

## **Vulnerability Discovery**

### Table of Contents

1. Executive Summary .....	3
2. Methodology .....	4
2.1. Tools utilized .....	5
2.2. Findings .....	6
▪ Vulnerability 1 Title - No Content Security Policy (CSP) Header .....	6
▪ Description .....	6
▪ Vulnerability Discovery .....	7
▪ Severity Rating .....	7
▪ Proof of Concept (PoC) .....	8
▪ Exploitation .....	9
▪ Impact .....	10
▪ Remediation .....	10
▪ Vulnerability 2 Title – PII Disclosure .....	11
▪ Description .....	12
▪ Vulnerability Discovery .....	12
▪ Severity Rating .....	12
▪ Proof of Concept (PoC) .....	13
▪ Exploitation .....	14
▪ Impact .....	15
▪ Remediation .....	15
▪ Vulnerability 3 Title - Vulnerable JS Library .....	16
▪ Description .....	17
▪ Vulnerability Discovery .....	17
▪ Severity Rating .....	17
▪ Proof of Concept (PoC) .....	18
▪ Exploitation .....	20
▪ Impact .....	21
▪ Remediation .....	21
3. Conclusions and Reflections .....	22
3.1. What you learned from the process .....	22
3.2. Challenges faced .....	22

### **1. Executive Summary**

This report presents the vulnerability assessment of three websites, Tide (financial services application) SoundCloud (global music-sharing website) and Bullish (crypto-trading website). The main goal of the analysis was to verify whether there would be any loopholes in these websites which might put users or their data, as well as any other projects by the company itself at risk.

The analysis revealed three major vulnerabilities. The first was that Tide had no Content Security Policy (CSP) header. A CSP is a feature that allows to block loading of any malware website. If it does not exist, hackers can manipulate the user interface or run unauthorized scripts which will then result in data breaches or get stolen.

The second vulnerability which was found was Personally Identifiable Information (PII) on SoundCloud. PII disclosures happen when personal information, such as banking information or names, is inadvertently available on the site. The disclosures have the potential to allow malicious users to utilize personal information to stage targeted attacks or violate privacy.

The third threat found in Bullish was the incorporation of an insecure JavaScript library in its installation. Old or insecure libraries can grant hackers access to a site, interfere with its proper functioning, or even steal users' information. It is used for cryptocurrency trading, such an attack can have serious monetary implications.

The methodology adopted considers potential implications, well established proof of concept which contains evidence, and utilized industry-accepted tools. All vulnerabilities were rated for severity, and remediation has been recommended to minimize the risk.

The findings indicate the necessity of continuous upgrade, proper privacy policies, and up-to-date security. Though the vulnerabilities are of numerous different types, they all lead us to the same point: high-profile and reputed sites are at risk if there is no continuous scanning and updating of the security. Such issues being solved in a timely manner will enhance the users' confidence, safeguard individual data, and ensure that the long-term success of these websites is sustained.

### **2. Methodology**

vulnerability assessment was done with manual tools to ensure that the findings were valid and evidence based. Analysis went ahead with target websites Tide, SoundCloud, and Bullish in relation to how critical they were to finance services, music sharing, and cryptocurrency trading. The websites were chosen because they had many customers, and it was crucial to ensure users trust and safety in their services.

Tools like Burp Suite and OWASP ZAP were used both for vulnerability scanning, and browser developer tools were used to scan headers, JavaScript libraries. Manual validation was performed to check results and prevent miss-information, and first impressions were obtained using OWASP ZAP.

For each vulnerability, the discovery process was as below in a formal manner: initial reconnaissance, detection of vulnerabilities, verification, and documentation. After detecting a likely issue, it was examined in depth to determine its root cause, potential steps for exploiting it, and threat to the system. Proof of Concept (PoC) was developed to prove the vulnerabilities with evidence without causing any harm to the targeted systems.

All findings were documented in a standardized style, including description, discovery technique, severity rating, exploitation methods, possible impact, and remediation. This systematic process ensured that all findings were uniform and provided actionable information.

### 2.1. Tools utilized

#### ➤ **Browser Developer Tools**

Built-in function in web browsers, these tools let you inspect what a website sends and shows in real time (page source code, network traffic, and response headers). Used to quickly check missing headers (like CSP), view loaded scripts, and confirm what data is sent or received, helpful for fast and visual validation.

#### ➤ **OWASP ZAP**

An automated scanner and a manual scanner designed for web application security testing. It is to crawl the sites and find issues automatically, providing initial findings that were then reviewed manually to remove errors.

#### ➤ **Burp Suite**

A web security toolkit used to intercept, modify, and replay web traffic between the browser and server. Used manual testing, validating vulnerabilities and capturing evidence without harming the services.

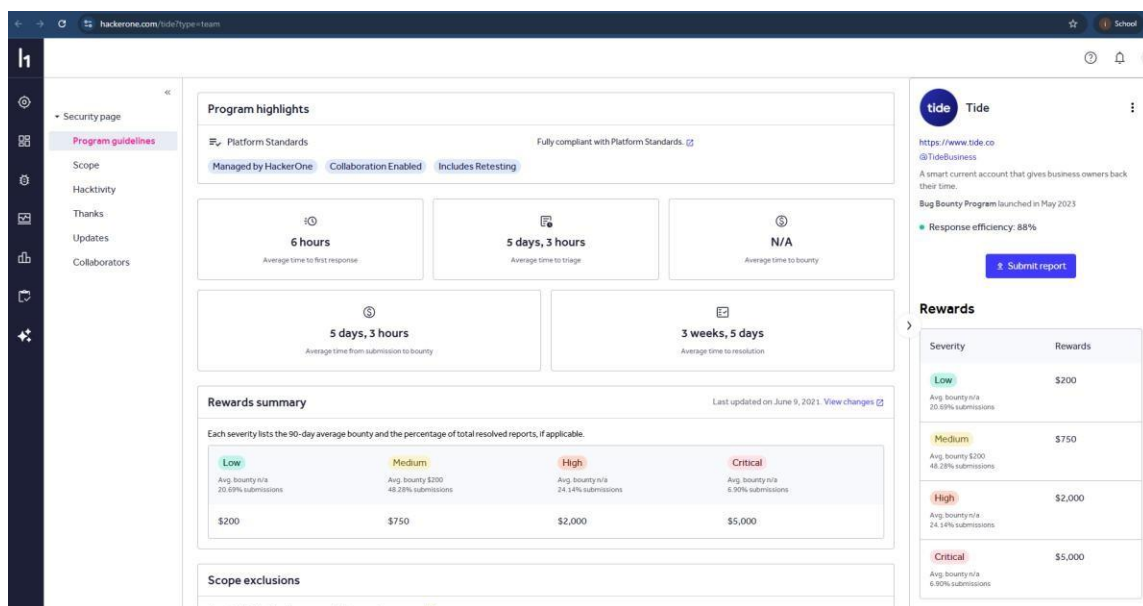
#### ➤ **Kali Basic Commands (e.g., curl)**

Simple command line tools available in Kali Linux, curl is used to perform HTTP requests and quickly view raw responses. Useful for reproducing and documenting vulnerabilities. For example, showing a missing header or exposed data.

### 2.2. Findings

#### ■ Vulnerability 1 Title - No Content Security Policy (CSP) Header

Content Security Policy Header not set is a vulnerability found in <https://www.tide.co/> as a bug bounty program hosted in HackerOne site. It is a finance web application, so security plays a major role.



#### ■ Description

Content Security Policy (CSP) is a security feature used in websites to specify the permitted content sources in a web page. No Content Security Policy is a vulnerability found in the site (<https://www.tide.co/>)

The browser receives no proper guidance what is safe to run on it without a CSP header. This increases the risk of execution of malicious scripts or malicious third-party content

This vulnerability relates to OWASP Top 10 – A05:2021 (Security Misconfiguration), since failing to set security headers is a classic misconfiguration. A07:2021 (Identification and Authentication Failures) is less accurate due to the lack of CSP paired with weak session management or user-controlled input that allows credential/session compromise. For example if a malicious script is injected through query parameters to form fields, no policy is there to block the execution of payloads.

### ▪ Vulnerability Discovery

- By Inspecting the Site

Step 1 - Open the browser and navigate to the <https://www.tide.co/> site.

Step 2 – Press F12 to **Inspect** the page and it pops the network tab and refresh by pressing CTRL +R

Step 3 – under **Response Headers**, check for Content-Security-Policy header or Content-Security-Policy-Report-Only header.

- By OWASP ZAP

Step 1 – Start OWASP ZAP and enter the URL - <https://www.tide.co/>

Step 2 – after completing the scan navigate to the **Alerts**, by analyzing it Content-Security-Policy header is missing.

- By Basic Kali Commands

Step 1 – In a terminal type the **curl -I [https://www.tide.co](https://www.tide.co/)** (it sends requests to the URL and shows only headers.

### ▪ Severity Rating

Risk level – Medium

Domain - <https://www.tide.co/>

OWASP Top 10 – A05:2021 (Security Misconfiguration)

# Vulnerability Discovery

## ▪ Proof of Concept (PoC)

### • By Inspecting the Site

Name	X	Headers	Preview	Response	Initiator	Timing	Cookies
www.tide.co		Remote Address					[26064/00:b812:1e1]j:443
IQdR27epKQ5E3DwyrRT-RJR-L...		Referrer Policy					strict-origin-when-cross-origin
index.html?templateId=5419b...		▼ Response Headers					
index.html?templateId=5419b...		Cache-Control					public, max-age=1800
IQdR27epKQ5E3DwyrRT-RJR-L...		CF-Cache-Status					DYNAMIC
sw_iframe.html?origin=https%...		CF-Ray					980ee64a0cb7a8f8-SIN
IQdR27epKQ5E3DwyrRT-RJR-L...		Date					Thu, 18 Sep 2025 06:48:47 GMT
		Etag					"af9183a2299ad1d57b346218df266793"
		Last-Modified					Wed, 17 Sep 2025 12:34:50 GMT
		Referrer-Policy					strict-origin-when-cross-origin
		Server					cloudflare
		Strict-Transport-Security					max-age=31536000; includeSubDomains; preload
		Vary					Origin
		Via					1.1 2ba05c1608b7148404c7fdd295985ea.cloudfront.net (CloudFront)
		X-Amz-CF-Id					7REaq8OeEdLSBFT_BNRjkbXzCYNbkW07JEXObXgXOkPwArSS3w==
		X-Amz-CF-Pop					LHR50-P2
		X-Amz-Server-Side-Encryption					AES256
		X-Amz-Version-Id					J9fGerjT1K5ZKOf6dSaKNf_DbReJRMm
		X-Cache					Miss from cloudfront
		X-Content-Type-Options					nosniff
		X-Frame-Options					ALLOW-FROM https://uniclient-demo.web.app
		X-Headers-Script-Version					0.0.5
		X-Xss-Protection					1; mode=block

Below is a basic example of Content Security policy

Content-Security-Policy: default-src 'self'; script-src 'self' https://asset.google.com; style-src 'self' 'unsafe-inline'; img-src 'self' data;;

According to the above image, CSP header is missing.

### • By OWASP ZAP

The screenshot shows the OWASP ZAP interface. On the left, the 'Alerts' pane lists 19 alerts, with 'Content Security Policy (CSP) Header Not Set (709)' selected. The main pane displays the details for this alert, including the URL 'https://www.tide.co/', risk level 'Medium', confidence 'High', and a description of CSP. The alert is classified as a medium risk issue.

Content Security Policy (CSP) Header Not Set
URL: https://www.tide.co/
Risk: Medium
Confidence: High
Parameter:
Attack:
Evidence:
CWE ID: 693
WASC ID: 15
Source: Passive (10038 - Content Security Policy (CSP) Header Not Set)
Alert Reference: 10038-1
Input Vector:
Description:
Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content t such as Java applets, ActiveX, audio and video files.
Other Info:
Solution:

Above image confirms that CSP header is missing in the site <https://www.tide.co/>. The scan classified it as a medium risk issue.



- By Basic Kali Commands

```
(kali@kali)-[~]
$ curl -I https://www.tide.co

HTTP/2 200
date: Wed, 17 Sep 2025 19:24:32 GMT
content-type: text/html
cf-ray: 980afbff9e324ccd-CMB
accept-ranges: bytes
cache-control: public, max-age=1800
last-modified: Wed, 17 Sep 2025 12:34:50 GMT
referrer-policy: strict-origin-when-cross-origin
strict-transport-security: max-age=31536000; includeSubDomains; preload
vary: Origin
via: 1.1 ce8f85a4dd9437febbc40094aa7d575a.cloudfront.net (CloudFront)
x-amz-cf-id: vIyzwySisUE4v4Sodt94Htluf0ycgv4KCHwXpZCAoren14By2Ii7ow==
x-amz-cf-pop: LHR50-P2
x-amz-server-side-encryption: AES256
x-amz-version-id: J9fGerj.T1K5ZkOF6dSaKNf_DbReJRMm
x-cache: Miss from cloudfront
x-content-type-options: nosniff
x-frame-options: ALLOW-FROM https://uniclient-demo.web.app
x-headers-script-version: 0.0.5
x-xss-protection: 1; mode=block
cf-cache-status: DYNAMIC
set-cookie: _cfuvid=_Icdpx4reY7kCS7yKsGFpwYcNAvLvHouYjSyPg5phtQ-1758137072856-0.0.1.1-604800000; path=
/; domain=.tide.co; HttpOnly; Secure; SameSite=None
server: cloudflare
```

By using the code `curl -I https://www.tide.co`, headers in the site and be identified. The above Screenshot again reveals CSP header is missing.

### ▪ Exploitation

Step 1 - An attacker identifies the NO CSP Header Vulnerability in <https://www.tide.co>

Step 2 – Attacker Creates a Malicious iframe to redirect to another site.

Example-

```
<iframe src="http://spy.com" width="500" height="500"></iframe>
```

Step 3 – Embed the iframe in site.

Step 4 – Exploit triggers because no CSP header to block inline scripts.

### ▪ Impact

Absence of a CSP header increases the risk of client-side attacks, without CSP header browser has no restrictions on the permitted sources. Which means it is easy for an attacker to inject malicious codes.

- Data theft – cookie stores the sensitive user data which can be exfiltrated from the browser.
- Clickjacking – without “*frame-ancestors*”, attackers can embed an *iframe* and trick users.
- Compliance Risks – vulnerabilities can lead to data breaches which could affect a user’s trust towards the site.

### ▪ Remediation

- Implement a CSP header – only allow content from trusted sources and restrict malicious scripts. An example of a CSP Header,

*Content-Security-Policy: default-src 'self';*

- Prevent clickjacking to avoid attackers tricking users to perform particular tasks. An example to avoid clickjacking,

*frame-ancestors 'none';*

- Use report-only mode – It identifies blocked content before enforcing the policy.

*Content-Security-Policy-Report-Only: default-src 'self'; report-uri https://tide.co.*

- Analyze CSP reports and do changes if necessary to avoid any errors before enforcing it.
- Acknowledge developers to use best practices write codes with the necessary headers.

# Vulnerability Discovery

## ■ Vulnerability 2 Title – PII Disclosure

PII Disclosure is a vulnerability found in <https://soundcloud.com/> as a bug bounty program hosted in Bugcrowd site. SoundCloud is a platform designed to share music similar to spotify.

The screenshot shows the Bugcrowd page for the SoundCloud bug bounty program. The page is titled "Engagements > SoundCloud" and features the SoundCloud logo. It includes a "Submit report" button and a "Follow" button. The page also displays the "Scope rating" as 4 out of 4, the "Testing period" as "Ongoing" (started at Jan 09, 2018), and the "Status" as "In progress" (09 Jan 2018 20:00:00 UTC). A "Details" tab is selected, showing "Vulnerabilities rewarded: 243", "Validation within: 7 days" (75% of submissions are accepted or rejected within 7 days in last 3 months), and "Average payout: \$555" (last 3 months). The "On this page" section includes links for "Overview" and "Description".

The screenshot shows the SoundCloud homepage. It features a large hero section with the text "Discover. Get Discovered." and a "Get Started" button. The hero section also includes a "Sign in" button, a "Create account" button, and a "For Artists" link. Below the hero section, there is a search bar with the text "Search for artists, bands, tracks, podcasts" and a "or Upload your own" button. The page also displays a "Now available" banner at the top and a "Hear what's trending for free in the SoundCloud community" section with an "Explore trending playlists" button.

### ▪ Description

Personally Identifiable Information (PII) Disclosure is a vulnerability found in the site <https://soundcloud.com/147calboy/calboy> where the sensitive information of an individual is exposed due to insufficient protection mechanisms. These data can include phone numbers, banking details, medical records or login details.

This vulnerability could occur due to improper server configurations, inadequate data protection or saving sensitive data in plain text. There is a high risk that these data could lead to financial frauds, identity theft, reputational damage to user and to organizations.

Credit card number of a Maestro card found in this domain, which shouldn't be exposed or included plain text in the client-side. This vulnerability maps to OWASP Top 10 - A01:2021 (Broken Access Control) and A04:2021 (Insecure Design)

### ▪ Vulnerability Discovery

Step 1 – Navigate to the site <https://soundcloud.com/> and find a publicly available domain in the site which can be accessed without any login credentials.

Step 2 – Start OWASP ZAP and paste the SoundCloud URL and scan for vulnerabilities.

Step 3 - Scanning found 3 PII Disclosure, analyze the response and identify the evidence. Credit card type and bank identification number are found.

Step 4 – Analyze the evidence and verify the credit card pattern.

Step 5 – Inspect the <https://soundcloud.com/147calboy/calboy> URL and validate the evidence.

### ▪ Severity Rating

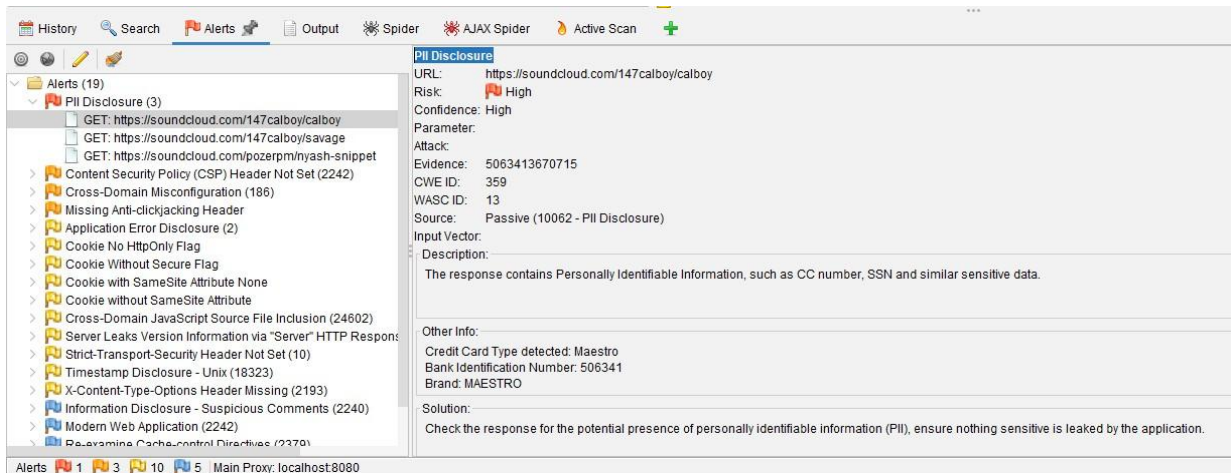
Risk level – High

Domain – <https://soundcloud.com/147calboy/calboy>

OWASP Top 10 - A01:2021 (Broken Access Control) and A04:2021 (Insecure Design)

## Vulnerability Discovery

### ■ Proof of Concept (PoC) OWASP ZAP Scan



**PII Disclosure**

URL: <https://soundcloud.com/147calboy/calboy>

Risk: High

Confidence: High

Parameter:

Attack:

Evidence: **5063413670715**

CWE ID: 359

WASC ID: 13

Source: Passive (10062 - PII Disclosure)

Input Vector:

Description:

The response contains Personally Identifiable Information, such as CC number, SSN and similar sensitive data.

Other Info:

Credit Card Type detected: Maestro  
Bank Identification Number: 506341  
Brand: MAESTRO

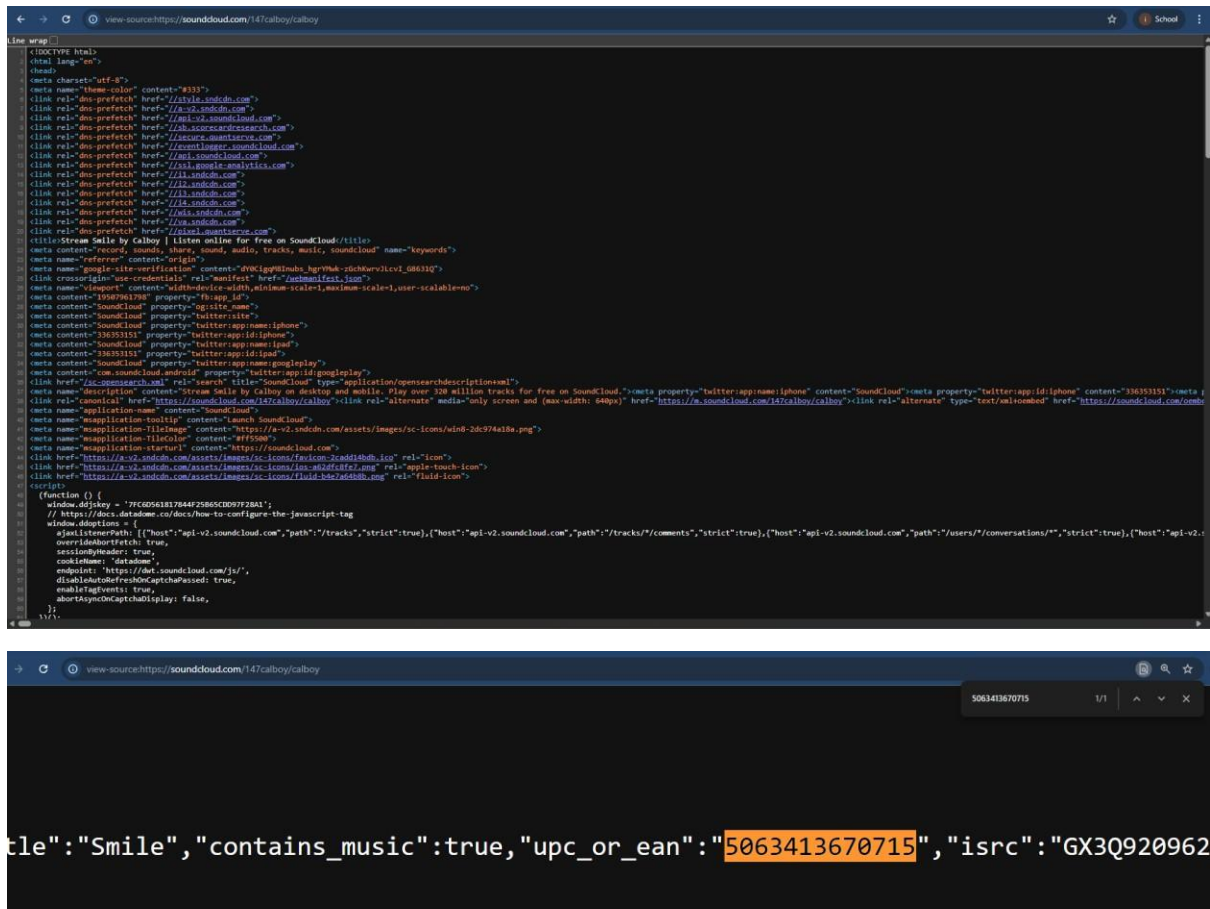
Solution:

Check the response for the potential presence of personally identifiable information (PII), ensure nothing sensitive is leaked by the application.

Above images are the results of vulnerability scans conducted in OWASP ZAP. Scan was conducted in SoundCloud site. Results depict high level of risk in the vulnerability along with the card type and bank identification number. The vulnerability is mainly because of the insufficient server-side sanitization and should be analyzed immediately to prevent data breaches.

Mastercard Maestro is a brand of debit cards and prepaid cards owned by Mastercard.(  
Reference - [https://en.wikipedia.org/wiki/Maestro\\_\(debit\\_card\)](https://en.wikipedia.org/wiki/Maestro_(debit_card)))

# Vulnerability Discovery



Above screenshot is the source of the page which contains the bank identification number. This verifies the data is publicly available without any protocols and in plain text.

## ■ Exploitation

Step 1 – identify the PII Disclosure vulnerability in

<https://soundcloud.com/147calboy/calboy>

Step 2 – check if there are any authorizations in the site, if not find the available personal data in the site (example – banking details, email or passwords)

Step 3 – If the personal data is found send a phishing mail or if the banking details are found attacker can transfer the money.

### ▪ Impact

- Identity theft – An attacker can steal the personal details to open fake accounts and to make unauthorized transactions.
- Phishing attacks – attacker can use the personal data to trick the user.
- Reputational damage – exposure of personal data can cause damage to the user and the organization, which could result in loss of trust towards the organization.
- Recovery cost – After an incident organization might have to compensate the user. And pay for legal defense. Long term funds may be needed to rebuild trust of users.

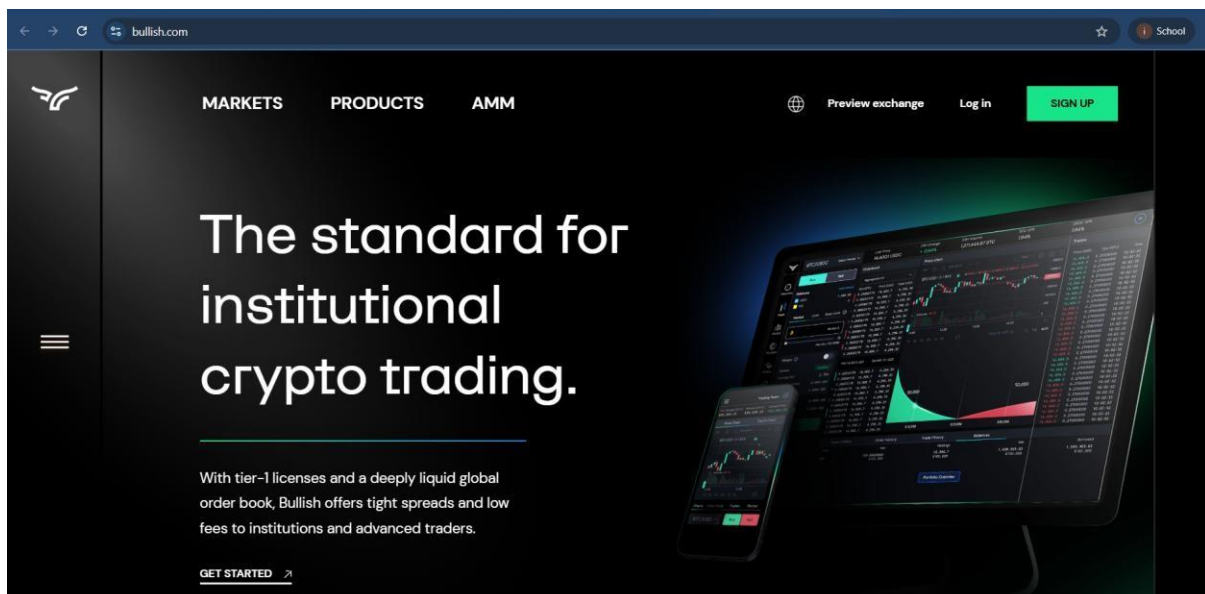
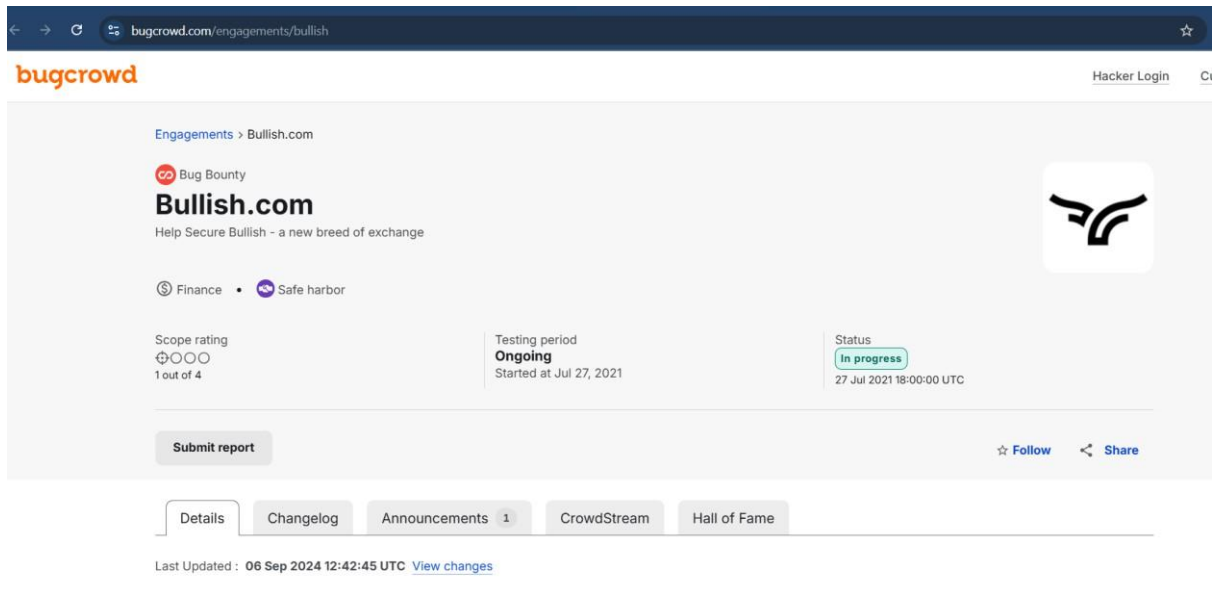
### ▪ Remediation

- Enforcing a strong access control – Enforce principle of least privilege to limit the access of a particular content.
- Remove the leaked content – Analyze and identify the leaked content and remove it from the site.
- Encrypt and store securely – Encrypt the sensitive content and store it properly.
- Monitoring – To avoid any further data breaches enforce data loss prevention tools.
- Compliance with the standards – follow the relevant frameworks like PCI DSS (Payment Card Industry Data Security Standard) to avoid any further breaches or data loss.

## Vulnerability Discovery

### ▪ Vulnerability 3 Title - Vulnerable JS Library

Vulnerable JS Library is a vulnerability found in <https://bullish.com/> as a bug bounty program hosted in Bugcrowd site. This site is focused on crypto trading, which means security is very critical.





### ▪ Description

Vulnerability java script Library is a problem found in bullish site, it is a crypto trading website where trading happens with digital assets and more secure. Bullish site loads Next.js version 12.3.4 which is vulnerable to bypass authorization (CVE-2025-29927 - <https://nvd.nist.gov/vuln/detail/CVE-2025-29927>)

This vulnerability falls under OWASP Top 10 - A06:2021 (Vulnerable and Outdated Components) and A05:2021 (Security Misconfiguration). If an attacker intercept and manipulate the data sent to vulnerable next() functions through reflected inputs trigger prototype and escalate privileges in the JavaScript context. Attackers can exploit client-side attacks like Cross-Site Scripting, data breaches and privilege escalation.

### ▪ Vulnerability Discovery

Step 1 – Open ZAP and scan the <https://bullish.com/> site.

Step 2 – After the scan we can get the vulnerability in [https://bullish.com/\\_next/static/chunks/main-3fd8e90ae090f348.js](https://bullish.com/_next/static/chunks/main-3fd8e90ae090f348.js) domain with the risk level high.

Step 3 – To confirm the vulnerability, start Burp suite and capture the request and send to the repeater.

Step 4 – check the Java Script version by searching “version” on the search bar in the response panel.

So the version “12.3.4” will appear, which is vulnerable for security misconfigurations.

### ▪ Severity Rating

Risk level – High

Domain – [https://bullish.com/\\_next/static/chunks/main-3fd8e90ae090f348.js](https://bullish.com/_next/static/chunks/main-3fd8e90ae090f348.js)

OWASP Top 10 - A06:2021 (Vulnerable and Outdated Components) and A05:2021 (Security Misconfiguration).

# Vulnerability Discovery

## ■ Proof of Concept (PoC)



```
HTTP/1.1 200 OK
Date: Sun, 21 Sep 2025 15:51:37 GMT
Content-Type: application/javascript; charset=UTF-8
Connection: keep-alive
CF-RAY: 982ab99a3af58336-SIN
Age: 49415
Cache-Control: public, max-age=691200
cache-status: "Netlify Edge"; hit
etag: W/"5f526e7a1627c4a48024456442b64959-ssl-df"
netlify-vary: cookie=_next_preview_data:presence|__prerender_bypass:presence
strict-transport-security: max-age=31536000; includeSubDomains; preload
vary: Accept-Encoding
x-content-type-options: nosniff
x-frame-options: DENY
x-middleware-next: 1
version="12.3.4";t.router=h;var T=y.default();t.emitter=T;var N,I,D,B,q,U,H,F,W,z,G=function(e){return[].slice.call(e)},Z=void 0,$=!1;se
peof self&&self.cancelIdleCallback&&self.cancelIdleCallback.bind(window)||function(e){return clearTimeout(e)};t.cancelIdleCallback=n,("f
erty("isPrototypeOf"))},18286:function(e,t){t.use strict;Object.defineProperty(t,"__esModule",{value:!0}),t.default=function(){var e=Obj
nfo({route:ce,pathname:Y,query:J,as:n,resolvedAs:oe,routeProps:$,locale:y.locale,isPreview:y.isPreview,hasMiddleware:ie});case 111:if("ro
ext)/data/","").replace(/\.json$/,"").split("/")",d=f[0];l.pathname="index"!==(f[1]?"/".concat(f.slice(1).join("/")):""/".concat(f[0].join("/")):"/");l.buildId=d;if(c):
ute with the same specificity as a optional catch-all route ("'.concat(o,'" and '"').concat(o,"[...").concat(this.optionalRestSlugName, '
refix:y,suffix:"",pattern:m|i,modifier:f("MODIFIER")|""))}else{var g=h||f("ESCAPED_CHAR");if(g)l+=g;else if(l&&(u.push(l),l=""),f("OPEN'
rEach((function(e){delete q[e.id]}))),H=function(e,t){t=t||{};var r=[200,500];"interactionCount"in performance||u||(u=p("event",T,{type
```

The above screenshot show a high level of risk in the JavaScript file [https://bullish.com/\\_next/static/chunks/main-3fd8e90ae090f348.js](https://bullish.com/_next/static/chunks/main-3fd8e90ae090f348.js) , which uses next.js version “12.3.4” which is vulnerable for improper authorization. The search results show two JS vulnerabilities which use next.js so by upgrading to a new version of next.js

# Vulnerability Discovery

The screenshot shows the Burp Suite interface. At the top, there's a menu bar with options like Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, and Learn. Below the menu bar, there's a tabbed interface with 'Intercept', 'HTTP history', 'WebSockets history', 'Match and replace', and 'Proxy settings'. The 'HTTP history' tab is active, showing a list of HTTP requests. The table has columns for #, Host, Method, URL, Params, Edited, Status code, Length, MIME type, Extension, Title, Notes, TLS, IP, Cookies, Time, and List. The first request is a GET to https://bullish.com/\_next/static/chunks/main-3fd8e90ae090f348.js. The 'Request' pane on the left shows the raw HTTP request details, including headers like Host, Cookie, User-Agent, and Accept. The 'Response' pane on the right shows the raw HTTP response details, including status code 200 OK, Date, Content-Type, and various headers. The 'Inspector' pane on the far right shows the request and response attributes, cookies, headers, and body.

This screenshot provides a detailed view of the 'Request' and 'Response' panes. The 'Request' pane on the left shows the raw HTTP request for the file \_next/static/chunks/main-3fd8e90ae090f348.js. The 'Response' pane on the right shows the raw HTTP response. The response body is a JavaScript file, and a red box highlights the 'version' field in the code, which is set to '12.3.4'. The 'Inspector' pane on the far right shows the request and response attributes, cookies, headers, and body.

Above screenshot of Burp suite again confirms that [https://bullish.com/\\_next/static/chunks/main-3fd8e90ae090f348.js](https://bullish.com/_next/static/chunks/main-3fd8e90ae090f348.js) uses next.js version “12.3.4”

### [Improper Authorization](#)

`next` is a react framework.

Affected versions of this package are vulnerable to Improper Authorization due to the improper handling of the `x-middleware-subrequest` header. An attacker can bypass authorization checks by sending crafted requests containing this specific header.

How to fix Improper Authorization?

Upgrade next to version 12.3.5, 13.5.9, 14.2.25, 15.2.3, 15.3.0-canary.12 or higher.

`>=11.1.4 <12.3.5`

`>=13.0.0 <13.5.9`

`>=14.0.0 <14.2.25`

`>=15.0.0-rc.0 <15.2.3`

`>=15.3.0-canary.0 <15.3.0-canary.12`

Link - <https://security.snyk.io/package/npm/next/12.3.4>

According to the above site JavaScript version 12.3.4 is vulnerable due to improper authorization and an attacker can bypass the authorization.

#### ▪ **Exploitation**

Step 1 – Identify the outdated or vulnerable libraries using tools like browser dev or Snyk.

Step 2 – Compare the library version with CVE listings to find vulnerabilities.

Step 3 – Create a payload.

Step 4 – Inject a malicious script to exploit the vulnerability.

Step 5 – Create a fake form to execute the malicious script.

### ▪ Impact

- Sensitive data is exposed through cookies or forms and can be leaked.
- Session takeover - Attackers can perform unauthorized actions on behalf of users.
- Reputational damage – exposure of personal data can cause damage to the user and the organization, which could result in loss of trust towards the organization.
- Increase in number of surface attacks since multiple modules may inherit the same vulnerability due to its dependency.
- Possibility of phishing attacks – An attacker can modify the UI to inject fake logins and trick users to reveal login credentials.

### ▪ Remediation

- Conduct regular scans to detect vulnerabilities using manual or automated tools.
- Update the next.js library to a newly released version to avoid any outdated bugs.
- Limit third party access to scripts through principle of least privilege.
- Adapt to a well-maintained library by removing obsolete packages or unused packages.
- Maintain a dependency list with their versions and particular libraries track the package integrity.

### 3. **Conclusions and Reflections**

This report emphasizes critical security vulnerabilities in three platforms Tide, SoundCloud, and Bullish. The report showed that even established businesses cannot escape it, if security best practices such as security headers, data protectors for privacy, and library patching are not adhered to. All the vulnerabilities have the potential to cause harm to users, from data leakage to financial loss. The analysis confirmed the value of supplementing automated code with human checks to produce results that are accurate. Moreover, the report highlights the importances of continuous security monitoring, and proactive interventions to maintain trust, protect sensitive information, and achieve long-term resiliency against changing threats.

#### 3.1. **What you learned from the process**

- To use manual tools and find vulnerabilities.
- Impact of vulnerability if it is not addressed.
- Proper documentation is also important to communicate with non-technical people.
- Importance of proper monitoring is a continuous responsibility of an organization.
- How different areas like music, crypto and finance have overlapping security risks.

#### 3.2. **Challenges faced**

- Using manual tools is time consuming and difficult to discover vulnerabilities.
- Time limitations made it difficult to find vulnerabilities in depth.
- Limitation in scope of certain web applications made it difficult to find vulnerabilities.
- Inaccurate results of some tools made it difficult to figure out what is correct and wrong.
- Lack of knowledge about real-world exploitation made it challenging to understand and write how an attack is exploited.