BLOG >

Denial of Service and Source Code Exposure in React Server Components

December 11, 2025 by The React Team

Security researchers have found and disclosed two additional vulnerabilities in React Server Components while attempting to exploit the patches in last week's critical vulnerability.

These new vulnerabilities do not allow for Remote Code Execution. The patch for React2Shell remains effective at mitigating the Remote Code Execution exploit.

The new vulnerabilities are disclosed as:

- **Denial of Service - High Severity**: CVE-2025-55184 and CVE-2025-67779 (CVSS 7.5)
- **Source Code Exposure - Medium Severity**: CVE-2025-55183 (CVSS 5.3)

We recommend upgrading immediately due to the severity of the newly disclosed vulnerabilities.

Note

The patches published earlier are vulnerable.

If you already updated for the Critical Security Vulnerability last week, you will need to update again.

If you updated to 19.0.2, 19.1.3, and 19.2.2, these are incomplete and you will need to update again.

Please see the instructions in the previous post for upgrade steps.

Further details of these vulnerabilities will be provided after the rollout of the fixes are complete.

Immediate Action Required

These vulnerabilities are present in the same packages and versions as CVE-2025-55182.

This includes versions 19.0.0, 19.0.1, 19.0.2, 19.1.0, 19.1.1, 19.1.2, 19.1.2, 19.2.0, 19.2.1 and 19.2.2 of:

- react-server-dom-webpack
- react-server-dom-parcel
- react-server-dom-turbopack

Fixes were backported to versions 19.0.3, 19.1.4, and 19.2.3. If you are using any of the above packages please upgrade to any of the fixed versions immediately.

As before, if your app's React code does not use a server, your app is not affected by these vulnerabilities. If your app does not use a framework, bundler, or bundler plugin that supports React Server Components, your app is not affected by these vulnerabilities.

Note

It's common for critical CVEs to uncover follow-up vulnerabilities.

When a critical vulnerability is disclosed, researchers scrutinize adjacent code paths looking for variant exploit techniques to test whether the initial mitigation can be bypassed.

This pattern shows up across the industry, not just in JavaScript. For example, after Log4Shell, additional CVEs (1, 2) were reported as the community probed the original fix.

Additional disclosures can be frustrating, but they are generally a sign of a healthy response cycle.

Affected frameworks and bundlers

Some React frameworks and bundlers depended on, had peer dependencies for, or included the vulnerable React packages. The following React frameworks & bundlers are affected: next, react-router, waku, @parcel/rsc, @vite/rsc-plugin, and rwsdk.

Please see the instructions in the previous post for upgrade steps.

Hosting Provider Mitigations

As before, we have worked with a number of hosting providers to apply temporary mitigations.

You should not depend on these to secure your app, and still update immediately.

React Native

For React Native users not using a monorepo or `react-dom`, your `react` version should be pinned in your `package.json`, and there are no additional steps needed.

If you are using React Native in a monorepo, you should update *only* the impacted packages if they are installed:

- `react-server-dom-webpack`
- `react-server-dom-parcel`
- `react-server-dom-turbopack`

This is required to mitigate the security advisories, but you do not need to update `react` and `react-dom` so this will not cause the version mismatch error in React Native.

See this issue for more information.

High Severity: Denial of Service

CVEs: CVE-2025-55184 and CVE-2025-67779
Base Score: 7.5 (High)

Security researchers have discovered that a malicious HTTP request can be crafted and sent to any Server Functions endpoint that, when deserialized by React, can cause an infinite loop that hangs the server process and consumes CPU. Even if your app does not implement any React Server Function endpoints it may still be vulnerable if your app supports React Server Components.

This creates a vulnerability vector where an attacker may be able to deny users from accessing the product, and potentially have a performance impact on the server environment.

The patches published today mitigate by preventing the infinite loop.

Note

Additional fix published

The original fix addressing the DoS in CVE-2025-55184 was incomplete.

This left versions 19.0.2, 19.1.3, 19.2.2 vulnerable. Versions 19.0.3, 19.1.4, 19.2.3 are safe.

We've fixed the additional cases and filed CVE-2025-67779 for the vulnerable versions.

Medium Severity: Source Code Exposure

CVE: CVE-2025-55183
Base Score: 5.3 (Medium)

A security researcher has discovered that a malicious HTTP request sent to a vulnerable Server Function may unsafely return the source code of any Server

Function. Exploitation requires the existence of a Server Function which explicitly or implicitly exposes a stringified argument:

```
'use server';

export async function serverFunction(name) {
  const conn = db.createConnection('SECRET KEY');
  const user = await conn.createUser(name); // implicitly stringified, leaked in db

  return {
   id: user.id,
   message: `Hello, ${name}!` // explicitly stringified, leaked in reply
  }}
```

An attacker may be able to leak the following:

```
0:{"a":"$@1","f":"","b":"Wy43RxUKdxmr5iuBzJ1pN"}
1:{"id":"tva1sfodwq","message":"Hello, async function(a){console.log(\"serverFuncti
```

The patches published today prevent stringifying the Server Function source code.

Note

Only secrets in source code may be exposed.

Secrets hardcoded in source code may be exposed, but runtime secrets such as `process.env.SECRET` are not affected.

The scope of the exposed code is limited to the code inside the Server Function, which may include other functions depending on the amount of inlining your bundler provides.

Always verify against production bundles.

Timeline

- **December 3rd**: Leak reported to Vercel and Meta Bug Bounty by Andrew MacPherson.
- **December 4th**: Initial DoS reported to Meta Bug Bounty by RyotaK.
- **December 6th**: Both issues confirmed by the React team, and the team began investigating.
- **December 7th**: Initial fixes created and the React team began verifying and planning new patch.
- **December 8th**: Affected hosting providers and open source projects notified.
- **December 10th**: Hosting provider mitigations in place and patches verified.
- **December 11th**: Additional DoS reported to Meta Bug Bounty by Shinsaku Nomura.
- **December 11th**: Patches published and publicly disclosed as CVE-2025-55183 and CVE-2025-55184.
- **December 11th**: Missing DoS case found internally, patched and publicly disclosed as CVE-2025-67779.

Attribution

Thank you to Andrew MacPherson (AndrewMohawk) for reporting the Source Code Exposure, RyotaK from GMO Flatt Security Inc and Shinsaku Nomura of Bitforest Co., Ltd. for reporting the Denial of Service vulnerabilities.

PREVIOUS
Blog

NEXT
Critical Security Vulnerability in React Server Components

uwu?

Installation

Describing the UI

Adding Interactivity

Managing State

Escape Hatches

React DOM APIs

Community

Code of Conduct

Meet the Team

Docs Contributors

Acknowledgements

More

Blog

React Native

Privacy

Terms