

Coding

1. Assume external systems/data are insecure; reject non-compliant requests.
2. Use SBOM for the software supply chain.
3. Review all development tools (IDE, version control, laptops, etc.).
4. Keep frameworks/dependencies up to date and patch systems.
5. Prioritize usability when developing security features; test them.
6. Create threat models for all critical systems.
7. Automate with SAST tools, linters, and unit tests.
8. Report reproducible security bugs in languages/frameworks.
9. Use languages and frameworks as intended.
10. Encrypt data when appropriate, especially during transfers.
11. Use prepared statements or ORMs for database operations.
12. Prefer proven solutions for auth, sessions, access, and encryption.
13. Never store secrets in code; use vaults.
14. Scan code for secrets (e.g. Git pre-commit hooks).
15. Apply token-based security for web apps and APIs.
16. Capture and handle exceptions properly.
17. Use managed and strongly typed languages for memory safety.
18. Never deserialize untrusted data.
19. Prevent race conditions with proper resource locking.
20. Write readable, testable, and maintainable code.
21. Be cautious with direct system calls - or avoid them.
22. Avoid building up technical debt.
23. Enable strict mode in your language when available.
24. Always initialize variables; don't leave behavior to chance.
25. Treat compiler warnings as errors; fix them.
26. Use signed Git commits.
27. Adopt conventional commits for traceable versioning and patching.
28. Favor trunk-based development over Git Flow (reduce MTR).
29. Implement Docker security: review images, use multistage builds.
30. Avoid storing secrets in Dockerfiles.
31. Conduct code reviews; use protected branches.
32. Collaborate closely with operations.
33. Apply a zero-tolerance policy for Greenfield projects ("Broken Window Theory").

Operations & Infrastructure

1. Apply: Least Privilege, Zero Trust, Defense in Depth, and Security by Obscurity.
2. Follow security policies defined by your security team.
3. Use HTTPS for all web-based systems and set appropriate security headers.
4. Ensure tightly monitored access control for source code and application data.
5. Review systems affecting you and perform hardening measures.
6. Enforce rate limits and restrictions - nothing should ever be "unlimited."
7. Use service accounts for systems that operate in a networked environment.
8. Decommission legacy systems/devices with intention; document the process.
9. Offer MFA to users whenever possible.
10. Consider physical security for your systems.
11. Plan and test backups and disaster recovery scenarios.
12. Never use wildcards like * in configurations (e.g., CORS).
13. Don't use production data in dev/test environments unless it's masked.
14. Never run production systems as root.
15. Use Infrastructure as Code (IaC).
16. Force strong collaboration with developers.

Organisation & Training

1. Ensure full compliance with all rules, policies, and regulations.
2. Follow any security framework adopted by your organization.
3. Share knowledge openly - including failures. Mistakes are acceptable; repeating them is not.
4. Conduct annual software inventory across the company.
5. Provide regular secure coding training.
6. Consider establishing a "Security Champion" program.
7. Involve employees in incident response and postmortems.
8. Define clear roles and responsibilities for handling incidents.
9. Run cybersecurity tabletop exercises.

Architecture

1. Design systems to remain functional even after a breach; plan for failure scenarios.
2. Avoid storing passwords directly - use identity providers or strong encryption.
3. Protect the privacy of users, citizens, customers, and employees.
4. Enable the use of password managers for user convenience and safety.
5. Default app settings should prioritize strong security and privacy.
6. Avoid placing critical logic in the front-end.
7. Never trust the browser.
8. Use "allowlist" instead of "denylist".
9. Document key architecture decisions using ADRs.
10. Utilize issue trackers, SBOM management, and patch management tools.