ClientX

Vulnerability Assessment Report

CloudPro™ Services Corporation

Table of Contents

1. Executive Summary

ClientX engaged CloudPro to perform vulnerability scan of ClientX application. ClientX is a CRM application tailored to construction business. It serves as a communication/marketing hub between construction customers and ClientX.

The assessment was performed in accordance with the "best-in-class" practices as defined by ISECOM's Open Source Security Testing Methodology Manual (OSSTMM) and the Open Web Application Security Project (OWASP). During the assessment, CloudPro applied its advanced tools, comprehensive application security knowledge base, and security vulnerability assessment methodology for the detection of security issues and exposures within the ClientX code base and the related runtime platform environment.

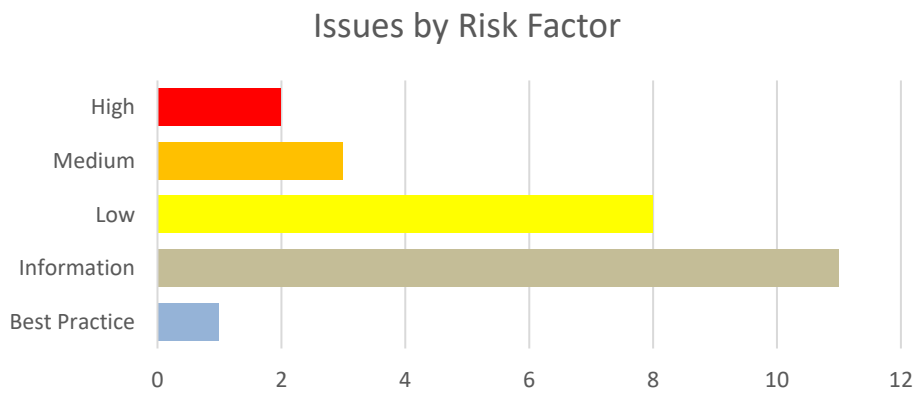
CloudPro conducted the vulnerability assessment during the period between February 22, 2020 and February 29, 2020. Gray-box security testing was performed for the ClientX application on its staging server using authentication credentials provided by ClientX. Limited scope black-box security testing was performed for the ClientX application running on its production server.

The application was accessible at the following URL:

- *Production*: http://clientx.app/
- *QA*: https://webqa.clientx.net/

During the course of this assessment, CloudPro was able to identify several security vulnerabilities. This report summarizes what CloudPro believes are the most important issues to address in the tested application. The table below shows the numbers of issues identified in different categories. Issues are classified according to severity as High, Medium, Low or Information. This reflects the likely impact of each issue for a typical organization. Issues are also classified according to confidence as Certain, Firm or Tentative. This reflects the inherent reliability of the technique that was used to identify the issue.

	Certain	Firm	Tentative	Total
High	2			2
Medium	1	1	1	3
Low	7	1		8
Information	8	3		11
Best Practice	1			1



CloudPro recommends that the findings contained in this report should be addressed to minimize the attack surface available to an attacker and to ensure the overall security of their applications.

Criteria for Risk Ratings

The table below outlines the general rules for assigning risk ratings to identified vulnerabilities: Risk Rating Description.

Risk Rating	Description
HIGH	**Fix immediately/Soon:** These issues identify conditions that could directly result in the compromise or unauthorized access of a network, system, application or sensitive information. Examples of High-Risk issues include remote execution of commands, known buffer overflows; unauthorized access and disclosure of sensitive information.
MEDIUM	**Fix Soon:** These issues identify conditions that do not immediately or directly result in the compromise or unauthorized access of a network, system, application of information, but do provide a capability or information that could, in combination with other capabilities or information, result in the compromise or unauthorized access of a network, application or information. Examples of Medium Risk issues include directory browsing, partial access to files on the system; disclosure of security mechanisms and unauthorized use of services.
LOW	**Consider fixing:** These issues identify conditions that do not immediately or directly result in compromise of a network, system, application or information, but do provide information that could be used in combination with other information to gain insight into how to compromise or gain unauthorized access to a network, system, application or information.
INFORMATION	We do not even call these alerts vulnerabilities. They are reported simply for your information as a website owner, as they may not have a direct impact but could help an attacked to gain a better understanding of your underlying systems.
BEST PRACTICE	The Best Practice Severity Level is for detected issues that are recommended practices but are not vulnerabilities and so are not as serious as the preceding severity levels. Depending on the Best Practice suggestions, impacts might include damage to the privacy of the user, detraction from a site's extra layers of security, or failure to meet with industry standards.

Criteria for Confidence

Confidence	Description
Certain	90% of the time it is a real flaw
Firm	60% it's not a false positive
Tentative	is probably false positive

2. Assessment Methodology

CloudPro based the finding and recommendations, outlined in this report, on network and application vulnerability scans and manual penetration testing performed against the application.

Automated Application Scan

CloudPro used several tools to survey the targeted environment and identify potential vulnerabilities. The automated scanning software identifies application-level vulnerabilities. The scope of testing includes but not limited by the following:

- Parameter Injection
- SQL Injection
- Cross-Site Scripting
- XML External Entities
- Broken Access Control
- Directory Traversal
- Parameter Overflow
- Buffer Overflow
- Parameter Addition
- Path Manipulation
- Character Encoding
- SSL Strength
- Sensitive Data Exposure
- Permissions Assessment
- Brute Force Authentication attacks
- Security Misconfiguration
- Insecure Deserialization
- Using Components with Known Vulnerabilities

3. Assessment Findings:

3.1. Summary

The following table provides a summary of the issues identified by the vulnerability assessment.

Clicking on a link will take you to the details. Within the details, text highlighted in yellow provides proof of the vulnerability if available.

Vulnerabilities, Information and Best Practices	Severity
Cross Site Scripting Weakness (Persistent in JSON Response)	High
Cross-origin resource sharing: arbitrary origin trusted	High

Missing X-Frame-Options Header	Medium
Cross-origin resource sharing	Medium
XML injection	Medium
Strict transport security not enforced	Low
SSL cookie without secure flag set	Low
TLS V1.0 Supported	`
Content type incorrectly stated	Low
Internal Server Error	Low
Server Leaks Information via "X-Powered-By" HTTP Response Header Field	Low
X-Content-Type-Options Header Missing	Low
Cross Site Scripting Weakness (Reflected in JSON Response)	Low
Cross-domain script include	Information
Frameable response (potential Clickjacking)	Information
Cacheable HTTPS response	Information
OPTIONS Method Enabled	Information
Input returned in response (reflected)	Information
Cross-domain Referer leakage	Information
Backup file	Information
Email addresses disclosed	Information
SSL certificate	Information
Sensitive Information Disclosure in URL	Information
Sensitive Information Disclosure in HTTP Referrer Header	Information
SameSite Cookie Not Implemented	Best Practice

3.2. High Risk Finds

3.2.1. Cross Site Scripting Weakness (Persistent in JSON Response)

Risk Rating: HIGH

Summary

Severity	**High**
Confidence	**Confirm**
Host	**https://api-qa.clientx.net/**
Classification	**CWE-79, WASC-8**

Issue Background

An XSS attack was found in a JSON response, this might leave content consumers vulnerable to attack if they don't appropriately handle the data (response).

In stored XSS, an attacker directly submits code to a web application via a user input field, and that script is stored on the web page. It is run whenever a user loads the site in their web browser, with the normal privileges of that website.

Impact

If an attacker can control a script that is executed in the victim's browser, then they can typically fully compromise that user. The attacker can carry out any of the actions that are applicable to the impact of reflected XSS vulnerabilities.

In terms of exploitability, the key difference between reflected and stored XSS is that a stored XSS vulnerability enables attacks that are self-contained within the application itself. The attacker does not need to find an external way of inducing other users to make a particular request containing their exploit. Rather, the attacker places their exploit into the application itself and simply waits for users to encounter it.

The self-contained nature of stored cross-site scripting exploits is particularly relevant in situations where an XSS vulnerability only affects users who are currently logged in to the application. If the XSS is reflected, then the attack must be fortuitously timed: a user who is induced to make the attacker's request at a time when they are not logged in will not be compromised. In contrast, if the XSS is stored, then the user is guaranteed to be logged in at the time they encounter the exploit.

Issue Remediation

Phase: Architecture and Design

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket.

Phases: Implementation; Architecture and Design

Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies.

For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters.

Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed.

Phase: Architecture and Design

For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server.

If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated.

Phase: Implementation

For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping.

To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that

support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set.

Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a blacklist). However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.

When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue."

Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.

References

- Cross Site Scripting
- Stored XSS (or Persistent)

Vulnerability Classification

- CWE-79
- WASC-8

Request

```
GET https://api-
qa.clientx.net/api/locations/5c499121d41d39001102a0af?orgId=5c498d5ed41d39001102a0ae HTTP/1.1
Connection: keep-alive
Pragma: no-cache
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/80.0.3987.122 Safari/537.36
Cahce-Control: no-cache
Accept: application/json, text/plain, */*
Sec-Fetch-Dest: empty
Content-Type: application/json
x-access-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVlNThhNThjYjQzY2U3MDAxMmU3YmY4MSIsInR5cGUiOiJsb2
MtYWRtaW4iLCJsb2NhdGlvbnMiOlsiNWM0OTkxMjFkNDFkMzkwMDExMDJhMGFmIiwiNWM0OTkyYmNkNDFkMzkwMDExMDJhM
GI5IiwiNWQ4YjJkNzRhZDYxMzkwMDExMmIwZjI4IiwiNWQ5MWNkZmE1ODI3MTMwMDExMjZiZDVkIl0sIm9yZ0lkIjoiNWM0
OThkNWVkNDFkMzkwMDExMDJhMGFlIiwiZW1haWwiOiJheWVzaGEuc2FnaGVlckBubXhnbG9iYWxzb2Z0d2FyZS5jb20iLCJ
uYW1lIjoiQXllc2hhIiwibm90aWZ5RW1haWwiOnRydWUsIm5vdGlmeVNNUyI6ZmFsc2UsInBob25lIjoiMjM0NTY3OCIsIn
ByZWZzIjpbeyJrZXkiOiJtZWFzdXJlbWVudCIsInZhbHVlIjoiaW1wZXJpYWwifSx7ImtleSI6Imxhc3RBbm5vdW5jZW1lb
nRSZWFkRGF0ZSIsInZhbHVlIjoiMjAyMC0wMi0yOVQwNDo1Mjo1My41NDJaIn1dLCJsYW5ndWFnZSI6ImVuLVVTIiwiZGF0
ZUZvcm1hdCI6Ik1NL2RkL3l5eXkiLCJmaXJzdFRpbWVMb2dpbiI6ZmFsc2UsImNyZWF0ZWRBdCI6IjIwMjAtMDItMjhUMDU
6MzA6NTIuOTY2WiIsImlzUmVtZW1iZXIiOnRydWUsImlhdCI6MTU4Mjk2NDA0NiwiZXhwIjoxNTkwNzQwMDQ2fQ.XDAb9qJ
tSv_pZh2AqUTroaWgx88e0vnzRs9CkWHMUoc
Origin: https://webqa.clientx.net
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Referer: https://webqa.clientx.net/locations
Accept-Language: en-US,en;q=0.9
Content-Length: 0
Host: api-qa.clientx.net
Cookie: connect.sid=s%3AQBHr4jaOzW3fB4ML6Z2siw-
3Ygy2yrlw.4uSqnm8wJURl0Q9btWTfAaoPNWbTx99b07%2BFl5Ss6zM
```

Response

```
HTTP/1.1 200 OK
Date: Sat, 29 Feb 2020 08:58:28 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 157441
Connection: keep-alive
X-Powered-By: Express
Access-Control-Allow-Origin: *
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
Access-Control-Expose-Headers: refresh-token
ETag: W/"26701-fcKMVdC2a0RmSYJh5R53dViMVfU"
Vary: Accept-Encoding

{"status":200,"messageCode":"GET_LOCATION_SUCCESS","message":"Get                    location
success","data":{"geo":{"coordinates":[36.5661303,-
89.21588629999997],"type":"Point","timeZoneId":"America/Chicago"},"storageUnits":[{"notify":["5
e58a58cb43ce70012e7bf81","5e58a519b43ce70012e7bf13"],"devices":[],"shouldRunHotSpot":true,"hots
potlevel":0,"_id":"5e5a2637d5c68c001254d199","name":"\"><!--#EXEC         cmd=\"ls          /\"--
><","type":"bunker_steel","cableType":"none","moistureType":"emc","thresholds":[{"enabled":true
,"_id":"5e5a2637d5c68c001254d19a","sensorType":"temp","high":66.11111111111111,"low":-
20,"createdBy":"security.engineer@cloudpro.services","createdAt":"2020-02-
29T08:52:07.183Z","updatedAt":"2020-02-
29T08:52:07.183Z"},{"enabled":false,"_id":"5e5a2637d5c68c001254d19b","sensorType":"hum","high":
100,"low":0,"createdBy":"security.engineer@cloudpro.services","createdAt":"2020-02-
29T08:52:07.183Z","updatedAt":"2020-02-
29T08:52:07.183Z"},{"enabled":false,"_id":"5e5a2637d5c68c001254d19c","sensorType":"emc","high":
25,"low":5,"createdBy":"security.engineer@cloudpro.services","createdAt":"2020-02-
29T08:52:07.183Z","updatedAt":"2020-02-
29T08:52:07.183Z"}],"tags":[{"key":"1","value":"test"}],"locId":"5c499121d41d39001102a0af","cre
atedBy":"security.engineer@cloudpro.services","orgId":"5c498d5ed41d39001102a0ae","productType":
"canola","cables":[],"summaries":[],"levelSummaries":[],"updatedAt":"2020-02-
29T08:52:07.183Z","__v":0},{"notify":["5e58a58cb43ce70012e7bf81","5e58a519b43ce70012e7bf13"],"d
evices":[],"shouldRunHotSpot":true,"hotspotlevel":0,"_id":"5e5a26f6d5c68c001254d357","name":"'\
"<script>alert(1);</script>","type":"bunker_steel","cableType":"none","moistureType":"emc","thr
esholds":[{"enabled":true,"_id":"5e5a26f6d5c68c001254d358","sensorType":"temp","high":66.111111
11111111,"low":-20,"createdBy":"security.engineer@cloudpro.services","createdAt":"2020-02-
29T08:55:18.700Z","updatedAt":"2020-02-
…
```

3.2.2. Cross-origin resource sharing: arbitrary origin trusted

Risk Rating: HIGH

Summary

Severity	**High**
Confidence	**Certain**
Host	**https://api-qa.clientx.net**
Classification	**CWE-942: Overly Permissive Cross-domain Whitelist**

Issue Background

There are 6 instances of this issue:

- /api
- /api/alert
- /api/alert/user
- /api/announcement
- /api/login
- /api/logout

An HTML5 cross-origin resource sharing (CORS) policy controls whether and how content running on other domains can perform two-way interaction with the domain that publishes the policy. The policy is fine-grained and can apply access controls per-request based on the URL and other features of the request.

Trusting arbitrary origins effectively disables the same-origin policy, allowing two-way interaction by third-party web sites. Unless the response consists only of unprotected public content, this policy is likely to present a security risk.

If the site specifies the header Access-Control-Allow-Credentials: true, third-party sites may be able to carry out privileged actions and retrieve sensitive information. Even if it does not, attackers may be able to bypass any IP-based access controls by proxying through users' browsers

Impact

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin https://vxyemweglirr.com

Since the Vary: Origin header was not present in the response, reverse proxies and intermediate servers may cache it. This may enable an attacker to carry out cache poisoning attacks.

Issue Remediation

Rather than using a wildcard or programmatically verifying supplied origins, use a whitelist of trusted domains.

References

- Exploiting CORS Misconfigurations

Vulnerability Classification

- CWE-942: Overly Permissive Cross-domain Whitelist

Request

```
GET / HTTP/1.1
Host: api-qa.clientx.net
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;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/69.0.3497.100 Safari/537.36
Connection: close
Cache-Control: max-age=0
Referer: https://webqa.clientx.net/main.2e77111944f4d56a67b8.js
Origin: https://vxyemweglirr.com
```

Response

```
HTTP/1.1 401 Unauthorized
Date: Sat, 29 Feb 2020 05:38:00 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 74
Connection: close
X-Powered-By: Express
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: https://vxyemweglirr.com
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
ETag: W/"4a-Php9aKHtjyxv6ihqICJ/7fT+upY"
Set-Cookie:
connect.sid=s%3ArPErYXACObKDLfVNWdHSEfgTjQ5z2kR6.3ykLW0Y%2BzthoJpGqB0FYJoD6HID0VTrpKQXaCVu9140;
Path=/; HttpOnly
Vary: Accept-Encoding

{"status":401,"message":"Unauthenticated","messageCode":"UNAUTHENTICATED"}
```

3.3. Medium Risk Finding

3.3.1. Missing X-Frame-Options Header

Risk Rating: MEDIUM

Summary

Severity	**Medium**
Confidence	**Firm**
Host	**https://webqa.clientx.net**
Classification	**OWASP 2013-A5 OWASP 2017-A6 CWE-693 CAPEC-103**

Issue Background

CloudPro detected a missing X-Frame-Options header in the HTTP which means that this website could be at risk of a clickjacking attack.

The X-Frame-Options HTTP header field indicates a policy that specifies whether the browser should render the transmitted resource within a frame or an iframe. Servers can declare this policy in the header of their HTTP responses to prevent clickjacking attacks, which ensures that their content is not embedded into other pages or frames.

Impact

Clickjacking is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on a framed page when they were intending to click on the top-level page. Thus, the attacker is "hijacking" clicks meant for their page and routing them to other another page, most likely owned by another application, domain, or both.

Using a similar technique, keystrokes can also be hijacked. With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker.

Issue Remediation

- Sending the proper X-Frame-Options in HTTP response headers that instruct the browser to not allow framing from other domains.
    - ***X-Frame-Options: DENY*** It completely denies to be loaded in frame/iframe.
    - ***X-Frame-Options: SAMEORIGIN*** It allows only if the site which wants to load has a same origin.
    - ***X-Frame-Options: ALLOW-FROM*** *URL* It grants a specific URL to load itself in a iframe. However please pay attention to that, not all browsers support this.
- Employing defensive code in the UI to ensure that the current frame is the most top level window.

References

- Clickjacking
- Can I Use X-Frame-Options
- X-Frame-Options HTTP Header
- Clickjacking Defense Cheat Sheet

Vulnerability Classification

- OWASP 2013-A5

- OWASP 2017-A6
- CWE-693
- CAPEC-103

Request

```
GET https://webqa.clientx.net/ HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:71.0) Gecko/20100101 Firefox/71.0
Pragma: no-cache
Cache-Control: no-cache
Content-Length: 0
Host: webqa.clientx.net
```

Response

```
HTTP/1.1 200 OK
Date: Sat, 29 Feb 2020 06:18:43 GMT
Content-Type: text/html
Content-Length: 3358
Connection: keep-alive
Server: Apache/2.4.41 (Unix)
Last-Modified: Fri, 28 Feb 2020 07:28:20 GMT
ETag: "d1e-59f9dc8236d00"
Accept-Ranges: bytes
Cache-Control: max-age=0, no-cache, no-store, must-revalidate
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Vary: Accept-Encoding
Pragma: no-cache
```

3.3.2. Cross-origin resource sharing

Risk Rating: **MEDIUM**

Summary

Severity	**Medium**
Confidence	**Certain**
Host	**https://api-qa.clientx.net**
Classification	**CWE-942: Overly Permissive Cross-domain Whitelist**

Issue Background

There are 18 instances of this issue are identified on QA:

- /
- /api
- /api/alert
- /api/alert/count
- /api/announcements/count
- /api/invites
- /api/locations
- /api/locations/barge
- /api/locations/units
- /api/login
- /api/logout
- /api/storage/summary-balls

- /api/storage/summary-cable-detail
- /api/storage/summary-cables
- /api/storage/summary-spears
- /api/storage/temp/hum
- /api/usersagent
- /api/usersagent/
- /robots.txt

An HTML5 cross-origin resource sharing (CORS) policy controls whether and how content running on other domains can perform two-way interaction with the domain that publishes the policy. The policy is fine-grained and can apply access controls per-request based on the URL and other features of the request.

If another domain is allowed by the policy, then that domain can potentially attack users of the application. If a user is logged in to the application, and visits a domain allowed by the policy, then any malicious content running on that domain can potentially retrieve content from the application, and sometimes carry out actions within the security context of the logged in user.

Even if an allowed domain is not overtly malicious in itself, security vulnerabilities within that domain could potentially be leveraged by an attacker to exploit the trust relationship and attack the application that allows access. CORS policies on pages containing sensitive information should be reviewed to determine whether it is appropriate for the application to trust both the intentions and security posture of any domains granted access.

Impact

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request.

As, Origin header was not present in the response, reverse proxies and intermediate servers may cache it. This may enable an attacker to carry out cache poisoning attacks.

Issue Remediation

Any inappropriate domains should be removed from the CORS policy.

References

- Exploiting CORS Misconfigurations

Vulnerability Classification

- CWE-942: Overly Permissive Cross-domain Whitelist

Request

```
GET / HTTP/1.1
Host: api-qa.clientx.net
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;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/69.0.3497.100 Safari/537.36
Connection: close
Cache-Control: max-age=0
Referer: https://webqa.clientx.net/main.2e77111944f4d56a67b8.js
Origin: https://api-qa.clientx.net
```

Response

```
HTTP/1.1 401 Unauthorized
Date: Sat, 29 Feb 2020 05:37:56 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 74
Connection: close
X-Powered-By: Express
Access-Control-Allow-Origin: *
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
```

```
ETag: W/"4a-Php9aKHtjyxv6ihqICJ/7fT+upY"
Set-Cookie: connect.sid=s%3AOxs9PpbRIb9bU0V4g68FsHetzU33g-
n8.vrmvGXK32qSIlvPIBTPjzpS2x7GpnqEAkD4KSiQiF6o; Path=/; HttpOnly
Vary: Accept-Encoding

{"status":401,"message":"Unauthenticated","messageCode":"UNAUTHENTICATED"}
```

3.3.3. XML injection

Risk Rating: **MEDIUM**

Summary

Severity	**Medium**
Confidence	**Tentative**
Host	**https://api-qa.clientx.net**
Classification	**CWE-91, CWE-116, CWE-159, CWE-611, CWE-776**

Issue Background

The onlyActive parameter appears to be vulnerable to XML injection. The payload]] >>< was appended to the value of the onlyActive parameter. The application's response indicated that this input may have caused an error within a server-side XML or SOAP parser, suggesting that the input has been inserted into an XML document or SOAP message without proper sanitization.

Impact

XML or SOAP injection vulnerabilities arise when user input is inserted into a server-side XML document or SOAP message in an unsafe way. It may be possible to use XML metacharacters to modify the structure of the resulting XML. Depending on the function in which the XML is used, it may be possible to interfere with the application's logic, to perform unauthorized actions or access sensitive data.

This kind of vulnerability can be difficult to detect and exploit remotely; you should review the application's response, and the purpose that the relevant input performs within the application's functionality, to determine whether it is indeed vulnerable.

Issue Remediation

The application should validate or sanitize user input before incorporating it into an XML document or SOAP message. It may be possible to block any input containing XML metacharacters such as < and >. Alternatively, these characters can be replaced with the corresponding entities: < and >

References

- XML Injection - WS-Attacks
- What is XML Injection Attack

Vulnerability Classification

- CWE-91: XML Injection (aka Blind XPath Injection)
- CWE-116: Improper Encoding or Escaping of Output
- CWE-159: Failure to Sanitize Special Element
- CWE-611: Improper Restriction of XML External Entity Reference ('XXE')
- CWE-776: Improper Restriction of Recursive Entity References in DTDs ('XML Entity Expansion')

Request

```
GET
/api/users?size=0&page=0&sort=name&order=asc&search=&onlyActive=true]]%3e%3e%3c&forBox=true&org
Id=5c498d5ed41d39001102a0ae HTTP/1.1
Host: api-qa.clientx.net
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cahce-Control: no-cache
Pragma: no-cache
Content-Type: application/json
x-access-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVlNThhNTE5YjQzY2U3MDAxMmU3YmYxMyIsInR5cGUiOiJvcm
ctYWRtaW4iLCJsb2NhdGlvbnMiOltdLCJvcmdJZCI6IjVjNDk4ZDVlZDQxZDM5MDAxMTAyYTBhZSIsImVtYWlsIjoiZWh0Y
XNoYW11bC5oYXFAbm14Z2xvYmFsc29mdHdhcmUuY29tIiwibmFtZSI6IkVodGFzaGFtIFVsIEhhcSIsIm5vdGlmeUVtYWls
Ijp0cnVlLCJub3RpZnlTTVMiOnRydWUsInBob25lIjoiKzkyIDMyMSA1MjUyMzQ1IiwicHJlZnMiOlt7ImtleSI6Im1lYXN
1cmVtZW50IiwidmFsdWUiOiJtZXRyaWMifSx7ImtleSI6Im1hcFZpZXciLCJ2YWx1ZSI6Imxpc3QifSx7ImtleSI6InNob3
dPbmx5V29ya2luZ0JhbGxzIiwidmFsdWUiOiJmYWxzZSJ9LHsiaJV5Ijoic2hvd09ubHlXb3JraW5nQ2FibGVzIiwidmFsd
WUiOiJmYWxzZSJ9LHsiaJV5Ijoic2hvd09ubHlXb3JraW5nU3BlYXJzIiwidmFsdWUiOiJmYWxzZSJ9LHsiaJV5IjoibGFz
dEFubm91bmNlbWVudFJlYWREYXRlIiwidmFsdWUiOiIyMDIwLTAyLTI5VDA4OjEwOjA2LjY1M1oifV0sImxhbmd1YWdlIjo
iZW4tVVMiLCJkYXRlRm9ybWF0IjoiTU0vZGQveXl5eSIsImZpcnN0VGltZUxvZ2luIjpmYWxzZSwiY3JlYXRlZEF0IjoiMj
AyMC0wMi0yOFQwNToyODo1Ny42MjRaIiwiaXNSZW1lbWJlciI6dHJ1ZSwiaWF0IjoxNTgyOTY2OTEzLCJleHAiOjE1OTA3N
DI5MTN9.CEvnNEg1vrEMtzlNRVNpuAI_YwjQGcuB9w52xQYL0S0
Origin: https://webqa.clientx.net
Connection: close
Referer: https://webqa.clientx.net/storage-unit/new
```

Response

```
HTTP/1.1 200 OK
Date: Sat, 29 Feb 2020 13:19:54 GMT
Content-Type: application/json; charset=utf-8
Connection: close
X-Powered-By: Express
Access-Control-Allow-Origin: *
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
Access-Control-Expose-Headers: refresh-token
ETag: W/"bb19-p0iKouPeXLtv2NXn6j/a2zJYNV4"
Vary: Accept-Encoding
Content-Length: 47897

{"status":200,"messageCode":"GET_LIST_USERS_SUCCESS","message":"Get list users
success","total":76,"data":[{"type":"org-
admin","status":"active","notes":"","locations":[],"notifyEmail":true,"dateForma
...[SNIP]...
<jey xmlns=\"http://a.b/\" xmlns:xsi=\"http://www.w3.org/2001/xmlschema-instance\"
xsi:schemalocation=\"http://a.b/
http://y10zd5gbvx6hkheicyr4ddskvb16pyfmfa7xxlm.burpcollaborator.net/jey.xsd\">
...[SNIP]...
```

3.4. Low Risk Finding

3.4.1. Strict transport security not enforced

Risk Rating: **LOW**

Summary

Severity	**Low**
Confidence	**Certain**

Host	**https://clientx.app** **https://tsng-api.clientx.net** **https://api-qa.clientx.net**
Classification	**CWE-523, OWASP 2013-A6, OWASP 2017-A3, CAPEC-217**

Issue Background

CloudPro identified that HTTP Strict Transport Security (HSTS) policy is not enabled.

There are 7 instances of this issue that were identified on production:

- https://clientx.app/login
- https://tsng-api.clientx.net/
- https://tsng-api.clientx.net/api
- https://tsng-api.clientx.net/api/forgot-password
- https://tsng-api.clientx.net/api/login
- https://tsng-api.clientx.net/favicon.ico
- https://tsng-api.clientx.net/test

There are 12 instances of this issue that were identified on QA server:

- https://api-qa.clientx.net/
- https://api-qa.clientx.net/api
- https://api-qa.clientx.net/api/alert/count
- https://api-qa.clientx.net/api/announcement
- https://api-qa.clientx.net/api/locations
- https://api-qa.clientx.net/api/login
- https://api-qa.clientx.net/api/logout
- https://api-qa.clientx.net/api/users/
- https://api-qa.clientx.net/robots.txt
- https://webqa.clientx.net/
- https://webqa.clientx.net/favicon.ico
- https://webqa.clientx.net/robots.txt

The target website is being served from not only HTTPS but also HTTP and it lacks of HSTS policy implementation.

HTTP Strict Transport Security (HSTS) is a web security policy mechanism whereby a web server declares that complying user agents (such as a web browser) are to interact with it using only secure (HTTPS) connections. The HSTS Policy is communicated by the server to the user agent via a HTTP response header field named "Strict-Transport-Security". HSTS Policy specifies a period of time during which the user agent shall access the server in only secure fashion.

When a web application issues HSTS Policy to user agents, conformant user agents behave as follows:

- Automatically turn any insecure (HTTP) links referencing the web application into secure (HTTPS) links. (For instance, http://example.com/some/page/ will be modified to https://example.com/some/page/ before accessing the server.)
- If the security of the connection cannot be ensured (e.g. the server's TLS certificate is self-signed), user agents show an error message and do not allow the user to access the web application.

Impact

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 defences 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.

Issue 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

References

- HTTP Strict Transport Security
- sslstrip
- HSTS Preload Form

Vulnerability Classification

- CWE-523: Unprotected Transport of Credentials
- OWASP 2013-A6
- OWASP 2017-A3
- CAPEC-217

3.4.1.1. https://clientx.app/login

Request

```
GET /login HTTP/1.1
Host: clientx.app
Connection: close
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/79.0.3945.130 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,applicati
on/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: _ga=GA1.2.461878407.1582423339; _gid=GA1.2.276150750.1582423339;
_gat_gtag_UA_146125476_1=1; _gat=1; G_ENABLED_IDPS=google
```

Response

```
HTTP/1.1 200 OK
Date: Sun, 23 Feb 2020 02:04:50 GMT
Content-Type: text/html
Content-Length: 3368
Connection: close
Server: Apache/2.4.41 (Unix)
Last-Modified: Wed, 22 Jan 2020 07:58:11 GMT
ETag: "d28-59cb5e2c30ac0-gzip"
Accept-Ranges: bytes
Cache-Control: max-age=0, no-cache, no-store, must-revalidate
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Vary: Accept-Encoding
Pragma: no-cache
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>Clientx</title>
<base href="/" />
<meta name="viewport" content="width=device-width, initial-scale=1
...[SNIP]...
```

3.4.1.2. https://api-qa.clientx.net/

Request

```
GET / HTTP/1.1
Host: api-qa.clientx.net
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;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/69.0.3497.100 Safari/537.36
Connection: close
Cache-Control: max-age=0
Referer: https://webqa.clientx.net/main.2e77111944f4d56a67b8.js
```

Response

```
HTTP/1.1 401 Unauthorized
Date: Sat, 29 Feb 2020 05:34:47 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 74
Connection: close
X-Powered-By: Express
Access-Control-Allow-Origin: *
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
ETag: W/"4a-Php9aKHtjyxv6ihqICJ/7fT+upY"
Set-Cookie: connect.sid=s%3Ak1PT8nCDUeX7hfcsaa-6PUG6PNq-
Y0JV.%2B5NtJGUrgvCfQyWutyjbscDb%2BeoMJaeLYZ7Y2RXLKMM; Path=/; HttpOnly
Vary: Accept-Encoding

{"status":401,"message":"Unauthenticated","messageCode":"UNAUTHENTICATED"}
```

3.4.2. SSL cookie without secure flag set

Risk Rating: **LOW**

Summary

Severity	**Low**
Confidence	**Certain**
Host	**https://tsng-api.clientx.net** **https://api-qa.clientx.net**
Classification	**OWASP 2013-A6 OWASP 2017-A3 PCI V3.2-6.5.10 CWE-614 CAPEC-102 WASC-15**

Issue Background

There are 3 instances of this issue that were identified on production:

- /
- /api/forgot-password
- /api/login

There are 7 instances of this issue:

- /
- /api
- /api/apikeys
- /api/invite
- /api/locations/undefined/units
- /api/login
- /robots.txt

CloudPro found that the following cookie was issued by the application and does not have the secure flag set:

- connect.sid

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.

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.

Impact

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 defences 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.

Issue 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.

References

- NET Cookie.Secure Property
- How to Create Totally Secure Cookies

Vulnerability Classification

- OWASP 2013-A6
- OWASP 2017-A3
- PCI V3.2-6.5.10
- CWE-614
- CAPEC-102
- WASC-15

3.4.2.1. https://tsng-api.clientx.net/

Request

```
GET / HTTP/1.1
Host: tsng-api.clientx.net
Connection: close
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/79.0.3945.130 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,applicati
on/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

Response

```
HTTP/1.1 401 Unauthorized
Date: Sun, 23 Feb 2020 02:05:02 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 74
Connection: close
X-Powered-By: Express
Access-Control-Allow-Origin: *
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
ETag: W/"4a-Php9aKHtjyxv6ihqICJ/7fT+upY"
Set-Cookie:
connect.sid=s%3Aj7mYKWLQYpk9lIq5yugITdtuLtrWt6_V.vtv8KM3fJI4nPUu5MSsth0ddeddz%2Frsuf%2BK80TqkbK
4; Path=/; HttpOnly
Vary: Accept-Encoding

{"status":401,"message":"Unauthenticated","messageCode":"UNAUTHENTICATED"}
```

3.4.2.2. https://api-qa.clientx.net/

Request

```
GET / HTTP/1.1
Host: api-qa.clientx.net
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;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/69.0.3497.100 Safari/537.36
Connection: close
Cache-Control: max-age=0
Referer: https://webqa.clientx.net/main.2e77111944f4d56a67b8.js
```

Response

```
HTTP/1.1 401 Unauthorized
Date: Sat, 29 Feb 2020 05:34:47 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 74
Connection: close
X-Powered-By: Express
Access-Control-Allow-Origin: *
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
ETag: W/"4a-Php9aKHtjyxv6ihqICJ/7fT+upY"
Set-Cookie: connect.sid=s%3Ak1PT8nCDUeX7hfcsaa-6PUG6PNq-
Y0JV.%2B5NtJGUrgvCfQyWutyjbscDb%2BeoMJaeLYZ7Y2RXLKMM; Path=/; HttpOnly
Vary: Accept-Encoding

{"status":401,"message":"Unauthenticated","messageCode":"UNAUTHENTICATED"}
```

3.4.3. TLS V1.0 Supported

Risk Rating: **LOW**

Summary

Severity	**Low**
Confidence	**Certain**
Host	**https://tsng-api.clientx.net** **https://api-qa.clientx.net**
Classification	**OWASP 2013-A6 OWASP 2017-A9 PCI V3.2-6.5.4 CWE-327 CAPEC-217 WASC-4 HIPAA-164.306**

Issue Background

This issue was also identified on production as well as on QA server.

It has been identified that your web server is supporting an insecure transportation security protocol (TLS 1.0).

TLS 1.0 has many security flawsseveral flaws. Also, websites using TLS 1.0 are considered non-compliant by PCI since 30 June 2018. PCI standards require that TLS 1.0 can no longer be used for secure communications. All web servers and clients must transition to TLS 1.1 or above.

Impact

Among other weaknesses, TLS 1.0 is vulnerable to man-in-the-middle attacks, risking the integrity and authentication

of data sent between a website and a browser. An attacker can cause connection failures and they can trigger the use of TLS 1.0 to exploit vulnerabilities like BEAST (Browser Exploit Against SSL/TLS).

Disabling TLS 1.0 support on your server is sufficient to mitigate this issue.

Issue Remediation

The most effective way to ensure your server is secure is to disable TLS 1.0 and below on your web-server. Please note, disabling TLS 1.0 on your website will most likely mean most XP/IE 6.0 users are no longer supported for secure sessions.

Apache

To disable TLS 1.0 on your Apache server you can configure it using the following.

SSLProtocol All -SSLv2 -SSLv3 -TLSv1

This will give you support for TLS 1.1 and TLS 1.2, but explicitly removes support for TLS 1.0 and below. Check the config and then restart Apache.

apachectl configtest
sudo service apache2 restart

NGINX

To disable TLS 1.0 and below on your Apache server you can configure it using the following.

ssl_protocols TLSv1.1 TLSv1.2;

Similar to the Apache config above, you will get TLS 1.1+ support and no TLS 1.0 or SSL. You can check the config and restart.

sudo nginx -t
sudo service nginx restart

IIS

Its strongly recommended that all users upgrade to Microsoft Internet Information Services (IIS) version 7.0 running on Microsoft Windows Server 2008. IIS 7.0

IIS requires some registry tweaks and a server reboot. Microsoft have a support article at https://support.microsoft.com/en-us/help/187498/how-to-disable-pct-1.0,-ssl-2.0,-ssl-3.0,-or-tls-1.0-in-internet-information-services which deals with this topic. All you need to do is modify/create a registry DWORD value.

HKey_Local_Machine\System\CurrentControlSet\Control\SecurityProviders \SCHANNEL\Protocols

In the protocols directory, you will most likely have an SSL 2.0 key already. Create keys called 'SSL 3.0' and 'TLS 1.0' alongside it if needed. Under those, create Server keys and inside them a DWORD value called 'Enabled' and assign it a value of 0.

Once that's done reboot the server for the changes to take effect.

References

- How to disable TLS v1.0
- OWASP - Insecure Configuration Management
- OWASP - Insufficient Transport Layer Protection
- How to disable PCT 1.0, SSL 2.0, SSL 3.0, or TLS 1.0 in Internet Information Services
- IIS Crypto is a free tool that gives administrators the ability to enable or disable protocols, ciphers, hashes and key exchange algorithms on Windows Server 2003, 2008 and 2012
- Date Change for Migrating from SSL and Early TLS
- Browser Exploit Against SSL/TLS Attack (BEAST)

Vulnerability Classification

- OWASP 2013-A6
- OWASP 2017-A9
- PCI V3.2-6.5.4
- CWE-327
- CAPEC-217
- WASC-4
- HIPAA-164.306

3.4.3.1. https://clientx.app/

3.4.3.2. https://webqa.clientx.net/

3.4.4. Content type incorrectly stated

Risk Rating: **LOW**

Summary

Severity	**Low**
Confidence	**Firm**
Host	**https://webqa.clientx.net**
Classification	**CWE-436: Interpretation Conflict**

Issue Background

If a response specifies an incorrect content type then browsers may process the response in unexpected ways. If the content type is specified to be a render able text-based format, then the browser will usually attempt to interpret the response as being in that format, regardless of the actual contents of the response. Additionally, some other specified content types might sometimes be interpreted as HTML due to quirks in particular browsers. This behaviour might lead to otherwise "safe" content such as images being rendered as HTML, enabling cross-site scripting attacks in certain conditions.

The presence of an incorrect content type statement typically only constitutes a security flaw when the affected resource is dynamically generated, uploaded by a user, or otherwise contains user input. You should review the contents of affected responses, and the context in which they appear, to determine whether any vulnerability exists.

Impact

The response states that the content type is binary/octet-stream. However, it actually appears to contain unrecognized content.

If the URL path can be manipulated to end with ".html", the following browsers may interpret the response as HTML:

- Internet Explorer 11
- Internet Explorer 11 (Compatibility Mode)

Issue Remediation

For every response containing a message body, the application should include a single Content-type header that correctly and unambiguously states the MIME type of the content in the response body.

Additionally, the response header "X-content-type-options: nosniff" should be returned in all responses to reduce the likelihood that browsers will interpret content in a way that disregards the Content-type header.

Vulnerability Classification

- CWE-16: Configuration
- CWE-436: Interpretation Conflict

Request

```
GET /fontawesome-webfont.af7ae505a9eed503f8b8.woff2?v=4.7.0 HTTP/1.1
Host: webqa.clientx.net
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: application/font-woff2;q=1.0,application/font-woff;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: https://webqa.clientx.net/styles.958a919bad2e4966d3e9.css
Cookie: _ga=GA1.2.686046242.1582966525; _gid=GA1.2.1947657979.1582966525;
_gat_gtag_UA_146125476_2=1; _gat_%2F=1; _gat=1
```

Response

```
HTTP/1.1 200 OK
Date: Sat, 29 Feb 2020 08:57:21 GMT
Content-Type: font/woff2
Content-Length: 77160
Connection: close
Server: Apache/2.4.41 (Unix)
Last-Modified: Fri, 28 Feb 2020 07:28:20 GMT
ETag: "12d68-59f9dc8236d00"
Accept-Ranges: bytes
Cache-Control: max-age=86400
Expires: Sun, 01 Mar 2020 08:57:21 GMT

wOF2......-h.........-.........................?FFTM.. .`..r..
..(..X.6.$..p..... .....u[R    rGa...*..'.=.:..&..=r.*
......].t..E.n........1F...@....|....f.m.`.$...@d[BQ.$([U<+(..@P.5..`....>.P..;.(..
...[SNIP]...
```

3.4.5. Internal Server Error

Risk Rating: **LOW**

Summary

Severity	**Low**
Confidence	**Certain**
Host	**https://api-qa.clientx.net**
Classification	

Issue Background

Internal server error was identified in 11 instances on QA.

The server responded with an HTTP status 500, indicating there is a server-side error. Reasons may vary, and the behaviour should be analysed carefully.

Impact

The impact may vary depending on the condition. Generally, this indicates poor coding practices, not enough error

checking, sanitization and whitelisting. However, there might be a bigger issue, such as SQL injection.

Issue Remediation

Analyse this issue and review the application code in order to handle unexpected errors; this should be a generic practice, which does not disclose further information upon an error. All errors should be handled server-side only.

3.4.5.1. https://api-qa.clientx.net/api/locations?alert=%22%26ping%20-w%2025%20127.0.0.1%20%26%22&size=10&page=1&sort=3&order=3&type=3

Request

```
GET /api/locations?alert=%22%26ping%20-
w%2025%20127.0.0.1%20%26%22&size=10&page=1&sort=3&order=3&type=3&weather=true&orgId=5c498d5ed41
d39001102a0ae HTTP/1.1
Host: api-qa.clientx.net
Accept: application/json, text/plain, */*
Accept-Encoding: gzip,deflate
Accept-Language: en-US,en;q=0.9,en-us,en;q=0.5
Cache-Control: no-cache, no-store,no-cache
Cahce-Control: no-cache
Content-Type: application/json
Cookie: connect.sid=s%3A_Be58IKc3tzNvQeWMUq4d9sF52Qs-
XIv.UmQx%2FuOy4EfXxKvpu0IT2elvveylTZ5Es6xsz4XPHjI
Origin: https://webqa.clientx.net
Pragma: no-cache
Referer: https://webqa.clientx.net/locations
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/54.0.2840.99 Safari/537.36
x-access-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVlNThhNTE5YjQzY2U3MDAxMmU3YmYxMyIsInR5cGUiOiJvcm
ctYWRtaW4iLCJsb2NhdGlvbnMiOltdLCJvcmdJZCI6IjVjNDk4ZDVlZDQxZDM5MDAxMTAyYTBhZSIsImVtYWlsIjoiZWh0Y
XNoYW11bC5oYXFAbm14Z2xvYmFsc29mdHdhcmUuY29tIiwibmFtZSI6IkVodGFzaGFtIFVsIEhhcSIsIm5vdGlmeUVtYWls
Ijp0cnVlLCJub3RpZnlTTVMiOmZhbHNlLCJwcmVmcyI6W3sia2V5IjoibWVhc3VyZW1lbnQiLCJ2YWx1ZSI6ImltcGVyaWF
sIn0seyJrZXkiOiJtYXBWaWV3IiwidmFsdWUiOiJsaXN0In1dLCJsYW5ndWFnZSI6ImVuLVVTIiwiZGF0ZUZvcm1hdCI6Ik
1NL2RkL3l5eXkiLCJmaXJzdFRpbWVMb2dpbiI6ZmFsc2UsImNyZWF0ZWRBdCI6IjIwMjAtMDItMjhUMDU6Mjg6NTcuNjI0W
iIsImlzUmVtZW1iZXIiOnRydWUsImlhdCI6MTU4
```

Response

```
HTTP/1.1 500 Internal Server Error

X-Powered-By: Express
Connection: keep-alive
x-xss-protection: 1; mode=block
Content-Length: 86
x-frame-options: SAMEORIGIN
Access-Control-Allow-Origin: *
Vary: Accept-Encoding
Content-Type: application/json; charset=utf-8
Access-Control-Expose-Headers: refresh-token
Date: Sat, 29 Feb 2020 08:12:55 GMT
ETag: W/"56-65u2T2YxYVV2Oik18qCXrPiNxIo"

{"status":500,"message":"Server internal error","messageCode":"SERVER_INTERNAL_ERROR"}
```

3.4.6. Server Leaks Information via "X-Powered-By" HTTP Response Header Field

Risk Rating: **LOW**

Summary

Severity	**Low**

Confidence	**Certain**
Host	**https://api-qa.clientx.net**
Classification	**CWE-200, WASC-13**

Issue Background

The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Issue Remediation

Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

References

- Response headers leaking information

Vulnerability Classification

- CWE-200
- WASC-13

Request

```
OPTIONS /api/storage-
units/temp/hum?page=0&size=10&tempMin=&tempMax=&humMin=&humMax=&emcMin=&emcMax=&locId=&status=&
sort=priority&orgId=5c498d5ed41d39001102a0ae HTTP/1.1
Host: api-qa.clientx.net
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Access-Control-Request-Method: GET
Access-Control-Request-Headers: cahce-control,content-type,pragma,x-access-token
Referer: https://webqa.clientx.net/storage-unit
Origin: https://webqa.clientx.net
Connection: close
```

Response

```
HTTP/1.1 204 No Content
Date: Sat, 29 Feb 2020 11:21:03 GMT
Content-Length: 0
Connection: close
X-Powered-By: Express
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET,HEAD,PUT,PATCH,POST,DELETE
Vary: Access-Control-Request-Headers
Access-Control-Allow-Headers: cahce-control,content-type,pragma,x-access-token
```

3.4.7. X-Content-Type-Options Header Missing

Risk Rating: **LOW**

Summary

Severity	**Low**
Confidence	**Certain**

Host	**https://api-qa.clientx.net** **https:// webqa.clientx.net**
Classification	**OWASP 2013-A5 , OWASP 2017-A6, CWE-16, WASC-15**

Issue Background

The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

Impact

MIME type sniffing is a standard functionality in browsers to find an appropriate way to render data where the HTTP headers sent by the server are either inconclusive or missing.

This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the intended content type.

The problem arises once a website allows users to upload content which is then published on the web server. If an attacker can carry out XSS (Cross-site Scripting) attack by manipulating the content in a way to be accepted by the web application and rendered as HTML by the browser, it is possible to inject code in e.g. an image file and make the victim execute it by viewing the image.

Issue Remediation

- When serving resources, make sure you send the content-type header to appropriately match the type of the resource being served. For example, if you are serving an HTML page, you should send the HTTP header: Content-Type: text/html
- Add the X-Content-Type-Options header with a value of "nosniff" to inform the browser to trust what the site has sent is the appropriate content-type, and to not attempt "sniffing" the real content-type.X-Content-Type-Options: nosniff

References

- MIME Sniffing: feature or vulnerability
- X-Content-Type-Options HTTP Header

Vulnerability Classification

- OWASP 2013-A5
- OWASP 2017-A6
- CWE-16
- WASC-15

Request

```
GET https://api-qa.clientx.net/api/alerts/count?orgId=5c498d5ed41d39001102a0ae HTTP/1.1
Connection: keep-alive
Pragma: no-cache
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/80.0.3987.122 Safari/537.36
Cahce-Control: no-cache
Accept: application/json, text/plain, */*
Sec-Fetch-Dest: empty
Content-Type: application/json
x-access-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVlNThhNThjYjQzY2U3MDAxMmU3YmY4MSIsInR5cGUiOiJsb2
MtYWRtaW4iLCJsb2NhdGlvbnMiOlsiNWM0OTkxMjFkNDFkMzkwMDExMDJhMGFmIiwiNWM0OTkyYmNkNDFkMzkwMDExMDJhM
```

```
GI5IiwiNWQ4YjJkNzRhZDYxMzkwMDExMmIwZjI4IiwiNWQ5MWNkZmE1ODI3MTMwMDExMjZiZDVkIl0sIm9yZ0lkIjoiNWM0
OThkNWVkNDFkMzkwMDExMDJhMGFlIiwiZW1haWwiOiJheWVzaGEuc2FnaGVlckBubXhnbG9iYWxzb2Z0d2FyZS5jb20iLCJ
uYW1lIjoiQXllc2hhIiwibm90aWZ5RW1haWwiOnRydWUsIm5vdGlmeVNNUyI6ZmFsc2UsInBob25lIjoiMjM0NTY3OCIsIn
ByZWZzIjpbeyJrZXkiOiJtZWFzdXJlbWVudCIsInZhbHVlIjoiaW1wZXJpYWwifSx7ImtleSI6Imxhc3RBbm5vdW5jZW1lb
nRSZWFkRGF0ZSIsInZhbHVlIjoiMjAyMC0wMi0yOVQwNDo1Mjo1My41NDJaIn1dLCJsYW5ndWFnZSI6ImVuLVVTIiwiZGF0
ZUZvcm1hdCI6Ik1NL2RkL3l5eXkiLCJmaXJzdFRpbWVMb2dpbiI6ZmFsc2UsImNyZWF0ZWRBdCI6IjIwMjAtMDItMjhUMDU
6MzA6NTIuOTY2WiIsImlzUmVtZW1iZXIiOnRydWUsImlhdCI6MTU4Mjk2NDA0NiwiZXhwIjoxNTkwNzQwMDQ2fQ.XDAb9qJ
tSv_pZh2AqUTroaWgx88e0vnzRs9CkWHMUoc
Origin: https://webqa.clientx.net
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Referer: https://webqa.clientx.net/locations
Accept-Language: en-US,en;q=0.9
Host: api-qa.clientx.net
```

Response

```
HTTP/1.1 200 OK
Date: Sat, 29 Feb 2020 08:14:17 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 142
Connection: keep-alive
X-Powered-By: Express
Access-Control-Allow-Origin: *
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
Access-Control-Expose-Headers: refresh-token
ETag: W/"8e-BrG/oInOighKDAuq/0B1D8sXIaY"
Set-Cookie:
connect.sid=s%3Aeoh5fz3BrZO5Pict1_V8eDztYq75esnf.M0zpkHxccDfyGBpXfRjiKgqB5WiGEvr4K4d0hlolI%2F4;
Path=/; HttpOnly
Vary: Accept-Encoding

{"status":200,"messageCode":"GET_LIST_ALERTS_SUCCESS","message":"Get alerts
success","data":{"new":102,"acked":0,"resolved":6095,"snoozed":0}}
```

3.4.8. Cross Site Scripting Weakness (Reflected in JSON Response)

Risk Rating: **LOW**

Summary

Severity	**Low**
Confidence	**Certain**
Host	**https://api-qa.clientx.net**
Classification	**CWE-79, CWE-80, CWE-116, CWE-159**

Issue Background

Reflected cross-site scripting vulnerabilities were identified. These 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).

Impact

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.

Issue 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.

References

- Cross-site scripting
- Reflected cross-site scripting
- Using Burp to Find XSS issues

Vulnerability Classification

- CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
- CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)
- CWE-116: Improper Encoding or Escaping of Output
- CWE-159: Failure to Sanitize Special Element

3.4.8.1. https://api-qa.clientx.net/api/alerts/5e4f2c0aa6bd6e75c6b256ea/snooze/24

Request

```
PUT /api/alerts/5e4f2c0aa6bd6e75c6b256ea/snooze/24?orgId=5c498d5ed41d39001102a0ae HTTP/1.1
Host: api-qa.clientx.net
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cahce-Control: no-cache
Pragma: no-cache
Content-Type: application/json
x-access-token:
```

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVlNThhNTE5YjQzY2U3MDAxMmU3YmYxMyIsInR5cGUiOiJvcm
ctYWRtaW4iLCJsb2NhdGlvbnMiOltdLCJvcmdJZCI6IjVjNDk4ZDVlZDQxZDM5MDAxMTAyYTBhZSIsImVtYWlsIjoiZWh0Y
XNoYW11bC5oYXFAbm14Z2xvYmFsc29mdHdhcmUuY29tIiwibmFtZSI6IkVodGFzaGFtIFVsIEhhcSIsIm5vdGlmeUVtYWls
Ijp0cnVlLCJub3RpZnlTTVMiOnRydWUsInBob25lIjoiKzkyIDMyMSA1MjUyMzQ1IiwicHJlZnMiOlt7ImtleSI6Im1lYXN
1cmVtZW50IiwidmFsdWUiOiJtZXRyaWMifSx7ImtleSI6Im1hcFZpZXciLCJ2YWx1ZSI6Imxpc3QifSx7ImtleSI6InNob3
dPbmx5V29ya2luZ0JhbGxzIiwidmFsdWUiOiJmYWxzZSJ9LHsiaJV5Ijoic2hvd09ubHlXb3JraW5nQ2FibGVzIiwidmFsd
WUiOiJmYWxzZSJ9LHsiaJV5Ijoic2hvd09ubHlXb3JraW5nU3BlYXJzIiwidmFsdWUiOiJmYWxzZSJ9LHsiaJV5IjoibGFz
dEFubm91bmNlbWVudFJlYWREYXRlIiwidmFsdWUiOiIyMDIwLTAyLTI5VDA4OjEwOjA2LjY1M1oifV0sInRpdGxlIjoiMyI
sImxhbmd1YWdlIjoiZW4tVVMiLCJkYXRlRm9ybWF0IjoiTU0vZGQveXl5eSIsImNyZWF0ZWRBdCI6IjIwMjAtMDItMjhUMD
U6Mjg6NTcuNjI0WiIsImlzUmVtZW1iZXIiOmZhbHNlLCJpYXQiOjE1ODI5Nzg4NTIsImV4cCI6MTU4Mjk5MzI1Mn0.5JtAk
Ks6bu5wWsT012UgHEA_BW5CsmYu7Sxg3ZJsomM
Content-Length: 2
Origin: https://webqa.clientx.net
Connection: close
Referer: https://webqa.clientx.net/storage-unit/5d8b2d74ad613900112b0f2b/detail

{}imv2x<script>alert(1)</script>md8dk
```

Response

```
HTTP/1.1 400 Bad Request
Date: Sat, 29 Feb 2020 13:06:50 GMT
Content-Type: application/json; charset=utf-8
Connection: close
X-Powered-By: Express
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
Vary: Accept-Encoding
Content-Length: 936

{
"error": {
"message": "Unexpected token i in JSON at position 2",
"stack": "SyntaxError: Unexpected token i in JSON at position 2\n at JSON.parse (<anonymous>)\n
at parse (/app/tsng-
...[SNIP]...
at endReadableNT (_stream_readable.js:1183:12)\n at processTicksAndRejections
(internal/process/task_queues.js:80:21)",
"expose": true,
"statusCode": 400,
"status": 400,
"body": "{}imv2x<script>alert(1)</script>md8dk",
"type": "entity.parse.failed"
}
}
```

3.5. Information Risk Finding

3.5.1. Cross-domain script include

Risk Rating: **INFORMATION**

Summary

Severity	**Information**
Confidence	**Certain**
Host	**https://clientx.app** **https://webqa.clientx.net**
Classification	**CWE-829**

Issue Background

An instance of this issue was identified on production:

- https://clientx.app

There are 6 instances of this issue identified on QA server:

- /
- /alerts
- /assets/version.json
- /locations
- /robots.txt
- /storage-unit

The response dynamically includes the following scripts from other domains:

- https://apis.google.com/js/platform.js
- https://www.googletagmanager.com/gtag/js?id=UA-146125476-1

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.

Impact

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.

Issue 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.

References

- **Subresource Integrity**

Vulnerability Classification

- **CWE-829: Inclusion of Functionality from Untrusted Control Sphere**

3.5.1.1. https://clientx.app

Request

```
GET /login HTTP/1.1
Host: clientx.app
Connection: close
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/79.0.3945.130 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,applicati
on/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: _ga=GA1.2.461878407.1582423339; _gid=GA1.2.276150750.1582423339;
_gat_gtag_UA_146125476_1=1; _gat=1; G_ENABLED_IDPS=google
```

Response

```
HTTP/1.1 200 OK
Date: Sun, 23 Feb 2020 02:04:50 GMT
Content-Type: text/html
Content-Length: 3368
Connection: close
Server: Apache/2.4.41 (Unix)
Last-Modified: Wed, 22 Jan 2020 07:58:11 GMT
ETag: "d28-59cb5e2c30ac0-gzip"
Accept-Ranges: bytes
Cache-Control: max-age=0, no-cache, no-store, must-revalidate
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Vary: Accept-Encoding
Pragma: no-cache

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>Clientx</title>
<base href="/" />

<meta name="viewport" content="width=device-width, initial-scale=1
...[SNIP]...
</app-root>
<script src="https://apis.google.com/js/platform.js" async defer></script>
...[SNIP]...
```

3.5.1.2. https://webqa.clientx.net/

Request

```
GET / HTTP/1.1
Host: webqa.clientx.net
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;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/69.0.3497.100 Safari/537.36
Connection: close
Cache-Control: max-age=0
```

Response

```
Server: Apache/2.4.41 (Unix)
Last-Modified: Fri, 28 Feb 2020 07:28:20 GMT
ETag: "d1e-59f9dc8236d00-gzip"
Accept-Ranges: bytes
Cache-Control: max-age=0, no-cache, no-store, must-revalidate
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Vary: Accept-Encoding
Pragma: no-cache

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>ClientX</title>
<base href="/" />

<meta name="viewport" content="width=device-width, initial-scale=1
...[SNIP]...
</app-root>
<script src="https://apis.google.com/js/platform.js" async defer></script>
...[SNIP]...
```

3.5.2. Frameable response (potential Clickjacking)

Risk Rating: INFORMATION

Summary

Severity	**Information**
Confidence	**Firm**
Host	**https://clientx.app** **https://webqa.clientx.net/**
Classification	**CWE-693**

Issue Background

An instance of this issue was identified on production.

- /login

There are 6 instances of this issue:

- /
- /alerts
- /assets/version.json
- /locations
- /robots.txt
- /storage-unit

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 defences 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 defence 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.

Impact

Clickjacking is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on a framed page when they were intending to click on the top level page. Thus, the attacker is "hijacking" clicks meant for their page and routing them to other another page, most likely owned by another application, domain, or both.

Using a similar technique, keystrokes can also be hijacked. With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker.

Issue 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.

References

- X-Frame-Options

Vulnerability Classification

- CWE-693: Protection Mechanism Failure

3.5.2.1. https://clientx.app/login

Request

```
GET /login HTTP/1.1
Host: clientx.app
Connection: close
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/79.0.3945.130 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,applicati
on/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: _ga=GA1.2.461878407.1582423339; _gid=GA1.2.276150750.1582423339;
_gat_gtag_UA_146125476_1=1; _gat=1; G_ENABLED_IDPS=google
```

Response

```
HTTP/1.1 200 OK
Date: Sun, 23 Feb 2020 02:04:50 GMT
Content-Type: text/html
Content-Length: 3368
Connection: close
Server: Apache/2.4.41 (Unix)
Last-Modified: Wed, 22 Jan 2020 07:58:11 GMT
ETag: "d28-59cb5e2c30ac0-gzip"
Accept-Ranges: bytes
Cache-Control: max-age=0, no-cache, no-store, must-revalidate
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Vary: Accept-Encoding
Pragma: no-cache

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>Clientx</title>
<base href="/" />

<meta name="viewport" content="width=device-width, initial-scale=1
...[SNIP]...
```

3.5.2.2. https://webqa.clientx.net/

Request

```
GET / HTTP/1.1
Host: webqa.clientx.net
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;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/69.0.3497.100 Safari/537.36
Connection: close
Cache-Control: max-age=0
```

Response

```
HTTP/1.1 200 OK
Date: Sat, 29 Feb 2020 05:33:51 GMT
Content-Type: text/html
Content-Length: 3358
Connection: close
Server: Apache/2.4.41 (Unix)
Last-Modified: Fri, 28 Feb 2020 07:28:20 GMT
ETag: "d1e-59f9dc8236d00-gzip"
Accept-Ranges: bytes
Cache-Control: max-age=0, no-cache, no-store, must-revalidate
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Vary: Accept-Encoding
Pragma: no-cache

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>ClientX</title>
<base href="/" />

<meta name="viewport" content="width=device-width, initial-scale=1
...[SNIP]...
```

3.5.3. Cacheable HTTPS response

Risk Rating: **INFORMATION**

Summary

Severity	**Information**
Confidence	**Certain**
Host	**https://tsng-api.clientx.net** **https://api-qa.clientx.net**
Classification	**CWE-524, CWE-525**

Issue Background

An instance of this issue identified on production:

- /api/forgot-password

There are 6 instances of this issue identified on QA server:

- https://api-qa.clientx.net/api
- https://api-qa.clientx.net/api/apikey

- https://api-qa.clientx.net/api/invites
- https://api-qa.clientx.net/api/undefined/units
- https://api-qa.clientx.net/api/login
- https://api-qa.clientx.net/api/users/5e58a519b43ce70012e7bf13

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.

Impact

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.

Issue 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

References

- Web Content Caching

Vulnerability Classification

- CWE-524: Information Exposure Through Caching
- CWE-525: Information Exposure Through Browser Caching

3.5.3.1. https://tsng-api.clientx.net/api/forgot-password

Request

```
POST /api/forgot-password HTTP/1.1
Host: tsng-api.clientx.net
Connection: close
Content-Length: 25
Accept: application/json, text/plain, */*
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/79.0.3945.130 Safari/537.36
Content-Type: application/json
Origin: https://clientx.app
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: cors
Referer: https://clientx.app/forgot
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
{"email":"test@test.com"}
```

Response

```
HTTP/1.1 200 OK
Date: Sun, 23 Feb 2020 02:06:20 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 90
Connection: close
X-Powered-By: Express
Access-Control-Allow-Origin: *
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
Access-Control-Expose-Headers: refresh-token
ETag: W/"5a-7NwPhzcHxWdcdgALZHZ5gZjz/so"
Set-Cookie:
connect.sid=s%3ALRsGeQzKz2D40rdR95mAs6mzbzVBQ8Bb.eEmh4wVGdrCzNVwfb6W9%2BUSvGnXGU83MTw96PUEujRw;
Path=/; HttpOnly
Vary: Accept-Encoding

{"status":200,"message":"Forgot password success","messageCode":"FORGOT_PASSWORD_SUCCESS"}
```

3.5.3.2. https://api-qa.clientx.net/api

Request

```
PUT api-qa.clientx.net/api/users/5e58a519b43ce70012e7bf13 HTTP/1.1
Host: api-qa.clientx.net
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cahce-Control: no-cache
Pragma: no-cache
Content-Type: application/json
x-access-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVlNThhNTE5YjQzY2U3MDAxMmU3YmYxMyIsInR5cGUiOiJvcm
ctYWRtaW4iLCJsb2NhdGlvbnMiOltdLCJvcmdJZCI6IjVjNDk4ZDVlZDQxZDM5MDAxMTAyYTBhZSIsImVtYWlsIjoiZWh0Y
XNoYW11bC5oYXFAbm14Z2xvYmFsc29mdHdhcmUuY29tIiwibmFtZSI6IkVodGFzaGFtIFVsIEhhcSIsIm5vdGlmeUVtYWls
Ijp0cnVlLCJub3RpZnlTTVMiOnRydWUsInBob25lIjoiKzkyIDMyMSA1MjUyMzQ1IiwicHJlZnMiOlt7ImtleSI6Im1lYXN
1cmVtZW50IiwidmFsdWUiOiJtZXRyaWMifSx7ImtleSI6Im1hcFZpZXciLCJ2YWx1ZSI6Imxpc3QifSx7ImtleSI6InNob3
dPbmx5V29ya2luZ0JhbGxzIiwidmFsdWUiOiJmYWxzZSJ9LHsiaJV5Ijoic2hvd09ubHlXb3JraW5nQ2FibGVzIiwidmFsd
WUiOiJmYWxzZSJ9LHsiaJV5Ijoic2hvd09ubHlXb3JraW5nU3BlYXJzIiwidmFsdWUiOiJmYWxzZSJ9LHsiaJV5IjoibGFz
dEFubm91bmNlbWVudFJlYWREYXRlIiwidmFsdWUiOiIyMDIwLTAyLTI5VDA4OjEwOjA2LjY1M1oifV0sInRpdGxlIjoiMyI
sImxhbmd1YWdlIjoiZW4tVVMiLCJkYXRlRm9ybWF0IjoiTU0vZGQveXl5eSIsImNyZWF0ZWRBdCI6IjIwMjAtMDItMjhUMD
U6Mjg6NTcuNjI0WiIsImlzUmVtZW1iZXIiOmZhbHNlLCJpYXQiOjE1ODI5Nzg4NTIsImV4cCI6MTU4Mjk5MzIlMn0.5JtAk
Ks6bu5wWsT012UgHEA_BW5CsmYu7Sxg3ZJsomM
Content-Length: 2
Origin: https://webqa.clientx.net
Connection: close
Referer: https://webqa.clientx.net/storage-unit/5d8b2d74ad613900112b0f2b/detail

{}
```

Response

```
HTTP/1.1 200 OK
Date: Sat, 29 Feb 2020 12:21:09 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 84
Connection: close
X-Powered-By: Express
Access-Control-Allow-Origin: *
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
Access-Control-Expose-Headers: refresh-token
refresh-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVlNThhNTE5YjQzY2U3MDAxMmU3YmYxMyIsInR5cGUiOiJvcm
ctYWRtaW4iLCJsb2NhdGlvbnMiOltdLCJvcmdJZCI6IjVjNDk4ZDVlZDQxZDM5MDAxMTAyYTBhZSIsImVtYWlsIjoiZWh0Y
XNoYW11bC5oYXFAbm14Z2xvYmFsc29mdHdhcmUuY29tIiwibmFtZSI6IkVodGFzaGFtIFVsIEhhcSIsIm5vdGlmeUVtYWls
Ijp0cnVlLCJub3RpZnlTTVMiOnRydWUsInBob25lIjoiKzkyIDMyMSA1MjUyMzQ1IiwicHJlZnMiOlt7ImtleSI6Im1lYXN
1cmVtZW50IiwidmFsdWUiOiJtZXRyaWMifSx7ImtleSI6Im1hcFZpZXciLCJ2YWx1ZSI6Imxpc3QifSx7ImtleSI6InNob3
```

```
dPbmx5V29ya2luZ0JhbGxzIiwidmFsdWUiOiJmYWxzZSJ9LHsiaJV5Ijoic2hvd09ubHlXb3JraW5nQ2FibGVzIiwidmFsd
WUiOiJmYWxzZSJ9LHsiaJV5Ijoic2hvd09ubHlXb3JraW5nU3BlYXJzIiwidmFsdWUiOiJmYWxzZSJ9LHsiaJV5IjoibGFz
dEFubm91bmNlbWVudFJlYWREYXRlIiwidmFsdWUiOiIyMDIwLTAyLTI5VDA4OjEwOjA2LjY1M1oifV0sInRpdGxlIjoiMyI
sImxhbmd1YWdlIjoiZW4tVVMiLCJkYXRlRm9ybWF0IjoiTU0vZGQveXl5eSIsImNyZWF0ZWRBdCI6IjIwMjAtMDItMjhUMD
U6Mjg6NTcuNjI0WiIsImlzUmVtZW1iZXIiOmZhbHNlLCJpYXQiOjE1ODI5Nzg4NjksImV4cCI6MTU4Mjk5MzI2OX0.Dkzkm
yJDea2jNTamDOqS1iis6RP5KFvNvnOx80g2kFk
ETag: W/"54-RveJQyGnIbvfAJg+iTp2xhXCaL4"
Set-Cookie: connect.sid=s%3Az-a1aLUDg_ZV0wGP8D-
2Jjqim1h9oF6p.oe4pFj2QNIXjm0kIFMwjz9NudOojH0Wl6wzVIKZ%2Bteg; Path=/; HttpOnly
Vary: Accept-Encoding

{"status":200,"messageCode":"SNOOZE_ALERT_SUCCESS","message":"Snooze alert success"}
```

3.5.4. OPTIONS Method Enabled

Risk Rating: **INFORMATION**

Summary

Severity	**Information**
Confidence	**Certain**
Host	**https://clientx.app** **https://webqa.clientx.net**
Classification	**OWASP 2013-A5 OWASP 2017-A6 CWE-16 CAPEC-107 WASC-14**

Issue Background

There are 3 instances of this issue identified on production:

- /asset/
- /asset/images/
- /asset/images/flag/

There are 6 instances of this issue identified on QA server:

- /
- /asset/
- /asset/images/
- /admin/
- /admin/devices/
- /admin/devices/5d52521f611ec7001153740a/

It has been detected that OPTIONS method is allowed. This issue is reported as extra information.

Impact

Information disclosed from this page can be used to gain additional information about the target system

Issue Remediation

Disable OPTIONS method in all production systems.

References

- Testing for HTTP Methods and XST (OWASP-CM-008)
- HTTP/1.1: Method Definitions

Vulnerability Classification

- OWASP 2013-A5
- OWASP 2017-A6
- CWE-16
- CAPEC-107
- WASC-14

3.5.4.1. https://clientx.app/assets/

Request

```
OPTIONS /assets/ HTTP/1.1
Host: clientx.app
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-us,en;q=0.5
Cache-Control: no-cache
Content-Length: 0
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/54.0.2840.99 Safari/537.36
```

Response

```
HTTP/1.1 200 OK
Server: Apache/2.4.41 (Unix)
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Connection: keep-alive
Allow: POST,OPTIONS,HEAD,GET,TRACE
Content-Length: 0
Pragma: no-cache
Content-Type: text/html
Date: Sun, 23 Feb 2020 02:18:15 GMT
Cache-Control: max-age=0, no-cache, no-store, must-revalidate
```

3.5.4.2. https://webqa.clientx.net/

Request

```
OPTIONS / HTTP/1.1
Host: webqa.clientx.net
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-us,en;q=0.5
Cache-Control: no-cache
Content-Length: 0
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/54.0.2840.99 Safari/537.36
```

Response

```
HTTP/1.1 200 OK
Server: Apache/2.4.41 (Unix)
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Connection: keep-alive
Allow: GET,POST,OPTIONS,HEAD,TRACE
Content-Length: 0
Pragma: no-cache
Content-Type: text/html
Date: Sat, 29 Feb 2020 08:10:21 GMT
Cache-Control: max-age=0, no-cache, no-store, must-revalidate
```

3.5.5. Input returned in response (reflected)

Risk Rating: INFORMATION

Summary

Severity	**Information**
Confidence	**Certain**
Host	**https://api-qa.clientx.net**
Classification	**CWE-20, CWE-116**

Issue Background

There are 4 instances of this issue identified on QA server:

- /api/alerts [URL path filename]
- /api/alerts [URL path folder 1]
- /api/alerts/5e4f2c0aa6bd6e75c6b256ea/snooze/24
- /api/alerts/5e5a53b3e83df70599c4dfc6/ack/true

Impact

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.

Vulnerability Classification

- CWE-20: Improper Input Validation
- CWE-116: Improper Encoding or Escaping of Output

3.5.5.1. https://api-qa.clientx.net/api/alerts [URL path filename]

The value of the URL path filename is copied into the application's response

Request

```
GET
/api/alerts/5e4f2c0aa6bd6e75c6b256ea/snooze/24?size=0&page=0&sort=&order=&acked=null&snoozed=fa
lse&resolved=false&type=&status=new&suId=5d8b2d74ad613900112b0f2b&chart=false&orgId=5c498d5ed41
d39001102a0ae HTTP/1.1
Host: api-qa.clientx.net
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cahce-Control: no-cache
Pragma: no-cache
Content-Type: application/json
x-access-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVlNThhNTE5YjQzY2U3MDAxMmU3YmYxMyIsInR5cGUiOiJvcm
ctYWRtaW4iLCJsb2NhdGlvbnMiOltdLCJvcmdJZCI6IjVjNDk4ZDVlZDQxZDM5MDAxMTAyYTBhZSIsImVtYWlsIjoiZWh0Y
XNoYW11bC5oYXFAbm14Z2xvYmFsc29mdHdhcmUuY29tIiwibmFtZSI6IkVodGFzaGFtIFVsIEhhcSIsIm5vdGlmeUVtYWls
Ijp0cnVlLCJub3RpZnlTTVMiOnRydWUsInBob25lIjoiKzkyIDMyMSA1MjUyMzQ1IiwicHJlZnMiOlt7ImtleSI6Im1lYXN
1cmVtZW50IiwidmFsdWUiOiJtZXRyaWMifSx7ImtleSI6ImlhcFZpZXciLCJ2YWx1ZSI6Imxpc3QifSx7ImtleSI6InNob3
dPbmx5V29ya2luZ0JhbGxzIiwidmFsdWUiOiJmYWxzZSJ9LHsiaJV5Ijoic2hvd09ubHlXb3JraW5nQ2FibGVzIiwidmFsd
WUiOiJmYWxzZSJ9LHsiaJV5Ijoic2hvd09ubHlXb3JraW5nU3BlYXJzIiwidmFsdWUiOiJmYWxzZSJ9LHsiaJV5IjoibGFz
dEFubm91bmNlbWVudFJlYWREYXRlIiwidmFsdWUiOiIyMDIwLTAyLTI5VDA4OjEwOjA2LjY1M1oifV0sInRpdGxlIjoiMyI
sImxhbmd1YWdlIjoiZW4tVVMiLCJkYXRlRm9ybWF0IjoiTU0vZGQveXl5eSIsImNyZWF0ZWRBdCI6IjIwMjAtMDItMjhUMD
U6Mjg6NTcuNjI0WiIsImlzUmVtZW1iZXIiOmZhbHNlLCJpYXQiOjE1ODI5Nzg4NjksImV4cCI6MTU4Mjk5MzI2OX0.Dkzkm
yJDea2jNTamDOqS1iis6RP5KFvNvnOx80g2kFk
Origin: https://webqa.clientx.net
Connection: close
Referer: https://webqa.clientx.net/storage-unit/5d8b2d74ad613900112b0f2b/detail
If-None-Match: W/"8369-kzu32Meb3l1iro/tukY9pem73ns"
```

Response

```
HTTP/1.1 404 Not Found
Date: Sat, 29 Feb 2020 12:46:46 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 159
Connection: close
X-Powered-By: Express
Access-Control-Allow-Origin: *
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
Access-Control-Expose-Headers: refresh-token
Content-Security-Policy: default-src 'none'
X-Content-Type-Options: nosniff
Vary: Accept-Encoding

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Cannot GET /api/alertsat43q8q3ez</pre>
</body>
</html>
```

3.5.6. Cross-domain Referer leakage

Risk Rating: **INFORMATION**

Summary

Severity	**Information**

Confidence	**Certain**
Host	**https://webqa.clientx.net**
Classification	**OWASP 2017-A3 , OWASP 2013-A6, CWE-200**

Issue Background

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.

Impact

If there is no adequate prevention in place, the URL itself, and even sensitive information contained in the URL will be leaked to the cross-site.

The lack of Referrer-Policy header might affect privacy of the users and site's itself

Issue 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.

References

- Referrer Policy
- Referrer-Policy - MDN
- Referrer-Policy HTTP Header
- A New Security Header: Referrer Policy
- Can I Use Referrer-Policy

Vulnerability Classification

- OWASP 2017-A3
- OWASP 2013-A6
- CWE-200

3.5.6.1. https://webqa.clientx.net/assets/version.json

The page was loaded from a URL containing a query string:

- https://webqa.clientx.net/assets/version.json

The response contains the following links to other domains:

- https://apis.google.com/js/platform.js
- https://fonts.googleapis.com/css?family=Titillium+Web:300,300i,400,400i,600,600i,700,700i,900
- https://fonts.googleapis.com/icon?family=Material+Icons
- https://pro.fontawesome.com/releases/v5.12.1/css/all.css
- https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css
- https://www.googletagmanager.com/gtag/js?id=UA-146125476-2
- https://www.googletagmanager.com/ns.html?id=GTM-KZFMNRJ

Request

```
GET /assets/version.json?t=1582967434442 HTTP/1.1
Host: webqa.clientx.net
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: https://webqa.clientx.net/locations
Cookie: _ga=GA1.2.686046242.1582966525; _gid=GA1.2.1947657979.1582966525; G_ENABLED_IDPS=google
```

Response

```
HTTP/1.1 200 OK
Date: Sat, 29 Feb 2020 09:12:17 GMT
Content-Type: text/html
Content-Length: 3358
Connection: close
Server: Apache/2.4.41 (Unix)
Last-Modified: Fri, 28 Feb 2020 07:28:20 GMT
ETag: "d1e-59f9dc8236d00-gzip"
Accept-Ranges: bytes
Cache-Control: max-age=0, no-cache, no-store, must-revalidate
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Vary: Accept-Encoding
Pragma: no-cache

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>ClientX</title>
<base href="/" />

<meta name="viewport" content="width=device-width, initial-scale=1
...[SNIP]...
</app-root>
<script src="https://apis.google.com/js/platform.js" async defer></script>
...[SNIP]...
```

3.5.7. Backup file

Risk Rating: **INFORMATION**

Summary

Severity	**Information**
Confidence	**Certain**
Host	**https://api-qa.clientx.net**

Classification	**PCI v3.1-6.5.8, PCI v3.2-6.5.8, CAPEC-87, CWE-530, HIPAA-530, ISO27001-A.18.1.3, WASC-34, OWASP 2013-A7, OWASP 2017-A5**

Issue Background

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.

Impact

Backup files can contain old or current versions of a file on the web server. This could include sensitive data such as password files or even the application's source code. This form of issue normally leads to further vulnerabilities or, at worst, sensitive information disclosure.

Issue 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 recurring.

References

- Review Old, Backup and Unreferenced Files for Sensitive Information

Vulnerability Classification

- CWE-530: Exposure of Backup File to an Unauthorized Control Sphere
- CAPEC-87
- ISO27001-A.18.1.3
- WASC-34
- OWASP 2013-A7
- OWASP 2017-A5

3.5.7.1. https://api-qa.clientx.net/api/alerts/bak_count

Request

```
GET /api/alerts/bak_count?orgId=5c498d5ed41d39001102a0ae HTTP/1.1
Host: api-qa.clientx.net
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cahce-Control: no-cache
Pragma: no-cache
x-access-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVlNThhNTE5YjQzY2U3MDAxMmU3YmYxMyIsInR5cGUiOiJvcm
ctYWRtaW4iLCJsb2NhdGlvbnMiOltdLCJvcmdJZCI6IjVjNDk4ZDVlZDQxZDM5MDAxMTAyYTBhZSIsImVtYWlsIjoiZWh0Y
XNoYW11bC5oYXFAbm14Z2xvYmFsc29mdHdhcmUuY29tIiwibmFtZSI6IkVodGFzaGFtIFVsIEhhcSIsIm5vdGlmeUVtYWls
Ijp0cnVlLCJub3RpZnlTTVMiOnRydWUsInBob25lIjoiKzkyIDMyMSA1MjEFubm91bmNlbWVudFJlYWREYXRlIiwidmFsdW
UiOiIyMDIwLTAyLTI5VDA4OjEwOjA2LjY1M1oifV0sInRpdGxlIjoiMyIsImxhbmd1YWdlIjoiZW4tVVMiLCJkYXRlRm9yb
WF0IjoiTU0vZGQveXl5eSIsImNyZWF0ZWRBdCI6IjIwMjAtMDItMjhUMDU6Mjg6NTcuNjI0WiIsImlzUmVtZW1iZXIiOmZh
bHNlLCJpYXQiOjE1ODI5Nzg4NjksImV4cCI6MTU4Mjk5MzI2OX0.DkzkmyJDea2jNTamDOqS1iis6RP5KFvNvnOx80g2kFk
Origin: https://webqa.clientx.net
Connection: close
Referer: https://webqa.clientx.net/storage-unit/5d8b2d74ad613900112b0f2b/detail
If-None-Match: W/"8e-IWWo3/Vo+e1Dpfg9rKCvbwLvIfs"
Cookie:

…
```

Response

```
HTTP/1.1 500 Internal Server Error
Date: Sat, 29 Feb 2020 13:20:38 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 86
Connection: close
X-Powered-By: Express
Access-Control-Allow-Origin: *
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
Access-Control-Expose-Headers: refresh-token
refresh-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVlNThhNTE5YjQzY2U3MDAxMmU3YmYxMyIsInR5cGUiOiJvcm
ctYWRtaW4iLCJsb2NhdGlvbnMiOltdLCJvcmdJZCI6IjVjNDk4ZDVlZDQxZDM5MDAxMTAyYTBhZSIsImVtYWlsIjoiZWh0Y
XNoYW11bC5oYXFAbm14Z2xvYmFsc29mdHdhcmUuY29tIiwibmFtZSI6IkVodGFzaGFtIFVsIEhhcSIsIm5vdGlmeUVtYWls
Ijp0cnVlLCJub3RpZnlTTVMiOnRydWUsInBob25lIjoiKzkyIDMyMSA1MjUyMzQ1IiwicHJlZnMiOlt7ImtleSI6Im1lYXN
1cmVtZW50IiwidmFsdWUiOiJtZXRyaWMifSx7ImtleSI6Im1hcFZpZXciLCJ2YWx1ZSI6Imxpc3QifSx7ImtleSI6InNob3
dPbmx5V29ya2luZ0JhbGxzIiwidmFsdWUiOiJmYWxzZSJ9LHsiaJV5Ijoic2hvd09ubHlXb3JraW5nQ2FibGVzIiwidmFsd
WUiOiJmYWxzZSJ9LHsiaJV5Ijoic2hvd09ubHlXb3JraW5nU3BlYXJzIiwidmFsdWUiOiJmYWxzZSJ9LHsiaJV5IjoibGFz
dEFubm91bmNlbWVudFJlYWREYXRlIiwidmFsdWUiOiIyMDIwLTAyLTI5VDA4OjEwOjA2LjY1M1oifV0sInRpdGxlIjoiMyI
sImxhbmd1YWdlIjoiZW4tVVMiLCJkYXRlRm9ybWF0IjoiTU0vZGQveXl5eSIsImNyZWF0ZWRBdCI6IjIwMjAtMDItMjhUMD
U6Mjg6NTcuNjI0WiIsImlzUmVtZW1iZXIiOmZhbHNlLCJpYXQiOjE1ODI5ODI0MzgsImV4cCI6MTU4Mjk5NjgzOH0.tr6oS
FAvdFUv78PelCFl_vvM1nsiVV3ItAp8R7n2aTI
ETag: W/"56-65u2T2YxYVV2Oik18qCXrPiNxIo"
Vary: Accept-Encoding

{"status":500,"message":"Server internal error","messageCode":"SERVER_INTERNAL_ERROR"}
```

3.5.8. Email addresses disclosed

Risk Rating: **INFORMATION**

Summary

Severity	**Information**
Confidence	**Certain**
Host	**https://api-qa.clientx.net**
Classification	**CAPEC-118, CWE-200, ISO27001-A.9.4.1, WASC-13, OWASP PC-C7**

Issue Background

There are 2 instances of this issue identified on QA:

- https://api-qa.clientx.net/api
- https://webqa.clientx.net/main.2e77111944f4d56a67b8.js

The presence of email addresses within application responses does not necessarily constitute a security vulnerability. Email addresses may appear intentionally within contact information, and many applications (such as web mail) include arbitrary third-party email addresses within their core content.

However, email addresses of developers and other individuals (whether appearing on-screen or hidden within page source) may disclose information that is useful to an attacker; for example, they may represent usernames that can be used at the application's login, and they may be used in social engineering attacks against the organization's personnel. Unnecessary or excessive disclosure of email addresses may also lead to an increase in the volume of spam email received

Impact

Email addresses discovered within the application can be used by both spam email engines and also brute-force tools. Furthermore, valid email addresses may lead to social engineering attacks.

Issue Remediation

Consider removing any email addresses that are unnecessary, or replacing personal addresses with anonymous mailbox addresses (such as helpdesk@example.com).

To reduce the quantity of spam sent to anonymous mailbox addresses, consider hiding the email address and instead providing a form that generates the email server-side, protected by a CAPTCHA if necessary

Vulnerability Classification

- CWE-200: Information Exposure
- CAPEC-118
- WASC-13
- OWASP PC-C7

3.5.8.1. https://api-qa.clientx.net/api

The following email addresses were disclosed in the response:

- thanhha3105@gmail.com
- ali@clientx.com
- trong@clientx.com
- achint@clientx.com
- johngh@gmail.com
- anh@gmail.com
- glenn@clientx.com
- thanhha3106@gmail.com
- ngocanhla168@gmail.com
- anh@clientx.com
- org@clientx.com

Numerous email addresses were found to be disclosed and the above are a sample subset.

This issue was found in multiple locations under the reported path.

Request

```
GET
/api/alerts?size=0&page=0&sort=&order=&acked=null&snoozed=false&resolved=false&type=&status=new
&suId=5d8b2d74ad613900112b0f2b&chart=false&orgId=5c498d5ed41d39001102a0ae HTTP/1.1
Host: api-qa.clientx.net
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cahce-Control: no-cache
Pragma: no-cache
Content-Type: application/json
x-access-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVlNThhNTE5YjQzY2U3MDAxMmU3YmYxMyIsInR5cGUiOiJvcm
ctYWRtaW4iLCJsb2NhdGlvbnMiOltdLCJvcmdJZCI6IjVjNDk4ZDVlZDQxZDM5MDAxMTAyYTBhZSIsImVtYWlsIjoiZWh0Y
XNoYW11bC5oYXFAbm14Z2xvYmFsc29mdHdhcmUuY29tIiwibmFtZSI6IkVodGFzaGFtIFVsIEhhcSIsIm5vdGlmeUVtYWls
Ijp0cnVlLCJub3RpZnlTTVMiOnRydWUsInBob25lIjoiKzkyIDMyMSA1MjUyMzQ1IiwicHJlZnMiOlt7ImtleSI6Im1lYXN
1cmVtZW50IiwidmFsdWUiOiJtZXRyaWMifSx7ImtleSI6Im1hcFZpZXciLCJ2YWx1ZSI6Imxpc3QifSx7ImtleSI6InNob3
dPbmx5V29ya2luZ0JhbGxzIiwidmFsdWUiOiJmYWxzZSJ9LHsiaJV5Ijoic2hvd09ubHlXb3JraW5nQ2FibGVzIiwidmFsd
WUiOiJmYWxzZSJ9LHsiaJV5Ijoic2hvd09ubHlXb3JraW5nU3BlYXJzIiwidmFsdWUiOiJmYWxzZSJ9LHsiaJV5IjoibGFz
dEFubm91bmNlbWVudFJlYWREYXRlIiwidmFsdWUiOiIyMDIwLTAyLTI5VDA4OjEwOjA2LjY1M1oifV0sInRpdGxlIjoiMyI
sImxhbmd1YWdlIjoiZW4tVVMiLCJkYXRlRm9ybWF0IjoiTU0vZGQveXl5eSIsImNyZWF0ZWRBdCI6IjIwMjAtMDItMjhUMD
U6Mjg6NTcuNjI0WiIsImlzUmVtZW1iZXIiOmZhbHNlLCJpYXQiOjE1ODI5Nzg4NjksImV4cCI6MTU4Mjk5MzI2OX0.Dkzkm
yJDea2jNTamDOqS1iis6RP5KFvNvnOx80g2kFk
Origin: https://webqa.clientx.net
Connection: close
Referer: https://webqa.clientx.net/storage-unit/5d8b2d74ad613900112b0f2b/detail
```

```
If-None-Match: W/"8369-kzu32Meb3l1iro/tukY9pem73ns"
```

Response

```
HTTP/1.1 200 OK
Date: Sat, 29 Feb 2020 12:21:11 GMT
Content-Type: application/json; charset=utf-8
Connection: close
X-Powered-By: Express
Access-Control-Allow-Origin: *
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
Access-Control-Expose-Headers: refresh-token
refresh-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVlNThhNTE5YjQzY2U3MDAxMmU3YmYxMyIsInR5cGUiOiJvcm
ctYWRtaW4iLCJsb2NhdGlvbnMiOltdLCJvcmdJZCI6IjVjNDk4ZDVlZDQxZDM5MDAxMTAyYTBhZSIsImVtYWlsIjoiZWh0Y
XNoYW11bC5oYXFAbm14Z2xvYmFsc29mdHdhcmUuY29tIiwibmFtZSI6IkVodGFzaGFtIFVsIEhhcSIsIm5vdGlmeUVtYWls
Ijp0cnVlLCJub3RpZnlTTVMiOnRydWUsInBob25lIjoiKzkyIDMyMSA1MjUyMzQ1IiwicHJlZnMiOlt7ImtleSI6Im1lYXN
1cmVtZW50IiwidmFsdWUiOiJtZXRyaWMifSx7ImtleSI6Im1hcFZpZXciLCJ2YWx1ZSI6Imxpc3QifSx7ImtleSI6InNob3
dPbmx5V29ya2luZ0JhbGxzIiwidmFsdWUiOiJmYWxzZSJ9LHsiaJV5Ijoic2hvd09ubHlXb3JraW5nQ2FibGVzIiwidmFsd
WUiOiJmYWxzZSJ9LHsiaJV5Ijoic2hvd09ubHlXb3JraW5nU3BlYXJzIiwidmFsdWUiOiJmYWxzZSJ9LHsiaJV5IjoibGFz
dEFubm91bmNlbWVudFJlYWREYXRlIiwidmFsdWUiOiIyMDIwLTAyLTI5VDA4OjEwOjA2LjY1M1oifV0sInRpdGxlIjoiMyI
sImxhbmd1YWdlIjoiZW4tVVMiLCJkYXRlRm9ybWF0IjoiTU0vZGQveXl5eSIsImNyZWF0ZWRBdCI6IjIwMjAtMDItMjhUMD
U6Mjg6NTcuNjI0WiIsImlzUmVtZW1iZXIiOmZhbHNlLCJpYXQiOjE1ODI5Nzg4NzEsImV4cCI6MTU4Mjk5MzI3MX0.ooVhU
yOZvphGoxBCPUCZPmxgf3fayHqXuUZLIpJt0rI
ETag: W/"7874-CsbyGLcsPYtfloHHNqlHlsFoqtk"
Set-Cookie:
connect.sid=s%3AuozjZfXwbWBCiYLPRUnYlpZZDADLIyIm.HV4C%2BXeRz3uLnE8WmXd5aWDFY%2FZpBbhQIeJARr2Azh
Y; Path=/; HttpOnly
Vary: Accept-Encoding
Content-Length: 30836

{"status":200,"messageCode":"GET_LIST_ALERTS_SUCCESS","message":"Get alerts
success","total":11,"moistureType":"both","data":[{"_id":"5e4f2c0aa6bd6e75c6b256ed","orgId":"5c
498d5ed41d39001102a0ae","locI
...[SNIP]...
},{"_id":"5d8b2d74ad613900112b0f2c","name":"cable
2","type":"7","number":1,"sensorNodes":["2951000014bf:4:1"]}],"moistureType":"emc","thresholds"
:[{"sensorType":"temp","high":22,"low":-
20,"createdBy":"thanhha3105@gmail.com","enabled":true,"createdAt":"2020-02-
29T12:18:41.570Z","updatedAt":"2020-02-
29T12:18:41.570Z"},{"sensorType":"hum","high":100,"low":0,"createdBy":"thanhha3105@gmail.com","
enabled":false,"updatedBy":"thanhha3105@gmail.com","createdAt":"2020-02-
29T12:18:41.570Z","updatedAt":"2020-02-
29T12:18:41.570Z"},{"sensorType":"emc","high":25,"low":5,"createdBy":"thanhha3105@gmail.com","e
nabled":false,"updatedBy":"thanhha3105@gmail.com","createdAt":"2020-02-
29T12:18:41.570Z","updatedAt":"2020-02-
29T12:18:41.570Z"}]},"deviceName":"2951000014bf:4","sensorType":"temp","sensorNodeIndex":0,"sen
sorIndex":4,"sensorValue":22,"state":"NEW","
...[SNIP]...
```

3.5.9. SSL certificate

Risk Rating: **INFORMATION**

Summary

Severity	**Information**
Confidence	**Certain**
Host	**https://api-qa.clientx.net** **https://webqa.clientx.net**

Classification	**CWE-295: Improper Certificate Validation** **CWE-326: Inadequate Encryption Strength** **CWE-327: Use of a Broken or Risky Cryptographic Algorithm**

Issue Background

SSL (or TLS) 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 SSL 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, SSL connections to the server will not provide the full protection for which SSL is designed.

The server presented the following certificates:

Server certificate

Issued to: *.clientx.net, clientx.net
Issued by: Sectigo RSA Domain Validation Secure Server CA
Valid from: Tue Oct 22 05:00:00 PKT 2019
Valid to: Thu Oct 28 04:59:59 PKT 2021

Certificate chain #1

Issued to: Sectigo RSA Domain Validation Secure Server CA
Issued by: USERTrust RSA Certification Authority
Valid from: Fri Nov 02 05:00:00 PKT 2018
Valid to: Wed Jan 01 04:59:59 PKT 2031

Certificate chain #2

Issued to: USERTrust RSA Certification Authority
Issued by: AddTrust External CA Root
Valid from: Tue May 30 15:48:38 PKT 2000
Valid to: Sat May 30 15:48:38 PKT 2020

Certificate chain #3

Issued to: AddTrust External CA Root
Issued by: AddTrust External CA Root
Valid from: Tue May 30 15:48:38 PKT 2000
Valid to: Sat May 30 15:48:38 PKT 2020

Impact

This issue is purely informational.

It should be noted that various attacks exist against SSL 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 SSL connections without user detection even when a valid SSL certificate is used.

References

- SSL/TLS Configuration Guide

Vulnerability Classification

- CWE-295: Improper Certificate Validation
- CWE-326: Inadequate Encryption Strength
- CWE-327: Use of a Broken or Risky Cryptographic Algorithm

3.5.10. Sensitive Information Disclosure in URL

Risk Rating: INFORMATION

Summary

Severity	**Information**
Confidence	**Firm**
Host	**https://api-qa.clientx.net**
Classification	**CWE-200, WASC-13**

Issue Background

The request appeared to contain sensitive information leaked in the URL. This can violate PCI and most organizational compliance policies. You can configure the list of strings for this check to add or remove values specific to your environment.

Impact

Information disclosure is when an application fails to properly protect sensitive and confidential information from parties that are not supposed to have access to the subject matter in normal circumstances. These types of issues are not exploitable in most cases, but are considered as web application security issues because they allow malicious hackers to gather relevant information which can be used later in the attack lifecycle, in order to achieve more than they could if they didn't get access to such information

Issue Remediation

Do not pass sensitive information in URIs.

Vulnerability Classification

- CWE-200
- WASC-13

Request

```
GET https://api-qa.clientx.net/api/users/?userType=loc-admin,org-
observer&orgId=5c498d5ed41d39001102a0ae HTTP/1.1
Connection: keep-alive
Pragma: no-cache
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/80.0.3987.122 Safari/537.36
Cahce-Control: no-cache
Accept: application/json, text/plain, */*
Sec-Fetch-Dest: empty
Content-Type: application/json
x-access-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVlNThhNThjYjQzY2U3MDAxMmU3YmY4MSIsInR5cGUiOiJsb2
MtYWRtaW4iLCJsb2NhdGlvbnMiOlsiNWM0OTkxMjFkNDFkMzkwMDExMDJhMGFmIiwiNWM0OTkyYmNkNDFkMzkwMDExMDJhM
GI5IiwiNWQ4YjJkNzRhZDYxMzkwMDExMmIwZjI4IiwiNWQ5MWNkZmE1ODI3MTMwMDExMjZiZDVkIl0sIm9yZ0lkIjoiNWM0
OThkNWVkNDFkMzkwMDExMDJhMGFlIiwiZW1haWwiOiJheWVzaGEuc2FnaGVlckBubXhnbG9iYWxzb2Z0d2FyZS5jb20iLCJ
uYW1lIjoiQXllc2hhIiwibm90aWZ5RW1haWwiOnRydWUsIm5vdGlmeVNNUyI6ZmFsc2UsInBob25lIjoiMjM0NTY3OCIsIn
```

```
ByZWZzIjpbeyJrZXkiOiJtZWFzdXJlbWVudCIsInZhbHVlIjoiaW1wZXJpYWwifSx7ImtleSI6Imxhc3RBbm5vdW5jZW1lb
nRSZWFkRGF0ZSIsInZhbHVlIjoiMjAyMC0wMi0yOVQwNDo1Mjo1My41NDJaIn1dLCJsYW5ndWFnZSI6ImVuLVVTIiwiZGF0
ZUZvcm1hdCI6Ik1NL2RkL3l5eXkiLCJmaXJzdFRpbWVMb2dpbiI6ZmFsc2UsImNyZWF0ZWRBdCI6IjIwMjAtMDItMjhUMDU
6MzA6NTIuOTY2WiIsImlzUmVtZW1iZXIiOnRydWUsImlhdCI6MTU4Mjk2NDA0NiwiZXhwIjoxNTkwNzQwMDQ2fQ.XDAb9qJ
tSv_pZh2AqUTroaWgx88e0vnzRs9CkWHMUoc
Origin: https://webqa.clientx.net
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Referer: https://webqa.clientx.net/locations
Accept-Language: en-US,en;q=0.9
Host: api-qa.clientx.net
```

Response

```
HTTP/1.1 200 OK
Date: Sat, 29 Feb 2020 08:14:17 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 2051
Connection: keep-alive
X-Powered-By: Express
Access-Control-Allow-Origin: *
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
Access-Control-Expose-Headers: refresh-token
ETag: W/"803-jI61XlN1k78NNh82sWXNk/xDZ0o"
Set-Cookie:
connect.sid=s%3AHBZMuUt6uynGZOdPUuG4M1_x3ekPk5fC.HcYM%2FwcMudg9iVHyfsQmLXGDbUgY2ISRCkGIeHM6Pt0;
Path=/; HttpOnly
Vary: Accept-Encoding

{"status":200,"messageCode":"GET_LIST_USERS_SUCCESS","message":"Get list users
success","total":3,"data":[{"type":"loc-
admin","status":"active","notes":"","locations":["5c499121d41d39001102a0af","5c4992bcd41d390011
02a0b9","5d8b2d74ad613900112b0f28","5d91cdfa582713001126bd5d"],"notifyEmail":true,"dateFormat":
"MM/dd/yyyy","_id":"5e58a58cb43ce70012e7bf81","orgId":"5c498d5ed41d39001102a0ae","email":"secur
ity.engineer@cloudpro.services","statusDate":"1582867852956","createdBy":"ali@clientx.com","tok
ens":[],"prefs":[{"key":"measurement","value":"imperial"},{"key":"lastAnnouncementReadDate","va
lue...[SNIP]...
```

3.5.11. Sensitive Information Disclosure in HTTP Referrer Header

Risk Rating: **INFORMATION**

Summary

Severity	**Information**
Confidence	**Firm**
Host	**https://api-qa.clientx.net**
Classification	**CWE-200, WASC-13**

Issue Background

The HTTP header may have leaked a potentially sensitive parameter to another domain. This can violate PCI and most organizational compliance policies. You can configure the list of strings for this check to add or remove values specific to your environment.

Impact

Information disclosure is when an application fails to properly protect sensitive and confidential information from parties that are not supposed to have access to the subject matter in normal circumstances. These types of issues are not exploitable in most cases, but are considered as web application security issues because they allow malicious

hackers to gather relevant information which can be used later in the attack lifecycle, in order to achieve more than they could if they didn't get access to such information

Issue Remediation

Do not pass sensitive information in URIs.

Vulnerability Classification

- CWE-200
- WASC-13

Request

```
OPTIONS https://api-qa.clientx.net/api/forgot-password HTTP/1.1
Connection: keep-alive
Accept: */*
Access-Control-Request-Method: POST
Access-Control-Request-Headers: content-type
Origin: https://webqa.clientx.net
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Referer: https://webqa.clientx.net/forgot?email=test@gmail.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/80.0.3987.122 Safari/537.36
Accept-Language: en-US,en;q=0.9
Host: api-qa.clientx.net
Content-Length: 0
```

Response

```
HTTP/1.1 204 No Content
Date: Sat, 29 Feb 2020 15:35:11 GMT
Content-Length: 0
Connection: keep-alive
X-Powered-By: Express
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET,HEAD,PUT,PATCH,POST,DELETE
Vary: Access-Control-Request-Headers
Access-Control-Allow-Headers: content-type
```

3.6. Best Practice Risk Finding

3.6.1. SameSite Cookie Not Implemented

Risk Rating: **Best Practice**

Summary

Severity	**Best Practice**
Confidence	**Certain**
Host	**https://clientx.app** **https://webqa.clientx.net**
Classification	**CWE -16, WASC-13**

Issue Background

There are 3 instances of this issue identified on production.

- /sitemap.xml
- /
- /gtag.js

There are 7 instances of this issue identified on QA.

- /assets/images/
- /
- /sitemap.xml
- /sitemap.xml.gz
- /admin/devices/5d52521f611ec7001153740a/
- /admin/
- /admin/devices/

Cookies have been set without the SameSite attribute, which means that the cookie can be sent as a result of a 'cross-site' request. The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.

Issue Remediation

Ensure that the SameSite attribute is set to either 'lax' or ideally 'strict' for all cookies.

References

- Using the Same-Site Cookies Attribute to Prevent CSRF Attacks
- Same-site Cookies
- Preventing CSRF with the same-site cookie attribute

Vulnerability Classification

- CWE -16
- WASC-13

3.6.1.1. https://clientx.app/sitemap.xml

Cookie Details

Identified Cookie: G_ENABLED_IDP

Cookie Source: CustomField_CookieSourceJs

Request

```
GET /sitemap.xml HTTP/1.1
Host: clientx.app
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-us,en;q=0.5
Cache-Control: no-cache
Connection: Keep-Alive
Referer: http://clientx.app/sitemap.xml
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/54.0.2840.99 Safari/537.36
```

Response

```
HTTP/1.1 200 OK
Server: Apache/2.4.41 (Unix)
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Connection: keep-alive
Content-Length: 1578
Last-Modified: Wed, 22 Jan 2020 07:58:11 GMT
Accept-Ranges: bytes
Vary: Accept-Encoding
```

```
Content-Type: text/html
Content-Encoding:
Pragma: no-cache
Date: Sun, 23 Feb 2020 02:15:49 GMT
ETag: "d28-59cb5e2c30ac0-gzip"
Cache-Control: max-age=0, no-cache, no-store, must-revalidate

…
```

3.6.1.2. https://webqa.clientx.net/

Cookie Information

Identified Cookie: G_ENABLED_IDP

Cookie Source: CustomField_CookieSourceJs

Request

```
GET / HTTP/1.1
Host: webqa.clientx.net
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-us,en;q=0.5
Cache-Control: no-cache
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/54.0.2840.99 Safari/537.36
```

Response

```
HTTP/1.1 200 OK
Server: Apache/2.4.41 (Unix)
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Connection: keep-alive
Content-Length: 1585
Last-Modified: Fri, 28 Feb 2020 07:28:20 GMT
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Type: text/html
Content-Encoding:
Pragma: no-cache
Date: Sat, 29 Feb 2020 07:27:12 GMT
ETag: "d1e-59f9dc8236d00-gzip"
Cache-Control: max-age=0, no-cache, no-store, must-revalidate

…
```