TECHNOLOGY TRACK: AI FOR CYBERSECURITY WITH IBM QRADAR

PROJECT TITLE: Design an AI-driven incident response platform that assists cybersecurity teams in automating incident triage and response tasks.

TEAM NUMBER: 6.1

TEAM MEMBERS:

- Raghavendra Reddy Orra
- Greeshma Reddy Basireddy
- Shaik Muhammed Faizaan Ali
- Farzeen Naiz

1.Vulnerability name: Cross site scripting (stored)

CWE: 79

OWASP category: A03:2021 -Injections

Description: It occurs when a malicious script is injected directly into a vulnerable web application. Reflected XSS involves the reflection of a malicious script of a web application, onto a user's browser.

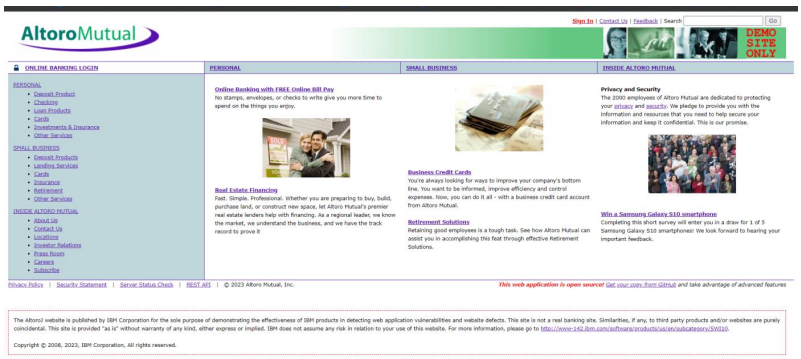
Business Impact: The application stores dangerous data in a database, message forum, visitor log, or other trusted data store. Later, the dangerous data is subsequently read back into the application and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user.

Vulnerability path: http://testfire.net/

Vulnerabilityparameter:
http://testfire.net/search.jsp?query=%27%3E%3Cscript%3Ealert%28%27hacked%27%29%3C%2Fscript%3E

Steps to Reproduce:

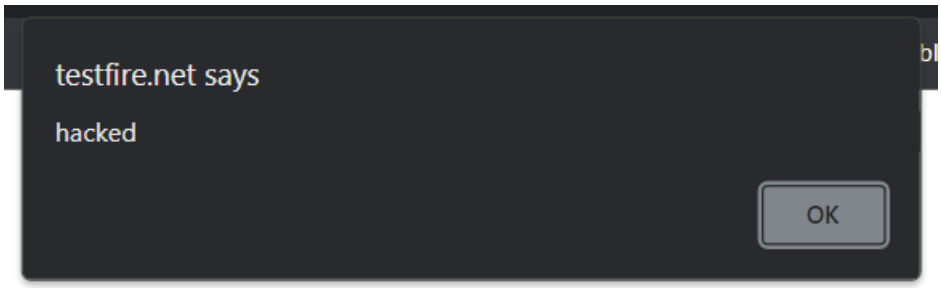
Access the URL



In the search box we will input some code to perform the vulnerability.

The Script we will be inputting is '><script>alert('hacked')</script>

This displays a harmless pop up alert box with the text saying 'hacked'



Recommendation:

- When interacting with databases, use parameterized statements or prepared statements to avoid SQL injection, which can be a vector for XSS.
- Sanitize and validate all user inputs and ensure that any data displayed on the web page is properly encoded. Use output encoding libraries or functions to prevent script injection.

---

2. Vulnerability name: Broken access control

CWE: 285

Description: When access control checks are not applied consistently - or not at all - users are able to access data or perform actions that they should not be allowed to perform. This can lead to a wide range of problems, including information exposures, denial of service, and arbitrary code execution.

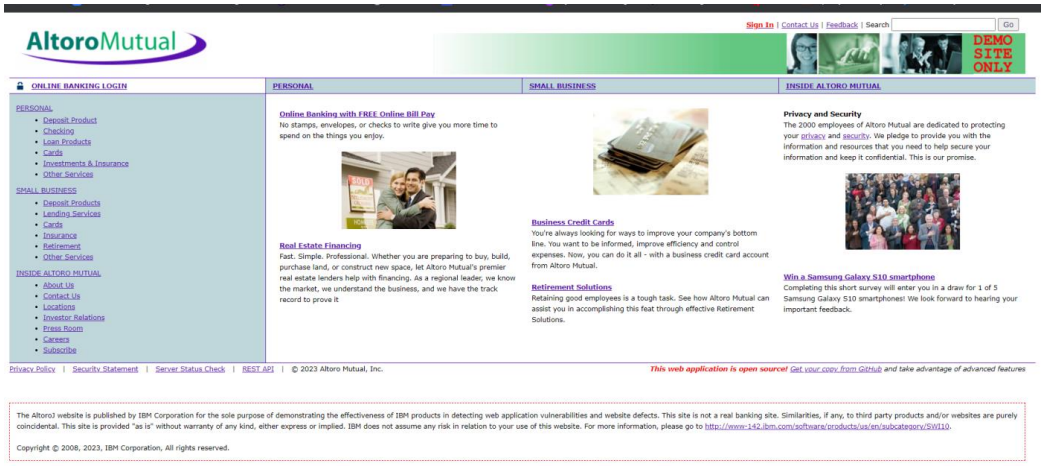
Business Impact: wide-ranging business impact, including financial losses, reputation damage, legal consequences, operational disruption, and loss of customer trust. Addressing this vulnerability is crucial to protect the organization and its stakeholders from these potential negative effects.

Vulnerability path: http://testfire.net/

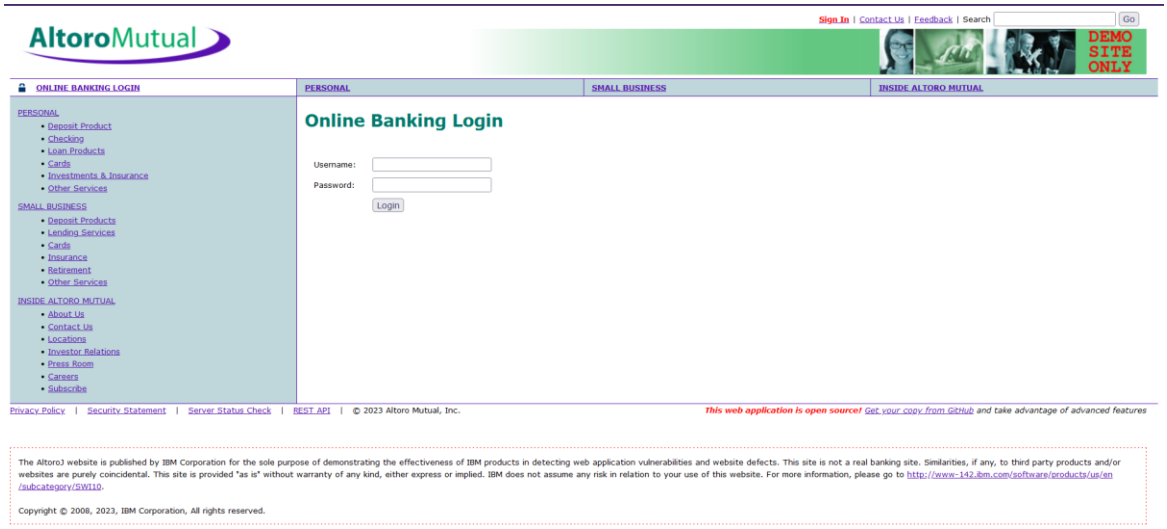
Vulnerability parameter: http://testfire.net/login.jsp

Steps to Reproduce:

Access the URL

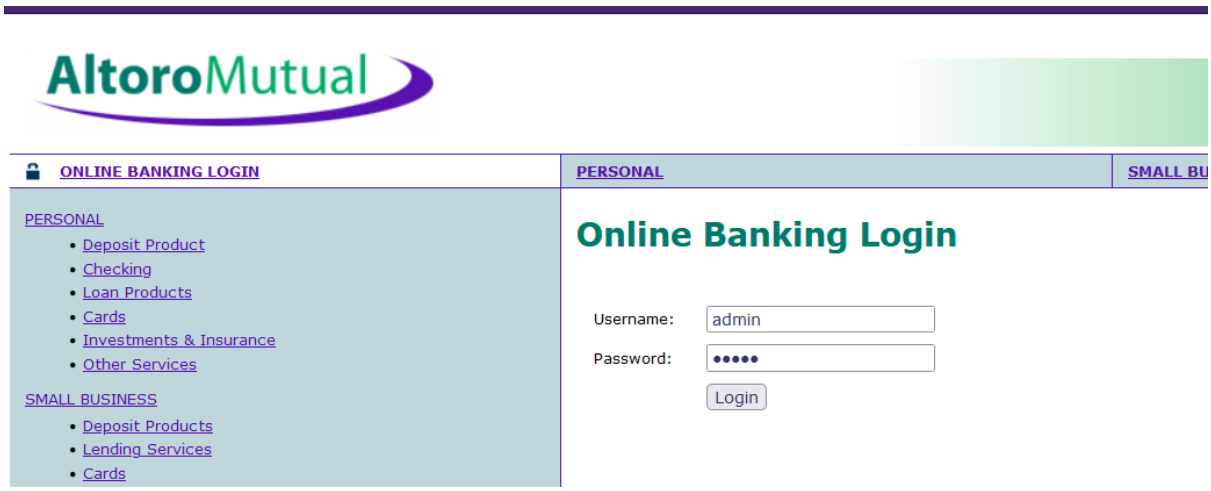


Now we will try to sign in to this website with admin privileges



Also, we will be using burp suite to get requests from the website and know additional information.

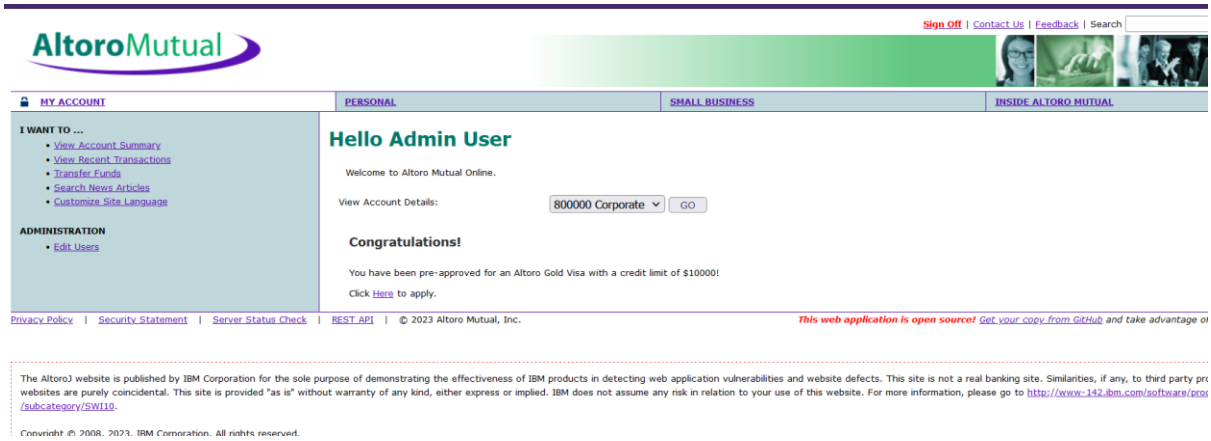
We use 'admin' for the username and password.



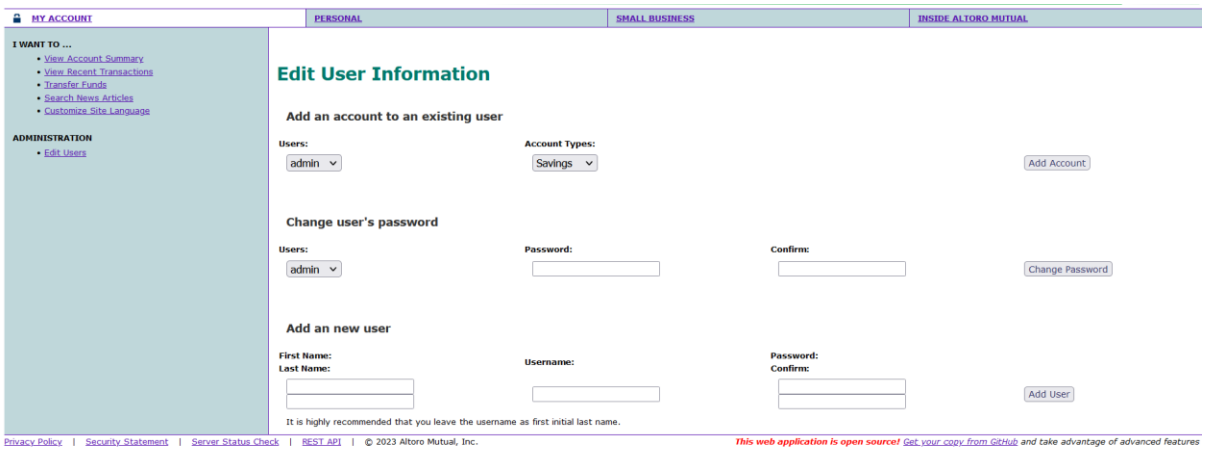
This request has been received in the burp suite with the username and password as well in clear text.

```
Pretty   Raw   Hex
1  POST /doLogin HTTP/1.1
2  Host: testfire.net
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/117.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate, br
7  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 37
9  Origin: http://testfire.net
10 Connection: close
11 Referer: http://testfire.net/login.jsp
12 Cookie: JSESSIONID=B6ADB36ACD5C83083787343A1F97F853; AltoroAccounts="ODAwMDAwfkNvcnBvcmFOZX45LjQ30TA1MTE2MUU3fDgwMDAwMX5DaGVja2luZ34tNC4yMjcONjkONTZFN3w="
13 Upgrade-Insecure-Requests: 1
14
15 uid=admin&passw=admin&btnSubmit=Login
```

now we click on forward request in the burp suite and then we will be redirected to the admin user details.



We can even edit the other users data and delete their login details as well.



Recommendations:

- Ensure that the application has a robust authentication and authorization mechanism in place. Users should be authenticated before any sensitive actions are performed, and authorization checks should be conducted to verify that users have the necessary permissions for the requested operation.
- Ensure that session management is secure. Use strong session IDs, enforce session timeouts, and regenerate session tokens after login. Implement session fixation protection to prevent session hijacking.

3.Vulnerability name: SQL injection

CWE: 89

Description: The product constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component.

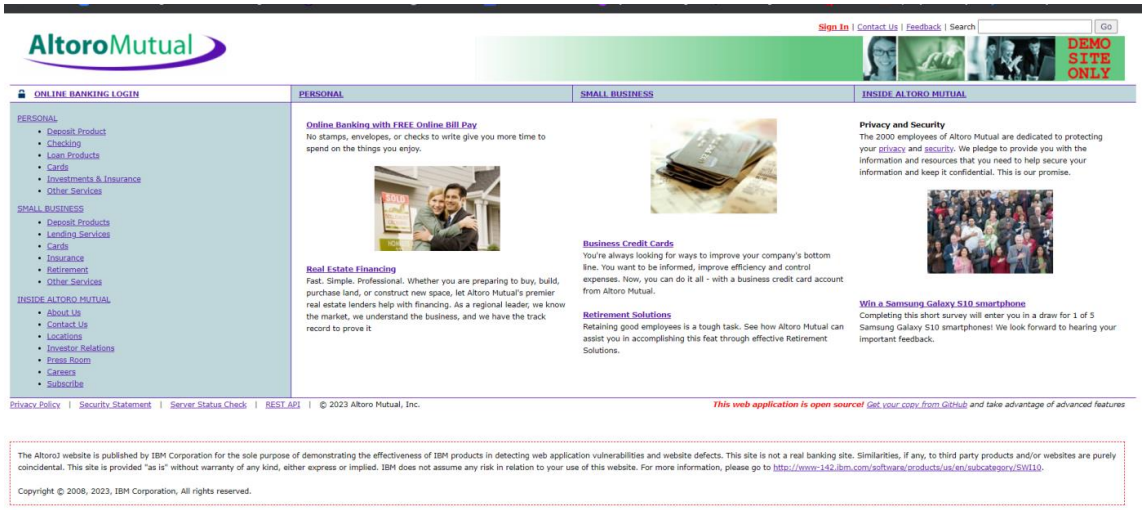
Business Impact: In summary, it is crucial to underscore that CWE-89, known as SQL Injection, can exert a profound and diverse business impact. This encompasses critical facets such as data breaches, financial setbacks, harm to reputation, legal ramifications, and operational turmoil. Hence, the imperative of preventing and remedying SQL injection vulnerabilities cannot be overstated, as it is indispensable for fortifying the security and continuity of an organization's applications and data.

Vulnerability path: http://testfire.net/

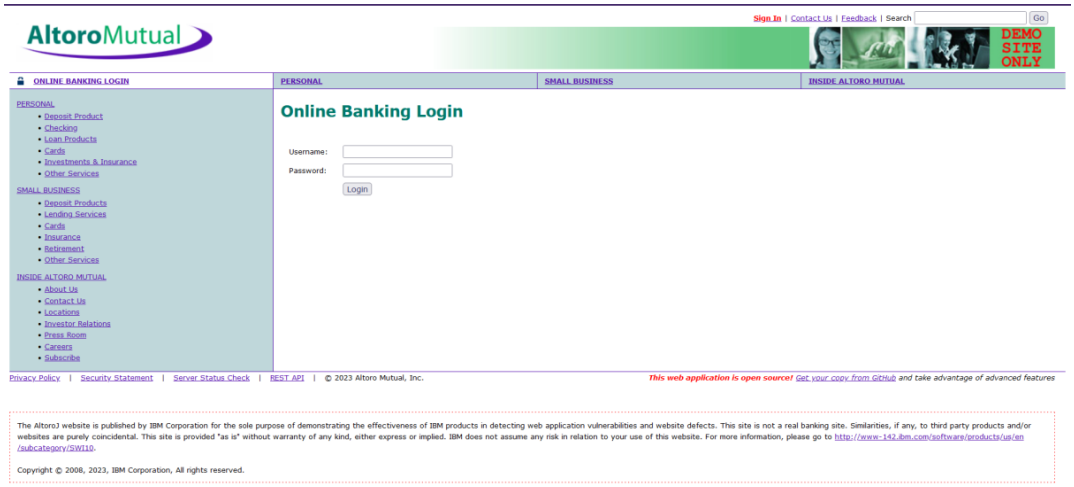
Vulnerability parameter: http://testfire.net/login.jsp

Steps to Reproduce:

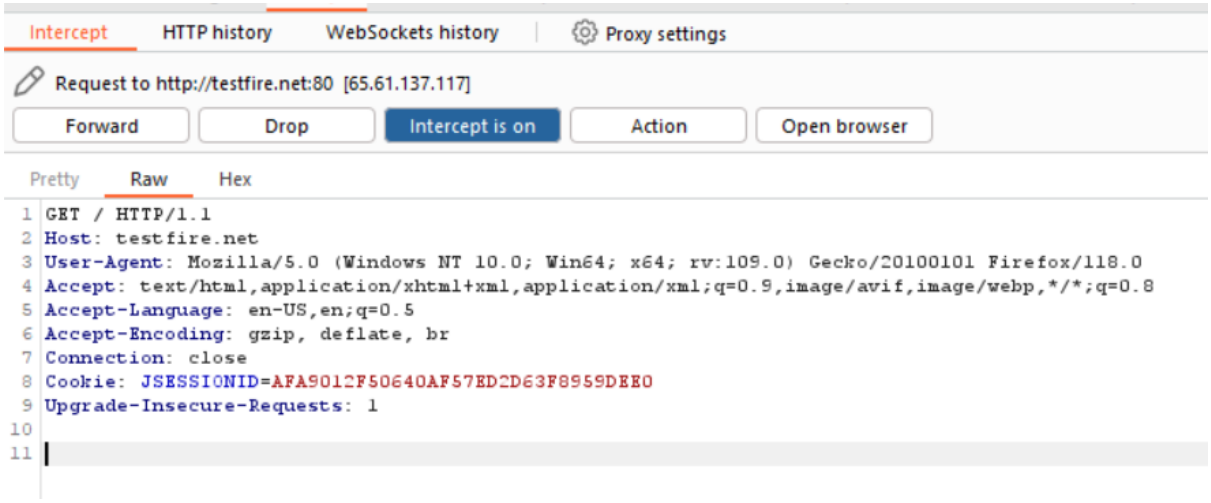
Access the URL

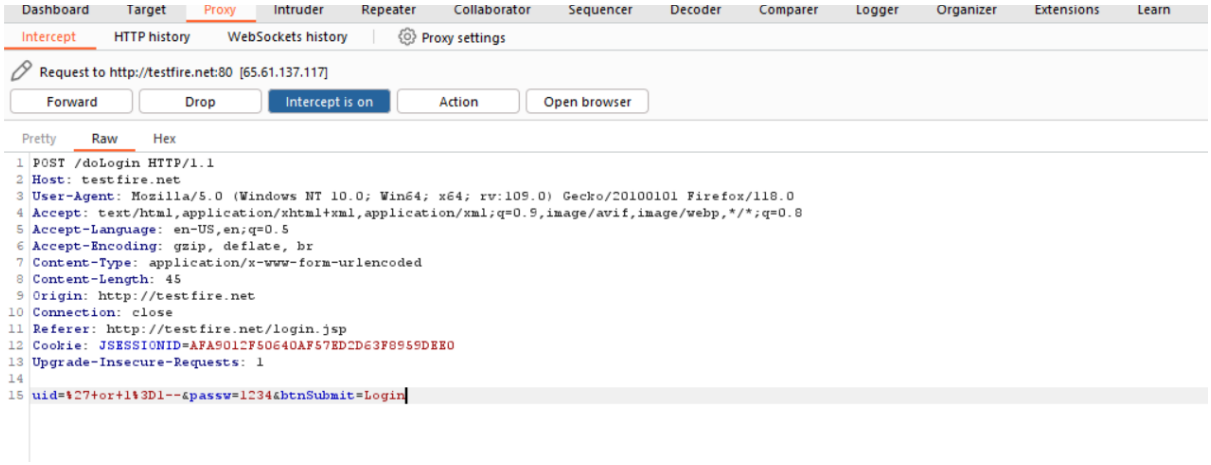


Now we will try to sign in to this website with admin privileges but using SQL injection

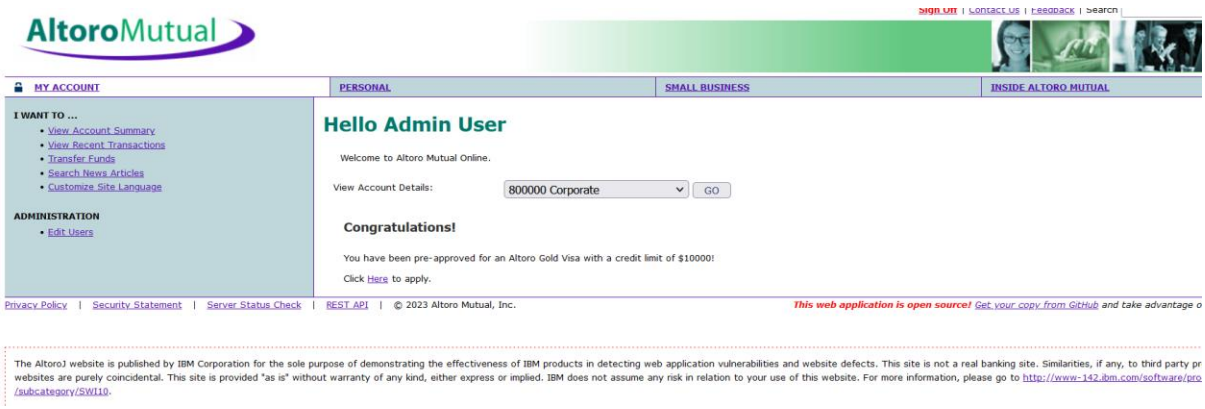


Also, we will be using burp suite to get requests from the website and know additional information.





Click forward request multiple times to proceed to the login page.



With this we can know that sql injection worked and we got the admin privileges.

Recommendations:

- If parameterized statements are not feasible, use proper input validation and escaping mechanisms to sanitize user inputs before they are used in SQL queries. This helps prevent malicious code injection.
- Utilize stored procedures for database operations. This can help separate SQL code from application code and reduce the risk of SQL injection.
- Implement strict input validation for all user inputs, ensuring that data adheres to the expected format and structure.

4.**Vulnerability name:** Broken authentication

CWE: 285

Description: The system's authorization functionality does not prevent one user from gaining access to another user's data or record by modifying the key value identifying the data.

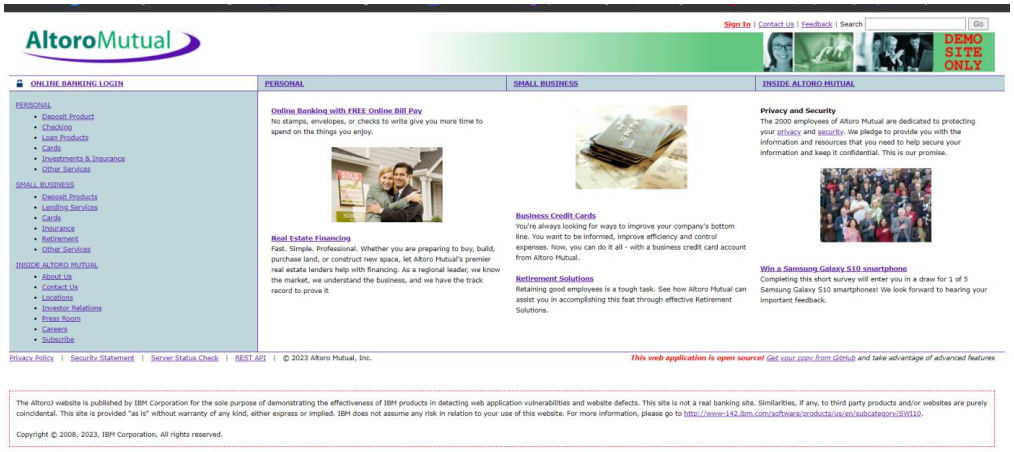
Business Impact: To effectively mitigate the business impact of CWE-285, it is imperative that organizations place a strong emphasis on bolstering their authentication and session management practices. This should encompass the adoption of multi-factor authentication, the secure storage of credentials, meticulous session handling, and a commitment to conducting routine security assessments and testing. The rectification of these vulnerabilities stands as a critical imperative, safeguarding sensitive data, user identities, and the organization's overarching security stature and reputation.

Vulnerability path: http://testfire.net/

Vulnerability parameter: http://testfire.net/login.jsp

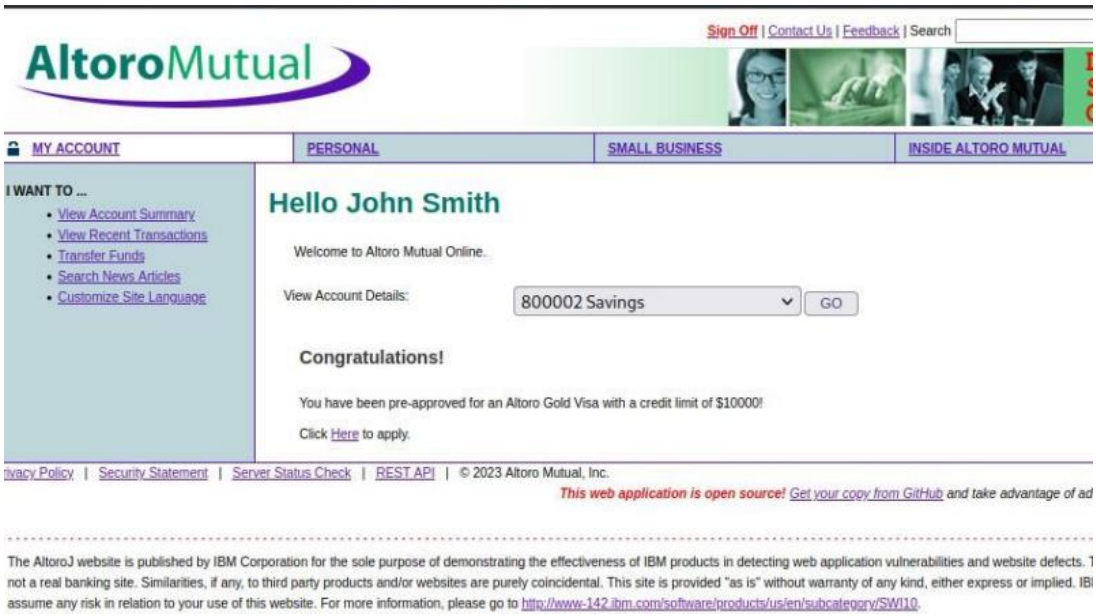
Steps to Reproduce:

Access the URL



Now we will try to login using some different approach.

As we know which users are present in the database of this website by using the admin privileges. We can directly access a particular user by simply knowing their username; we will add some characters after his user name as a sql injection to simply bypass the password.



This leads us to the details of this person's account.

Recommendations:

- Implement strong authentication mechanisms, such as multi-factor authentication (MFA) or two-factor authentication (2FA), to ensure that only authorized users can access the system.
- Implement proper authorization controls to ensure that users have the necessary permissions to access specific functions or data. Use role-based access control (RBAC) to manage user privileges effectively.
- Deploy intrusion detection and prevention systems (IDS/IPS) to monitor and block suspicious activities related to authentication and authorization.

5.Vulnerability Name: Insecure Direct object Reference

CWE: 639

OWASP Category: A01: Broken Access Control

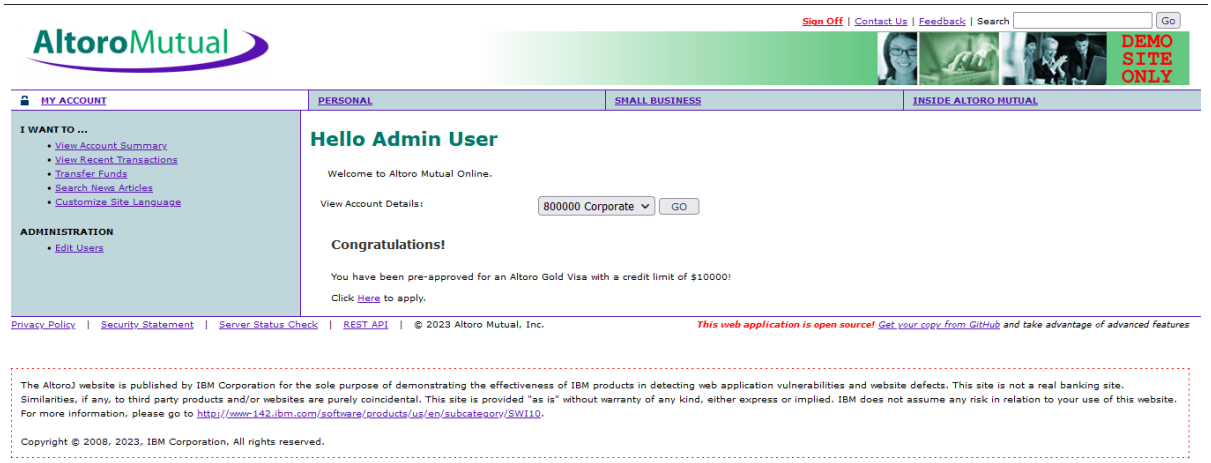
Description:
Insecure Direct Object Reference (IDOR) is a vulnerability that arises when attackers can access or modify objects by manipulating identifiers used in a web application's URLs or parameters. It occurs due to missing access control checks, which fail to verify whether a user should be allowed to access specific data.

Business Impact:
To effectively mitigate the business impact of CWE-639, organizations must make it a top priority to fortify their access control and authorization mechanisms. This entails the implementation of robust security measures, the regular conduct of comprehensive security assessments, and the deployment of intrusion detection systems to promptly identify and counter unauthorized access attempts. These measures stand as absolutely critical in the defense of sensitive data, the overall security of systems, and the preservation of the organization's esteemed reputation.

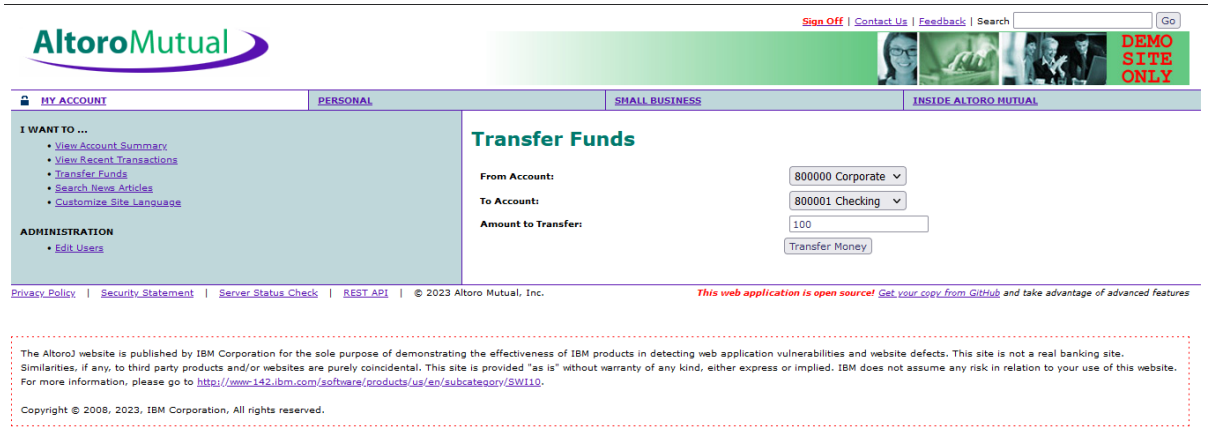
Vulnerability Path: http://testfire.net

Vulnerability Parameter: http://testfire.net/bank/transfer.jsp

Steps to Reproduce:
Go to the domain.
Sign into the site.



Open “Transfer Money” on the left side and fill in the details. On the intercept and click transfer



testfire.net

Open burp and notice the change.

Pretty Raw Hex

```
1  POST /bank/doTransfer HTTP/1.1
2  Host: testfire.net
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate, br
7  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 78
9  Origin: http://testfire.net
10 Connection: close
11 Referer: http://testfire.net/bank/transfer.jsp
12 Cookie: JSESSIONID=17BCE438BBB750A14D269616C7623BB0; AltoroAccounts=
   "ODAwMDAwfkNvcnBvcmFOZX4xLjYzNTczNTM1NjM10Dc0NEUy0Xw4MDAwMDF+Q2hlY2tpbmd+LTEuNjM1NzM1MzU2MzU4NzQ0RTI5fA=="
13 Upgrade-Insecure-Requests: 1
14
15 fromAccount=800000&toAccount=800001&transferAmount=100&transfer=Transfer+Money
```

In the 15th line change the amount from 100 to 1000 and click forward

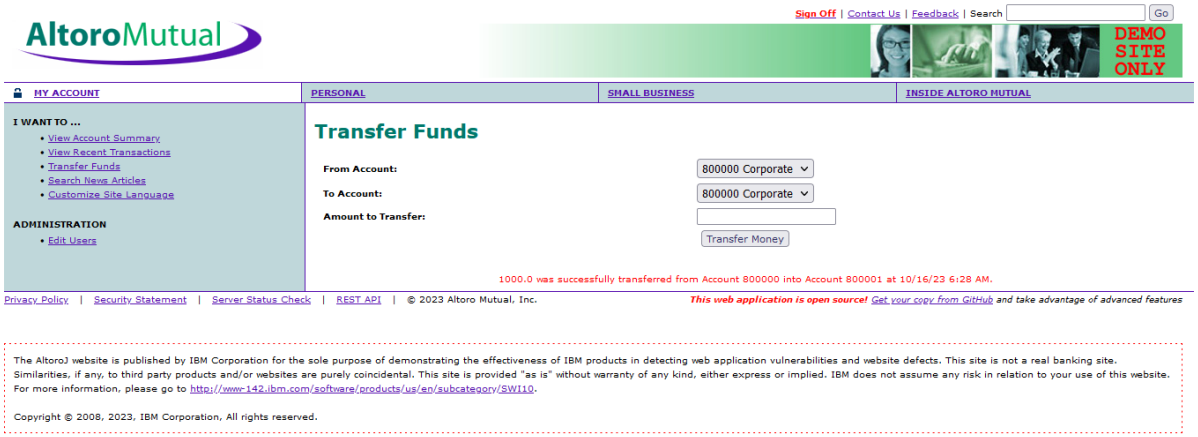
```
fromAccount=800000&toAccount=800001&transferAmount=100&transfer=Transfer+Money
```

```
fromAccount=800000&toAccount=800001&transferAmount=1000&transfer=Transfer+Money
```

Pretty Raw Hex

```
1  GET /v1/tiles HTTP/1.1
2  Host: contile.services.mozilla.com
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0
4  Accept: */*
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate, br
7  Te: trailers
8  Connection: close
9
10
```

Look at the site, we can notice the msg that shows the transfer of 1000.



Off the Intercept and open “View Recent Transactions”.

I WANT TO ...

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

ADMINISTRATION

- Edit Users

Recent Transactions

After yyyy-mm-dd Before yyyy-mm-dd Submit

Transaction ID	Transaction Time	Account ID	Action	Amount
9016	2023-10-16 06:28	800001	Deposit	$1000.00
9015	2023-10-16 06:28	800000	Withdrawal	-$1000.00
8920	2023-10-16 04:46	800001	Deposit	$23345.00
8919	2023-10-16 04:46	800000	Withdrawal	-$23345.00
8916	2023-10-16 04:44	800001	Deposit	$23345.00
8915	2023-10-16 04:44	800000	Withdrawal	-$23345.00
8894	2023-10-16 04:40	800001	Deposit	$23345.00
8893	2023-10-16 04:40	800000	Withdrawal	-$23345.00
8892	2023-10-16 04:40	800001	Deposit	$23345.00
8891	2023-10-16 04:40	800000	Withdrawal	-$23345.00
7565	2023-10-16 04:17	800000	Withdrawal	-$10.00
7228	2023-10-16 04:14	800000	Deposit	$1000.00
6802	2023-10-16 04:09	800001	Deposit	$100000000000000.00
6801	2023-10-16 04:09	800000	Withdrawal	-$100000000000000.00
5388	2023-10-16 03:58	800000	Deposit	$10000000000.00
4628	2023-10-16 03:36	800001	Deposit	$20190.00
4627	2023-10-16 03:36	800000	Withdrawal	-$20190.00
4626	2023-10-16 03:35	800001	Deposit	$20190.00
4625	2023-10-16 03:35	800000	Withdrawal	-$20190.00
4624	2023-10-16 03:35	800001	Deposit	$600.00
4623	2023-10-16 03:35	800000	Withdrawal	-$600.00
4616	2023-10-16 03:10	800000	Deposit	$87446.00
4615	2023-10-16 03:10	800001	Withdrawal	-$87446.00

Recommendation:

- Utilize parameterized queries or prepared statements when interacting with databases. This prevents user-controlled input from being executed as SQL code.
- Implement a Content Security Policy to mitigate the risk of cross-site scripting (XSS) attacks, which can be used to execute unauthorized queries.
- Implement account lockout mechanisms that temporarily suspend accounts after a specified number of unauthorized access attempts to prevent brute force attacks.

---

6. **Vulnerability name:** Security Misconfiguration

CWE:732

Description: The product specifies permissions for a security-critical resource in a way that allows that resource to be read or modified by unintended actors.

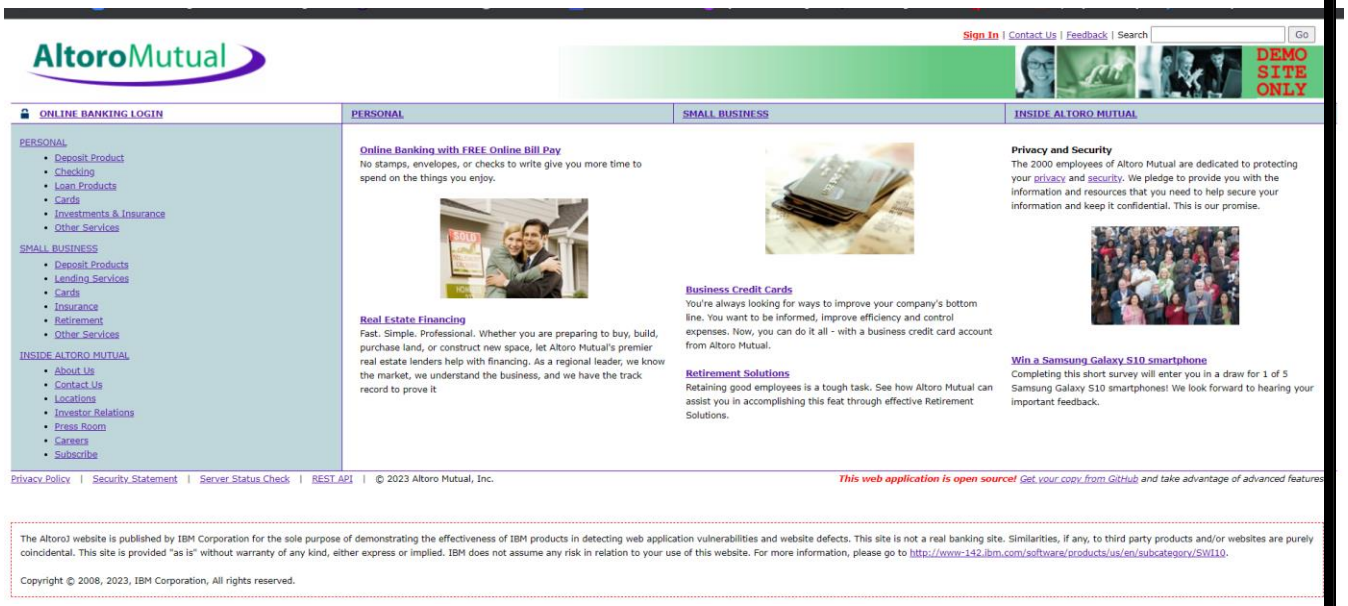
Business Impact: To effectively lessen the business consequences stemming from CWE-732, organizations must place a strong emphasis on bolstering their configuration management practices. This involves the establishment of secure default settings, the routine conduct of comprehensive security evaluations, and the deployment of intrusion detection systems to promptly detect and address misconfigurations and vulnerabilities. These actions are of paramount importance in safeguarding sensitive data, fortifying system security, and preserving the organization's esteemed reputation.

Vulnerability path: http://testfire.net/

Vulnerability parameter: http://testfire.net/login.jsp

Steps to Reproduce:

Access the URL



We will be using the uniscan tool which is available in the kali.

```
┌──(kali㉿kali)-[~]
└─$ sudo uniscan -h
[sudo] password for kali:
#####################################
# Uniscan project                   #
# http://uniscan.sourceforge.net/   #
#####################################
V. 6.3

OPTIONS:
        -h      help
        -u      <url> example: https://www.example.com/
        -f      <file> list of url's
        -b      Uniscan go to background
        -q      Enable Directory checks
        -w      Enable File checks
        -e      Enable robots.txt and sitemap.xml check
        -d      Enable Dynamic checks
        -s      Enable Static checks
        -r      Enable Stress checks
        -i      <dork> Bing search
        -o      <dork> Google search
        -g      Web fingerprint
        -j      Server fingerprint
usage:
[1] perl ./uniscan.pl -u http://www.example.com/ -qweds
[2] perl ./uniscan.pl -f sites.txt -bqweds
[3] perl ./uniscan.pl -i uniscan
[4] perl ./uniscan.pl -i "ip:xxx.xxx.xxx.xxx"
[5] perl ./uniscan.pl -o "inurl:test"
[6] perl ./uniscan.pl -u https://www.example.com/ -r
```

We will use this tool to scan for hidden directories that can be access due to poor security measures taken in this website.

```
┌──(kali㉿kali)-[~]
└─$ sudo uniscan -u http://testfire.net/ -qweds
[sudo] password for kali:
######################################
# Uniscan project                    #
# http://uniscan.sourceforge.net/    #
######################################
V. 6.3


Scan date: 17-10-2023 7:18:22
==============================================================================
| Domain: http://testfire.net/
| Server: Apache-Coyote/1.1
| IP: 65.61.137.117
==============================================================================
|
| Directory check:
| [+] CODE: 200 URL: http://testfire.net/admin/fckeditor/admins/
| [+] CODE: 200 URL: http://testfire.net/admin/
| [+] CODE: 200 URL: http://testfire.net/admin/scripts/fckeditor/
| [+] CODE: 200 URL: http://testfire.net/admin/FCKeditor/
| [+] CODE: 200 URL: http://testfire.net/aux/
| [+] CODE: 200 URL: http://testfire.net/bank/
==============================================================================
|
```

We found directories that can be accessed easily. Also all the files that are present in the website has also been found

```
| File System
| File check:
| [+] CODE: 200 URL: http://testfire.net/admin/account.asp
| [+] CODE: 200 URL: http://testfire.net/admin/account.html
| [+] CODE: 200 URL: http://testfire.net/admin/admin.php
| [+] CODE: 200 URL: http://testfire.net/admin/admin_phpinfo.php4
| [+] CODE: 200 URL: http://testfire.net/admin/account.php
| [+] CODE: 200 URL: http://testfire.net/admin/admin.shtml
| [+] CODE: 200 URL: http://testfire.net/admin/aindex.htm
| [+] CODE: 200 URL: http://testfire.net/admin/auth.php
| [+] CODE: 200 URL: http://testfire.net/admin/cfg/configscreen.inc.php+
| [+] CODE: 200 URL: http://testfire.net/admin/cfg/configsite.inc.php+
| [+] CODE: 200 URL: http://testfire.net/admin/cfg/configsql.inc.php+
| [+] CODE: 200 URL: http://testfire.net/admin/cfg/configtache.inc.php+
| [+] CODE: 200 URL: http://testfire.net/admin/cms/htmltags.php
| [+] CODE: 200 URL: http://testfire.net/admin/config.php
| [+] CODE: 200 URL: http://testfire.net/admin/contextAdmin/contextAdmin.html
| [+] CODE: 200 URL: http://testfire.net/admin/controlpanel.asp
| [+] CODE: 200 URL: http://testfire.net/admin/controlpanel.php
| [+] CODE: 200 URL: http://testfire.net/admin/controlpanel.html
| [+] CODE: 200 URL: http://testfire.net/admin/cplogfile.log
| [+] CODE: 200 URL: http://testfire.net/admin/cp.php
| [+] CODE: 200 URL: http://testfire.net/admin/cp.html
| [+] CODE: 200 URL: http://testfire.net/admin/database/wwForum.mdb
| [+] CODE: 200 URL: http://testfire.net/admin/credit_card_info.php
| [+] CODE: 200 URL: http://testfire.net/admin/datasource.asp
| [+] CODE: 200 URL: http://testfire.net/admin/db.php
| [+] CODE: 200 URL: http://testfire.net/admin/home.asp
| [+] CODE: 200 URL: http://testfire.net/admin/home.php
| [+] CODE: 200 URL: http://testfire.net/admin/index.asp
| [+] CODE: 200 URL: http://testfire.net/admin/index.html
| [+] CODE: 200 URL: http://testfire.net/admin/index.php
| [+] CODE: 200 URL: http://testfire.net/admin/phpinfo.php
| [+] CODE: 200 URL: http://testfire.net/admin/settings.inc.php+
| [+] CODE: 200 URL: http://testfire.net/admin/system_footer.php
| [+] CODE: 200 URL: http://testfire.net/admin/upload.php
| [+] CODE: 200 URL: http://testfire.net/admin/templates/header.php
| [+] CODE: 200 URL: http://testfire.net/admin/wg_user-info.ml
| [+] CODE: 200 URL: http://testfire.net/admin/script.php
| [+] CODE: 200 URL: http://testfire.net/login.jsp
==============================================================================
|
```

Let us go through one webpage to see the vulnerability

Here we can see the sensitive data of job applications which can help an attacker.

Recommendations:

- Implement a change management process to document and review all changes to system configurations. This helps prevent unauthorized or unintended alterations.
- Follow security guidelines provided by software and hardware vendors to ensure secure configurations.
- Regularly perform compliance checks against industry standards and regulations to ensure that configurations remain in compliance with security requirements.

7.Vulnerability Name: Cross-Site Request Forgery

CWS: 352

OWASP Category: A08: Cross-Site Request Forgery

Description:
Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

Business Impact:
To effectively reduce the business repercussions of CWE-352, organizations must prioritize secure coding practices, including the incorporation of anti-CSRF tokens, while also conducting routine security assessments and testing to detect and rectify CSRF vulnerabilities. Furthermore, educating

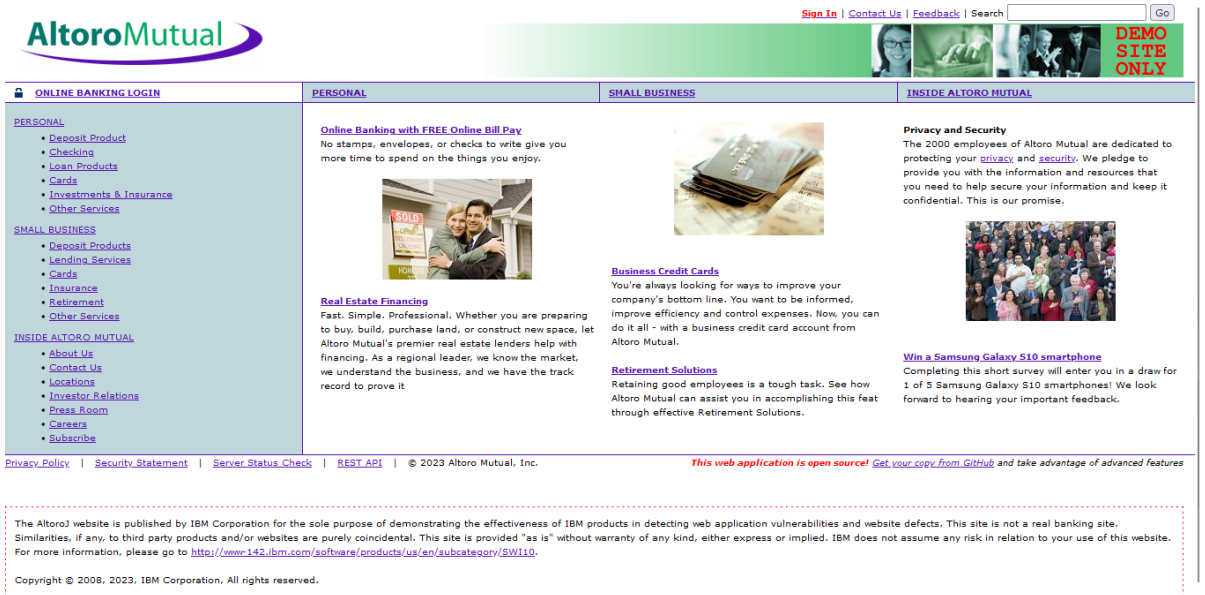
users on secure browsing practices is pivotal in preventing CSRF attacks. These actions are of paramount importance in the protection of data and in upholding the trust of customers and partners.

Vulnerability Path: http://testfire.net/

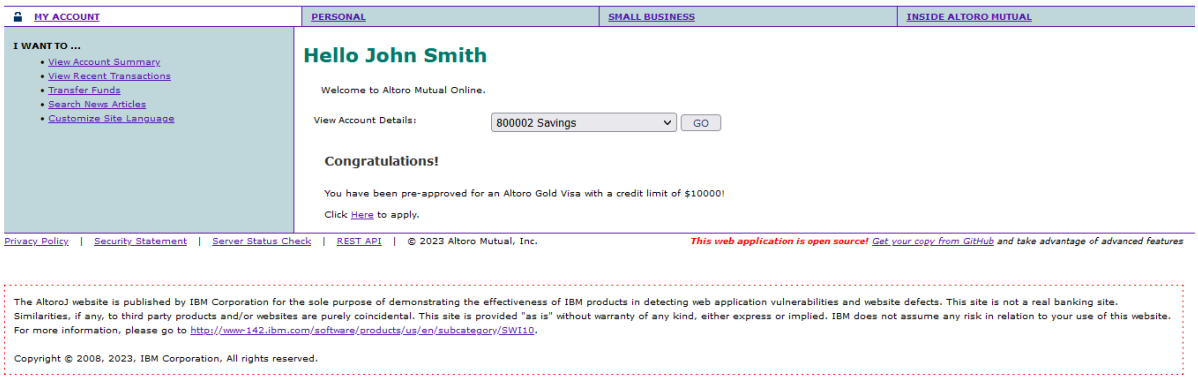
Vulnerability Parameter: http://testfire.net/bank/transfer.jsp

Steps to Reproduce:

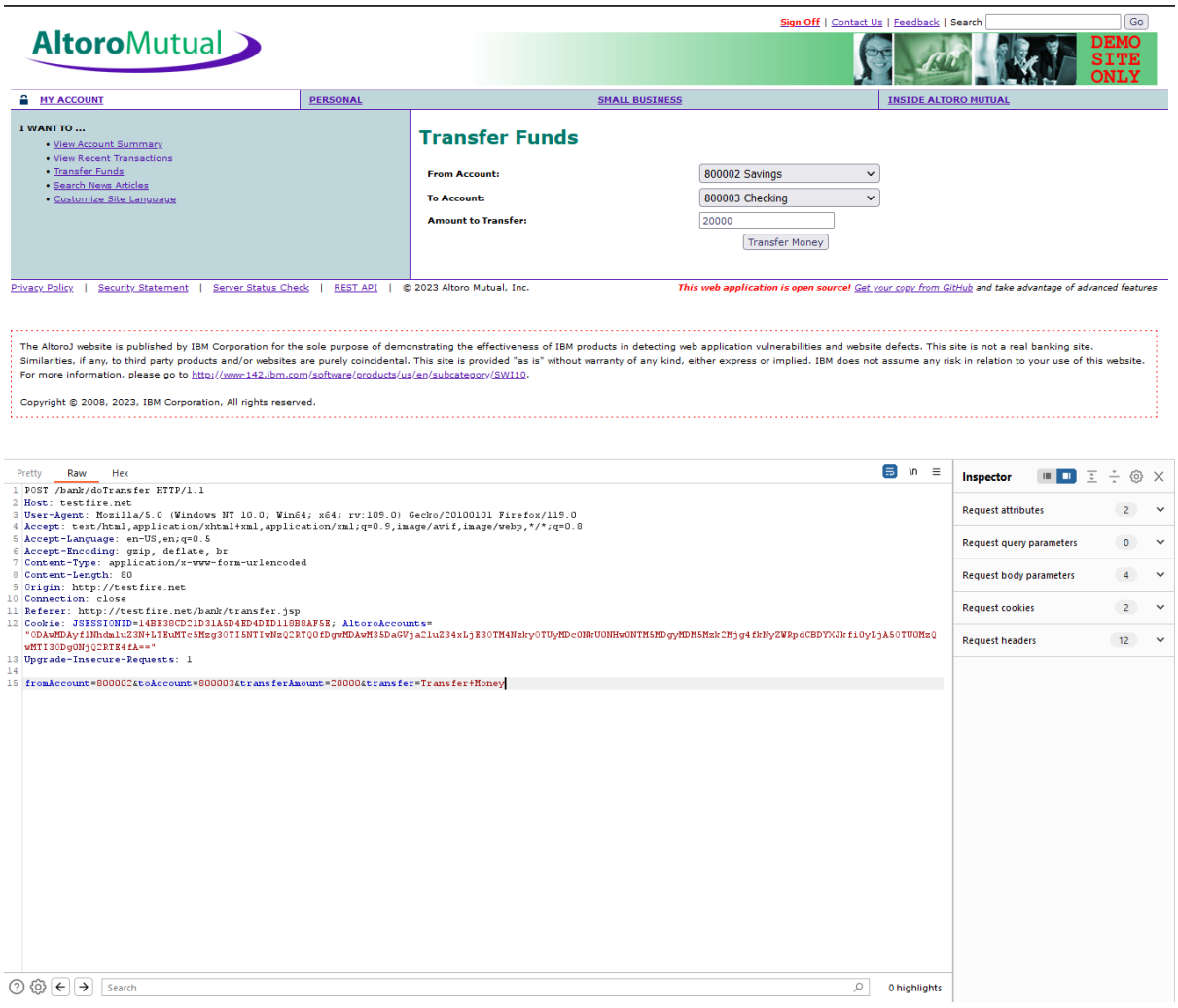
Open https://testfire.net/.



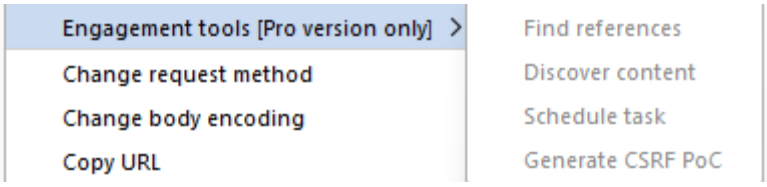
Login to it using default credentials, and you will find two accounts.



Capture the request in any Intercepting proxy (Burp Suite, Charles Proxy etc.) while transferring the amount from one account to another as shown in the below picture.



Generate CSRF POC using Burp Suite Engagement tools (by modifying the account numbers accordingly)



As the victim is already logged-in to the site an attacker can send the crafted malicious form/URL to the user to his email/text/by any other means and trick the victim using his social engineering skills/techniques like below.

```
Limited Time Offer Get a $100 Gift Card Now!!  (Evil :P)
```

User thinks, he could use a free $100 gift card! What could go wrong? You open the email and click the link/open the attachment.

Once the user opens the form a transaction is automatically performed from his account without his knowledge as the user is already logged in.

Recent Transactions

After yyyy-mm-dd Before yyyy-mm-dd Submit

Transaction ID	Transaction Time	Account ID	Action	Amount
12539	2023-10-17 06:38	800002	Deposit	$20000.00
12538	2023-10-17 06:38	800003	Withdrawal	-$20000.00
12537	2023-10-17 06:36	800003	Deposit	$1234.00
12536	2023-10-17 06:36	800003	Withdrawal	-$1234.00
12535	2023-10-17 06:36	800003	Deposit	$1234.00
12534	2023-10-17 06:36	800003	Withdrawal	-$1234.00
12533	2023-10-17 06:36	800003	Deposit	$1234.00
12532	2023-10-17 06:36	800003	Withdrawal	-$1234.00

Recommendations:

- Implement Synchronizer Token Patterns to include unique, randomly generated tokens in each HTTP request and validate these tokens on the server side to verify the authenticity of the request.
- Utilize the SameSite attribute in cookies to restrict their usage to same-site requests only, preventing them from being sent along with cross-site requests and effectively mitigating CSRF attacks.
- Utilize framework-specific protections and security features to prevent CSRF attacks. Many modern web frameworks have built-in mechanisms and libraries for handling CSRF vulnerabilities effectively.
- Implement a Content Security Policy to restrict the sources from which various types of content can be loaded. By specifying the trusted sources of content, you can minimize the risk of malicious code execution and reduce the likelihood of successful CSRF attacks.

8.Vulnerability name: Cleartext Transmission of Sensitive Information

CWE: 319

Description: The product transmits sensitive or security-critical data in cleartext in a communication channel that can be sniffed by unauthorized actors. Many communication channels can be "sniffed" (monitored) by adversaries during data transmission. For example, in networking, packets can traverse many intermediary nodes from the source to the destination, whether across the internet, an internal network, the cloud, etc. Some actors might have privileged access to a network interface or any link along the channel, such as a router, but they might not be authorized to collect the underlying data. As a result, network traffic could be sniffed by adversaries, spilling security-critical data.

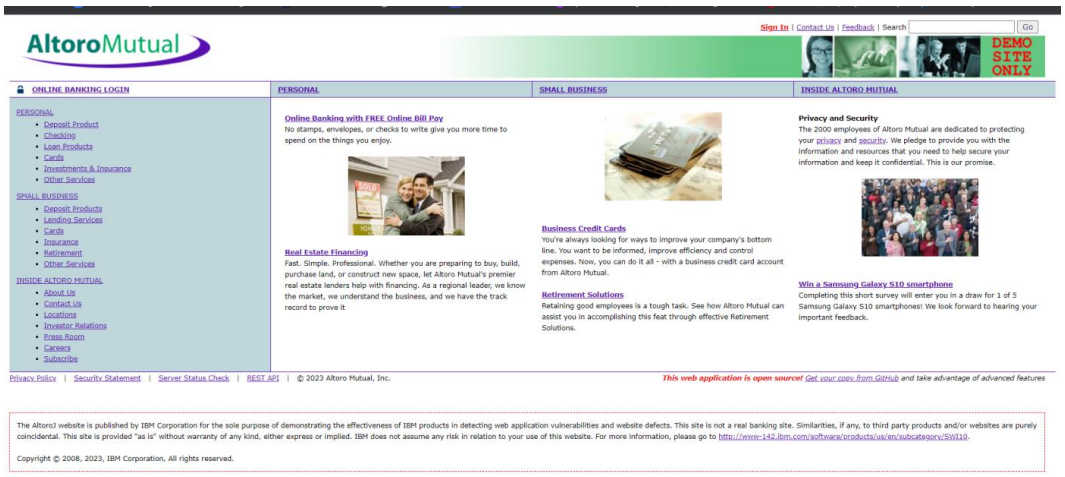
Business Impact: To effectively reduce the business consequences associated with CWE-319, organizations must prioritize the adoption of secure data transmission practices. This includes the utilization of encryption and robust, secure protocols. The routine conduct of security assessments and testing is pivotal in pinpointing and remedying vulnerabilities linked to data transmission. Furthermore, the education of users on secure data handling practices plays a vital role in proactively preventing data exposure incidents. These actions are of paramount importance in the protection of sensitive data and in preserving the trust of both customers and partners.

Vulnerability path: http://testfire.net/

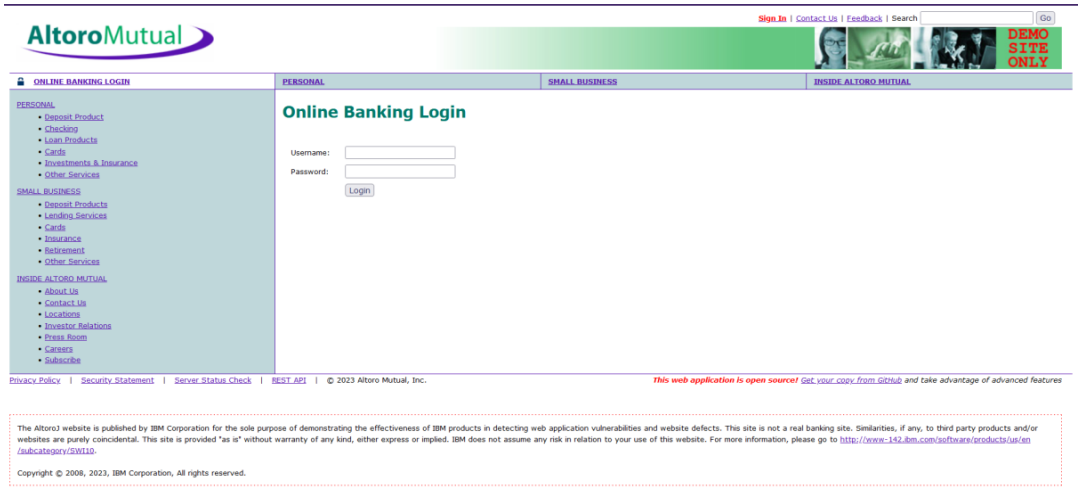
Vulnerability parameter: http://testfire.net/login.jsp

Steps to reproduce:

Access the URL

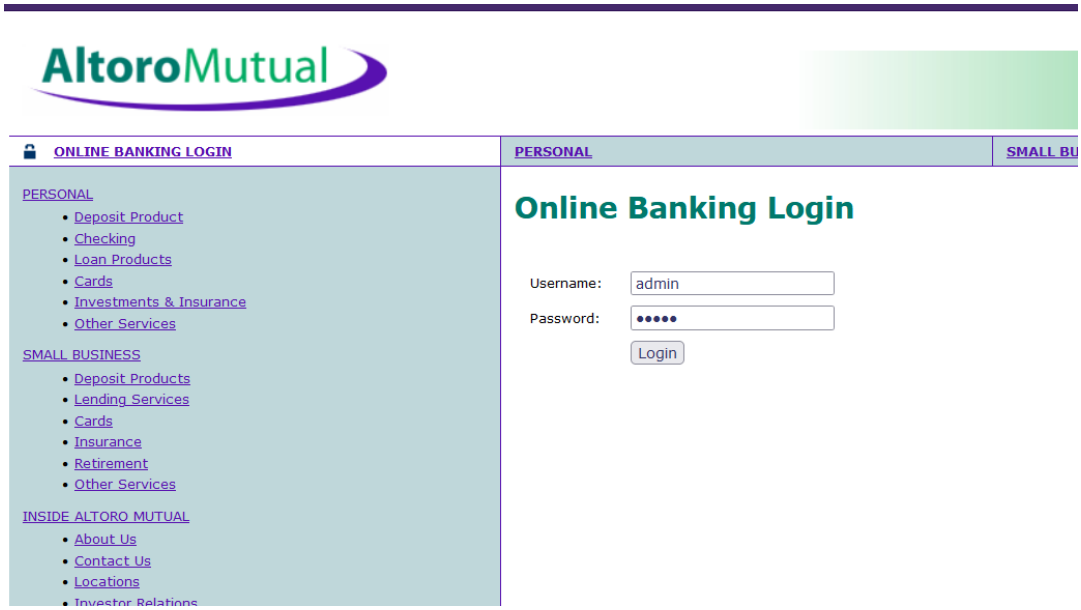


Now we will try to sign in to this website with admin privileges



Also, we will be using burp suite to get requests from the website and know additional information.

We use 'admin' for the username and password.



This request has been received in the burp suite with the username and password as well in clear text.

Now we click on forward request in the burp suite and then we will be redirected to the admin user details. Here in the burp suite, we can clearly see the login details in clear text. this is the clear indication of the vulnerability which can lead to data breach, monitored, and manipulated as well.

Recommendations:

- Educate users and employees on secure data handling practices, such as recognizing secure websites (look for "https") and avoiding insecure Wi-Fi networks.
- Employ data masking or redaction techniques to replace sensitive data with placeholders or cryptographic representations during transmission.
- Implement encryption for data in transit. Use secure encryption protocols such as TLS/SSL to protect sensitive information during transmission.
- Use secure communication protocols for transmitting data, such as HTTPS for web applications, and ensure that the selected protocols are kept up-to-date.

9.Vulnerability name: Clickjacking

CWE: 1021

Description: The web application does not restrict or incorrectly restricts frame objects or UI layers that belong to another application or domain, which can lead to user confusion about which interface the user is interacting with.

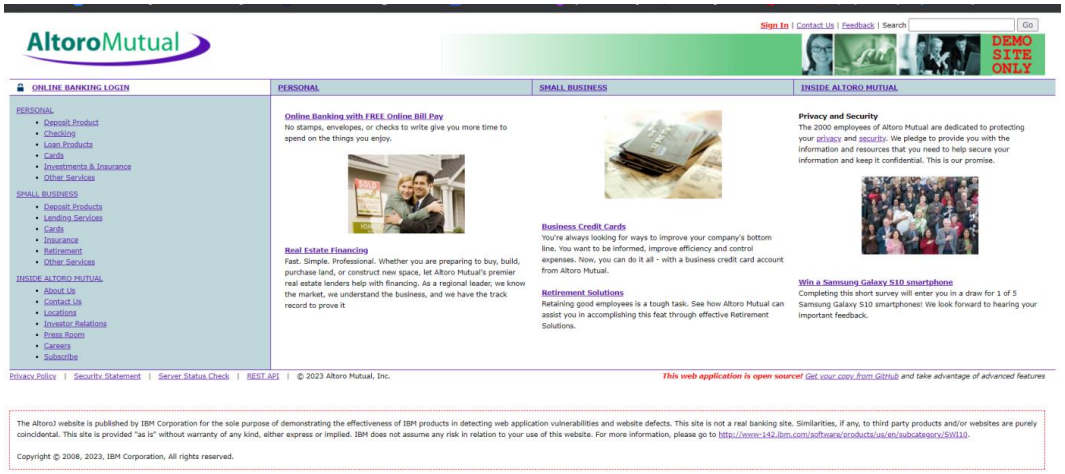
Business Impact: To effectively reduce the business consequences resulting from CWE-1021, organizations must prioritize the implementation of protective measures, such as frame-busting code. Simultaneously, educating users on safe browsing practices plays a crucial role in preventing clickjacking incidents. Additionally, routine security assessments and testing are pivotal for identifying and mitigating vulnerabilities associated with clickjacking. These actions are of paramount importance in upholding user trust, ensuring data protection, and safeguarding the organization's reputation.

Vulnerability path: http://testfire.net/

Vulnerability parameter: http://testfire.net/

Steps to reproduce:

Access the URL



Then take the URL and use it for the html code

we will be writing some html code to perform this vulnerability. we will be writing the code in vs code for better flexibility and functionality.

```
<html>
<body>
<title>Click jacking vulnerability</title>
<h2>This website is vulnerable to clickjacking</h2>
<iframe  src="http://testfire.net/"></iframe>
</body>
</html>
```

After the code has been written then we will be executing the code in the browser.

From this image we can see that the vulnerability has been found.

Recommendation:

- Set the X-Frame-Options HTTP response header to deny or same-origin to control how your site can be framed. This is supported by most modern browsers.
- Utilize frame-busting JavaScript code in web applications to prevent the embedding of your site within malicious iframes. This code can disrupt clickjacking attempts.
- Implement a Content Security Policy to restrict which domains can embed your site in iframes. This can help prevent unauthorized framing.
- Implement additional security controls to prevent UI redress attacks, such as clickjacking, within your web application.

10.Vulnerability name: web server allows password auto-completion

CWE: 310

Description:

Weaknesses in this category are related to the design and implementation of data confidentiality and integrity. Frequently these deal with the use of encoding techniques, encryption libraries, and hashing algorithms. The weaknesses in this category could lead to a degradation of the quality data if they are not addressed.

Business impact:

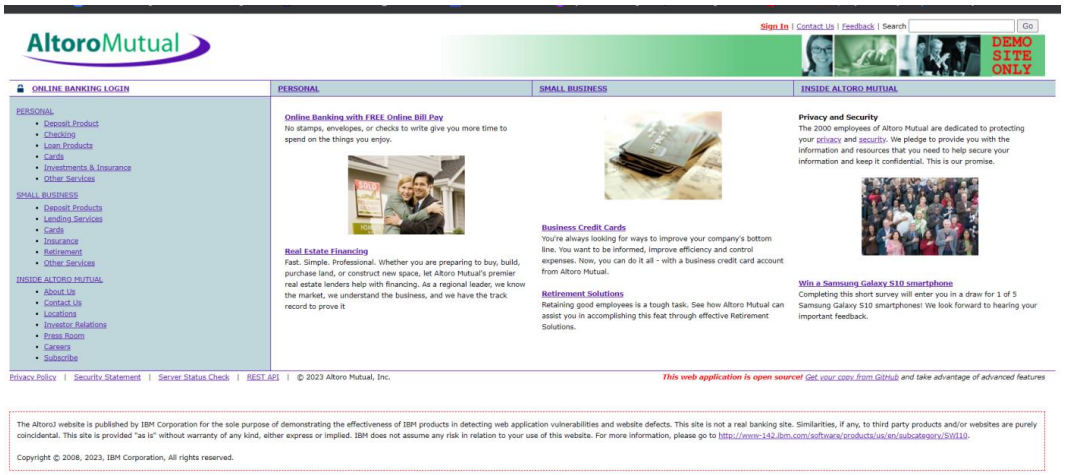
To effectively lessen the business consequences associated with CWE-310, organizations must prioritize the adoption of secure cryptographic practices. This entails ensuring proper password storage and robust encryption key management. Simultaneously, conducting routine security assessments and testing is pivotal in detecting and mitigating vulnerabilities related to cryptographic issues. Striving for compliance with pertinent data protection regulations and industry standards is equally essential. These actions are of paramount importance in the protection of sensitive data, the upholding of user trust, and the preservation of the organization's esteemed reputation.

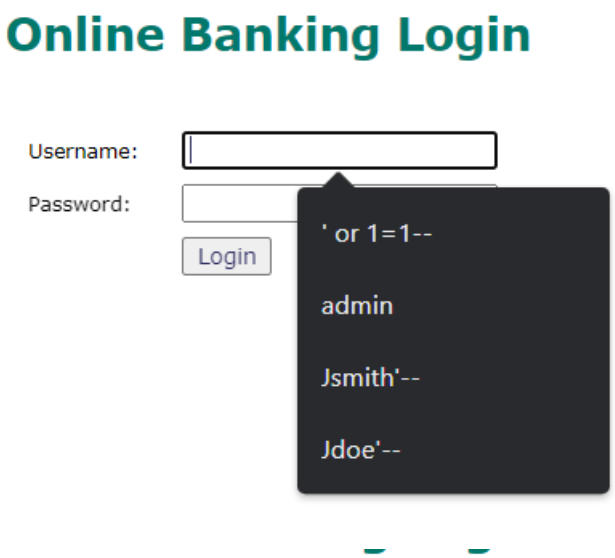
Vulnerability path: http://testfire.net/

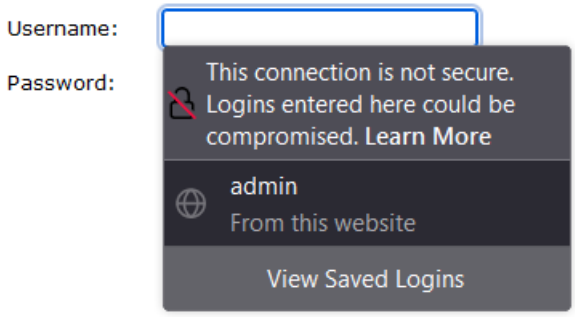
Vulnerability parameter: http://testfire.net/login.jsp

Steps to reproduce:

Access the URL







From this image we can see the usernames and the passwords getting auto filled. This is a potential vulnerability as this can be a doorway for attackers.

Responsibilities:

- Implement secure password storage mechanisms, such as using strong and salted cryptographic hashing algorithms like bcrypt or scrypt.
- Implement robust encryption key management practices, including secure key storage, key rotation, and access controls.
- Utilize well-established and proven cryptographic libraries and algorithms for encryption and decryption.
- Use secure encryption protocols, like TLS/SSL, for data transmission over networks to protect data in transit.