Sri Lanka Institute of Information Technology

PII Disclosure

IE2062 - Web Security

IT23269484 - T. H. Ranasinghe

PII Disclosure on hemi.xyz

1. Vulnerability Title

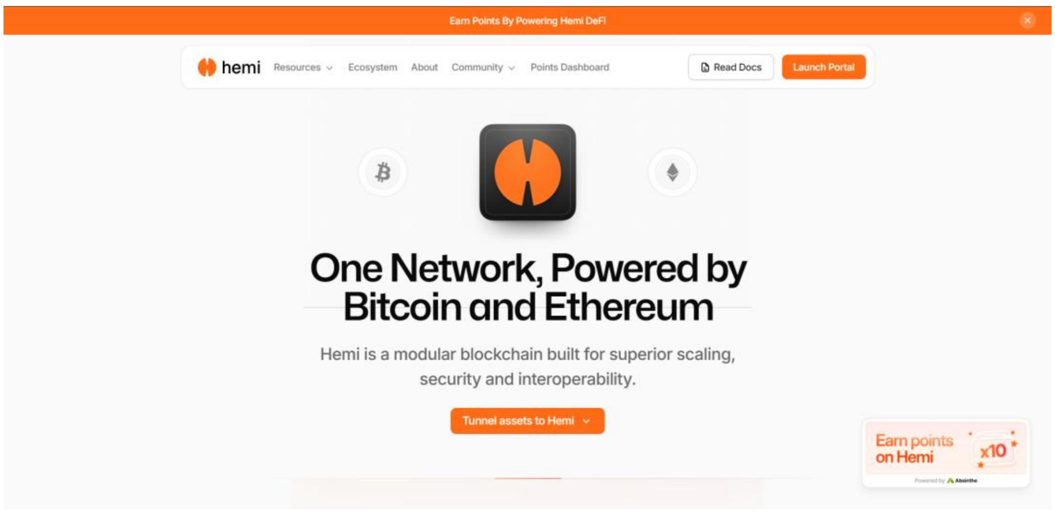
PII Disclosure on **https://hemi.xyz**

2. Vulnerability Description

While testing the website https://hemi.xyz , which I found through the HackerOne bug bounty platform, I discovered a critical security issue. The website's blog pages were openly sharing sensitive personal information, such as people's full names and email addresses. This problem is known as Personally Identifiable Information (PII) disclosure.

PII is any data that can identify a person, like their name or email. When this information is exposed, it's a big deal because hackers can use it for bad things. For example, they could pretend to be someone else (identity theft), send fake emails to trick people (phishing), or flood inboxes with spam. Exposed PII could also lead to legal problems for the website, like fines from laws such as GDPR (a European rule that protects user privacy). Worst of all, users might stop trusting the website if they learn their personal details are not safe.

I found this issue while checking the website's blog section, specifically on pages like https://hemi.xyz/blog/category/learn-center/page/4/. The personal information was visible in the HTML code of the page, which means anyone visiting the page could see it without needing special access. This is a high-risk problem because it's easy for attackers to find and misuse this data.


Earn Points By Powering Hemi DeFi
hemi
Resources
Ecosystem
About
Community
Points Dashboard
Read Docs
Launch Portal
One Network, Powered by Bitcoin and Ethereum
Hemi is a modular blockchain built for superior scaling, security and interoperability.
Tunnel assets to Hemi
Earn points on Hemi
x10

This report explains exactly what I found, how attackers could use this vulnerability, why it's dangerous, and what the website should do to fix it. I've included screenshots from my tests to show proof of the issue and make it clear how I found it.

3. Affected Components

Website URL:

- https://hemi.xyz (The main website where the issue was found)

Specific Pages with Issues:

- https://hemi.xyz/ (The homepage, which may also contain some PII)
- https://hemi.xyz/blog/category/learn-center/page/4/ (A blog page with clear PII exposure in its HTML response)

Server Details:

- The website is hosted behind a Cloudflare Proxy (IP address: 104.18.22.222), which helps protect it but didn't stop this issue.
- It uses WP Engine, a WordPress hosting service, detected through server headers. This means the site runs on WordPress, a common website platform.

4. Impact Assessment

This PII disclosure vulnerability is a big problem because it can cause serious harm. Here's what could happen:

- **Identity Theft:** Hackers can use exposed names and emails to pretend to be someone else. For example, they could open fake bank accounts or apply for loans in someone's name.
- **Phishing Attacks:** Attackers can send fake emails that look real, tricking people into sharing passwords or credit card details.

- **Spam:** Exposed emails can be added to spam lists, flooding users with unwanted messages.
- **Legal Trouble:** Laws like GDPR require websites to protect user data. If PII is leaked, the website could face hefty fines or lawsuits.
- **Loss of Trust:** If users find out their personal information is exposed, they might stop using the website, hurting its reputation.

These risks don't just affect users they can also damage the website's business and credibility. For example, a similar PII leak on a social media platform in 2021 exposed millions of emails, leading to widespread phishing attacks and a damaged reputation.

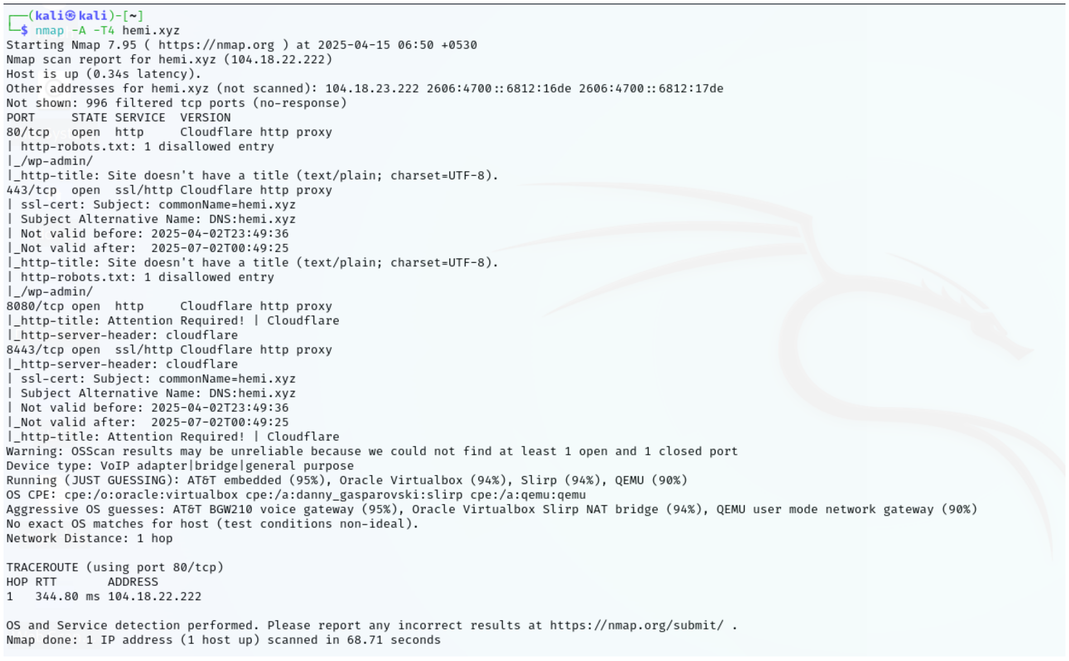
5. Steps to Reproduce

To find this vulnerability, I used several security tools to test the website thoroughly. Below are the steps I followed, along with screenshots showing what I found. These steps can be repeated to see the same issue.

- **Nmap Scan**

I started by scanning the website's server to understand its setup. I used Nmap, a tool that checks for open ports and server details. The command I ran was:

nmap -A -T4 hemi.xyz



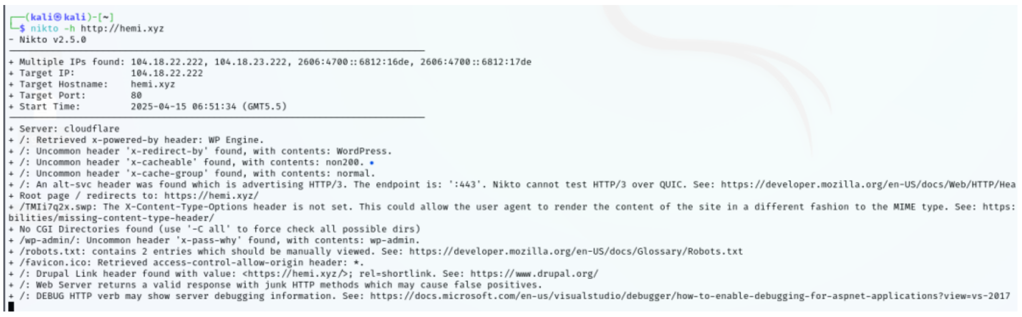
```
┌──(kali㉿kali)-[~]
└─$ nmap -A -T4 hemi.xyz
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-15 06:50 +0530
Nmap scan report for hemi.xyz (104.18.22.222)
Host is up (0.34s latency).
Other addresses for hemi.xyz (not scanned): 104.18.23.222 2606:4700::6812:16de 2606:4700::6812:17de
Not shown: 996 filtered tcp ports (no-response)
PORT     STATE SERVICE  VERSION
80/tcp   open  http     Cloudflare http proxy
| http-robots.txt: 1 disallowed entry
|_/wp-admin/
|_http-title: Site doesn't have a title (text/plain; charset=UTF-8).
443/tcp  open  ssl/http Cloudflare http proxy
| ssl-cert: Subject: commonName=hemi.xyz
| Subject Alternative Name: DNS:hemi.xyz
| Not valid before: 2025-04-02T23:49:36
|_Not valid after:  2025-07-02T00:49:25
|_http-title: Site doesn't have a title (text/plain; charset=UTF-8).
| http-robots.txt: 1 disallowed entry
|_/wp-admin/
8080/tcp open  http     Cloudflare http proxy
|_http-title: Attention Required! | Cloudflare
|_http-server-header: cloudflare
8443/tcp open  ssl/http Cloudflare http proxy
|_http-server-header: cloudflare
| ssl-cert: Subject: commonName=hemi.xyz
| Subject Alternative Name: DNS:hemi.xyz
| Not valid before: 2025-04-02T23:49:36
|_Not valid after:  2025-07-02T00:49:25
|_http-title: Attention Required! | Cloudflare
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: VoIP adapter|bridge|general purpose
Running (JUST GUESSING): AT&T embedded (95%), Oracle Virtualbox (94%), Slirp (94%), QEMU (90%)
OS CPE: cpe:/o:oracle:virtualbox cpe:/a:danny_gasparovski:slirp cpe:/a:qemu:qemu
Aggressive OS guesses: AT&T BGW210 voice gateway (95%), Oracle Virtualbox Slirp NAT bridge (94%), QEMU user mode network gateway (90%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

TRACEROUTE (using port 80/tcp)
HOP RTT       ADDRESS
1   344.80 ms 104.18.22.222

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 68.71 seconds
```

- The website had open ports: 80 (HTTP), 443 (HTTPS), 8080, and 8443. These are common for web servers, but 8080 and 8443 suggested additional services.
- It runs WordPress behind Cloudflare, a security service that hides the real server.
- A robots.txt file listed wp-admin, which is typical for WordPress sites but can give hackers clues about the site's structure.

- **Nikto Scan**

Next, I used Nikto, a tool that checks for common web server issues. I ran:

nikto -h http://hemi.xyz

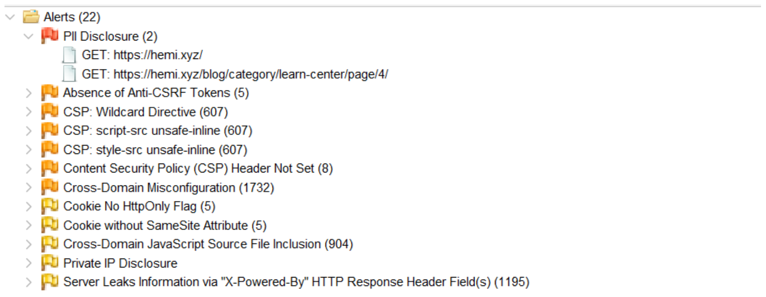


```
┌──(kali㉿kali)-[~]
└─$ nikto -h http://hemi.xyz
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Multiple IPs found: 104.18.22.222, 104.18.23.222, 2606:4700::6812:16de, 2606:4700::6812:17de
+ Target IP:          104.18.22.222
+ Target Hostname:    hemi.xyz
+ Target Port:        80
+ Start Time:         2025-04-15 06:51:34 (GMT5.5)
---------------------------------------------------------------------------
+ Server: cloudflare
+ /: Retrieved x-powered-by header: WP Engine.
+ /: Uncommon header 'x-redirect-by' found, with contents: WordPress.
+ /: Uncommon header 'x-cacheable' found, with contents: non200.
+ /: Uncommon header 'x-cache-group' found, with contents: normal.
+ /: An alt-svc header was found which is advertising HTTP/3. The endpoint is: ':443'. Nikto cannot test HTTP/3 over QUIC. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Hea
+ Root page / redirects to: https://hemi.xyz/
+ /TMIi7q2x.swp: 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:
bilities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /wp-admin/: Uncommon header 'x-pass-why' found, with contents: wp-admin.
+ /robots.txt: contains 2 entries which should be manually viewed. See: https://developer.mozilla.org/en-US/docs/Glossary/Robots.txt
+ /favicon.ico: Retrieved access-control-allow-origin header: *.
+ /: Drupal Link header found with value: <https://hemi.xyz/>; rel=shortlink. See: https://www.drupal.org/
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.
+ /: DEBUG HTTP verb may show server debugging information. See: https://docs.microsoft.com/en-us/visualstudio/debugger/how-to-enable-debugging-for-aspnet-applications?view=vs-2017
```

- The server headers confirmed it uses WP Engine and WordPress.
- There were signs that sensitive files might be accessible, which could help attackers find more weaknesses.
- No specific PII was flagged, but the server setup showed it wasn't fully locked down.

- **OWASP ZAP Scan**

I used OWASP ZAP, a security tool that crawls websites and looks for problems. I set it up to explore the homepage and blog pages, focusing on https://hemi.xyz/blog/category/learn-center/page/4/ .



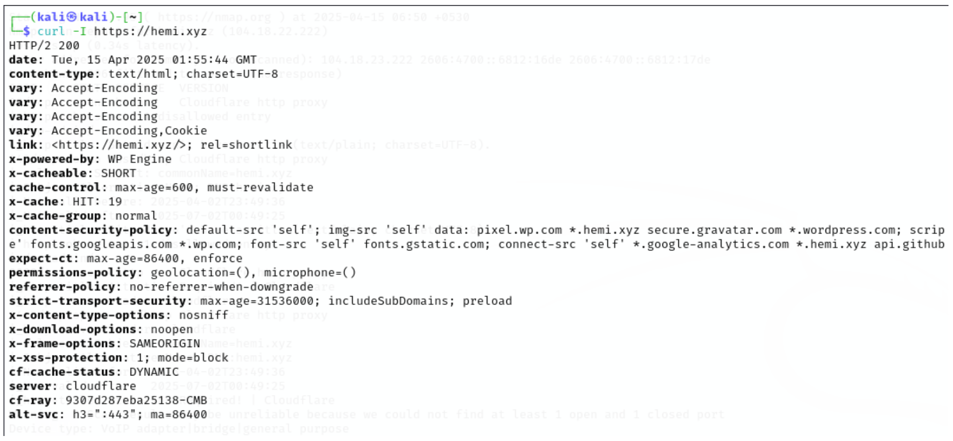
```
Alerts (22)
  PII Disclosure (2)
    GET: https://hemi.xyz/
    GET: https://hemi.xyz/blog/category/learn-center/page/4/
  Absence of Anti-CSRF Tokens (5)
  CSP: Wildcard Directive (607)
  CSP: script-src unsafe-inline (607)
  CSP: style-src unsafe-inline (607)
  Content Security Policy (CSP) Header Not Set (8)
  Cross-Domain Misconfiguration (1732)
  Cookie No HttpOnly Flag (5)
  Cookie without SameSite Attribute (5)
  Cross-Domain JavaScript Source File Inclusion (904)
  Private IP Disclosure
  Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) (1195)
```

- The blog page was leaking PII, like names and email addresses, in the HTML response.
- OWASP ZAP marked this as a high-risk issue because anyone could see the data by visiting the page.
- The tool also noted other minor issues, but the PII leak was the biggest concern.

- **Curl Header Check**

I checked the website's HTTP headers to see how it was configured. I used this command:

curl -I https://hemi.xyz

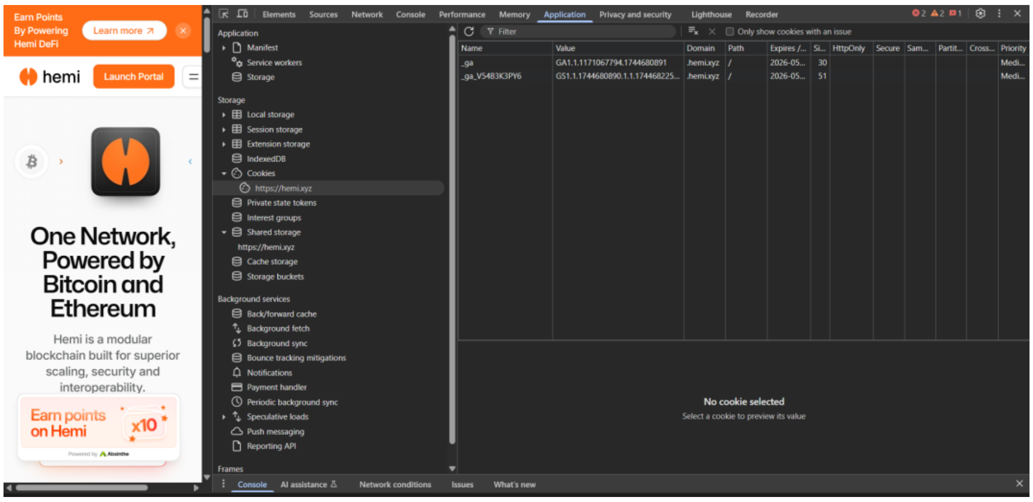


```
┌──(kali㉿kali)-[~]
└─$ curl -I https://hemi.xyz
HTTP/2 200
date: Tue, 15 Apr 2025 01:55:44 GMT
content-type: text/html; charset=UTF-8
vary: Accept-Encoding
vary: Accept-Encoding
vary: Accept-Encoding
vary: Accept-Encoding,Cookie
link: <https://hemi.xyz/>; rel=shortlink
x-powered-by: WP Engine
x-cacheable: SHORT
cache-control: max-age=600, must-revalidate
x-cache: HIT: 19
x-cache-group: normal
content-security-policy: default-src 'self'; img-src 'self' data: pixel.wp.com *.hemi.xyz secure.gravatar.com *.wordpress.com; scrip
e' fonts.googleapis.com *.wp.com; font-src 'self' fonts.gstatic.com; connect-src 'self' *.google-analytics.com *.hemi.xyz api.github
expect-ct: max-age=86400, enforce
permissions-policy: geolocation=(), microphone=()
referrer-policy: no-referrer-when-downgrade
strict-transport-security: max-age=31536000; includeSubDomains; preload
x-content-type-options: nosniff
x-download-options: noopen
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
cf-cache-status: DYNAMIC
server: cloudflare
cf-ray: 9307d287eba25138-CMB
alt-svc: h3=":443"; ma=86400
```

- The server sent an "X-Powered-By" header, confirming it uses WP Engine.
- There weren't enough security headers to protect against data leaks, which made the PII issue worse.

- **Browser Cookie Check**

I visited the website in a browser and used Developer Tools (F12 > Application > Cookies) to check how cookies were set. Cookies are small files websites use to track users, and they need strong security settings.



Name	Value	Domain	Path	Expires /...	Si...	HttpOnly	Secure	Sam...	Partit...	Cross...	Priority
_ga	GA1.1.1171067794.1744680891	.hemi.xyz	/	2026-05...	30						Medi...
_ga_V54B3K3PY6	GS1.1.1744680890.1.1.174468225...	.hemi.xyz	/	2026-05...	51						Medi...

- The cookies were missing important settings, like HttpOnly and SameSite, which protect against certain attacks.
- Weak cookies could make the PII leak even more dangerous if attackers combine it with other tricks, like stealing session data.

6. Proof of Concept (PoC)

To prove the PII disclosure, I created a simple test that anyone can try:

GET https://hemi.xyz/blog/category/learn-center/page/4/

The server sends back a webpage with names and email addresses in plain text, embedded in the HTML code. You don't need to log in or do anything special just visit the page, and the sensitive data is there for anyone to see. This makes it super easy for attackers to collect this information.

7. Proposed Mitigation or Fix

To fix this PII disclosure and keep users safe, the website needs to act. Here are the steps they should follow, explained simply:

Stop Sharing PII:

- Remove names, emails, and other personal details from blog pages unless they're necessary. For example, don't include user data in public posts.

- Check all pages to make sure no sensitive information is accidentally shared.

Add Access Controls:

- Lock down sensitive data so only authorized users (like logged-in admins) can see it.
- Use passwords or other checks to hide PII from public visitors.

Classify Data:

- Sort information into “public” (safe to share) and “private” (needs protection).
- Keep private data, like emails, off public pages entirely.

Extra Safety Steps:

- Monitor Access: Keep a record of who visits pages with sensitive data. If something looks suspicious, investigate it.
- Run Regular Tests: Use tools like OWASP ZAP to check for PII leaks and other issues at least monthly.
- Automate Scans: Set up tools to automatically scan the website for exposed data, so problems are caught early.
- Train Staff: Teach the website’s developers to avoid adding PII to public pages and to check their work.

These steps will help stop the current leak and prevent future ones.

8. Conclusion

The security test of https://hemi.xyz revealed a high-risk PII disclosure issue. Names and emails were exposed on a public blog page, risking identity theft, phishing, GDPR fines, and loss of user trust. Quick action is needed to remove PII, add access controls, and monitor data. Regular tests will prevent future leaks, ensuring user safety and compliance.