Sri Lanka Institute of Information Technology

Web Security – IE2062

Topic: Bug Bounty Report 4

Y2S2.WE.CS

Name: S.D.W.Gunaratne

(IT23241978)

Table of Content

1) How I started?

2) Introduction

2.1 Domain

1.2 Severity

3) Vulnerability

3.1 Vulnerability title

3.2 Vulnerability description

3.3 Affected components

3.4 Impact assessment

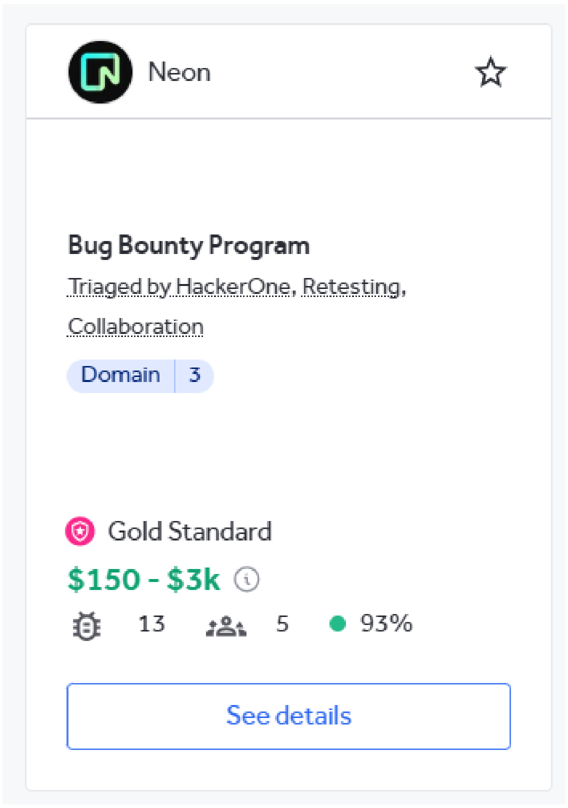
3.5 Steps to reproduce

3.6 Proof of concept

3.7 Proposed mitigation or fix

How I started?

1. Once I search from hacker one, I saw a Neon bug bounty program.



Neon

Bug Bounty Program

Triaged by HackerOne, Retesting, Collaboration

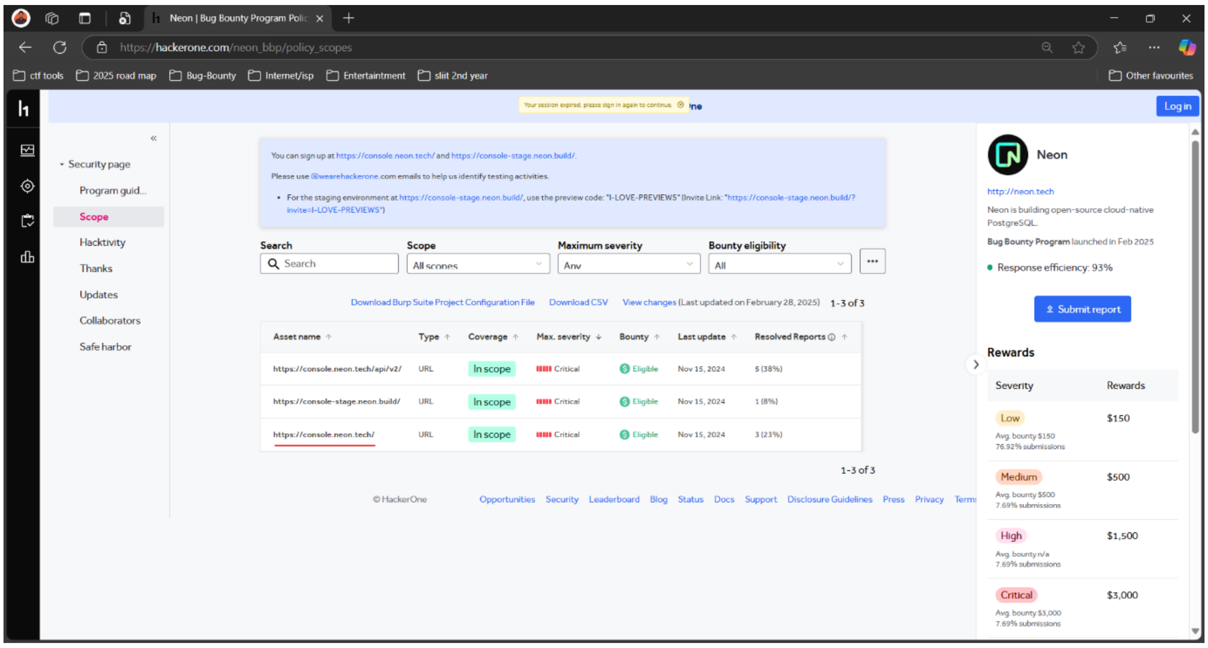
Domain | 3

Gold Standard

$150 - $3k

13 5 ● 93%

2. Then, I discovered allowed domains scope, so that I choose https://console.neon.tech/.



You can sign up at https://console.neon.tech/ and https://console-stage.neon.build/.

Please use @wearehackerone.com emails to help us identify testing activities.

- For the staging environment at https://console-stage.neon.build/, use the preview code: "I-LOVE-PREVIEWS" (Invite Link: "https://console-stage.neon.build/?invite=I-LOVE-PREVIEWS")

Search: Search | Scope: All scopes | Maximum severity: Any | Bounty eligibility: All

Download Burp Suite Project Configuration File Download CSV View changes (Last updated on February 28, 2025) 1-3 of 3

Asset name	Type	Coverage	Max. severity	Bounty	Last update	Resolved Reports
https://console.neon.tech/api/v2/	URL	In scope	Critical	Eligible	Nov 15, 2024	5 (38%)
https://console-stage.neon.build/	URL	In scope	Critical	Eligible	Nov 15, 2024	1 (8%)
https://console.neon.tech/	URL	In scope	Critical	Eligible	Nov 15, 2024	3 (23%)

1-3 of 3

© HackerOne Opportunities Security Leaderboard Blog Status Docs Support Disclosure Guidelines Press Privacy Terms

Neon

http://neon.tech

Neon is building open-source cloud-native PostgreSQL.

Bug Bounty Program launched in Feb 2025

● Response efficiency: 93%

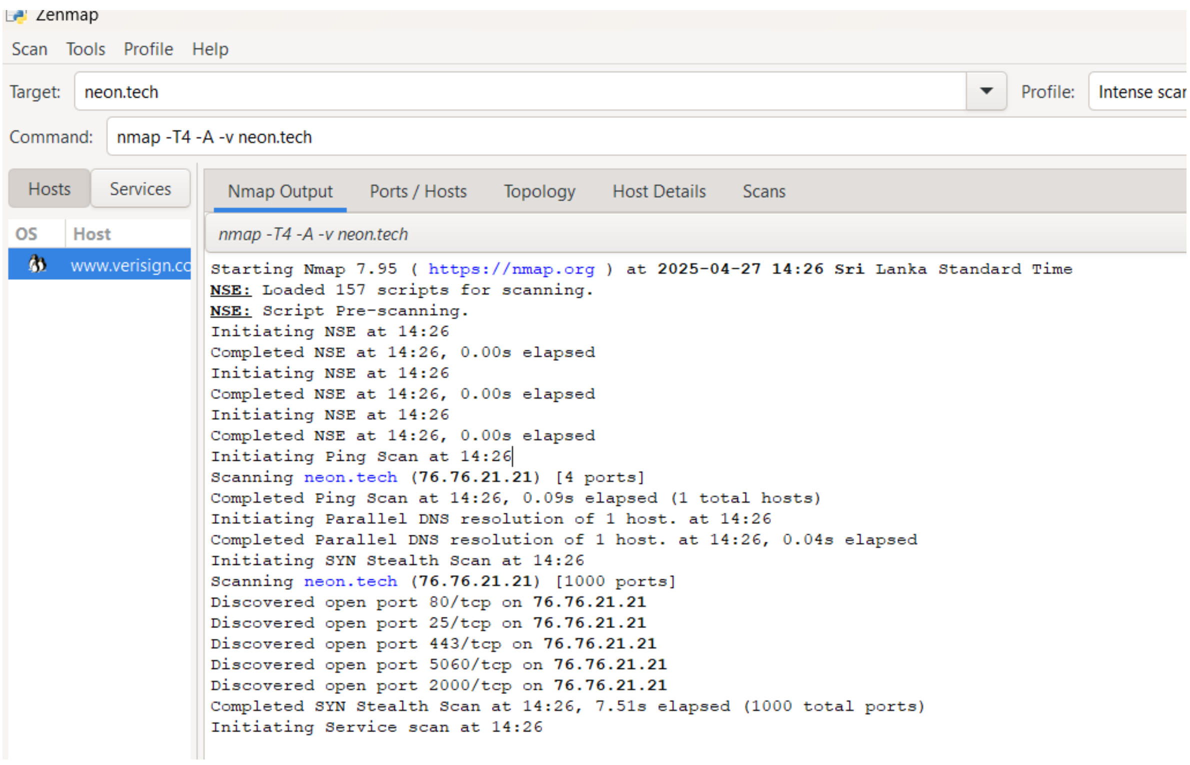
Submit report

Rewards

Severity	Rewards
Low Avg. bounty $150 76.92% submissions	$150
Medium Avg. bounty $500 7.69% submissions	$500
High Avg. bounty n/a 7.69% submissions	$1,500
Critical Avg. bounty $3,000 7.69% submissions	$3,000

3. I use several methods/tools to do penetration testing.

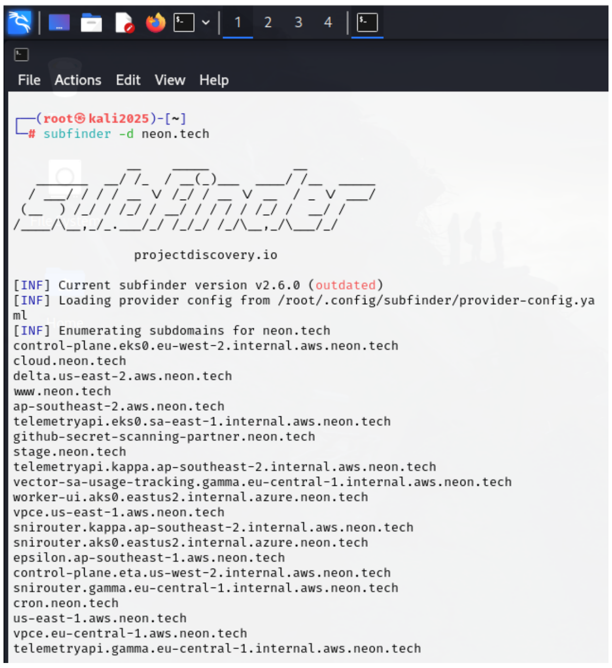
4. First, I used Nmap. It helps me to find what are the open ports, Identify the web technologies such as webservers.



Target: neon.tech
Profile: Intense scan
Command: nmap -T4 -A -v neon.tech

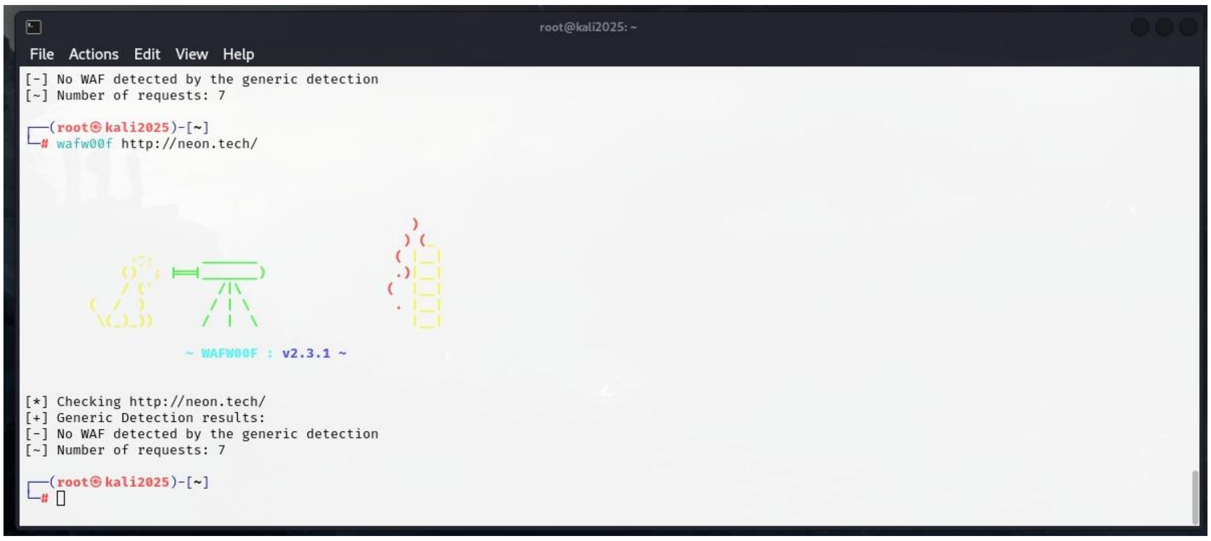
```
nmap -T4 -A -v neon.tech

Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-27 14:26 Sri Lanka Standard Time
NSE: Loaded 157 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 14:26
Completed NSE at 14:26, 0.00s elapsed
Initiating NSE at 14:26
Completed NSE at 14:26, 0.00s elapsed
Initiating NSE at 14:26
Completed NSE at 14:26, 0.00s elapsed
Initiating Ping Scan at 14:26
Scanning neon.tech (76.76.21.21) [4 ports]
Completed Ping Scan at 14:26, 0.09s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 14:26
Completed Parallel DNS resolution of 1 host. at 14:26, 0.04s elapsed
Initiating SYN Stealth Scan at 14:26
Scanning neon.tech (76.76.21.21) [1000 ports]
Discovered open port 80/tcp on 76.76.21.21
Discovered open port 25/tcp on 76.76.21.21
Discovered open port 443/tcp on 76.76.21.21
Discovered open port 5060/tcp on 76.76.21.21
Discovered open port 2000/tcp on 76.76.21.21
Completed SYN Stealth Scan at 14:26, 7.51s elapsed (1000 total ports)
Initiating Service scan at 14:26
```

5. Secondly, I used Subfider tool to find hidden or forgotten web asserts. Because hidden web assert can have poor security, unpatched vulnerabilities.



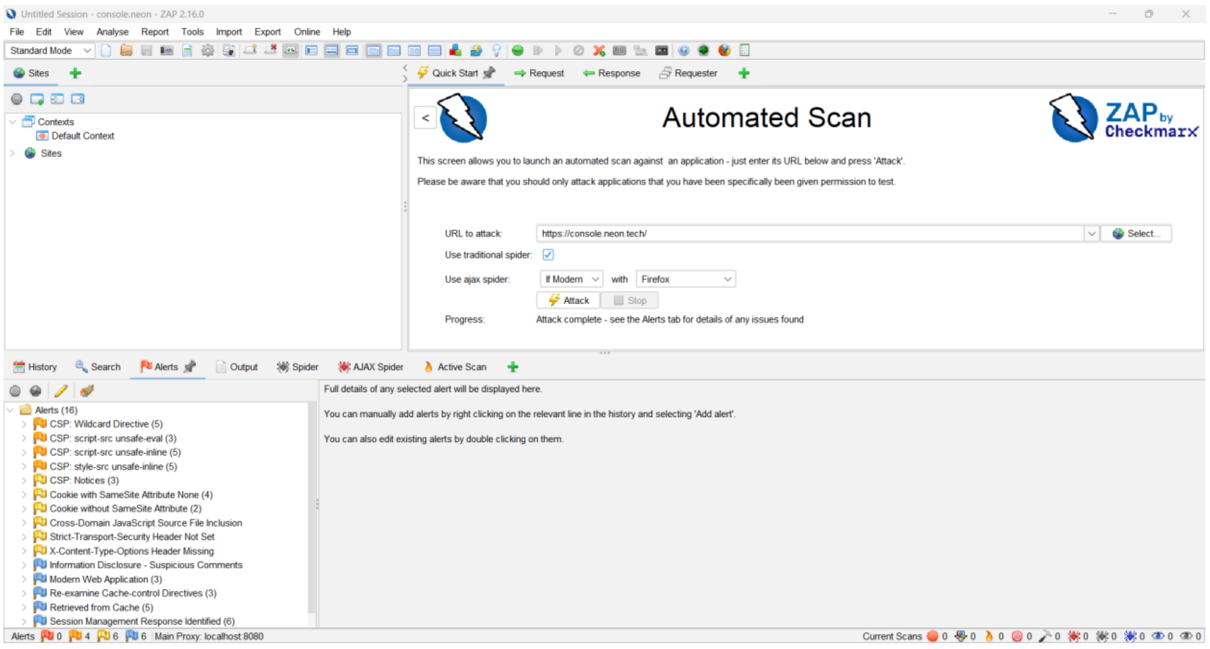
```
┌──(root㉿kali2025)-[~]
└─# subfinder -d neon.tech

                projectdiscovery.io

[INF] Current subfinder version v2.6.0 (outdated)
[INF] Loading provider config from /root/.config/subfinder/provider-config.ya
ml
[INF] Enumerating subdomains for neon.tech
control-plane.eks0.eu-west-2.internal.aws.neon.tech
cloud.neon.tech
delta.us-east-2.aws.neon.tech
www.neon.tech
ap-southeast-2.aws.neon.tech
telemetryapi.eks0.sa-east-1.internal.aws.neon.tech
github-secret-scanning-partner.neon.tech
stage.neon.tech
telemetryapi.kappa.ap-southeast-2.internal.aws.neon.tech
vector-sa-usage-tracking.gamma.eu-central-1.internal.aws.neon.tech
worker-ui.aks0.eastus2.internal.azure.neon.tech
vpce.us-east-1.aws.neon.tech
snirouter.kappa.ap-southeast-2.internal.aws.neon.tech
snirouter.aks0.eastus2.internal.azure.neon.tech
epsilon.ap-southeast-1.aws.neon.tech
control-plane.eta.us-west-2.internal.aws.neon.tech
snirouter.gamma.eu-central-1.internal.aws.neon.tech
cron.neon.tech
us-east-1.aws.neon.tech
vpce.eu-central-1.aws.neon.tech
telemetryapi.gamma.eu-central-1.internal.aws.neon.tech
```

6. Thirdly, I used Wafwoof tool to find website is protected by a WAF (web application firewall). Because if WAF is active, so pen tester do their test without blocked, and they can do their testing with bypass WAF.



```
[-] No WAF detected by the generic detection
[~] Number of requests: 7

┌──(root㉿kali2025)-[~]
└─# wafw00f http://neon.tech/

                    ~ WAFW00F : v2.3.1 ~

[*] Checking http://neon.tech/
[+] Generic Detection results:
[-] No WAF detected by the generic detection
[~] Number of requests: 7

┌──(root㉿kali2025)-[~]
└─#
```

7.Finaly, I use OWASP zap to automatically find the vulnerabilities.


Untitled Session - console.neon - ZAP 2.16.0
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Quick Start Request Response Requester
Automated Scan
ZAP by Checkmarx
History Search Alerts Output Spider AJAX Spider Active Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: https://console.neon.tech/
Use traditional spider: ✓
Use ajax spider: If Modern with Firefox
Progress: Attack complete - see the Alerts tab for details of any issues found

- Contexts
  - Default Context
- Sites

- Alerts (16)
  - CSP: Wildcard Directive (5)
  - CSP: script-src unsafe-eval (3)
  - CSP: script-src unsafe-inline (5)
  - CSP: style-src unsafe-inline (5)
  - CSP: Notices (3)
  - Cookie with SameSite Attribute None (4)
  - Cookie without SameSite Attribute (2)
  - Cross-Domain JavaScript Source File Inclusion
  - Strict-Transport-Security Header Not Set
  - X-Content-Type-Options Header Missing
  - Information Disclosure - Suspicious Comments
  - Modern Web Application (3)
  - Re-examine Cache-control Directives (3)
  - Retrieved from Cache (5)
  - Session Management Response Identified (6)

Full details of any selected alert will be displayed here.

You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.

You can also edit existing alerts by double clicking on them.

Alerts 0 4 6 6 Main Proxy: localhost:8080

Current Scans 0 0 0 0 0 0 0 0 0 0

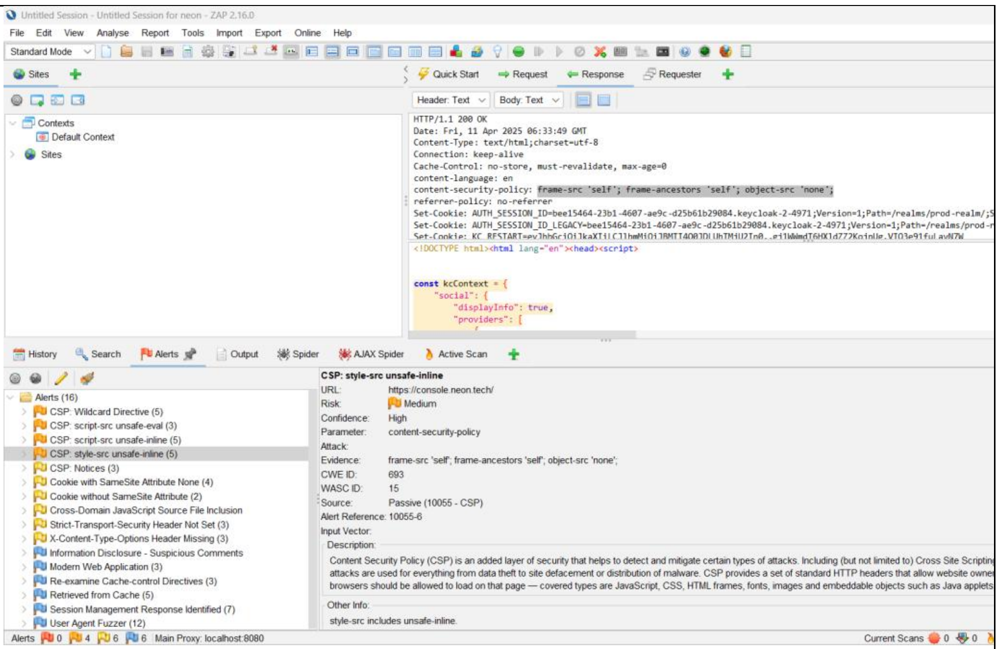
With getting these tool's support, I found below details about vulnerability.

2) Introduction

2.1 Domain	https://console.neon.tech/
2.2 Severity	Medium

3) Vulnerability

3.1 Vulnerabiliy title	CSP: style-src unsafe-inline CVE ID:693 OWASP_2021_A05
3.2 Vulnerability description	Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) Cross Site Scripting (XSS), and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. 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 embeddable objects such as Java applets, ActiveX, audio and video files.
3.3 Affected components	• User Interfaces: Dynamic UIs that rely on inline CSS or user-inputted style information. • Browsers: Modern browsers that enforce CSP directives but allow inline styles when unsafe inline is used. • JavaScript Functionality: Any scripts that dynamically inject inline styles into HTML elements (e.g., via element.style or document.createElement('style')).
3.4 Impact assessment	The web application defines a partial CSP as follows –



```
HTTP/1.1 200 OK
Date: Fri, 11 Apr 2025 06:33:49 GMT
Content-Type: text/html;charset=utf-8
Connection: keep-alive
Cache-Control: no-store, must-revalidate, max-age=0
content-language: en
content-security-policy: frame-src 'self'; frame-ancestors 'self'; object-src 'none';
referrer-policy: no-referrer
Set-Cookie: AUTH_SESSION_ID=bee15464-23b1-4607-ae9c-d25b61b29084.keycloak-2-4971;Version=1;Path=/realms/prod-realm/;S
Set-Cookie: AUTH_SESSION_ID_LEGACY=bee15464-23b1-4607-ae9c-d25b61b29084.keycloak-2-4971;Version=1;Path=/realms/prod-r
Set-Cookie: KC_RESTART=eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0..gi1WWmdT6HX1d772KoinUg.VTO3e91fuLavN7W
```

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


const kcContext = {
    "social": {
        "displayInfo": true,
        "providers": [
```

Alerts (16)
- CSP: Wildcard Directive (5)
- CSP: script-src unsafe-eval (3)
- CSP: script-src unsafe-inline (5)
- CSP: style-src unsafe-inline (5)
- CSP: Notices (3)
- Cookie with SameSite Attribute None (4)
- Cookie without SameSite Attribute (2)
- Cross-Domain JavaScript Source File Inclusion
- Strict-Transport-Security Header Not Set (3)
- X-Content-Type-Options Header Missing (3)
- Information Disclosure - Suspicious Comments
- Modern Web Application (3)
- Re-examine Cache-control Directives (3)
- Retrieved from Cache (5)
- Session Management Response Identified (7)
- User Agent Fuzzer (12)

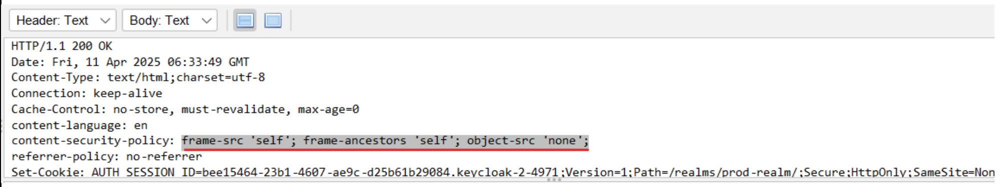
CSP: style-src unsafe-inline

URL:	https://console.neon.tech/
Risk:	Medium
Confidence:	High
Parameter:	content-security-policy
Attack:	
Evidence:	frame-src 'self'; frame-ancestors 'self'; object-src 'none';
CWE ID:	693
WASC ID:	15
Source:	Passive (10055 - CSP)
Alert Reference:	10055-6
Input Vector:	

Description:
Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) Cross Site Scripting attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owne browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets

Other Info:
style-src includes unsafe-inline.

Alerts 0 4 6 6 Main Proxy: localhost:8080 Current Scans 0 0



```
HTTP/1.1 200 OK
Date: Fri, 11 Apr 2025 06:33:49 GMT
Content-Type: text/html;charset=utf-8
Connection: keep-alive
Cache-Control: no-store, must-revalidate, max-age=0
content-language: en
content-security-policy: frame-src 'self'; frame-ancestors 'self'; object-src 'none';
referrer-policy: no-referrer
Set-Cookie: AUTH_SESSION_ID=bee15464-23b1-4607-ae9c-d25b61b29084.keycloak-2-4971;Version=1;Path=/realms/prod-realm/;Secure;HttpOnly;SameSite=Non
```

However, critical directives like script-src, style-src, and default-src are missing. This leads to several security concerns:

7. No restriction on inline scripts or styles unless enforced by browser defaults.

8. In older browsers (e.g., Internet Explorer 11), inline JavaScript and CSS may execute, making the application vulnerable to XSS or CSS Injection if an input-based vulnerability exists.

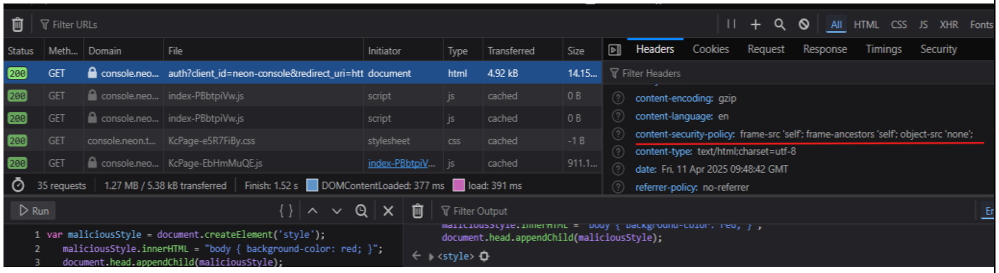
9. The CSP does not provide full mitigation against client-side attacks.

10. This misconfiguration increases the attack surface, especially for applications that handle sensitive data or user inputs.

3.5 Steps to reproduce

1. Open the target website in a browser (e.g., Chrome, Firefox).

2. Press F12 to open Developer Tools.

	3. Go to the Network tab and reload the page.
	
	4. Click on the request for the page (e.g., index.html).
	
	5. In the Headers section, locate the Content-Security-Policy header.
	
	6. Observe the following value:
	
	
	Status · Meth... · Domain · File · Initiator · Type · Transferred · Size
	200 · GET · console.neo... · auth?client_id=neon-console&redirect_uri=htt · document · html · 4.92 kB · 14.15...
	200 · GET · console.neo... · index-PBbtpiVw.js · script · js · cached · 0 B
	200 · GET · console.neo... · index-PBbtpiVw.js · script · js · cached · 0 B
	200 · GET · console.neon.t... · KcPage-e5R7FiBy.css · stylesheet · css · cached · -1 B
	200 · GET · console.neo... · KcPage-EbHmMuQE.js · index-PBbtpiV... · js · cached · 911.1...
	35 requests · 1.27 MB / 5.38 kB transferred · Finish: 1.52 s · DOMContentLoaded: 377 ms · load: 391 ms
	content-encoding: gzip
	content-language: en
	content-security-policy: frame-src 'self'; frame-ancestors 'self'; object-src 'none';
	content-type: text/html;charset=utf-8
	date: Fri, 11 Apr 2025 09:48:42 GMT
	referrer-policy: no-referrer
	`var maliciousStyle = document.createElement('style');`
	`maliciousStyle.innerHTML = "body { background-color: red; }";`
	`document.head.appendChild(maliciousStyle);`
	`<style>`
	
	7. Notice that script-src, style-src, and default-src are not defined.

3.6 Proof of concept	Pen testing: - Inject Inline CSS via Console
	
	1. Open the "Console" tab in Developer Tools.
	2. Paste and run the following JavaScript:
	
	var maliciousStyle = document.createElement('style');
	maliciousStyle.innerHTML = "body { background-color: red; }";
	document.head.appendChild(maliciousStyle);
	
	Modern Browsers Result:
	No visible change — likely blocked by browser defaults due to missing style-src.
	
	Older Browsers Result (e.g., Internet Explorer 11):
	Background color changes to red — proves inline styles are allowed if browser does not apply strict defaults.
	
	** This test does not affect the server or other users.*
	**However, it proves that the application does not explicitly block inline content.*

3.7 Proposed mitigation or fix	Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header. To improve the security of the web application, define a stricter CSP:

	Content-Security-Policy: default-src 'self'; script-src 'self'; style-src 'self'; object-src 'none'; frame-ancestors 'self'; frame-src 'self'; Avoid using 'unsafe-inline' and consider using CSP nonces or hashes for inline scripts and styles.