TypeScript: which security bugs does the compiler not catch?

Before your next release, parse every untrusted input with a runtime schema, make typescript-eslint's type-aware `no-unsafe-*` rules errors and check authorization in each handler, not only in middleware. Judge everything else by whether outside data can reach a shell, a query, a file path, a URL, an object merge, `eval` or raw HTML unchecked.

Reviewed **5 Nov 2026** | Next review **5 Dec 2026** | Version **1.1**

Written for
Developers and tech leads writing production TypeScript for Node.js servers and React front ends, and the security people who review it

Latest version
sheets.einblick.dev/secure-typescript/

If you only do five things

1. **Parse every input at runtime.**
   The compiler erases types and `as` casts[1], and `JSON.parse()` and `.json()` return `any`[2]. Parse bodies, params, headers and messages with a schema such as Zod's `z.strictObject()`, which rejects unknown keys[3].
2. **Make the compiler and linter refuse `any`.**
   Keep `strict`, the default since TypeScript 6.0[4], add `noUncheckedIndexedAccess`[6] and lint with typescript-eslint's `strictTypeChecked`[7]. Run `tsc --noEmit` in CI, because Node.js runs `.ts` files unchecked[8].
3. **Authorize where the data is read.**
   CVE-2025-29927 let attackers bypass authorization in Next.js middleware[9], and five more bypasses followed in 2026[10]. Check the user's right to each record in every handler and Server Action[12].
4. **Keep input away from shells, SQL, `eval` and raw HTML.**
   Cross-site scripting, SQL injection, OS command injection and code injection rank 1st, 2nd, 9th and 10th in the 2025 CWE Top 25[13]. Use parameterized queries and `execFile` with an argument array, never `eval` input, and render user text as JSX children.
5. **Patch the framework within days.**
   CVE-2025-55182 (React2Shell), an unauthenticated remote code execution bug in React Server Components rated CVSS 10, was disclosed on 3 December 2025[14]. CISA listed it as exploited two days later and now flags known ransomware use[16]. Patch within the patching sheet's deadlines and run Node.js 22, 24 or 26[17].

Review a TypeScript pull request

1. **Where does outside data enter?**
   Find the handlers, Server Actions, webhooks and queue consumers the change touches, including Next.js folders in brackets[12], and check that each parses its input first.
2. **Did it add `any`, `as`, `!` or `@ts-expect-error`?**
   Ask why for each one near outside data, then follow that data to the sinks in the traps below.
3. **Who may do this, to which record?**
   Missing authorization ranks 4th in the CWE Top 25[13]. Check ownership of the record, not just the login. Server Actions can be called with a direct POST, not just from your UI, so each one checks for itself[12].
4. **Does it run code it didn't write?**
   Neither `vm`[24] nor the Permission Model is a sandbox: the model is a seat belt for trusted code[29], and October's release closed two more gaps in it[22]. Run untrusted code in its own container without secrets.
5. **What leaves the server?**
   Return only the fields the client needs, and keep secrets out of `NEXT_PUBLIC_` variables, which reach the browser[12]. Send new dependencies through the supply-chain sheet's checks.

TypeScript traps and what to use instead

- **`JSON.parse(body) as Order`.** The cast checks nothing, so a missing field or a wrong type flows on[1]. Use `Order.parse(JSON.parse(body))` and treat `as` on outside data as a bug.
- **`exec("convert " + file)`.** `exec` runs a shell, and Node.js says never to pass it unsanitized input[18]. Use `execFile("convert", [file])`, or `spawn` without `shell: true`, and reject a `file` that starts with `-`. A React Native CLI command injection is on CISA's exploited list[16].
- **`merge(target, req.body)` and `obj[a][b] = value`.** `__proto__` or `constructor.prototype` keys in input can add properties to every object in the process. oRPC's deserializer polluted the prototype before Zod validation ran (CVE-2026-28794, critical)[19]. Use a `Map` or `Object.create(null)`[20], and make merge code skip those keys.
- **`path.join(root, req.query.name)`.** Path traversal ranks 6th in the CWE Top 25[13]. Resolve the path with `path.resolve` or `fs.realpath` and reject it unless it starts with the root folder plus `path.sep`.
- **`fetch(req.body.webhookUrl)`.** It can reach internal services and the cloud metadata address `169.254.169.254`. Allowlist hosts, validate the IP address you connect to and disable redirects[21] with `redirect: "error"`. WHATWG URL rules read `0127.0.0.1` as `87.0.0.1`, some OS resolvers as `127.0.0.1`[22].
- **`new RegExp(input)` and nested quantifiers like `(a+)*`.** One crafted string can take exponential time and block the event loop. Cap the input length and use the `re2` package for patterns users supply[23].
- **`eval`, `new Function` and `node:vm`.** `vm` is not a security mechanism[24]. Remote code execution through n8n's expression evaluation is on CISA's exploited list[16]. Run user code only in its own container. `--disallow-code-generation-from-strings` blocks `eval` and `new Function`[25].
- **`dangerouslySetInnerHTML` and user-supplied URLs.** React says raw HTML must come from a trusted source and be sanitized[26]. Accept only `https:` URLs, parsed with `new URL()`, and enforce Trusted Types with `require-trusted-types-for 'script'`[27].

Tools that catch it

- **tsc.** `strict` plus `noUncheckedIndexedAccess`, which `strict` omits[6].
- **typescript-eslint.** `strictTypeChecked` with `parserOptions.projectService: true` enables the `no-unsafe-*` rules, `no-implied-eval` and `no-non-null-assertion`. Add `no-unsafe-type-assertion` yourself[7].
- **CodeQL.** `js/command-line-injection`, `js/path-injection`, `js/request-forgery`, `js/prototype-polluting-assignment` and `js/polynomial-redos` cover TypeScript too[28].
- **Node.js flags.** `--disable-proto=delete` removes `__proto__`[25] but not `constructor.prototype`[20]. `--permission` with `--allow-fs-read` limits what your code reads through `node:fs`[29].

What you can skip

- **Hand-written type guards for request bodies.** They drift from the types they guard. Derive types from the schema with `z.infer`.
- **eslint-plugin-security as your main check.** Its README warns of many false positives[30]. Triage typed lint and CodeQL findings first.
- **Freezing built-in prototypes.** `--frozen-intrinsics` is experimental[25] and freezing can break libraries[20]. Fix the merges instead.
- **Escaping shell arguments yourself.** Drop the shell instead.

What changed in 2026

- **React Server Components had a second critical bug.** An unauthenticated remote code execution flaw, exploitable only in rare cases, was fixed on 14 October in 19.0.9, 19.1.10, 19.2.9 and 19.3.1[15].
- **Next.js shipped more middleware bypasses and critical fixes.** Five middleware or proxy bypasses came in May and July, and five critical remote code execution fixes from August to October[10]. Run at least 16.4.1 or 15.5.28[11].
- **TypeScript turned `strict` on by default.** TypeScript 6.0 did so in March[4]. TypeScript 7.0, the native compiler, has no API until 7.1, so typescript-eslint needs TypeScript 6 alongside[5].
- **Node.js 20 reached end of life on 30 April.** Node.js 26 has been Active LTS since 28 October, and 22 ends on 30 April 2027[17].
- **Trusted Types work in every major browser.** They became Baseline in February[27], and React passes `TrustedHTML` to the DOM unchanged[26].

What this looks like for your code

New project

- Start on Node.js 26 with `strict`, `noUncheckedIndexedAccess` and `strictTypeChecked` as errors.
- Write one schema per input and derive the types from it.
- In Next.js, keep data access and secrets in a `server-only` layer that authorizes[12].

Existing codebase

- Turn on the `no-unsafe-*` rules as warnings and fix request handlers first.
- Search for `exec(`, `shell: true`, `eval(`, `dangerouslySetInnerHTML` and `as any`.
- Check authorization in handlers, not only middleware, and leave Node.js 20.

High-assurance or regulated

- Make CodeQL's `security-extended` suite a required check; private code needs a paid GitHub Code Security license[31].
- Run services with `--permission`, `--disable-proto=delete` and minimal `--allow-*` grants.
- Pen-test `proxy.ts` and route handlers regularly, as Next.js advises[12].

Version history

5 Nov 2026, version 1.1 Reviewed for company readers: CodeQL on private code needs a paid GitHub Code Security license.

Sources

1. TypeScript Handbook: Everyday Types. Microsoft, updated 2 Nov 2026. typescriptlang.org
2. Avoiding anys with Linting and TypeScript. typescript-eslint, 21 Jan 2025. typescript-eslint.io
3. Zod: Defining schemas. Zod, checked 3 Nov 2026. zod.dev
4. Announcing TypeScript 6.0. Microsoft, 23 Mar 2026. devblogs.microsoft.com
5. Announcing TypeScript 7.0. Microsoft, 8 Jul 2026. devblogs.microsoft.com
6. TSConfig Reference. Microsoft, checked 3 Nov 2026. typescriptlang.org
7. strict-type-checked config. typescript-eslint, checked 3 Nov 2026. github.com
8. Modules: TypeScript. Node.js, checked 3 Nov 2026. nodejs.org
9. Next.js advisory GHSA-f82v-jwr5-mffw (CVE-2025-29927). Vercel (Next.js), 21 Mar 2025. github.com
10. Next.js security advisories. Vercel (Next.js), checked 3 Nov 2026. github.com
11. Next.js advisory GHSA-2464-fhr4-w9xw (CVE-2026-96889). Vercel (Next.js), 14 Oct 2026. github.com
12. Data security in Next.js. Next.js Docs, updated 6 Oct 2026. nextjs.org
13. 2025 CWE Top 25. MITRE CWE, Dec 2025. cwe.mitre.org
14. Critical Security Vulnerability in React Server Components. React Blog, 3 Dec 2025. react.dev
15. React advisory GHSA-xqc2-xf4c-x3h3. React (Meta), 14 Oct 2026. github.com
16. Known Exploited Vulnerabilities Catalog. CISA, checked 3 Nov 2026. cisa.gov
17. Node.js release schedule. Node.js, checked 3 Nov 2026. github.com
18. Child process. Node.js, checked 3 Nov 2026. nodejs.org
19. @orpc/client advisory (CVE-2026-28794). GitHub, 2 Mar 2026. github.com
20. Prototype Pollution Prevention Cheat Sheet. OWASP, checked 3 Nov 2026. cheatsheetseries.owasp.org
21. Server-Side Request Forgery Prevention Cheat Sheet. OWASP, checked 3 Nov 2026. cheatsheetseries.owasp.org
22. October 2026 security releases. Node.js, 21 Oct 2026. nodejs.org
23. Don't Block the Event Loop. Node.js, checked 3 Nov 2026. nodejs.org
24. VM (executing JavaScript). Node.js, checked 3 Nov 2026. nodejs.org
25. Command-line API. Node.js, checked 3 Nov 2026. nodejs.org
26. React DOM common components. React Docs, checked 3 Nov 2026. react.dev
27. Trusted Types API. MDN Web Docs, updated 14 May 2026. developer.mozilla.org
28. CodeQL JavaScript and TypeScript queries. GitHub, checked 3 Nov 2026. codeql.github.com
29. Permissions. Node.js, checked 3 Nov 2026. nodejs.org
30. eslint-plugin-security. ESLint Community, checked 3 Nov 2026. github.com
31. CodeQL CLI. GitHub Docs, checked 5 Nov 2026. docs.github.com