Web Application Security Testing (*AltoroMutual*)

Web Application Security Testing (*AltoroMutual*)
- ❖ STATEMENT OF CONFIDENTIALITY
- ❖ INTRODUCTION
- ❖ DOCUMENT PROPERTIES
- ❖ VERSION CONTROL
- ❖ EXECUTIVE SUMMARY
  - Scope of work
  - Project objectives
  - Assumption
  - Timeline
  - Summary of Findings
  - Summary of Recommendation
- ❖ REPORTING
  - SQL Injections (Login bypass, Attack on Admin)
  - XSS
  - Web Application Potentially Vulnerable to Clickjacking
  - HTML Injections
  - Cookie Injection Scripting
  - Absence of Anti-CSRF Tokens
  - Web Server Transmits Cleartext Credentials
  - HTTP Cookie without 'Secure' attribute
  - HTTP Cookie without 'SameSite' attribute
  - Missing or Permissive "X-Frame-Options" and "CSP frame-ancestors" HTTP Response Headers
  - HyperText Transfer Protocol (HTTP) Information

STATEMENT OF CONFIDENTIALITY

The contents of this document were developed by Selivonchik S. I believe that the contents of this document are not proprietary or business confidential information. This information should be used for educational purposes only. This document may be transferred to any other person without my prior consent. In addition, any part of this document may be transmitted, reproduced, copied or distributed without prior consent.

The contents of this document do not constitute legal advice and should not be construed as such. This is the result of my own educational process. The assessment detailed here is conducted on a fictitious company for training and testing purposes, and the vulnerabilities do not impact external or internal infrastructure in any way.

INTRODUCTION

Black Box WAST

for altoro.testfire.net

FINAL TEST V1.0

January 24th, 2024

By: *Selivonchyk S.*

DOCUMENT PROPERTIES

Title	**Black Box Penetration** WAST **Report**	**Contacts**
Version	V1.0	
Author	Selivonchyk S.	
WAST-testers	Selivonchyk S.	
Reviewed By	Kutaisov M.	
Approved By	Kutaisov M.	
Classification	Not confidential	

VERSION CONTROL

Version	Date	Author	Description
V1.0	Jan 24th, 2024	Selivonchyk Sergey	WAST

Table of Content

STATEMENT OF CONFIDENTIALITY 2
INTRODUCTION 3
DOCUMENT PROPERTIES 4
VERSION CONTROL 5
EXECUTIVE SUMMARY 7

- Scope of work 8
- Project objectives 9
- Assumption 9
- Timeline 9
- Summary of Findings 9
- Summary of Recommendation 10

REPORTING 11

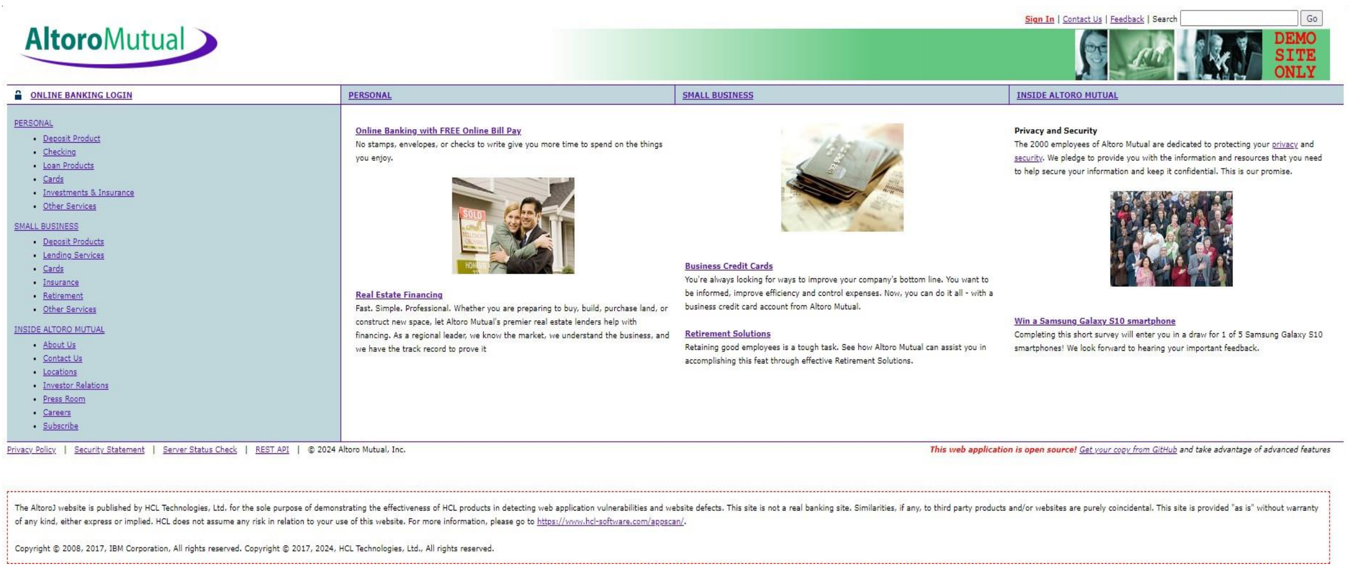
- SQL Injections (Login bypass, Attack on Admin) 11
- XSS 13
- Web Application Potentially Vulnerable to Clickjacking 15
- HTML Injections 17
- Cookie Injection Scripting 18
- Absence of Anti-CSRF Tokens 19
- Web Server Transmits Cleartext Credentials 22
- HTTP Cookie without 'Secure' attribute 23
- HTTP Cookie without 'SameSite' attribute 24
- Missing or Permissive "X-Frame-Options" and "CSP frame-ancestors" HTTP Response Headers 25
- HyperText Transfer Protocol (HTTP) Information 26

EXECUTIVE SUMMARY

Altoro Mutual Banking Service is a publicly available application designed to test security scanners. This application can be used to evaluate and calibrate both automated vulnerability scanners and students taking web application security testing courses. This report describes methods and techniques for detecting vulnerabilities through manual testing. As well as testing the skills of documenting all research results in an understandable and reproducible form and developing the skills to provide recommendations for correcting situations.

Scope of work

This work involves remotely testing a web application for real-world security vulnerabilities. Their existence must be confirmed, but without increasing attacks. The target runs on one available server located at http://altoro.testfire.net.



Online Banking with FREE Online Bill Pay
No stamps, envelopes, or checks to write give you more time to spend on the things you enjoy.

Real Estate Financing
Fast. Simple. Professional. Whether you are preparing to buy, build, purchase land, or construct new space, let Altoro Mutual's premier real estate lenders help with financing. As a regional leader, we know the market, we understand the business, and we have the track record to prove it

Business Credit Cards
You're always looking for ways to improve your company's bottom line. You want to be informed, improve efficiency and control expenses. Now, you can do it all - with a business credit card account from Altoro Mutual.

Retirement Solutions
Retaining good employees is a tough task. See how Altoro Mutual can assist you in accomplishing this feat through effective Retirement Solutions.

Privacy and Security
The 2000 employees of Altoro Mutual are dedicated to protecting your privacy and security. We pledge to provide you with the information and resources that you need to help secure your information and keep it confidential. This is our promise.

Win a Samsung Galaxy S10 smartphone
Completing this short survey will enter you in a draw for 1 of 5 Samsung Galaxy S10 smartphones! We look forward to hearing your important feedback.

Privacy Policy | Security Statement | Server Status Check | REST API | © 2024 Altoro Mutual, Inc.

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

The AltoroJ 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/appscan/.

Copyright © 2008, 2017, IBM Corporation, All rights reserved. Copyright © 2017, 2024, HCL Technologies, Ltd., All rights reserved.

Figure 1 *altoro.testfire.net*

The assessment was conducted from a "*black box*" perspective without any authority or prior knowledge of the internal environment. Only the URL of the server being tested is provided. No other information was expected.

In-Scope Assets

Host/URL/IP Address/Port	Description
`altoro.testfire.net`	Published by HCL Technologies, Ltd

> **Table 1:** *Scope details*

Project objectives

Identify vulnerabilities and potential flaws in its design. Testing was done remotely using a URL. Each identified vulnerability is assigned a threat-based risk rating and manually reviewed to determine its potential for exploitation and escalation.

Assumption

When writing this report, we assume that the resource has a special application designed for testing security scanners. This application can be used to evaluate and calibrate automated vulnerability scanners. It does not store or collect any sensitive data, and according to the home page, there is the following statement:

> The AltoroJ 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 warranties of any kind, express or implied. HCL does not accept any risk in connection with your use of this website. For more information, go to https://www.hcl-software.com/appscan/.

Timeline

The timeline of the test is as below:

Vulnerability detection	Start date/time	End date/time
WAST 1	01/23/2024	01/25/2024

> **Table 2:** *Web Application*

Summary of Findings

This table shows the number of alerts for each **risk** and **confidence level** included in the report.

Risk:	High	Medium	Low	Info	TOTAL
Amount	2	4	3	2	11

> **Table 3:** *Total Risk and Confidence Rating*

Summary of Recommendation

1. If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated.
2. To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature.
3. Assume all input is malicious. Use an "*accept known good*" input validation strategy, i.e., use a list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications or transform it into something that does.
4. **Strong input validation** - All user-controllable input must be validated and filtered for illegal characters as well as SQL content. Keywords such as `UNION`, `SELECT` or `INSERT` must be filtered in addition to characters such as a single-quote (`'`) or SQL-comments (`--`) based on the context in which they appear.
5. **Use of parameterized queries** or stored procedures.
6. **Use of custom error pages** - Attackers can glean information about the nature of queries from descriptive error messages.
7. Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.
8. Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script.
9. Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable.
10. Check the HTTP Referrer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.
11. Return the X-Frame-Options or Content-Security-Policy (*with the 'frame-ancestors' directive*) HTTP header with the page's response.
12. This prevents the page's content from being rendered by another site when using the frame or iframe HTML tags.
13. Carefully check each input parameter against a rigorous positive specification (*allowlist*) defining the specific characters and format allowed. All input should be neutralized, not just parameters that the user is supposed to specify, but all data in the request, including hidden fields, cookies, headers, the URL itself, and so forth. A common mistake that leads to continuing XSS vulnerabilities is to validate only fields that are expected to be redisplayed by the site.
14. Restrict access to the vulnerable application.
15. Store state information and sensitive data on the server side only.
16. Ensure that the system definitively and unambiguously keeps track of its own state and user state and has rules defined for legitimate state transitions.
17. Do not allow any application user to affect state directly in any way other than through legitimate actions leading to state transitions.
18. Make sure that every sensitive form transmits content over HTTPS.
19. Always set the secure attribute when the cookie should send via HTTPS only.
20. Check your server configuration to always use HTTPS.
21. Set a properly configured *X-Frame-Options* and *Content-Security-Policy frame-ancestor's* headers for all requested resources
22. The use of *X-Frame-Options* and *Content-Security-Policy frame-ancestors* allows developers of web content to restrict the usage of their application within the form of overlays, frames, or iFrames. The developer can indicate from which domains can frame the content.
23. A developer can use a "*frame-breaker*" script in each page that should not be framed.

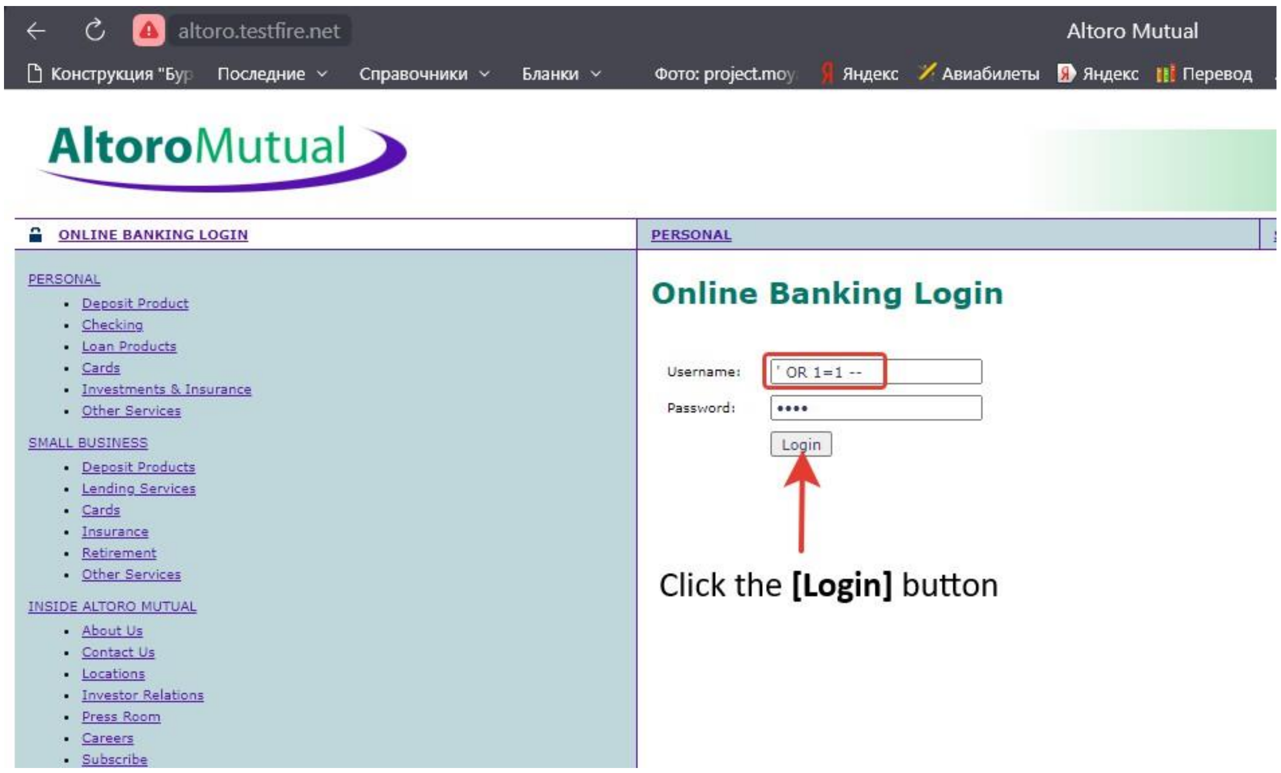
REPORTING

SQL Injections (*Login bypass, Attack on Admin*)

Synopsis	This attack exploits target software that constructs SQL statements based on user input. An attacker crafts input strings so that when the target software constructs SQL statements based on the input, the resulting SQL statement performs actions other than those the application intended. SQL Injection results from failure of the application to appropriately validate input.
Description	When specially crafted user-controlled input consisting of SQL syntax is used without proper validation as part of SQL queries, it is possible to glean information from the database in ways not envisaged during application design. Depending upon the database and the design of the application, it may also be possible to leverage injection to have the database execute system-related commands of the attackers' choice. SQL Injection enables an attacker to interact directly to the database, thus bypassing the application completely. Successful injection can cause information disclosure as well as ability to add or modify data in the database.
Prerequisites	- SQL queries used by the application to store, retrieve, or modify data. - User-controllable input that is not properly validated by the application as part of SQL queries
Risk Factor	*High*
Solution	- **Strong input validation** - All user-controllable input must be validated and filtered for illegal characters as well as SQL content. Keywords such as `UNION`, `SELECT` or `INSERT` must be filtered in addition to characters such as a single-quote (`'`) or SQL-comments (`--`) based on the context in which they appear. - **Use of parameterized queries** or stored procedures. - **Use of custom error pages** - Attackers can glean information about the nature of queries from descriptive error messages.

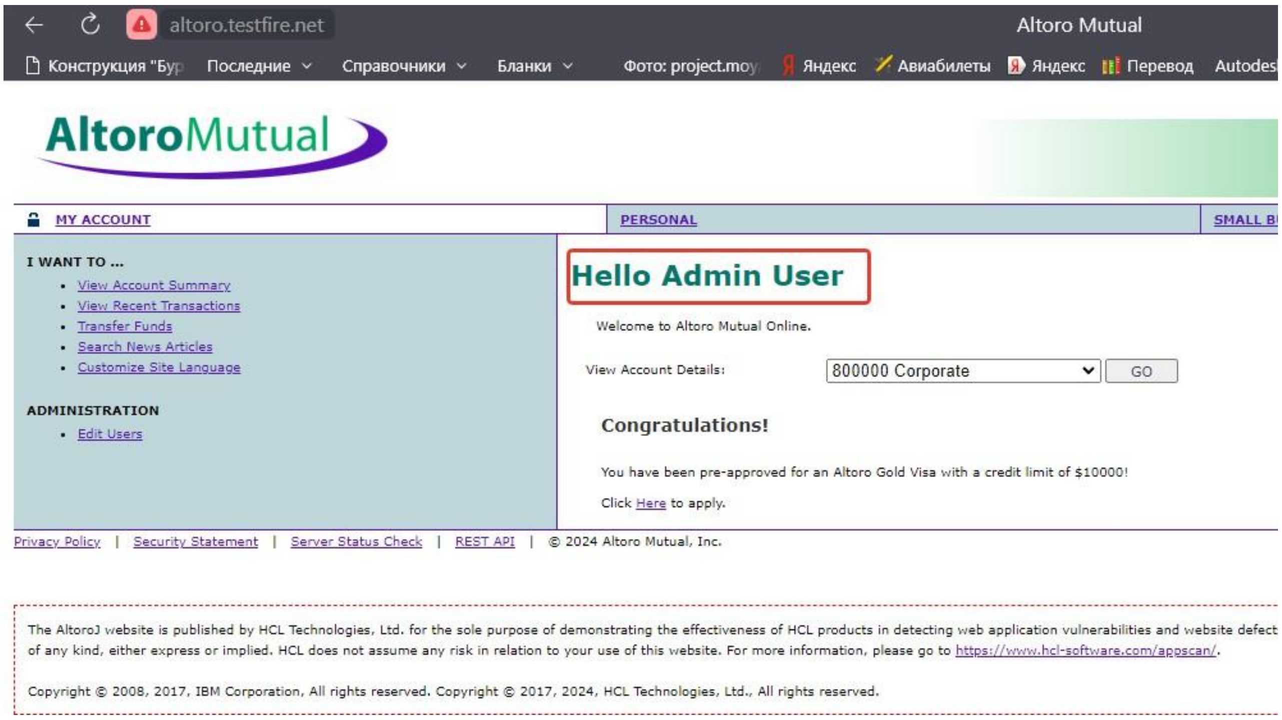
Proof Of Concept (PoC):

1. Open the website https://altoro.testfire.net.
2. Click the `[Sign in]` button.
3. Enter *Username:* `' OR 1=1 --` and *Password:* `q` to check for the SQLi vulnerability.



Username: ' OR 1=1 --
Password: ••••

4. Click the `[Login]` button.



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: 800000 Corporate

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 | © 2024 Altoro Mutual, Inc.

The AltoroJ 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 defect 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/appscan/.

Copyright © 2008, 2017, IBM Corporation, All rights reserved. Copyright © 2017, 2024, HCL Technologies, Ltd., All rights reserved.

5. Total:
   1. Successfully carried out an **attack on the administrator**,
   2. Successfully **bypassed the logging** system,
   3. All this became possible only thanks to the presence of the **SQL Injection** vulnerability.

XSS

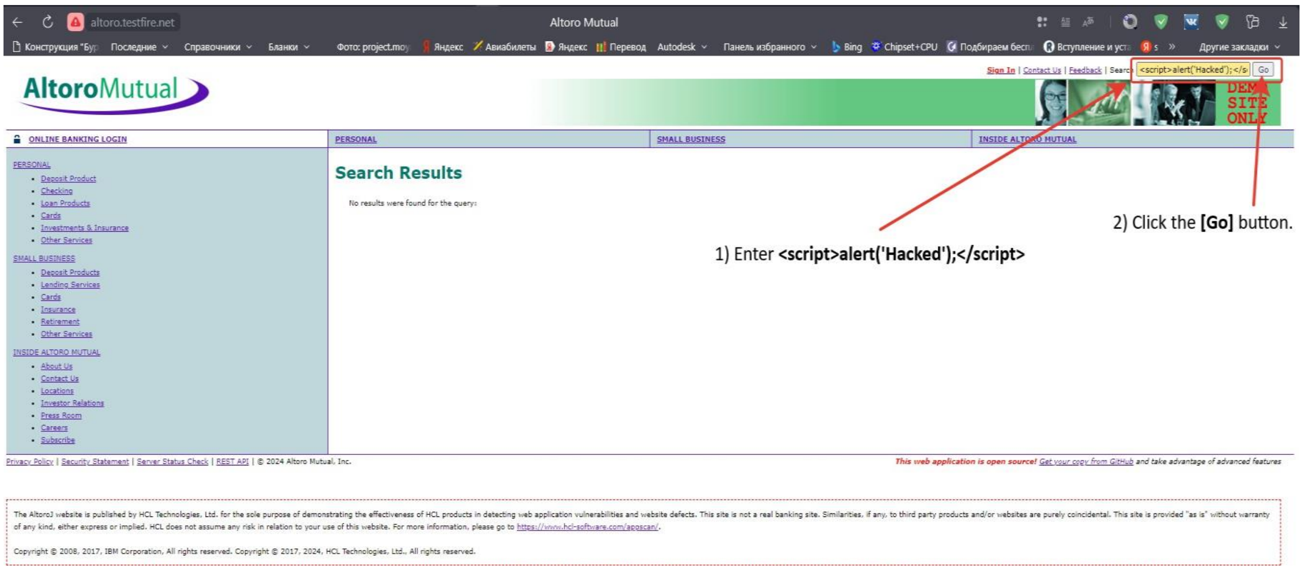
Synopsis	The web application doesn't neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users.
Description	Once the malicious script is injected, the attacker can perform a variety of malicious activities. The attacker could transfer private information, such as cookies that may include session information, from the victim's machine to the attacker. The attacker could send malicious requests to a web site on behalf of the victim, which could be especially dangerous to the site if the victim has administrator privileges to manage that site. Phishing attacks could be used to emulate trusted web sites and trick the victim into entering a password, allowing the attacker to compromise the victim's account on that web site. Finally, the script could exploit a vulnerability in the web browser itself possibly taking over the victim's machine, sometimes referred to as "*drive-by hacking*".
Prerequisites	- Untrusted data enters a web application, typically from a web request. - The web application dynamically generates a web page that contains this untrusted data. - During page generation, the application does not prevent the data from containing content that is executable by a web browser, such as JavaScript, HTML tags, HTML attributes, mouse events, Flash, ActiveX, etc. - A victim visits the generated web page through a web browser, which contains malicious script that was injected using the untrusted data. - Since the script comes from a web page that was sent by the web server, the victim's web browser executes the malicious script in the context of the web server's domain. - This effectively violates the intention of the web browser's same-origin policy, which states that scripts in one domain should not be able to access resources or run code in a different domain.
Risk Factor	*High*
Solution	- If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. - To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature - Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications or transform it into something that does.

Proof Of Concept (PoC):

1. Open the website https://altoro.testfire.net.
2. Enter the following code into the search bar.

```
<script>alert('Hacked');</script>
```

3. Click the `[Go]` button.



Search Results

No results were found for the query:

1) Enter **<script>alert('Hacked');</script>**

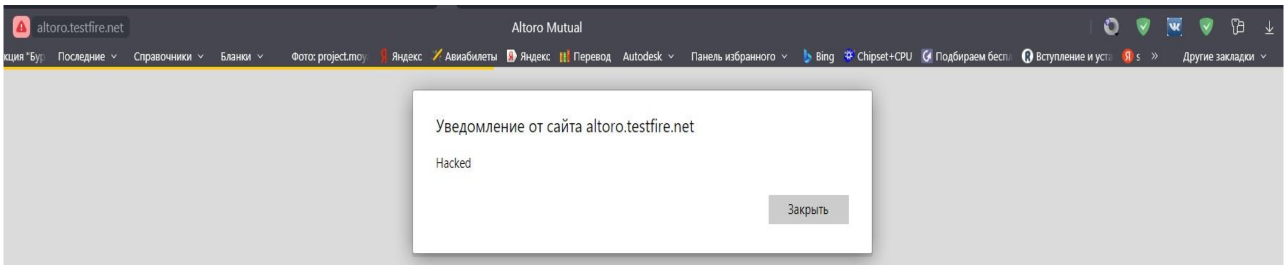
2) Click the **[Go]** button.

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

The AltoroJ 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/appscan/.

Copyright © 2008, 2017, IBM Corporation, All rights reserved. Copyright © 2017, 2024, HCL Technologies, Ltd., All rights reserved.

4. As a result, the message "*Hacked*" will appear.



Уведомление от сайта altoro.testfire.net

Hacked

Web Application Potentially Vulnerable to Clickjacking

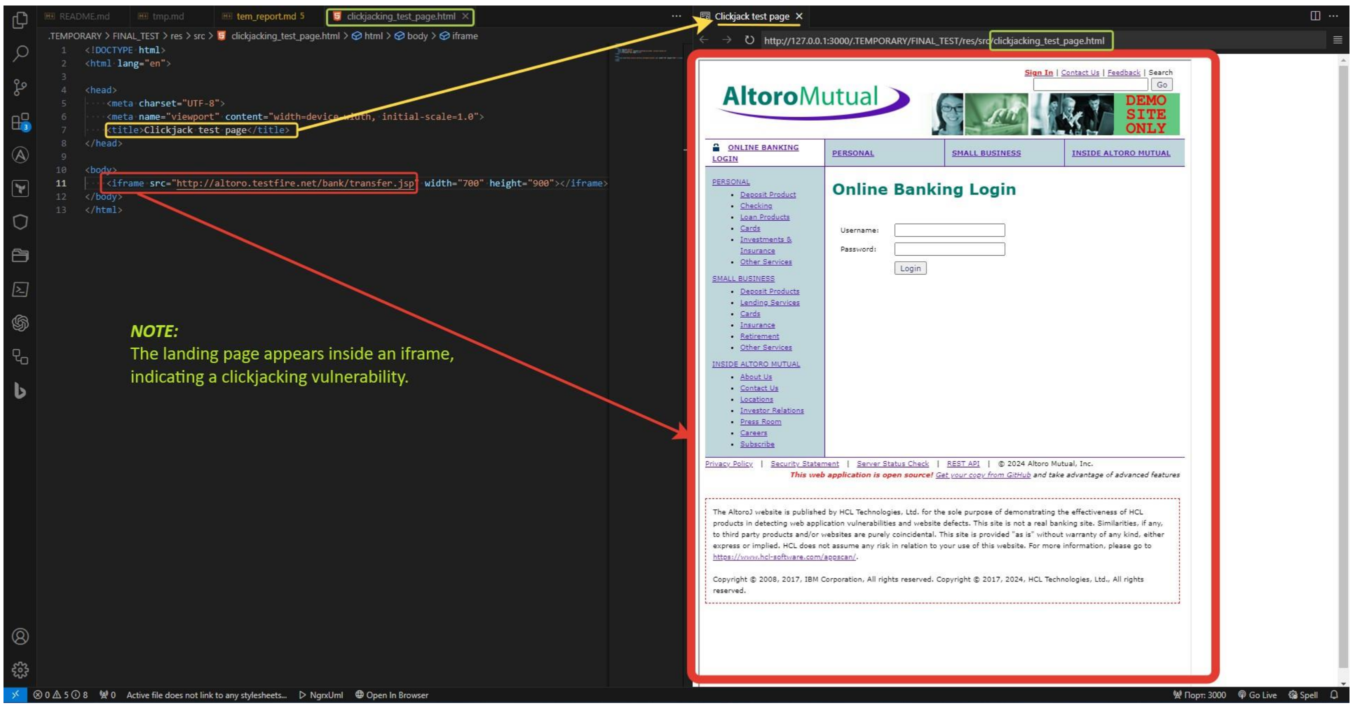
Synopsis	Clickjacking, a subset of UI redressing, is a malicious technique whereby a web user is deceived into interacting (*in most cases by clicking*) with something other than what the user believes they are interacting with.
Description	While being logged in to some target system, the victim visits the adversary's malicious site which displays a UI that the victim wishes to interact with. The clickjacked page has a transparent layer above the visible UI with action controls that the adversary wishes the victim to execute. The victim clicks on buttons or other UI elements they see on the page which triggers the action controls in the transparent overlaying layer. Depending on what that action control is, the adversary may have just tricked the victim into executing some potentially privileged (*and most certainly undesired*) functionality in the target system to which the victim is authenticated
Prerequisites	1. The victim is communicating with the target application via a web-based UI and not a thick client. 2. The victim's browser security policies allow at least one of the following JavaScript, Flash, iframes, ActiveX, or CSS. 3. The victim uses a modern browser that supports UI elements like clickable buttons (*i.e. not using an old text only browser*) 4. The victim has an active session with the target system. 5. The target system's interaction window is open in the victim's browser and supports the ability for initiating sensitive actions on behalf of the user in the target system
Risk Factor	*Medium*
Solution	6. Return the X-Frame-Options or Content-Security-Policy (*with the 'frame-ancestors' directive*) HTTP header with the page's response. 7. This prevents the page's content from being rendered by another site when using the frame or iframe HTML tags

Proof Of Concept (PoC):

1. Open any text editor.
2. Create a new HTML file.
3. Enter the following code.

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Clickjack test page</title>
</head>

<body>
    <iframe src="http://altoro.testfire.net/bank/transfer.jsp" width="700"
height="900"></iframe>
</body>
</html>
```

4. Save and open this page in your browser.
5. If the landing page appears, this tells us about a clickjacking vulnerability.



```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Clickjack test page</title>
</head>

<body>
    <iframe src="http://altoro.testfire.net/bank/transfer.jsp" width="700" height="900"></iframe>
</body>
</html>
```

NOTE:
The landing page appears inside an iframe, indicating a clickjacking vulnerability.

http://127.0.0.1:3000/.TEMPORARY/FINAL_TEST/res/src/clickjacking_test_page.html

Online Banking Login

Username:
Password:

Privacy Policy | Security Statement | Server Status Check | REST API | © 2024 Altoro Mutual, Inc.
This web application is open source! Get your copy from GitHub and take advantage of advanced features

The AltoroJ 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/appscan/.

Copyright © 2008, 2017, IBM Corporation, All rights reserved. Copyright © 2017, 2024, HCL Technologies, Ltd., All rights reserved.

HTML Injections

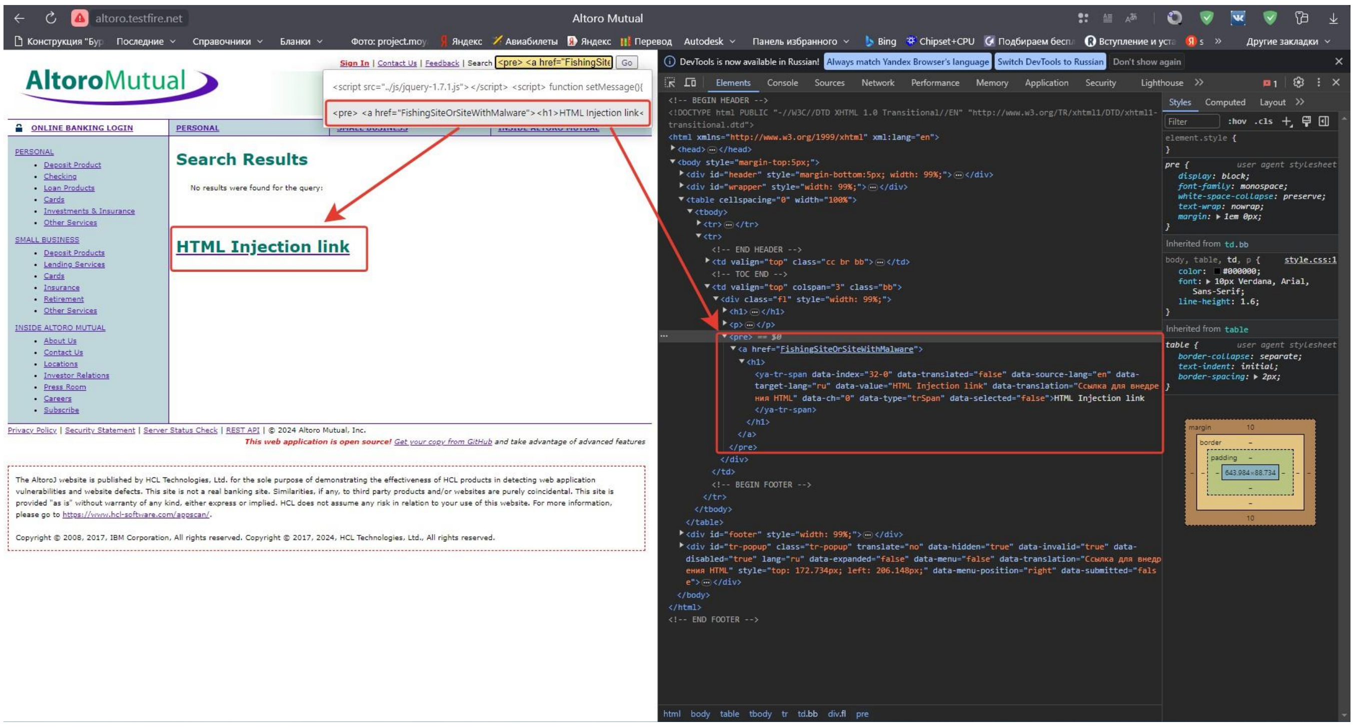
Synopsis	The web application receives input from an upstream component, but it does not neutralize or incorrectly neutralizes special characters such as `<`, `>`, and `&` that could be interpreted as web-scripting elements when they are sent to a downstream component that processes web pages.
Description	This may allow such characters to be treated as control characters, which are executed client-side in the context of the user's session. Although this can be classified as an injection problem, the more pertinent issue is the improper conversion of such special characters to respective context-appropriate entities before displaying them to the user
Prerequisites	Development and implementation stage that the development team did not foresee.
Risk Factor	*Medium*
Solution	Carefully check each input parameter against a rigorous positive specification (*allowlist*) defining the specific characters and format allowed. All input should be neutralized, not just parameters that the user is supposed to specify, but all data in the request, including hidden fields, cookies, headers, the URL itself, and so forth. A common mistake that leads to continuing XSS vulnerabilities is to validate only fields that are expected to be redisplayed by the site

Proof Of Concept (PoC):

1. Open the website https://altoro.testfire.net.
2. Enter the following code into the search bar.

```
<pre>
    <a href="FishingSiteOrSiteWithMalware"><h1>HTML Injection link</h1></a>
</pre>
```

3. Press the `[F12]` button (*to open the developer toolbar*).
4. As we can see, our HTML code is injected in the DOM.



<script src="../js/jquery-1.7.1.js"></script> <script> function setMessage(){

<pre> <a href="FishingSiteOrSiteWithMalware"><h1>HTML Injection link<

```
<!-- BEGIN HEADER -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
 ▸<head>…</head>
 ▾<body style="margin-top:5px;">
   ▸<div id="header" style="margin-bottom:5px; width: 99%;">…</div>
   ▸<div id="wrapper" style="width: 99%;">…</div>
   ▾<table cellspacing="0" width="100%">
     ▾<tbody>
       ▸<tr>…</tr>
       ▾<tr>
          <!-- END HEADER -->
         ▸<td valign="top" class="cc br bb">…</td>
          <!-- TOC END -->
         ▾<td valign="top" colspan="3" class="bb">
           ▾<div class="fl" style="width: 99%;">
             ▸<h1>…</h1>
             ▸<p>…</p>
             ▾<pre> == $0
               ▾<a href="FishingSiteOrSiteWithMalware">
                 ▾<h1>
                    <ya-tr-span data-index="32-0" data-translated="false" data-source-lang="en" data-
                    target-lang="ru" data-value="HTML Injection link" data-translation="Ссылка для внедре
                    ния HTML" data-ch="0" data-type="trSpan" data-selected="false">HTML Injection link
                    </ya-tr-span>
                  </h1>
                </a>
              </pre>
            </div>
          </td>
          <!-- BEGIN FOOTER -->
        </tr>
      </tbody>
    </table>
   ▸<div id="footer" style="width: 99%;">…</div>
   ▸<div id="tr-popup" class="tr-popup" translate="no" data-hidden="true" data-invalid="true" data-
    disabled="true" lang="ru" data-expanded="false" data-menu="false" data-translation="Ссылка для внедр
    ения HTML" style="top: 172.734px; left: 206.148px;" data-menu-position="right" data-submitted="fals
    e">…</div>
  </body>
</html>
<!-- END FOOTER -->
```

Cookie Injection Scripting

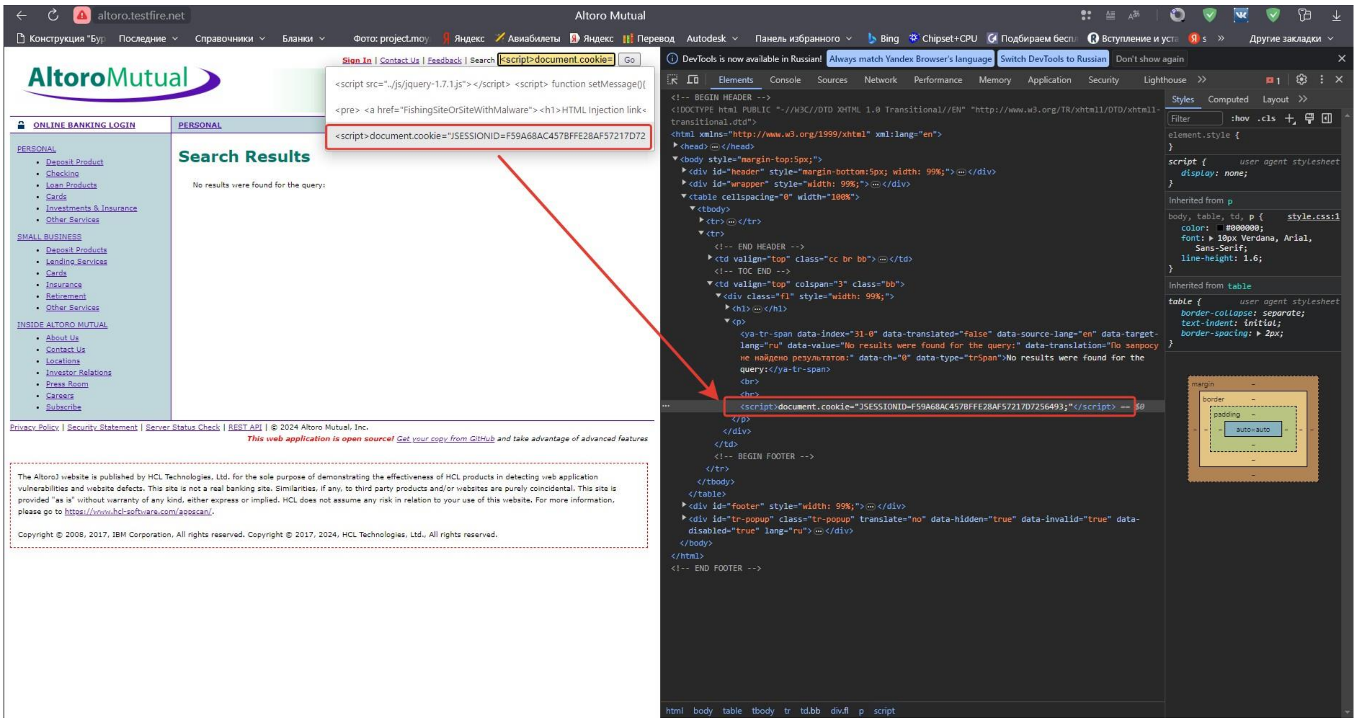
Synopsis	A web application that is potentially vulnerable to cookie attacks.
Description	The web application does not strip special characters from query strings, and an attacker could change the state of cookies to bypass authentication. By exploiting this issue, an attacker could inject arbitrary cookies or launch a "*Session Fixation*" attack.
Prerequisites	- Possibility of an XSS attack (*Cross-Site Scripting*) for the purpose of introducing malicious code - The CSRF security token is not used - The SID regeneration requirements is not performed for every request. - The Retrieve only server-generated SIDs attribut is missing. - The HttpOnly and SameSite cookie attribute is missing.
Risk Factor	*Medium*
Solution	- Restrict access to the vulnerable application. - Store state information and sensitive data on the server side only. - Ensure that the system definitively and unambiguously keeps track of its own state and user state and has rules defined for legitimate state transitions. - Do not allow any application user to affect state directly in any way other than through legitimate actions leading to state transitions.

Proof Of Concept (PoC):

1. Open the website https://altoro.testfire.net.
2. Enter the following code into the search bar.

```
<script>document.cookie="JSESSIONID=F59A68AC457BFFE28AF57217D7256493;"
</script>
```

3. Press the `[F12]` button (*to open the developer toolbar*).
4. As we can see, our HTML code is injected in the DOM.



```
<script src="../js/jquery-1.7.1.js"></script> <script> function setMessage(){
<pre> <a href="FishingSiteOrSiteWithMalware"><h1>HTML Injection link<
<script>document.cookie="JSESSIONID=F59A68AC457BFFE28AF57217D72
```

Search Results

No results were found for the query:

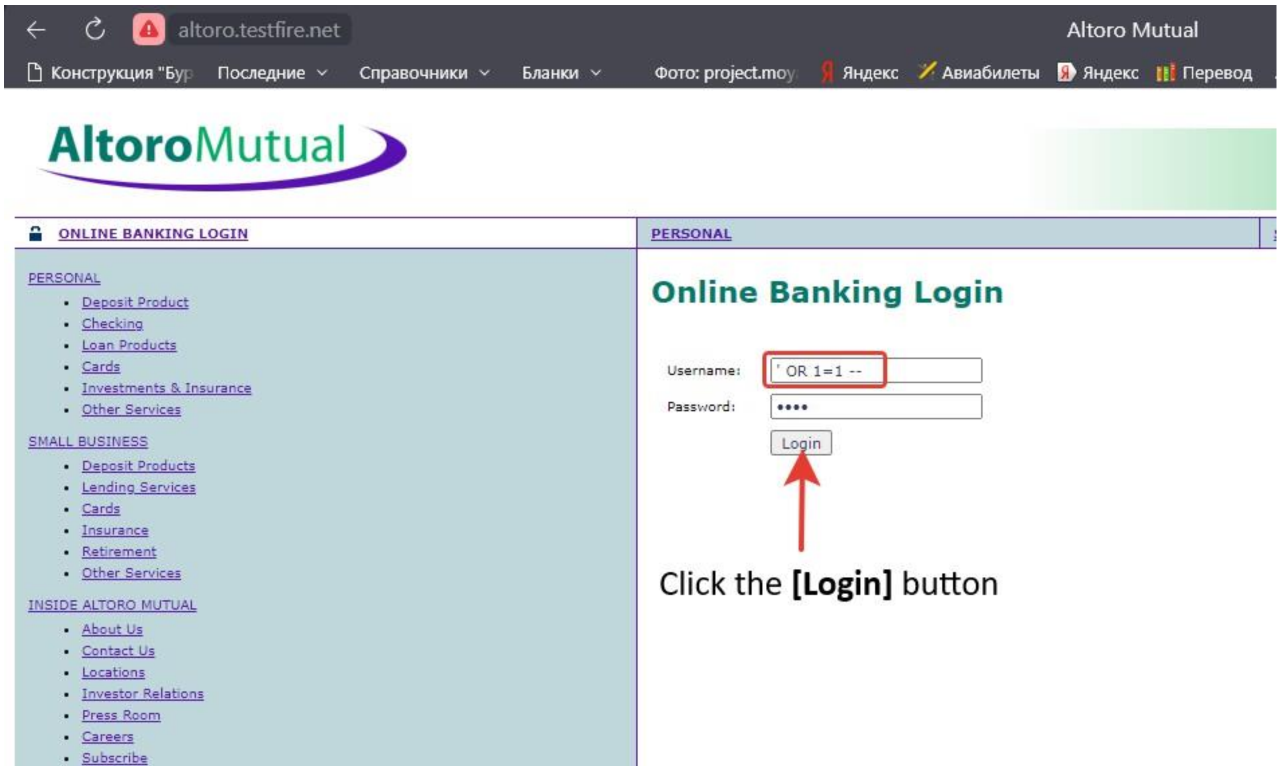
```
<!-- BEGIN HEADER -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
 <head>…</head>
 <body style="margin-top:5px;">
  <div id="header" style="margin-bottom:5px; width: 99%;">…</div>
  <div id="wrapper" style="width: 99%;">…</div>
  <table cellspacing="0" width="100%">
   <tbody>
    <tr>…</tr>
    <tr>
      <!-- END HEADER -->
     <td valign="top" class="cc br bb">…</td>
      <!-- TOC END -->
     <td valign="top" colspan="3" class="bb">
      <div class="fl" style="width: 99%;">
       <h1>…</h1>
       <p>
         <ya-tr-span data-index="31-0" data-translated="false" data-source-lang="en" data-target-
         lang="ru" data-value="No results were found for the query:" data-translation="По запросу
         не найдено результатов:" data-ch="0" data-type="trSpan">No results were found for the
         query:</ya-tr-span>
         <br>
         <br>
         <script>document.cookie="JSESSIONID=F59A68AC457BFFE28AF57217D7256493;"</script> == $0
       </p>
      </div>
     </td>
      <!-- BEGIN FOOTER -->
    </tr>
   </tbody>
  </table>
  <div id="footer" style="width: 99%;">…</div>
  <div id="tr-popup" class="tr-popup" translate="no" data-hidden="true" data-invalid="true" data-
  disabled="true" lang="ru">…</div>
 </body>
</html>
<!-- END FOOTER -->
```

Absence of Anti-CSRF Tokens

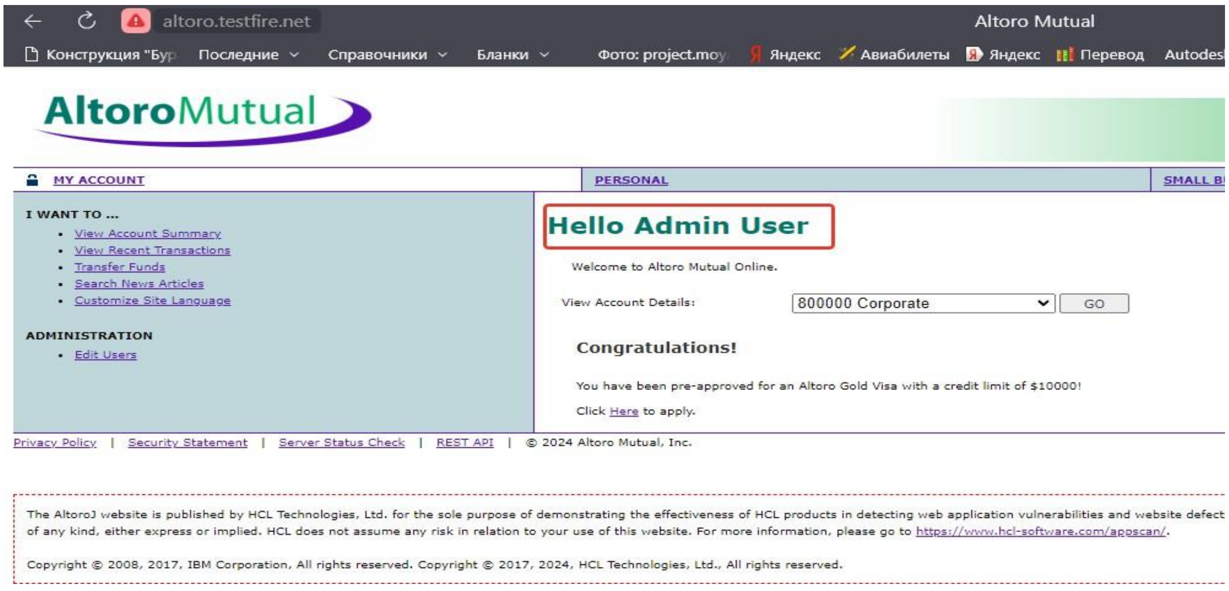
Synopsis	No Anti-CSRF tokens were found in a HTML submission form.
Description	A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.
Prerequisites	- The victim has an active session on the target site. - The victim is authenticated via HTTP auth on the target site. - The victim is on the same local network as the target site.
Risk Factor	*Medium*
Solution	- Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. - Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script. - Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable. - Check the HTTP Referrer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referrer for privacy reasons.

Proof Of Concept (PoC):

1. Open the website https://altoro.testfire.net.
2. Click the `[Sign in]` button.
3. Enter *Username:* `' OR 1=1 --` and *Password:* `q` to check for the SQLi vulnerability.



4. Click the `[Login]` button.



altoro.testfire.net — Altoro Mutual

MY ACCOUNT | PERSONAL | SMALL B

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: 800000 Corporate GO

Congratulations!

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

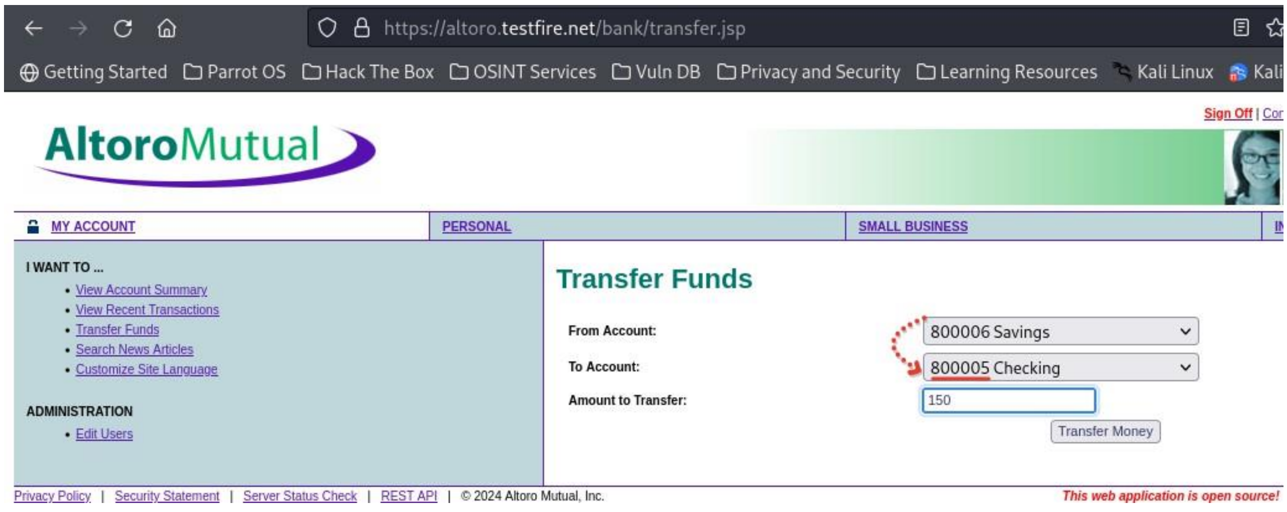
Click Here to apply.

Privacy Policy | Security Statement | Server Status Check | REST API | © 2024 Altoro Mutual, Inc.

The AltoroJ 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 defect 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/appscan/.

Copyright © 2008, 2017, IBM Corporation, All rights reserved. Copyright © 2017, 2024, HCL Technologies, Ltd., All rights reserved.

5. Go to the "*Transfer Funds*" page.
6. Prepare a transaction from user *800006* to **800005** for *150*.



https://altoro.testfire.net/bank/transfer.jsp

Transfer Funds

From Account: 800006 Savings

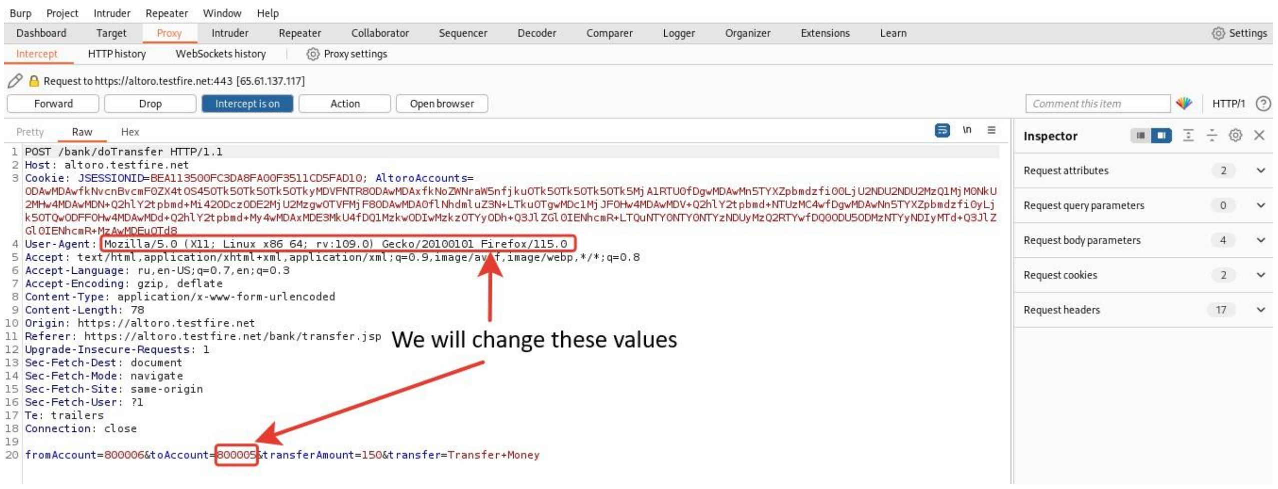
To Account: 800005 Checking

Amount to Transfer: 150

Transfer Money

Privacy Policy | Security Statement | Server Status Check | REST API | © 2024 Altoro Mutual, Inc. *This web application is open source!*

7. To get the admin session ID, launch the "*Burp Suite*".
8. Go to the "*Proxy*" tab.
9. Click the `[Intercept is Off]` button (*to activate interception*).
10. Return to the browser and click the `[Transfer money]` button.
11. At this moment, the traffic is intercepted and we, in the person of the attacker, receive the administrator's session ID.



Request to https://altoro.testfire.net:443 [65.61.137.117]

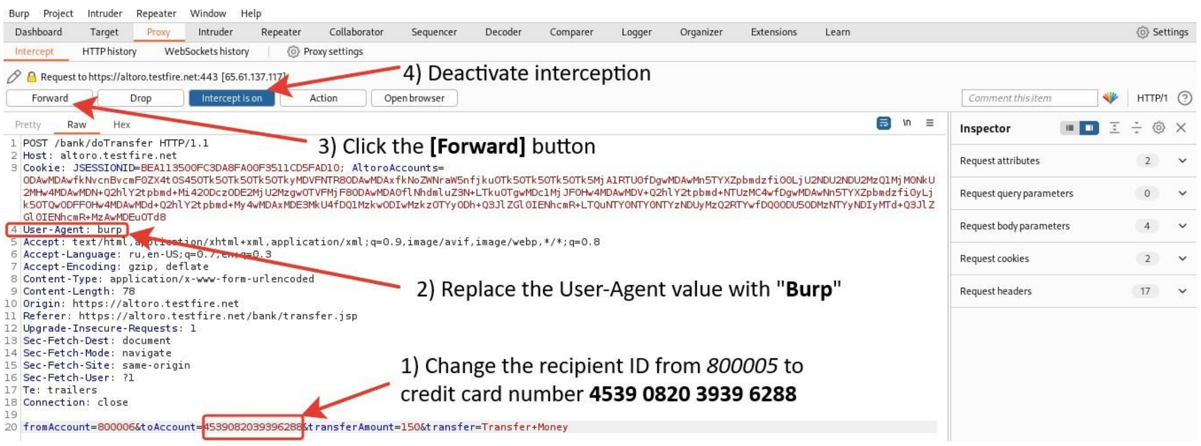
```
POST /bank/doTransfer HTTP/1.1
Host: altoro.testfire.net
Cookie: JSESSIONID=BEA113500FC3DA8FA00F3511CD5FAD10; AltoroAccounts=
ODAwMDAwfkNvcnBvcmF0ZX4tOS450Tk5OTk5OTk5OTkyMDVFNTR8ODAwMDAxfkNoZWNraW5nfjkuOTk5OTk5OTk5OTk5MjA1RTU0fDgwMDAwMn5TYXZpbmdzfi0OLjU2NDU2NDU2MzQ1MjM0NkU
2MHw4MDAwMDN+Q2hlY2tpbmd+Mi420Dcz0DE2MjU2MzgwOTVFMjF8ODAwMDA0flNhdmluZ3N+LTkuOTgwMDc1MjJFOHw4MDAwMDV+Q2hlY2tpbmd+NTUzMC4wfDgwMDAwNn5TYXZpbmdzfi0yLj
k5OTQwODFFOHw4MDAwMDd+Q2hlY2tpbmd+My4wMDAxMDE3MkU4fDQ1MzkwODIwMzkzOTYyODh+Q3JlZGl0IENhcmR+LTQuNTY0NTY0NTYzNDUyMzQ2RTYwfDQ0ODU5ODMzNTYyNDIyMTd+Q3JlZ
Gl0IENhcmR+MzAwMDEuOTd8
User-Agent: Mozilla/5.0 (X11; Linux x86 64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: ru,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 78
Origin: https://altoro.testfire.net
Referer: https://altoro.testfire.net/bank/transfer.jsp
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Connection: close

fromAccount=800006&toAccount=800005&transferAmount=150&transfer=Transfer+Money
```

12. To confirm that the request came from a location other than the user's browser, replace the User-Agent

value with `Burp`.

> **IMPORTANT:**
>
> If there was a form authentication system, attempts to **manipulate data from another location would be impossible!**

13. To change the recipient, change the recipient ID from *800005* to the credit card number *4539082039396288*.
14. Click the `[Forward]` button.
15. Click the `[Intercept is on]` button (*to disable interception*).

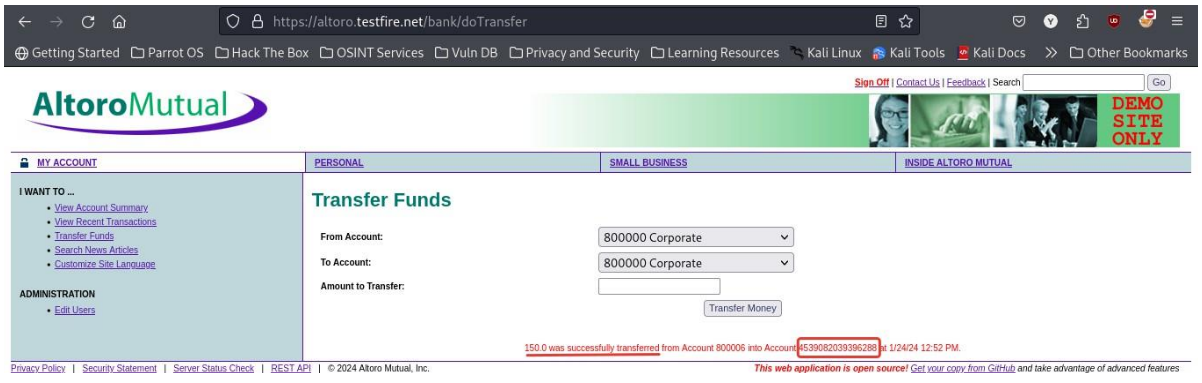


Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings
Intercept HTTP history WebSockets history Proxy settings
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 4
Request cookies 2
Request headers 17
4) Deactivate interception
3) Click the **[Forward]** button
2) Replace the User-Agent value with "**Burp**"
1) Change the recipient ID from *800005* to credit card number **4539 0820 3939 6288**

Request to https://altoro.testfire.net:443 [65.61.137.117]

```
POST /bank/doTransfer HTTP/1.1
Host: altoro.testfire.net
Cookie: JSESSIONID=BEA113500FC3DA8FA00F3511CD5FAD10; AltoroAccounts=
ODAwMDAwfkNvcnBvcmF0ZX4tOS450Tk5OTk5OTk5OTkyMDVFNTR8ODAwMDAxfkNoZWNraW5nfjkuOTk5OTk5OTk5MjA1RTU0fDgwMDAwMn5TYXZpbmdzfi0OLjU2NDU2NDU2MzQ1MjM0NkU
2MHw4MDAwMDN+Q2hlY2tpbmd+Mi420DczODE2MjU2MzgwOTVFMjF8ODAwMDA0flNhdmluZ3N+LTkuOTgwMDc1MjJFOHw4MDAwMDV+Q2hlY2tpbmd+NTUzMC4wfDgwMDAwNn5TYXZpbmdzfi0yLj
k5OTQwODFFOHw4MDAwMDd+Q2hlY2tpbmd+My4wMDAxMDE3MkU4fDQ1MzkwODIwMzkzOTYyODh+Q3JlZGl0IENhcmR+LTQuNTY0NTY0NTYzNDUyMzQ2RTYwfDQ0ODU5ODMzNTYyNDIyMTd+Q3JlZ
Gl0IENhcmR+MzAwMDEuOTd8
User-Agent: burp
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: ru,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 78
Origin: https://altoro.testfire.net
Referer: https://altoro.testfire.net/bank/transfer.jsp
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Connection: close

fromAccount=800006&toAccount=4539082039396288&transferAmount=150&transfer=Transfer+Money
```

16. Let's return to the browser.
17. A message at the bottom of the page indicates that the amount of "**150.0 was successfully transferred from Account 800006 into 4539082039396288**" instead of user account *800005*.



https://altoro.testfire.net/bank/doTransfer
Getting Started Parrot OS Hack The Box OSINT Services Vuln DB Privacy and Security Learning Resources Kali Linux Kali Tools Kali Docs Other Bookmarks
Sign Off | Contact Us | Feedback | Search Go
AltoroMutual
DEMO SITE ONLY
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
Transfer Money
Privacy Policy | Security Statement | Server Status Check | REST API | © 2024 Altoro Mutual, Inc.
This web application is open source! Get your copy from GitHub and take advantage of advanced features

Transfer Funds

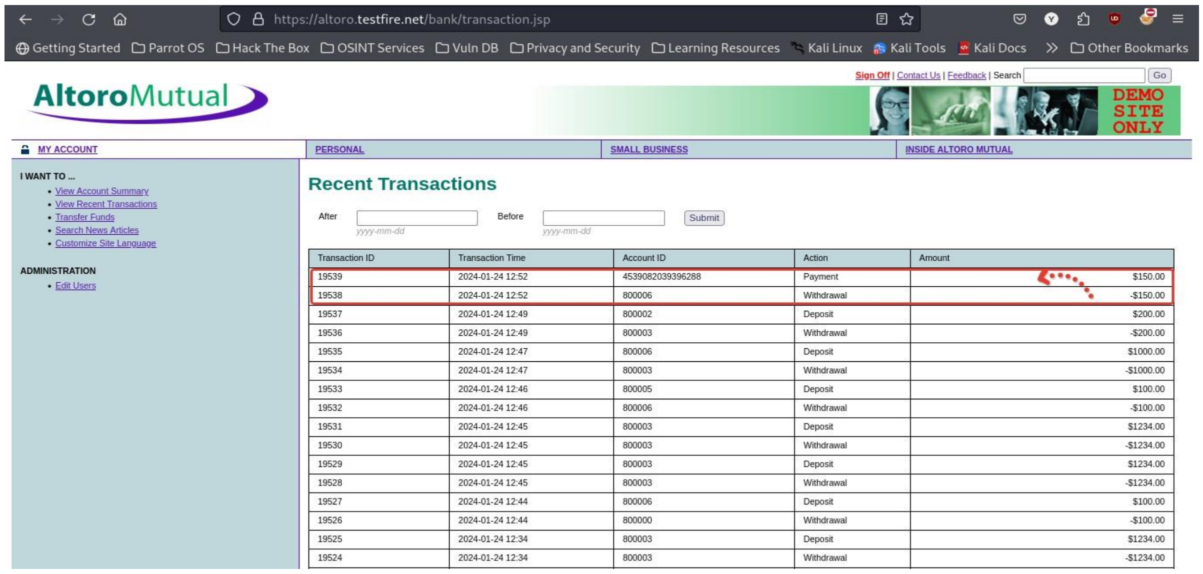
From Account: 800000 Corporate

To Account: 800000 Corporate

Amount to Transfer:

150.0 was successfully transferred from Account 800006 into Account 4539082039396288 at 1/24/24 12:52 PM.

18. Open the "*View Recent Transactions*" page and make sure once again that the system performed actions not on behalf of the user, but on behalf of the attacker.



https://altoro.testfire.net/bank/transaction.jsp
Getting Started Parrot OS Hack The Box OSINT Services Vuln DB Privacy and Security Learning Resources Kali Linux Kali Tools Kali Docs Other Bookmarks
Sign Off | Contact Us | Feedback | Search Go
AltoroMutual
DEMO SITE ONLY
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
After yyyy-mm-dd Before yyyy-mm-dd Submit

Recent Transactions

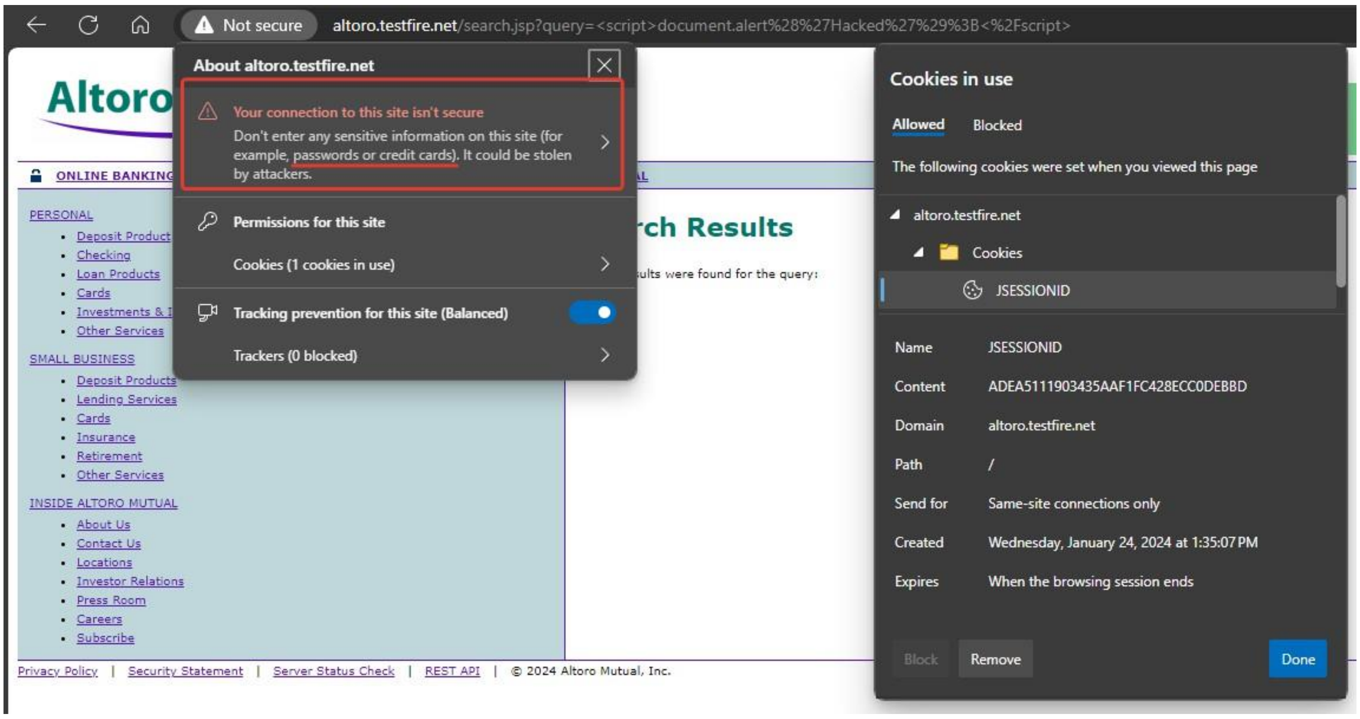
Transaction ID	Transaction Time	Account ID	Action	Amount
19539	2024-01-24 12:52	4539082039396288	Payment	$150.00
19538	2024-01-24 12:52	800006	Withdrawal	-$150.00
19537	2024-01-24 12:49	800002	Deposit	$200.00
19536	2024-01-24 12:49	800003	Withdrawal	-$200.00
19535	2024-01-24 12:47	800006	Deposit	$1000.00
19534	2024-01-24 12:47	800003	Withdrawal	-$1000.00
19533	2024-01-24 12:46	800005	Deposit	$100.00
19532	2024-01-24 12:46	800006	Withdrawal	-$100.00
19531	2024-01-24 12:45	800003	Deposit	$1234.00
19530	2024-01-24 12:45	800003	Withdrawal	-$1234.00
19529	2024-01-24 12:45	800003	Deposit	$1234.00
19528	2024-01-24 12:45	800003	Withdrawal	-$1234.00
19527	2024-01-24 12:44	800006	Deposit	$100.00
19526	2024-01-24 12:44	800000	Withdrawal	-$100.00
19525	2024-01-24 12:34	800003	Deposit	$1234.00
19524	2024-01-24 12:34	800003	Withdrawal	-$1234.00

Web Server Transmits Cleartext Credentials

Synopsis	The web application transmits sensitive or security-critical data in cleartext in a communication channel that can be sniffed by unauthorized actors.
Description	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.
Prerequisites	- This weakness is caused by missing a security tactic during the architecture and design phase. - For hardware, this may be introduced when design does not plan for an attacker having physical access while a legitimate user is remotely operating the device
Risk Factor	*Low*
Solution	Make sure that every sensitive form transmits content over HTTPS.

Proof Of Concept (PoC):

1. Open the website https://altoro.testfire.net.
2. On the left side of the address bar, we see an exclamation mark (!), when you click on it, a message appears stating that only the HTTP protocol is used and all data is transmitted without encoding.



Not secure altoro.testfire.net/search.jsp?query=<script>document.alert%28%27Hacked%27%29%3B<%2Fscript>

About altoro.testfire.net

Your connection to this site isn't secure
Don't enter any sensitive information on this site (for example, passwords or credit cards). It could be stolen by attackers.

Permissions for this site
Cookies (1 cookies in use)
Tracking prevention for this site (Balanced)
Trackers (0 blocked)

Cookies in use
Allowed Blocked
The following cookies were set when you viewed this page
altoro.testfire.net
Cookies
JSESSIONID

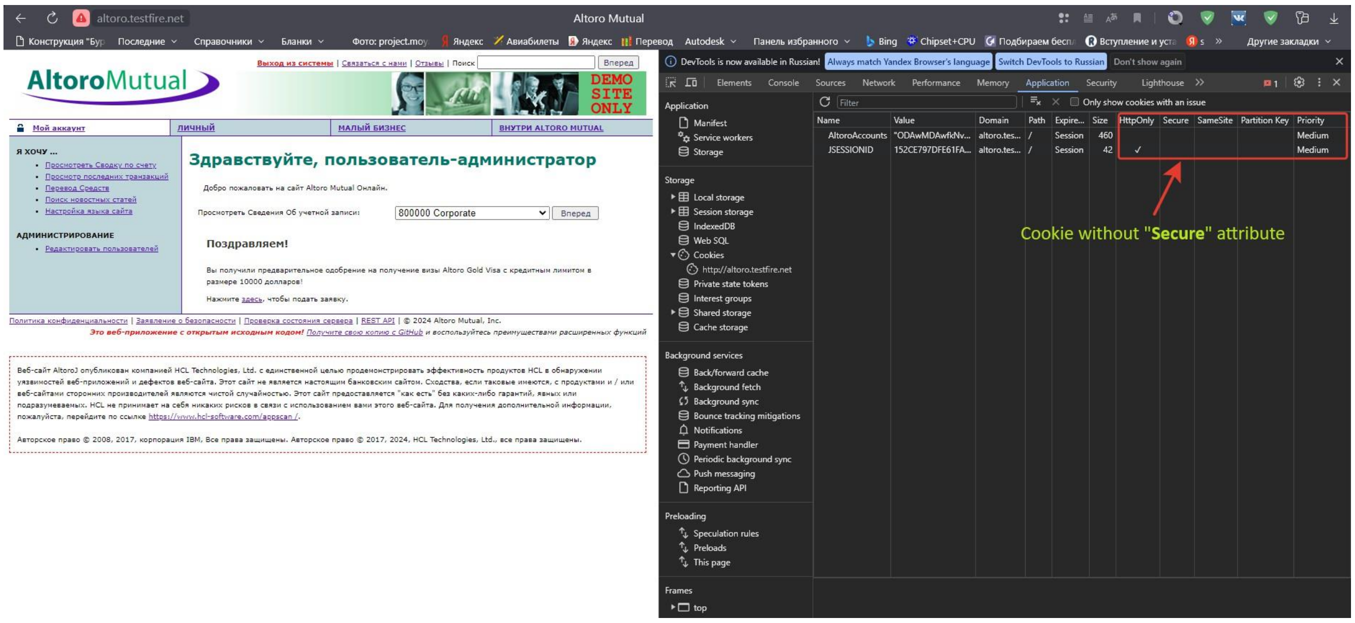
Name	JSESSIONID
Content	ADEA5111903435AAF1FC428ECC0DEBBD
Domain	altoro.testfire.net
Path	/
Send for	Same-site connections only
Created	Wednesday, January 24, 2024 at 1:35:07 PM
Expires	When the browsing session ends

HTTP Cookie without '*Secure*' attribute

Synopsis	HTTP session cookies might be transmitted in cleartext.
Description	The web application runs over unencrypted HTTP, or the cookies are not marked as "*secure*", which means that sensitive cookies are sent over an unencrypted channel. As a result, a remote attacker could intercept these cookies.
Prerequisites	The web application does not have a security certificate or is not configured correctly.
Risk Factor	*Low*
Solution	- Always set the secure attribute when the cookie should send via HTTPS only. - Check your server configuration to always use HTTPS.

Proof Of Concept (PoC):

1. Open the website https://altoro.testfire.net.
2. Press the `[F12]` button (*to open the developer toolbar*).
3. Go to the *Application/Storage* tab.
4. Expand the *cookies* menu.
5. We see that there are no security flags.



Я ХОЧУ ...
- Просмотреть Сводку по счету
- Просмотр последних транзакций
- Перевод Средств
- Поиск новостных статей
- Настройка языка сайта

АДМИНИСТРИРОВАНИЕ
- Редактировать пользователей

Здравствуйте, пользователь-администратор

Добро пожаловать на сайт Altoro Mutual Онлайн.

Просмотреть Сведения Об учетной записи: 800000 Corporate Вперед

Поздравляем!

Вы получили предварительное одобрение на получение визы Altoro Gold Visa с кредитным лимитом в размере 10000 долларов!

Нажмите здесь, чтобы подать заявку.

Политика конфиденциальности | Заявление о безопасности | Проверка состояния сервера | REST API | © 2024 Altoro Mutual, Inc.

Это веб-приложение с открытым исходным кодом! Получите свою копию с GitHub и воспользуйтесь преимуществами расширенных функций

Веб-сайт AltoroJ опубликован компанией HCL Technologies, Ltd. с единственной целью продемонстрировать эффективность продуктов HCL в обнаружении уязвимостей веб-приложений и дефектов веб-сайта. Этот сайт не является настоящим банковским сайтом. Сходства, если таковые имеются, с продуктами и / или веб-сайтами сторонних производителей являются чистой случайностью. Этот сайт предоставляется "как есть" без каких-либо гарантий, явных или подразумеваемых. HCL не принимает на себя никаких рисков в связи с использованием вами этого веб-сайта. Для получения дополнительной информации, пожалуйста, перейдите по ссылке https://www.hcl-software.com/appscan/.

Авторское право © 2008, 2017, корпорация IBM, Все права защищены. Авторское право © 2017, 2024, HCL Technologies, Ltd., все права защищены.

DevTools is now available in Russian! Always match Yandex Browser's language | Switch DevTools to Russian | Don't show again

Filter | Only show cookies with an issue

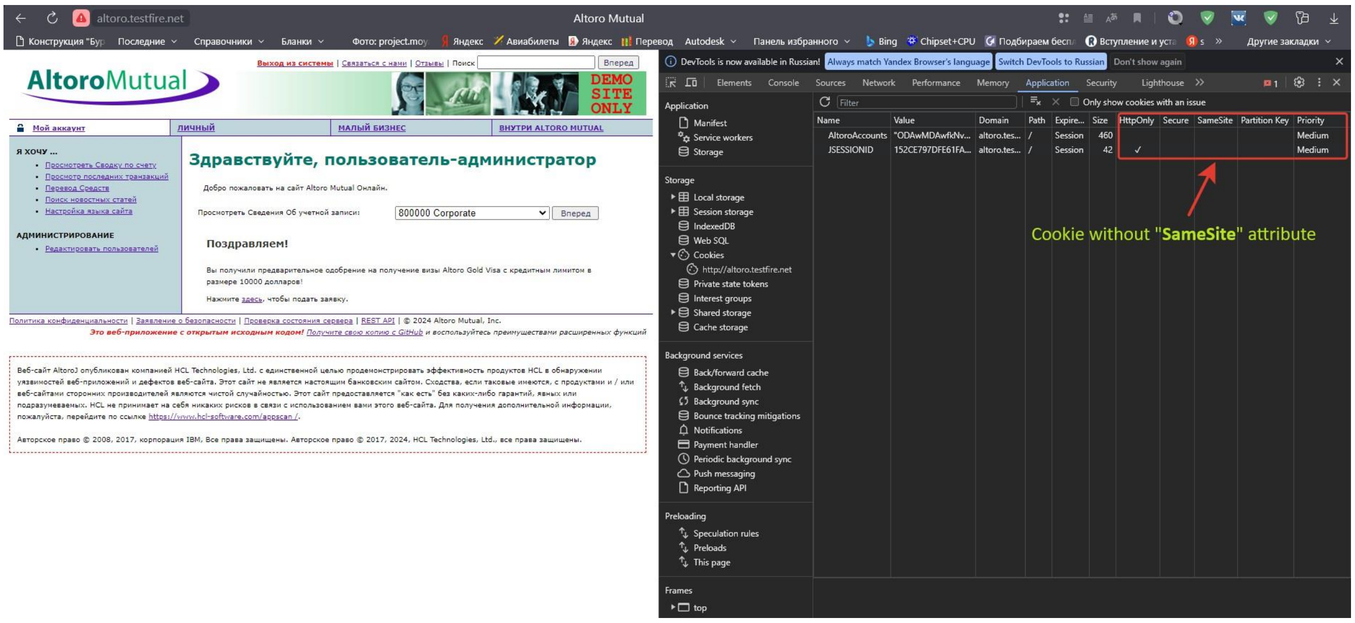
Name	Value	Domain	Path	Expire...	Size	HttpOnly	Secure	SameSite	Partition Key	Priority
AltoroAccounts	"ODAwMDAwfkNv...	altoro.tes...	/	Session	460					Medium
JSESSIONID	152CE797DFE61FA...	altoro.tes...	/	Session	42	✓				Medium

HTTP Cookie without '*SameSite*' attribute

Synopsis	The SameSite attribute for sensitive cookies is not set, or an insecure value is used.
Description	The SameSite attribute controls how cookies are sent for cross-domain requests. This attribute may have three values: '*Lax*', '*Strict*', or '*None*'. If the '*None*' value is used, a website may create a cross-domain POST HTTP request to another website, and the browser automatically adds cookies to this request. This may lead to Cross-Site-Request- Forgery (*CSRF*) attacks if there are no additional protections in place (*such as Anti-CSRF tokens*).
Prerequisites	The web application does not have a security certificate or is not configured correctly.
Risk Factor	*Low*
Solution	- Always set the secure attribute when the cookie should send via HTTPS only. - Check your server configuration to always use HTTPS.

Proof Of Concept (PoC):

1. Open the website https://altoro.testfire.net.
2. Press the `[F12]` button (*to open the developer toolbar*).
3. Go to the *Application/Storage* tab.
4. Expand the *cookies* menu.
5. We see that there are no security flags.



Здравствуйте, пользователь-администратор

Добро пожаловать на сайт Altoro Mutual Онлайн.

Просмотреть Сведения Об учетной записи: 800000 Corporate

Поздравляем!

Вы получили предварительное одобрение на получение визы Altoro Gold Visa с кредитным лимитом в размере 10000 долларов!

Нажмите здесь, чтобы подать заявку.

Политика конфиденциальности | Заявление о безопасности | Проверка состояния сервера | REST API | © 2024 Altoro Mutual, Inc.

Это веб-приложение с открытым исходным кодом! Получите свою копию с GitHub и воспользуйтесь преимуществами расширенных функций

Веб-сайт AltoroJ опубликован компанией HCL Technologies, Ltd. с единственной целью продемонстрировать эффективность продуктов HCL в обнаружении уязвимостей веб-приложений и дефектов веб-сайта. Этот сайт не является настоящим банковским сайтом. Сходства, если таковые имеются, с продуктами и / или веб-сайтами сторонних производителей являются чистой случайностью. Этот сайт предоставляется "как есть" без каких-либо гарантий, явных или подразумеваемых. HCL не принимает на себя никаких рисков в связи с использованием вами этого веб-сайта. Для получения дополнительной информации, пожалуйста, перейдите по ссылке https://www.hcl-software.com/appscan/.

Авторское право © 2008, 2017, корпорация IBM, Все права защищены. Авторское право © 2017, 2024, HCL Technologies, Ltd., все права защищены.

Name	Value	Domain	Path	Expire...	Size	HttpOnly	Secure	SameSite	Partition Key	Priority
AltoroAccounts	"ODAwMDAwfkNv...	altoro.tes...	/	Session	460					Medium
JSESSIONID	152CE797DFE61FA...	altoro.tes...	/	Session	42	✓				Medium

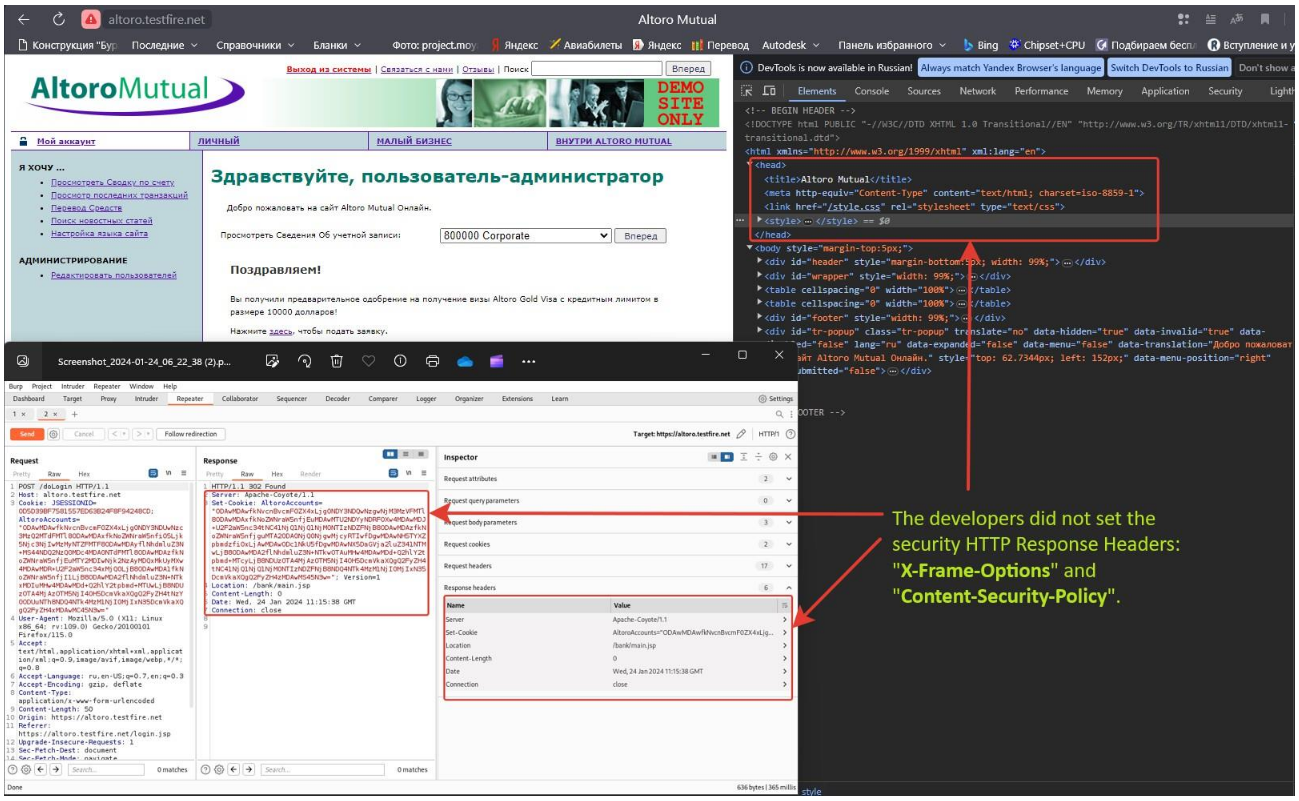
Cookie without "**SameSite**" attribute

Missing or Permissive "*X-Frame-Options*" and "*CSP frame-ancestors*" HTTP Response Headers

Synopsis	**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.**
Description	A web application is expected to place restrictions on whether it is allowed to be rendered within frames, iframes, objects, embed or applet elements. Without the restrictions, users can be tricked into interacting with the application when they were not intending to.
Prerequisites	The developers did not set the security HTTP Response Headers: "**X-Frame-Options**" and "**Content-Security-Policy**".
Risk Factor	*Info*
Solution	- Set a properly configured *X-Frame-Options* and *Content-Security-Policy frame-ancestor's* headers for all requested resources - The use of *X-Frame-Options* and *Content-Security-Policy frame-ancestors* allows developers of web content to restrict the usage of their application within the form of overlays, frames, or iframes. The developer can indicate from which domains can frame the content. - A developer can use a "*frame-breaker*" script in each page that should not be framed.

Proof Of Concept (PoC):

1. Open the website https://altoro.testfire.net.
2. Press the `[F12]` button (*to open the developer toolbar*).
3. This header is missing inside the `<head>` tag.
4. We will get the same result if we use any network traffic analysis tools (*Burp Suite, Wireshark, etc.*).



```
<!-- BEGIN HEADER -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Altoro Mutual</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <link href="/style.css" rel="stylesheet" type="text/css">
    <style>…</style> == $0
  </head>
  <body style="margin-top:5px;">
    <div id="header" style="margin-bottom:5px; width: 99%;">…</div>
    <div id="wrapper" style="width: 99%;">…</div>
    <table cellspacing="0" width="100%">…</table>
    <table cellspacing="0" width="100%">…</table>
    <div id="footer" style="width: 99%;">…</div>
```

Request

```
POST /doLogin HTTP/1.1
Host: altoro.testfire.net
Cookie: JSESSIONID=0D5D39BF7581557ED63B24F8F94248CD; AltoroAccounts="ODAwMDAwfkNvcnBvcmF0ZX4xLjgONDY3NDUwNzc3MzQ2MTdFMTl8ODAwMDAxfkNoZWNraW5nfi0SLjk5Njc3NjIwMzMyNTZFMTF8ODAwMDAyflNhdmluZ3N+MS44NDQ2NzQ0MDc4MDA0NTdFMTl8ODAwMDAzfkNoZWNraW5nfjEuMTY2MDIwNjk2NzAyMDQxMkUyMXw4MDAwMDR+U2F2aW5nc34xMjQ0LjB8ODAwMDA1fkNoZWNraW5nfjI1LjB8ODAwMDA2flNhdmluZ3N+NTkxMDIuMHw4MDAwMDd+Q2hlY2tpbmd+MTUwLjB8NDUzOTA4MjAzOTM5NjI4OH5DcmVkaXQgQ2FyZH4tNzY0ODUuNTh8NDQ4NTk4MzM1NjI0MjIxN35DcmVkaXQgQ2FyZH4xMDAwMC45Nzw="
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: ru,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 50
Origin: https://altoro.testfire.net
Referer: https://altoro.testfire.net/login.jsp
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
```

Response

```
HTTP/1.1 302 Found
Server: Apache-Coyote/1.1
Set-Cookie: AltoroAccounts="ODAwMDAwfkNvcnBvcmF0ZX4xLjgONDY3NDQwNzgwNjM3MzVFMTl8ODAwMDAxfkNoZWNraW5nfjEuMDAwMTU2NDYyNDRFOXw4MDAwMDJ+U2F2aW5nc34tNC41NjQ1NjQ1NjM0NTIzNDZFNjB8ODAwMDAzfkNoZWNraW5nfjguMTA2ODA0NjQ0NjgwMjcyRTIwfDgwMDAwNH5TYXZpbmdzfi0xLjAwMDAwODc1NkU5fDgwMDAwNX5DaGVja2luZ341NTMwLjB8ODAwMDA2flNhdmluZ3N+NTkwOTAuMHw4MDAwMDd+Q2hlY2tpbmd+MTcyLjB8NDUzOTA4MjAzOTM5NjI4OH5DcmVkaXQgQ2FyZH4tNC41NjQ1NjQ1NjM0NTIzNDZFNjB8NDQ4NTk4MzM1NjI0MjIxN35DcmVkaXQgQ2FyZH4zMDAwMS45N3w="; Version=1
Location: /bank/main.jsp
Content-Length: 0
Date: Wed, 24 Jan 2024 11:15:38 GMT
Connection: close
```

Response headers 6

Name	Value
Server	Apache-Coyote/1.1
Set-Cookie	AltoroAccounts="ODAwMDAwfkNvcnBvcmF0ZX4xLjg...
Location	/bank/main.jsp
Content-Length	0
Date	Wed, 24 Jan 2024 11:15:38 GMT
Connection	close

The developers did not set the security HTTP Response Headers: "**X-Frame-Options**" and "**Content-Security-Policy**".

5. XSS (*Cross-Site Scripting*), *clickjacking*, and *cross-site leak* vulnerabilities are possible if these headers are missing.

Hyper Text Transfer Protocol (*HTTP*) Information

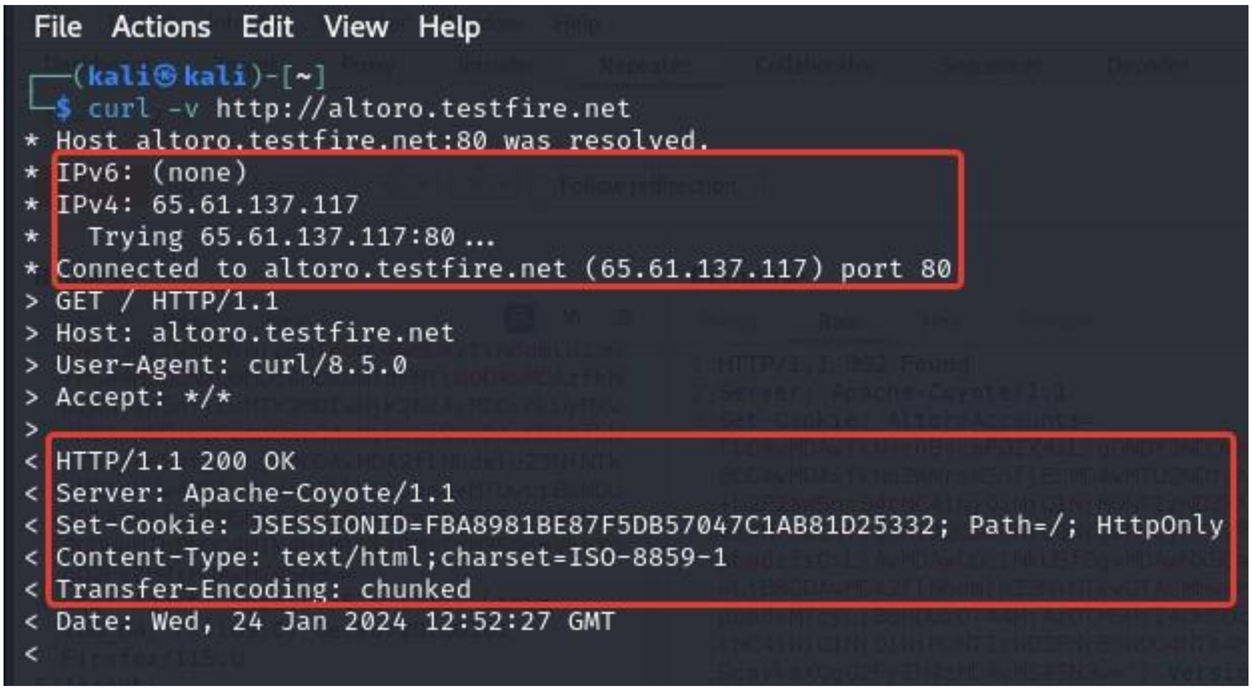
Synopsis	Some information about the remote HTTP configuration can be extracted.
Description	Preventing version disclosure is often considered best practice as it slows down hackers. Conventional wisdom holds that you should prevent any information from getting out at all, or on the other end of the scale, that the disclosing the versions of software you are using doesn't make any difference.
Prerequisites	*None*
Risk Factor	*Info*
Solution	*None*

Proof Of Concept (PoC):

1. Open a terminal (`[Ctrl]`+`[Alt]`+`[T]`).
2. Enter the following command.

```
curl -v http://altoro.testfire.net
```

3. As you can see, here is information about the protocol, port, server version, etc.



```
┌──(kali㉿kali)-[~]
└─$ curl -v http://altoro.testfire.net
* Host altoro.testfire.net:80 was resolved.
* IPv6: (none)
* IPv4: 65.61.137.117
*   Trying 65.61.137.117:80 ...
* Connected to altoro.testfire.net (65.61.137.117) port 80
> GET / HTTP/1.1
> Host: altoro.testfire.net
> User-Agent: curl/8.5.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Set-Cookie: JSESSIONID=FBA8981BE87F5DB57047C1AB81D25332; Path=/; HttpOnly
< Content-Type: text/html;charset=ISO-8859-1
< Transfer-Encoding: chunked
< Date: Wed, 24 Jan 2024 12:52:27 GMT
<
```