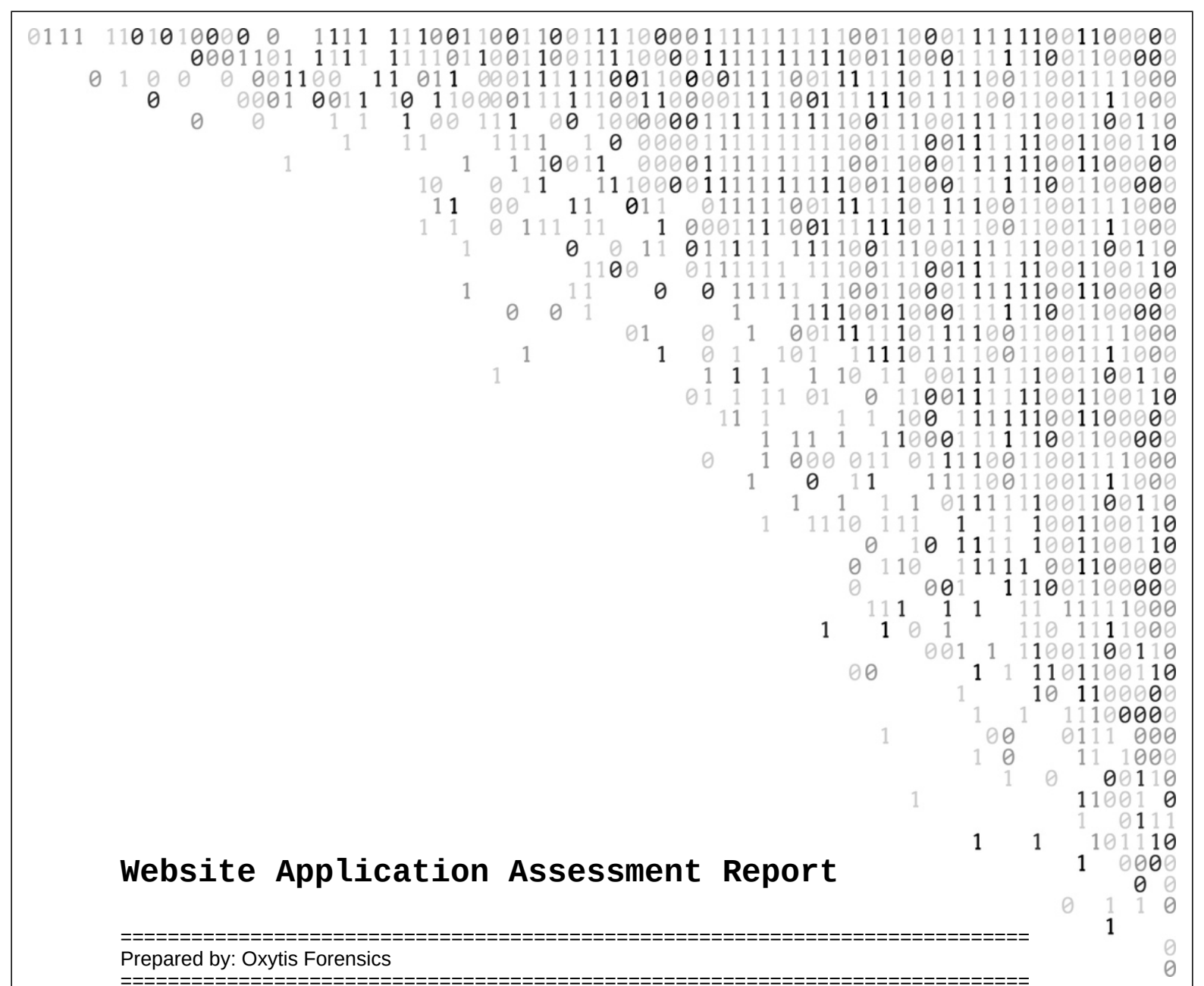

Website Application Assessment Report

==========
Prepared by: Oxytis Forensics
==========

Altoro Mutual

==========
Date: 04 July 2023
==========

Content

1.0 Executive Summary

The following report summarizes the results of a Website Application Security Assessment for Altoro Mutual. Oxytis Forensics performs real time security assessments on networks and applications. These assessments attempt to uncover security issues in the target network and applications, highlighting the impact and risks associated with any discovered issues.

The objective of this engagement was to perform a penetration test to assess the overall level of security of the Altoro website application. Oxytis Forensics defines vulnerabilities as the potential to either gain unauthorized access to a target system or extract sensitive data from it. Events including but not limited to: login bypass, ability to run commands on a target system, extraction of data from a database, a successful session hijack, credential theft, escalation of privileges are threat vectors that lead to compromises. This report includes potential vulnerabilities that may require an additional attack vector beyond the scope of this engagement to leverage a compromise.

Oxytis Forensics performed a website application assessment for Altoro using the OWASP Top 10 as the foundation for its assessment. The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications. Using the OWASP Top 10 is perhaps the most effective first step towards changing the software development culture within your organization into one that produces more secure code.

As part of our pre-engagement interactions, Oxytis Forensics collaborated with Altoro to arrive at the testing parameters. Additionally, this report reflects the overall Oxytis Forensics effort and success or failure at discovering among several classes of technical vulnerabilities, such as those enumerated in the OWASP top ten and SANS CWE 25.

A01:2021-Broken Access Control moves up from the fifth position; The 34 Common Weakness Enumerations (CWEs) mapped to Broken Access Control had more occurrences in applications than any other category.

A02:2021-Cryptographic Failures shifts up one position to #2, previously known as Sensitive Data Exposure, which was broad symptom rather than a root cause. The renewed focus here is on failures related to cryptography which often leads to sensitive data exposure or system compromise.

A03:2021-Injection slides down to the third position. 94% of the applications were tested for some form of injection, and the 33 CWEs mapped into this category have the second most occurrences in applications. Cross-site Scripting is now part of this category in this edition.

A04:2021-Insecure Design is a new category for 2021, with a focus on risks related to design flaws. If we genuinely want to "move left" as an industry, it calls for more use of threat modeling, secure design patterns and principles, and reference architectures.

A05:2021-Security Misconfiguration moves up from #6 in the previous edition; 90% of applications were tested for some form of misconfiguration. With more shifts into highly configurable software, it's not surprising to see this category move up. The former category for XML External Entities (XXE) is now part of this category.

A06:2021-Vulnerable and Outdated Components was previously titled Using Components with Known Vulnerabilities and is #2 in the Top 10 community survey, but also had enough data to make the Top 10 via data analysis. This category moves up from #9 in 2017 and is a known issue that we struggle to test and assess risk. It is the only category not to have any Common Vulnerability and Exposures (CVEs) mapped to the included CWEs, so a default exploit and impact weights of 5.0 are factored into their scores.

A07:2021-Identification and Authentication Failures was previously Broken Authentication and is sliding down from the second position, and now includes CWEs that are more related to identification failures. This category is still an integral part of the Top 10, but the increased availability of standardized frameworks seems to be helping.

A08:2021-Software and Data Integrity Failures is a new category for 2021, focusing on making assumptions related to software updates, critical data, and CI/CD pipelines without verifying integrity. One of the highest weighted impacts from Common Vulnerability and Exposures/Common Vulnerability Scoring System (CVE/CVSS) data mapped to the 10 CWEs in this category. Insecure Deserialization from 2017 is now a part of this larger category.

A09:2021-Security Logging and Monitoring Failures was previously Insufficient Logging & Monitoring and is added from the industry survey (#3), moving up from #10 previously. This category is expanded to include more types of failures, is challenging to test for, and isn't well represented in the CVE/CVSS data. However, failures in this category can directly impact visibility, incident alerting, and forensics.

A10:2021-Server-Side Request Forgery is added from the Top 10 community survey (#1). The data shows a relatively low incidence rate with above average testing coverage, along with above-average ratings for Exploit and Impact potential. This category represents the scenario where the security community members are telling us this is important,

even though it's not illustrated in the data at this time.

1.1 Findings

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Curabitur pretium tincidunt lacus. Nulla gravida orci a odio. Nullam varius, turpis et commodo pharetra, est eros bibendum elit, nec luctus magna felis sollicitudin mauris. Integer in mauris eu nibh euismod gravida. Duis ac tellus et risus vulputate vehicula. Donec lobortis risus a elit. Etiam tempor. Ut ullamcorper, ligula eu tempor congue, eros est euismod turpis, id tincidunt sapien risus a quam. Maecenas fermentum consequat mi. Donec fermentum. Pellentesque malesuada nulla a mi. Duis sapien sem, aliquet nec, commodo eget, consequat quis, neque. Aliquam faucibus, elit ut dictum aliquet, felis nisl adipiscing sapien, sed malesuada diam lacus eget erat. Cras mollis scelerisque nunc. Nullam arcu. Aliquam consequat. Curabitur augue lorem, dapibus quis, laoreet et, pretium ac, nisi. Aenean magna nisl, mollis quis, molestie eu, feugiat in, orci. In hac habitasse platea dictumst.

URLs in Findings: https://altoromutual.com

Finding	Count	Severity
File path manipulation	1	High
Cross-site scripting (reflected)	4	High
SQL injection	1	High
XPath injection	3	High
Client-side desync	1	High
Strict Transport Security Misconfiguration	27	Medium
Cross-site request forgery	3	Medium
Password field with autocomplete enabled	3	Low
Strict transport security not enforced	6	Low
Browser cross-site scripting filter misconfiguration	27	Low
Software Version Numbers Revealed	3	Low
Content Sniffing not disabled	27	Low
Arbitrary host header accepted	1	Low
Link manipulation (DOM-based)	2	Low
Open redirection (DOM-based)	4	Low
Input returned in response (reflected)	4	Info
TLS cookie without secure flag set	1	Info
Cross-domain Referer leakage	4	Info
Cross-domain script include	1	Info
Cookie without HttpOnly flag set	1	Info
Backup file	2	Info
Cacheable HTTPS response	9	Info
TLS certificate	1	Info
Mixed content	2	Info
Frameable response (potential Clickjacking)	8	Info
DOM data manipulation (DOM-based)	2	Info
Base64-encoded data in parameter	10	Info
Path-relative style sheet import	1	Info

1.2 Recommendations

-- Lorem ipsum dolor sit amet, consectetur adipiscing elit
-- Curabitur pretium tincidunt lacus
-- Aenean magna nisl, mollis quis, molestie eu
-- Lorem ipsum dolor sit amet, consectetur adipiscing elit

2.0 Penetration Testing

Oxytis Forensics lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Curabitur pretium tincidunt lacus. Nulla gravida orci a odio. Nullam varius, turpis et commodo pharetra, est eros bibendum elit, nec luctus magna felis sollicitudin mauris. Integer in mauris eu nibh euismod gravida. Duis ac tellus et risus vulputate vehicula. Donec lobortis risus a elit. Etiam tempor. Ut ullamcorper, ligula eu tempor congue, eros est euismod turpis, id tincidunt sapien risus a quam. Maecenas fermentum consequat mi. Donec fermentum. Pellentesque malesuada nulla a mi. Duis sapien sem, aliquet nec, commodo eget, consequat quis, neque. Aliquam faucibus, elit ut dictum aliquet, felis nisl adipiscing sapien, sed malesuada diam lacus eget erat. Cras mollis scelerisque nunc. Nullam arcu. Aliquam consequat. Curabitur augue lorem, dapibus quis, laoreet et, pretium ac, nisi. Aenean magna nisl, mollis quis, molestie eu, feugiat in, orci. In hac habitasse platea dictumst.

2.1 Compromise Analysis

In hac habitasse platea dictumst:

Linux 2.6.x
Windows 7 or 8
Windows NT kernel
Windows NT kernel 5.x

In hac habitasse platea dictumst:

In hac habitasse platea dictumst:

In hac habitasse platea dictumst:

192.168.1.1

In hac habitasse platea dictumst:

In hac habitasse platea dictumst:

In hac habitasse platea dictumst:

In hac habitasse platea dictumst:

In hac habitasse platea dictumst:

In hac habitasse platea dictumst:

1 France
1 Ireland
3 Italy
1 Japan
15 Netherlands
1 United Kingdom
148 United States

In hac habitasse platea dictumst:

7 Akamai Technologies, Inc.
13 Amazon.com, Inc.
63 Amazon Technologies Inc.
7 Facebook, Inc.

2 Google LLC
1 Level 3 Parent, LLC
3 Microsoft Corporation
3 Red Hat, Inc.
1 StackPath, LLC.
1 Sucuri

Credentials Harvested:

192.168.1.50 TCP 80 USER: acme_user PASS: acme_password

3.0 Issue Identification Listing

High Findings		
Vulnerability	**Count**	**Severity**
File path manipulation	1	High

Description

File path manipulation vulnerabilities arise when user-controllable data is placed into a file or URL path that is used on the server to access local resources, which may be within or outside the web root. If vulnerable, an attacker can modify the file path to access different resources, which may contain sensitive information. Even where an attack is constrained within the web root, it is often possible to retrieve items that are normally protected from direct access, such as application configuration files, the source code for server-executable scripts, or files with extensions that the web server is not configured to serve directly.
The content parameter appears to be vulnerable to file path manipulation attacks.

The payload ../WEB-INF/web.xml was submitted in the content parameter. The file WEB-INF/web.xml was returned.

Output

```
GET /index.jsp?content=..%2fWEB-INF%2fweb.xml HTTP/1.1
Host: altoromutual.com
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 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://altoromutual.com/index.jsp?content=inside.htm
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Date: Thu, 03 Nov 2022 10:49:47 GMT
Connection: close
Content-Length: 14467
```

Remediation

Ideally, application functionality should be designed in such a way that user-controllable data does not need to be placed into file or URL paths in order to access local resources on the server. This can normally be achieved by referencing known files via an index number rather than their name.

If it is considered unavoidable to place user data into file or URL paths, the data should be strictly validated against a whitelist of accepted values. Note that when accessing resources within the web root, simply blocking input containing file path traversal sequences (such as dot-dot-slash) is not always sufficient to prevent retrieval of sensitive information, because some protected items may be accessible at the original path without using any traversal sequences.

Affected paths

/index.jsp

References

https://portswigger.net/web-security/file-path-traversal
https://cwe.mitre.org/data/definitions/22.html
https://cwe.mitre.org/data/definitions/23.html
https://cwe.mitre.org/data/definitions/35.html
https://cwe.mitre.org/data/definitions/36.html
https://capec.mitre.org/data/definitions/126.html

Vulnerability	Count	Severity
Cross-site scripting (reflected)	4	High

Description

Reflected cross-site scripting vulnerabilities arise when data is copied from a request and echoed into the application's immediate response in an unsafe way. An attacker can use the vulnerability to construct a request that, if issued by another application user, will cause JavaScript code supplied by the attacker to execute within the user's browser in the context of that user's session with the application.

The attacker-supplied code can perform a wide variety of actions, such as stealing the victim's session token or login credentials, performing arbitrary actions on the victim's behalf, and logging their keystrokes.

Users can be induced to issue the attacker's crafted request in various ways. For example, the attacker can send a victim a link containing a malicious URL in an email or instant message. They can submit the link to popular web sites that allow content authoring, for example in blog comments. And they can create an innocuous looking web site that causes anyone viewing it to make arbitrary cross-domain requests to the vulnerable application (using either the GET or the POST method).

The security impact of cross-site scripting vulnerabilities is dependent upon the nature of the vulnerable application, the kinds of data and functionality that it contains, and the other applications that belong to the same domain and organization. If the application is used only to display non-sensitive public content, with no authentication or access control functionality, then a cross-site scripting flaw may be considered low risk. However, if the same application resides on a domain that can access cookies for other more security-critical applications, then the vulnerability could be used to attack those other applications, and so may be considered high risk. Similarly, if the organization that owns the application is a likely target for phishing attacks, then the vulnerability could be leveraged to lend credibility to such attacks, by injecting Trojan functionality into the vulnerable application and exploiting users' trust in the organization in order to capture credentials for other applications that it owns. In many kinds of application, such as those providing online banking functionality, cross-site scripting should always be considered high risk.
The value of the query request parameter is copied into the HTML document as plain text between tags. The payload t1duy<script>alert(1)</script>iah91 was submitted in the query parameter. This input was echoed unmodified in the application's response.

This proof-of-concept attack demonstrates that it is possible to inject arbitrary JavaScript into the application's response.

Output

```
GET /search.jsp?query=asdfasdft1duy%3cscript%3ealert(1)%3c%2fscript%3eiah91 HTTP/1.1
Host: altoromutual.com
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; W
Chrome/107.0.5304.63 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,ima
change;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://altoromutual.com/index.jsp?content=busi
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 7012
Date: Thu, 03 Nov 2022 10:53:12 GMT
Connection: close
```

Remediation

In most situations where user-controllable data is copied into application responses, cross-site scripting attacks can be prevented using two layers of defenses:

Input should be validated as strictly as possible on arrival, given the kind of content that it is expected to contain. For example, personal names should consist of alphabetical and a small range of typographical characters, and be relatively short; a year of birth should consist of exactly four numerals; email addresses should match a well-defined regular expression. Input which fails the validation should be rejected, not sanitized.
User input should be HTML-encoded at any point where it is copied into application responses. All HTML metacharacters, including < > " ' and =, should be replaced with the corresponding HTML entities (< > etc).

In cases where the application's functionality allows users to author content using a restricted subset of HTML tags and attributes (for example, blog comments which allow limited formatting and linking), it is necessary to parse the supplied HTML to validate that it does not use any dangerous syntax; this is a non-trivial task.

Affected paths

/index.jsp
/sendFeedback
/doLogin

References

https://portswigger.net/web-security/cross-site-scripting
https://portswigger.net/web-security/cross-site-scripting/reflected
https://support.portswigger.net/customer/portal/articles/1965737-Methodology_XSS.html
https://cwe.mitre.org/data/definitions/79.html
https://cwe.mitre.org/data/definitions/80.html
https://cwe.mitre.org/data/definitions/116.html
https://cwe.mitre.org/data/definitions/159.html
https://capec.mitre.org/data/definitions/591.html

Vulnerability	Count	Severity
SQL injection	1	High

Description

SQL injection vulnerabilities arise when user-controllable data is incorporated into database SQL queries in an unsafe manner. An attacker can supply crafted input to break out of the data context in which their input appears and interfere with the structure of the surrounding query.

A wide range of damaging attacks can often be delivered via SQL injection, including reading or modifying critical application data, interfering with application logic, escalating privileges within the database and taking control of the database server.
The passw parameter appears to be vulnerable to SQL injection attacks. The payloads 76245223' or '8550'='8550 and 10648157' or '5738'='5741 were each submitted in the passw parameter. These two requests resulted in different responses, indicating that the input is being incorporated into a SQL query in an unsafe way.

Note that automated difference-based tests for SQL injection flaws can often be unreliable and are prone to false positive results. You should manually review the reported requests and responses to confirm whether a vulnerability is actually present.

Output

```
POST /doLogin HTTP/1.1
Host: altoromutual.com
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D
Content-Length: 37
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://altoromutual.com
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/107.0.5304.63 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://altoromutual.com/login.jsp
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Date: Thu, 03 Nov 2022 10:49:22 GMT
Connection: close
Content-Length: 8618

Remediation

The most effective way to prevent SQL injection attacks is to use parameterized queries (also known as prepared statements) for all database access. This method uses two steps to incorporate potentially tainted data into SQL queries: first, the application specifies the structure of the query, leaving placeholders for each item of user input; second, the application specifies the contents of each placeholder. Because the structure of the query has already been defined in the first step, it is not possible for malformed data in the second step to interfere with the query structure. You should review the documentation for your database and application platform to determine the appropriate APIs which you can use to perform parameterized queries. It is strongly recommended that you parameterize every variable data item that is incorporated into database queries, even if it is not obviously tainted, to prevent oversights occurring and avoid vulnerabilities being introduced by changes elsewhere within the code base of the application.

You should be aware that some commonly employed and recommended mitigations for SQL injection vulnerabilities are not always effective:

One common defense is to double up any single quotation marks appearing within user input before incorporating that input into a SQL query. This defense is designed to prevent malformed data from terminating the string into which it is inserted. However, if the data being incorporated into queries is numeric, then the defense may fail, because numeric data may not be encapsulated within quotes, in which case only a space is required to break out of the data context and interfere with the query. Further, in second-order SQL injection attacks, data that has been safely escaped when initially inserted into the database is subsequently read from the database and then passed back to it again. Quotation marks that have been doubled up initially will return to their original form when the data is reused, allowing the defense to be bypassed.
Another often cited defense is to use stored procedures for database access. While stored procedures can provide security benefits, they are not guaranteed to prevent SQL injection attacks. The same kinds of vulnerabilities that arise within standard dynamic SQL queries can arise if any SQL is dynamically constructed within stored procedures. Further, even if the procedure is sound, SQL injection can arise if the procedure is invoked in an unsafe manner using user-controllable data.

Affected paths

/index.jsp
/sendFeedback
/doLogin

References

https://portswigger.net/web-security/sql-injection
https://support.portswigger.net/customer/portal/articles/1965677-using-burp-to-test-for-injection-flaws
https://portswigger.net/web-security/sql-injection/cheat-sheet
https://cwe.mitre.org/data/definitions/89.html
https://cwe.mitre.org/data/definitions/94.html
https://cwe.mitre.org/data/definitions/116.html
https://capec.mitre.org/data/definitions/66.html

Vulnerability	Count	Severity
XPath injection	3	High

Description

XPath injection vulnerabilities arise when user-controllable data is incorporated into XPath queries in an unsafe manner. An attacker can supply crafted input to break out of the data context in which their input appears and interfere with the structure of the surrounding query.

Depending on the purpose for which the vulnerable query is being used, an attacker may be able to exploit an XPath injection flaw to read sensitive application data or interfere with application logic.
The Referer HTTP header appears to be vulnerable to XPath injection attacks. The payload ' was

submitted in the Referer HTTP header, and an XPath error message was returned. You should review the contents of the error message, and the application's handling of other input, to confirm whether a vulnerability is present.

Output

```
POST /doLogin HTTP/1.1
Host: altoromutual.com
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D
Content-Length: 37
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://altoromutual.com
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/107.0.5304.63 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://altoromutual.com/login.jsp'
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 6263
Date: Thu, 03 Nov 2022 10:50:55 GMT
Connection: close
```

Remediation

User input should be strictly validated before being
incorporated into XPath queries.
In most cases, it will be appropriate to accept input containing only short
alphanumeric strings. At the very least, input containing any XPath metacharacters
such as " ' / @ = * [ ] (and) should be rejected.

Affected paths

/
/doLogin

References

https://portswigger.net/web-security/dom-based/client-side-xpath-injection
https://cwe.mitre.org/data/definitions/94.html
https://cwe.mitre.org/data/definitions/116.html
https://cwe.mitre.org/data/definitions/159.html
https://cwe.mitre.org/data/definitions/643.html
https://capec.mitre.org/data/definitions/83.html

Vulnerability	Count	Severity
Client-side desync	1	High

Description

Client-side desync (CSD) vulnerabilities occur when a web server fails to correctly process the Content-Length of POST requests. By exploiting this behavior, an attacker can force a victim's browser to desynchronize its connection with the website, typically leading to XSS.

The server appears to be vulnerable to client-side desync attacks. A POST request was sent to the path '/bank/doTransfer' with a second request sent as the body. The server ignored the Content-Length header and did not close the connection, leading to the smuggled request being interpreted as the next request.

Output

```
GET /bank/doTransfer HTTP/1.1
Host: altoromutual.com
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 Safari/537.36
Connection: close
Cache-Control: max-age=0
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D; AltoroAccounts=ODAwMDAwfkNvcnBvcmF0ZX4xLjIxNTk0MDIwMzk0NjA5OTlFMTF8ODAwMDAxfkNoZWNraW5nfi0xLjIxNTQ0MTc5Njc3NTZFMTF8

HTTP/1.1 400 Bad Request
Server: Apache-Coyote/1.1
Date: Thu, 03 Nov 2022 10:45:26 GMT
Connection: close
Content-Length: 0
```

Remediation

A Strict-Transport-Security HTTP header should be sent with each HTTPS response. The syntax is as follows: <pre>Strict-Transport-Security: max-age=<seconds>[; includeSubDomains]</pre>
The parameter max-age gives the time frame for requirement of HTTPS in seconds and should be chosen quite high, e.g. several months. A value below 7776000 is considered as too low by this scanner check. The flag includeSubDomains defines that the policy applies also for sub domains of the sender of the response.

You can resolve this vulnerability by patching the server so that it either processes POST requests correctly, or closes the connection after handling them. You could also disable connection reuse entirely, but this may reduce performance.

You can also resolve this issue by enabling HTTP/2.

Affected paths

/bank/customize.jsp
/status_check.jsp
/login.jsp
/bank/queryxpath.jsp
/logout.jsp
/bank/ccApply
/bank/showAccount
/subscribe.jsp
/bank/stocks.jsp
/bank/transaction.jsp
/survey_questions.jsp
/swagger/index.html
/bank/transfer.jsp
/bank/apply.jsp
/sendFeedback
/doLogin
/bank/doTransfer
/bank/main.jsp
/a'a%5c'b%22c%3e%3f%3e%25%7d%7d%25%25%3ec%3c[[%3f$%7b%7b%25%7d%7dcake%5c/customize.jsp
/default.jsp
/disclaimer.htm
/index.jsp
/admin/admin.jsp
/search.jsp
/subscribe.swf
/feedback.jsp

References

https://portswigger.net/web-security/request-smuggling
https://portswigger.net/research/browser-powered-desync-attacks
https://cwe.mitre.org/data/definitions/444.html
https://capec.mitre.org/data/definitions/33.html

Medium Findings

Vulnerability	Count	Severity
Strict Transport Security Misconfiguration	27	Medium

Description

The HTTP Strict Transport Security policy defines a timeframe where a browser must connect to the web server via HTTPS. Without a Strict Transport Security policy the web application may be vulnerable against several attacks: <ul> If the web application mixes usage of HTTP and HTTPS, an attacker can manipulate pages in the unsecured area of the application or change redirection targets in a manner that the switch to the secured page is not performed or done in a manner, that the attacker remains between client and server. If there is no HTTP server, an attacker in the same network could simulate a HTTP server and motivate the user to click on a prepared URL by a social engineering attack.
The protection is effective only for the given amount of time. Multiple occurrence of this header could cause undefined behaviour in browsers and should be avoided.
There was no "Strict-Transport-Security" header in the server response.

Output

```
GET / HTTP/1.1
Host: altoromutual.com
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 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: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D; Path=/; Secure; HttpOnly
Content-Type: text/html;charset=ISO-8859-1
Date: Thu, 03 Nov 2022 10:40:41 GMT
Connection: close
Content-Length: 9369
```

Remediation

A Strict-Transport-Security HTTP header should be sent with each HTTPS response. The syntax is as follows: <pre>Strict-Transport-Security: max-age=<seconds>[; includeSubDomains]</pre>
The parameter max-age gives the time frame for requirement of HTTPS in seconds and should be chosen quite high, e.g. several months. A value below 7776000 is considered as too low by this scanner check. The flag includeSubDomains defines that the policy applies also for sub domains of the sender of the response.

User input should be strictly validated before being
incorporated into XPath queries.
In most cases, it will be appropriate to accept input containing only short
alphanumeric strings. At the very least, input containing any XPath metacharacters
such as " ' / @ = * [ ] (and) should be rejected.

Affected paths

/bank/customize.jsp
/status_check.jsp
/login.jsp
/bank/queryxpath.jsp
/logout.jsp
/bank/ccApply
/bank/showAccount
/subscribe.jsp
/bank/stocks.jsp
/bank/transaction.jsp
/survey_questions.jsp

/swagger/index.html
/bank/transfer.jsp
/bank/apply.jsp
/sendFeedback
/doLogin
/bank/doTransfer
/bank/main.jsp
/a'a%5c'b%22c%3e%3f%3e%25%7d%7d%25%25%3ec%3c[[%3f$%7b%7b%25%7d%7dcake%5c/customize.jsp
/default.jsp
/disclaimer.htm
/index.jsp
/admin/admin.jsp
/search.jsp
/subscribe.swf
/feedback.jsp

References

https://portswigger.net/web-security/dom-based/client-side-xpath-injection
https://cwe.mitre.org/data/definitions/94.html
https://cwe.mitre.org/data/definitions/116.html
https://cwe.mitre.org/data/definitions/159.html
https://cwe.mitre.org/data/definitions/643.html
https://capec.mitre.org/data/definitions/83.html

Vulnerability	Count	Severity
Cross-site request forgery	3	Medium

Description

Cross-site request forgery (CSRF) vulnerabilities may arise when applications rely solely on HTTP cookies to identify the user that has issued a particular request. Because browsers automatically add cookies to requests regardless of their origin, it may be possible for an attacker to create a malicious web site that forges a cross-domain request to the vulnerable application. For a request to be vulnerable to CSRF, the following conditions must hold:

The request can be issued cross-domain, for example using an HTML form. If the request contains non-standard headers or body content, then it may only be issuable from a page that originated on the same domain.
The application relies solely on HTTP cookies or Basic Authentication to identify the user that issued the request. If the application places session-related tokens elsewhere within the request, then it may not be vulnerable.
The request performs some privileged action within the application, which modifies the application's state based on the identity of the issuing user.
The attacker can determine all the parameters required to construct a request that performs the action. If the request contains any values that the attacker cannot determine or predict, then it is not vulnerable.
The request appears to be vulnerable to cross-site request forgery (CSRF) attacks against authenticated users.

Output

POST /admin/admin.jsp HTTP/1.1
Host: altoromutual.com
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D; AltoroAccounts=ODAwMDAwfkNvcnBvcmF0ZX4xLjIxNTk0MDIwMTYxNjA5OTlFMTF8ODAwMDAxfkNoZWNraW5nfi0xLjIxNTQ0MTc5NDQ0NTZFMTF8
Content-Length: 32
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://altoromutual.com
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/107.0.5304.63 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://SkoKyLKEUPvo.com/admin/admin.jsp
Accept-Encoding: gzip, deflate

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

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Date: Thu, 03 Nov 2022 10:49:12 GMT
Connection: close
Content-Length: 9224

Remediation

The most effective way to protect against CSRF vulnerabilities is to include within relevant requests an additional token that is not transmitted in a cookie: for example, a parameter in a hidden form field. This additional token should contain sufficient entropy, and be generated using a cryptographic random number generator, such that it is not feasible for an attacker to determine or predict the value of any token that was issued to another user. The token should be associated with the user's session, and the application should validate that the correct token is received before performing any action resulting from the request.

An alternative approach, which may be easier to implement, is to validate that Host and Referer headers in relevant requests are both present and contain the same domain name. However, this approach is somewhat less robust: historically, quirks in browsers and plugins have often enabled attackers to forge cross-domain requests that manipulate these headers to bypass such defenses.

Affected paths

/doLogin
/disclaimer.htm
/bank/doTransfer

References

https://portswigger.net/web-security/csrf
https://support.portswigger.net/customer/portal/articles/1965674-using-burp-to-test-for-cross-site-request-forgery-csrf-
https://media.blackhat.com/eu-13/briefings/Lundeen/bh-eu-13-deputies-still-confused-lundeen-wp.pdf
https://cwe.mitre.org/data/definitions/352.html
https://capec.mitre.org/data/definitions/62.html

Low Findings

Vulnerability	Count	Severity
Password field with autocomplete enabled	3	Low

Description

Most browsers have a facility to remember user credentials that are entered into HTML forms. This function can be configured by the user and also by applications that employ user credentials. If the function is enabled, then credentials entered by the user are stored on their local computer and retrieved by the browser on future visits to the same application.

The stored credentials can be captured by an attacker who gains control over the user's computer. Further, an attacker who finds a separate application vulnerability such as cross-site scripting may be able to exploit this to retrieve a user's browser-stored credentials.
The page contains a form with the following action URL:
https://altoromutual.com/admin/admin.jsp
The form contains the following password fields with autocomplete enabled:
password1
password2

Output

GET /admin/admin.jsp HTTP/1.1
Host: altoromutual.com
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D; AltoroAccounts=ODAwMDAwfkNvcnBvcmF0ZX4xLjIxNTk0MDIwMzk0NjA5OTlFMTF8ODAwMDAxfkNoZWNraW5nfi0xLjIxNTQ0MTc5Njc3NTZFMTF8
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 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://altoromutual.com/bank/customize.jsp
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Date: Thu, 03 Nov 2022 10:41:58 GMT
Connection: close
Content-Length: 9224

Remediation

To prevent browsers from storing credentials entered into HTML forms, include the attribute autocomplete="off" within the FORM tag (to protect all form fields) or within the relevant INPUT tags (to protect specific individual fields).

Please note that modern web browsers may ignore this directive. In spite of this there is a chance that not disabling autocomplete may cause problems obtaining PCI compliance.

Affected paths

/bank/apply.jsp
/
/login.jsp

References

https://cwe.mitre.org/data/definitions/200.html

Vulnerability	Count	Severity
Strict transport security not enforced	6	Low

Description

The application fails to prevent users from connecting to it over unencrypted connections. An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP, so that if a targeted user follows a link to the site from an HTTP page, their browser never attempts to use an encrypted connection. The sslstrip tool automates this process.

To exploit this vulnerability, an attacker must be suitably positioned to intercept and modify the victim's network traffic.This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.
This issue was found in multiple locations under the reported path.

Output

GET /favicon.ico HTTP/1.1
Host: altoromutual.com
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 Safari/537.36
Sec-Ch-Ua-Platform: "Linux"
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: https://altoromutual.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 404 Not Found
Server: Apache-Coyote/1.1

Content-Type: text/html;charset=ISO-8859-1
Content-Length: 6922
Date: Thu, 03 Nov 2022 10:40:43 GMT
Connection: close

Remediation

The application should instruct web browsers to only access the application using HTTPS. To do this, enable HTTP Strict Transport Security (HSTS) by adding a response header with the name 'Strict-Transport-Security' and the value 'max-age=expireTime', where expireTime is the time in seconds that browsers should remember that the site should only be accessed using HTTPS. Consider adding the 'includeSubDomains' flag if appropriate.

Note that because HSTS is a "trust on first use" (TOFU) protocol, a user who has never accessed the application will never have seen the HSTS header, and will therefore still be vulnerable to SSL stripping attacks. To mitigate this risk, you can optionally add the 'preload' flag to the HSTS header, and submit the domain for review by browser vendors.

Affected paths

/bank/customize.jsp
/
/bank/doTransfer
/bank/apply.jsp
/admin/admin.jsp
/bank/main.jsp

References

https://developer.mozilla.org/en-US/docs/Web/Security/HTTP_strict_transport_security
https://github.com/moxie0/sslstrip
https://hstspreload.appspot.com/
https://cwe.mitre.org/data/definitions/523.html
https://capec.mitre.org/data/definitions/94.html
https://capec.mitre.org/data/definitions/157.html

Vulnerability	Count	Severity
Browser cross-site scripting filter misconfiguration	27	Low

Description

Cross-site scripting (XSS) filters in browsers check if the URL contains possible harmful XSS payloads and if they are reflected in the response page. If such a condition is recognized, the injected code is changed in a way, that it is not executed anymore to prevent a succesful XSS attack. The downside of these filters is, that the browser has no possibility to distinguish between code fragments which were reflected by a vulnerable web application in an XSS attack and these which are already present on the page. In the past, these filters were used by attackers to deactivate JavaScript code on the attacked web page. Sometimes the XSS filters itself are vulnerable in a way, that web applications which were protected properly against XSS attacks became vulnerable under certain conditions.
No X-XSS-Protection header was set in the response. This means that the browser uses default behavior that detection of a cross-site scripting attack never prevents rendering.

Output

GET / HTTP/1.1
Host: altoromutual.com
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 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: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D; Path=/; Secure; HttpOnly

Content-Type: text/html;charset=ISO-8859-1
Date: Thu, 03 Nov 2022 10:40:41 GMT
Connection: close
Content-Length: 9369

Remediation

The following header should be set:
<pre>X-XSS-Protection: 1; mode=block</pre>
It is considered as better practice to instruct the browser XSS filter to never render the web page if an XSS attack is detected.

Affected paths

/bank/customize.jsp
/status_check.jsp
/login.jsp
/bank/queryxpath.jsp
/logout.jsp
/bank/ccApply
/bank/showAccount
/subscribe.jsp
/bank/stocks.jsp
/bank/transaction.jsp
/survey_questions.jsp
/swagger/index.html
/bank/transfer.jsp
/bank/apply.jsp
/sendFeedback
/doLogin
/bank/doTransfer
/bank/main.jsp
/a'a%5c'b%22c%3e%3f%3e%25%7d%7d%25%25%3ec%3c[[%3f$%7b%7b%25%7d%7dcake%5c/customize.jsp
/default.jsp
/disclaimer.htm
/
/index.jsp
/admin/admin.jsp
/search.jsp
/subscribe.swf
/feedback.jsp

References

https://developer.mozilla.org/en-US/docs/Web/Security/HTTP_strict_transport_security
https://github.com/moxie0/sslstrip
https://hstspreload.appspot.com/
https://cwe.mitre.org/data/definitions/523.html
https://capec.mitre.org/data/definitions/94.html
https://capec.mitre.org/data/definitions/157.html

Vulnerability	Count	Severity
Software Version Numbers Revealed	3	Low

Description

Cross-site scripting (XSS) filters in browsers check if the URL contains possible harmful XSS payloads and if they are reflected in the response page. If such a condition is recognized, the injected code is changed in a way, that it is not executed anymore to prevent a succesful XSS attack. The downside of these filters is, that the browser has no possibility to distinguish between code fragments which were reflected by a vulnerable web application in an XSS attack and these which are already present on the page. In the past, these filters were used by attackers to deactivate JavaScript code on the attacked web page. Sometimes the XSS filters itself are vulnerable in a way, that web applications which were protected properly against XSS attacks became vulnerable under certain conditions.
The server software versions used by the application are revealed by the web server.
Displaying version information of software information could allow an attacker to determine which vulnerabilities are present in the software, particularly if an outdated software version is in use with published vulnerabilities.

The following software appears to be in use:

Apache Coyote (Tomcat): 1.1

Output

GET / HTTP/1.1
Host: altoromutual.com
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"

Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 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: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D; Path=/; Secure; HttpOnly
Content-Type: text/html;charset=ISO-8859-1
Date: Thu, 03 Nov 2022 10:40:41 GMT
Connection: close
Content-Length: 9369

Remediation

The following header should be set:
<pre>X-XSS-Protection: 1; mode=block</pre>
It is considered as better practice to instruct the browser XSS filter to never render the web page if an XSS attack is detected.

Affected paths

/swagger/properties.json
/
/bank/doTransfer

References

Vulnerability	Count	Severity
Content Sniffing not disabled	27	Low

Description

There was no "X-Content-Type-Options" HTTP header with the value nosniff set in the response. The lack of this header causes that certain browsers, try to determine the content type and encoding of the response even when these properties are defined correctly. This can make the web application vulnerable against Cross-Site Scripting (XSS) attacks. E.g. the Internet Explorer and Safari treat responses with the content type text/plain as HTML, if they contain HTML tags.
The server software versions used by the application are revealed by the web server.
Displaying version information of software information could allow an attacker to determine which vulnerabilities are present in the software, particularly if an outdated software version is in use with published vulnerabilities.

The following software appears to be in use:

Apache Coyote (Tomcat): 1.1<li>Generic: 1.0.2

Output

GET / HTTP/1.1
Host: altoromutual.com
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 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: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D; Path=/; Secure; HttpOnly
Content-Type: text/html;charset=ISO-8859-1
Date: Thu, 03 Nov 2022 10:40:41 GMT
Connection: close
Content-Length: 9369

Remediation

Set the following HTTP header at least in all responses which contain user input: <pre>X-Content-Type-Options: nosniff</pre>

Affected paths

/bank/customize.jsp
/status_check.jsp
/login.jsp
/bank/queryxpath.jsp
/logout.jsp
/bank/ccApply
/bank/showAccount
/subscribe.jsp
/bank/stocks.jsp
/bank/transaction.jsp
/survey_questions.jsp
/swagger/index.html
/bank/transfer.jsp
/bank/apply.jsp
/sendFeedback
/doLogin
/bank/doTransfer
/bank/main.jsp
/a'a%5c'b%22c%3e%3f%3e%25%7d%7d%25%25%3ec%3c[[%3f$%7b%7b%25%7d%7dcake%5c/customize.jsp
/default.jsp
/disclaimer.htm
/index.jsp
/admin/admin.jsp
/search.jsp
/subscribe.swf
/feedback.jsp

References

Vulnerability	Count	Severity
Arbitrary host header accepted	1	Low

Description

There was no "X-Content-Type-Options" HTTP header with the value nosniff set in the response. The lack of this header causes that certain browsers, try to determine the content type and encoding of the response even when these properties are defined correctly. This can make the web application vulnerable against Cross-Site Scripting (XSS) attacks. E.g. the Internet Explorer and Safari treat responses with the content type text/plain as HTML, if they contain HTML tags.
The application appears to be accessible using arbitrary HTTP Host headers.

This is a serious issue if the application is not externally accessible or uses IP-based access restrictions. Attackers can use DNS Rebinding to bypass any IP or firewall based access restrictions that may be in place, by proxying through their target's browser.
Note that modern web browsers' use of DNS pinning does not effectively prevent this attack. The only effective mitigation is server-side: https://bugzilla.mozilla.org/show_bug.cgi?id=689835#c13

Additionally, it may be possible to directly bypass poorly implemented access restrictions by sending a Host header of 'localhost'.

Resources:

https://portswigger.net/web-security/host-header

Output

GET /disclaimer.htm?url=http://www.microsoft.com&cachebust=1667472473.14 HTTP/1.1

Host: v6ufxu.altoromutual.com
Cache-Control: no-cache
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 Safari/537.36
Connection: close
Cache-Control: max-age=0

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Accept-Ranges: bytes
ETag: W/"2083-1497451722000"
Last-Modified: Wed, 14 Jun 2017 14:48:42 GMT
Content-Type: text/html
Content-Length: 2083
Date: Thu, 03 Nov 2022 10:47:52 GMT
Connection: close

Remediation

Set the following HTTP header at least in all responses which contain user input: <pre>X-Content-Type-Options: nosniff</pre>

Affected paths

/bank/customize.jsp
/status_check.jsp
/login.jsp
/bank/queryxpath.jsp
/logout.jsp
/bank/ccApply
/bank/showAccount
/subscribe.jsp
/bank/stocks.jsp
/bank/transaction.jsp
/survey_questions.jsp
/swagger/index.html
/bank/transfer.jsp
/bank/apply.jsp
/sendFeedback
/doLogin
/bank/doTransfer
/bank/main.jsp
/a'a%5c'b%22c%3e%3f%3e%25%7d%7d%25%25%3ec%3c[[%3f$%7b%7b%25%7d%7dcake%5c/customize.jsp
/default.jsp
/disclaimer.htm
/index.jsp
/admin/admin.jsp
/search.jsp
/subscribe.swf
/feedback.jsp

References

Vulnerability	Count	Severity
Link manipulation (DOM-based)	2	Low

Description

DOM-based vulnerabilities arise when a client-side script reads data from a controllable part of the DOM (for example, the URL) and processes this data in an unsafe way.

DOM-based link manipulation arises when a script writes controllable data to a navigation target within the current page, such as a clickable link or the submission URL of a form. An attacker may be able to use the vulnerability to construct a URL that, if visited by another application user, will modify the target of links within the response. An attacker may be able to leverage this to perform various attacks, including:

Causing the user to redirect to an arbitrary external URL, to facilitate a phishing attack.
Causing the user to

submit sensitive form data to a server controlled by the attacker.
Causing the user to perform an unintended action within the application, by changing the file or query string associated with a link.
Bypassing browser anti-XSS defenses by injecting on-site links containing XSS exploits, since browser anti-XSS defenses typically do not operate on on-site links.

Burp Suite automatically identifies this issue using dynamic and static code analysis. Static analysis can lead to false positives that are not actually exploitable. If Burp Scanner has not provided any evidence resulting from dynamic analysis, you should review the relevant code and execution paths to determine whether this vulnerability is indeed present, or whether mitigations are in place that would prevent exploitation.
The application may be vulnerable to DOM-based link manipulation. Data is read from document.URL and passed to the 'href' property of a DOM element.

Output

```
GET /disclaimer.htm HTTP/1.1
Host: altoromutual.com
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 Safari/537.36
Connection: close
Cache-Control: max-age=0

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Accept-Ranges: bytes
ETag: W/"2083-1497451722000"
Last-Modified: Wed, 14 Jun 2017 14:48:42 GMT
Content-Type: text/html
Content-Length: 2083
Date: Thu, 03 Nov 2022 10:42:37 GMT
Connection: close
```

Remediation

The most effective way to avoid DOM-based link manipulation vulnerabilities is not to dynamically set the target URLs of links or forms using data that originated from any untrusted source. If the desired functionality of the application means that this behavior is unavoidable, then defenses must be implemented within the client-side code to prevent malicious data from introducing an arbitrary URL as a link target. In general, this is best achieved by using a whitelist of URLs that are permitted link targets, and strictly validating the target against this list before setting the link target.

Affected paths

/disclaimer.htm

References

https://portswigger.net/web-security/dom-based/link-manipulation
https://cwe.mitre.org/data/definitions/20.html
https://capec.mitre.org/data/definitions/153.html

Vulnerability	Count	Severity
Open redirection (DOM-based)	4	Low

Description

DOM-based vulnerabilities arise when a client-side script reads data from a controllable part of the DOM (for example, the URL) and processes this data in an unsafe way.

DOM-based open redirection arises when a script writes controllable data into the target of a redirection in an unsafe way. An attacker may be able to use the vulnerability to construct a URL that, if visited by another application user, will cause a redirection to an arbitrary external domain. This behavior can be leveraged to facilitate phishing attacks against users of the application. The ability to use an authentic application URL, targeting the correct domain and with a valid SSL certificate (if SSL is used), lends credibility to the phishing attack because many users, even if they verify these features, will not notice the subsequent redirection to a different domain.

Note: If an attacker is able to control the start of the string that is passed to the redirection API, then it may be possible to escalate this vulnerability into a JavaScript injection attack, by using a URL with the javascript: pseudo-protocol to execute arbitrary script code when the URL is processed by the browser.

Burp Suite automatically identifies this issue using dynamic and static code analysis. Static analysis can lead to false positives that are not actually exploitable. If Burp Scanner has not provided any evidence resulting from dynamic

analysis, you should review the relevant code and execution paths to determine whether this vulnerability is indeed present, or whether mitigations are in place that would prevent exploitation.
The application may be vulnerable to DOM-based open redirection. Data is read from document.URL and passed to window.location.href.

Output

```
GET /disclaimer.htm HTTP/1.1
Host: altoromutual.com
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 Safari/537.36
Connection: close
Cache-Control: max-age=0

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Accept-Ranges: bytes
ETag: W/"2083-1497451722000"
Last-Modified: Wed, 14 Jun 2017 14:48:42 GMT
Content-Type: text/html
Content-Length: 2083
Date: Thu, 03 Nov 2022 10:42:37 GMT
Connection: close
```

Remediation

The most effective way to avoid DOM-based open redirection vulnerabilities is not to dynamically set redirection targets using data that originated from any untrusted source. If the desired functionality of the application means that this behavior is unavoidable, then defenses must be implemented within the client-side code to prevent malicious data from introducing an arbitrary URL as a redirection target. In general, this is best achieved by using a whitelist of URLs that are permitted redirection targets, and strictly validating the target against this list before performing the redirection.

Affected paths

/swagger/index.html
/index.jsp
/disclaimer.htm

References

https://portswigger.net/web-security/dom-based/open-redirection
https://cwe.mitre.org/data/definitions/601.html

Information Findings

Vulnerability	Count	Severity
Input returned in response (reflected)	4	Information

Description

Reflection of input arises when data is copied from a request and echoed into the application's immediate response. Input being returned in application responses is not a vulnerability in its own right. However, it is a prerequisite for many client-side vulnerabilities, including cross-site scripting, open redirection, content spoofing, and response header injection. Additionally, some server-side vulnerabilities such as SQL injection are often easier to identify and exploit when input is returned in responses. In applications where input retrieval is rare and the environment is resistant to automated testing (for example, due to a web application firewall), it might be worth subjecting instances of it to focused manual testing.
The value of the content request parameter is copied into the application's response.

Output

```
GET /index.jsp?content=business.htmwt3whkxgoq HTTP/1.1
Host: altoromutual.com
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 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://altoromutual.com/index.jsp?content=inside.htm
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 6921
Date: Thu, 03 Nov 2022 10:49:14 GMT
Connection: close

Remediation

The most effective way to avoid DOM-based open redirection vulnerabilities is not to dynamically set redirection targets using data that originated from any untrusted source. If the desired functionality of the application means that this behavior is unavoidable, then defenses must be implemented within the client-side code to prevent malicious data from introducing an arbitrary URL as a redirection target. In general, this is best achieved by using a whitelist of URLs that are permitted redirection targets, and strictly validating the target against this list before performing the redirection.

Affected paths

/sendFeedback
/search.jsp
/doLogin

References

https://portswigger.net/web-security/dom-based/open-redirection
https://cwe.mitre.org/data/definitions/20.html
https://cwe.mitre.org/data/definitions/116.html

Vulnerability	Count	Severity
TLS cookie without secure flag set	1	Information

Description

If the secure flag is set on a cookie, then browsers will not submit the cookie in any requests that use an unencrypted HTTP connection, thereby preventing the cookie from being trivially intercepted by an attacker monitoring network traffic. If the secure flag is not set, then the cookie will be transmitted in clear-text if the user visits any HTTP URLs within the cookie's scope. An attacker may be able to induce this event by feeding a user suitable links, either directly or via another web site. Even if the domain that issued the cookie does not host any content that is accessed over HTTP, an attacker may be able to use links of the form http://example.com:443/ to perform the same attack.

To exploit this vulnerability, an attacker must be suitably positioned to eavesdrop on the victim's network traffic. This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.
The following cookie was issued by the application and does not have the secure flag set:
AltoroAccounts
The cookie does not appear to contain a session token, which may reduce the risk associated with this issue. You should review the contents of the cookie to determine its function.

Output

POST /doLogin HTTP/1.1
Host: altoromutual.com
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D
Content-Length: 37
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://altoromutual.com
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/107.0.5304.63 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://altoromutual.com/login.jsp
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 302 Found
Server: Apache-Coyote/1.1
Set-Cookie:
AltoroAccounts=ODAwMDAwfkNvcnBvcmF0ZX4xLjIxNTk0MDIwMzk0NjA5OTlFMTF8ODAwMDAxfkNoZWNraW5nfi0xLjIxNTQ0MTc5Njc3NTZFMTF8
Location: /bank/main.jsp
Content-Length: 0
Date: Thu, 03 Nov 2022 10:41:30 GMT
Connection: close

Remediation

The secure flag should be set on all cookies that are used for transmitting sensitive data when accessing content over HTTPS. If cookies are used to transmit session tokens, then areas of the application that are accessed over HTTPS should employ their own session handling mechanism, and the session tokens used should never be transmitted over unencrypted communications.

Affected paths

/sendFeedback
/search.jsp
/doLogin

References

https://cwe.mitre.org/data/definitions/614.html

Vulnerability	Count	Severity
Cross-domain Referer leakage	4	Information

Description

When a web browser makes a request for a resource, it typically adds an HTTP header, called the "Referer" header, indicating the URL of the resource from which the request originated. This occurs in numerous situations, for example when a web page loads an image or script, or when a user clicks on a link or submits a form.

If the resource being requested resides on a different domain, then the Referer header is still generally included in the cross-domain request. If the originating URL contains any sensitive information within its query string, such as a session token, then this information will be transmitted to the other domain. If the other domain is not fully trusted by the application, then this may lead to a security compromise.

You should review the contents of the information being transmitted to other domains, and also determine whether those domains are fully trusted by the originating application.

Today's browsers may withhold the Referer header in some situations (for example, when loading a non-HTTPS resource from a page that was loaded over HTTPS, or when a Refresh directive is issued), but this behavior should not be relied upon to protect the originating URL from disclosure.

Note also that if users can author content within the application then an attacker may be able to inject links referring to a domain they control in order to capture data from URLs used within the application.
The application contains the following link to another domain from URLs containing a query string:
https://github.com/AppSecDev/AltoroJ/
This issue was found in multiple locations under the reported path.

Output

GET /bank/showAccount?listAccounts=800000 HTTP/1.1
Host: altoromutual.com
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D;
AltoroAccounts=ODAwMDAwfkNvcnBvcmF0ZX4xLjIxNTk0MDIwMzk0NjA5OTlFMTF8ODAwMDAxfkNoZWNraW5nfi0xLjIxNTQ0MTc5Njc3NTZFMTF8

Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 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://altoromutual.com/bank/showAccount?listAccounts=800000
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Date: Thu, 03 Nov 2022 10:41:32 GMT
Connection: close
Content-Length: 80569

Remediation

Applications should never transmit any sensitive information within the URL query string. In addition to being leaked in the Referer header, such information may be logged in various locations and may be visible on-screen to untrusted parties. If placing sensitive information in the URL is unavoidable, consider using the Referer-Policy HTTP header to reduce the chance of it being disclosed to third parties.

Affected paths

/bank/showAccount
/index.jsp
/search.jsp

References

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referrer-Policy
https://portswigger.net/web-security/information-disclosure
https://cwe.mitre.org/data/definitions/200.html

Vulnerability	Count	Severity
Cross-domain script include	1	Information

Description

When an application includes a script from an external domain, this script is executed by the browser within the security context of the invoking application. The script can therefore do anything that the application's own scripts can do, such as accessing application data and performing actions within the context of the current user.

If you include a script from an external domain, then you are trusting that domain with the data and functionality of your application, and you are trusting the domain's own security to prevent an attacker from modifying the script to perform malicious actions within your application.
The response dynamically includes the following script from another domain:
http://demo-analytics.testfire.net/urchin.js

Output

GET /index.jsp?content=personal_investments.htm HTTP/1.1
Host: altoromutual.com
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 Safari/537.36
Connection: close
Cache-Control: max-age=0

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1

Content-Type: text/html;charset=ISO-8859-1
Date: Thu, 03 Nov 2022 10:42:38 GMT
Connection: close
Content-Length: 8223

Remediation

Scripts should ideally not be included from untrusted domains. Applications that rely on static third-party scripts should consider using Subresource Integrity to make browsers verify them, or copying the contents of these scripts onto their own domain and including them from there. If that is not possible (e.g. for licensing reasons) then consider reimplementing the script's functionality within application code.

Affected paths

/bank/showAccount
/index.jsp
/search.jsp

References

https://developer.mozilla.org/en-US/docs/Web/Security/Subresource_Integrity
https://cwe.mitre.org/data/definitions/829.html

Vulnerability	Count	Severity
Cookie without HttpOnly flag set	1	Information

Description

If the HttpOnly attribute is set on a cookie, then the cookie's value cannot be read or set by client-side JavaScript. This measure makes certain client-side attacks, such as cross-site scripting, slightly harder to exploit by preventing them from trivially capturing the cookie's value via an injected script.
The following cookie was issued by the application and does not have the HttpOnly flag set:
AltoroAccounts
The cookie does not appear to contain a session token, which may reduce the risk associated with this issue. You should review the contents of the cookie to determine its function.

Output

POST /doLogin HTTP/1.1
Host: altoromutual.com
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D
Content-Length: 37
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://altoromutual.com
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/107.0.5304.63 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://altoromutual.com/login.jsp
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 302 Found
Server: Apache-Coyote/1.1
Set-Cookie:
AltoroAccounts=ODAwMDAwfkNvcnBvcmF0ZX4xLjIxNTk0MDIwMzk0NjA5OTlFMTF8ODAwMDAxfkNoZWNraW5nfi0xLjIxNTQ0MTc5Njc3NTZFMTF8
Location: /bank/main.jsp
Content-Length: 0
Date: Thu, 03 Nov 2022 10:41:30 GMT
Connection: close

Remediation

There is usually no good reason not to set the HttpOnly flag on all cookies. Unless you specifically require legitimate client-side scripts within your application to read or set a cookie's value, you should set the HttpOnly flag by including this attribute within the relevant Set-cookie directive.

You should be aware that the restrictions imposed by the HttpOnly flag can potentially be circumvented in some circumstances, and that numerous other serious attacks can be delivered by client-side script injection, aside from simple cookie stealing.

Affected paths

/doLogin

References

https://portswigger.net/web-security/cross-site-scripting/exploiting
https://portswigger.net/research/web-storage-the-lesser-evil-for-session-tokens#httponly
https://cwe.mitre.org/data/definitions/16.html
https://capec.mitre.org/data/definitions/31.html

Vulnerability	Count	Severity
Backup file	2	Information

Description

Publicly accessible backups and outdated copies of files can provide attackers with extra attack surface. Depending on the server configuration and file type, they may also expose source code, configuration details, and other information intended to remain secret.
The following cookie was issued by the application and does not have the HttpOnly flag set:
AltoroAccounts
The cookie does not appear to contain a session token, which may reduce the risk associated with this issue. You should review the contents of the cookie to determine its function.

Output

```
GET /bank/icy.jsp.7z HTTP/1.1
Host: altoromutual.com
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D;
AltoroAccounts=ODAwMDAwfkNvcnBvcmF0ZX4xLjIxNTk0MDIwMzgyNjA5OTlFMTF8ODAwMDAxfkNoZWNraW5nfi0xLjIxNTQ0MTc5NjY1NTZFMTF8
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 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://altoromutual.com/bank/showAccount?listAccounts=800000
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 404 Not Found
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 6918
Date: Thu, 03 Nov 2022 10:48:33 GMT
Connection: close
```

Remediation

Review the file to identify whether it's intended to be publicly accessible, and remove it from the server's web root if it isn't. It may also be worth auditing the server contents to find other outdated files, and taking measures to prevent the problem from reoccurring.

Affected paths

/
/bank/transfer.jsp

References

https://portswigger.net/web-security/information-disclosure/exploiting#source-code-disclosure-via-backup-files
https://cwe.mitre.org/data/definitions/530.html
https://capec.mitre.org/data/definitions/37.html
https://capec.mitre.org/data/definitions/204.html

Vulnerability	Count	Severity
Cacheable HTTPS response	9	Information

Description

Unless directed otherwise, browsers may store a local cached copy of content received from web servers. Some browsers, including Internet Explorer, cache content accessed via HTTPS. If sensitive information in application responses is stored in the local cache, then this may be retrieved by other users who have access to the same computer at a future time.
This issue was found in multiple locations under the reported path.

Output

```
POST /bank/doTransfer HTTP/1.1
Host: altoromutual.com
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D; AltoroAccounts=ODAwMDAwfkNvcnBvcmF0ZX4xLjIxNTk0MDIwMzk0NjA5OTlFMTF8ODAwMDAxfkNoZWNraW5nfi0xLjIxNTQ0MTc5Njc3NTZFMTF8
Content-Length: 76
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://altoromutual.com
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/107.0.5304.63 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://altoromutual.com/bank/transfer.jsp
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 7406
Date: Thu, 03 Nov 2022 10:41:47 GMT
Connection: close
```

Remediation

Applications should return caching directives instructing browsers not to store local copies of any sensitive data. Often, this can be achieved by configuring the web server to prevent caching for relevant paths within the web root. Alternatively, most web development platforms allow you to control the server's caching directives from within individual scripts. Ideally, the web server should return the following HTTP headers in all responses containing sensitive content:

Cache-control: no-store
Pragma: no-cache

Affected paths

/bank/showAccount
/login.jsp
/index.jsp
/sendFeedback
/
/bank/transfer.jsp
/bank/main.jsp
/search.jsp

/feedback.jsp

References

https://portswigger.net/web-security/information-disclosure
https://cwe.mitre.org/data/definitions/524.html
https://cwe.mitre.org/data/definitions/525.html
https://capec.mitre.org/data/definitions/37.html

Vulnerability	Count	Severity
TLS certificate	1	Information

Description

TLS (or SSL) helps to protect the confidentiality and integrity of information in transit between the browser and server, and to provide authentication of the server's identity. To serve this purpose, the server must present an TLS certificate that is valid for the server's hostname, is issued by a trusted authority and is valid for the current date. If any one of these requirements is not met, TLS connections to the server will not provide the full protection for which TLS is designed.

It should be noted that various attacks exist against TLS in general, and in the context of HTTPS web connections in particular. It may be possible for a determined and suitably-positioned attacker to compromise TLS connections without user detection even when a valid TLS certificate is used.
The server presented a valid, trusted TLS certificate. This issue is purely informational.

The server presented the following certificates:

<h4>Server certificate</h4><table><tr><td>Issued to: </td><td>demo.testfire.net, altoromutual.com</td></tr><tr><td>Issued by: </td><td>Sectigo RSA Domain Validation Secure Server CA</td></tr><tr><td>Valid from: </td><td>Tue Jun 14 20:00:00 EDT 2022</td></tr><tr><td>Valid to: </td><td>Sun Jul 16 19:59:59 EDT 2023</td></tr></table><h4>Certificate chain #1</h4><table><tr><td>Issued to: </td><td>Sectigo RSA Domain Validation Secure Server CA</td></tr><tr><td>Issued by: </td><td>USERTrust RSA Certification Authority</td></tr><tr><td>Valid from: </td><td>Thu Nov 01 20:00:00 EDT 2018</td></tr><tr><td>Valid to: </td><td>Tue Dec 31 18:59:59 EST 2030</td></tr></table><h4>Certificate chain #2</h4><table><tr><td>Issued to: </td><td>USERTrust RSA Certification Authority</td></tr><tr><td>Issued by: </td><td>AAA Certificate Services</td></tr><tr><td>Valid from: </td><td>Mon Mar 11 20:00:00 EDT 2019</td></tr><tr><td>Valid to: </td><td>Sun Dec 31 18:59:59 EST 2028</td></tr></table><h4>Certificate chain #3</h4><table><tr><td>Issued to: </td><td>AAA Certificate Services</td></tr><tr><td>Issued by: </td><td>AAA Certificate Services</td></tr><tr><td>Valid from: </td><td>Wed Dec 31 19:00:00 EST 2003</td></tr><tr><td>Valid to: </td><td>Sun Dec 31 18:59:59 EST 2028</td></tr></table><h4>Certificate chain #4</h4><table><tr><td>Issued to: </td><td>AAA Certificate Services</td></tr><tr><td>Issued by: </td><td>AAA Certificate Services</td></tr><tr><td>Valid from: </td><td>Wed Dec 31 19:00:00 EST 2003</td></tr><tr><td>Valid to: </td><td>Sun Dec 31 18:59:59 EST 2028</td></tr></table>

Output

```
POST /sendFeedback HTTP/1.1
Host: altoromutual.com
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D
Content-Length: 87
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://altoromutual.com
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/107.0.5304.63 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://altoromutual.com/feedback.jsp
Accept-Encoding: gzip, deflate
```

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

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 7163
Date: Thu, 03 Nov 2022 10:41:14 GMT
Connection: close

Remediation

Applications should return caching directives instructing browsers not to store local copies of any sensitive data. Often, this can be achieved by configuring the web server to prevent caching for relevant paths within the web root. Alternatively, most web development platforms allow you to control the server's caching directives from within individual scripts. Ideally, the web server should return the following HTTP headers in all responses containing sensitive content:

Cache-control: no-store
Pragma: no-cache

Affected paths

/bank/showAccount
/login.jsp
/index.jsp
/sendFeedback
/
/bank/transfer.jsp
/bank/main.jsp
/search.jsp
/feedback.jsp

References

https://wiki.mozilla.org/Security/Server_Side_TLS
https://cwe.mitre.org/data/definitions/295.html
https://cwe.mitre.org/data/definitions/326.html
https://cwe.mitre.org/data/definitions/327.html

Vulnerability	Count	Severity
Mixed content	2	Information

Description

The application loads pages over HTTPS that load other resources over unencrypted connections. An attacker suitably positioned to view a legitimate user's network traffic could record and monitor their interactions with these resources, which may indirectly disclose information about the user's activity on the application itself. Furthermore, an attacker able to modify traffic could alter these resources and potentially influence the application's appearance and behavior. Due to these concerns, users' web browsers may automatically display warnings and disable affected components of the page. As a result, this vulnerability currently has more of an impact on usability than security.
To exploit this vulnerability, an attacker must be suitably positioned to eavesdrop on the victim's network traffic. This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.
The response is loaded over HTTPS, but loads other resources over an unencrypted connection. The following "active" resource is loaded over HTTP. An attacker able to modify traffic could cause execution of arbitrary script within the user's browser:
http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab

Output

GET /index.jsp?content=inside_contact.htm HTTP/1.1
Host: altoromutual.com
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 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://altoromutual.com/search.jsp?query=asdfasdf
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Date: Thu, 03 Nov 2022 10:41:06 GMT
Connection: close
Content-Length: 10113

Remediation

Ensure that all external resources the page references are loaded using HTTPS.

Affected paths

/index.jsp
/

References

https://developer.mozilla.org/en/docs/Security/MixedContent
https://cwe.mitre.org/data/definitions/16.html
https://cwe.mitre.org/data/definitions/319.html
https://capec.mitre.org/data/definitions/117.html

Vulnerability	Count	Severity
Frameable response (potential Clickjacking)	8	Information

Description

If a page fails to set an appropriate X-Frame-Options or Content-Security-Policy HTTP header, it might be possible for a page controlled by an attacker to load it within an iframe. This may enable a clickjacking attack, in which the attacker's page overlays the target application's interface with a different interface provided by the attacker. By inducing victim users to perform actions such as mouse clicks and keystrokes, the attacker can cause them to unwittingly carry out actions within the application that is being targeted. This technique allows the attacker to circumvent defenses against cross-site request forgery, and may result in unauthorized actions.

Note that some applications attempt to prevent these attacks from within the HTML page itself, using "framebusting" code. However, this type of defense is normally ineffective and can usually be circumvented by a skilled attacker.

You should determine whether any functions accessible within frameable pages can be used by application users to perform any sensitive actions within the application.
This issue was found in multiple locations under the reported path.

Output

GET /bank/queryxpath.jsp HTTP/1.1
Host: altoromutual.com
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D; AltoroAccounts=ODAwMDAwfkNvcnBvcmF0ZX4xLjIxNTk0MDIwMzk0NjA5OTIFMTF8ODAwMDAxfkNoZWNraW5nfi0xLjIxNTQ0MTc5Njc3NTZFMTF8
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 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://altoromutual.com/bank/doTransfer
Accept-Encoding: gzip, deflate

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

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 5822
Date: Thu, 03 Nov 2022 10:41:57 GMT
Connection: close

Remediation

To effectively prevent framing attacks, the application should return a response header with the name X-Frame-Options and the value DENY to prevent framing altogether, or the value SAMEORIGIN to allow framing only by pages on the same origin as the response itself. Note that the SAMEORIGIN header can be partially bypassed if the application itself can be made to frame untrusted websites.

Affected paths

/bank/showAccount
/login.jsp
/index.jsp
/swagger/index.html
/bank/transfer.jsp
/bank/main.jsp
/search.jsp
/feedback.jsp

References

https://portswigger.net/web-security/clickjacking
https://developer.mozilla.org/en-US/docs/Web/HTTP/X-Frame-Options
https://cwe.mitre.org/data/definitions/693.html
https://capec.mitre.org/data/definitions/103.html

Vulnerability	Count	Severity
DOM data manipulation (DOM-based)	2	Information

Description

DOM-based vulnerabilities arise when a client-side script reads data from a controllable part of the DOM (for example, the URL) and processes this data in an unsafe way.

DOM data manipulation arises when a script writes controllable data to a field within the DOM that is used within the visible UI or client-side application logic. An attacker may be able to use the vulnerability to construct a URL that, if visited by another application user, will modify the appearance or behavior of the client-side UI. An attacker may be able to leverage this to perform virtual defacement of the application, or possibly to induce the user to perform unintended actions.

Burp Suite automatically identifies this issue using dynamic and static code analysis. Static analysis can lead to false positives that are not actually exploitable. If Burp Scanner has not provided any evidence resulting from dynamic analysis, you should review the relevant code and execution paths to determine whether this vulnerability is indeed present, or whether mitigations are in place that would prevent exploitation.
The application may be vulnerable to DOM-based DOM data manipulation. Data is read from location.hash and passed to history.pushState.

Output

GET /swagger/index.html HTTP/1.1
Host: altoromutual.com
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 Safari/537.36
Connection: close
Cache-Control: max-age=0

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Accept-Ranges: bytes
ETag: W/"1427-1548795420000"
Last-Modified: Tue, 29 Jan 2019 20:57:00 GMT

Content-Type: text/html
Content-Length: 1427
Date: Thu, 03 Nov 2022 10:42:39 GMT
Connection: close

Remediation

The most effective way to avoid DOM-based DOM data manipulation vulnerabilities is not to dynamically write to DOM data fields any data that originated from any untrusted source. If the desired functionality of the application means that this behavior is unavoidable, then defenses must be implemented within the client-side code to prevent malicious data from being stored. In general, this is best achieved by using a whitelist of permitted values.

Affected paths

/swagger/index.html
/

References

https://portswigger.net/web-security/dom-based/dom-data-manipulation
https://cwe.mitre.org/data/definitions/20.html
https://capec.mitre.org/data/definitions/153.html

Vulnerability	Count	Severity
Base64-encoded data in parameter	10	Information

Description

Applications sometimes Base64-encode parameters in an attempt to obfuscate them from users or facilitate transport of binary data. The presence of Base64-encoded data may indicate security-sensitive information or functionality that is worthy of further investigation. The data should be reviewed to determine whether it contains any interesting information, or provides any additional entry points for malicious input.
The following parameter appears to contain Base64-encoded data: AltoroAccounts = 800000~Corporate~1.2159402039460999E11|800001~Checking~-1.2154417967756E11| This issue was found in multiple locations under the reported path.

Output

GET /bank/customize.jsp HTTP/1.1
Host: altoromutual.com
Cookie: JSESSIONID=4A04E1BF4690DA39EB73F7FDAC30E79D; AltoroAccounts=ODAwMDAwfkNvcnBvcmF0ZX4xLjIxNTk0MDIwMzk0NjA5OTlFMTF8ODAwMDAxfkNoZWNraW5nfi0xLjIxNTQ0MTc5Njc3NTZFMTF8
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 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://altoromutual.com/bank/queryxpath.jsp
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 5778
Date: Thu, 03 Nov 2022 10:41:57 GMT
Connection: close

Remediation

The most effective way to avoid DOM-based DOM data manipulation vulnerabilities is not to dynamically write to DOM data fields any data that originated from any untrusted source. If the desired functionality of the application means that this behavior is unavoidable, then defenses must be implemented within the client-side code to prevent malicious data

from being stored. In general, this is best achieved by using a whitelist of permitted values.

Affected paths

/bank/customize.jsp
/bank/showAccount
/swagger/index.html
/bank/doTransfer
/bank/main.jsp
/admin/admin.jsp
/bank/queryxpath.jsp
/bank/transfer.jsp

References

https://portswigger.net/web-security/dom-based/dom-data-manipulation
https://cwe.mitre.org/data/definitions/310.html
https://cwe.mitre.org/data/definitions/311.html
https://capec.mitre.org/data/definitions/37.html

Vulnerability	Count	Severity
Path-relative style sheet import	1	Information

Description

Path-relative style sheet import vulnerabilities arise when the following conditions hold:

A response contains a style sheet import that uses a path-relative URL (for example, the page at "/original-path/file.php" might import "styles/main.css").
When handling requests, the application or platform tolerates superfluous path-like data following the original filename in the URL (for example, "/original-path/file.php/extra-junk/"). When superfluous data is added to the original URL, the application's response still contains a path-relative stylesheet import.
The response in condition 2 can be made to render in a browser's quirks mode, either because it has a missing or old doctype directive, or because it allows itself to be framed by a page under an attacker's control.
When a browser requests the style sheet that is imported in the response from the modified URL (using the URL "/original-path/file.php/extra-junk/styles/main.css"), the application returns something other than the CSS response that was supposed to be imported. Given the behavior described in condition 2, this will typically be the same response that was originally returned in condition 1.
An attacker has a means of manipulating some text within the response in condition 4, for example because the application stores and displays some past input, or echoes some text within the current URL.

Given the above conditions, an attacker can execute CSS injection within the browser of the target user. The attacker can construct a URL that causes the victim's browser to import as CSS a different URL than normal, containing text that the attacker can manipulate.

Being able to inject arbitrary CSS into the victim's browser may enable various attacks, including:

Executing arbitrary JavaScript using IE's expression() function.
Using CSS selectors to read parts of the HTML source, which may include sensitive data such as anti-CSRF tokens.
Capturing any sensitive data within the URL query string by making a further style sheet import to a URL on the attacker's domain, and monitoring the incoming Referer header.
The application may be vulnerable to path-relative style sheet import (PRSSI) attacks. The response contains a path-relative style sheet import, and so condition 1 for an exploitable vulnerability is present (see issue background). The response can also be made to render in a browser's quirks mode. The page does not contain a doctype directive, and so it will always be rendered in quirks mode. Further, the response does not prevent itself from being framed, so an attacker can frame the response within a page that they control, to force it to be rendered in quirks mode. (Note that this technique is IE-specific and due to P3P restrictions might sometimes limit the impact of a successful attack.) This means that condition 3 for an exploitable vulnerability is probably present if condition 2 is present.

Burp was not able to confirm that the other conditions hold, and you should manually investigate this issue to confirm whether they do hold.

Output

```
GET /swagger/index.html HTTP/1.1
Host: altoromutual.com
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63 Safari/537.36
Connection: close
Cache-Control: max-age=0

HTTP/1.1 200 OK
```

Server: Apache-Coyote/1.1
Accept-Ranges: bytes
ETag: W/"1427-1548795420000"
Last-Modified: Tue, 29 Jan 2019 20:57:00 GMT
Content-Type: text/html
Content-Length: 1427
Date: Thu, 03 Nov 2022 10:42:39 GMT
Connection: close

Remediation

The root cause of the vulnerability can be resolved by not using path-relative URLs in style sheet imports. Aside from this, attacks can also be prevented by implementing all of the following defensive measures:
Setting the HTTP response header "X-Frame-Options: deny" in all responses. One method that an attacker can use to make a page render in quirks mode is to frame it within their own page that is rendered in quirks mode. Setting this header prevents the page from being framed.
Setting a modern doctype (e.g. "<!doctype html>") in all HTML responses. This prevents the page from being rendered in quirks mode (unless it is being framed, as described above).
Setting the HTTP response header "X-Content-Type-Options: nosniff" in all responses. This prevents the browser from processing a non-CSS response as CSS, even if another page loads the response via a style sheet import.

Affected paths

/bank/customize.jsp
/bank/showAccount
/swagger/index.html
/bank/doTransfer
/bank/main.jsp
/admin/admin.jsp
/bank/queryxpath.jsp
/bank/transfer.jsp

References

https://portswigger.net/research/detecting-and-exploiting-path-relative-stylesheet-import-prssi-vulnerabilities
https://cwe.mitre.org/data/definitions/16.html
https://capec.mitre.org/data/definitions/154.html
https://capec.mitre.org/data/definitions/468.html

4.0 ASSESSMENT METHODOLOGY

Network Footprinting (Reconnaissance): Gather as much information as possible about the selected network. Reconnaissance can take two forms i.e. active and passive. A passive attack is always the best starting point as this would normally defeat intrusion detection systems and other forms of protection etc. afforded to the network. This would usually involve trying to discover publicly available information by utilizing a web browser and visiting newsgroups etc. An active form would be more intrusive and may show up in audit logs and may take the form of an attempted DNS zone transfer or a social engineering type of attack.

Discovery & Probing: Enumeration can serve two distinct purposes in an assessment: OS Fingerprinting Remote applications being served. OS fingerprinting or TCP/IP stack fingerprinting is the process of determining the operating system being utilized on a remote host. This is carried out by analyzing packets received from the host in question. There are two distinct ways to OS fingerprint, actively (i.e. nmap) or passively (i.e. scanrand). Passive OS fingerprinting determines the remote OS utilizing the packets received only and does not require any packets to be sent. Active OS fingerprinting is very noisy and requires packets to be sent to the remote host and waits for a reply, (or lack thereof). Disparate OSs respond differently to certain types of packet, (the response is governed by an RFC and any proprietary responses the vendor (notably Microsoft) has enabled within the system) and so custom packets may be sent. Remote applications being served on a host can be determined by an open port on that host. By port scanning it is then possible to build up a picture of what applications are running and tailor the test accordingly.

Vulnerability Assessment: Utilizing vulnerability scanners all discovered hosts can then be tested for vulnerabilities. The result would then be analyzed to determine if there any vulnerabilities that could be exploited to gain access to a target host on a network. A number of tests carried out by these scanners are just banner grabbing/ obtaining version information, once these details are known, the version is compared with any common vulnerabilities and exploits (CVE) that have been released and reported to the user. Other tools actually use manual pen testing methods and display the output received i.e. showmount -e ip_address would display the NFS shares available to the scanner which would then need to be verified by the tester.

Forensic Assessment: Delivers only relevant vulnerabilities and those with exploits or having exploitable conditions typically attempted during a penetration test. Additionally, this assessment derives a unique threat model from the assessment data to discover possible misuse, potentially unwanted programs and backdoors in the environment. Essential for post breach assessments, discovery of protocol misuse, blacklisted IPs and those on blocklists by malware, suspicious domains, safe browser checks, DGA typically used by malware, unusual traffic, geographic irregularities, remote access misuse, and various other telemetry are provided to discover indicators of compromise typically not found in traditional vulnerability assessment or penetration test.

NVD is the U.S. government repository of standards based vulnerability management data represented using the Security Content Automation Protocol (SCAP). This data enables automation of vulnerability management, security measurement, and compliance. NVD includes databases of security checklists, security related software flaws, misconfigurations, product names, and impact metrics.

CVE is a dictionary of publicly known information security vulnerabilities and exposures. CVEs common identifiers enable data exchange between security products and provide a baseline index point for evaluating coverage of tools and services.

OWASP: The Open Web Application Security Project (OWASP) is a 501(c)(3) worldwide not-for-profit charitable organization focused on improving the security of software. https://www.owasp.org/index.php/Top_10-2017_Top_10

PTES: Defines baseline methods that have been used in the industry covering everything related to a penetration test - from the initial communication and reasoning behind a pentest to reporting.

4/Critical
Description
A vulnerability whose exploitation could allow code execution without user interaction. These scenarios include self-propagating malware (e.g. network worms), or unavoidable common use scenarios where code execution occurs without warnings or prompts. This could mean browsing to a web page or opening email. Apply Critical updates immediately.
3/High
Description
XSS and other client-side attacks, extremely significant information disclosure (e.g. administrator hashes) Significant information disclosure, PII, user passwords. Valuable information disclosure (e.g. DNS zone transfers, account enumeration)
2/Medium
Description
A vulnerability which is known to lead to the compromise of the application or system being tested, however a mitigating factor is present. A vulnerability which may lead to compromise of the application or system being tested, however several mitigating factors are present (e.g. network configuration prevents talking directly to the required port) or exploitation is possible but only theoretical.
1/Low
Description
Other application or system specific information disclosure which does not fit (e.g. development notes, configuration files) Minor information disclosure, OS, patch levels (if current), etc. Non exploitable vulnerabilities which should still be addressed or low value information disclosure.