Secure development: which practices prevent vulnerabilities, and can you prove it?

This quarter, publish a vulnerability disclosure policy with a security.txt contact, block secrets at push and make review plus one precise static analysis check required before merge. Judge every other practice by whether it removes a whole class of bug by default or leaves evidence that a customer or regulator can check.

Reviewed
5 Nov 2026

Next review
5 Dec 2026

Version
1.2

Written for
Engineering leads, product security teams and developers who build and run their own software, from small product teams to regulated enterprises

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

If you only do five things

1. **Publish a disclosure policy and a contact.**
   Authorize good-faith testing, rule out legal action against it, give a reporting channel and allow coordinated publication[1]. Serve `/.well-known/security.txt` with `Contact`, `Policy` and an `Expires` date under a year away[2]. From December 2027, the CRA requires a disclosure policy and a contact address[3].
2. **Make the safe way the default way.**
   Cross-site scripting and SQL injection top the 2025 CWE Top 25[4]. Offer parameterized queries, auto-escaping templates and shell-free command APIs[1,5]. Write new components in memory-safe languages: memory safety bugs fell below 20% of Android's vulnerabilities in 2025[6].
3. **Block secrets before they reach the repository.**
   Turn on push protection everywhere. On GitHub, private repositories need Secret Protection for it[8]; on GitLab, it takes Ultimate[9]. Rotate leaked secrets at once[7], and load secrets from a secret manager, not code, images or environment variables[10].
4. **Require review and one precise scanner before merge.**
   On the default branch, require a pull request with one approval, **Require approval of the most recent reviewable push** and passing checks[11]. Run CodeQL's `default` suite[12] and block high and critical alerts with **Require code scanning results**[13].
5. **Fix by exploitation and publish what you fixed.**
   Ship fixes within 90 days of a report, or 7 once exploited, per Google Project Zero[14]. Issue a CVE with CWE and CPE for each critical or high flaw that's exploited or that users must patch[1].

Handle a vulnerability report

1. **Is it real?**
   Acknowledge it within three business days and reproduce it before anyone spends a week. Expect volume: by April, curl got twice 2025's rate of reports, nearly all AI-assisted, and 15 to 16% were confirmed[15].
2. **Is it exploited?**
   Check CISA KEV, ENISA's EU KEV catalog[16] and your own logs. If it is and the product is on the EU market, send an early warning through ENISA's platform within 24 hours[3,18]. The EU security laws sheet has the later clocks.
3. **When does the fix ship?**
   Set the date by exploitation and exposure, then severity, and tell the reporter. On internet-facing services you run, an exploited flaw gets the patching sheet's 72 hours. If the fix will take longer, publish a workaround first[20]. Ship security fixes separately from feature updates where feasible, as the CRA requires[3].
4. **What do you publish?**
   An advisory with a CVE, a CVSS score and a CWE, crediting the reporter. NVD no longer routinely adds its own score where the CNA gave one[19], so your record is what customers see.
5. **Where else is the same bug?**
   Search for the pattern across the code base, add a regression test and a rule that catches the next one, as the SSDF advises[20]. The TypeScript, Rust and Python sheets list rules per language.

Automated checks that earn their place

- **Secret scanning with push protection.** It covers Git history, issues and pull requests, and validity checks show which secrets still work[7].
- **Dependency scanning.** If a KEV-listed flaw in a component is exploitable in your product, CISA wants a free patch within 30 days of its fix[5].
- **SAST with a precise rule set.** Start with CodeQL's `default` suite, not the less precise `security-extended`[12]. Each month, tune out rules that are mostly false positives.
- **A DAST baseline in CI.** ZAP's baseline scan spiders for a minute and doesn't attack, so it can run against production[21].
- **Tests for what tools miss.** Tools can't fully verify access control[22], the top risk in the OWASP Top 10:2025[23]. Test that one user can't reach another's records.

Before a risky change ships

- **Decide what counts as risky.** Authentication, authorization, payments, uploads, parsers and new interfaces get a threat model and a reviewer who knows security. The NCSC wants closer oversight of AI-written code that handles authentication, sensitive personal data or secrets[24].
- **Threat model it in half an hour.** Ask what you're working on, what can go wrong, what you'll do about it and whether you did a good enough job, with STRIDE's six categories as prompts[25]. Keep the answers with the code.
- **Pick requirements from ASVS 5.0.** Level 1 holds about 20% of some 350 requirements, and most applications should aim for Level 2[26]. Cite them with the version, as in `v5.0.0-1.2.5`[26].
- **Ship secure defaults.** No shared default passwords, MFA on for admins and logs for customers[5]. The CRA requires a secure default configuration and a reset[3].

Evidence customers and regulators ask for

- **Answers to buyers' standard questions.** CISA's Secure by Demand guide tells buyers to ask for a disclosure policy, CVEs with CWE and CPE fields, a plan to end vulnerability classes, an SBOM and MFA by default[27].
- **A mapping to SSDF 1.1.** Name the task each practice meets, such as PW.7.2 for code review[20]. US agencies may still ask for the attestation form, though it's no longer required[28].
- **CRA technical documentation.** Keep the risk assessment current and the documentation for 10 years or the support period, if longer. SaaS that isn't remote data processing for a product is outside the CRA[3].

What you can skip

- **A paid bug bounty before you can triage.** curl ended its bounty in early 2026 after a flood of AI-written junk reports[15]. A disclosure policy is free.
- **Black-box penetration tests.** OWASP favors source-led tests, and it certifies no one for ASVS[22].
- **Rewriting history after a leak.** Revoke the secret. GitHub calls scrubbing history time-consuming and often unnecessary[7].
- **Waiting for SSDF 1.2.** NIST's December 2025 draft adds continuous improvement and staged updates with rollback[31]. Map to 1.1 now.

What changed in 2026

- **CRA reporting started.** Since 11 September, manufacturers must report actively exploited vulnerabilities through ENISA's platform[18]. The Commission's draft guidance of 27 July explains scope, remote data processing and support periods[29].
- **AI finds flaws at scale.** The NCSC expects a forced correction of technical debt and an influx of updates of every severity[30]. Plan who fixes the findings before you scan with a model.
- **EU vendors can issue CVEs under ENISA.** ENISA, a CVE Root since November 2025, had 20 CNAs by August, vendors among them[17].

Mistakes that cost the most

- **Fixing silently.** Customers can't tell they need the update, and the CRA requires public information on fixed vulnerabilities[3].
- **Letting security.txt expire.** `Expires` is required[2]. Put its renewal in the release calendar.
- **Merging AI-written code nobody read.** Review, understand and check it where the stakes are high[24]. Require approval from someone other than the person who prompted the agent.

What this looks like at your size

Small team, no security staff

- Turn on Dependabot, push protection and CodeQL. Private GitHub repos need the Team plan or higher, with paid Secret Protection and Code Security[8,11].
- Publish security.txt with a mailbox two people read, and require one review before merge.
- Keep one list of open flaws with ship dates.

Growing, with IT or security staff

- Name a product security owner and write playbooks for zero-days and exploited flaws[20].
- Threat model risky changes and take ASVS Level 2 for internet-facing apps.
- Add a ZAP baseline scan against staging and report time to fix monthly.

Enterprise or regulated

- Become a CNA, under ENISA's Root if it covers you[17], and publish CVEs with CWE and CPE.
- Keep CRA technical documentation per product, with risk assessment and vulnerability records.
- Fork ASVS into an internal standard naming your approved libraries[26].

Version history

5 Nov 2026, version 1.2 Clarified that the CRA requires a disclosure policy and a contact address, not security.txt as such.

Sources

1. Secure by Design Pledge. CISA, checked 4 Nov 2026. cisa.gov
2. RFC 9116: security.txt. IETF, Apr 2022. rfc-editor.org
3. Regulation (EU) 2024/2847 (Cyber Resilience Act), Articles 3, 13, 14 and 71, recital 12 and Annex I. EUR-Lex, 2024. eur-lex.europa.eu
4. 2025 CWE Top 25. MITRE, updated 15 Dec 2025. cwe.mitre.org
5. Product Security Bad Practices v2. CISA and FBI, 17 Jan 2025. cisa.gov
6. Rust in Android: move fast and fix things. Google, 13 Nov 2025. blog.google
7. Secret scanning. GitHub Docs, checked 4 Nov 2026. docs.github.com
8. About GitHub Advanced Security. GitHub Docs, checked 4 Nov 2026. docs.github.com
9. Secret push protection. GitLab Docs, checked 4 Nov 2026. docs.gitlab.com
10. Secrets Management Cheat Sheet. OWASP, checked 4 Nov 2026. cheatsheetseries.owasp.org
11. Available rules for rulesets. GitHub Docs, checked 4 Nov 2026. docs.github.com
12. CodeQL query suites. GitHub Docs, checked 4 Nov 2026. docs.github.com
13. Set code scanning merge protection. GitHub Docs, checked 4 Nov 2026. docs.github.com
14. Vulnerability Disclosure Policy. Google Project Zero, checked 4 Nov 2026. projectzero.google
15. High-Quality Chaos. Daniel Stenberg (curl), 22 Apr 2026. daniel.haxx.se
16. Vulnerability Services. ENISA, checked 4 Nov 2026. enisa.europa.eu
17. ENISA scales up its role in the CVE Program. ENISA, 6 Aug 2026. enisa.europa.eu
18. The CRA Single Reporting Platform is launched. ENISA, 11 Sep 2026. enisa.europa.eu
19. NIST Updates NVD Operations to Address Record CVE Growth. NIST, 15 Apr 2026. nist.gov
20. SP 800-218, SSDF Version 1.1. NIST, Feb 2022. csrc.nist.gov
21. ZAP Baseline Scan. ZAP Docs, checked 4 Nov 2026. zaproxy.org
22. ASVS 5.0.0: Assessment and Certification. OWASP, 30 May 2025. github.com
23. OWASP Top 10:2025, Introduction. OWASP, 2025. top10.owasp.org
24. The 'vibe coding spectrum' approach to AI-assisted software development. NCSC (UK), 18 Jun 2026. ncsc.gov.uk
25. Threat Modeling Cheat Sheet. OWASP, checked 4 Nov 2026. cheatsheetseries.owasp.org
26. ASVS 5.0.0: What is the ASVS. OWASP, 30 May 2025. github.com
27. Secure by Demand Guide. CISA and FBI, Aug 2024. cisa.gov
28. M-26-05: Adopting a Risk-based Approach to Software and Hardware Security. OMB, 23 Jan 2026. whitehouse.gov
29. Commission guidance on the application of the CRA, C(2026) 5252. European Commission, 27 Jul 2026. digital-strategy.ec.europa.eu
30. Preparing for a 'vulnerability patch wave'. NCSC (UK), 1 May 2026. ncsc.gov.uk
31. SP 800-218 Rev. 1, SSDF Version 1.2 (draft). NIST, 17 Dec 2025. csrc.nist.gov