exalate

How to Best Secure Agile Development (Complete 2023 Guide)

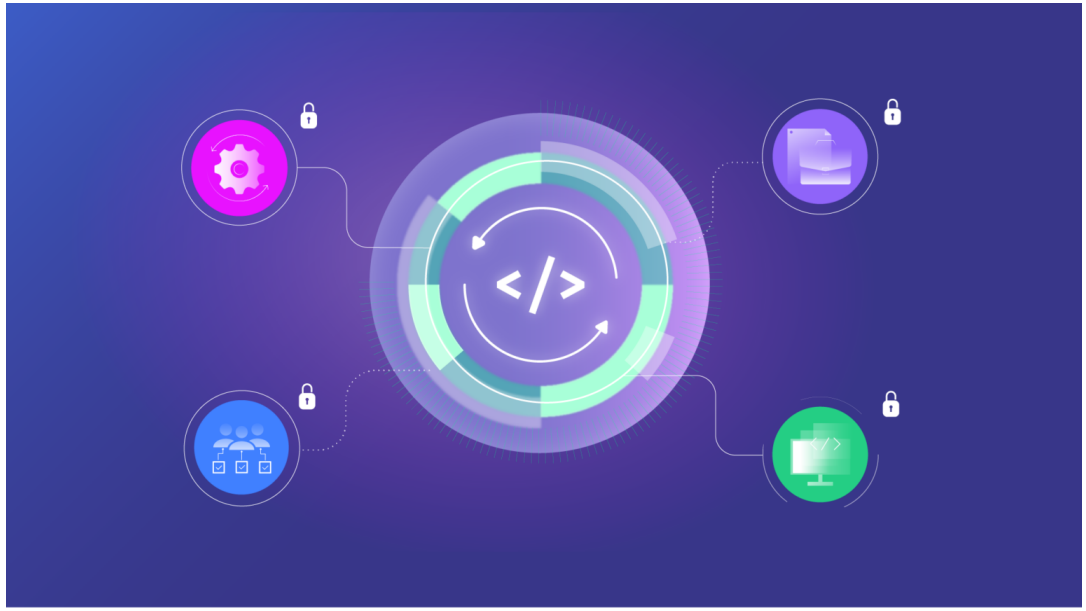


Table of contents

This article was written by Borislav Kiprin from Crashtest Security.

Agile development helps software firms adopt lightweight, iterative development cycles. The development methodology emphasizes small, manageable teams with cross-functional collaboration for frequent updates and releases.

While Agile relies on massive cultural shifts in code development and testing, the model also necessitates a novel approach for implementing security in Agile.

This article discusses pertinent risks that arise from improper implementation of security techniques while delving into the best practices for securing an Agile development cycle.

Risks of Not Considering the Security in Your Agile Development

Agile processes typically open up production and development environments to distributed teams, leading to leakage of important information due to authorization creep. While Agile methodologies have evolved to accommodate innovation and time-bound delivery, security often takes a back seat since security assessments slow down development.

Security practices are particularly challenging to implement uniformly across the software development life cycle. The build cycle is continuous, and security tests need to be carried out outside the development workflow. This presents various security risks for the organization, including:

Insufficient Policies Governing the Use of Open-Source Components

The core principle of Agile is to discourage re-inventing the wheel. Additionally, teams are encouraged to include open-source and commercial proprietary solutions to shorten development lifecycles. However, though such open-source integrations offer various advantages, they also introduce significant security risks that may impact development.

A typical scenario is when developers are unaware of module dependencies and the number of code libraries they are importing. As a result, tracking open-source code changes is complex to manage, making it challenging to ensure the software works reliably and securely.

Reliance on open-source integrations also complicates extracting information about specific vulnerabilities and fixes discovered by the contributing community. In the absence of defined policies that govern open-source integrations, tool selection, and tech stack

management, unveil challenges to mitigate threats.

Unlimited Access to Development Pipelines and Source Code Repositories

Agile promotes efficient collaboration among cross-functional, distributed teams that follow an iterative development approach. To ensure teams work on a *single-source-of-truth*, organizations rely on public and private source code repositories for maintaining, sharing, and versioning codes.

While repositories enable seamless version control, teams often fail to interpret the version of source code in use, making it difficult to pinpoint specific vulnerabilities and apply the mitigation.

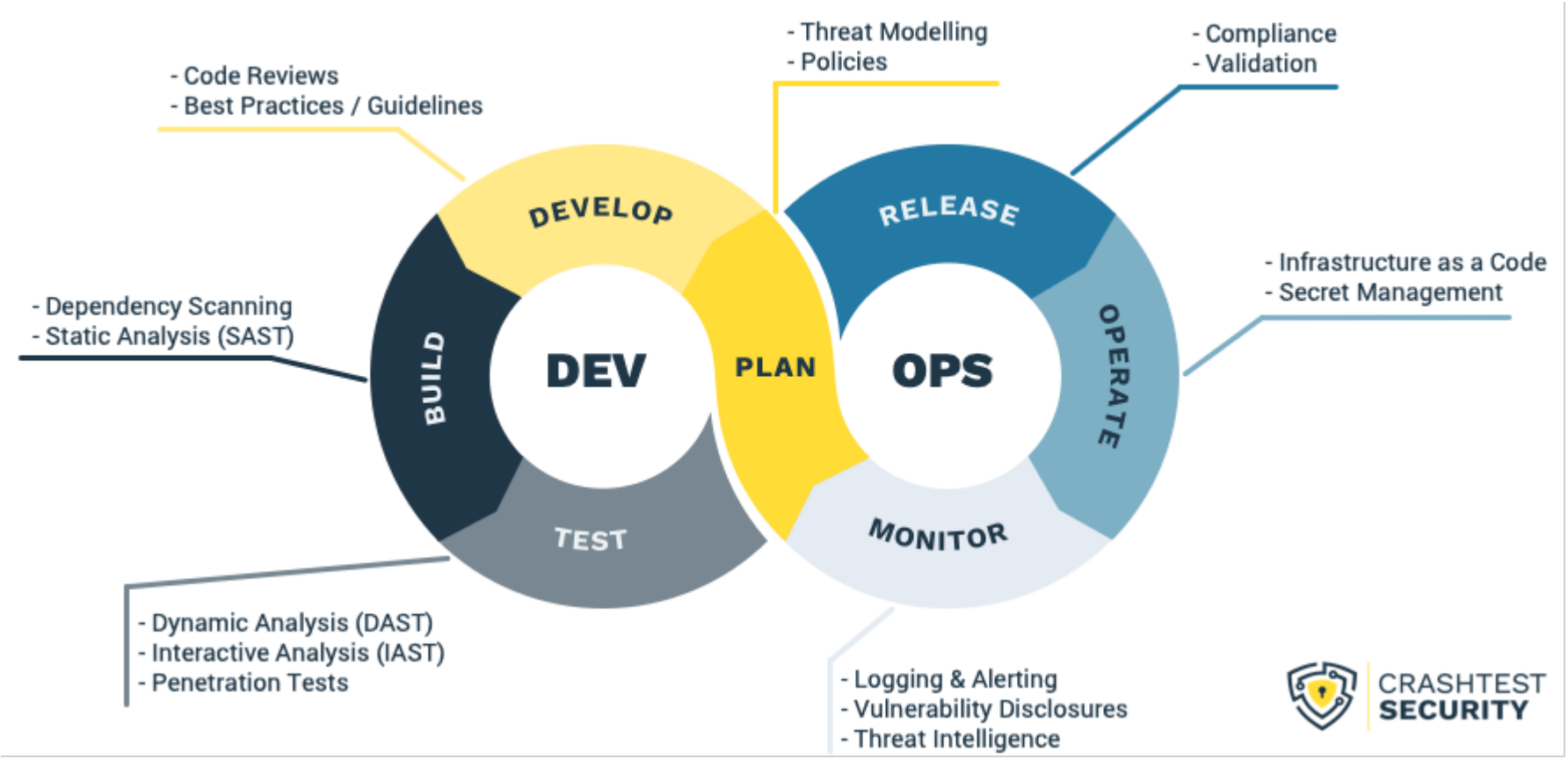
Additionally, when pushed into source code repositories, privately-crafted code exposes vulnerable entry points. Malicious actors target such exposed repositories to identify hidden functionalities and then combine them with reconnaissance tactics to orchestrate attacks.

Insecure Management of Sensitive Data

The modern Agile development process integrates multiple sources for organization-wide analytics and data operations. A complex integration requires systematic and accurate classification of data to control the level of access to sensitive organizational data.

Besides this, chunks of test data produced by the test team are often a real cut of the live data. Though such data is used internally for testing purposes, the absence of adequate data sanitization invites attack vectors to abuse vulnerabilities.

Most Agile organizations lack the policies to protect their data from internal threat actors. Due to the lack of strict data classification policies, team members are usually unaware of a comprehensive security requirement, data sensitivity, and their responsibility in safeguarding it. The collaboration rate required for seamless delivery also makes it difficult for a security team to implement security controls.



Security Methodologies in Agile Development

Agile software development introduces organization-wide cultural changes focusing on modular, more manageable cross-functional teams. While these changes enable development that is highly flexible and responsive to evolving requirements, organizations require the adoption of special techniques to secure code and deployment environments.

Following are some of the best practices to secure applications in an Agile environment:

Securing Agile Delivery Pipelines

The Agile methodology emphasizes Continuous Integration (CI) and Continuous Delivery (CD) pipelines to enable quicker deployments. To minimize security risks, Agile teams should consider integrating automated testing mechanisms into every stage of the SDLC. In addition, all commits made by the development team should go through security experts to ensure the application developed is secure.

Apart from manual code reviews, there should also be static code analyzers integrated with the CI pipeline and automated unit tests for running scheduled checks. Active security audits and automatic security checks are also recommended during the CI/CD process to help identify security vulnerabilities before a code moves to production.

The cost of identifying a security vulnerability at an early stage is considerably less compared to finding one in production, which can lead to severe consequences, including sensitive data theft, substandard application performance, and loss of reputation.

Enforce Continuous, Organization-Wide Participation

Compared to traditional practices, an Agile process presents a shift in development practices that should be accompanied by changes in organizational culture toward security practices. It is recommended that security professionals engage with developers to form an approved guideline of coding standards, design patterns, and design decisions.

Organizations should also undertake diligent threat modeling while formulating continuous security training that focuses on behavioral change rather than just awareness. Security professionals should also document secure development criteria on shared platforms so that other organizations can reference them in their processes.

Firms can also enable Agile security practices by instrumenting security policies in newly developed microservices, applications, integrations, and APIs.

Adopt Passive Reviews

The agile development methodology is characterized by shorter iterations, quick builds, and frequent releases for new features. It is, therefore, necessary to review all application changes that require iteration and how they affect the system's security before committing them.

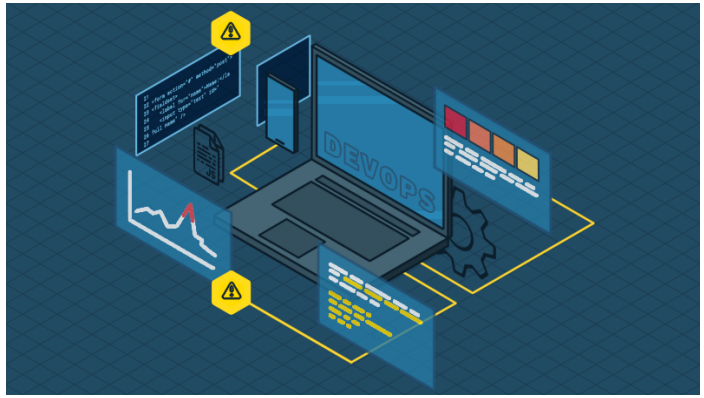
Passive code reviews help QA professionals to understand applications without having to scan every single line of source code. Instead, QA professionals can skim through the code, the code review systems monitor their movements around the codebase and offer timely, appropriate supplementary information for the source code. Some passive review security tools also generate model diagrams that visualize the logic implemented by the code, simplifying troubleshooting.

Ensure Proactive Controls for Agile Development Teams

An organization should support developers in implementing controls that promote a secure code development practice. OWASP lists various security techniques that ensure all tiers of the application are built, emphasizing security. Some of these techniques include:

- **Defining security requirements** - Agile methods should be built on industry standards, past vulnerabilities, and applicable laws to ensure the product satisfies all security properties.
- **Utilizing security libraries and frameworks** - These libraries and frameworks have embedded security measures to protect code against design and implementation flaws that lead to security flaws.
- **Keeping data stores secure** - Teams should ensure both NoSQL and relational databases are kept safe using secure queries, authentication, configuration, and communication.
- **Encoding and escaping data** - Encoding and escaping data are used to prevent injection vulnerabilities. **Escaping** involves using special characters to avoid misinterpretation of strings, while **Encoding** refers to the translation of characters into equal values with different formats, making them inaccessible to malicious interpreters.

Other proactive controls for comprehensive Agile security include **Input Validation, Identity and Access Management (IAM), Error & Exception Handling,** and **Security Logging & Monitoring**.



Secure Agile Development FAQs

How Does Security in Agile Development Differ from Traditional Development Approaches?

While Agile and Traditional security relies on adopting similar security policy, the implementation of security features varies widely. In Agile methods, rapid release cycles imply frequent security test runs, which means that a detailed code analysis may ultimately slow down development and delivery.

To tackle this, security risk indicators should be set to match the delivery rate, enabling Agile teams to undertake security reviews while dealing with frequent changes. Agile security practices should also be categorized into those completed at the beginning of development and those implemented during every sprint, depending on whether they need to be performed once or continuously.

Why do Agile Teams Fail to Secure Sensitive Data?

The most common mistake for Agile teams trying to secure sensitive information is improper data classification. Most teams fail to specify the data that needs special protection, making it difficult to enforce an effective data classification policy.

Additionally, software development teams fail to optimally encrypt sensitive data *in transit* or *at rest*, making it easily accessible to threat actors. Another point of concern occurs when teams misuse cloud storage platforms, storing credential and configuration data without correctly interpreting the provider's security policies. This makes it challenging to implement a comprehensive strategy from a security perspective.

Recommended Reads:

- The Project Management Blueprint for Agile Collaboration Between Teams
- ITIL 4 and Agile Integration (podcast)

- Project Management Best Practices for Aligning Disparate Teams
- B2B Integration: The Comprehensive Guide