10 TIPS FOR
SECURE DEVELOPMENT

If you've seen the damage done to brands from security breaches over the last few years, you know keeping code safe is serious business. The good news: Secure development doesn't have to come at the cost of productivity or collaboration. Here are 10 tips for building secure processes that keep your team working at their best.

1. Build a culture of security

Create processes that make security everyone's job. Protect all physical devices like laptops and tablets, enforce strong password policies, and use two-factor authentication. Secure access to internal corporate networks with VPNs and hold regular trainings on phishing and the dangers of public networks.

2. Always encrypt data

Strong encryption can protect data transmission even if a bad actor obtains access. Encrypt your data both at rest, when it's being stored, and in transit, when it's transmitted between storage and end user.

3. Adopt strong identity management practices

Use a centralized identity management system like LDAP, coupled with a Single Sign On (SSO) solution. The more passwords people have to remember, the more likely they are to default to insecure practices. They might write passwords down or reuse easy-to-remember (and easy to crack) passwords.

4. Prevent database attacks

Bad actors can use incoming data to try to access your systems, using tricks like SQL injection. To stop them, build validation into your code. Only accept validated data into production systems, monitor systems for errors and edge cases, and restrict access to data with an identity management system.

5. Segregate sensitive data

Limit the places sensitive data exists and the opportunities attackers have to exploit them. Store your passwords in a password manager rather than entrusting them to individuals. And never commit passwords, access/API tokens, encryption keys, and other sensitive data to publicly-accessible repositories.

6. Enforce protections on source code

Control access to sensitive information and the source code for the software that manages it. More granular controls help lock down this information without creating an excessively restrictive, security-bound environment.

7. Include legacy core systems

Some development organizations use tools like mainframes, which can be powerful but hard to secure. Periodically evaluate your legacy systems. Then balance the cost of updating them against the risk of security breaches they pose.

8. Automate processes to prevent bugs and errors

Identify and automate any critical tasks that are repetitive or tedious. For example: consider Continuous Integration and Delivery (CI/CD) tools, which test and evaluate your code every time a commit is pushed to a repository. Use CI/CD to check your code and its dependencies for security flaws.

9. Enforce manual reviews with required code review

The more people review a codebase, the more likely they are to find errors or vulnerabilities. Reviews also share institutional knowledge and provide learning and mentoring opportunities for developers of all skill levels.

10. Move security all the way left

Shift security from the delivery side of your timeline all the way to the beginning. This move ensures that security remains front and center throughout the entire product lifecycle. Involving security teams early can also influence design, development, and maintenance decisions before they become too difficult and expensive to change.

It can be difficult to balance security considerations with the freedom developers need to innovate, but with the right tools and partners, it's well within your reach. GitHub is here to help you go beyond the basics and start building customer trust through a solid security strategy.

Let us know how we can help you get there.