hex</div> <div> <pre> 1 GET / HTTP/1.1 2 Host: demo.testfire.net 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:137.0) Gecko/20100101 Firefox/137.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Connection: close 8 Upgrade-Insecure-Requests: 1 9 Priority: u=0, i 10 11 </pre> </div> </div>	<div> <div>Pretty Raw Hex Render</div> <div> <pre> 1 HTTP/1.1 200 OK 2 Server: Apache-Coyote/1.1 3 Set-Cookie: JSESSIONID=564E400ECC29ECA268818FB064987495; Path=/; HttpOnly 4 Content-Type: text/html; charset=ISO-8859-1 5 Date: Thu, 10 Apr 2025 04:46:41 GMT 6 Connection: close 7 Content-Length: 9405 8 9 10 11 12 13 14 15 16 17 18 </pre> </div> </div>

Recommended Remediation:

1. Implement HSTS Header

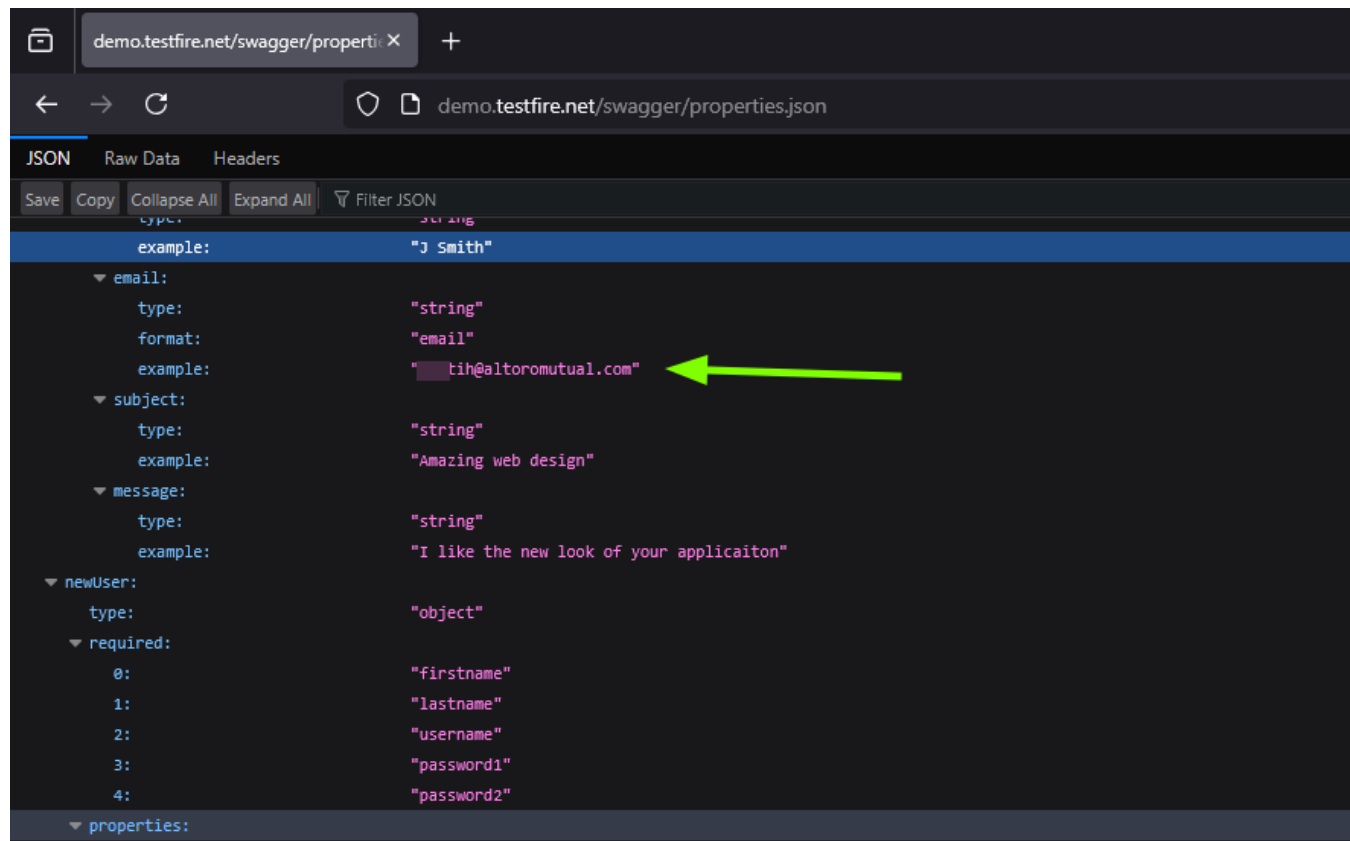
2. **Redirect All HTTP Requests to HTTPS:**
3. **Submit Domain for HSTS Preload List (Optional):**

2.7 Email Address Exposed

Parameter	Description
Severity	low
Description	An email address of some user have been disclosed.
Vulnerable parameter	-
Affected URL	https://demo.testfire.net/swagger/properties.json
Threat	<ul style="list-style-type: none"> • Phishing Risk • Social Engineering • Spam & Abuse • Reconnaissance
Tools used	Burp suite

Steps to Reproduce:-

1. go to url : <https://demo.testfire.net/swagger/properties.json>
2. look for email, it is shown below:



Recommended Remediation:

1. Avoid Unnecessary Disclosure:

- Do not display email addresses unless absolutely required for public communication.

2. Obfuscate Displayed Emails:

- Use formats like support [at] testfire [dot] net to prevent harvesting by bots.

3. Access Control:

- If emails are needed for user support or internal use, restrict them to authenticated users only.

4. Review Source Code & API Responses:

- Ensure no hardcoded or exposed emails are unintentionally revealed in comments, JS files, or API outputs.