astra

www.getastra.com

Security Assessment Report

Automated Full Scan

Tractor Zoom Pro

Tractor Zoom Pro

Report dated **June 28, 2024**

About Astra

Astra is a leading Penetration Testing as a Service (PTaaS) platform which combines continuous automated scanning with on-demand manual pentests by security experts. Astra follows the highest standards for security testing, vulnerability scanning and is an active contributor to industry leading Open source security standards and tools (OWASP WSTG, OWASP ZAP).

The assessment was performed within the predefined scope of this engagement, and its findings and recommendations have been shared with the customer. A penetration test is considered a snapshot in time. The findings and recommendations solely reflect the information gathered during the assessment period and do not account for any subsequent changes or modifications.

Astra IT Inc.
help@getastra.com

2093 Philadelphia Pike 4080,
Claymont, Delaware, 19703,
United States

Table of Contents

Overview

Executive Summary

Astra was engaged by Tractor Zoom Pro to perform a security assessment of **1** target during the period **19th November 2023** to **19th November 2023**. Manual pentest was performed on **0** targets , and automated vulnerability scanning was performed on **1** target.

The testing was performed from a remote attacker's perspective with the following goals:

- To perform automated vulnerability scanning and identify security loopholes, known vulnerabilities, and evaluate effectiveness of existing security controls in the application.
- Recommend technical security best practices to improve security posture of the target applications audited.
- Explain the potential impact of the identified vulnerabilities, such as the extent of data exposure, potential financial losses, or reputational damage that could occur if they were exploited by malicious actors.
- Provide clear and actionable recommendations for addressing the identified vulnerabilities.

A total of **13 vulnerabilities/recommendations** were reported. Out of a score of 10, the highest risk score assigned to a vulnerability was **6.5**, the lowest was **2.3**, and the average score was **4.4**.

The reported vulnerabilities have been found during an automated vulnerability scan, and have not been vetted by Astra's security analysts.

Scope of the Assessment

The assessment was performed within the predefined scope of this engagement as listed below. No assumptions about the application were made.

Type	Name	Scope	Start Grade	Closure Grade
Web App	Tractor Zoom Pro	https://dev.tractorzoompro.com	C	C

Resolution Statistics

Severity	Solved	Unsolved	Help Wanted	Under Review	Accepted Risk	Grand Total
Critical	0	0	0	0	0	0
High	0	2	0	0	0	2
Medium	0	4	0	0	0	4
Low	0	3	0	0	0	3
Info	0	4	0	0	0	4
Grand Total	0	13	0	0	0	13

Overall vulnerability statistics

Scan Details

Assessment Methodology

An automated scan was performed using Astra's cloud-based vulnerability scanner on the targets.

Using the same techniques as sophisticated real-world attackers, the scanner scans for **9300+ vulnerabilities** based on recently found CVEs, and industry standards such as OWASP Web Security Testing Guide (WSTG), OWASP Top 10, OWASP Application Security Verification Standard (ASVS), NIST 800-115 etc.

Scan Authentication

Scan was performed using suitable authenticated test accounts. For this assessment you can see the number of user roles tested in the table below.

Testers have full access to information about the platform being tested. This often includes accounts (including administrative users), and access to discuss functionality with developers during the testing process.

Assessment Duration and Dates

Scan Mode	Target Name	Authentication	Started	Completed
Automated (Full)	Tractor Zoom Pro	1 user	19th Nov 2023	19th Nov 2023

Certificates

No certificates have been issued for the scope covered as per this report. Certificates are issued when 90% or more vulnerabilities have been fixed after a manual pentest. The remaining vulnerabilities have to be of Info or Low severity. They can also be of Medium severity, if they're not immediately fixable.

Vulnerabilities

Overview Table

No.	Target	Title	Severity	Risk Score	Status	Link
1	Tractor Zoom Pro	Google API Key Disclosed	High	6.5	Unsolved	Open
2	Tractor Zoom Pro	Session Token Found in url	High	6.5	Unsolved	Open
3	Tractor Zoom Pro	CORS Misconfiguration	Medium	4.9	Unsolved	Open
4	Tractor Zoom Pro	Clickjacking possible due to missing X-Frame-Options header	Medium	4.7	Unsolved	Open
5	Tractor Zoom Pro	Content Security Policy (CSP) Header Not Set	Medium	4.5	Unsolved	Open
6	Tractor Zoom Pro	Session ID in URL Rewrite	Medium	4.5	Unsolved	Open
7	Tractor Zoom Pro	DNSSEC not Found	Low	2.5	Unsolved	Open
8	Tractor Zoom Pro	Incomplete or No Cache-control Header Set	Low	2.5	Unsolved	Open
9	Tractor Zoom Pro	Permissions Policy Header Not Set	Low	2.3	Unsolved	Open
10	Tractor Zoom Pro	[Recommendation] Implement Idempotency Header	Info	n/a	Unsolved	Open
11	Tractor Zoom Pro	S3 Bucket URL Disclosed	Info	n/a	Unsolved	Open
12	Tractor Zoom Pro	Retrieved from Cache	Info	n/a	Unsolved	Open
13	Tractor Zoom Pro	Unexpected Content-Type Not Being Rejected	Info	n/a	Unsolved	Open

Details of Vulnerabilities Found

1. Google API Key Disclosed	
Severity	High
Status	Unsolved
Risk Score	6.5/10
CWE	200: Exposure of Sensitive Information to an Unauthorized Actor
CVSS	7 (CVSS:3.1/AV:A/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:L)
Labels	SOC 2 - Confidentiality, SOC 2 - Privacy, SOC 2

Description

Possible Leak of Google API Key in the response body.

Suggested Fix

Make sure that the disclosed key is removed or has sufficient permissions to prevent exploitation.

Findings

Uri :
https://dev.tractorzoompro.com/_next/static/chunks/pages/_app-613e4add5280ca89.js

Uri :
https://dev.tractorzoompro.com/_next/static/chunks/pages/_app-613e4add5280ca89.js

2. Session Token Found in url	
Severity	High
Status	Unsolved
Risk Score	6.5/10
CWE	200: Exposure of Sensitive Information to an Unauthorized Actor
CVSS	5.3 (CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N)
Labels	SOC 2 - Privacy, SOC 2, SOC 2 - Confidentiality

Description

Session Token Found in URL is a security issue where sensitive session information, like login tokens or authentication keys, is included in the web address (URL). This is risky because URLs are visible in the browser's address bar, can be logged, and may appear in web history. If malicious actors get hold of these tokens, they can impersonate the user, potentially gaining unauthorized access to an account

Suggested Fix

The session should be maintained using cookies (or hidden input fields).

Findings

Uri :
https://dev.tractorzoompro.com/stat/www?v=2&tid=G-QF1FJW2NJ9&gtm=45je3ap0v9108028329&_p=245237538&gcd=11l1l1l1l1&cid=1425626291.1700425858&ul=en-us&sr=800x600&tt=internal&uaa=x86&uab=64&uafvl=Not.A%252FBrand%3B8.0.0.0%7CChromium%3B114.0.5735.198%7CHeadlessChrome%3B114.0.5735.198&uamb=0&uam=&uap=Linux&uapv=5.15.109&uaw=0&_eu=CA&_s=1&dp=%2F&sid=1700425857&sct=1&seg=0&dl=https%3A%2F%2Fdev.tractorzoompro.com%2F&dt=Tractor%20Zoom%20Pro&en=page_view&_fv=1&_nsi=1&_ss=2&_ee=1

Uri :
https://dev.tractorzoompro.com/stat/www?v=2&tid=G-QF1FJW2NJ9&gtm=45je3ap0v9108028329&_p=245237538&gcd=11l1l1l1l1&cid=1425626291.1700425858&ul=en-us&sr=800x600&tt=internal&uaa=x86&uab=64&uafvl=Not.A%252FBrand%3B8.0.0.0%7CChromium%3B114.0.5735.198%7CHeadlessChrome%3B114.0.5735.198&uamb=0&uam=&uap=Linux&uapv=5.15.109&uaw=0&_eu=CA&_s=3&dp=%2F&sid=1700425857&sct=1&seg=0&dl=https%3A%2F%2Fdev.tractorzoompro.com%2F&dt=Tractor%20Zoom%20Pro&en=user_engagement&_et=2478

3. CORS Misconfiguration	
Severity	Medium
Status	Unsolved
Risk Score	4.9/10
CWE	942: Permissive Cross-domain Policy with Untrusted Domains
Labels	OWASP 2021 - A01 - Broken Access Control, OWASP 2021, SOC 2 - Security, HIPAA, PCI DSS, ISO 27001

Description

CORS (Cross-Origin Resource Sharing) misconfiguration is a vulnerability that occurs when a web application does not properly configure the CORS policy. CORS is a security mechanism that allows web pages from one domain to access resources from another domain. A misconfigured CORS policy can allow an attacker to access sensitive information or perform unauthorized actions on behalf of a user.

Impact

CORS (Cross-Origin Resource Sharing) misconfiguration can have significant security impacts. A misconfigured CORS policy may allow attackers to access sensitive information from a web application, modify data, or execute unauthorized actions on behalf of legitimate users. The impact of CORS misconfiguration may vary based on the specific vulnerability and context of the web application.

For example, an attacker may be able to steal sensitive information such as user credentials or session tokens by exploiting a misconfigured CORS policy. They could also use a vulnerable web application to launch attacks on other applications or systems, bypassing security measures such as firewalls or authentication mechanisms. In some cases, attackers may also use CORS misconfiguration to execute phishing attacks or deliver malicious content to users.

Steps to Reproduce

Here are some general steps that could be taken to reproduce CORS Misconfiguration:

1. Identify a website that uses cross-origin resource sharing (CORS) to allow requests from other domains.
2. Determine the allowed origins for the website by inspecting the Access-Control-Allow-Origin header in the response of a valid request.
3. Attempt to make a request to the website from a different domain that is not listed in the Access-Control-Allow-Origin header.
4. Observe the response to the request. If the CORS policy is misconfigured, the request may be allowed, and the response may contain sensitive information that should not be accessible from a different origin.
5. Test for other misconfigurations, such as allowing all origins using the wildcard character '*', allowing all methods, or allowing credentials to be sent in cross-origin requests.

Suggested Fix

Here are some suggested fixes for CORS misconfiguration:

1. Limit the Origins: Server administrators should configure the web server to only allow requests from trusted domains. This can be done by specifying the origins that are allowed to access the resources.
2. Configure Headers: Set appropriate headers such as Access-Control-Allow-Origin, Access-Control-Allow-Headers, Access-Control-Allow-Methods, and Access-Control-Allow-Credentials.
3. Use Pre-Flight Requests: Pre-flight requests are used to determine whether a request is safe to send, and the server should respond accordingly.
4. Use Authentication: Authentication can help prevent malicious attacks from untrusted domains. Implement authentication on the server side and add an authentication layer to the application.

Additional References

<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS&gt; <https://portswigger.net/web-security/cors&gt;

Findings

Uri :

https://dev.tractorzoompro.com/stat/www?v=2&tid=G-QF1FJW2NJ9&gtm=45je3ap0v9108028329&_p=245237538&gcd=11l1l1l1l1&cid=1425626291.1700425858&ul=en-us&sr=800x600&tt=internal&uaa=x86&uab=64&uafvl=Not.A%252FBrand%3B8.0.0.0%7CChromium%3B114.0.5735.198%7CHeadlessChrome%3B114.0.5735.198&uamb=0&uam=&uap=Linux&uapv=5.15.109&uaw=0&_eu=CA&_s=1&dp=%2F&sid=1700425857&sct=1&seg=0&dl=https%3A%2F%2Fdev.tractorzoompro.com%2F&dt=Tractor%20Zoom%20Pro&en=page_view&_fv=1&_nsi=1&_ss=2&_ee=1

Attack Reason :

origin: https://XD0uKi8Y.com

Uri :

https://dev.tractorzoompro.com/stat/www?v=2&tid=G-QF1FJW2NJ9&gtm=45je3ap0v9108028329&_p=1006429084&gcd=11l1l1l1l1&cid=1425626291.1700425858&ul=en-us&sr=800x600&tt=internal&uaa=x86&uab=64&uafvl=Not.A%252FBrand%3B8.0.0.0%7CChromium%3B114.0.5735.198%7CHeadlessChrome%3B114.0.5735.198&uamb=0&uam=&uap=Linux&uapv=5.15.109&uaw=0&_eu=CA&_s=1&dp=%2F&sid=1700425857&sct=1&seg=1&dl=https%3A%2F%2Fdev.tractorzoompro.com%2F&dt=Tractor%20Zoom%20Pro&en=page_view&_ee=1

Attack Reason :

origin: https://XD0uKi8Y.com

4. Clickjacking possible due to missing X-Frame-Options header	
Severity	Medium
Status	Unsolved
Risk Score	4.7/10
CWE	1021: Improper Restriction of Rendered UI Layers or Frames
CVSS	6.1 (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N)
Labels	OWASP 2021 - A05 - Security Misconfiguration, OWASP 2021

Description

Clickjacking is a type of attack that tricks users into clicking on a button or link that appears to be legitimate but actually triggers an action on a different website. This attack can be carried out by embedding a website in an iframe on a malicious website, and then using CSS to make the iframe invisible or transparent.

To prevent clickjacking attacks, web developers can use the `X-Frame-Options` header to indicate whether a website can be embedded in an iframe on another domain. If this header is missing, it is possible for an attacker to carry out a clickjacking attack on the vulnerable website.The server didn't return an `X-Frame-Options` header which means that this website could be at risk of a clickjacking attack.

Impact

If an attacker is successful in carrying out a clickjacking attack, they may be able to perform actions on the vulnerable website on behalf of the user, such as making unauthorized purchases or stealing sensitive information.

Suggested Fix

- Modern Web browsers support the `Content-Security-Policy` and `X-Frame-Options` HTTP headers. Ensure one of them is set on all web pages returned by your site/app.
- To mitigate this vulnerability, web developers should add the `X-Frame-Options` header to their website's HTTP response headers. This header can be set to one of three values: `DENY`, `SAMEORIGIN`, or `ALLOW-FROM <URI>`.
    - `DENY` will prevent the website from being embedded in any iframe.
    - `SAMEORIGIN` will allow the website to be embedded only in iframes on the same domain. If you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use `SAMEORIGIN`, otherwise if you never expect the page to be framed, you should use `DENY`.
    - `ALLOW-FROM <URI>` will allow the website to be embedded in iframes on the specified URI. By setting this header, web developers can prevent clickjacking attacks on their website.

  Alternatively consider implementing Content Security Policy's `frame-ancestors` directive.

Additional References

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options&gt;

Findings

Uri :

https://dev.tractorzoompro.com

Uri :

https://dev.tractorzoompro.com/

5. Content Security Policy (CSP) Header Not Set	
Severity	Medium
Status	Unsolved
Risk Score	4.5/10
CWE	693: Protection Mechanism Failure
Labels	OWASP 2021 - A05 - Security Misconfiguration, OWASP 2021

Description

A Content Security Policy (CSP) was not detected.

CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embedded objects such as Java applets, ActiveX, audio and video files.

Impact

CSP helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware.

A content security policy will prevent most script-injection attacks from occurring because it can be set up to limit JavaScript to loading only from trusted places. Leveraging a strict policy can prevent a myriad of issues that stem from loading scripts from unauthorized locations, be it XSS or content injections.

Suggested Fix

Ensure that your web server, application server, load balancer, etc. is configured to set the `Content-Security-Policy` header

Step 1 - Define your CSP

Make a list of policies or directives and source values that state which resources your site will allow or restrict. You can use a wizard such as https://report-uri.com/home/generate to generate a policy. You can refer to some examples at https://content-security-policy.com/examples/

Step 2 - Test your CSP before implementing it

Test out your CSP to make sure you didn't forget to include any trusted domain origins your site needs.

- You can receive alerts of violations to your policy without blocking the content, by setting the HTTP Response header to `Content-Security-Policy-Report-Only`.
- You can also add the directive report-uri with a URL where you would like to see reports about CSP violations. For eg: https://report-uri.com/

Step 3 - Add the Response Header to your server

Apache

If you have an Apache web server, you will define the CSP in the **.htaccess** file of your site, **VirtualHost**, or in **httpd.conf**.

Depending on the directives you chose, it will look something like this:

```
Header set Content-Security-Policy-Report-Only "default-src 'self'; img-src *"
```

Note: **mod_headers** is required to inject headers in Apache. More information at **Apache HTTP Server Tutuorial**.

Nginx

The HTTP response header is modified through the corresponding config files within the **server blocks**. By adding an **[add_header]** directive, you set the response header.

In Nginx, it looks like this:

```
add_header Content-Security-Policy "default-src 'self'; img-src *"
```

Step 4 - Review violations, and disable Report Only mode

It is recommended to set the CSP header in Report Only mode for a few weeks so that no critical functionality is blocked unintentionally. After looking at the logs, update your policy accordingly and disable the Report only mode

Additional References

- <https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy&gt;
- <https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html&gt;
- <http://www.w3.org/TR/CSP&gt; <http://w3c.github.io/webappsec/specs/content-security-policy/csp-specification.dev.html&gt; < http://www.html5rocks.com/en/tutorials/security/content-security-policy&gt;
- <http://caniuse.com/#feat=contentsecuritypolicy&gt; <http://content-security-policy.com&gt;/

Findings

Uri :

https://dev.tractorzoompro.com/robots.txt

Uri :

https://dev.tractorzoompro.com

6. Session ID in URL Rewrite	
Severity	Medium
Status	Unsolved
Risk Score	4.5/10
CWE	200: Exposure of Sensitive Information to an Unauthorized Actor
Labels	OWASP 2021 - A01 - Broken Access Control, OWASP 2021

Description

URL rewrite is used to track user session ID. The session ID may be disclosed via cross-site referer header. In addition, the session ID might be stored in browser history or server logs.

Suggested Fix

For secure content, put session ID in a cookie. To be even more secure consider using a combination of cookie and URL rewrite.

Additional References

<http://seclists.org/lists/webappsec/2002/Oct-Dec/0111.html&gt;

Findings

Uri :
https://dev.tractorzoompro.com/stat/www?v=2&tid=G-QF1FJW2NJ9&gtm=45je3ap0v9108028329&_p=245237538&gcd=11l1l1l1l1&cid=1425626291.1700425858&ul=en-us&sr=800x600&tt=internal&uaa=x86&uab=64&uafvl=Not.A%252FBrand%3B8.0.0.0%7CChromium%3B114.0.5735.198%7CHeadlessChrome%3B114.0.5735.198&uamb=0&uam=&uap=Linux&uapv=5.15.109&uaw=0&_eu=CA&_s=1&dp=%2F&sid=1700425857&sct=1&seg=0&dl=https%3A%2F%2Fdev.tractorzoompro.com%2F&dt=Tractor%20Zoom%20Pro&en=page_view&_fv=1&_nsi=1&_ss=2&_ee=1

Uri :
https://dev.tractorzoompro.com/stat/www?v=2&tid=G-QF1FJW2NJ9&gtm=45je3ap0v9108028329&_p=245237538&gcd=11l1l1l1l1&cid=1425626291.1700425858&ul=en-us&sr=800x600&tt=internal&uaa=x86&uab=64&uafvl=Not.A%252FBrand%3B8.0.0.0%7CChromium%3B114.0.5735.198%7CHeadlessChrome%3B114.0.5735.198&uamb=0&uam=&uap=Linux&uapv=5.15.109&uaw=0&_eu=CEA&_s=2&dp=%2F&sid=1700425857&sct=1&seg=0&dl=https%3A%2F%2Fdev.tractorzoompro.com%2F&dt=Tractor%20Zoom%20Pro&en=scroll&epn.percent_scrolled=90&_et=9

7. DNSSEC not Found	
Severity	Low
Status	Unsolved
Risk Score	2.5/10
CWE	350: Reliance on Reverse DNS Resolution for a Security-Critical Action
CVSS	3.7 (CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:L/A:N)
Labels	OWASP 2021, OWASP 2021 - A05 - Security Misconfiguration

Description

DNSSEC (Domain Name System Security Extensions) is an extension of DNS which provides security against attacks such as packet interception, spoofing, and cache poisoning, potentially compromising all future communications to a host. It uses digital signatures and public key cryptography to provide authentication and integrity to the DNS and helps establish a "chain of trust" with the source of DNS.

Impact

The absence of DNSSEC makes the DNS susceptible and vulnerable to attacks like packet interception, spoofing, Man in the Middle and cache poisoning. As DNS does not support security, establishment of DNSSEC is very important in maintaining integrity and authentication in a name server.

Suggested Fix

- You will have to enable DNSSEC and create the records within your domain/DNS solution. Here is a reference article of the process.
- You can verify your DNSSEC configuration with the third-party DNSViz tool.

Findings

Uri :
https://dev.tractorzoompro.com

8. Incomplete or No Cache-control Header Set	
Severity	Low
Status	Unsolved
Risk Score	2.5/10
CWE	525: Use of Web Browser Cache Containing Sensitive Information
CVSS	3.7 (CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N)

Description

Cache-control is an HTTP header used to specify browser caching policies in both client requests and server responses. Policies include how a resource is cached, where it's cached and its maximum age before expiring (i.e., time to live).

The cache-control header is broken up into directives

Impact

Even after the session has been closed, it might be possible to access the private or sensitive data exchanged within the session through the web browser or proxy cache.

The 'Cache-control' HTTP header holds instructions for caching in both requests and responses. The 'Pragma' header is used for backwards compatibility with HTTP/1.0 where the 'Cache-control' header is not yet presented.

Suggested Fix

Make sure the 'Cache-control' HTTP header is set with 'no-cache, no-store, must-revalidate' and the 'Pragma' header is set to 'no-cache' on HTTP response where possible.

Example: Cache-Control: no-cache, no-store, must-revalidate

Additional References

<https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#web-content-caching&gt; < https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control&gt;

Findings

Uri :
https://dev.tractorzoompro.com
Uri :
https://dev.tractorzoompro.com/

9. Permissions Policy Header Not Set	
Severity	Low
Status	Unsolved
Risk Score	2.3/10
CWE	693: Protection Mechanism Failure
Labels	OWASP 2021 - A01 - Broken Access Control, OWASP 2021

Description

Permissions Policy Header is an added layer of security that helps to restrict from unauthorized access or usage of browser/client features by web resources. This policy ensures the user privacy by limiting or specifying the features of the browsers can be used by the web resources. Permissions Policy provides a set of standard HTTP headers that allow website owners to limit which features of browsers can be used by the page such as camera, microphone, location, full screen etc.

Suggested Fix

Ensure that your web server, application server, load balancer, etc. is configured to set the Permissions-Policy header.

Additional References

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Feature-Policy&gt; <
https://developers.google.com/web/updates/2018/06/feature-policy&gt; <https://scotthelme.co.uk/a-new-security-header-feature-policy&gt; <
https://w3c.github.io/webappsec-feature-policy&gt; <https://www.smashingmagazine.com/2018/12/feature-policy&gt;

Findings

Uri :
https://dev.tractorzoompro.com/robots.txt

Uri :
https://dev.tractorzoompro.com

10. [Recommendation] Implement Idempotency Header	
Severity	Info
Status	Unsolved
Risk Score	0/10
CWE	0: Recommendation

Description

The Idempotency Header is a security measure that is used to prevent duplicate or unauthorized requests from being processed by a web server or API. When a request is made with the Idempotency Header, the server checks to see if the request has already been processed. If it has, the server will not process the request again and will instead return the response from the original request.

HTTP Methods OPTIONS, HEAD, GET, PUT and DELETE are natively idempotent. It is recommended to use custom Idempotency Header for HTTP PATCH and POST methods as well.

Impact

- Failure to implement the Idempotency Header can result in multiple requests being processed, which can lead to unexpected behavior or errors in the application.
- In addition, it can make the application vulnerable to denial-of-service (DoS) attacks, where an attacker repeatedly sends the same request in an attempt to exhaust the server's resources.
- Idempotency is important in building a fault-tolerant HTTP API. For many use cases of HTTP API, duplicate resources are a severe problem from a business perspective.

Suggested Fix

- To ensure the security of their applications, developers should always implement the Idempotency Header when designing APIs or web applications that accept requests from users or other systems.
- The idempotency key that is supplied as part of every POST request MUST be unique and MUST not be reused with another request with a different request payload.
- Uniqueness of the key MUST be defined by the resource owner and MUST be implemented by the clients of the resource server. It is RECOMMENDED that UUID [RFC4122] or a similar random identifier be used as an idempotency key.

Additional References

ietf.org

Findings

Uri :

https://dev.tractorzoompro.com/stat/www?v=2&tid=G-QF1FJW2NJ9&gtm=45je3ap0v9108028329&_p=245237538&gcd=11l1l1l1l1&cid=1425626291.1700425858&ul=en-us&sr=800x600&tt=internal&uaa=x86&uab=64&uafvl=Not.A%252FBrand%3B8.0.0.0%7CChromium%3B114.0.5735.198%7CHeadlessChrome%3B114.0.5735.198&uamb=0&uam=&uap=Linux&uapv=5.15.109&uaw=0&_eu=CA&_s=3&dp=%2F&sid=1700425857&sct=1&seg=0&dl=https%3A%2F%2Fdev.tractorzoompro.com%2F&dt=Tractor%20Zoom%20Pro&en=user_engagement&_et=2478

Uri :

https://dev.tractorzoompro.com/_vercel/insights/view

11. S3 Bucket URL Disclosed	
Severity	Info
Status	Unsolved
Risk Score	0/10
CWE	200: Exposure of Sensitive Information to an Unauthorized Actor
CVSS	7 (CVSS:3.1/AV:A/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:L)
Labels	SOC 2 - Privacy, SOC 2, SOC 2 - Confidentiality

Description

Possible Leak of S3 Bucket URL. These URLs often contain sensitive data and, if accessed by unauthorized individuals, can lead to data breaches and privacy violations. Please confirm that it is not publicly readable or writable.

Suggested Fix

Make sure that the disclosed key is removed or has sufficient permissions to prevent exploitation.

Findings

Uri :
https://dev.tractorzoompro.com/_next/static/chunks/pages/_app-613e4add5280ca89.js

Uri :
https://dev.tractorzoompro.com/_next/static/chunks/pages/_app-613e4add5280ca89.js

12. Retrieved from Cache	
Severity	Info
Status	Unsolved
Risk Score	0/10
CWE	524: Use of Cache Containing Sensitive Information
CVSS	4.6 (CVSS:3.1/AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N)

Description

The content was retrieved from a shared cache. If the response data is sensitive, personal or user-specific, this may result in sensitive information being leaked. In some cases, this may even result in a user gaining complete control of the session of another user, depending on the configuration of the caching components in use in their environment.

This is primarily an issue where caching servers such as "proxy" caches are configured on the local network. This configuration is typically found in corporate or educational environments, for instance.

Impact

- An attacker exploiting the Retrieved From Cache vulnerability can potentially access sensitive information such as login credentials, session tokens, or other personal information. This information can be used to carry out identity theft, financial fraud, or other malicious activities.
- If caching is not set up correctly, then the browser will not be able to validate the cached content and the page may load outdated content, which can have a negative impact on the user experience.
- Caching entities include third-party proxy servers shared by multiple users which are always at risk of being compromised. If caching servers are hacked then it impacts all users connected to this server.

Suggested Fix

Validate that the response does not contain sensitive, personal or user-specific information. If it does, consider the use of the following HTTP response headers, to limit, or prevent the content being stored and retrieved from the cache by another user: Cache-Control: no-cache, no-store, must-revalidate, private Pragma: no-cache Expires: 0 This configuration directs both HTTP 1.0 and HTTP 1.1 compliant caching servers to not store the response, and to not retrieve the response (without validation) from the cache, in response to a similar request.

Additional References

rfc-editor.org/7234

rfc-editor.org/7231

w3.org (obsoleted by rfc7234)

Findings

Uri :
https://dev.tractorzoompro.com/robots.txt

Uri :
https://dev.tractorzoompro.com

13. Unexpected Content-Type Not Being Rejected	
Severity	Info
Status	Unsolved
Risk Score	0/10
CWE	1349: Vulnerability
Labels	OWASP 2021 - A05 - Security Misconfiguration, OWASP 2021, OWASP 2021 - A06 - Vulnerable and Outdated Components

Description

A content-sniffing vulnerability has been identified in the web application. This vulnerability arises from the server's failure to properly reject requests with unexpected or arbitrary Content-Type values. Content-sniffing occurs when a web browser attempts to determine the type and character encoding of a response by analyzing its content, instead of relying solely on the declared Content-Type header. By allowing requests with random or nonsensical Content-Type values to proceed, an attacker can exploit content-sniffing to manipulate how the browser interprets the content. This can lead to various security risks, such as cross-site scripting (XSS) attacks or data injection.

Suggested Fix

Implement robust server-side validation mechanisms to only accept valid and expected Content-Type values. Reject requests with unexpected or nonsensical Content-Type headers.

Additional References

<https://www.keycdn.com/support/what-is-mime-sniffing&gt;

Findings

Uri :
https://dev.tractorzoompro.com/_vercel/insights
Uri :
https://dev.tractorzoompro.com/_vercel/insights
Attack Reason :
false/type

Appendix

APPENDIX A — MEASUREMENT SCALES

Astra determines severity ratings using in-house expertise and industry-standard rating methodologies such as the Open Web Application Security Project (OWASP) and the Common Vulnerability Scoring System (CVSS).

The severity of each finding in this report was determined independently of the severity of other findings. Vulnerabilities assigned a higher severity have more significant technical and business impact and achieve that impact through fewer dependencies on other flaws.

Critical: Vulnerability is an otherwise high-severity issue with additional security implications that could lead to exceptional business impact. Findings are marked as critical severity to communicate an exigent need for immediate remediation. Examples include threats to human safety, permanent loss or compromise of business-critical data, and evidence of prior compromise.

High: Vulnerability introduces significant technical risk to the system that is not contingent on other issues being present to exploit. Examples include creating a breach in the confidentiality or integrity of sensitive business data, customer information, or administrative and user accounts.

Medium: Vulnerability does not in isolation lead directly to the exposure of sensitive business data. However, it can be leveraged in conjunction with another issue to expose business risk. Examples include insecurely storing user credentials, transmitting sensitive data unencrypted, and improper network segmentation.

Low: Vulnerability may result in limited risk or require the presence of multiple additional vulnerabilities to become exploitable. Examples include overly verbose error messages, insecure TLS configurations, and detailed banner information disclosure.

Informational: Finding does not have a direct security impact but represents an opportunity to add an additional layer of security, is a deviation from best practices, or is a security-relevant observation that may lead to exploitable vulnerabilities in the future. Examples include vulnerable yet unused source code and missing HTTP security headers.

APPENDIX B - RESOLUTION STATUS

Unsolved: This status indicates that the security team has reported an issue or vulnerability to the customer, but it is yet to be resolved by the customer. Further actions are required to address the reported security concern.

Under Review: This status is assigned when the customer has fixed the reported issue or vulnerability. The security team will now evaluate and validate the fix to ensure it has been implemented correctly and effectively mitigates the identified risk.

Accepted Risk: This status reflects the customer's decision not to address or resolve the reported issue or vulnerability. By accepting the associated risk, the customer has chosen not to pursue any further action in mitigating the identified security concern.

Help Wanted: When assigned this status, it indicates that the customer has requested additional clarification or assistance from the security team. This could involve seeking further guidance, recommendations, or expertise to better understand and address the reported security issue.

Solved: This status signifies that the customer has successfully resolved the reported vulnerability, and it has been verified by the security team. The necessary measures have been taken to mitigate the risk, ensuring that the identified security concern is no longer present.

APPENDIX C — RISK SCORE

Security recommendations/long-term tasks are marked as "Unsolved" or "Accepted Risk". Astra has evaluated the risk based on information provided and has provided feedback on a case-to-case basis as requested.

Security grades are assigned for vulnerabilities needing immediate attention and does not include security best practices, although reported. For each vulnerability, a risk score is assigned which signifies real world possibility of an attack being orchestrated using the vulnerability. The risk score is calculated using a correlation of multiple factors including potential loss value of a vulnerability, CVSS score, historic data about attacks performed using similar vulnerability & severity of such vulnerabilities assigned by our security engineers in the past.

APPENDIX D — TEST CASES

The following lists of tests are suggestive & not limited to the ones listed. Most importantly, every test case has multiple sub-test cases ranging from a few to sometimes 1000+ sub tests. Additional test cases will be performed based on factors such as:

1. Business Logic
2. Technology Stack
3. Framework/CMS/APIs
4. Application specific features

OWASP Top 10

#	For Applications
1	Broken Access Control
2	Cryptographic Failures
3	Injection
4	Insecure Design
5	Security Misconfiguration
6	Vulnerable and Outdated Components
7	Identification and Authentication Failures
8	Software and Data Integrity Failures
9	Security Logging and Monitoring Failuresies
10	Server-Side Request Forgery

SANS 25 Software Errors/Tests

#	SANS 25
1	Improper Restriction of Operations within the Bounds of a Memory Buffer
2	Improper Neutralization of Input During Web Page Generation ('XSS')
3	Improper Input Validation
4	Information Exposure
5	Out-of-bounds Read
6	Improper Neutralization of Special Elements used in an SQL Command (SQLi)
7	Use After Free
8	Integer Overflow or Wraparound
9	Cross-Site Request Forgery (CSRF)ies
10	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
11	Improper Neutralization of Special Elements used in an OS Command
12	Out-of-bounds Write
13	Improper Authentication
14	NULL Pointer Dereference
15	Incorrect Permission Assignment for Critical Resource
16	Unrestricted Upload of File with Dangerous Type
17	Improper Restriction of XML External Entity Reference
18	Improper Control of Generation of Code ('Code Injection')
19	Use of Hard-coded Credentials
20	Uncontrolled Resource Consumption
21	Missing Release of Resource after Effective Lifetime
22	Untrusted Search Path
23	Deserialization of Untrusted Data
24	Improper Privilege Management
25	Improper Certificate Validation

174 Other Test Cases

#	Test Performed	Typical Severity
1	OS Command Injection	High
2	SQL Injection (Second Order)	High
3	XML External Entity Injection	High
4	LDAP Injection	High
5	XPath Injection	High
6	XML Injection	High
7	ASP.NET Debugging Enabled	High
8	DoS Locking Customer Accounts	Medium
9	DoS Buffer Overflows	Medium
10	Storing too much data in session (DoS)	High
11	Writing user-provided data to disk (DoS)	High
12	HTTP Insecure methods available on Server	High
13	Out of band resource load (HTTP)	High
14	File path manipulation	High
15	Server-site JavaScript code injection	High
16	Perl code injection	High
17	Ruby code injection	High
18	Python code injection	High
19	Expression Language injection	High
20	Unidentified code injection	High
21	Server-side template injection	High
22	SSL injection	High
23	Stored XSS	High
24	HTTP response header injection	High
25	Reflected XSS	High
26	Client-side template injection	High
27	DOM-based XSS	High
28	Reflected DOM-based XSS	High
29	Stored DOM-based XSS	High
30	DOM-based JavaScript Injection	High
31	Reflected DOM-based JavaScript Injection	High
32	Stored DOM-based JavaScript Injection	High
33	Path-relative style sheet import	Information
34	Client-side SQLi (DOM-based)	High
35	Client-side SQLi (Reflected DOM-based)	High
36	Client-side SQLi (Stored DOM-based)	High
37	WebSocket Hijacking (DOM-based)	High
38	WebSocket Hijacking (Reflected DOM-based)	High
39	WebSocket Hijacking (Stored DOM-based)	High
40	Local Path Manipulation (DOM-based)	High
41	Local Path Manipulation (Reflected DOM)	High
42	Local Path Manipulation (Stored DOM-based)	High
43	Client-side XPATH Injection (DOM-based)	Low
44	Client-side XPATH Injection (Reflected DOM)	Low
45	Client-side XPATH Injection (Stored DOM)	Low

#	Test Performed	Typical Severity
46	Client-side JSON Injection (DOM-based)	Low
47	Client-side JSON Injection (Reflected DOM)	Low
48	Client-side JSON Injection (Stored DOM-based)	Low
49	Flash cross-domain policy	High
50	Cross-origin resource sharing	
51	Cross-origin resource sharing (arbitrary)	High
52	Cross-origin resource sharing (encrypted)	Low
53	Cross-origin resource sharing (all sub-domains)	Low
54	Cross-site Request Forgery (CSRF)	Medium
55	SMTP header injection	Medium
56	Cleartext submission of password	High
57	External service interaction (DNS)	High
58	External service interaction (HTTP)	High
59	External service interaction (SMTP)	Information
60	Referrer dependent response	Information
61	Spoofable client IP address	Information
62	User-agent dependent response	Information
63	Password returned in a later response	Medium
64	Password submitted using GET method	Low
65	Password returned in URL query string	Low
66	SQL statement in request parameter	Medium
67	Cross-domain POST	Information
68	ASP.NET ViewState without MAC Enabled	
69	XML entity expansion	Medium
70	Long redirection response	Information
71	Serialized object in HTTP message	
72	Duplicate cookies set	Information
73	WebSocket Hijacking (DOM-based)	High
74	WebSocket Hijacking (Reflected DOM-based)	High
75	WebSocket Hijacking (Stored DOM-based)	High
76	Local Path Manipulation (DOM-based)	High
77	Local Path Manipulation (Reflected DOM)	High
78	Local Path Manipulation (Stored DOM-based)	High
79	Client-side XPATH Injection (DOM-based)	Low
80	Client-side XPATH Injection (Reflected DOM)	Low
81	Client-side XPATH Injection (Stored DOM)	Low
82	Client-side JSON Injection (DOM-based)	Low
83	Client-side JSON Injection (Reflected DOM)	Low
84	Client-side JSON Injection (Stored DOM-based)	Low
85	Flash cross-domain policy	High
86	Cross-origin resource sharing	
87	Cross-origin resource sharing (arbitrary)	High
88	Cross-origin resource sharing (encrypted)	Low
89	Cross-origin resource sharing (all sub-domains)	Low
90	Cross-site Request Forgery (CSRF)	Medium
91	SMTP header injection	Medium
92	Cleartext submission of password	High
93	External service interaction (DNS)	High

#	Test Performed	Typical Severity
94	External service interaction (HTTP)	High
95	External service interaction (SMTP)	Information
96	Referrer dependent response	Information
97	Spoofable client IP address	Information
98	User-agent dependent response	Information
99	Password returned in a later response	Medium
100	Password submitted using GET method	Low
101	Password returned in URL query string	Low
102	SQL statement in request parameter	Medium
103	Cross-domain POST	Information
104	ASP.NET ViewState without MAC Enabled	
105	XML entity expansion	Medium
106	Long redirection response	Information
107	Serialized object in HTTP message	
108	Duplicate cookies set	Information
109	Input returned in response (stored)	Information
110	Input returned in response (reflected)	Information
111	Suspicious input transformation (reflected)	Information
112	Suspicious input transformation (stored)	Information
113	Open redirection (stored)	Low
114	Open redirection (reflected)	Medium
115	Open redirection (DOM-based)	Low
116	Open redirection (Stored DOM-based)	Low
117	Open redirection (Reflected DOM-based)	Medium
118	SSL cookie without secure flag set	Medium
119	Cookie scoped to parent domain	Low
120	Cross-domain referrer leakage	Information
121	Cross-domain script include	Information
122	Cookie without HTTPOnly flag set	
123	Session token in URL	
124	Password field with autocomplete enabled	
125	Password value set in cookie	Medium
126	Browser cross-site scripting disabled	Information
127	HTTP TRACE method is enabled	Information
128	Cookie manipulation (DOM-based)	Low
129	Cookie manipulation (reflected DOM-based)	Low
130	Cookie manipulation (DOM-based)	Low
131	Ajax request header manipulation (DOM-based)	Low
132	Ajax request header manipulation (reflected)	Low
133	Ajax request header manipulation (stored DOM)	Low
134	Denial of service (DOM-based)	Information
135	Denial of service (reflected DOM-based)	Information
136	Denial of service (stored DOM-based)	Low
137	HTML5 web message manipulation DOM-based	Information
138	HTML5 web message manipulation (reflected)	Information
139	HTML5 web message manipulation (stored DOM)	Information
140	HTML5 storage manipulation (DOM-based)	Information
141	HTML5 storage manipulation (reflected DOM)	Information

#	Test Performed	Typical Severity
142	HTML5 storage manipulation (stored DOM)	Information
143	Link manipulation (DOM-based)	Low
144	Link manipulation (reflected DOM-based)	Low
145	Link manipulation (stored DOM-based)	Low
146	Link manipulation (reflected & stored)	Information
147	Document domain manipulation (DOM-based)	Medium
148	Document domain manipulation reflected DOM	Medium
149	Document domain manipulation (stored DOM)	Medium
150	DOM data manipulation (DOM-based)	Information
151	CSS Injection (reflected & stored)	Medium
152	Client-side HTTP parameter pollution (reflected)	Low
153	Client-side HTTP parameter pollution (Stored)	Low
154	Form action hijacking (reflected)	Medium
155	Form action hijacking (stored)	Medium
156	Database connection string disclosed	Medium
157	Source code disclosure	
158	Directory listing	Information
159	Email addresses disclosed	Information
160	Private IP addresses disclosed	Information
161	Social security numbers disclosed	Information
162	Credit card numbers disclosed	Information
163	Private key disclosed	Information
164	Cacheable HTTPS response	Information
165	Base64 encoded data in parameter	Information
166	Multiple content types specified	Information
167	HTML does not specify charset	Information
168	HTML uses unrecognized charset	Information
169	Content type incorrectly stated	Low
170	Content type is not specified	Information
171	SSL certificate	Medium
172	Unencrypted communications	Low
173	Strict transport security not enforced	Low
174	Mixed content	Low