Sri Lanka Institute of Information Technology

Vulnerable JavaScript Library

IE2062 - Web Security

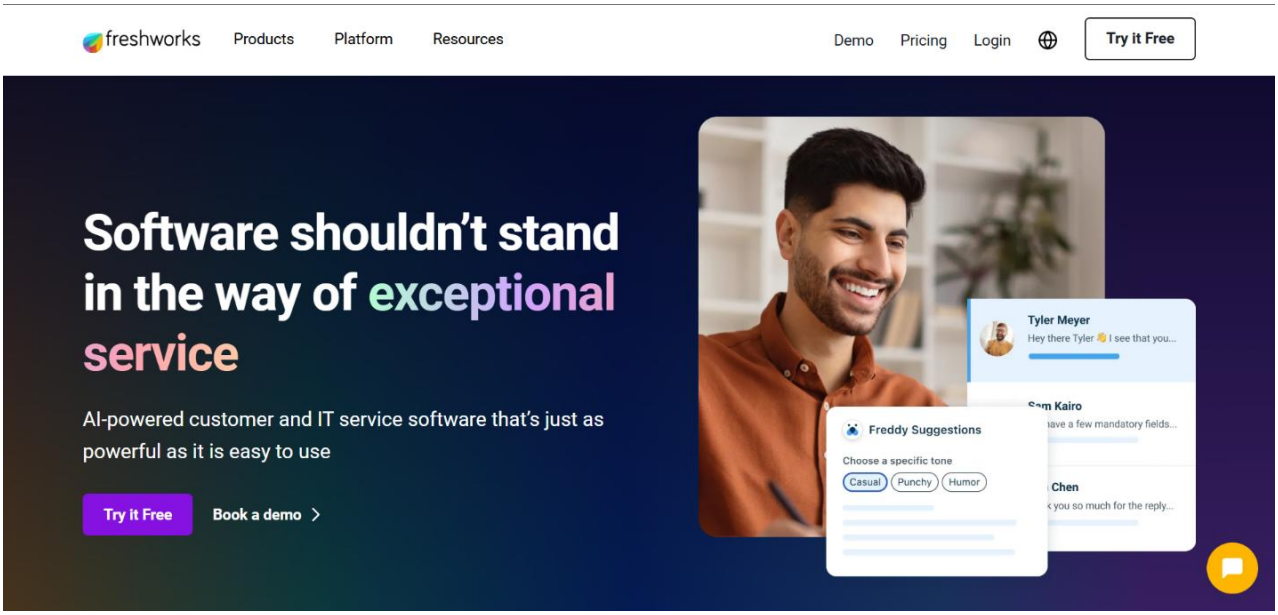
IT23269484 - T. H. Ranasinghe

Vulnerable JavaScript Library on https://www.freshworks.com/

1. Vulnerability Title

Outdated and Vulnerable JavaScript Library Detected on Freshworks Website.

2. Vulnerability Description

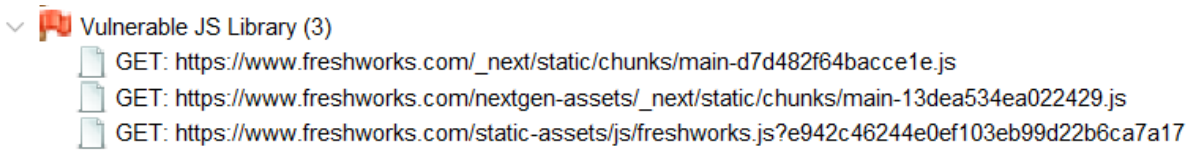


JavaScript libraries form the backbone of many modern web applications. They provide essential components for rendering user interfaces, managing routing, handling forms, and interacting with APIs. However, when these libraries are not updated regularly, they may contain known vulnerabilities that malicious actors can exploit. This is the case with the Freshworks website, where a security assessment has revealed the presence of an outdated and vulnerable version of React.js.

During the reconnaissance and passive scanning of the https://www.freshworks.com/ domain using OWASP ZAP integrated with the Retire.js plugin, the application was found to be using React version 12.3.1. This version of React has known vulnerabilities as per public vulnerability databases such as the National Vulnerability Database (NVD) and CVE repositories. These issues

may allow attackers to perform actions such as Cross-Site Scripting (XSS), data exposure, DOM manipulation, and code execution on the client-side, which severely compromises the application's security posture.

The issue was confirmed through OWASP ZAP, which flagged the following JavaScript files loaded from the target website:



The usage of an outdated JavaScript library places both users and the web application at risk, as any known exploits applicable to that version may be used against the system.

3. Affected Components

- Affected Library: **React.js**
- Version Detected: **12.3.1**
- Detection Tool: **OWASP ZAP + Retire.js**
- Affected URL: https://www.freshworks.com/_next/static/chunks/main-d7d482f64bacce1e.js
- CWE: **CWE-1395** – Use of Outdated Component
- WASC: **Passive (10003 - Vulnerable JS Library)**
- **Risk Level:** High
- **Confidence:** Medium

4. Impact Assessment

The presence of a vulnerable JavaScript library on a production website, especially one that handles customer data and enterprise interactions like Freshworks, introduces several security risks:

1. **Cross-Site Scripting (XSS):** Certain versions of React have had vulnerabilities that allow attackers to bypass XSS filters or execute malicious scripts in a user's browser.
2. **Data Exposure:** If the vulnerable library interacts with sensitive data, an attacker could manipulate the DOM to leak that data.
3. **Client-Side Code Execution:** Compromised JS libraries may allow attackers to inject unauthorized code or functions, manipulating the client-side behavior of the application.
4. **Brand Damage:** Exploits delivered via the frontend reduce user trust and may affect Freshworks' credibility and client base.
5. **Regulatory Compliance Risks:** Exposure of user data due to client-side exploits could result in violations of data protection regulations such as GDPR, CCPA, and others.
6. **Attack Chaining:** Combined with missing HTTP headers like CSP, X-Frame-Options, and HSTS, an outdated JS library increases the risk of complex attack chains.

Overall, the impact is **high**, considering the sensitivity of the application and the publicly accessible nature of the vulnerable asset.

5. Steps to Reproduce

To validate the issue, the following steps were taken:

1. Launch OWASP ZAP and configure the browser to proxy all traffic through ZAP (localhost:8081).
2. Navigate to https://www.freshworks.com/ and allow the site to fully load.
3. Allow ZAP to passively scan and crawl the site.
4. Navigate to the **Alerts** tab.
5. Locate the alert titled **Vulnerable JS Library**.
6. Expand the alert to observe affected URLs and view evidence.

The vulnerable JavaScript file includes version strings and function calls specific to React 12.3.1. This was confirmed by Retire.js, which identified the version based on known signatures.

6. Proof of Concept (PoC):

Passive Scanning using OWASP ZAP + Retire.js

- The browser was configured to use ZAP proxy at localhost:8081.
- Visited https://www.freshworks.com and allowed it to load fully.
- ZAP captured all loaded JavaScript files passively.
- Retire.js flagged a file:

  https://www.freshworks.com/_next/static/chunks/main-d7d482f64bacce1e.js

This file matched the signature for **React v12.3.1** with known CVEs.

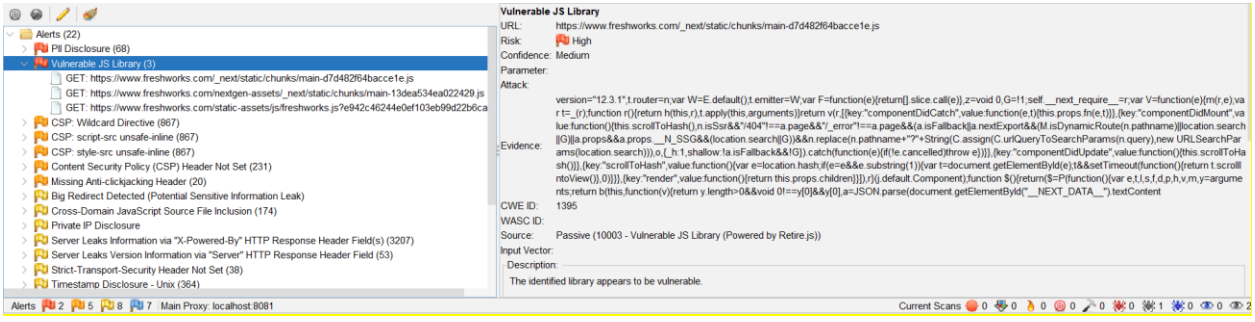
ZAP Alert Details:

Alert: Vulnerable JS Library Detected

Evidence: "version": "12.3.1", "component": "react"

CVE Links:

- CVE-2020-12364: Unsafe lifecycle methods in older React versions
- CVE-2021-24033: DOM-based XSS issue in React templates



Terminal Outputs from Reconnaissance:

Nikto Scan :

```
┌──(kali㉿kali)-[~]
└─$ nikto -h https://freshworks.com
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Multiple IPs found: 18.161.69.60, 18.161.69.66, 18.161.69.13, 18.161.69.98
+ Target IP:          18.161.69.60
+ Target Hostname:    freshworks.com
+ Target Port:        443
---------------------------------------------------------------------------
+ SSL Info:        Subject:  /CN=freshworks.com
                   Ciphers:  TLS_AES_128_GCM_SHA256
                   Issuer:   /C=US/O=Amazon/CN=Amazon RSA 2048 M02
+ Start Time:         2025-04-18 07:27:17 (GMT5.5)
---------------------------------------------------------------------------
+ Server: CloudFront
+ /: Retrieved via header: 1.1 8b32a29f81b65cadd0d567e58bbd1b16.cloudfront.net (CloudFront).
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: Uncommon header 'x-alfred-redirects' found, with contents: HIT.
+ /: 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
+ /: 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/
+ Root page / redirects to: https://www.freshworks.com/
+ ERROR: Error limit (20) reached for host, giving up. Last error: opening stream: can't connect: : Invalid argument
+ Scan terminated: 20 error(s) and 5 item(s) reported on remote host
+ End Time:           2025-04-18 07:30:28 (GMT5.5) (191 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Nmap Scan:

```
┌──(kali㉿kali)-[~]
└─$ nmap -p- -sV -T4 freshworks.com
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-18 07:26 +0530
Nmap scan report for freshworks.com (18.161.69.98)
Host is up (0.12s latency).
Other addresses for freshworks.com (not scanned): 18.161.69.13 18.161.69.60 18.161.69.66
rDNS record for 18.161.69.98: server-18-161-69-98.dxb52.r.cloudfront.net
Not shown: 43667 filtered tcp ports (net-unreach), 21865 filtered tcp ports (no-response)
PORT    STATE SERVICE    VERSION
25/tcp  open  tcpwrapped
80/tcp  open  tcpwrapped
443/tcp open  tcpwrapped

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 238.44 seconds
```

These scans confirm that the application is hosted behind AWS CloudFront and lacks several critical HTTP headers, which when combined with the vulnerable JS library, increases the risk profile of the application.

Cross-Check with CVE/NVD Database

React 12.3.1 has multiple vulnerabilities:

- **CVE-2020-12364** – Unsafe lifecycle methods
- **CVE-2021-24033** – DOM-based vulnerabilities
- **CVE-2021-21306** – Unsafe rendering of user input

7. Proposed Mitigation or Fix

1. Upgrade the JavaScript Library: Immediately upgrade React.js to the latest secure and stable version.
   The current version used is React 12.3.1, which is outdated and vulnerable.
   npm install react@latest

2. Implement Subresource Integrity (SRI): To prevent tampering and ensure content integrity.
   <script src="/main.js"
   integrity="sha384-xyz..."
   crossorigin="anonymous"></script>

3. Use a Strict Content Security Policy (CSP): Add a strong CSP header to limit sources for scripts and prevent inline script execution.
   Content-Security-Policy: default-src 'self'; script-src 'self'

4. Enable HTTP Security Headers:
   - **X-Frame-Options: DENY**
   - **X-Content-Type-Options: nosniff**
   - **Strict-Transport-Security: max-age=31536000**; **includeSubDomains; preload**

5. Automate Dependency Auditing: Integrate Retire.js, Snyk, or npm audit in CI/CD workflows to prevent future outdated dependency issues.

6. Security Monitoring: Set up regular scans using ZAP, Nikto, and dependency checkers.

8. Conclusion

This report has highlighted a serious vulnerability in Freshworks' production environment namely the usage of an outdated version of React.js, detected by OWASP ZAP via Retire.js. This vulnerability, if left unpatched, can expose users to significant risks such as cross-site scripting, data theft, and client-side code manipulation.

The risk is compounded by the absence of modern HTTP security headers like CSP, HSTS, and X-Frame-Options, as confirmed via Nikto and Nmap scanning.
It is recommended that Freshworks urgently:

- Update all third-party libraries,
- Implement secure coding practices,
- Employ automated tools to audit dependencies,
- Enforce secure HTTP headers,
- Monitor their web assets for outdated components.

Failure to address these issues may result in exploitation by threat actors, damage to brand reputation, and potential regulatory penalties.