



# OWASP TOP 10 Vulnerabilities

Vulnerability Assessment & Penetration  
Testing Report

Vulnerability Assessment  
& Penetration Testing  
on **Altoro Mutual, Inc.**

**Performed By:**

*Mr. Dhruv Makani*

*Ms. Malini*

*Mr. Sarim Syed*

*Mr. Suhas Dhole*

## Acknowledgement

As the team leader, Mr. Suhas Dhole would like to express our sincere gratitude to Mr. Altaf Balsing for their guidance and support throughout our cyber-security internship project at Quantum Learnings. Mr. Altaf Balsing provided valuable insights and direction that helped us to complete this project successfully.

We also want to thank our colleagues at Quantum Learnings for their assistance and valuable insights during the project. We couldn't have completed this project without their help and encouragement. In particular, we want to thank Mr. Dhruv Makani, Ms. Malini, Mr. Sarim Syed for their support and contribution to the project.

We also want to acknowledge the support of our families and friends, who provided encouragement and motivation throughout the project. Thank you all for your help and support. This internship has been a valuable learning experience for Mr. Suhas Dhole and Mr. Dhruv Makani, Ms. Malini, Mr. Sarim Syed, and we are grateful for the opportunity to work with such a talented and dedicated team.

## INSTRUCTIONS

This report based on Open Web Application Security Project, Where, scanning and finding the defects in Web Applications based on TOP 10 OWASP like, Broken Access Control, Injection, Cross Site Scripting, Server-Side Request Forgery, etc. which is available on [owasp.org](https://owasp.org).

In this report we have performed Vulnerability Assessment & Penetration Testing (VAPT) on **Altoro Mutual, Inc.** by using various kinds of web application penetration techniques that hackers use to compromise web applications and also about how to secure them.

### Hosts

HOST: Altoro Mutual, Inc.

SCOPE: <https://demo.testfire.net/>

PORT: 443/ 80

Operating system: Unknown

Web server: Apache

Application Server: Tomcat

### Summary of security issues

High severity issues: 8

Medium severity issues: 1

Low severity issues: 5

-----

Total security issues: 14

# INDEX

1. **OWASP: BROKEN ACCESS CONTROL**
2. **VULNERABILITY NAME: BRUTE FORCE**
  
3. **OWASP: CRYPTOGRAPHIC FAILURES**
4. **VULNERABILITY NAME: SSL/TLS BAR MITZVAH ATTACK VULNERABILITY**
  
5. **OWASP: SENSITIVE DATA EXPOSURE**
6. **VULNERABILITY NAME: INFORMATION DISCLOSURE**
7. **VULNERABILITY NAME: IIS SHORT FILENAME DISCLOSURE VULNERABILITY**
8. **VULNERABILITY NAME: INFORMATION DISCLOSURE: MICROSOFT ASP.NET DEBUG ENABLED**
9. **VULNERABILITY NAME: APPLICATION ERROR EXISTS ON TARGET SERVER**
10. **VULNERABILITY NAME: EMAIL ADDRESS MODEL EXISTS IN TARGET URL**
  
11. **OWASP: INJECTION**
12. **VULNERABILITY NAME: SQL INJECTION**
  
13. **OWASP: CROSS SITE SCRIPTING (XSS)**
14. **VULNERABILITY NAME: REFLECTED XSS**
15. **VULNERABILITY NAME: CLICKJACKING: X-FRAME INJECTION**
  
16. **OWASP: SECURITY MISCONFIGURATION**
17. **VULNERABILITY NAME: DIRECTORY LISTING/ BRUTE FORCING DEFAULT CREDENTIALS**
  
18. **OWASP: VULNERABLE AND OUTDATED COMPONENTS/ USING COMPONENTS WITH KNOWN VULNERABILITIES**
19. **VULNERABILITY NAME: SSL/TLS BAR MITZVAH ATTACK**
  
20. **OWASP: IDENTIFICATION AND AUTHENTICATION FAILURES/ BROKEN AUTHENTICATION AND SESSION MANAGEMENT**
21. **VULNERABILITY NAME: SSLv3 SERIOUS DESIGN DEFECT**
  
22. **OWASP: SECURITY LOGGING AND MONITORING FAILURES\*/ INSUFFICIENT LOGGING & MONITORING**
23. **VULNERABILITY NAME: INSUFFICIENT LOGGING & MONITORING**
  
24. **TOOLS USED & REFERENCES:**

# OWASP: Broken Access Control

**VULNERABILITY NAME:** *Brute Force*

**SEVERITY:** *Medium-Risk*

## DESCRIPTION:

A brute-force attack consists of an attacker submitting many passwords or passphrases with the hope of eventually guessing correctly. The attacker systematically checks all possible passwords and passphrases until the correct one is found.

## AFFECTED RESOURCES/URL:

<http://demo.testfire.net/bank/login.aspx>

**PARAMETER:** login.aspx

## IMPACT:

- Attackers guess the website's user name and password by repeatedly attempting to conduct form-based login using user name and password in the dictionary.
- User name and password enumeration threatens websites that have no restriction on failed login attempts.

## RECOMMENDATION:

1. Restrict login failures to lock the account when certain amount of login failures has reached.
  - The user-registered password must meet the intended complexity requirements. For example, the password must be a combination of no less than 8 characters of the following: lowercase letters, uppercase letters, digits, and special characters ~!@#\$\$%^&\*()-+\_. For users with unqualified passwords, the registration should not be allowed.
2. Increase the complexity of the password.
  - Users are advised to change the default passwords occurring during installation to make them satisfy the preceding requirements.

## POC:

Username		Password
admin		admin
jsmith		Demo1234
sspeed		Demo1234
tuser		tuser

The screenshot displays the Burp Suite Professional interface. The top menu bar includes Window, Help, Turbo Intruder, and various tool tabs like Intruder, Repeater, Sequencer, etc. The main window shows the 'Attack' tab with a table of results for an intruder attack on https://demo.testfire.net.


Request	Payload 1	Payload 2	Status	Error	Timeout	Length	Comment
113	jsmith	Demo1234	302			434	
114	admin	admin	302			366	
115	sspeed	Demo1234	302			339	
209	tuser	tuser	302			264	
219	tuser	tuser	302			264	
1	Demo1234	admin	302			228	

Below the table, the 'Request' and 'Response' tabs are visible. The 'Request' tab is selected, showing a raw HTTP request. The 'Response' tab is also visible, showing a raw HTTP response. The status bar at the bottom indicates 'Finished'.


# Online Banking Login

Username:

Password:



[Sign Off](#) | [Contact Us](#)



[PERSONAL](#)

[SMALL BUSINESS](#)

## Hello Sam Speed

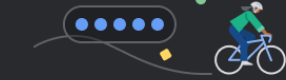
Welcome to Altoro Mutual Online.

View Account Details:

### Congratulations!

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

Click [Here](#) to apply.



Save password?

Username

Password

You can use saved passwords on any device. They're saved to Google Password Manager for suhasdhole.1799@gmail.com.

[t](#) | [Server Status Check](#) | [REST API](#) | © 2022 Altoro Mutual, Inc.

# OWASP: Cryptographic Failures

**VULNERABILITY NAME:** *SSL/TLS Bar Mitzvah Attack Vulnerability*

**SEVERITY:** **High-Risk**

## DESCRIPTION:

This vulnerability is caused by the weak and outdated RC4 cipher which reveals cipher texts in SSL/TLS-encrypted traffic, giving away the user name, password, credit card data, and other sensitive information to hackers.

**AFFECTED RESOURCES/URL:** <https://demo.testfire.net>

## IMPACT:

Scenario #1:

- An application encrypts credit card numbers in a database using automatic database encryption. However, this data is automatically decrypted when retrieved, allowing a SQL injection flaw to retrieve credit card numbers in clear text.

Scenario #2:

- A site doesn't use or enforce TLS for all pages or supports weak encryption. An attacker monitors network traffic (e.g., at an insecure wireless network), downgrades connections from HTTPS to HTTP, intercepts requests, and steals the user's session cookie. The attacker then replays this cookie and hijacks the user's (authenticated) session, accessing or modifying the user's private data. Instead of the above they could alter all transported data, e.g., the recipient of a money transfer.

Scenario #3:

- The password database uses unsalted or simple hashes to store everyone's passwords. A file upload flaw allows an attacker to retrieve the password database. All the unsalted hashes can be exposed with a rainbow table of pre-calculated hashes. Hashes generated by simple or



fast hash functions may be cracked by GPUs, even if they were salted.

## RECOMMENDATION:

- Prohibit the use of the RC4 cipher algorithm on the server.
- Prohibit the use of the RC4 cipher algorithm in the TLS configuration of the browser on the client.

## POC:

SSL Info:

Subject: /CN=demo.testfire.net

Ciphers: ECDHE-RSA-AES256-GCM-SHA384

Issuer: /C=GB/ST=Greater

Manchester/L=Salford/O=Sectigo Limited/CN=Sectigo  
RSA Domain Validation Secure Server CA

```
(kali㉿kali)-[~]  
$ nikto -h https://demo.testfire.net/  
- Nikto v2.1.6  
  
+ Target IP: 65.61.137.117  
+ Target Hostname: demo.testfire.net  
+ Target Port: 443  
  
+ SSL Info: Subject: /CN=demo.testfire.net  
Ciphers: ECDHE-RSA-AES256-GCM-SHA384  
Issuer: /C=GB/ST=Greater Manchester/L=Salford/O=Sectigo  
Limited/CN=Sectigo RSA Domain Validation Secure Server CA  
+ Start Time: 2022-12-21 20:45:29 (GMT5.5)
```

# OWASP: Sensitive Data Exposure

***VULNERABILITY NAME: Information Disclosure***

**SEVERITY: High-Risk**

## **DESCRIPTION:**

- Internet Information Services (IIS) is a set of Internet-based services developed by Microsoft for servers using Microsoft Windows.
- Microsoft IIS is prone to a file enumeration vulnerability that allows an attacker to enumerate files in the root directory of the network server.
- An attacker could exploit this vulnerability to launch a denial-of-service attack against .Net Framework in the IIS server by using the tilde (~) to guess or traverse filenames in the server.

## **AFFECTED RESOURCES/URL:**

[http://demo.testfire.net/index.jsp?content=inside\\_jobs.htm](http://demo.testfire.net/index.jsp?content=inside_jobs.htm)

**PARAMETER:** inside\_jobs.htm

## **IMPACT:**

- Sensitive Data Exposure happens when an application doesn't enough secure touchy data. The information can fluctuate and anything from passwords, meeting tokens, charge card information to private wellbeing information and more can be uncovered. As the finding just applies to delicate information, the potential effect is constantly viewed as high.
- What the information comprises of shifts thus does the effect. The peril lies in the information being uncovered, and the potential effect mirrors the information's affectability. Touchy information presentation happens because of not enough securing a database where data is put away.
- This may be a consequence of a huge number of things, for example, powerless encryption, no encryption, programming blemishes, or when somebody erroneously transfers information to an inaccurate database.
- Various sorts of information can be uncovered in a delicate information introduction. Banking account

numbers, Mastercard numbers, human services information, meeting tokens, Social Security number, street number, telephone numbers, dates of birth, and client account data, for example, usernames and passwords are a portion of the sorts of data that can be left uncovered.

- Most weaknesses inside this classification can't be filtered for because of two principle reasons:
  - To decide chance, it must be chosen what data is viewed as delicate, which can be a hard undertaking to complete consequently.
  - An outer pentester can't know whether inner information is scrambled or not as that isn't uncovered.

## RECOMMENDATION:

The initial step is to make sense of what information can be viewed as delicate and in this manner essential to secure. At the point when that is done, turn out every one of these information focuses and ensure that:

- The information is never put away in clear content.
- The information is never sent in clear content. Model among database and worker, or over the web.
- The calculations used to scramble the information are viewed as sufficient.
- The age of the keys is secure.
- Program headers are set to not store when the delicate information is introduced to end-client.
- Here are a few hints that can help.

## POC:

The response appears to contain suspicious comments which may help an attacker.

Note: Matches made within script blocks or files are against the entire content not only comments.

Screenshot of the rendered page containing this vulnerability.

Group	Date Posted	Title
Administration	Oct-23-2006	<a href="#">Executive Assistant</a>
Consumer Banking	Oct-19-2006	<a href="#">Teller</a>
Customer Service	Oct-26-2006	<a href="#">Customer Service Representative</a>
Marketing	Oct-25-2006	<a href="#">Loyalty Marketing Program Manager</a>
Risk Management	Oct-17-2006	<a href="#">Operational Risk Manager</a>
Sales	Oct-24-2006	<a href="#">Mortgage Lending Account Executive</a>

## ***VULNERABILITY NAME: IIS Short Filename Disclosure Vulnerability***

**SEVERITY:** **High-Risk**

### **DESCRIPTION:**

- Internet Information Services (IIS) is a set of Internet-based services developed by Microsoft for servers using Microsoft Windows.
- Microsoft IIS is prone to a file enumeration vulnerability that allows an attacker to enumerate files in the root directory of the network server.
- An attacker could exploit this vulnerability to launch a denial-of-service attack against .Net Framework in the IIS server by using the tilde (~) to guess or traverse filenames in the server.

**AFFECTED RESOURCES/URL:** <http://demo.testfire.net>

### **IMPACT:**

Brand Attacks that gain access into a system and are left to rummage around in unauthorized areas undetected can cause an immense amount of damage, sacrificing the integrity of an organization. Organizations suffer when they are the victim of a data breach.

### **RECOMMENDATION:**

- Disable NTFS 8.3 file compatibility. This function is enabled by default. However, it is unnecessary for most users to enable this function.
- For users of virtual host space, apply the following solutions:
- Change the following registry key to 1:
  - o HKLMSYSTEMCurrentControlSetControlFileSystemNtfsDisable8dot3NameCreation. This change can prevent creation of filenames in NTFS8.3 format, while existing short filenames cannot be removed.
  - o If asp.net is not required in your web environment, you choose "IIS Manager > Web Service Extensions > ASP.NET" and disable this function.
  - o Upgrade Microsoft .NET Framework to 4.0 or later.

- o Copy the content in the web folder to another directory, for example, from D:www to D:www.back. Then delete the original folder and rename the new directory as the original one. Only after you have completed the copy operation, the short filenames will disappear. For users of virtual host space, if this problem persists, contact the space provider.

**POC:**

(valid) [http://demo.testfire.net/\\*~1\\*\a.aspx](http://demo.testfire.net/*~1*\a.aspx)

(invalid) [http://demo.testfire.net/1234567890\\*~1\\*\a.aspx](http://demo.testfire.net/1234567890*~1*\a.aspx)

***VULNERABILITY NAME: Information disclosure:  
Microsoft ASP.NET Debug Enabled***

**SEVERITY:** Low-Risk

**DESCRIPTION:**

Microsoft ASP.NET is quite vulnerable to information disclosure attacks. Attackers can send a malicious request notifying whether to support debugging.

Attackers can send malicious requests through the word "DEBUG"

**AFFECTED RESOURCES/URL:** <http://demo.testfire.net>

**PARAMETER:** Request Method GET

**RECOMMENDATION:**

To disable debugging in ASP.NET, please edit your web.config file so that it has the following property:

```
<compilation debug="false" />
```

## ***VULNERABILITY NAME: Application Error Exists on Target Server***

**SEVERITY:** Low-Risk

### **DESCRIPTION:**

- If the attacker detects an application (such as the following examples) by forging requests containing parameters or parameter values that are not expected by the application, then the application may enter the vulnerable state. An attacker could obtain useful information from the application's response to the request, and exploit this information to identify weaknesses in the application.
- For example, if the parameter field should be a string enclosed in single quotes (as in the ASP script or SQL query), then injected single quotation marks will be terminate the string stream early, and thus changing the normal flow/grammar of the script.
- Another reason for disclosing important information in the error message is because of the configuration error of the scripting engine, Web server or database.
- The following are a number of different variants:
  - Exclude parameters
  - Exclude parameter values
  - Set parameter value to null
  - Set parameter value to numeral overflow (+/- 999999999)
  - Set parameter value to dangerous character, ' " ' \" ) ;
  - Add a string to a numeral parameter value

### **AFFECTED RESOURCES/URL:**

- <http://demo.testfire.net/comment.aspx>
- <http://demo.testfire.net/bank/login.aspx>

### **PARAMETER:**

- comment.aspx
- login.aspx

### **RECOMMENDATION:**

- Check the networking request to check whether all the expected parameters and values exist. When the parameter



is missing, send an appropriate error message, or use the default value.

- The application should verify that their input is composed of (decoded) valid characters. For example, input values containing empty bytes (encoded as % 00), single quotation marks, quotation marks, etc. should be rejected.
- Ensure that the scope and type of the value are in line with expectations. If the application expects that a specific parameter contains values in a collection, then the application should ensure that the received value actually belongs to the collection.
- For example, if the expected value is in the range [10-99], then the value should indeed be numbers in the range [10-99].
- Verify that the data belongs to the collection provided to the client.
- Do not generate debug error messages and anomalies in the production environment.

#### POC:

<http://demo.testfire.net/comment.aspx>

(POST) reset=Clear+Form&name=13800138000&email\_addr=atestu@example.com&comments=atestu&subject=atestu

<http://demo.testfire.net/comment.aspx>

(POST) name=13800138000&email\_addr=atestu@example.com&submit=Submit&comments=atestu&subject=atestu

<http://demo.testfire.net/bank/login.aspx>

(POST) btnSubmit=Login&passw="() [] {} ' / @ ^ \* \$ ; # , & uid=atestuser

***VULNERABILITY NAME: Email Address Model Exists in Target URL***

**SEVERITY:** Low-Risk

**DESCRIPTION:**

Searching Internet websites, Spambot begins to find email address to compose an address list for sending emails (spam). Responses from one or more emails address can be exploited to send spam. Some of the email addresses are for special use and are not accessible for others.

**AFFECTED RESOURCES/URL:**

- <http://demo.testfire.net/robots.txt>
- <http://demo.testfire.net/index.aspx>
- <http://demo.testfire.net/bank/mozxpath.js>
- <http://demo.testfire.net/default.htm>

**PARAMETER:**

- /robots.txt
- /index.aspx
- /mozxpath.js
- /default.htm

**RECOMMENDATION:**

Delete the email address from the Website so that malicious users cannot exploit it.

**POC:**

URL <http://demo.testfire.net/robots.txt>

Verify Message: test@test.com

Request Method: GET

URL <http://demo.testfire.net/index.aspx>

Verify Message: test@test.com

Request Method: GET

URL <http://demo.testfire.net/bank/mozxpath.js>

Verify Message: km0ti0n@gmail.com

Request Method: GET

URL <http://demo.testfire.net/default.htm>

Verify Message: skipfish@example.com

Request Method: GET

# OWASP: Injection

***VULNERABILITY NAME: SQL Injection***

**SEVERITY: High-Risk**

## **DESCRIPTION:**

- Sql injection (sqli) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database.
- It generally allows an attacker to view data that they are not normally able to retrieve.
- This might include data belonging to other users, or any other data that the application itself is able to access.
- In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.
- A successful sql injection attack can result in unauthorized access to sensitive data, such as passwords, credit card details, or personal user information.
- Many high-profile data breaches in recent years have been the result of sql injection attacks, leading to reputational damage and regulatory fines. In some cases, an attacker can obtain a persistent backdoor into an organization's systems, leading to a long-term compromise that can go unnoticed for an extended period.
- There are a wide variety of SQL injection vulnerabilities, attacks, and techniques, which arise in different situations. Some common SQL injection examples include:
  - Retrieving hidden data, where you can modify an SQL query to return additional results.
  - Subverting application logic, where you can change a query to interfere with the application's logic.
  - UNION attacks, where you can retrieve data from different database tables.
  - Examining the database, where you can extract information about the version and structure of the database.
  - Blind SQL injection, where the results of a query you control are not returned in the application's responses.

**AFFECTED RESOURCES/URL:** <http://demo.testfire.net/login.jsp>

**PARAMETER:** login.jsp

**IMPACT :**

- A successful sql injection attack can result in unauthorized access to sensitive data, such as passwords, credit card details, or personal user information
- Many high-profile data breaches in recent years have been the result of sql injection attacks, leading to reputational damage and regulatory fines.
- In some cases, an attacker can obtain a persistent backdoor into an organization's systems, leading to a long-term compromise that can go unnoticed for an extended period.

**RECOMMENDATION :**

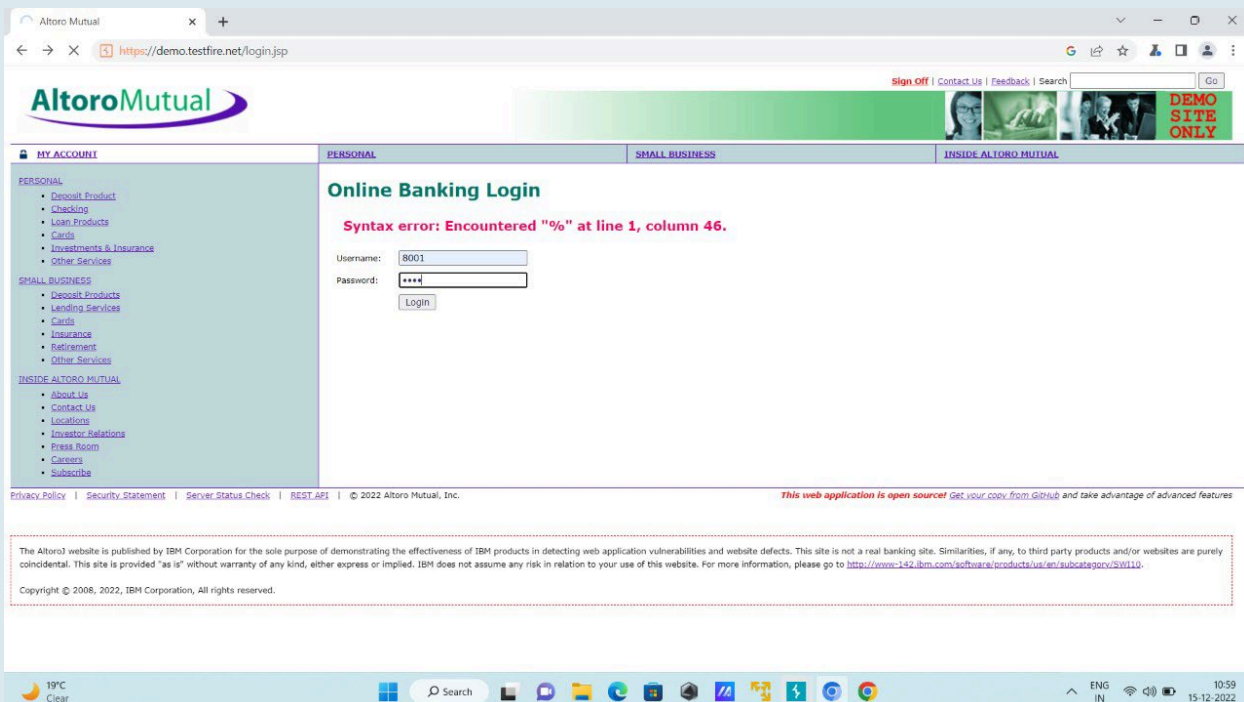
- Most instances of sql injection can be prevented by using parameterized queries (also known as prepared statements) instead of string concatenation within the query.
- The following code is vulnerable to SQL injection because the user input is concatenated directly into the query:
  - o `String query = "SELECT * FROM products WHERE category = '"+ input + "'";`
  - o `Statement statement = connection.createStatement();`
  - o `ResultSet resultSet = statement.executeQuery(query);`
  - o This code can be easily rewritten in a way that prevents the user input from interfering with the query structure:
  - o `PreparedStatement statement = connection.prepareStatement("SELECT * FROM products WHERE category = ?");`
  - o `statement.setString(1, input);`
  - o `ResultSet resultSet = statement.executeQuery();`
- Parameterized queries can be used for any situation where untrusted input appears as data within the query, including the WHERE clause and values in an INSERT or UPDATE statement. They can't be used to handle untrusted input in other parts of the query, such as table or column names, or the ORDER BY clause.
- Application functionality that places untrusted data into those parts of the query will need to take a different

approach, such as white-listing permitted input values, or using different logic to deliver the required behavior.

- For a parameterized query to be effective in preventing SQL injection, the string that is used in the query must always be a hard-coded constant, and must never contain any variable data from any origin.
- Do not be tempted to decide case-by-case whether an item of data is trusted, and continue using string concatenation within the query for cases that are considered safe. It is all too easy to make mistakes about the possible origin of data, or for changes in other code to violate assumptions about what data is tainted.

## POC :

Was able to easily get through the login page of demo.testfire.net, accessible at <http://demo.testfire.net/login.jsp> by using the payload ' Or '1'='1.



**Burp Suite Community Edition v2022.11.4 - Temporary Project**

**HTTP History**

Request	Position	Method	URL	Status	Size	Comment
186	2	POST	/doLogin HTTP/1.1	200	1208	
146	2	POST	/doLogin HTTP/1.1	200	1208	
147	2	POST	/doLogin HTTP/1.1	200	1208	
164	2	POST	/doLogin HTTP/1.1	200	1208	
173	2	POST	/doLogin HTTP/1.1	200	1208	
0	1	GET	/	302	145	
1	1	GET	/	302	145	
2	1	GET	/	302	145	
3	1	GET	/	302	145	
4	1	GET	/	302	145	
5	1	GET	/	302	145	

**Request Details**

Request: POST /doLogin HTTP/1.1

Host: demo.testfire.net

Cookie: JSESSIONID=P4P7D0AA610113273DFF546A1B0461; AltoraAccount=

Content-Type: application/x-www-form-urlencoded

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.95 Safari/537.36

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.9

Sec-Fetch-Site: same-origin

Sec-Fetch-Mode: navigate

Sec-Fetch-User: ?1

Sec-Fetch-Dest: document

Referer: https://demo.testfire.net/login.jsp

Upgrade-Insecure-Requests: 1

Origin: https://demo.testfire.net

Content-Type: application/x-www-form-urlencoded

Accept-Encoding: gzip, deflate

Accept-Language: en-US,en;q=0.9

Connection: close

uid=0001pass=0000&btnSubmit=Login

**Response Details**

Response: 200 OK

Content-Type: text/html

Content-Length: 59

Set-Cookie: JSESSIONID=P4P7D0AA610113273DFF546A1B0461

Cache-Control: max-age=0

Sec-Ch-Ua: "NotA\_Brand";v="8", "Chromium";v="108"

Sec-Ch-Ua-Mobile: ?0

Sec-Ch-Ua-Platform: "Windows"

Upgrade-Insecure-Requests: 1

Origin: https://demo.testfire.net

Content-Type: application/x-www-form-urlencoded

Accept-Encoding: gzip, deflate

Accept-Language: en-US,en;q=0.9

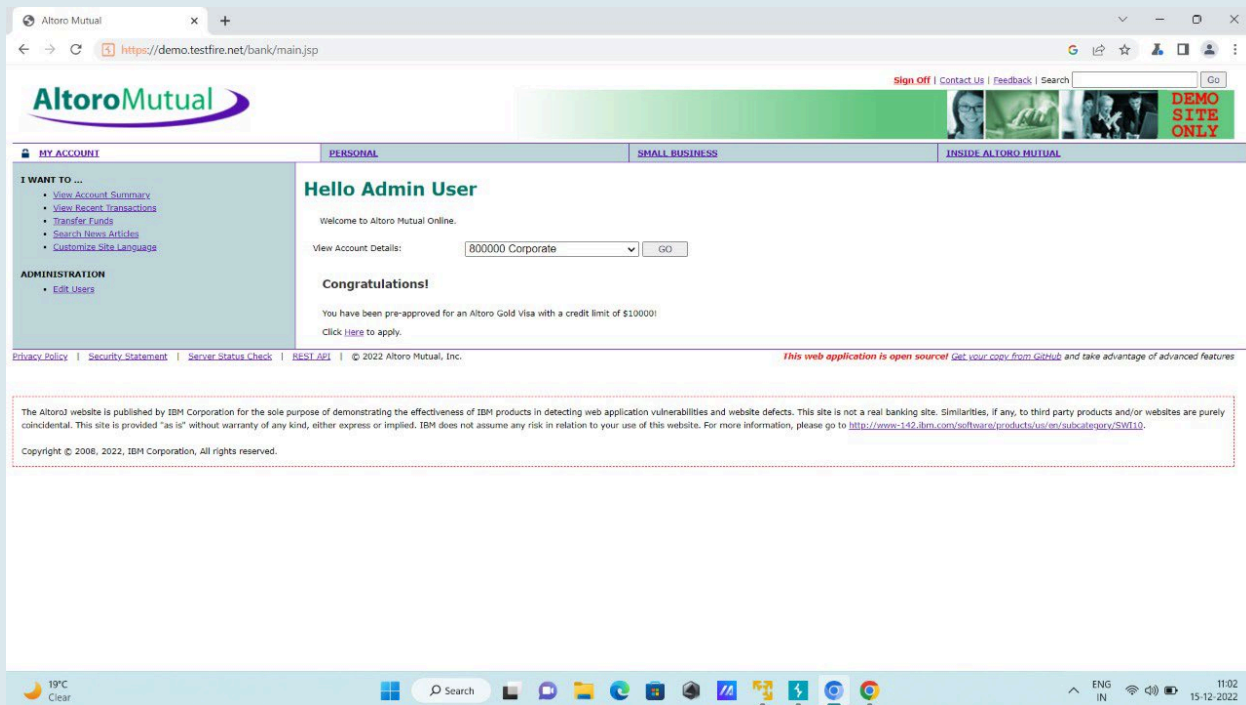
Connection: close

uid=0001pass=0000&btnSubmit=Login



19°C Clear Search [Taskbar Icons] ENG IN 11:01 15-12-2022





# OWASP: Cross Site Scripting (XSS)

**VULNERABILITY NAME:** *Reflected XSS*

**SEVERITY:** **High-Risk**

## DESCRIPTION:

Cross-site scripting (also known as XSS) is a web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application. It allows an attacker to circumvent the same origin policy, which is designed to segregate different websites from each other. Cross-site scripting vulnerabilities normally allow an attacker to masquerade as a victim user, to carry out any actions that the user is able to perform, and to access any of the user's data. If the victim user has privileged access within the application, then the attacker might be able to gain full control over all of the application's functionality and data.

## AFFECTED RESOURCES/URL:

- <https://demo.testfire.net/search.jsp>
- <https://demo.testfire.net/login.aspx>
- <https://demo.testfire.net/comment.aspx>

## PARAMETER:

- search.jsp  
login.aspx  
POST /sendFeedback  
name= XSS

## IMPACT:

- Cross site scripting (XSS) steals information from users through exploitation of website vulnerabilities. Users often click links on the page while browsing a website, using Instant Messaging software, or reading emails. Attackers can embed malicious code into the link and then steal user information or execute malicious code on the terminal user system.
- Main problems caused by XSS attacks include:
  - Account hijacking—attackers can hijack user sessions before session cookie expires, and operate with

- privileges of a login user, such as issuing database query and viewing result.
- o Malicious script execution—users can mistakenly execute JavaScript, VBScript, ActiveX, HTML and even Flash content that are embedded into the dynamically generated web page.
- o Worm spread—through Ajax applications, XSS can be spread like a virus. XSS loading can automatically embed itself to a page, and then easily embed itself to the same host, without any manual refresh of the page. Therefore, XSS can send multiple requests in the complicated HTTP mode, and spread itself invisibly.
- o Information theft—attackers can connect users to the malicious server of the attacker through website redirection and forgery, and obtain any information a user has typed.
- o Denial of service—attackers use malformed requests on the website containing XSS vulnerabilities to cause the website to perform self-queries again and again, leading to denial of service.
- o Browser redirection—on some websites that use frames, users may have already been redirected to a malicious website without notice, because the address in the address bar does not change.
- The reason is that not the whole page is redirected, but the JavaScript frame is executed.
- Control user setting—attackers can change user settings on purpose.

## RECOMMENDATION:

- Preventing cross-site scripting is trivial in some cases but can be much harder depending on the complexity of the application and the ways it handles user-controllable data.
- In general, effectively preventing XSS vulnerabilities is likely to involve a combination of the following measures:
- Filter input on arrival.
- At the point where user input is received, filter as strictly as possible based on what is expected or valid input.
- Encode data on output.

- At the point where user-controllable data is output in HTTP responses, encode the output to prevent it from being interpreted as active content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding.
- Use appropriate response headers.
- To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, you can use the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend.
- Content Security Policy.
- As a last line of defense, you can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occur.

### POC:

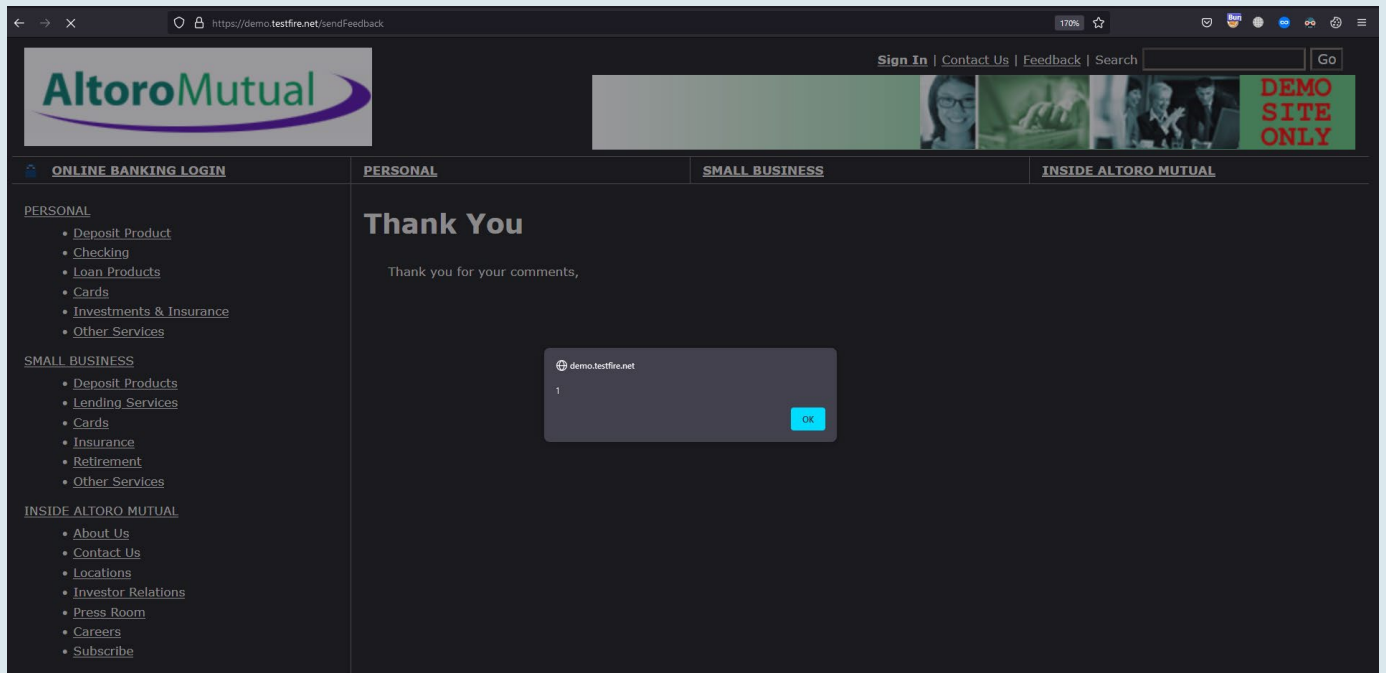
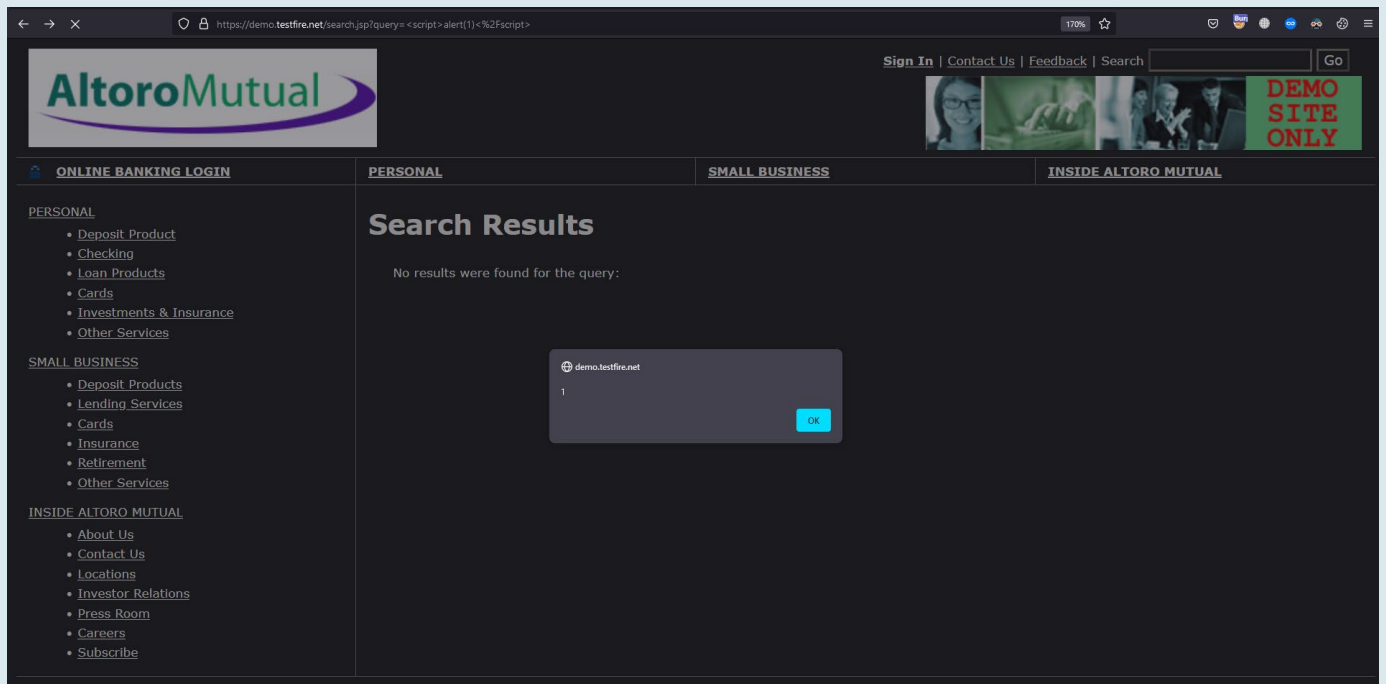
- You can confirm most kinds of XSS vulnerability by injecting a payload that causes your own browser to execute some arbitrary JavaScript. It's long been common practice to use the `alert()` function for this purpose because it's short, harmless, and pretty hard to miss when it's successfully called. In fact, you solve the majority of our XSS labs by invoking `alert()` in a simulated victim's browser.
- Unfortunately, there's a slight hitch if you use Chrome. From version 92 onward (July 20th, 2021), cross-origin iframes are prevented from calling `alert()`. As these are used to construct some of the more advanced XSS attacks, you'll sometimes need to use an alternative PoC payload. In this scenario, we recommend the `print()` function.
- Using Search Bar to find XSS Vulnerability

### Payloads:

```
<script>alert("XSS")</script>
```

```
<svg onload=confirm("XSS")>
```

```
<img src=x onerror=confirm("XSS")>
```



## ***VULNERABILITY NAME: Clickjacking: X-Frame Injection***

**SEVERITY: High-Risk**

### **DESCRIPTION:**

- Clickjacking is an interface-based attack in which a user is tricked into clicking on actionable content on a hidden website by clicking on some other content in a decoy website. Consider the following example:
- A web user accesses a decoy website (perhaps this is a link provided by an email) and clicks on a button to win a prize. Unknowingly, they have been deceived by an attacker into pressing an alternative hidden button and this results in the payment of an account on another site. This is an example of a clickjacking attack. The technique depends upon the incorporation of an invisible, actionable web page (or multiple pages) containing a button or hidden link, say, within an iframe. The iframe is overlaid on top of the user's anticipated decoy web page content. This attack differs from a CSRF attack in that the user is required to perform an action such as a button click whereas a CSRF attack depends upon forging an entire request without the user's knowledge or input.
- Protection against CSRF attacks is often provided by the use of a CSRF token: a session-specific, single-use number or nonce. Clickjacking attacks are not mitigated by the CSRF token as a target session is established with content loaded from an authentic website and with all requests happening on-domain. CSRF tokens are placed into requests and passed to the server as part of a normally behaved session. The difference compared to a normal user session is that the process occurs within a hidden iframe.

### **AFFECTED RESOURCES/URL:**

- <https://demo.testfire.net/search.jsp>
- <https://demo.testfire.net/feedback.jsp>

### **PARAMETER:**

- search.jsp  
POST /sendFeedback  
name= Clickjacking

email\_addr= Clickjacking

### **IMPACT :**


The Impact of Clickjacking The hacker has several ways they can use the redirected clicks for their own gain. A common form of clickjacking involves mirroring a login and password form on a website.

### **RECOMMENDATION :**


- We have discussed a commonly encountered browser-side prevention mechanism, namely frame busting scripts. However, we have seen that it is often straightforward for an attacker to circumvent these protections. Consequently, server driven protocols have been devised that constrain browser iframe usage and mitigate against clickjacking.
- Clickjacking is a browser-side behavior and its success or otherwise depends upon browser functionality and conformity to prevailing web standards and best practice. Server-side protection against clickjacking is provided by defining and communicating constraints over the use of components such as iframes. However, implementation of protection depends upon browser compliance and enforcement of these constraints. Two mechanisms for server-side clickjacking protection are X-Frame-Options and Content Security Policy.

### **POC :**

Payload: <iframe id="evil" src="https://evil.com" sandbox="allow-forms"></iframe>



[Sign In](#) | [Contact Us](#) | [Feedback](#) | Search



DEMO  
SITE  
ONLY

<a href="#">ONLINE BANKING LOGIN</a>	<a href="#">PERSONAL</a>	<a href="#">SMALL BUSINESS</a>	<a href="#">INSIDE ALTORO MUTUAL</a>
--------------------------------------	--------------------------	--------------------------------	--------------------------------------

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](#)

## Search Results

No results were found for the query:

www.evil.com

we get it... daily

January 15, 2022

Content

[Privacy Policy](#) | [Security Statement](#) | [Server Status Check](#) | [REST API](#) | © 2022 Altoro Mutual, Inc.

**This web application is open source!** [Get your copy from GitHub](#) and take advantage of advanced features



# OWASP: Security Misconfiguration

***VULNERABILITY NAME: Directory Listing/ Brute Forcing Default Credentials***

**SEVERITY:** Low-Risk

## **DESCRIPTION:**

- Security misconfigurations are security controls that are inaccurately configured or left insecure, putting your systems and data at risk.
- Basically, any poorly documented configuration changes, default settings, or a technical issue across any component in your endpoints could lead to a misconfiguration.
- Such Security misconfiguration can be achieved if attacker gains the admin privilege and Compromise the security and privacy of data.
- A Default Credential vulnerability is a type of vulnerability in a computing device that most commonly affects devices having some pre-set (default) administrative credentials to access all configuration settings.
- There are several Proof-of-Concept (POC), as well as real world worms running across internet, which are configured to search for systems set with a default username and password. Voyager Alpha Force, Zotob, and MySpooler are a few examples of POC malware which scan the Internet for specific devices, and try to login using the default credentials.

## **AFFECTED RESOURCES/URL:**

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

**PARAMETER:** login.jsp?uid=test&pass=1234

## **IMPACT:**

- A successful brute force attack for default credentials can result in unauthorized access to sensitive data, such as admin controls, credit card details, or personal user information.

- When not changed, default credentials make an organization more vulnerable to potential cyberattacks. Attackers can easily obtain these standard login details, allowing them access to the devices on your network - usually with admin rights - and leaving them open to takeover.

### **RECOMMENDATION:**

- To prevent such type of misconfiguration of security , Change or unify Admin Credentials from Default Credentials
- Try to adopt 8-length password policy with unique and difficult to crack passwords by brute forcers.

### **POC:**

The length sections as brute force attempted 36 combinations of uid and passwords , attempt no. 14 and 16 gave 283 length which differs from 145 length common for any other attempted combinations. Which predicting that these 2 combinations are success for brute forcing and giving us the default credentials.

5. Intruder attack of https://demo.testfire.net - Temporary attack - Not saved to project file

Results Positions Payloads Resource Pool Options

Filter: Showing all items

Request	Payload 1	Payload 2	Status	Error	Timeout	Length	Comment
2	salmon	frog	500	<input type="checkbox"/>	<input type="checkbox"/>	3563	
51	admin	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	280	
71	tuser	tuser	302	<input type="checkbox"/>	<input type="checkbox"/>	181	
0			302	<input type="checkbox"/>	<input type="checkbox"/>	145	

Request Response

Pretty Raw Hex

```

6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: http://99gocbo0y4tagdccogl413bq5hb9g34s.oastify.com/ref
9 Content-Type: application/x-www-form-urlencoded
10 Content-Length: 37
11 Origin: https://demo.testfire.net
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Te: trailers
18 Connection: close
19 Cache-Control: no-transform
20 X-Client-IP: spoofed.2uxhx49tjxe316x5996xmwwjqaw21vpk.oastify.com
21 CF-Connecting_IP: spoofed.85bn8akzu3p9cc8bkfh3x27plg78c30s.oastify.com
22 Contact: root@4y3jl6dvnzi55817dbazqy0luc0450tp.oastify.com
23 From: root@5x3k07cwm0h64908cc90pzzmtdz542sr.oastify.com
24 X-Forwarded-For: spoofed.ub99ewql0pvviyexqlnp3odb72duit6i.oastify.com
25 X-Originating-IP: spoofed.9gqojbv0540andjcvgs483iqchi9n9by.oastify.com
26 X-Wap-Profile: http://cp5rse43e79dwgsf4j17h6rtlkrcwdk2.oastify.com/wap.xml
27 True-Client-IP: spoofed.19v0cnocyggtmgpcooslg1fb25tblgn4c.oastify.com
28 X-Real-IP: spoofed.at7pwc81i5db0ewd8h5514vrpiva0do2.oastify.com
29 Client-IP: spoofed.e0ft3gf5p9kf7i3hflc9s82vwm2e7iv7.oastify.com
30 Forwarded:
for=spoofed.8oxnra3zd389vcrb3f03g2qpkgq8vdj2.oastify.com;by=spoofed.8oxnra3zd389vcrb3f0
3g2qpkgq8vdj2.oastify.com;host=spoofed.8oxnra3zd389vcrb3f03g2qpkgq8vdj2.oastify.com
31
32 uid=admin&passw=admin&btnSubmit>Login
  
```

0 matches

Finished

By using this default credential vulnerability we just discovered in the system,

demo.testfire.net/bank/main.jsp

[Sign Off](#) | [Contact Us](#)

# AltoroMutual

**MY ACCOUNT** | **PERSONAL** | **SMALL BUSINESS**

**I WANT TO ...**

- [View Account Summary](#)
- [View Recent Transactions](#)
- [Transfer Funds](#)
- [Search News Articles](#)
- [Customize Site Language](#)

**ADMINISTRATION**

- [Edit Users](#)

## Hello Admin User

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](#) | © 2022 Altoro Mutual, Inc.

*This web application is open source! [Get your copy from GitHub](#) and take advantage of adv*

Now, we can easily misconfigured the System security and gain admin privileges and rights to have information about users can change their password or Add and delete users.

[Sign Off](#) | [Contact Us](#) | [Feedback](#) | Search

# 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](#)

**ADMINISTRATION**

- [Edit Users](#)

## Edit User Information

**Add an account to an existing user**

Users:  Account Types:

**Change user's password**

Users:  Password:  Confirm:

**Add a new user**

First Name:  Last Name:  Username:  Password:  Confirm:

It is highly recommended that you leave the username as first initial last name.

[Privacy Policy](#) | [Security Statement](#) | [Server Status Check](#) | [REST API](#) | © 2022 Altoro Mutual, Inc.

*This web application is open source! [Get your copy from GitHub](#) and take advantage of advanced features*

# OWASP: Vulnerable and Outdated Components/ Using Components with Known Vulnerabilities

**VULNERABILITY NAME:** *SSL/TLS Bar Mitzvah Attack*

**SEVERITY:** **High-Risk**

## **DESCRIPTION:**

You are likely vulnerable:

- If you do not know the versions of all components you use (both client-side and server-side). This includes components you directly use as well as nested dependencies.
- If the software is vulnerable, unsupported, or out of date. This includes the OS, web/application server, database management system (DBMS), applications, APIs and all components, runtime environments, and libraries.
- If you do not scan for vulnerabilities regularly and subscribe to security bulletins related to the components you use.
- If you do not fix or upgrade the underlying platform, frameworks, and dependencies in a risk-based, timely fashion. This commonly happens in environments when patching is a monthly or quarterly task under change control, leaving organizations open to days or months of unnecessary exposure to fixed vulnerabilities.
- If software developers do not test the compatibility of updated, upgraded, or patched libraries.

## **AFFECTED RESOURCES/URL:**

65.61.137.117:443

<http://demo.testfire.net/>

## **RECOMMENDATION:**

There should be a patch management process in place to:

- Remove unused dependencies, unnecessary features, components, files, and documentation.
- Continuously inventory the versions of both client-side and server-side components (e.g., frameworks, libraries)

and their dependencies using tools like versions, OWASP Dependency Check, retire.js, etc. Continuously monitor sources like Common Vulnerability and Exposures (CVE) and National Vulnerability Database (NVD) for vulnerabilities in the components. Use software composition analysis tools to automate the process. Subscribe to email alerts for security vulnerabilities related to components you use.

- Only obtain components from official sources over secure links. Prefer signed packages to reduce the chance of including a modified, malicious component (See A08:2021-Software and Data Integrity Failures).
- Monitor for libraries and components that are unmaintained or do not create security patches for older versions. If patching is not possible, consider deploying a virtual patch to monitor, detect, or protect against the discovered issue.
- Every organization must ensure an ongoing plan for monitoring, triaging, and applying updates or configuration changes for the lifetime of the application or portfolio.

#### POC:

Found some extremely high-risk issues related to security misconfiguration:

- Debugger is enabled
- Incorrect directory permissions
- Using default accounts and passwords
- Setup/config pages enabled

#### SSL Info:

Subject: /CN=demo.testfire.net

Ciphers: ECDHE-RSA-AES256-GCM-SHA384

Issuer: /C=GB/ST=Greater Manchester/L=Salford/O=Sectigo Limited/CN=Sectigo RSA Domain Validation Secure Server CA

```
(kali㉿kali)-[~]  
$ nikto -h https://demo.testfire.net/  
- Nikto v2.1.6  
  
+ Target IP:          65.61.137.117  
+ Target Hostname:    demo.testfire.net  
+ Target Port:        443  
  
+ SSL Info:           Subject:  /CN=demo.testfire.net  
                      Ciphers: ECDHE-RSA-AES256-GCM-SHA384  
                      Issuer:   /C=GB/ST=Greater Manchester/L=Salford/O=Sectigo  
                      Limited/CN=Sectigo RSA Domain Validation Secure Server CA  
+ Start Time:         2022-12-21 20:45:29 (GMT5.5)
```

# OWASP: Identification and Authentication Failures/ Broken Authentication and Session Management

**VULNERABILITY NAME:** *SSLv3 Serious Design Defect*

**SEVERITY:** **High-Risk**

**DESCRIPTION:**

- An SSLv3 vulnerability (CVE-2014-3566) affects all implementation of SSLv3. By exploiting this vulnerability, an attacker can obtain transfer data (such as cookies) via man-in-the-middle attacks (as long as both ends of the hijacked session use SSL 3.0).
- To avoid exploitation of this vulnerability, disable SSLv3 for both the server and client.

**AFFECTED RESOURCES/URL:** IP:65.61.137.117 ; PORT:443

**PARAMETER:** 443

**RECOMMENDATION:**

Currently, the vendor has not provided any patches to fix this issue.

=====

Workaround:

Modify the Apache configuration file to disable SSLv3:

/etc/apache2/vhosts.d/00\_default\_ssl\_vhost.conf

Add the following lines prior to SSLCipherSuite  
HIGH:!ADH:!aNULL:

SSLProtocol all -SSLv2 -SSLv3

SSLHonorCipherOrder on

Save the configuration file and restart Apache

**POC:**

Supported ciphers (by Protocol)  
TLSv1.0



ECDHE-RSA-AES256-SHA: LUCKY13

DHE-RSA-AES256-SHA: LUCKY13

ECDHE-RSA-AES128-SHA: LUCKY13

DHE-RSA-AES128-SHA: LUCKY13

TLSv1.1

ECDHE-RSA-AES256-SHA: LUCKY13

DHE-RSA-AES256-SHA: LUCKY13

ECDHE-RSA-AES128-SHA: LUCKY13

DHE-RSA-AES128-SHA: LUCKY13

TLSv1.2

ECDHE-RSA-AES256-GCM-SHA384

ECDHE-RSA-AES256-SHA384: LUCKY13

ECDHE-RSA-AES256-SHA: LUCKY13

DHE-RSA-AES256-GCM-SHA384

DHE-RSA-AES256-SHA256: LUCKY13

DHE-RSA-AES256-SHA: LUCKY13

ECDHE-RSA-AES128-GCM-SHA256

ECDHE-RSA-AES128-SHA256: LUCKY13

ECDHE-RSA-AES128-SHA: LUCKY13

DHE-RSA-AES128-GCM-SHA256

DHE-RSA-AES128-SHA256: LUCKY13

DHE-RSA-AES128-SHA: LUCKY13

# OWASP: Security Logging and Monitoring Failures\*/ Insufficient Logging & Monitoring

**VULNERABILITY NAME:** *Insufficient Logging & Monitoring*

**SEVERITY:** Low-Risk

## DESCRIPTION:

- Security Logging and Monitoring Failures(which was renamed from Insufficient Logging and Monitoring)is the Failure to sufficiently log, monitor, or report security events, such as login attempts, makes suspicious behavior difficult to detect and significantly raises the likelihood that an attacker can successfully exploit your application
- It is used to help detect, escalate, and respond to active breaches. Without logging and monitoring, breaches cannot be detected. Insufficient logging, detection, monitoring, and active response occurs any time:
- Auditable events, such as logins, failed logins, and high-value transactions, are not logged.
- Warnings and errors generate no, inadequate, or unclear log messages.
- Logs of applications and APIs are not monitored for suspicious activity.
- Logs are only stored locally.
- Appropriate alerting thresholds and response escalation processes are not in place or effective.
- Penetration testing and scans by dynamic application security testing (DAST) tools (such as OWASP ZAP) do not trigger alerts.
- The application cannot detect, escalate, or alert for active attacks in real-time or near real-time.
- You are vulnerable to information leakage by making logging and alerting events visible to a user or an attacker

**AFFECTED RESOURCES/URL:** <http://demo.testfire.net/login.jsp>

**PARAMETER:** login.jsp

## IMPACT :

- Failing to log errors or attacks and poor monitoring practices can introduce a human element to security risks.
- Threat actors count on a lack of monitoring and slower remediation times so that they can carry out their attacks before you have time to notice or react.
- It's essential to have functional logging and monitoring systems, as they provide logs and information to give timely alerts to the system if any malfunction or error occurs. This protects the system from further damage.
- However, these issues don't frequently cause any vulnerability. Logging and monitoring become especially important in tracing back when the system shows any abnormal behavior. Their failure or absence highly impacts transparency, visibility, and incident alerting.
- If the system doesn't maintain any logging mechanism, or these mechanisms fail, there is no audit trail for events and security analysis. Therefore, attackers can keep damaging our system because their identity and method of attacking cannot be easily determined.
- The illustration below shows how logs help identify the patterns. The illustration also provides information for system improvement and maintenance

## RECOMMENDATION :

The following measures can be taken to avoid logging and monitoring failures:

- Make sure that all login and failed attempts are logged properly.
- Maintain an updated copy of all the logs that are useful in case the server faces any issues.
- The logs should be kept in a formatted manner that can be used by other functions and log management solutions. Unformatted logs can be a burden to look into.
- Ensure that the monitoring and logging system alerts in real time. Alerting and alarming the system after the damage has been done is not beneficial.
- Protect the logs to ensure their integrity.

## POC :

The screenshot shows the Burp Suite Professional v2022.8.5 interface. The main window displays an Intruder attack on the URL `https://demo.testfire.net`. The attack is titled "5. Intruder attack of https://demo.testfire.net - Temporary attack - Not saved to project file". The "Results" tab is active, showing a list of payloads and their corresponding status codes, error messages, timeouts, and lengths.

Request	Payload 1	Payload 2	Status	Error	Timeout	Length	Comment
111	august	admin1234	302			228	
112	3333	admin1234	302			228	
113	canada	admin1234	302			228	
114	blazer	admin1234	302			228	
115	cumming	admin1234	302			228	
116	hunting	admin1234	302			228	
117	kitty	admin1234	302			228	
118	rainbow	admin1234	302			228	
119	admin	admin1234	302			228	
120	admin123	admin1234	302			228	

The selected payload (119) is "admin". The detailed view of the selected payload shows the following information:

- Result 119 | Intruder attack**
- Payload 1:** admin
- Payload 2:** admin1234
- Status:** 302
- Length:** 228
- Timer:** 273

The "Request" and "Response" tabs are visible. The "Response" tab shows the following details:

- Request:** HTTP/1.1 302 Found
- Response:** Server: Apache-Coyote/1.1, Set-Cookie: JSESSIONID=B6294C589DC6B4676C0A85F1D240C45C; Path=/; Secure; HttpOnly, Location: login.jsp, Content-Length: 0, Date: Wed, 28 Dec 2022 09:01:41 GMT, Connection: close

The "Raw" tab is selected, showing the raw HTTP response in hexadecimal and ASCII format. The "Search" bar at the bottom indicates 0 matches.

## TOOLS USED:

- Burp Suite - Application Security Testing Software
- OWASP ZAP - Zed Attack Proxy
- SQLmap - Automatic SQL injection and database takeover tool
- Nmap - the Network Mapper
- Nikto - web server scanner
- dirsearch - Web path discovery

## REFERENCES:

<https://bugcrowd.com/vulnerability-rating-taxonomy>

<https://owasp.org/www-project-top-ten/>

[https://owasp.org/Top10/A01\\_2021-Broken Access Control/](https://owasp.org/Top10/A01_2021-Broken_Access_Control/)

[https://owasp.org/Top10/A02\\_2021-Cryptographic Failures/](https://owasp.org/Top10/A02_2021-Cryptographic_Failures/)

[https://owasp.org/Top10/A03\\_2021-Injection/](https://owasp.org/Top10/A03_2021-Injection/)

[https://owasp.org/Top10/A05\\_2021-Security Misconfiguration/](https://owasp.org/Top10/A05_2021-Security_Misconfiguration/)

[https://owasp.org/Top10/A06\\_2021-Vulnerable and Outdated Components/](https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/)

[https://owasp.org/Top10/A07\\_2021-Identification&AuthFailures/](https://owasp.org/Top10/A07_2021-Identification&AuthFailures/)

[https://owasp.org/Top10/A09\\_2021-SecurityLogging&MonitoringFailures/](https://owasp.org/Top10/A09_2021-SecurityLogging&MonitoringFailures/)

<https://portswigger.net/web-security/sql-injection>

<https://portswigger.net/web-security/authentication>

<https://portswigger.net/web-security/information-disclosure>

<https://portswigger.net/web-security/cross-site-scripting>

<https://portswigger.net/web-security/clickjacking>

<https://portswigger.net/web-security/dom-based>

<https://portswigger.net/web-security/deserialization>

<https://github.com/payloadbox/xss-payload-list>

<https://github.com/swisskyrepo/PayloadsAllTheThings>

<https://github.com/danielmiessler/SecLists>

