GitHub

Secure development best practices for financial services

Introduction

The way people build software is evolving fast—from mobile applications to the infrastructure of our largest financial institutions. The early days of Facebook and flip phones have come to an end, meaning security is no longer an acceptable afterthought for most developers.

Fast-paced development brings a greater risk of security breaches. It's not uncommon for major organizations to accidentally leave passwords in public source code, leak sensitive personal customer data, and deploy critical applications into production with clearly exposed vulnerabilities. As organizations strive to ship new software faster, they are struggling to ensure that software stays secure.

The tension between working fast and building securely is clear for financial institutions. New development methodologies bring code to market faster, but inadequate tooling and processes can lead to gaps in security. Fortunately, there are a few best practices teams can follow to keep their code safe from errors, breaches, and other exploitable vulnerabilities.

Manage and store your secrets securely

Always use a secret management system to manage the keys to your software and avoid leaving account information in repositories that are available to the public. Never store passwords, OAuth tokens, or any other sensitive production credential in a repository.

Sensitive data like personally identifiable information (PII) should be stored in a secure and encrypted database. This type of data requires stricter handling guidelines because of the risk it poses to the individuals if it's ever compromised. Data breaches involving PII can have far reaching consequences like identity theft. For financial institutions, it's crucial to separate PII storage from non-production environments to limit unnecessary access.

Automate your secure development workflows

With continuous integration, code scanning, and deployment tools, it is easier than ever to automate the testing and security of your code. CI and CD tools help you build automated workflows that test code for bugs, security vulnerabilities, and ensure they meet development standards before deployment.

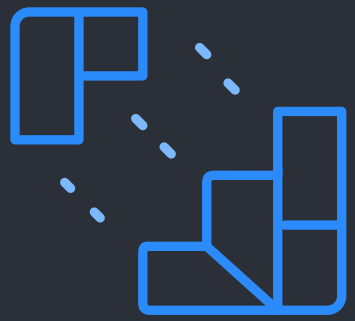
These tools help enforce secure development processes, and allow for a consistent, repeatable, and transparent build environment. Tooling has made it easier than ever to automate successful development environments without security labs.

For example, financial services companies can add gates that make sure the right people are reviewing code in compliance with the Sarbanes-Oxley Act (SOX).

Manage risk with detailed logs and audits

Processes can never entirely eliminate risks, but they can provide an effective way to manage them. Maintaining detailed logs of workflows, changes, and audit trails help your team trace the source of problems as they arise. When you can pinpoint an error, vulnerability, or breach, it can reduce the time it takes to fix issues and remove barriers to secure code.

This approach to auditing is an alternative to complex user permissions and settings that get in the way of efficient collaboration. While you can't guarantee that someone doesn't misuse or access your system without permission, you can make sure you have a complete audit trail to monitor and identify these actions. If and when mistakes or unwanted changes happen, organizations can fall back on built-in auditing to make corrections without bulky processes that prevent developers from working together.



Take advantage of external tools and integrations

Although it can be an effective option to create all of your tooling in-house while an organization is small, it becomes a major burden and security risk as teams grow. Instead of spreading organizational resources too thin, consider using external tools and integrations built by top-tier partners. Identify best-in-breed tools and integrations that solve your organizations core needs instead of trying to build them internally.

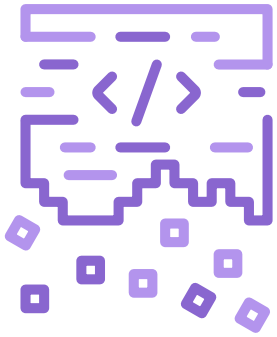
From CI to chat, external tools and integrations can provide your team with the best possible tools for every job. And with full technical support and consistent updates, your team will be able to focus on secure development practices without worrying about maintaining tools.

Automate your bug reporting

Manually tracking and reporting bugs can be time consuming and often inaccurate. Simplify bug reporting with tools that automatically detect bugs like Rollbar or Sentry through log files. Once detected, they're recorded into your bug tracking system.

Continuously improve your security processes

When it comes to security, best practices, technologies, and methodologies evolve constantly. Your approach should be equally flexible, adapting quickly to keep your software safe.



Avoid software rot

Software rot happens when the quality of your code—and the code it depends on—deteriorates over time. If one part of your code is outdated or unmaintained, problems can spread to other areas of your software. If one of your components contains a security vulnerable, your entire system is open to breaches.

To avoid rot, organizations should update code, unit tests, static code analysis tools, and other checks to find new problems, even in old code.

Clean and validate your data

Collecting and interacting with data is essential to how organizations use the software they build. Ensuring data is clean and validated keeps applications secure and protects both the organization and end-users from malicious misuse. Tools or form validations check to see if information meets validation parameters before it can be submitted to your system.

Preventative security processes around your data collection can also stop malicious attacks. SQL Injection, a common data-based vulnerability, can happen through your applications if there aren't proper validation processes in place.

With a SQL Injection, an attacker can:

- Bypass authentication or even impersonate specific users
- Gain access to sensitive data through queries
- Delete records and vital information from a database
- Reconfigure a database server to allow access to operating systems

Shift security left

Security was once an afterthought for many developers brought in at the end of development lifecycles and expected to fix bugs and solve problems without proper context. With more threats to security than ever before, times have changed.

"Shifting security left" means moving critical conversations to the beginning of the process. Engaging security teams early on can help facilitate discussions and influence design decisions before they are too difficult and costly to change.

Although the concept is simple enough, all teams involved in development have to take on the responsibility of security. Some companies even invite security specialists into scrum teams to make sure security is top of mind throughout the process.

Encrypt, encrypt, encrypt

A number of different cyber attacks can occur when organizations don't properly encrypt their data. One of the best ways to avoid unauthorized access is by using a secure, TLS encrypted connection over HTTPS. HTTPS helps secure communication using cryptographic protocol that validates a safe connection. The protocol provides layered end-to-end encryption of information sent from servers to users. The encryption helps prevent man-in-the-middle attacks and other breaches that take advantage of unsecured connections.

With the arrival of services like Let's Encrypt, the barriers to obtaining SSL certificates for your website are lower than ever. Any web-based effort should be HTTPS from the start. No excuses!

Building secure software takes time and a commitment to ensuring each aspect of your development lifecycle leverages security best practices. While security breaches, hacks, and exploitations continue to grow in sophistication, developers need to constantly improve their defenses. These best practices are a solid foundation for building more secure software—but they're only just the beginning.

Tell us how we can help.

For support questions, email sales@github.com