Bug Bounty Report

Gunasekara D T | IT23621138

IE2062 – Web Security

B.Sc. (Hons) in Information Technology specializing Cyber Security

Table of Contents

1. <u>Executive Summary</u>

This security assessment was carried out through Supabase's authorized bug bounty program on the HackerOne platform. The focus of the review was to evaluate important areas such as user login processes, how sessions are managed once users are logged in, and the overall security setup of the platform. The goal was to identify any weaknesses that could put user data or system reliability at risk.

The assessment highlighted three areas that would benefit attention:

1. **Authentication Process** – Parts of the system could potentially be misused under certain conditions, which may allow unauthorized requests.
2. **Outdated Software Component** – One of the software libraries used by the platform was found to be outdated, meaning it may contain known issues that attackers might already be aware of.
3. **Security Settings in Web Communication** – Some additional web security settings could be applied to the platform for further protection of users from specific types of online attacks.

Overall, Supabase maintains a solid level of security, supported by strong existing controls and a clear commitment to protecting its platform. The findings are not critical, but they do represent opportunities to improve.

By updating software components regularly, strengthening existing protections, and enhancing web security settings, Supabase can reduce future risks, stay ahead of new threats and build greater trust with its users.

2. Methodology

2.1 Tools and Technologies utilized

- **Kali Linux OS**: since it is pre-equipped with penetration testing tools, making it more suitable for security activities than standard operating systems.

- **BurpSuite Community:** for web application security testing, including traffic interception, request/response manipulation, and vulnerability analysis.

- **Nmap:** for network reconnaissance and service enumeration.

- **Amass:** for DNS enumeration and subdomain discovery.

- **Shodan**: for external, internet-wide reconnaissance; specifically, service enumeration (OSINT) of publicly reachable hosts and services.

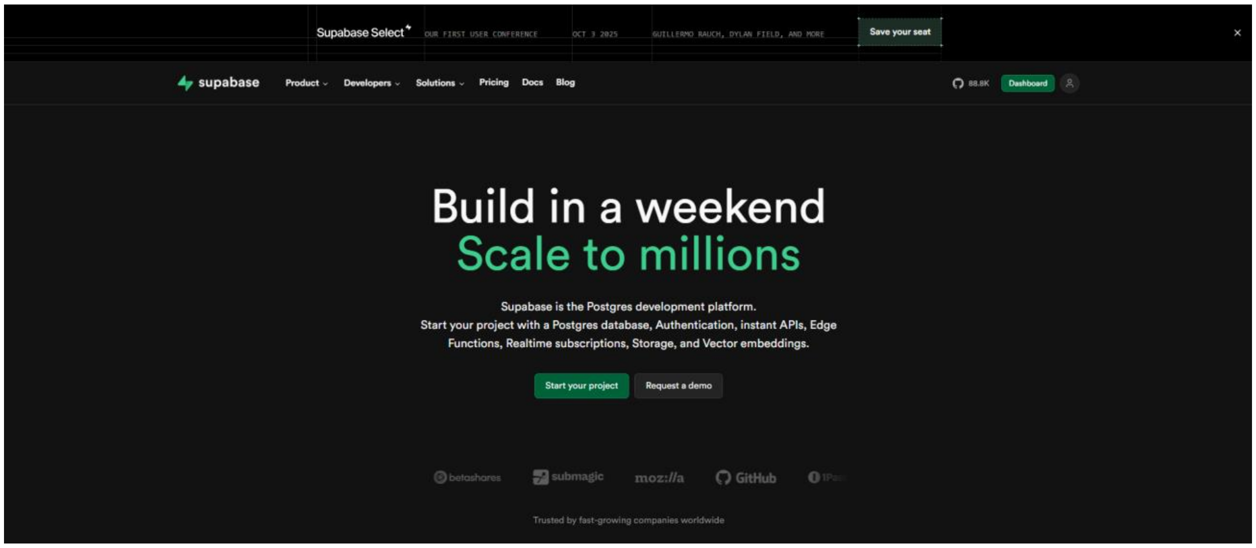
- **Browser Developer tools (Chrome & Mozilla FireFox):** for client-side security analysis and header inspection.

- **Command-line tools:** curl, grep for parsing and header checking.

2.2 Findings through Reconnaissance

The security assessment kicked off with a thorough reconnaissance to understand the platform infrastructure and measure the potential attack surfaces within the defined scope.

Main domain: https://supabase.com/



Build in a weekend
Scale to millions

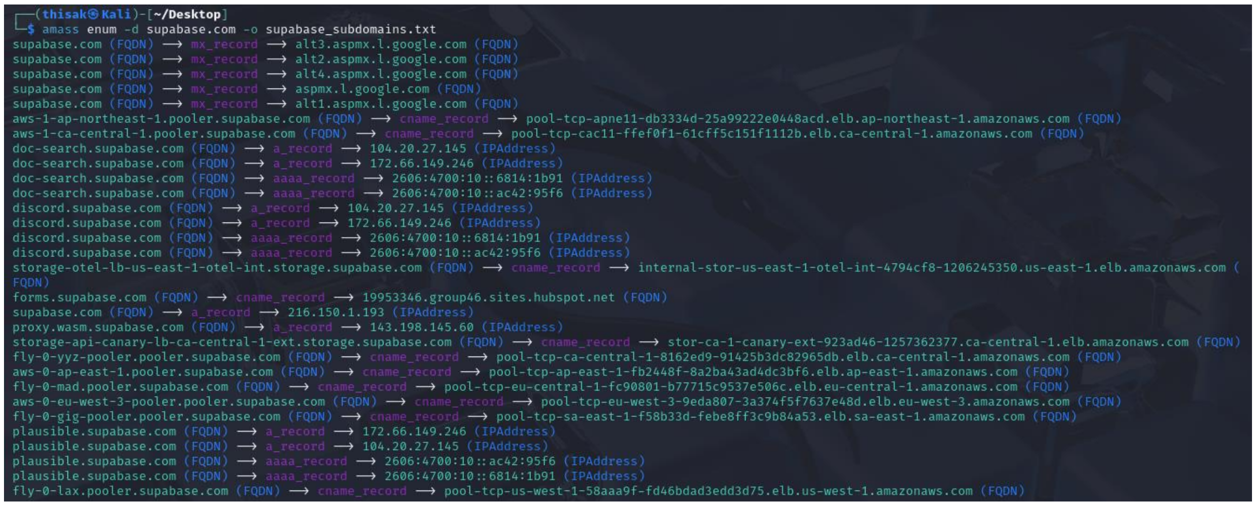
Supabase is the Postgres development platform.
Start your project with a Postgres database, Authentication, instant APIs, Edge Functions, Realtime subscriptions, Storage, and Vector embeddings.

Trusted by fast-growing companies worldwide

Figure 1: Target domain

Subdomain Identification:

Amass was utilized to identify the subdomains related to the target platform.



```
┌──(thisak㉿Kali)-[~/Desktop]
└─$ amass enum -d supabase.com -o supabase_subdomains.txt
supabase.com (FQDN) ⟶ mx_record ⟶ alt3.aspmx.l.google.com (FQDN)
supabase.com (FQDN) ⟶ mx_record ⟶ alt2.aspmx.l.google.com (FQDN)
supabase.com (FQDN) ⟶ mx_record ⟶ alt4.aspmx.l.google.com (FQDN)
supabase.com (FQDN) ⟶ mx_record ⟶ aspmx.l.google.com (FQDN)
supabase.com (FQDN) ⟶ mx_record ⟶ alt1.aspmx.l.google.com (FQDN)
aws-1-ap-northeast-1.pooler.supabase.com (FQDN) ⟶ cname_record ⟶ pool-tcp-apne11-db3334d-25a99222e0448acd.elb.ap-northeast-1.amazonaws.com (FQDN)
aws-1-ca-central-1.pooler.supabase.com (FQDN) ⟶ cname_record ⟶ pool-tcp-cac11-ffef0f1-61cff5c151f1112b.elb.ca-central-1.amazonaws.com (FQDN)
doc-search.supabase.com (FQDN) ⟶ a_record ⟶ 104.20.27.145 (IPAddress)
doc-search.supabase.com (FQDN) ⟶ a_record ⟶ 172.66.149.246 (IPAddress)
doc-search.supabase.com (FQDN) ⟶ aaaa_record ⟶ 2606:4700:10::6814:1b91 (IPAddress)
doc-search.supabase.com (FQDN) ⟶ aaaa_record ⟶ 2606:4700:10::ac42:95f6 (IPAddress)
discord.supabase.com (FQDN) ⟶ a_record ⟶ 104.20.27.145 (IPAddress)
discord.supabase.com (FQDN) ⟶ a_record ⟶ 172.66.149.246 (IPAddress)
discord.supabase.com (FQDN) ⟶ aaaa_record ⟶ 2606:4700:10::6814:1b91 (IPAddress)
discord.supabase.com (FQDN) ⟶ aaaa_record ⟶ 2606:4700:10::ac42:95f6 (IPAddress)
storage-otel-lb-us-east-1-otel-int.storage.supabase.com (FQDN) ⟶ cname_record ⟶ internal-stor-us-east-1-otel-int-4794cf8-1206245350.us-east-1.elb.amazonaws.com (
FQDN)
forms.supabase.com (FQDN) ⟶ cname_record ⟶ 19953346.group46.sites.hubspot.net (FQDN)
supabase.com (FQDN) ⟶ a_record ⟶ 216.150.1.193 (IPAddress)
proxy.wasm.supabase.com (FQDN) ⟶ a_record ⟶ 143.198.145.60 (IPAddress)
storage-api-canary-lb-ca-central-1-ext.storage.supabase.com (FQDN) ⟶ cname_record ⟶ stor-ca-1-canary-ext-923ad46-1257362377.ca-central-1.elb.amazonaws.com (FQDN)
fly-0-yyz-pooler.pooler.supabase.com (FQDN) ⟶ cname_record ⟶ pool-tcp-ca-central-1-8162ed9-91425b3dc82965db.elb.ca-central-1.amazonaws.com (FQDN)
aws-0-ap-east-1.pooler.supabase.com (FQDN) ⟶ cname_record ⟶ pool-tcp-ap-east-1-fb2448f-8a2ba43ad4dc3bf6.elb.ap-east-1.amazonaws.com (FQDN)
fly-0-mad.pooler.supabase.com (FQDN) ⟶ cname_record ⟶ pool-tcp-eu-central-1-fc90801-b77715c9537e506c.elb.eu-central-1.amazonaws.com (FQDN)
aws-0-eu-west-3-pooler.pooler.supabase.com (FQDN) ⟶ cname_record ⟶ pool-tcp-eu-west-3-9eda807-3a374f5f7637e48d.elb.eu-west-3.amazonaws.com (FQDN)
fly-0-gig-pooler.pooler.supabase.com (FQDN) ⟶ cname_record ⟶ pool-tcp-sa-east-1-f58b33d-febe8ff3c9b84a53.elb.sa-east-1.amazonaws.com (FQDN)
plausible.supabase.com (FQDN) ⟶ a_record ⟶ 172.66.149.246 (IPAddress)
plausible.supabase.com (FQDN) ⟶ a_record ⟶ 104.20.27.145 (IPAddress)
plausible.supabase.com (FQDN) ⟶ aaaa_record ⟶ 2606:4700:10::ac42:95f6 (IPAddress)
plausible.supabase.com (FQDN) ⟶ aaaa_record ⟶ 2606:4700:10::6814:1b91 (IPAddress)
fly-0-lax.pooler.supabase.com (FQDN) ⟶ cname_record ⟶ pool-tcp-us-west-1-58aaa9f-fd46bdad3edd3d75.elb.us-west-1.amazonaws.com (FQDN)
```

Figure 2: Amass tool listing subdomains

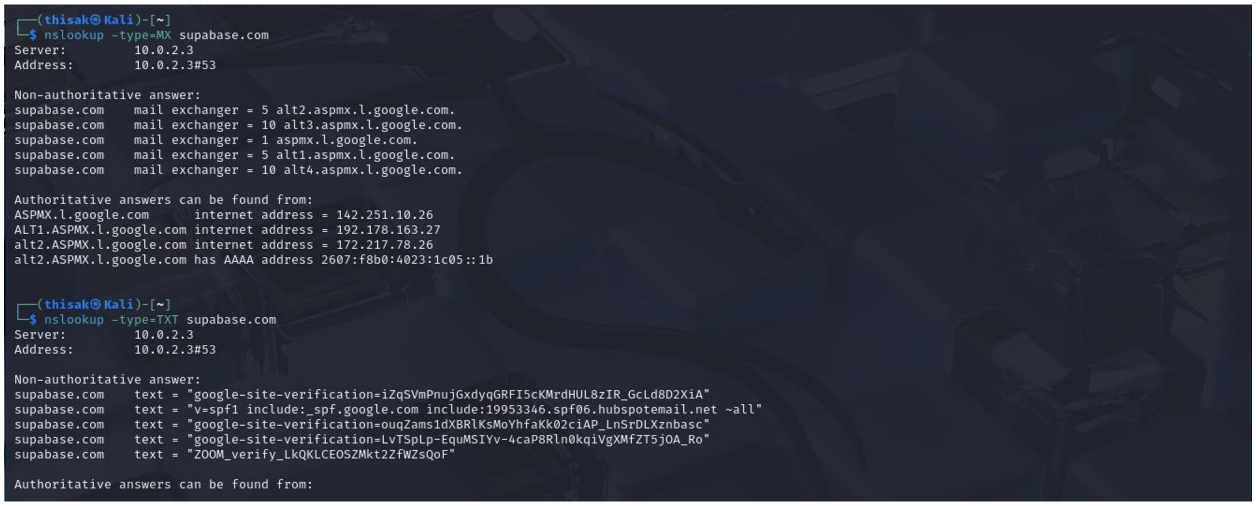
Amass was able to identify several subdomains including:

- **app.supabase.com** – Primary dashboard application
- **api.supabase.com** – main API endpoint
- **docs.supabase.com** – Documentation application
- **forms.supabase.com** – Form handling method

All these subdomains are within the ***.supabase.com** scope, which was avoided during the manual testing, along with ***. supabase.co**, which contains customer domains.

DNS mapping:

nslookup was utilized to recon on DNS and extract infrastructure information.

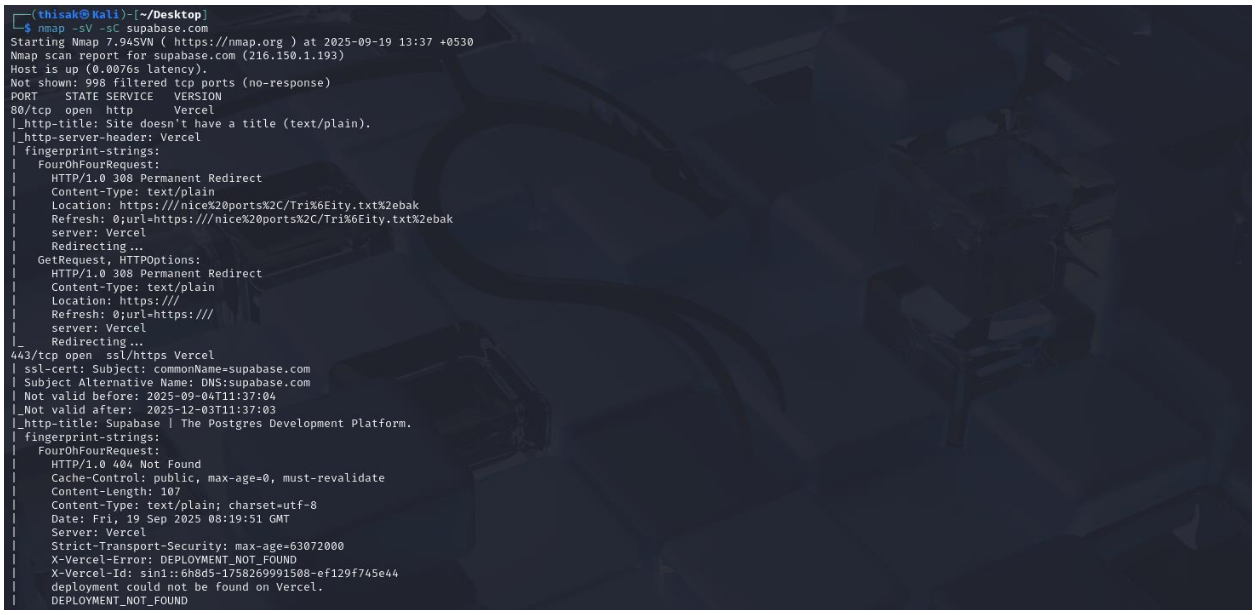


```
┌──(thisak㉿Kali)-[~]
└─$ nslookup -type=MX supabase.com
Server:         10.0.2.3
Address:        10.0.2.3#53

Non-authoritative answer:
supabase.com    mail exchanger = 5 alt2.aspmx.l.google.com.
supabase.com    mail exchanger = 10 alt3.aspmx.l.google.com.
supabase.com    mail exchanger = 1 aspmx.l.google.com.
supabase.com    mail exchanger = 5 alt1.aspmx.l.google.com.
supabase.com    mail exchanger = 10 alt4.aspmx.l.google.com.

Authoritative answers can be found from:
ASPMX.l.google.com      internet address = 142.251.10.26
ALT1.ASPMX.l.google.com internet address = 192.178.163.27
alt2.ASPMX.l.google.com internet address = 172.217.78.26
alt2.ASPMX.l.google.com has AAAA address 2607:f8b0:4023:1c05::1b


┌──(thisak㉿Kali)-[~]
└─$ nslookup -type=TXT supabase.com
Server:         10.0.2.3
Address:        10.0.2.3#53

Non-authoritative answer:
supabase.com    text = "google-site-verification=iZqSVmPnujGxdyqGRFI5cKMrdHUL8zIR_GcLd8D2XiA"
supabase.com    text = "v=spf1 include:_spf.google.com include:19953346.spf06.hubspotemail.net ~all"
supabase.com    text = "google-site-verification=ouqZams1dXBRlKsMoYhfaKk02ciAP_LnSrDLXznbasc"
supabase.com    text = "google-site-verification=LvTSpLp-EquMSIYv-4caP8Rln0kqiVgXMfZT5jOA_Ro"
supabase.com    text = "ZOOM_verify_LkQKLCEOSZMkt2ZfWZsQoF"

Authoritative answers can be found from:
```

Figure 3: nsalookup tool extracting information

Network Scanning:

Nmap was utilized to identify the open ports and running services related to the target platform.

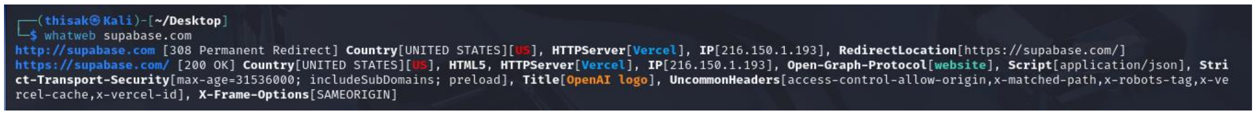


```
┌──(thisak㉿Kali)-[~/Desktop]
└─$ nmap -sV -sC supabase.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-09-19 13:37 +0530
Nmap scan report for supabase.com (216.150.1.193)
Host is up (0.0076s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT    STATE SERVICE   VERSION
80/tcp  open  http      Vercel
|_http-title: Site doesn't have a title (text/plain).
|_http-server-header: Vercel
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.0 308 Permanent Redirect
|     Content-Type: text/plain
|     Location: https:///nice%20ports%2C/Tri%6Eity.txt%2ebak
|     Refresh: 0;url=https:///nice%20ports%2C/Tri%6Eity.txt%2ebak
|     server: Vercel
|     Redirecting...
|   GetRequest, HTTPOptions:
|     HTTP/1.0 308 Permanent Redirect
|     Content-Type: text/plain
|     Location: https:///
|     Refresh: 0;url=https:///
|     server: Vercel
|_    Redirecting...
443/tcp open  ssl/https Vercel
| ssl-cert: Subject: commonName=supabase.com
| Subject Alternative Name: DNS:supabase.com
| Not valid before: 2025-09-04T11:37:04
|_Not valid after:  2025-12-03T11:37:03
|_http-title: Supabase | The Postgres Development Platform.
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.0 404 Not Found
|     Cache-Control: public, max-age=0, must-revalidate
|     Content-Length: 107
|     Content-Type: text/plain; charset=utf-8
|     Date: Fri, 19 Sep 2025 08:19:51 GMT
|     Server: Vercel
|     Strict-Transport-Security: max-age=63072000
|     X-Vercel-Error: DEPLOYMENT_NOT_FOUND
|     X-Vercel-Id: sin1::6h8d5-1758269991508-ef129f745e44
|     deployment could not be found on Vercel.
|     DEPLOYMENT_NOT_FOUND
```

Figure 4: nmap tool in action

Tech Stack Identification:

Whatweb was used to identify the technologies used by the target platform.

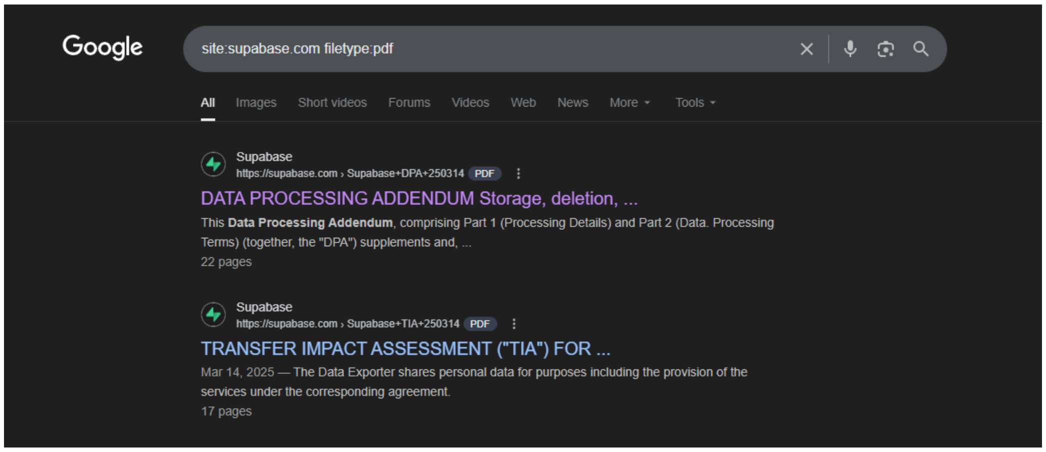


```
┌──(thisak㉿Kali)-[~/Desktop]
└─$ whatweb supabase.com
http://supabase.com [308 Permanent Redirect] Country[UNITED STATES][US], HTTPServer[Vercel], IP[216.150.1.193], RedirectLocation[https://supabase.com/]
https://supabase.com/ [200 OK] Country[UNITED STATES][US], HTML5, HTTPServer[Vercel], IP[216.150.1.193], Open-Graph-Protocol[website], Script[application/json], Stri
ct-Transport-Security[max-age=31536000; includeSubDomains; preload], Title[OpenAI logo], UncommonHeaders[access-control-allow-origin,x-matched-path,x-robots-tag,x-ve
rcel-cache,x-vercel-id], X-Frame-Options[SAMEORIGIN]
```

Figure 5: Whatweb tool listing the tech stack

Discovering publicly disclosed information:

Google Dorking was used to identify any information that standard recon tools might miss.



site:supabase.com filetype:pdf

Supabase
https://supabase.com › Supabase+DPA+250314 PDF

DATA PROCESSING ADDENDUM Storage, deletion, ...

This **Data Processing Addendum**, comprising Part 1 (Processing Details) and Part 2 (Data. Processing Terms) (together, the "DPA") supplements and, ...

22 pages

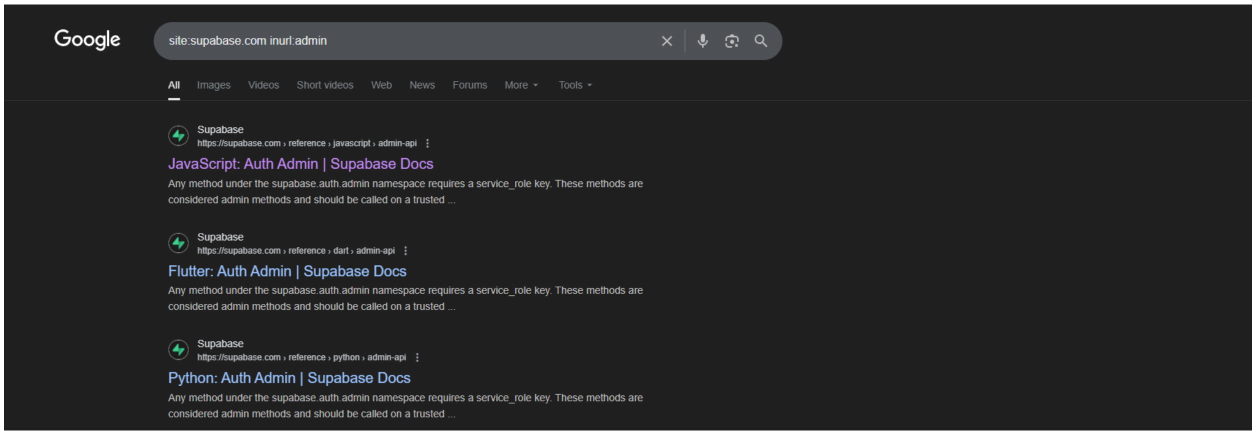
Supabase
https://supabase.com › Supabase+TIA+250314 PDF

TRANSFER IMPACT ASSESSMENT ("TIA") FOR ...

Mar 14, 2025 — The Data Exporter shares personal data for purposes including the provision of the services under the corresponding agreement.

17 pages

Figure 6: Google Dorking to identify pdf documents that are disclosed publicly

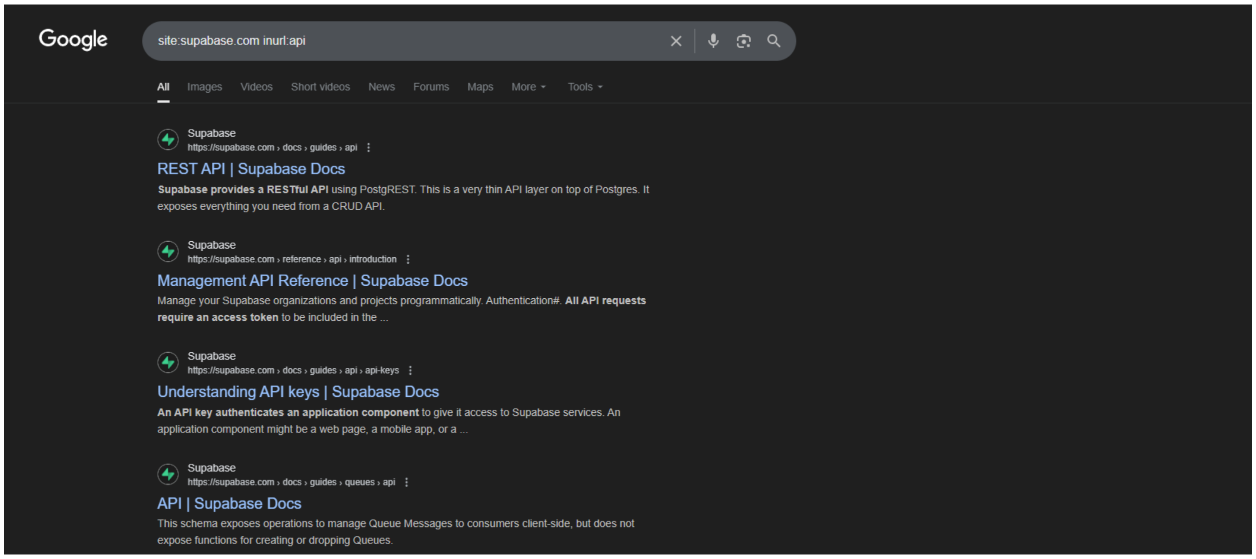


Supabase
https://supabase.com › reference › javascript › admin-api
JavaScript: Auth Admin | Supabase Docs
Any method under the supabase.auth.admin namespace requires a service_role key. These methods are considered admin methods and should be called on a trusted ...

Supabase
https://supabase.com › reference › dart › admin-api
Flutter: Auth Admin | Supabase Docs
Any method under the supabase.auth.admin namespace requires a service_role key. These methods are considered admin methods and should be called on a trusted ...

Supabase
https://supabase.com › reference › python › admin-api
Python: Auth Admin | Supabase Docs
Any method under the supabase.auth.admin namespace requires a service_role key. These methods are considered admin methods and should be called on a trusted ...

Figure 7: Google Dorking to identify any disclosed information that has 'admin' in it



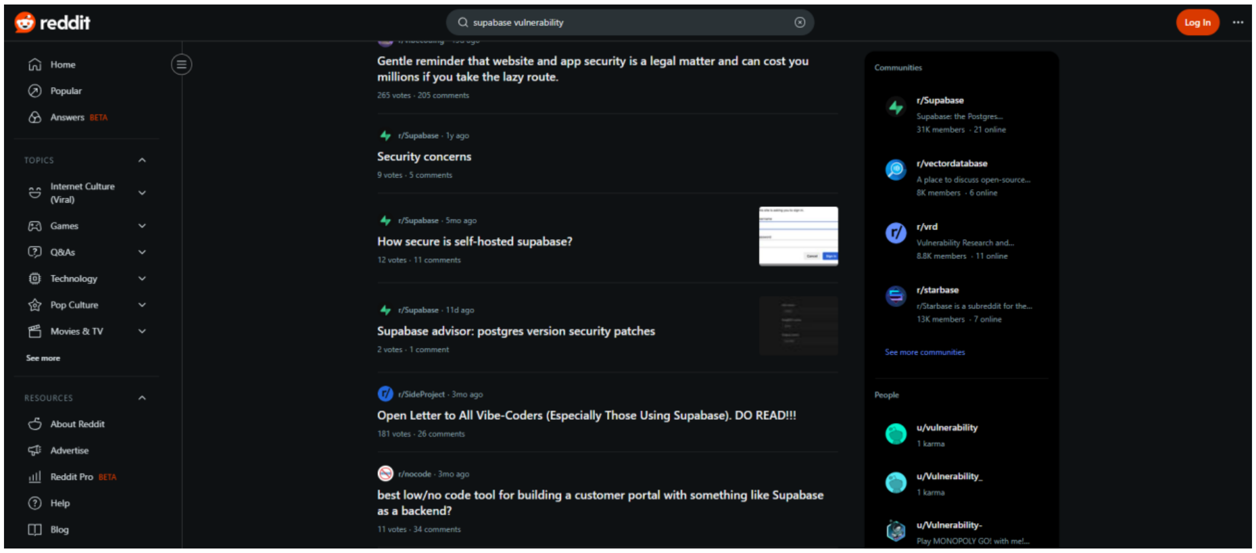
Supabase
https://supabase.com › docs › guides › api
REST API | Supabase Docs
Supabase provides a RESTful API using PostgREST. This is a very thin API layer on top of Postgres. It exposes everything you need from a CRUD API.

Supabase
https://supabase.com › reference › api › introduction
Management API Reference | Supabase Docs
Manage your Supabase organizations and projects programmatically. Authentication#. **All API requests require an access token** to be included in the ...

Supabase
https://supabase.com › docs › guides › api › api-keys
Understanding API keys | Supabase Docs
An API key authenticates an application component to give it access to Supabase services. An application component might be a web page, a mobile app, or a ...

Supabase
https://supabase.com › docs › guides › queues › api
API | Supabase Docs
This schema exposes operations to manage Queue Messages to consumers client-side, but does not expose functions for creating or dropping Queues.

Figure 8: Google Dorking to identify any disclosed information that has 'api' in it



Gentle reminder that website and app security is a legal matter and can cost you millions if you take the lazy route.
265 votes · 205 comments

r/Supabase · 1y ago
Security concerns
9 votes · 5 comments

r/Supabase · 5mo ago
How secure is self-hosted supabase?
12 votes · 11 comments

r/Supabase · 11d ago
Supabase advisor: postgres version security patches
2 votes · 1 comment

r/SideProject · 3mo ago
Open Letter to All Vibe-Coders (Especially Those Using Supabase). DO READ!!!
181 votes · 26 comments

r/nocode · 3mo ago
best low/no code tool for building a customer portal with something like Supabase as a backend?
11 votes · 34 comments

Communities

r/Supabase
Supabase: the Postgres...
31K members · 21 online

r/vectordatabase
A place to discuss open-source...
8K members · 6 online

r/vrd
Vulnerability Research and...
8.8K members · 11 online

r/starbase
r/Starbase is a subreddit for the...
13K members · 7 online

See more communities

People

u/vulnerability
1 karma

u/Vulnerability_
1 karma

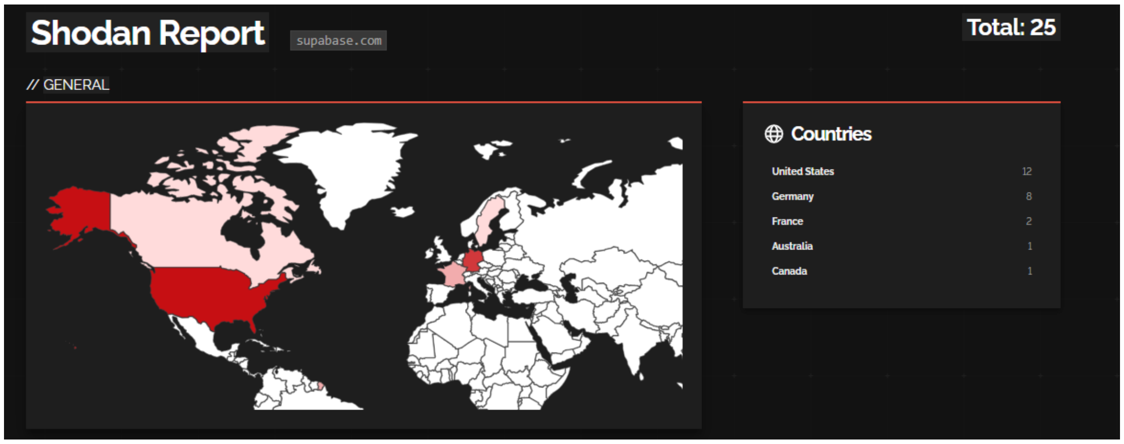
u/Vulnerability-
Play MONOPOLY GO! with me!...

Figure 9: Accessing Reddit to extract any information

/

Service Enumeration (OSINT):

Shodan was implemented to identify any information that can be gathered through OSINT.



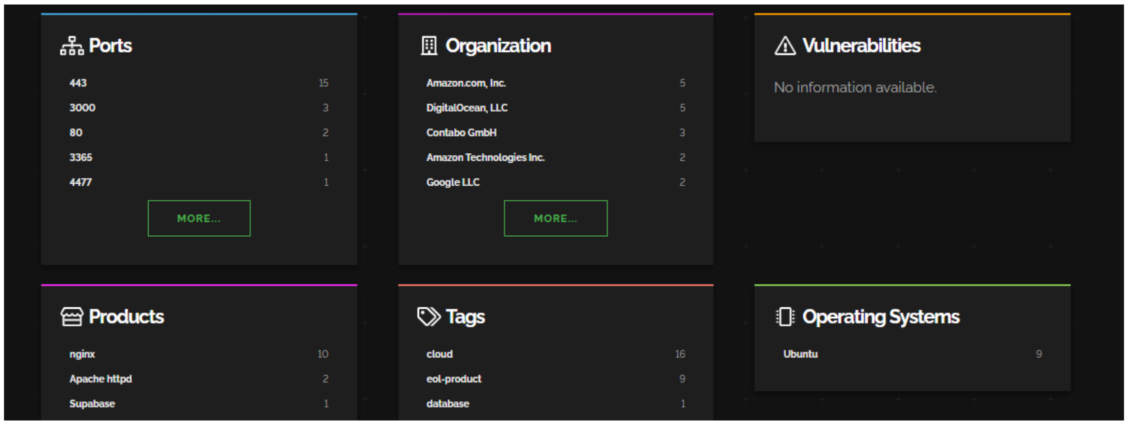
Shodan Report supabase.com — Total: 25

// GENERAL

Countries

United States	12
Germany	8
France	2
Australia	1
Canada	1

Figure 10: Shodan report pt.I



Ports

443	15
3000	3
80	2
3365	1
4477	1

MORE...

Organization

Amazon.com, Inc.	5
DigitalOcean, LLC	5
Contabo GmbH	3
Amazon Technologies Inc.	2
Google LLC	2

MORE...

Vulnerabilities

No information available.

Products

nginx	10
Apache httpd	2
Supabase	1

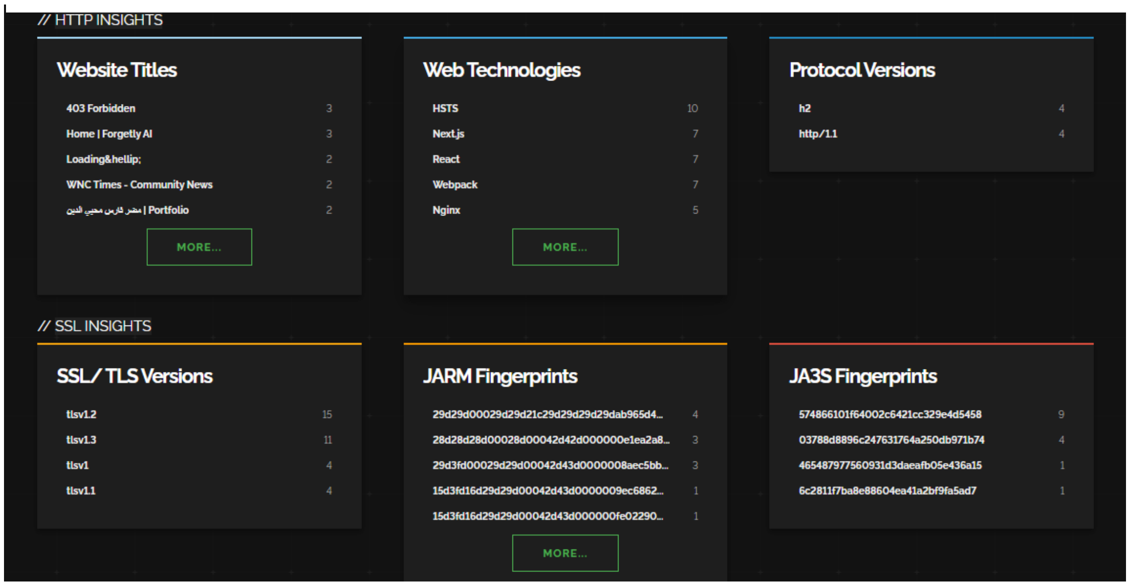
Tags

cloud	16
eol-product	9
database	1

Operating Systems

Ubuntu	9

Figure 11: Shodan report pt.II



// HTTP INSIGHTS

Website Titles

403 Forbidden	3
Home | Forgetly AI	3
Loading&hellip;	2
WNC Times - Community News	2
مضر فارس محيي الدين | Portfolio	2

MORE...

Web Technologies

HSTS	10
Next.js	7
React	7
Webpack	7
Nginx	5

MORE...

Protocol Versions

h2	4
http/1.1	4

// SSL INSIGHTS

SSL/ TLS Versions

tlsv1.2	15
tlsv1.3	11
tlsv1	4
tlsv1.1	4

JARM Fingerprints

29d29d00029d29d21c29d29d29d29dab965d4...	4
28d28d28d00028d00042d42d000000e1ea2a8...	3
29d3fd00029d29d00042d43d0000008aec5bb...	3
15d3fd16d29d29d00042d43d0000009ec6862...	1
15d3fd16d29d29d00042d43d000000fe02290...	1

MORE...

JA3S Fingerprints

574866101f64002c6421cc329e4d5458	9
03788d8896c247631764a250db971b74	4
465487977560931d3daeafb05e436a15	1
6c2811f7ba8e88604ea41a2bf9fa5ad7	1

Figure 12: Shodan report pt.III

2.3 Findings

2.3.1 Finding 01: Absence of ant-CSRF tokens

Vulnerability Title:

Missing CSRF Protection on Authenticated Dashboard Functions.

Description:

Cross-Site Request Forgery (CSRF) occurs when a victim's browser is induced to send authenticated requests to a site the victim is logged into, causing unintended state changes. During testing, authenticated POST endpoints on the dashboard did not include per-request or per-session anti-CSRF tokens.

CSRF attacks are effective when the victim has an active session at the target website, the victim is authenticated via HTTP on the target website, and the victim is on the same local network as the target site. The main affected components in a platform are web forms as they can do state changing actions without any anti-CSRF tokens.

Additionally, some authentication cookies were observed with SameSite=None, increasing the chances that a cross-origin request from a malicious page could be automatically sent by the browser. These factors indicate the application lacks standard CSRF defenses and may be susceptible to CSRF-style state changes under realistic browsing conditions.

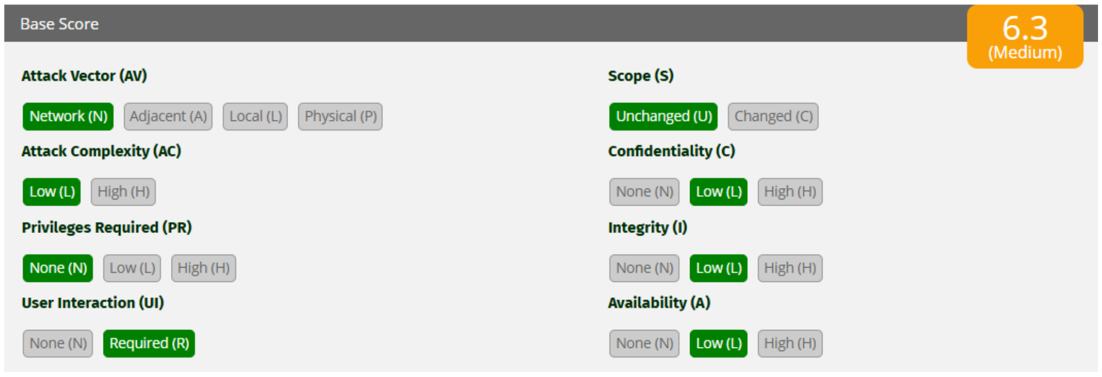
Vulnerability Discovery:

During authentication flow analysis of the platform with Burp Suite, I observed in responses that authenticated POST requests to dashboard functions lacked CSRF tokens.

A cookie analysis using Browser Dev Tools revealed that for some domains 'SameSite' configuration on authentication cookies was set to 'None', which led me to believe that there is a possibility of missing anti-CSRF tokens.

Severity Rating: Medium

A CVSS common vulnerability scoring system was used to estimate the severity of this vulnerability. (https://www.first.org/cvss/calculator/3-0)


Base Score 6.3 (Medium)

Attack Vector (AV): [✓ Network (N)] Adjacent (A) Local (L) Physical (P)
Attack Complexity (AC): [✓ Low (L)] High (H)
Privileges Required (PR): [✓ None (N)] Low (L) High (H)
User Interaction (UI): None (N) [✓ Required (R)]
Scope (S): [✓ Unchanged (U)] Changed (C)
Confidentiality (C): None (N) [✓ Low (L)] High (H)
Integrity (I): None (N) [✓ Low (L)] High (H)
Availability (A): None (N) [✓ Low (L)] High (H)

Figure 13: CVSS score of found vulnerability

Proof of Concept (PoC):

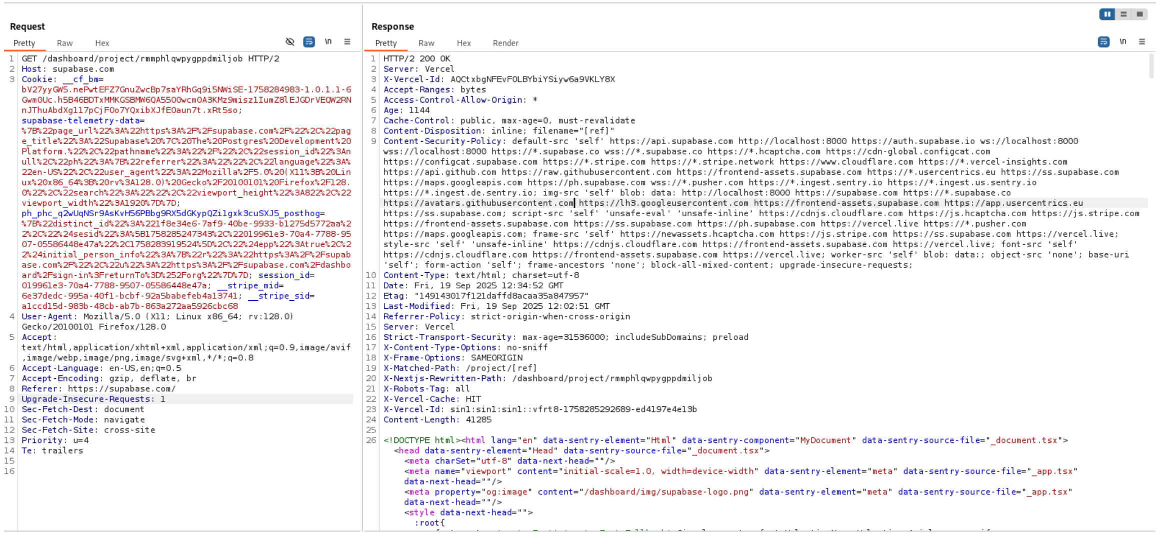
To avoid impacting production or other users, the following PoC was executed locally using a test account and a locally hosted HTML form that submits a POST to the dashboard endpoint.

Setup:

- Environment: Test account controlled by me, along with a locally produced cross-site request page.
- Target: Project page on the authenticated instance.
- Browser: Mozilla FireFox with Burp Suite proxy for observation.
- Cookie settings: Server uses SameSite=None for session cookies.

1. First, the request to load the dashboard was intercepted through the BurpSuite intercept, and after intercepting the request, it was forwarded so that there is an active session in the application.

2. When inspecting the intercepted POST request, there were no headers to be found that were related to anti-CSRF tokens.

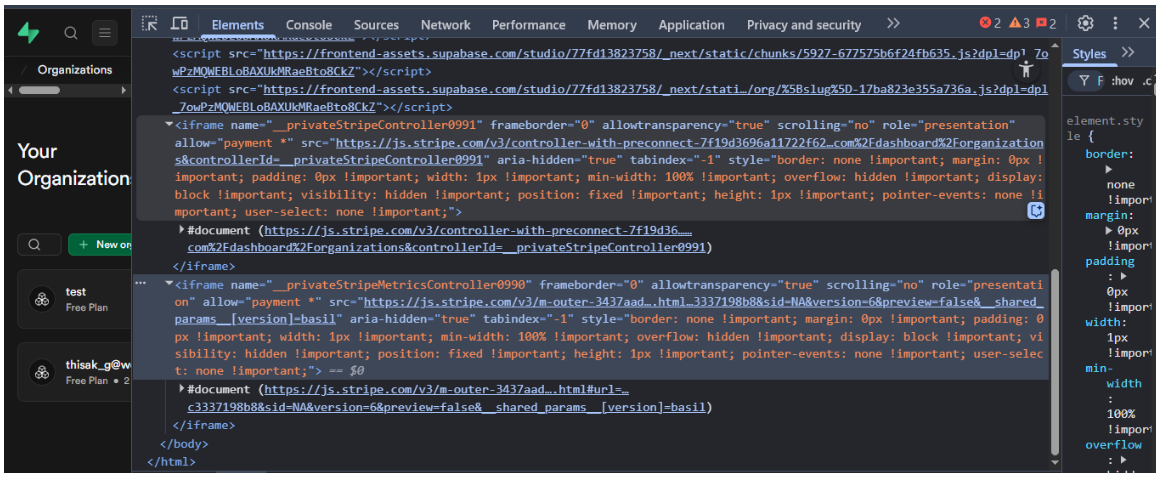


```
GET /dashboard/project/rmmphlqwpygppdmiljob HTTP/2
Host: supabase.com
Cookie: __cf_bm=
bV27yyGW5.nePwtEFZ7GnuZwcBp7saYRhGq9i5NWiSE-1758284983-1.0.1.1-6
Gwm0Uc.h5B46BDTxMMKGSBMW6QA5500wcm0A3KMz9misz1IumZ8lEJGDrVEQW2RN
nJThuAbdXg117pCjFOo7YQxibXJfEOaun7t.xRt5so;
supabase-telemetry-data=
%7B%22page_url%22%3A%22https%3A%2F%2Fsupabase.com%2F%22%2C%22pag
e_title%22%3A%22Supabase%20%7C%20The%20Postgres%20Development%20
Platform.%22%2C%22pathname%22%3A%22%2F%22%2C%22session_id%22%3An
ull%2C%22ph%22%3A%7B%22referrer%22%3A%22%22%2C%22language%22%3A%
22en-US%22%2C%22user_agent%22%3A%22Mozilla%2F5.0%20(X11%3B%20Lin
ux%20x86_64%3B%20rv%3A128.0)%20Gecko%2F20100101%20Firefox%2F128.
0%22%2C%22search%22%3A%22%22%2C%22viewport_height%22%3A822%2C%22
viewport_width%22%3A1920%7D%7D;
ph_phc_q2wUqNSr9AsKvH56PBbg9RX5dGKypQZi1gxk3cuSXJ5_posthog=
%7B%22distinct_id%22%3A%221f8e34e6-7af9-40be-9933-b1275d5772aa%2
2%2C%22%24sesid%22%3A%5B1758285247343%2C%22019961e3-70a4-7788-95
07-05586448e47a%22%2C1758283919524%5D%2C%22%24epp%22%3Atrue%2C%2
2%24initial_person_info%22%3A%7B%22r%22%3A%22https%3A%2F%2Fsupab
ase.com%2F%22%2C%22u%22%3A%22https%3A%2F%2Fsupabase.com%2Fdashbo
ard%2Fsign-in%3FreturnTo%3D%252Forg%22%7D%7D; session_id=
019961e3-70a4-7788-9507-05586448e47a; __stripe_mid=
6e37dedc-995a-40f1-bcbf-92a5babefeb4a13741; __stripe_sid=
a1ccd15d-983b-48cb-ab7b-863a272aa5926cbc68
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0)
Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif
,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://supabase.com/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
Priority: u=4
Te: trailers


```

```
HTTP/2 200 OK
Server: Vercel
X-Vercel-Id: AQCtxbgNFEvFOLBYbiYSiyw6a9VKLY8X
Accept-Ranges: bytes
Access-Control-Allow-Origin: *
Age: 1144
Cache-Control: public, max-age=0, must-revalidate
Content-Disposition: inline; filename="[ref]"
Content-Security-Policy: default-src 'self' https://api.supabase.com http://localhost:8000 https://auth.supabase.io ws://localhost:8000
wss://localhost:8000 https://*.supabase.co wss://*.supabase.co https://*.hcaptcha.com https://cdn-global.configcat.com
https://configcat.supabase.com https://*.stripe.com https://*.stripe.network https://www.cloudflare.com https://*.vercel-insights.com
https://api.github.com https://raw.githubusercontent.com https://frontend-assets.supabase.com https://*.usercentrics.eu https://ss.supabase.com
https://maps.googleapis.com https://ph.supabase.com wss://*.pusher.com https://*.ingest.sentry.io https://*.ingest.us.sentry.io
https://*.ingest.de.sentry.io; img-src 'self' blob: data: http://localhost:8000 https://supabase.com https://*.supabase.co
https://avatars.githubusercontent.com https://lh3.googleusercontent.com https://frontend-assets.supabase.com https://app.usercentrics.eu
https://ss.supabase.com; script-src 'self' 'unsafe-eval' 'unsafe-inline' https://cdnjs.cloudflare.com https://js.hcaptcha.com https://js.stripe.com
https://frontend-assets.supabase.com https://ss.supabase.com https://ph.supabase.com https://vercel.live https://*.pusher.com
https://maps.googleapis.com; frame-src 'self' https://newassets.hcaptcha.com https://js.stripe.com https://ss.supabase.com https://vercel.live;
style-src 'self' 'unsafe-inline' https://cdnjs.cloudflare.com https://frontend-assets.supabase.com https://vercel.live; font-src 'self'
https://cdnjs.cloudflare.com https://frontend-assets.supabase.com https://vercel.live; worker-src 'self' blob: data:; object-src 'none'; base-uri
'self'; form-action 'self'; frame-ancestors 'none'; block-all-mixed-content; upgrade-insecure-requests;
Content-Type: text/html; charset=utf-8
Date: Fri, 19 Sep 2025 12:34:52 GMT
Etag: "149143017f121daffd8acaa35a847957"
Last-Modified: Fri, 19 Sep 2025 12:02:51 GMT
Referrer-Policy: strict-origin-when-cross-origin
Server: Vercel
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
X-Content-Type-Options: no-sniff
X-Frame-Options: SAMEORIGIN
X-Matched-Path: /project/[ref]
X-Nextjs-Rewritten-Path: /dashboard/project/rmmphlqwpygppdmiljob
X-Robots-Tag: all
X-Vercel-Cache: HIT
X-Vercel-Id: sin1:sin1:sin1::vfrt8-1758285292689-ed4197e4e13b
Content-Length: 41285

<!DOCTYPE html><html lang="en" data-sentry-element="Html" data-sentry-component="MyDocument" data-sentry-source-file="_document.tsx">
  <head data-sentry-element="Head" data-sentry-source-file="_document.tsx">
    <meta charSet="utf-8" data-next-head=""/>
    <meta name="viewport" content="initial-scale=1.0, width=device-width" data-sentry-element="meta" data-sentry-source-file="_app.tsx"
    data-next-head=""/>
    <meta property="og:image" content="/dashboard/img/supabase-logo.png" data-sentry-element="meta" data-sentry-source-file="_app.tsx"
    data-next-head=""/>
    <style data-next-head="">
      :root{
```

Figure 14: Burpsuite Repeater tab used to identify the absence of anti-CSRF tokens

3. Since CSRF tokens are typically stored in a hidden input in form, the source code was checked using Browser Dev Tools; alas none were found.



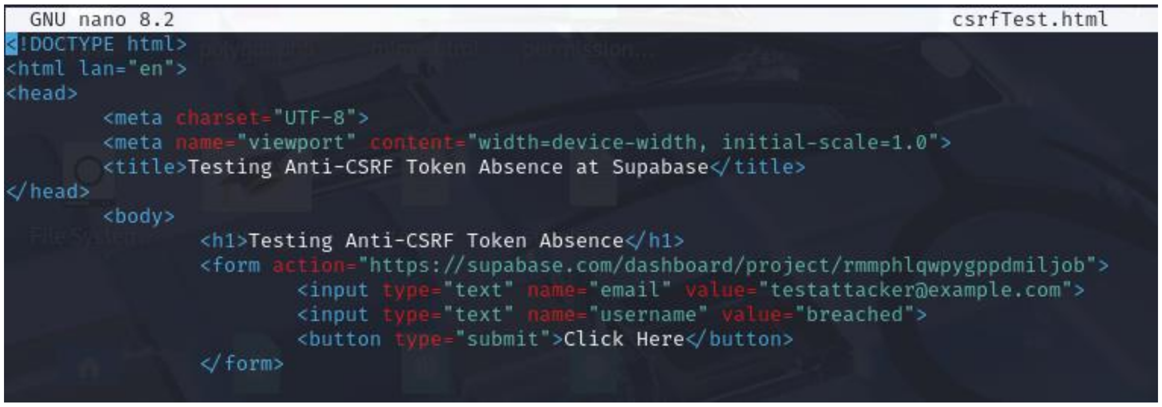
```
<script src="https://frontend-assets.supabase.com/studio/77fd13823758/_next/static/chunks/5927-677575b6f24fb635.js?dpl=dp1_7owPzMQWEBLoBAXUkMRaeBto8CkZ"></script>
<script src="https://frontend-assets.supabase.com/studio/77fd13823758/_next/stati…/org/%5Bslug%5D-17ba823e355a736a.js?dpl=dpl_7owPzMQWEBLoBAXUkMRaeBto8CkZ"></script>
▼<iframe name="__privateStripeController0991" frameborder="0" allowtransparency="true" scrolling="no" role="presentation" allow="payment *" src="https://js.stripe.com/v3/controller-with-preconnect-7f19d3696a11722f62…com%2Fdashboard%2Forganizations&controllerId=__privateStripeController0991" aria-hidden="true" tabindex="-1" style="border: none !important; margin: 0px !important; padding: 0px !important; width: 1px !important; min-width: 100% !important; overflow: hidden !important; display: block !important; visibility: hidden !important; position: fixed !important; height: 1px !important; pointer-events: none !important; user-select: none !important;">
  ▶#document (https://js.stripe.com/v3/controller-with-preconnect-7f19d36……com%2Fdashboard%2Forganizations&controllerId=__privateStripeController0991)
</iframe>
▼<iframe name="__privateStripeMetricsController0990" frameborder="0" allowtransparency="true" scrolling="no" role="presentation" allow="payment *" src="https://js.stripe.com/v3/m-outer-3437aad….html…3337198b8&sid=NA&version=6&preview=false&__shared_params__[version]=basil" aria-hidden="true" tabindex="-1" style="border: none !important; margin: 0px !important; padding: 0px !important; width: 1px !important; min-width: 100% !important; overflow: hidden !important; display: block !important; visibility: hidden !important; position: fixed !important; height: 1px !important; pointer-events: none !important; user-select: none !important;"> == $0
  ▶#document (https://js.stripe.com/v3/m-outer-3437aad….html#url=…c3337198b8&sid=NA&version=6&preview=false&__shared_params__[version]=basil)
</iframe>
</body>
</html>
```

```
element.style {
  border: ▶ none !import
  margin: ▶ 0px !import
  padding: ▶ 0px !import
  width: 1px !import
  min-width: 100% !import
  overflow: ▶
```

Figure 15: Chrome DevTools Elements tab to check source code

4. To clearly see if this platform is vulnerable to a CSRF attack, a simple test was carried out using a local server in Kali Linux and the test account created on the platform.

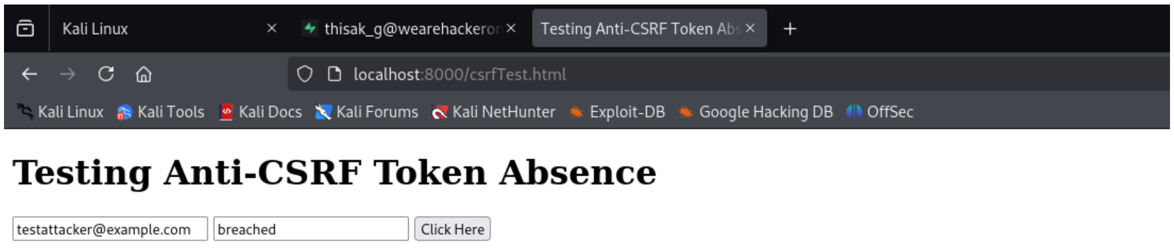
5. A simple HTML form was created to be hosted in the local server that submits data to the target application from another origin. (a simple form that has the URL of the application embedded in the form).



```
GNU nano 8.2                                                    csrfTest.html
<!DOCTYPE html>
<html lan="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Testing Anti-CSRF Token Absence at Supabase</title>
</head>
        <body>
                <h1>Testing Anti-CSRF Token Absence</h1>
                <form action="https://supabase.com/dashboard/project/rmmphlqwpygppdmiljob">
                        <input type="text" name="email" value="testattacker@example.com">
                        <input type="text" name="username" value="breached">
                        <button type="submit">Click Here</button>
                </form>
```

Figure 16: syntax of custom made HTML form

6. After running the HTML template on the local server, the code was executed to see if the locally simulated attack was successful.

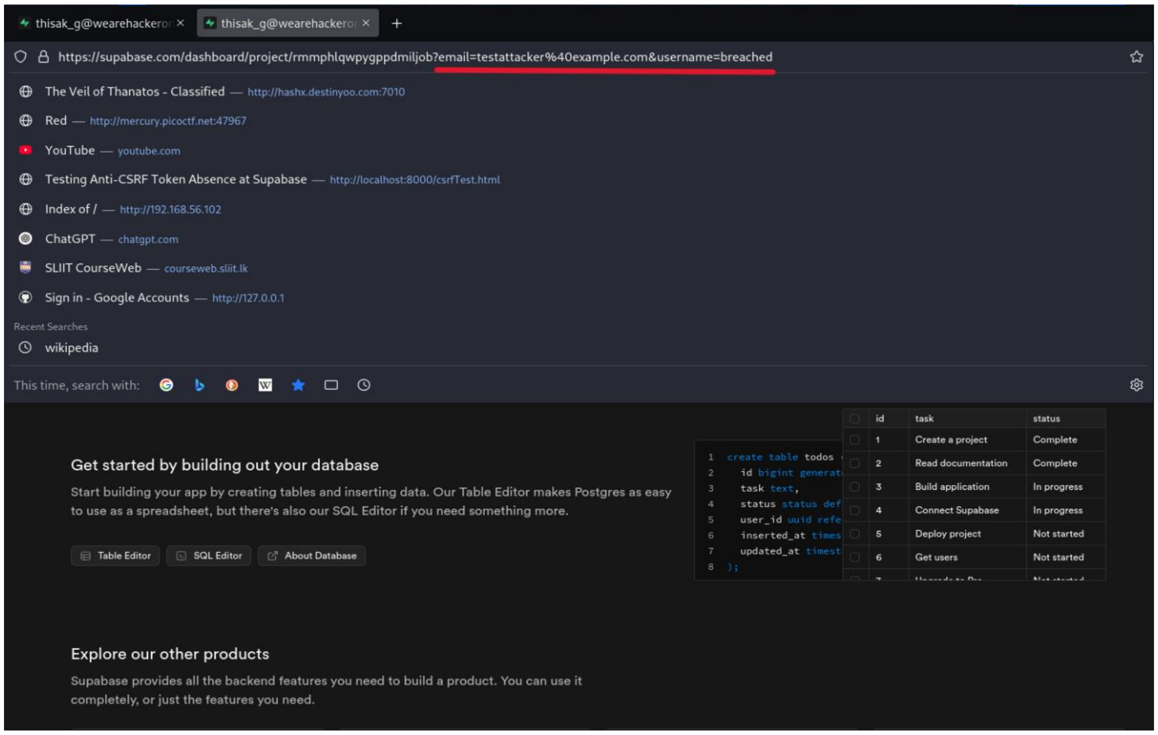


Testing Anti-CSRF Token Absence

testattacker@example.com | breached | Click Here

Figure 17: view of HTML form after hosting on local server

7. When the auto-submitting HTML form was executed from a separate origin while logged in with the test account, the Burp intercept showed a POST to the dashboard endpoint with no CSRF token present. The server processed the request and returned the dashboard response for the authenticated test account, indicating the application currently lacks per-request CSRF protection for this endpoint.



thisak_g@wearehackero × thisak_g@wearehackero ×

https://supabase.com/dashboard/project/rmmphlqwpygppdmiljob?email=testattacker%40example.com&username=breached

The Veil of Thanatos - Classified — http://hashx.destinyoo.com:7010
Red — http://mercury.picoctf.net:47967
YouTube — youtube.com
Testing Anti-CSRF Token Absence at Supabase — http://localhost:8000/csrfTest.html
Index of / — http://192.168.56.102
ChatGPT — chatgpt.com
SLIIT CourseWeb — courseweb.sliit.lk
Sign in - Google Accounts — http://127.0.0.1
Recent Searches
wikipedia
This time, search with:

Get started by building out your database

Start building your app by creating tables and inserting data. Our Table Editor makes Postgres as easy to use as a spreadsheet, but there's also our SQL Editor if you need something more.

Table Editor SQL Editor About Database

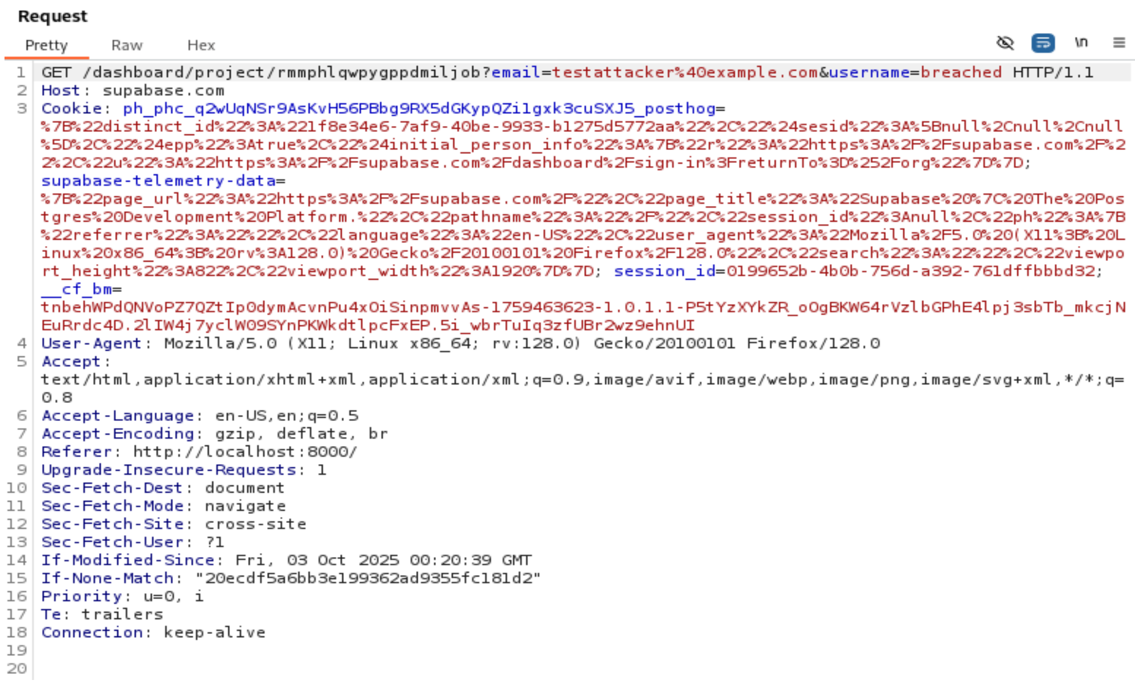
```
create table todos
  id bigint generat
  task text,
  status status def
  user_id uuid refe
  inserted_at times
  updated_at timest
);
```

id	task	status
1	Create a project	Complete
2	Read documentation	Complete
3	Build application	In progress
4	Connect Supabase	In progress
5	Deploy project	Not started
6	Get users	Not started

Explore our other products

Supabase provides all the backend features you need to build a product. You can use it completely, or just the features you need.

Figure 18: evidence of URL showing processed request



Request

Pretty Raw Hex

```
GET /dashboard/project/rmmphlqwpygppdmiljob?email=testattacker%40example.com&username=breached HTTP/1.1
Host: supabase.com
Cookie: ph_phc_q2wUqNSr9AsKvH56PBbg9RX5dGKypQZilgxk3cuSXJ5_posthog=
%7B%22distinct_id%22%3A%221f8e34e6-7af9-40be-9933-b1275d5772aa%22%2C%22%24sesid%22%3A%5Bnull%2Cnull%2Cnull
%5D%2C%22%24epp%22%3Atrue%2C%22%24initial_person_info%22%3A%7B%22r%22%3A%22https%3A%2F%2Fsupabase.com%2F%2
2%2C%22u%22%3A%22https%3A%2F%2Fsupabase.com%2Fdashboard%2Fsign-in%3FreturnTo%3D%252Forg%22%7D%7D;
supabase-telemetry-data=
%7B%22page_url%22%3A%22https%3A%2F%2Fsupabase.com%2F%22%2C%22page_title%22%3A%22Supabase%20%7C%20The%20Pos
tgres%20Development%20Platform.%22%2C%22pathname%22%3A%22%2F%22%2C%22session_id%22%3Anull%2C%22ph%22%3A%7B
%22referrer%22%3A%22%22%2C%22language%22%3A%22en-US%22%2C%22user_agent%22%3A%22Mozilla%2F5.0%20(X11%3B%20L
inux%20x86_64%3B%20rv%3A128.0)%20Gecko%2F20100101%20Firefox%2F128.0%22%2C%22search%22%3A%22%22%2C%22viewpo
rt_height%22%3A822%2C%22viewport_width%22%3A1920%7D%7D; session_id=0199652b-4b0b-756d-a392-761dffbbbd32;
__cf_bm=
tnbehWPdQNVoPZ7QZtIp0dymAcvnPu4xOiSinpmvvAs-1759463623-1.0.1.1-P5tYzXYkZR_oOgBKW64rVzlbGPhE4lpj3sbTb_mkcjN
EuRrdc4D.2lIW4j7yclW09SYnPKWkdtlpcFxEP.5i_wbrTuIq3zfUBr2wz9ehnUI
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=
0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://localhost:8000/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
Sec-Fetch-User: ?1
If-Modified-Since: Fri, 03 Oct 2025 00:20:39 GMT
If-None-Match: "20ecdf5a6bb3e199362ad9355fc181d2"
Priority: u=0, i
Te: trailers
Connection: keep-alive


```

Figure 19: view of the request captured through BurpSuite

Exploitation:

An example of an attack in action can be described in the following steps:

1. Attacker creates a malicious website containing a CSRF payload.
2. Authenticated Supabase user visits the attacker's website.
3. The browser automatically sends the authenticated request to Supabase API.
4. The payload action is performed without user consent due to missing CSRF protection in the application.
5. The attacker can successfully modify the user's account, projects, or even organization settings.

Impact of CSRF Attacks:

- Unauthorized modification of user projects and settings in the application.
- Potential account takeover scenarios through email/password changes.
- Organization membership detail manipulation.
- API key generation without the knowledge of the user.
- Reputation damage and loss of user trust in the platform.

Mitigation Strategies:

- Implementing CSRF tokens for all state-changing operations in the platform.
- Configuring session cookies with ‘SameSite’ set to Strict or Lax instead of None.
- Adding additional request validation such as custom headers.
- Implementing proper CORS (Cross Origin Resource Sharing) policies for sensitive endpoints.
- Passing the CSRF token as an AJAX header for AJAX requests.
- Rotating tokens periodically to limit token reuse.

2.3.1 Finding 02: Outdated JavaScript Library (DOMPurify v 3.1.7)

Vulnerability Title:

DOMPurify (versions older than 3.2.4) consists of a potential mutation-based XSS bypassing vulnerability when certain payload encodings or options are used.

Description:

DOMPurify is a client-side HTML sanitizer. Older versions (prior to 3.2.4) contain a known class of issues where template-literal/backtick-style payloads or unusual encodings can confuse the sanitizer and allow dangerous attributes/tags to survive sanitization in some contexts (leading to reflected or stored XSS).

Mutation-based XSS occurs when an attacker's payload is transformed or 'mutated' by the browser's parsing or sanitization logic into a dangerous form.

Even if applications currently escape rendered HTML, using an outdated sanitizer version increases the risk that a future change in rendering (e.g., innerHTML, dangerouslySetInnerHTML, markdown HTML pipelines) will enable stored XSS.

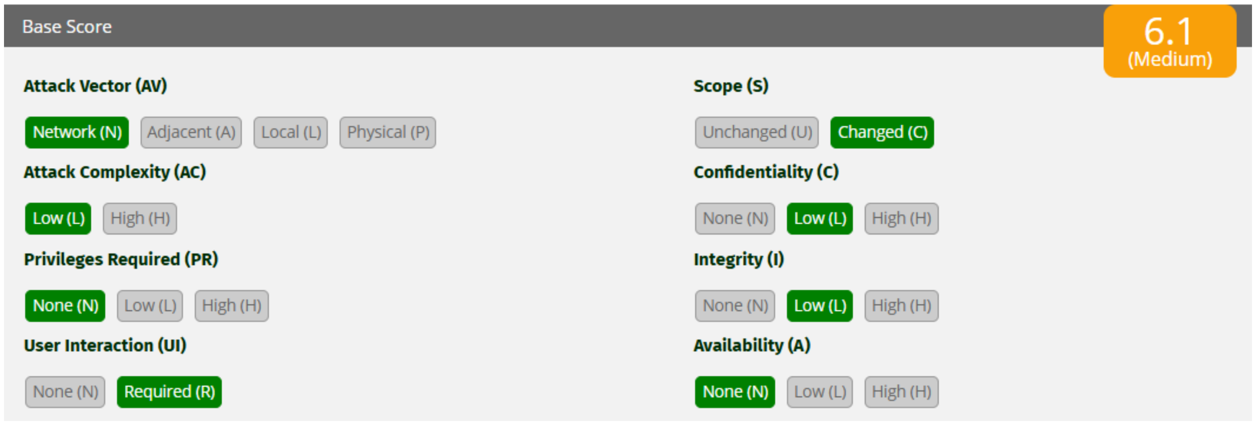
Vulnerability Discovery:

The initial intention was to identify if there were any XSS vulnerabilities present on the platform. After several attempts to carry out successful payloads, it became evident that the platform is well equipped in handling XSS attacks.

But during an investigation of the front-end of the platform, the source code showed a JavaScript Library that sanitizes user input. Upon further inspection with cross reference to the CVE database, it became apparent that there is a vulnerability due to the library being outdated.

Severity Rating: Medium

A CVSS common vulnerability scoring system was used to estimate the severity of this vulnerability. (https://www.first.org/cvss/calculator/3-0)



Base Score 6.1 (Medium)

Attack Vector (AV): [✓ Network (N)] Adjacent (A) Local (L) Physical (P)

Attack Complexity (AC): [✓ Low (L)] High (H)

Privileges Required (PR): [✓ None (N)] Low (L) High (H)

User Interaction (UI): None (N) [✓ Required (R)]

Scope (S): Unchanged (U) [✓ Changed (C)]

Confidentiality (C): None (N) [✓ Low (L)] High (H)

Integrity (I): None (N) [✓ Low (L)] High (H)

Availability (A): [✓ None (N)] Low (L) High (H)

Figure 20: CVSS score for found vulnerability

Proof of Concept (PoC):

This PoC is created to compare sanitizer behavior in DOMPurify 3.1.7 vs DOMPurify 3.2.4 and the steps taken to identify the existence of the previous version. If the older version leaves dangerous attributes/ tags after ‘sanitize( )’ while the newer version removes them, proving a library-level bypass. All testing was performed locally on controlled test pages to remain within program scope and avoid modifying any production code.

1. The first step was to locate any type of input field within the target application. A ‘create project’ section with input fields was located and a simple JavaScript payload was entered to see if it would cause any reaction.

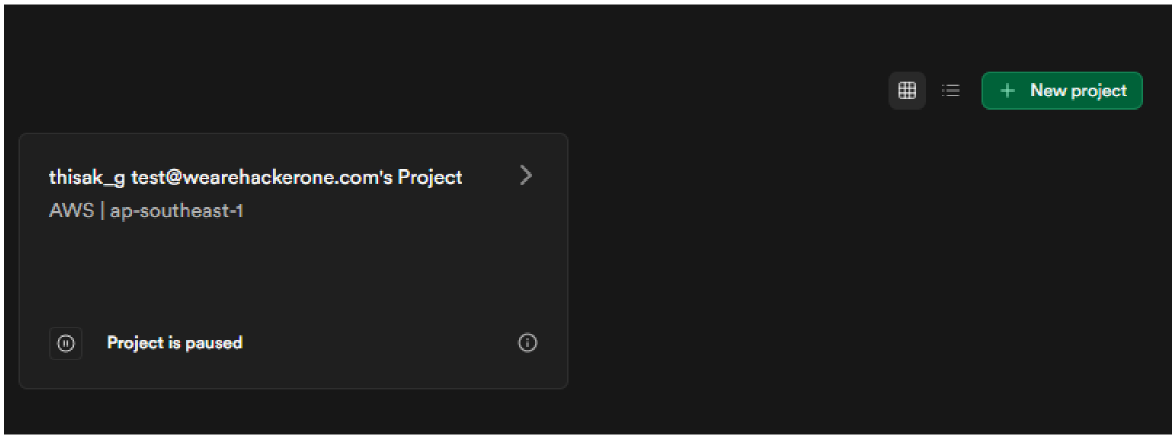
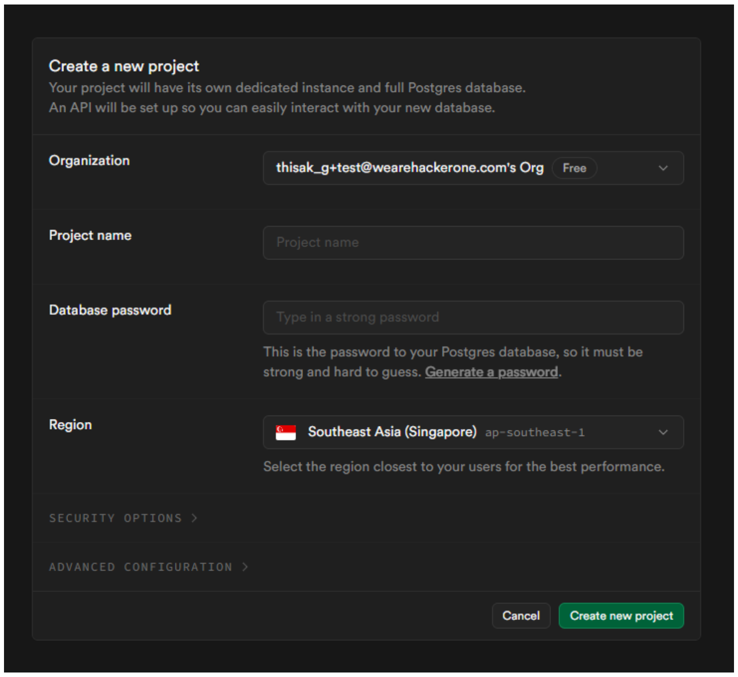


Figure 21: Dashboard contains a create new project option



Create a new project

Your project will have its own dedicated instance and full Postgres database.
An API will be set up so you can easily interact with your new database.

Organization: thisak_g+test@wearehackerone.com's Org Free

Project name: Project name

Database password: Type in a strong password

This is the password to your Postgres database, so it must be strong and hard to guess. Generate a password.

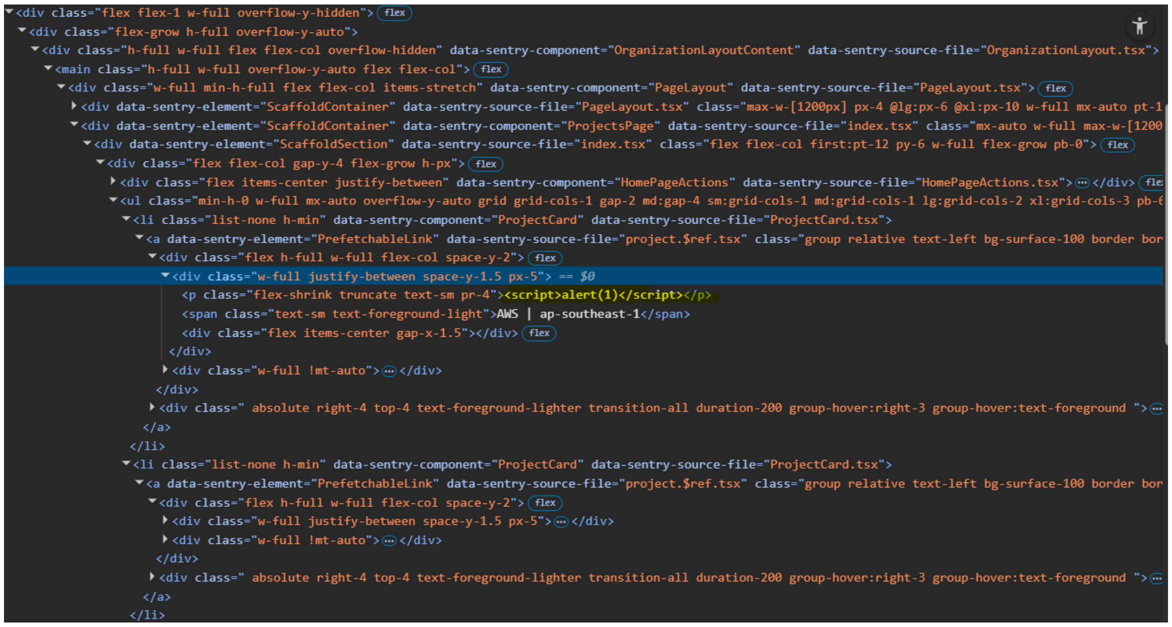
Region: Southeast Asia (Singapore) ap-southeast-1

Select the region closest to your users for the best performance.

Figure 23: The create new project form with input fields

2. The payload was as follows:
    - *<script>alert(1)</script>*

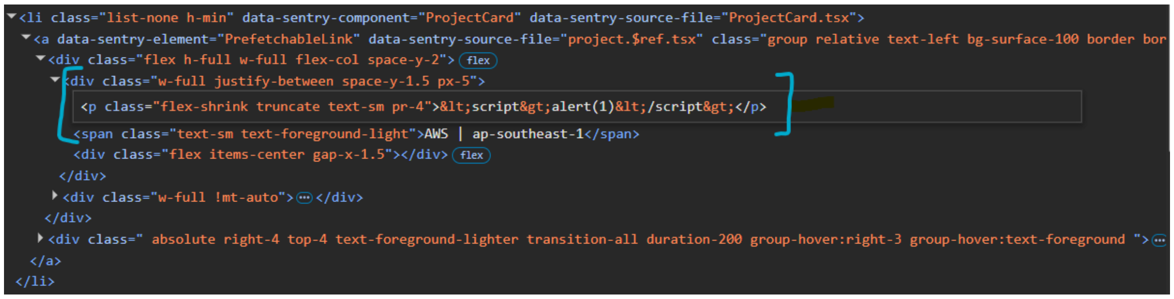
   Even though there weren't any alert pop-ups, a new project was created on the dashboard with the name '<script>alert(1)</script>'. This led me to believe that there might be some sort of HTML escaping present.



```
▼<div class="flex flex-1 w-full overflow-y-hidden"> flex
 ▼<div class="flex-grow h-full overflow-y-auto">
  ▼<div class="h-full w-full flex flex-col overflow-hidden" data-sentry-component="OrganizationLayoutContent" data-sentry-source-file="OrganizationLayout.tsx">
   ▼<main class="h-full w-full overflow-y-auto flex flex-col"> flex
    ▼<div class="w-full min-h-full flex flex-col items-stretch" data-sentry-component="PageLayout" data-sentry-source-file="PageLayout.tsx"> flex
     ▶<div data-sentry-element="ScaffoldContainer" data-sentry-source-file="PageLayout.tsx" class="max-w-[1200px] px-4 @lg:px-6 @xl:px-10 w-full mx-auto pt-1
     ▼<div data-sentry-element="ScaffoldContainer" data-sentry-component="ProjectsPage" data-sentry-source-file="index.tsx" class="mx-auto w-full max-w-[1200
      ▼<div data-sentry-element="ScaffoldSection" data-sentry-source-file="index.tsx" class="flex flex-col first:pt-12 py-6 w-full flex-grow pb-0"> flex
       ▼<div class="flex flex-col gap-y-4 flex-grow h-px"> flex
        ▶<div class="flex items-center justify-between" data-sentry-component="HomePageActions" data-sentry-source-file="HomePageActions.tsx"> … </div> fle
        ▼<ul class="min-h-0 w-full mx-auto overflow-y-auto grid grid-cols-1 gap-2 md:gap-4 sm:grid-cols-1 md:grid-cols-1 lg:grid-cols-2 xl:grid-cols-3 pb-6
         ▼<li class="list-none h-min" data-sentry-component="ProjectCard" data-sentry-source-file="ProjectCard.tsx">
          ▼<a data-sentry-element="PrefetchableLink" data-sentry-source-file="project.$ref.tsx" class="group relative text-left bg-surface-100 border bor
           ▼<div class="flex h-full w-full flex-col space-y-2"> flex
            ▼<div class="w-full justify-between space-y-1.5 px-5"> == $0
               <p class="flex-shrink truncate text-sm pr-4"><script>alert(1)</script></p>
               <span class="text-sm text-foreground-light">AWS | ap-southeast-1</span>
               <div class="flex items-center gap-x-1.5"></div> flex
             </div>
            ▶<div class="w-full !mt-auto"> … </div>
           </div>
          ▶<div class=" absolute right-4 top-4 text-foreground-lighter transition-all duration-200 group-hover:right-3 group-hover:text-foreground "> …
          </a>
         </li>
         ▼<li class="list-none h-min" data-sentry-component="ProjectCard" data-sentry-source-file="ProjectCard.tsx">
          ▼<a data-sentry-element="PrefetchableLink" data-sentry-source-file="project.$ref.tsx" class="group relative text-left bg-surface-100 border bor
           ▼<div class="flex h-full w-full flex-col space-y-2"> flex
            ▶<div class="w-full justify-between space-y-1.5 px-5"> … </div>
            ▶<div class="w-full !mt-auto"> … </div>
           </div>
          ▶<div class=" absolute right-4 top-4 text-foreground-lighter transition-all duration-200 group-hover:right-3 group-hover:text-foreground "> …
          </a>
         </li>
```

Figure 24: Source code showing the output of the attempted payload

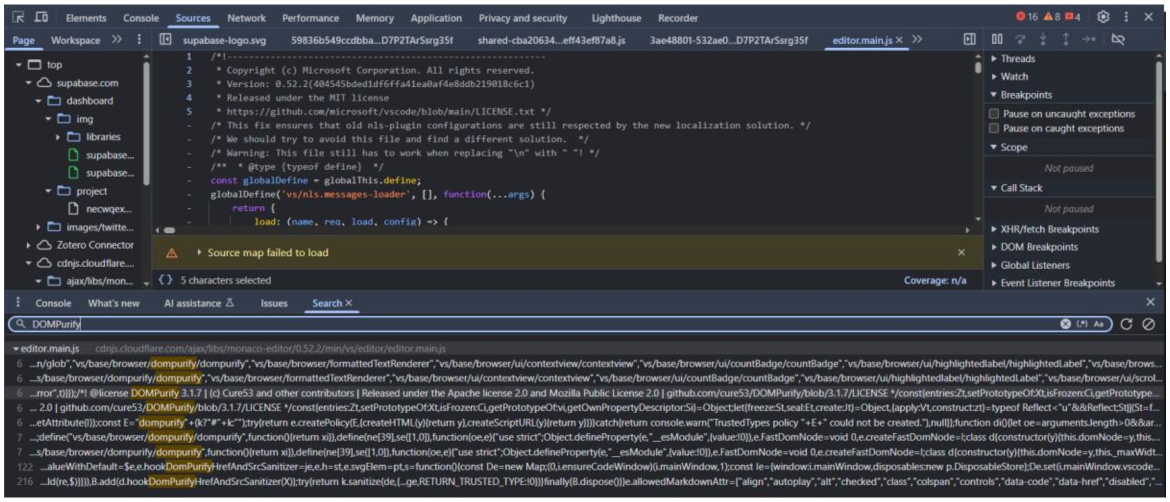
3. Upon investigating the source code, it became apparent that the page has rendered it as literal text, proving the output is encoded, not executed. To continue to see if there are any other ways to exploit an XSS vulnerability, an investigation on the presence of any sanitizing or filtering methods present in the target domain. This led to the finding of the DOMPurify library.



```
▼<li class="list-none h-min" data-sentry-component="ProjectCard" data-sentry-source-file="ProjectCard.tsx">
 ▼<a data-sentry-element="PrefetchableLink" data-sentry-source-file="project.$ref.tsx" class="group relative text-left bg-surface-100 border bor
  ▼<div class="flex h-full w-full flex-col space-y-2"> flex
   ▼<div class="w-full justify-between space-y-1.5 px-5">
     <p class="flex-shrink truncate text-sm pr-4">&lt;script&gt;alert(1)&lt;/script&gt;</p>
     <span class="text-sm text-foreground-light">AWS | ap-southeast-1</span>
     <div class="flex items-center gap-x-1.5"></div> flex
    </div>
   ▶<div class="w-full !mt-auto"> … </div>
   </div>
  ▶<div class=" absolute right-4 top-4 text-foreground-lighter transition-all duration-200 group-hover:right-3 group-hover:text-foreground "> …
  </a>
</li>
```

Figure 25: source code showing escaping HTML

4. The Chrome Browser Dev Tools was used to identify the presence of a DOMPurify build (version 3.1.7) referenced by the application.



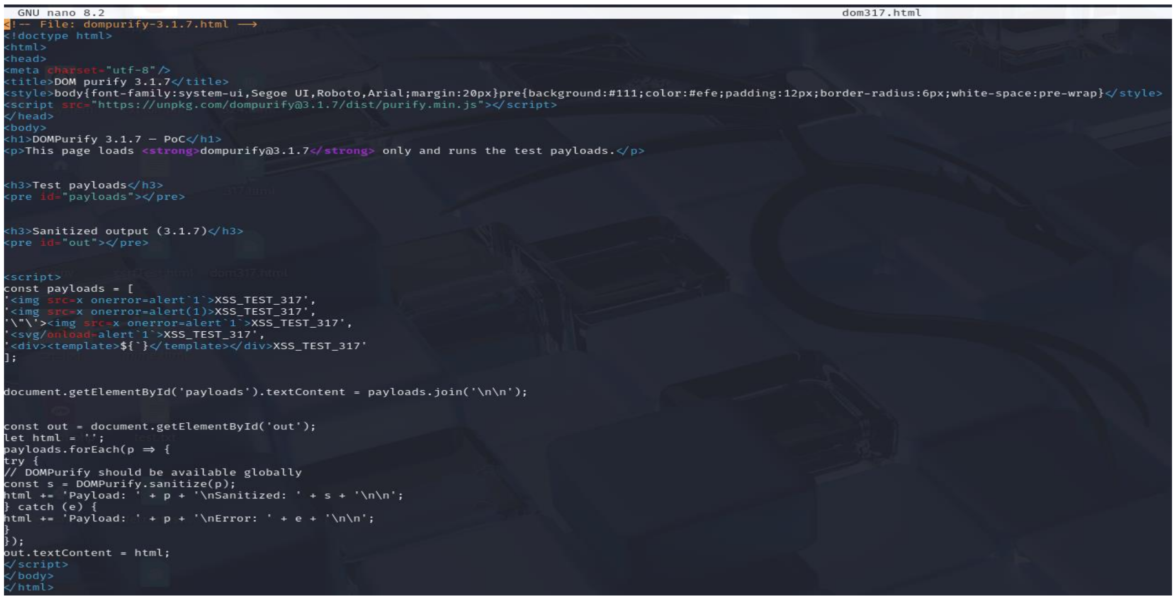
```
/*!-----------------------------------------------------------
 * Copyright (c) Microsoft Corporation. All rights reserved.
 * Version: 0.52.2(404545bded1df6ffa41ea0af4e8ddb219018c6c1)
 * Released under the MIT license
 * https://github.com/microsoft/vscode/blob/main/LICENSE.txt */
/* This fix ensures that old nls-plugin configurations are still respected by the new localization solution. */
/* We should try to avoid this file and find a different solution.  */
/* Warning: This file still has to work when replacing "\n" with " "! */
/**  * @type {typeof define}  */
const globalDefine = globalThis.define;
globalDefine('vs/nls.messages-loader', [], function(...args) {
    return {
        load: (name, req, load, config) => {
```

```
Source map failed to load
5 characters selected                                    Coverage: n/a
```

```
DOMPurify

editor.main.js   cdnjs.cloudflare.com/ajax/libs/monaco-editor/0.52.2/min/vs/editor/editor.main.js
6   ...n/glob","vs/base/browser/dompurify/dompurify","vs/base/browser/formattedTextRenderer","vs/base/browser/ui/contextview/contextview","vs/base/browser/ui/countBadge/countBadge","vs/base/browser/ui/highlightedlabel/highlightedLabel","vs/base/brows...
6   ...s/base/browser/dompurify/dompurify","vs/base/browser/formattedTextRenderer","vs/base/browser/ui/contextview/contextview","vs/base/browser/ui/countBadge/countBadge","vs/base/browser/ui/highlightedlabel/highlightedLabel","vs/base/browser/ui/scrol...
6   ...rror",t)}});/*! @license DOMPurify 3.1.7 | (c) Cure53 and other contributors | Released under the Apache license 2.0 and Mozilla Public License 2.0 | github.com/cure53/DOMPurify/blob/3.1.7/LICENSE */const{entries:Zt,setPrototypeOf:Xt,isFrozen:Ci,getPrototype...
6   ... 2.0 | github.com/cure53/DOMPurify/blob/3.1.7/LICENSE */const{entries:Zt,setPrototypeOf:Xt,isFrozen:Ci,getPrototypeOf:vi,getOwnPropertyDescriptor:Si}=Object;let{freeze:St,seal:Et,create:Jt}=Object,{apply:Vt,construct:zt}=typeof Reflect<"u"&&Reflect;St||(St=f...
6   ...etAttribute(l));const E="dompurify"+(k?"#"+k:"");try{return e.createPolicy(E,{createHTML(y){return y},createScriptURL(y){return y}})}catch{return console.warn("TrustedTypes policy "+E+" could not be created."),null}};function di(){let oe=arguments.length>0&&ar...
7   ...;define("vs/base/browser/dompurify/dompurify",function(){return xi}),define(ne[39],se([1,0]),function(oe,e){"use strict";Object.defineProperty(e,"__esModule",{value:!0}),e.FastDomNode=void 0,e.createFastDomNode=l;class d{constructor(y){this.domNode=y,this....
7   ...s/base/browser/dompurify/dompurify",function(){return xi}),define(ne[39],se([1,0]),function(oe,e){"use strict";Object.defineProperty(e,"__esModule",{value:!0}),e.FastDomNode=void 0,e.createFastDomNode=l;class d{constructor(y){this.domNode=y,this._maxWidt...
122 ...alueWithDefault=$e,e.hookDomPurifyHrefAndSrcSanitizer=je,e.h=st,e.svgElem=pt,s=function(){const De=new Map;(0,i.ensureCodeWindow)(i.mainWindow,1);const le={window:i.mainWindow,disposables:new p.DisposableStore};De.set(i.mainWindow.vscode...
216 ...ld(re,$)}}}),B.add(d.hookDomPurifyHrefAndSrcSanitizer(X));try{return k.sanitize(de,{...ge,RETURN_TRUSTED_TYPE:!0})}finally{B.dispose()}}e.allowedMarkdownAttr=["align","autoplay","alt","checked","class","colspan","controls","data-code","data-href","disabled",...
```

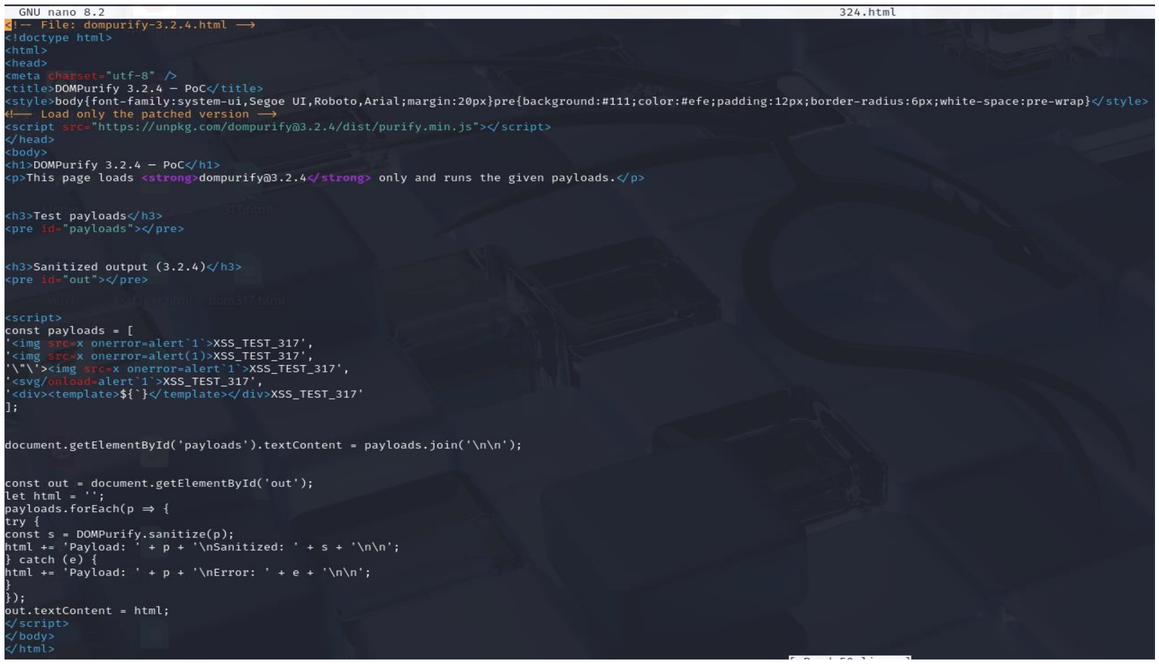
Figure 26: DevTools search option showing results

5. Next, to compare the sanitizer behavior locally, two test pages were created using HTML and JavaScript (317.html and 324.html). Both includes the same controlled payload and calls the sanitize function with default options to mirror common integration usage.



```
  GNU nano 8.2                                                                   dom317.html
<!-- File: dompurify-3.1.7.html -->
<!doctype html>
<html>
<head>
<meta charset="utf-8"/>
<title>DOM purify 3.1.7</title>
<style>body{font-family:system-ui,Segoe UI,Roboto,Arial;margin:20px}pre{background:#111;color:#efe;padding:12px;border-radius:6px;white-space:pre-wrap}</style>
<script src="https://unpkg.com/dompurify@3.1.7/dist/purify.min.js"></script>
</head>
<body>
<h1>DOMPurify 3.1.7 — PoC</h1>
<p>This page loads <strong>dompurify@3.1.7</strong> only and runs the test payloads.</p>


<h3>Test payloads</h3>
<pre id="payloads"></pre>


<h3>Sanitized output (3.1.7)</h3>
<pre id="out"></pre>


<script>
const payloads = [
'<img src=x onerror=alert`1`>XSS_TEST_317',
'<img src=x onerror=alert(1)>XSS_TEST_317',
'\"\'><img src=x onerror=alert`1`>XSS_TEST_317',
'<svg/onload=alert`1`>XSS_TEST_317',
'<div><template>${`}</template></div>XSS_TEST_317'
];


document.getElementById('payloads').textContent = payloads.join('\n\n');


const out = document.getElementById('out');
let html = '';
payloads.forEach(p => {
try {
// DOMPurify should be available globally
const s = DOMPurify.sanitize(p);
html += 'Payload: ' + p + '\nSanitized: ' + s + '\n\n';
} catch (e) {
html += 'Payload: ' + p + '\nError: ' + e + '\n\n';
}
});
out.textContent = html;
</script>
</body>
</html>
```

Figure 27: locally created HTML page for DOMPurify version 3.1.7



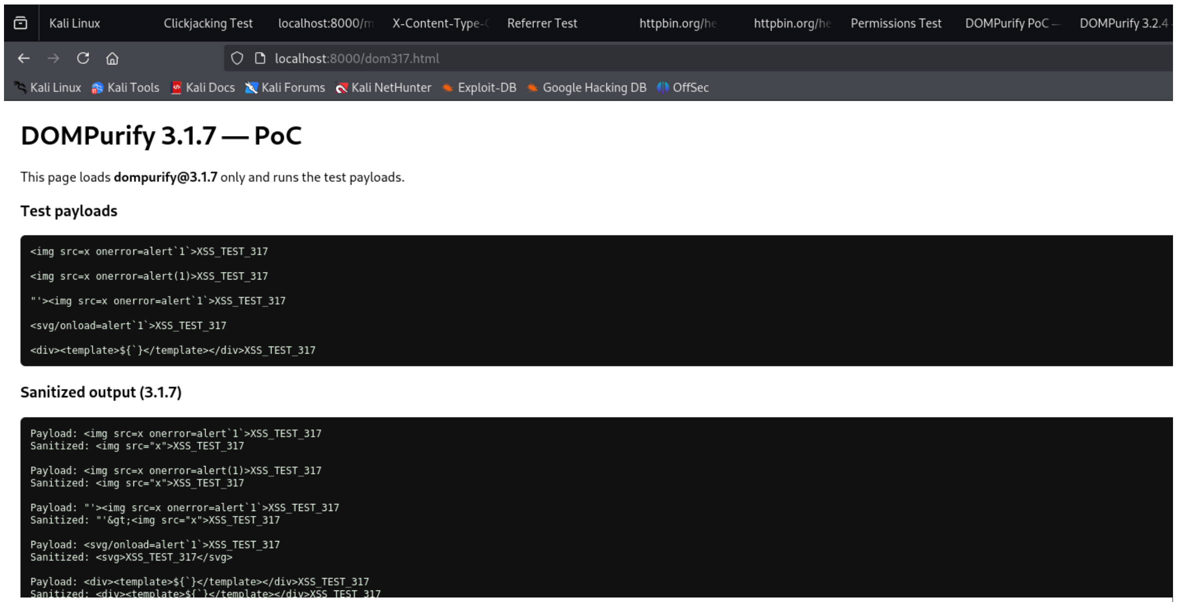
```
GNU nano 8.2                                                                324.html
<!-- File: dompurify-3.2.4.html -->
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>DOMPurify 3.2.4 — PoC</title>
<style>body{font-family:system-ui,Segoe UI,Roboto,Arial;margin:20px}pre{background:#111;color:#efe;padding:12px;border-radius:6px;white-space:pre-wrap}</style>
<!-- Load only the patched version -->
<script src="https://unpkg.com/dompurify@3.2.4/dist/purify.min.js"></script>
</head>
<body>
<h1>DOMPurify 3.2.4 — PoC</h1>
<p>This page loads <strong>dompurify@3.2.4</strong> only and runs the given payloads.</p>


<h3>Test payloads</h3>
<pre id="payloads"></pre>


<h3>Sanitized output (3.2.4)</h3>
<pre id="out"></pre>


<script>
const payloads = [
'<img src=x onerror=alert`1`>XSS_TEST_317',
'<img src=x onerror=alert(1)>XSS_TEST_317',
'\"\'><img src=x onerror=alert`1`>XSS_TEST_317',
'<svg/onload=alert`1`>XSS_TEST_317',
'<div><template>${`}</template></div>XSS_TEST_317'
];


document.getElementById('payloads').textContent = payloads.join('\n\n');


const out = document.getElementById('out');
let html = '';
payloads.forEach(p => {
try {
const s = DOMPurify.sanitize(p);
html += 'Payload: ' + p + '\nSanitized: ' + s + '\n\n';
} catch (e) {
html += 'Payload: ' + p + '\nError: ' + e + '\n\n';
}
});
out.textContent = html;
</script>
</body>
</html>
```

Figure 28: locally created HTML page for DOMPurify version 3.2.4

6. Then the two pages are loaded to compare using a local server (command: *python3 -m http.server 8000*) and opened in the same browser.

7. The expected PoC outputs should:
   - 3.1.7 version may produce sanitized text that includes 'onerror' or unexpected tags in some encodings.

   - 3.2.4 version should strip dangerous attributes and show only safe tags/values.



DOMPurify 3.1.7 — PoC

This page loads **dompurify@3.1.7** only and runs the test payloads.

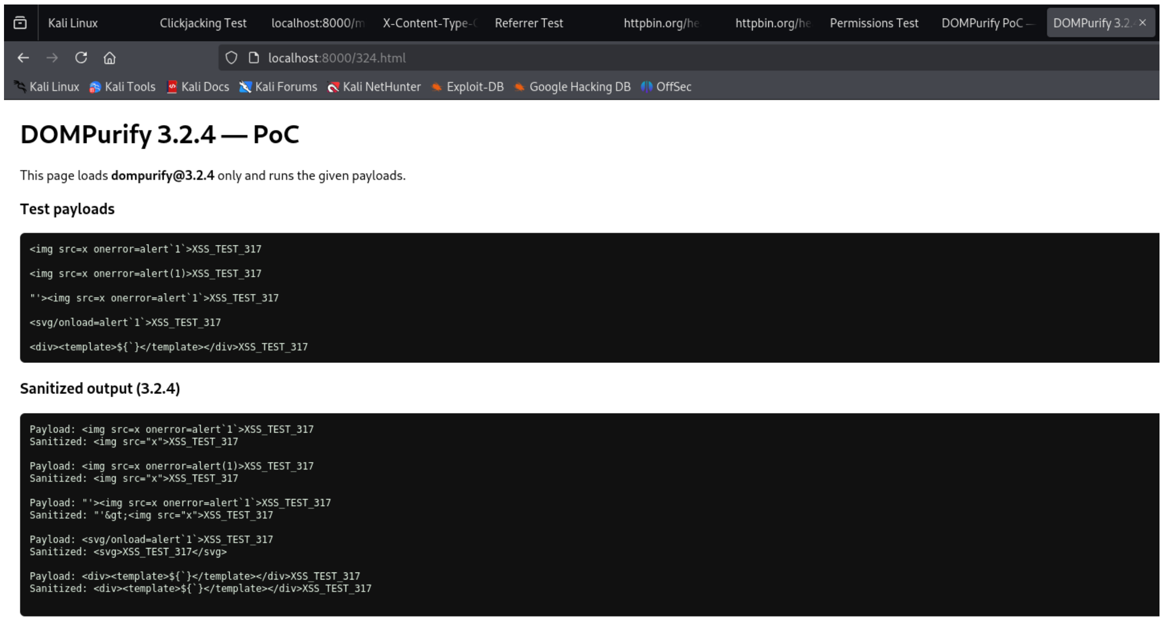
Test payloads

```
<img src=x onerror=alert`1`>XSS_TEST_317

<img src=x onerror=alert(1)>XSS_TEST_317

"'><img src=x onerror=alert`1`>XSS_TEST_317

<svg/onload=alert`1`>XSS_TEST_317

<div><template>${`}</template></div>XSS_TEST_317
```

Sanitized output (3.1.7)

```
Payload: <img src=x onerror=alert`1`>XSS_TEST_317
Sanitized: <img src="x">XSS_TEST_317

Payload: <img src=x onerror=alert(1)>XSS_TEST_317
Sanitized: <img src="x">XSS_TEST_317

Payload: "'><img src=x onerror=alert`1`>XSS_TEST_317
Sanitized: "'&gt;<img src="x">XSS_TEST_317

Payload: <svg/onload=alert`1`>XSS_TEST_317
Sanitized: <svg>XSS_TEST_317</svg>

Payload: <div><template>${`}</template></div>XSS_TEST_317
Sanitized: <div><template>${`}</template></div>XSS_TEST_317
```

Figure 29: result of DOMPUrify version 3.1.7


DOMPurify 3.2.4 — PoC

This page loads **dompurify@3.2.4** only and runs the given payloads.

Test payloads

```
<img src=x onerror=alert`1`>XSS_TEST_317

<img src=x onerror=alert(1)>XSS_TEST_317

"'><img src=x onerror=alert`1`>XSS_TEST_317

<svg/onload=alert`1`>XSS_TEST_317

<div><template>${`}</template></div>XSS_TEST_317
```

Sanitized output (3.2.4)

```
Payload: <img src=x onerror=alert`1`>XSS_TEST_317
Sanitized: <img src="x">XSS_TEST_317

Payload: <img src=x onerror=alert(1)>XSS_TEST_317
Sanitized: <img src="x">XSS_TEST_317

Payload: "'><img src=x onerror=alert`1`>XSS_TEST_317
Sanitized: "'&gt;<img src="x">XSS_TEST_317

Payload: <svg/onload=alert`1`>XSS_TEST_317
Sanitized: <svg>XSS_TEST_317</svg>

Payload: <div><template>${`}</template></div>XSS_TEST_317
Sanitized: <div><template>${`}</template></div>XSS_TEST_317
```

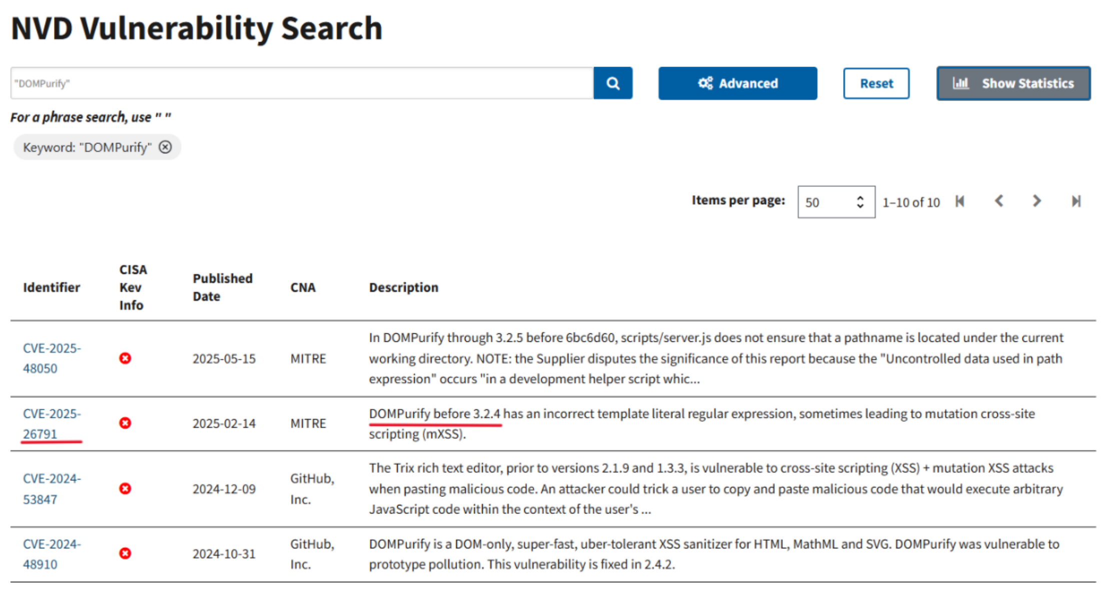
Figure 30: result of DOMPurify version 3.2.4

8. In my local comparison using the application's default sanitize options, the sanitized outputs were identical for both versions, and no immediate bypass was observed. However, this does not eliminate the risk: sanitizer behavior can differ depending on integration options or unusual input encodings.

9. Although the local sanitizer comparison produced identical sanitized outputs, multiple authoritative sources confirm a configuration-dependent mutation-XSS (mXSS) bypass affecting DOMPurify releases prior to 3.2.4. The issue is tracked in public vulnerability databases and SCA advisories and because the vulnerability is configuration-dependent (it may only trigger with certain options or parsing conditions), the absence of a local bypass does not negate the documented risk. Attached below are several references to prove the existence of the above vulnerability. Namely:

   - NVD/CVE Database
   - Snyk Advisory listing Database
   - GitHub Security Advisory

10. Therefore, the safest remediation is to upgrade DOMPurify to the patched release (3.2.4) to remove the class of known issues and reduce attack surface for future changes.



NVD Vulnerability Search

"DOMPurify"

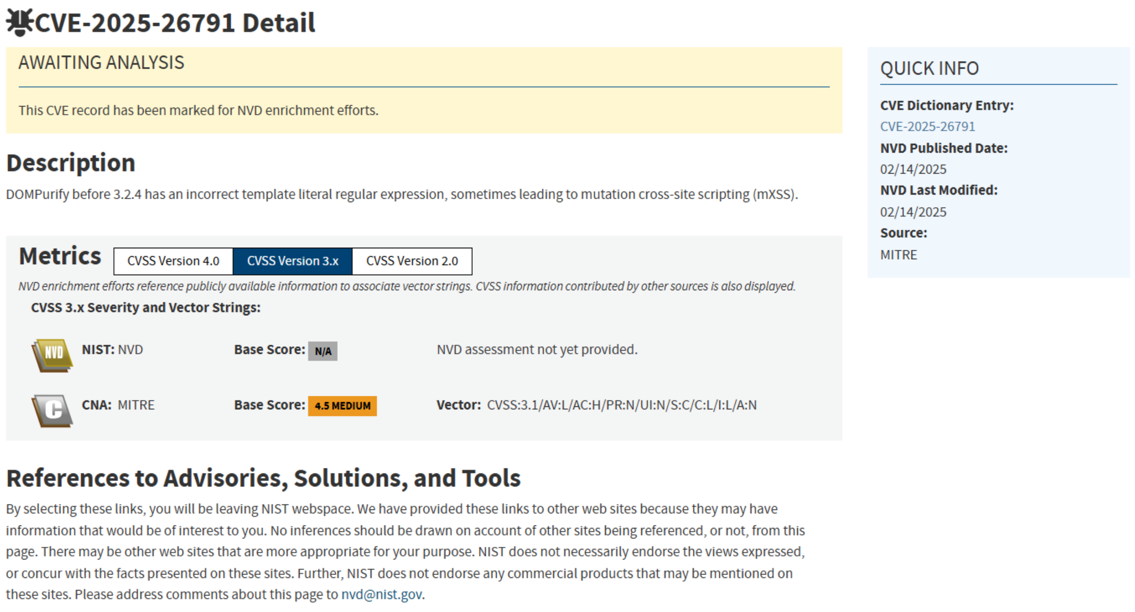
For a phrase search, use " "

Keyword: "DOMPurify"

Items per page: 50 1–10 of 10

Identifier	CISA Kev Info	Published Date	CNA	Description
CVE-2025-48050	✖	2025-05-15	MITRE	In DOMPurify through 3.2.5 before 6bc6d60, scripts/server.js does not ensure that a pathname is located under the current working directory. NOTE: the Supplier disputes the significance of this report because the "Uncontrolled data used in path expression" occurs "in a development helper script whic...
CVE-2025-26791	✖	2025-02-14	MITRE	DOMPurify before 3.2.4 has an incorrect template literal regular expression, sometimes leading to mutation cross-site scripting (mXSS).
CVE-2024-53847	✖	2024-12-09	GitHub, Inc.	The Trix rich text editor, prior to versions 2.1.9 and 1.3.3, is vulnerable to cross-site scripting (XSS) + mutation XSS attacks when pasting malicious code. An attacker could trick a user to copy and paste malicious code that would execute arbitrary JavaScript code within the context of the user's ...
CVE-2024-48910	✖	2024-10-31	GitHub, Inc.	DOMPurify is a DOM-only, super-fast, uber-tolerant XSS sanitizer for HTML, MathML and SVG. DOMPurify was vulnerable to prototype pollution. This vulnerability is fixed in 2.4.2.

Figure 31: Identification of Vulnerability in the NVD CVE Database



CVE-2025-26791 Detail

AWAITING ANALYSIS

This CVE record has been marked for NVD enrichment efforts.

Description

DOMPurify before 3.2.4 has an incorrect template literal regular expression, sometimes leading to mutation cross-site scripting (mXSS).

Metrics

CVSS Version 4.0 | CVSS Version 3.x | CVSS Version 2.0

NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.

CVSS 3.x Severity and Vector Strings:

NIST: NVD	**Base Score:** N/A	NVD assessment not yet provided.
CNA: MITRE	**Base Score:** 4.5 MEDIUM	**Vector:** CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:C/C:L/I:L/A:N

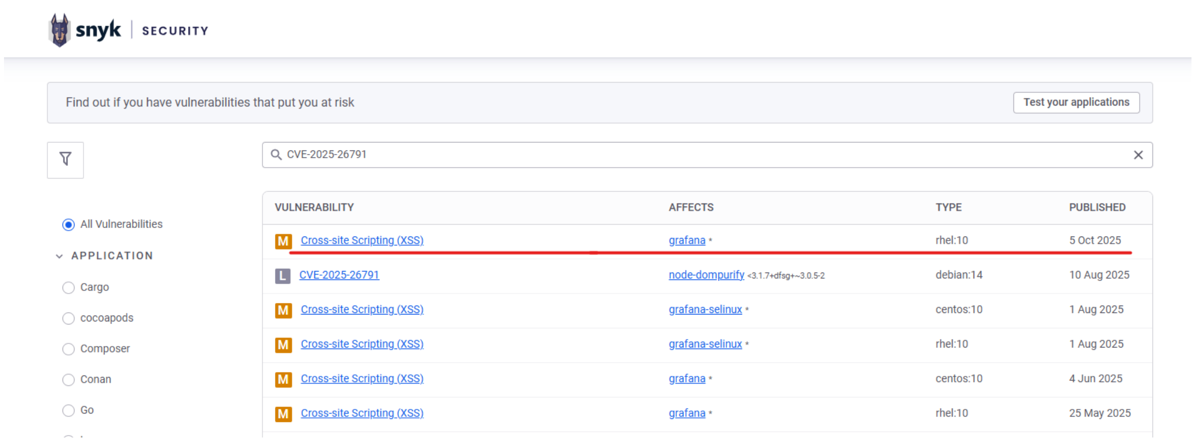
References to Advisories, Solutions, and Tools

By selecting these links, you will be leaving NIST webspace. We have provided these links to other web sites because they may have information that would be of interest to you. No inferences should be drawn on account of other sites being referenced, or not, from this page. There may be other web sites that are more appropriate for your purpose. NIST does not necessarily endorse the views expressed, or concur with the facts presented on these sites. Further, NIST does not endorse any commercial products that may be mentioned on these sites. Please address comments about this page to nvd@nist.gov.

QUICK INFO

CVE Dictionary Entry:
CVE-2025-26791
NVD Published Date:
02/14/2025
NVD Last Modified:
02/14/2025
Source:
MITRE

Figure 32: In-depth detail of the vulnerability described in the NVD CVE database



snyk | SECURITY

Find out if you have vulnerabilities that put you at risk

Test your applications

CVE-2025-26791

All Vulnerabilities

APPLICATION

Cargo

cocoapods

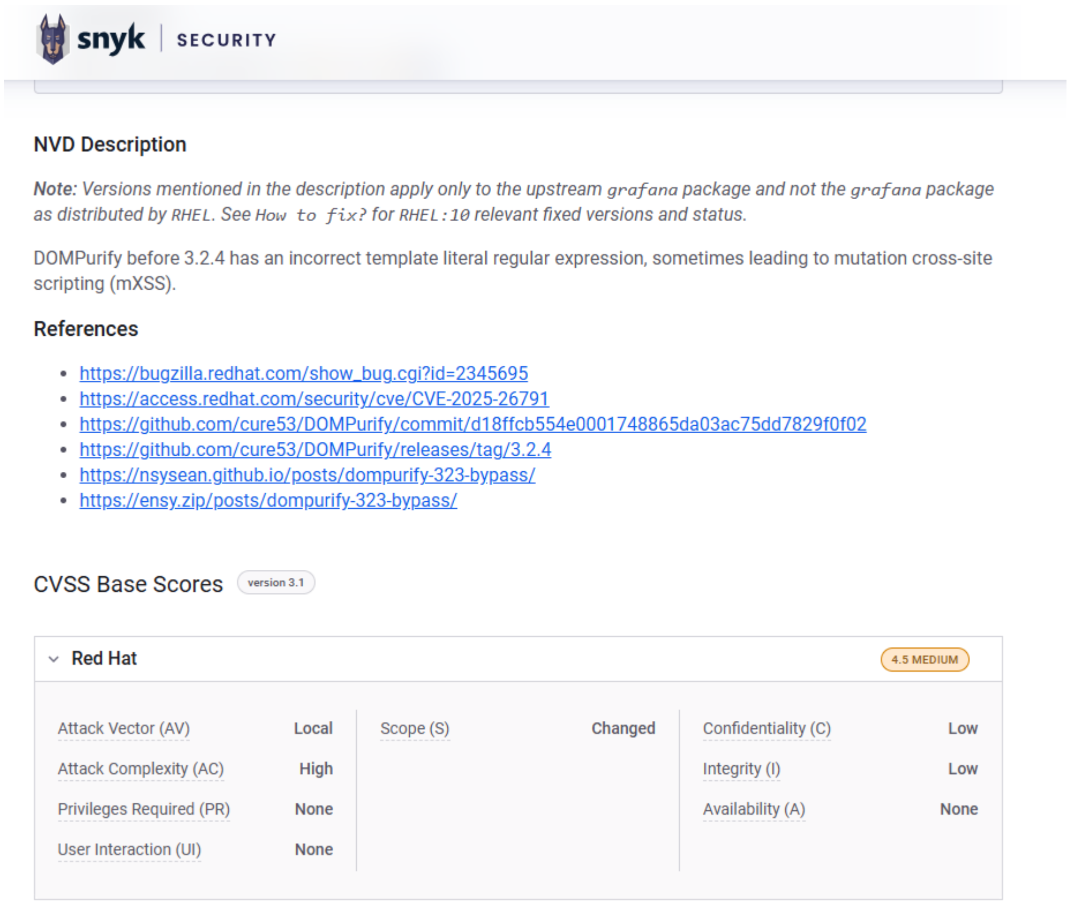
Composer

Conan

Go

VULNERABILITY	AFFECTS	TYPE	PUBLISHED
M Cross-site Scripting (XSS)	grafana *	rhel:10	5 Oct 2025
L CVE-2025-26791	node-dompurify <3.1.7+dfsg+~3.0.5-2	debian:14	10 Aug 2025
M Cross-site Scripting (XSS)	grafana-selinux *	centos:10	1 Aug 2025
M Cross-site Scripting (XSS)	grafana-selinux *	rhel:10	1 Aug 2025
M Cross-site Scripting (XSS)	grafana *	centos:10	4 Jun 2025
M Cross-site Scripting (XSS)	grafana *	rhel:10	25 May 2025

Figure 34: Vulnerability identification using Snyk Security



snyk | SECURITY

NVD Description

Note: Versions mentioned in the description apply only to the upstream `grafana` package and not the `grafana` package as distributed by `RHEL`. See `How to fix?` for `RHEL:10` relevant fixed versions and status.

DOMPurify before 3.2.4 has an incorrect template literal regular expression, sometimes leading to mutation cross-site scripting (mXSS).

References

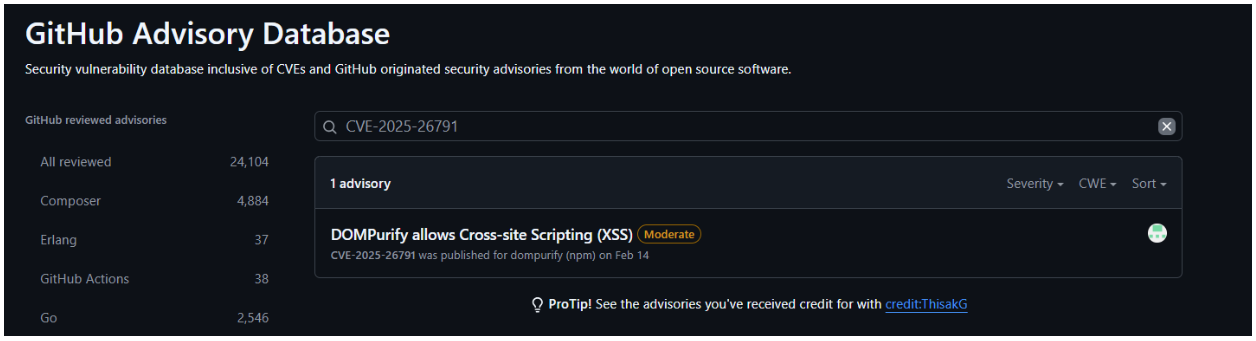
- https://bugzilla.redhat.com/show_bug.cgi?id=2345695
- https://access.redhat.com/security/cve/CVE-2025-26791
- https://github.com/cure53/DOMPurify/commit/d18ffcb554e0001748865da03ac75dd7829f0f02
- https://github.com/cure53/DOMPurify/releases/tag/3.2.4
- https://nsysean.github.io/posts/dompurify-323-bypass/
- https://ensy.zip/posts/dompurify-323-bypass/

CVSS Base Scores version 3.1

Red Hat 4.5 MEDIUM

Attack Vector (AV)	Local	Scope (S)	Changed	Confidentiality (C)	Low
Attack Complexity (AC)	High			Integrity (I)	Low
Privileges Required (PR)	None			Availability (A)	None
User Interaction (UI)	None				

Figure 334: Vulnerability description and severity rating in Snyk security database



GitHub Advisory Database

Security vulnerability database inclusive of CVEs and GitHub originated security advisories from the world of open source software.

GitHub reviewed advisories

All reviewed	24,104
Composer	4,884
Erlang	37
GitHub Actions	38
Go	2,546

CVE-2025-26791

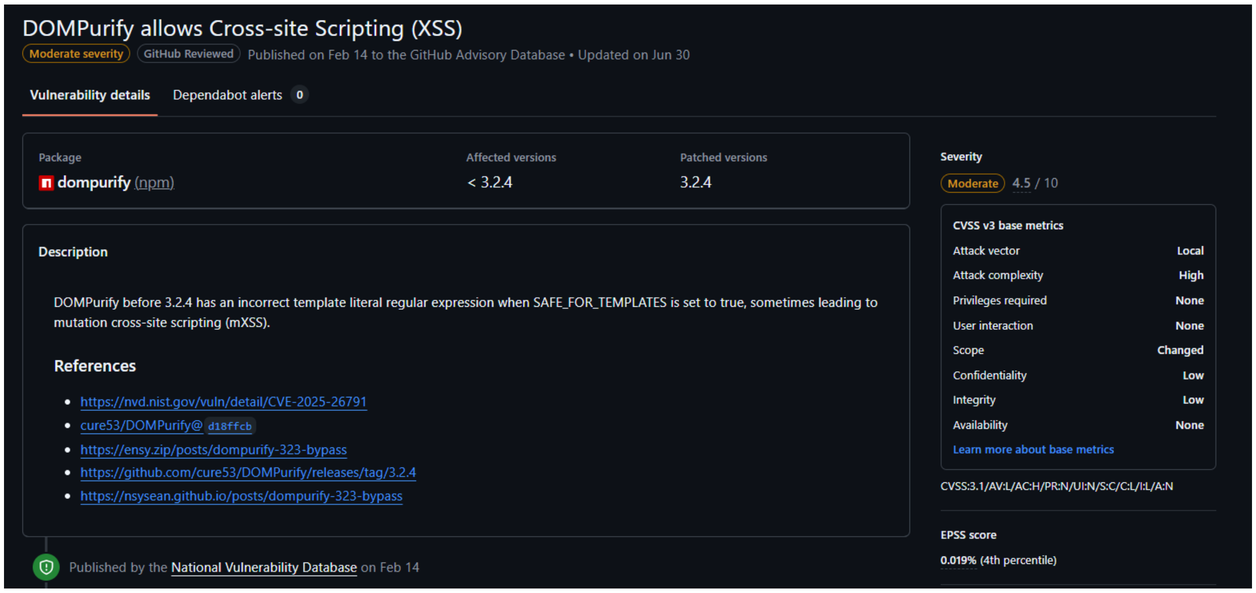
1 advisory

DOMPurify allows Cross-site Scripting (XSS) Moderate

CVE-2025-26791 was published for dompurify (npm) on Feb 14

ProTip! See the advisories you've received credit for with credit:ThisakG

Figure 35: Vulnerability identification using GitHub Advisory



DOMPurify allows Cross-site Scripting (XSS)

Moderate severity · GitHub Reviewed · Published on Feb 14 to the GitHub Advisory Database • Updated on Jun 30

Package	Affected versions	Patched versions
dompurify (npm)	< 3.2.4	3.2.4

Description

DOMPurify before 3.2.4 has an incorrect template literal regular expression when SAFE_FOR_TEMPLATES is set to true, sometimes leading to mutation cross-site scripting (mXSS).

References

- https://nvd.nist.gov/vuln/detail/CVE-2025-26791
- cure53/DOMPurify@ d18ffcb
- https://ensy.zip/posts/dompurify-323-bypass
- https://github.com/cure53/DOMPurify/releases/tag/3.2.4
- https://nsysean.github.io/posts/dompurify-323-bypass

Published by the National Vulnerability Database on Feb 14

Severity

Moderate 4.5 / 10

CVSS v3 base metrics

Attack vector	Local
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Changed
Confidentiality	Low
Integrity	Low
Availability	None

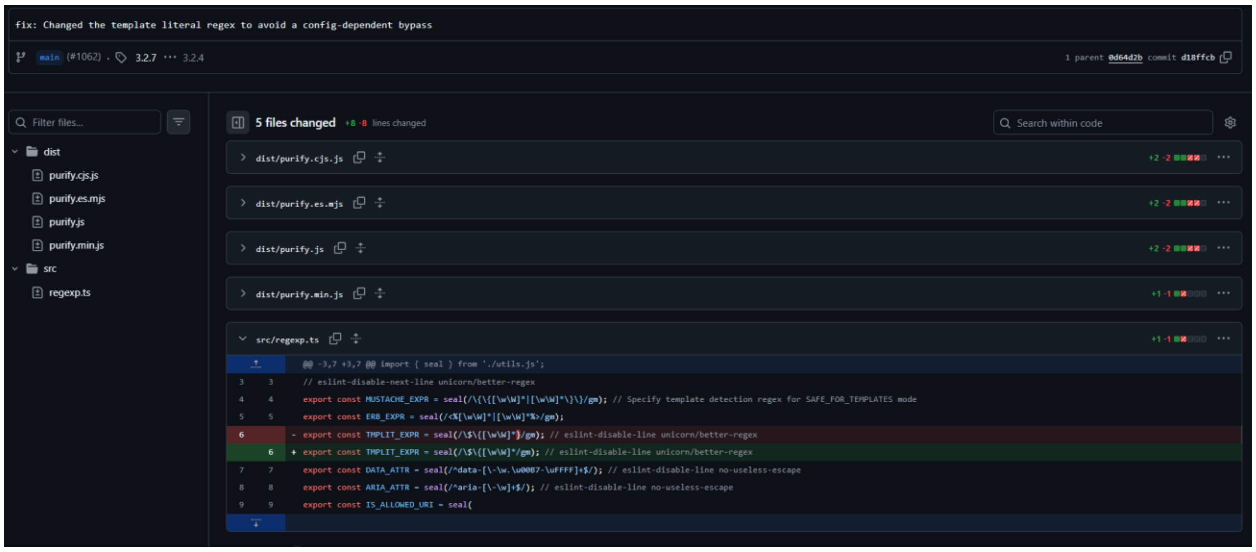
Learn more about base metrics

CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:C/C:L/I:L/A:N

EPSS score

0.019% (4th percentile)

Figure 36: Vulnerability description in GitHub Advisory Repository



fix: Changed the template literal regex to avoid a config-dependent bypass

main (#1062) · 3.2.7 ··· 3.2.4 — 1 parent 0d64d2b commit d18ffcb

5 files changed +8 -8 lines changed

dist/purify.cjs.js	+2 -2
dist/purify.es.mjs	+2 -2
dist/purify.js	+2 -2
dist/purify.min.js	+1 -1
src/regexp.ts	+1 -1

```
@@ -3,7 +3,7 @@ import { seal } from './utils.js';
  // eslint-disable-next-line unicorn/better-regex
  export const MUSTACHE_EXPR = seal(/\{\{[\w\W]*|[\w\W]*\}\}/gm); // Specify template detection regex for SAFE_FOR_TEMPLATES mode
  export const ERB_EXPR = seal(/<%[\w\W]*|[\w\W]*%>/gm);
- export const TMPLIT_EXPR = seal(/\$\{[\w\W]*}/gm); // eslint-disable-line unicorn/better-regex
+ export const TMPLIT_EXPR = seal(/\$\{[\w\W]*/gm); // eslint-disable-line unicorn/better-regex
  export const DATA_ATTR = seal(/^data-[\-\w.\u00B7-\uFFFF]+$/); // eslint-disable-line no-useless-escape
  export const ARIA_ATTR = seal(/^aria-[\-\w]+$/); // eslint-disable-line no-useless-escape
  export const IS_ALLOWED_URI = seal(
```

Figure 37: Mitigation strategies set by contributors

Exploitation:

An example of an attack in action can be described in the following steps:

1. An attacker finds an input point that accepts HTML such as a profile bio, a project name, comments, a markdown editor, or embedded fields.
2. The attacker submits a payload that the vulnerable DOMPurify fails to fully sanitize (the payload depends on the exact bypass, such as encoded/backtick/template payloads).
3. Then if the application later renders the stored value using innerHTML or a dangerous rendering path, the payload executes in victims' browsers, exploiting the stored XSS.
4. After a successful exploit, the attacker can steal session cookies, perform actions as the victim if no CSRF protection is available, or even exfiltrate sensitive data.

Impact of having un-updated JS libraries:

- Client-side code execution (XSS): executing JS in context of site can lead to session theft, account takeover and even data exfiltration.
- Phishing attacks: an attacker can inject UI to trick users into revealing credentials or performing actions.
- Supply chain risk: using an outdated sanitizer library can increase the attack surface for future changes or other components that reuse sanitize().

Mitigation Strategies:

- Upgrading DOMPurify to version 3.2.4 which is the latest version that directly addresses known CVEs.
- Auditing all uses of DOMPurify.sanitize() in the codebase:
    - Avoiding dangerouslySetInnerHTML and innerHTML wherever possible.
    - Not passing SAFE_FOR_TEMPLATES or relaxed options unless absolutely required and reviewed.
    - Ensuring ALLOWED_TAGS / FORBID_ATTR are explicitly set if app needs limited HTML.
- Implementing Content-Security-Policy (CSP): using default-src 'self'; script-src 'self' 'nonce-...' to mitigate impact of any XSS that bypasses sanitization.
- Taking use of HttpOnly cookies for auth tokens, using SameSite=strict/lax appropriately, rotating sessions on login and invalidating sessions upon suspicious activity.

2.3.1 Finding 03: Missing baseline security headers (X-Content-Type-Options, Referrer-Policy, Permissions-Policy)

Vulnerability Title:

Missing baseline HTTP security headers: X-Content-Type-Options: nosniff, Referrer-Policy, and Permissions-Policy are not present on responses from the site.

Description:

The application does not set several baseline security headers on HTTP responses:

- X-Content-Type-Options: nosniff which prevents MIME-type sniffing by browsers and reduces risk of executing resources with an incorrect Content-Type.
- Referrer-Policy which controls what referrer information the browser sends; without it, full URL referrers (including query strings) may leak to third parties.
- Permissions-Policy which restricts which powerful browser features (e.g., geolocation, camera, fullscreen, payment) web pages and iframes may use.

Their absence is a configuration weakness that increases attack surface and privacy exposure. Each missing header is a low-to-medium severity misconfiguration but together they degrade defenses set in place in the platform.

Vulnerability Discovery:

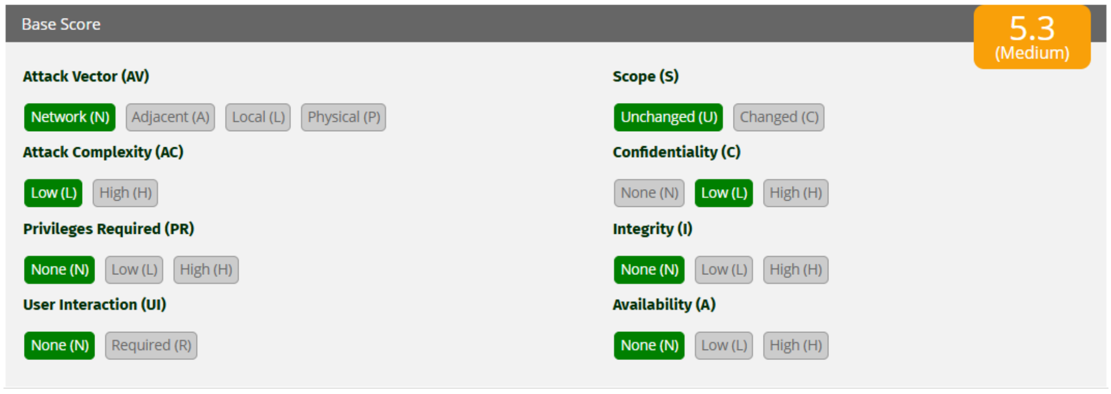
I inspected HTTP response headers for the target using curl -I and browser DevTools Network panel. The requests to primary pages such as the dashboard and user profiles show HSTS and X-Frame-Options present, but the three baseline headers above are missing.

To make sure in inspected the responses after sending requests through the intercept tab in BurpSuite, and the results were same as above.

Severity Rating: Medium

A CVSS common vulnerability scoring system was used to estimate the severity of this vulnerability. (https://www.first.org/cvss/calculator/3-0).

A – Missing X-Content-Type-Options: nosniff:



Base Score 5.3 (Medium)

Attack Vector (AV)
[✓ Network (N)] Adjacent (A) Local (L) Physical (P)

Attack Complexity (AC)
[✓ Low (L)] High (H)

Privileges Required (PR)
[✓ None (N)] Low (L) High (H)

User Interaction (UI)
[✓ None (N)] Required (R)

Scope (S)
[✓ Unchanged (U)] Changed (C)

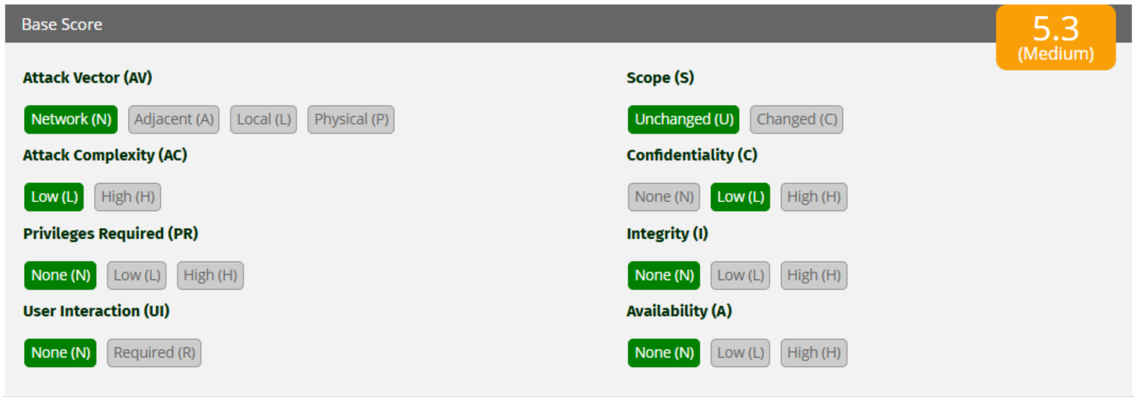
Confidentiality (C)
None (N) [✓ Low (L)] High (H)

Integrity (I)
[✓ None (N)] Low (L) High (H)

Availability (A)
[✓ None (N)] Low (L) High (H)

Figure 38: CVSS value of missing X-Content-Type-Options nosniff

B - Missing Referrer Policy:



Base Score 5.3 (Medium)

Attack Vector (AV)
[✓ Network (N)] Adjacent (A) Local (L) Physical (P)

Attack Complexity (AC)
[✓ Low (L)] High (H)

Privileges Required (PR)
[✓ None (N)] Low (L) High (H)

User Interaction (UI)
[✓ None (N)] Required (R)

Scope (S)
[✓ Unchanged (U)] Changed (C)

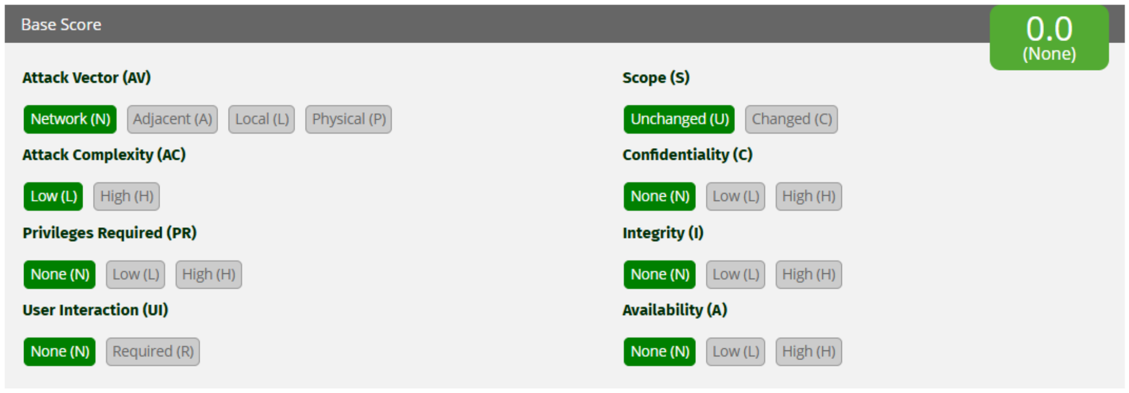
Confidentiality (C)
None (N) [✓ Low (L)] High (H)

Integrity (I)
[✓ None (N)] Low (L) High (H)

Availability (A)
[✓ None (N)] Low (L) High (H)

Figure 39: CVSS value of missing Referrer Policy

C – Missing Permissions Policy:



Base Score 0.0 (None)

Attack Vector (AV)
[✓ Network (N)] Adjacent (A) Local (L) Physical (P)

Attack Complexity (AC)
[✓ Low (L)] High (H)

Privileges Required (PR)
[✓ None (N)] Low (L) High (H)

User Interaction (UI)
[✓ None (N)] Required (R)

Scope (S)
[✓ Unchanged (U)] Changed (C)

Confidentiality (C)
[✓ None (N)] Low (L) High (H)

Integrity (I)
[✓ None (N)] Low (L) High (H)

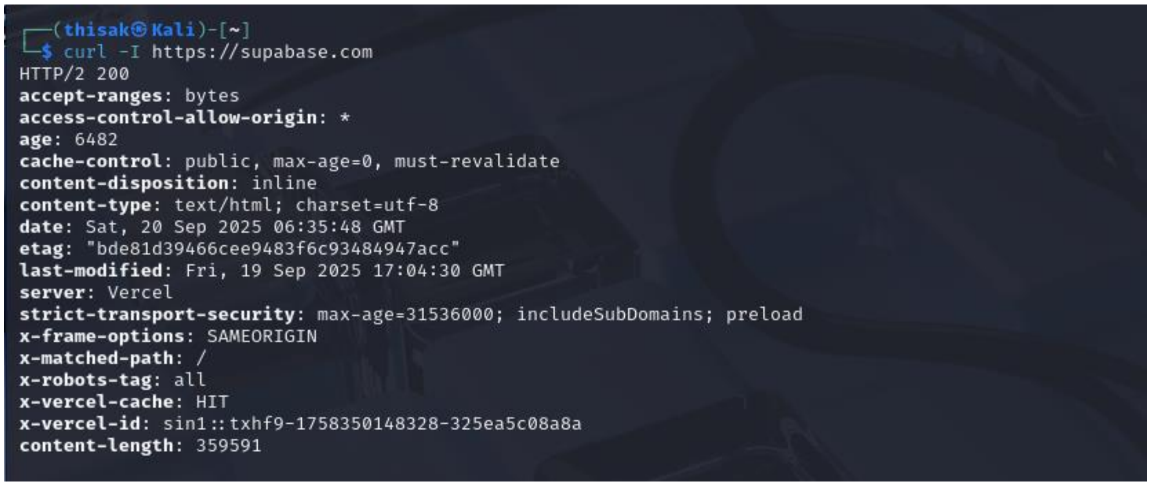
Availability (A)
[✓ None (N)] Low (L) High (H)

Figure 40: CVSS value of missing Permissions Policy

Proof of Concept (PoC):

This PoC is locally created to demonstrate the effect of missing headers using my own browser to stay within the program scope and not accidentally modify any code within the production environment.

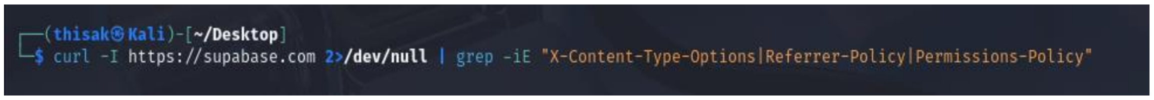
1. Firstly, I used the command line tool ‘curl’ to find out the server responses that the target platform generally has through a passive header inspection.



```
┌──(thisak㉿Kali)-[~]
└─$ curl -I https://supabase.com
HTTP/2 200
accept-ranges: bytes
access-control-allow-origin: *
age: 6482
cache-control: public, max-age=0, must-revalidate
content-disposition: inline
content-type: text/html; charset=utf-8
date: Sat, 20 Sep 2025 06:35:48 GMT
etag: "bde81d39466cee9483f6c93484947acc"
last-modified: Fri, 19 Sep 2025 17:04:30 GMT
server: Vercel
strict-transport-security: max-age=31536000; includeSubDomains; preload
x-frame-options: SAMEORIGIN
x-matched-path: /
x-robots-tag: all
x-vercel-cache: HIT
x-vercel-id: sin1::txhf9-1758350148328-325ea5c08a8a
content-length: 359591
```

Figure 41: using 'curl' command for header inspection

2. Next, the ‘grep’ command was utilized to filter the output to confirm that the baseline headers were truly absent. (no output upon command execution proves this).

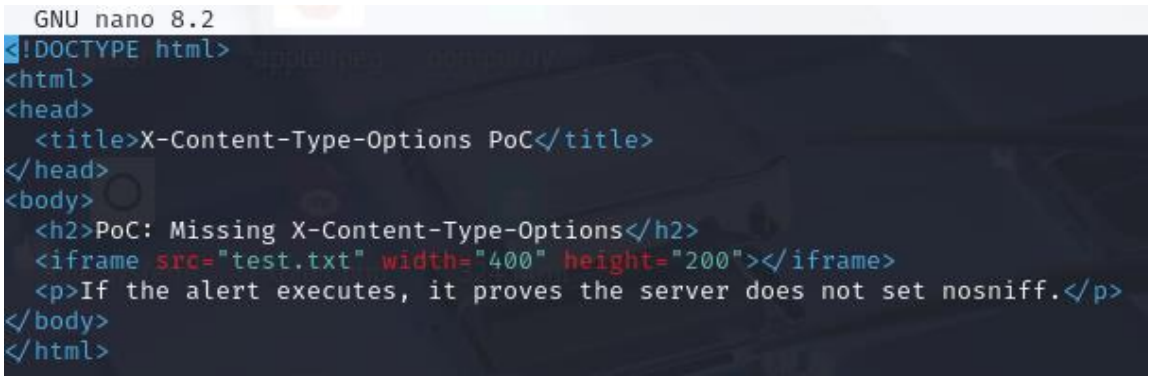


```
┌──(thisak㉿Kali)-[~/Desktop]
└─$ curl -I https://supabase.com 2>/dev/null | grep -iE "X-Content-Type-Options|Referrer-Policy|Permissions-Policy"
```

Figure 42: using 'curl' command along with 'grep' to see if base line headers are present

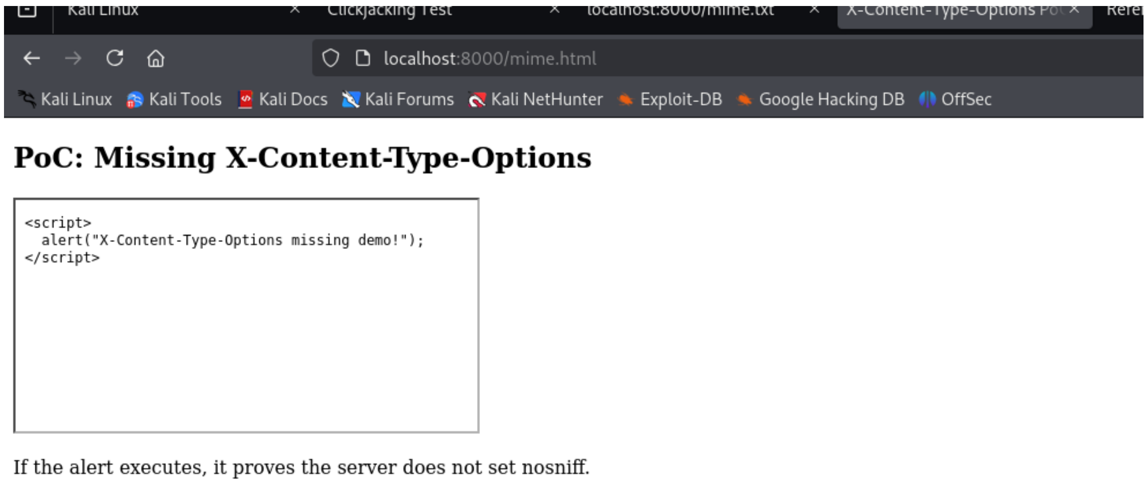
3. To illustrate the impact of the absent X-Content-Type-Options: nosniff header, a locally hosted HTML page was created. This page included an <iframe> referencing a text file (mime.txt) containing JavaScript code.

   When loaded in the browser, the execution of the embedded script was observed. Since the nosniff header was not set, certain browsers may MIME-sniff the content and execute the script despite the text/plain content type.



```
GNU nano 8.2
<!DOCTYPE html>
<html>
<head>
  <title>X-Content-Type-Options PoC</title>
</head>
<body>
  <h2>PoC: Missing X-Content-Type-Options</h2>
  <iframe src="test.txt" width="400" height="200"></iframe>
  <p>If the alert executes, it proves the server does not set nosniff.</p>
</body>
</html>
```

Figure 36: HTML code to understand the effect of missing X-Content-Type-Options



PoC: Missing X-Content-Type-Options

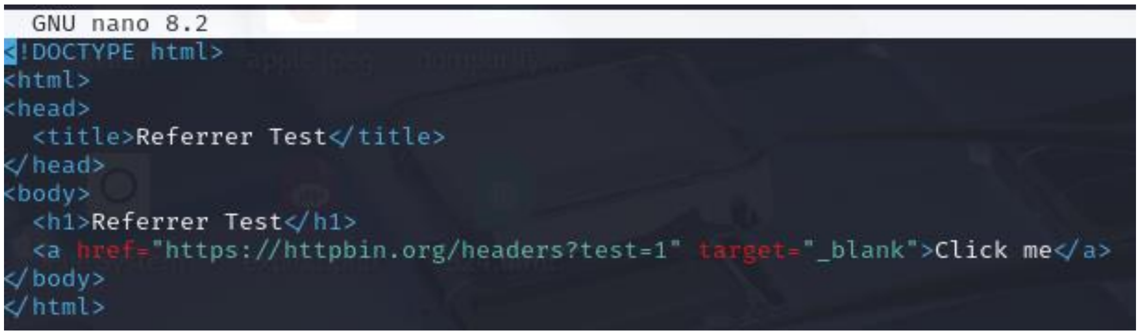
```
<script>
  alert("X-Content-Type-Options missing demo!");
</script>
```

If the alert executes, it proves the server does not set nosniff.

Figure 44: result of above code

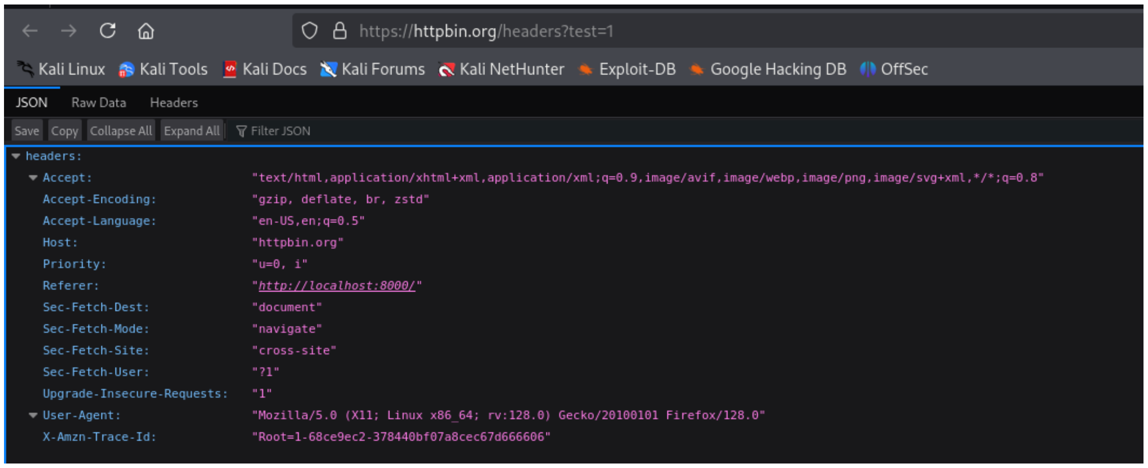
4. To demonstrate the effect of missing Referrer Policy header, the below experiment was conducted using an HTML page that was locally hosted. This page contained a simple link to ‘https://httpbin.org/headers?test=1’ with the target="_blank" attribute. When the link was clicked, the HTTP request sent to the destination was inspected to observe the Referrer header.

   Without a Referrer Policy set by the server, the browser sends the full URL including any query parameters in the Referrer header. The output captured from httpbin.org showed the full URL being transmitted, showing how sensitive information in the URL could be leaked if a proper Referrer Policy is not implemented.



```
GNU nano 8.2
<!DOCTYPE html>
<html>
<head>
  <title>Referrer Test</title>
</head>
<body>
  <h1>Referrer Test</h1>
  <a href="https://httpbin.org/headers?test=1" target="_blank">Click me</a>
</body>
</html>
```

Figure 38: HTML code to understand the effect of missing referrer policy

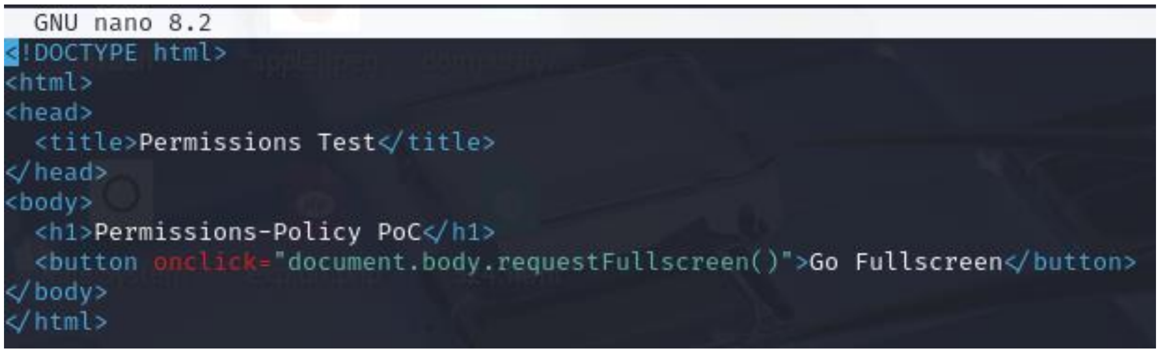


```
▼ headers:
  ▼ Accept:                     "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8"
    Accept-Encoding:            "gzip, deflate, br, zstd"
    Accept-Language:            "en-US,en;q=0.5"
    Host:                       "httpbin.org"
    Priority:                   "u=0, i"
    Referer:                    "http://localhost:8000/"
    Sec-Fetch-Dest:             "document"
    Sec-Fetch-Mode:             "navigate"
    Sec-Fetch-Site:             "cross-site"
    Sec-Fetch-User:             "?1"
    Upgrade-Insecure-Requests:  "1"
  ▼ User-Agent:                 "Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
    X-Amzn-Trace-Id:            "Root=1-68ce9ec2-378440bf07a8cec67d666606"
```

Figure 37: result of above code

5. To demonstrate the effect of missing Permissions Policy header, the below experiment was conducted using an HTML page that was locally hosted. The page contained a button that, when clicked, triggered the requestFullscreen() method to attempt fullscreen mode for the page content. Without a restrictive Permissions-Policy header in place, the browser allowed the fullscreen request to succeed.

   This demonstrates that, in the absence of a properly configured Permissions-Policy, certain sensitive APIs or features (like fullscreen, camera, or microphone access) may be accessible to page content or embedded frames without explicit control. Console logs and observed behavior were recorded to provide evidence of the potential security and privacy implications.



```
GNU nano 8.2
<!DOCTYPE html>
<html>
<head>
  <title>Permissions Test</title>
</head>
<body>
  <h1>Permissions-Policy PoC</h1>
  <button onclick="document.body.requestFullscreen()">Go Fullscreen</button>
</body>
</html>
```

Figure 39: HTML code to understand the effect of missing permissions policy

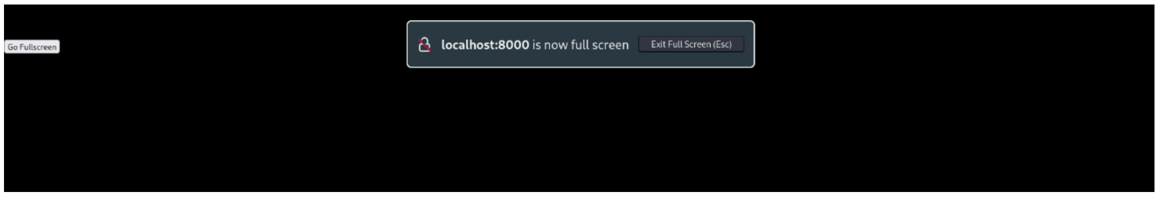


Figure 40: result of above code

Exploitation:

An example of an attack in action can be described in the following steps:

1. By missing nosniff, attackers can upload files to the server mislabels as Content-Type, some browsers may sniff and execute the content, allowing XSS via uploaded .txt files or other artifacts.

   Combined with other flaws like file upload that allows attacker-controlled content, this can lead to client-side code execution.

2. By missing Referrer-Policy, sensitive information in URLs like session tokens, API keys in query strings, or internal IDs can be leaked to third-party sites via Referrer headers when users click links. An attacker controlling a third-party site could collect these from victims.

3. By Missing Permissions-Policy, if the site is framed or includes third-party content, the lack of restrictions could allow that content to abuse browser features such as fullscreen, autoplay, payments or even sensors in ways that can facilitate phishing, UX abuse, or privacy leaks.

   All of these are enablers for more serious attacks when combined with other attacks such as XSS, bad file upload handling, or even open redirection.

Impact of Missing Baseline Headers:

- Privacy violations like leakage of full URLs and query strings to third parties (tokens, user IDs).

- An increased attack surface provides more opportunities to exploit XSS, file upload flaws, or UI redress techniques.

- Loss trust and Compliance due to missing headers reducing hardening that may fail security baseline checks.

Mitigation Strategies:

- Adding headers at the edge (CDN / reverse proxy) by configuring the CDN to inject these headers on all responses
    - X-Content-Type-Options: nosniff
    - Referrer-Policy: strict-origin-when-cross-origin
    - Permissions-Policy: geolocation=(), microphone=(), camera=(), fullscreen=(self), autoplay=()

- Adding to the web server config (nginx/Apache) if they cannot be set at the edge so static assets and HTML consistently include them.
    - add_header X-Content-Type-Options "nosniff" always
    - add_header Referrer-Policy "strict-origin-when-cross-origin" always;
    - add_header Permissions-Policy "geolocation=(), microphone=(), camera=(), fullscreen=(self), autoplay=()" always

- Ensuring the application frameworks and microservices also set the headers so JSON responses and dynamically generated pages aren’t missed.

- Hardening policies and monitoring, starting with strict-origin-when-cross-origin, restrict Permissions-Policy to only features used, and monitoring logs for any functional relapses.

3. <u>Conclusions and Reflections</u>

Learning Outcomes:

This security assessment provided valuable insights into modern web application security testing and the complications involved in authorized vulnerability research. Having conducted this assessment using a manual testing approach showed that having a systematic method of execution is essential for identifying security gaps that automated tools might miss out on.

Technical Knowledge Outcomes:

- Understanding how to operate various tools to gather information, analyze and simulate real-world attack scenarios.

- Learning how CSRF attacks operate, how protection mechanisms work in practice and why ‘SameSite’ cookie configurations are important.

- Understanding the importance of maintaining updated security-critical libraries.

- Developing skills in understanding HTTP security headers, how to analyze them and their role in web security.

Methodology Outcomes:

- Understanding reconnaissance and enumeration phases are crucial for understanding the attack surface and staying within scope boundaries.

- Getting to know that Burp Suite is one of the go-to tools for web analysis and is invaluable for documenting and validating security findings.

- Learning that manual testing often reveals business logic vulnerabilities that automated scanners cannot detect.

- Understanding proper documentation and evidence collection is as important as finding the vulnerabilities themselves.

Challenges Faced:

- **Scope Management:**
  The main challenge was maintaining clear boundaries between in-scope platform domains (*.supabase.com)* and out-of-scope customer domains *(*.supabase.co). This required constant checking to ensure testing remained within authorized settings while still conducting thorough security assessments.

- **Domain Infrastructure Analysis:**
  Supabase consisted of a distributed architecture with multiple subdomains, CDN configurations, and third-party integrations. This created a complex challenge in understanding the complete attack surface.

  Identifying what was testable and what was off-limits required detailed reconnaissance and informative judgment.

- **Evidence Collection:**
  Finding comprehensive proof-of-concept evidence while maintaining reproducible testing conditions meant needing careful coordination between testing tools.

  Each vulnerability took considerable time due to attention to detail ensuring that Burp Suite captures, screenshots, and code samples clearly demonstrated the flaws clearly.

- **Technical Limitations:**
  Unfamiliarity with certain tools and environments was challenging, particularly because this was not a controlled simulation but a real-world security assessment, where mistakes could carry serious legal and ethical consequences if handled improperly.

  Some testing approaches were limited due to the platform considering them as out of scope. This required creating safe and clearly defined test accounts and developing controlled testing methodologies that could show the vulnerabilities without affecting platform stability or other users' data.

- **Time Management:**
  Balancing other academic activities with this assessment was challenging, especially while self-learning new tools. This emphasized the need for timeboxing tasks and prioritizing findings with the most potential impact.

<u>Personal Reflections:</u>

- Improved confidence in testing real-world platforms responsibly and systematically as previous experiences were purely theoretical.

- Deeper understanding of ethical awareness, especially around ethical hacking, scope management and responsible disclosure.

- Better appreciation of preparation and recognizing the value of learning tools in advance to save time, rather than real-time learning and execution.

- Improvements in methodology, planning tests more efficiently before test conduction to maximize results.

- Seeing the connection to industry standards, tying lessons to OWASP Top 10 practices and current security expectations.