SECURE
SOFTWARE DEVELOPMENT
LIFECYCLE

WHO AM I?

LUÍS PAULO TONIETTE FRANÇA

SOFTWARE ENGINEERING MANAGER

INFORMATION SECURITY CONSULTANT

LOVELY HUSBAND, SON, BROTHER, FRIEND, DAD

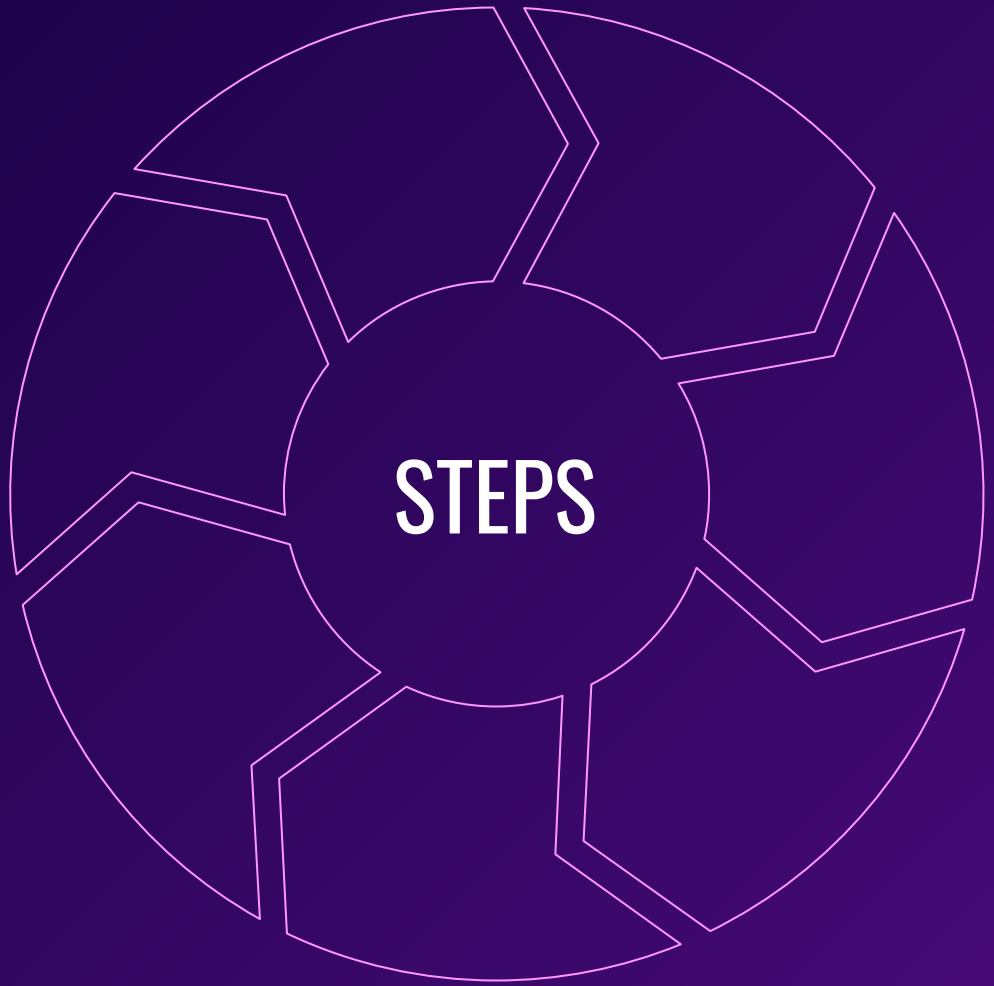
GETTING CONTEXT

Security plays a crucial role in any application that involves important features. This may range from safeguarding your database against malicious attacks to employing fraud detection procedures to assess potential customers before they are added to your platform.

Security must be considered at every stage of the software development life cycle (SDLC) and should be a top priority for your developers as they implement the software requirements. In this article, we will examine methods to establish a secure SDLC that can help you detect issues in the requirements phase before they become security problems in the production phase.

By devoting effort towards security concerns, SDLC issues can be resolved before the application is deployed in production. This lowers the chances of identifying security vulnerabilities in your app and aims to minimize their impact when they are detected.

The objective of a secure SDLC is not to entirely replace conventional security measures such as penetration tests, but instead to include security as a part of the developer's responsibility and enable them to develop secure applications from the very beginning.


STEPS

To implement a secure SDLC in an enterprise context, it is essential to establish security policies and guidelines that are followed throughout the software development process. This involves training developers and other stakeholders on security best practices, using security testing tools and techniques, and conducting regular security audits to ensure that the software remains secure over time. By integrating security into the SDLC, enterprises can reduce the risk of security breaches, protect sensitive data, and maintain customer trust.

1 - PLANNING
2 - ANALYSIS
3 - DESIGN
4 - IMPLEMENTATION
5 - TESTING
6 - DEPLOYMENT
7 - MAINTENANCE

1 - Provide Training

Security is everyone's job. Developers, service engineers, and program and product managers must understand security basics and know how to build security into software and services to make products more secure while still addressing business needs and delivering user value.

Effective training will complement and re-enforce security policies, SDL practices, standards, and requirements of software security, and be guided by insights derived through data or newly available technical capabilities.

Although security is everyone's job, it's important to remember that not everyone needs to be a security expert nor strive to become a proficient penetration tester. However, ensuring everyone understands the attacker's perspective, their goals, and the art of the possible will help capture the attention of everyone and raise the collective knowledge bar.

2 - Define Security Requirements

The need to consider security and privacy is a fundamental aspect of developing highly secure applications and systems and regardless of development methodology being used, security requirements must be continually updated to reflect changes in required functionality and changes to the threat landscape. Obviously, the optimal time to define the security requirements is during the initial design and planning stages as this allows development teams to integrate security in ways that minimize disruption. Factors that influence security requirements include (but are not limited to) the legal and industry requirements, internal standards and coding practices, review of previous incidents, and known threats. These requirements should be tracked through either a work-tracking system or through telemetry derived from the engineering pipeline.

3 - Define Metrics and Compliance Reporting

It is essential to define the minimum acceptable levels of security quality and to hold engineering teams accountable to meeting that criteria. Defining these early helps a team understand risks associated with security issues, identify and fix security defects during development, and apply the standards throughout the entire project. Setting a meaningful bug bar involves clearly defining the severity thresholds of security vulnerabilities (for example, all known vulnerabilities discovered with a “critical” or “important” severity rating must be fixed with a specified time frame) and never relaxing it once it's been set.

In order to track key performance indicators (KPIs) and ensure security tasks are completed, the bug tracking and/or work tracking mechanisms used by an organization (such as Azure DevOps) should allow for security defects and security work items to be clearly labeled as security and marked with their appropriate security severity. This allows for accurate tracking and reporting of security work.

4 - Perform Threat Modeling

Threat modeling should be used in environments where there is meaningful security risk. Threat modeling can be applied at the component, application, or system level. It is a practice that allows development teams to consider, document, and (importantly) discuss the security implications of designs in the context of their planned operational environment and in a structured fashion.

Applying a structured approach to threat scenarios helps a team more effectively and less expensively identify security vulnerabilities, determine risks from those threats, and then make security feature selections and establish appropriate mitigations.

5 - Establish Design Requirements

The SDL is typically thought of as assurance activities that help engineers implement “secure features”, in that the features are well engineered with respect to security. To achieve this, engineers will typically rely on security features, such as cryptography, authentication, logging, and others. In many cases, the selection or implementation of security features has proven to be so complicated that design or implementation choices are likely to result in vulnerabilities. Therefore, it’s crucially important that these are applied consistently and with a consistent understanding of the protection they provide.

6 - Define and Use Cryptography Standards

With the rise of mobile and cloud computing, it's critically important to ensure all data, including security-sensitive information and management and control data, is protected from unintended disclosure or alteration when it's being transmitted or stored. Encryption is typically used to achieve this. Making an incorrect choice in the use of any aspect of cryptography can be catastrophic, and it's best to develop clear encryption standards that provide specifics on every element of the encryption implementation. This should be left to experts. A good general rule is to only use industry-vetted encryption libraries and ensure they're implemented in a way that allows them to be easily replaced if needed.

7 - Manage the Security Risk of Using Third-Party Components

With the rise of mobile and cloud computing, it's critically important to ensure all data, including security-sensitive information and management and control data, is protected from unintended disclosure or alteration when it's being transmitted or stored. Encryption is typically used to achieve this. Making an incorrect choice in the use of any aspect of cryptography can be catastrophic, and it's best to develop clear encryption standards that provide specifics on every element of the encryption implementation. This should be left to experts. A good general rule is to only use industry-vetted encryption libraries and ensure they're implemented in a way that allows them to be easily replaced if needed.

8 - Use Approved Tools

Define and publish a list of approved tools and their associated security checks, such as compiler/linker options and warnings. Engineers should strive to use the latest version of approved tools, such as compiler versions, and to take advantage of new security analysis functionality and protections.

9 - Perform Static Analysis Security Testing

Analyzing the source code prior to compilation provides a highly scalable method of security code review and helps ensure that secure coding policies are being followed. SAST is typically integrated into the commit pipeline to identify vulnerabilities each time the software is built or packaged. However, some offerings integrate into the developer environment to spot certain flaws such as the existence of unsafe or other banned functions and replace those with safer alternatives as the developer is actively coding. There is no one size fits all solution and development teams should decide the optimal frequency for performing SAST and maybe deploy multiple tactics—to balance productivity with adequate security coverage.

10 - Perform Dynamic Analysis Security Testing

Analyzing the source code prior to compilation provides a highly scalable method of security code review and helps ensure that secure coding policies are being followed. SAST is typically integrated into the commit pipeline to identify vulnerabilities each time the software is built or packaged. However, some offerings integrate into the developer environment to spot certain flaws such as the existence of unsafe or other banned functions and replace those with safer alternatives as the developer is actively coding. There is no one size fits all solution and development teams should decide the optimal frequency for performing SAST and maybe deploy multiple tactics—to balance productivity with adequate security coverage.

11 - Perform Penetration Testing

Penetration testing is a security analysis of a software system performed by skilled security professionals simulating the actions of a hacker. The objective of a penetration test is to uncover potential vulnerabilities resulting from coding errors, system configuration faults, or other operational deployment weaknesses, and as such the test typically finds the broadest variety of vulnerabilities. Penetration tests are often performed in conjunction with automated and manual code reviews to provide a greater level of analysis than would ordinarily be possible.

12 - Establish a Standard Incident Response Process

Preparing an Incident Response Plan is crucial for helping to address new threats that can emerge over time. It should be created in coordination with your organization’s dedicated Product Security Incident Response Team (PSIRT). The plan should include who to contact in case of a security emergency, and establish the protocol for security servicing, including plans for code inherited from other groups within the organization and for third-party code. The incident response plan should be tested before it is needed!

WHAT DO THEY SAY?

"If you think technology can solve your security problems, then you don't understand the problems and you don't understand the technology."

—BRUCE SCHNEIER

FINALLY DONE!

THANK YOU!

DO YOU HAVE ANY QUESTIONS?