THE AGILE SECURITY MANIFESTO

The Agile Manifesto was released in 2001 as a reaction to documentation-heavy software development practices.

FIFTEEN YEARS LATER,

similar inefficiencies are plaguing application security efforts in software development. Security is often focused on testing, and security activities are often conducted outside and apart from the software development process. As a result, the outcomes of security activities are presented in documents and outputs that do not naturally fit any of the software development activities.

Security is seen as something separate from and external to software development. It is time to change the approach to building secure software within an agile methodology.

SYNOPSYS®

The Four Principles of Building Security Into Agile Development

As we build secure software in an agile environment, it is essential to focus on the four principles below. These principles are patterned after the ones in the original Agile Manifesto: while we value the things on the right, we must value the things on the left more.

The goal is to guide the development of new activities and make adjustments to existing activities to make it natural and efficient to build security into an agile process.

1 RELY ON DEVELOPERS AND TESTERS MORE THAN SECURITY SPECIALISTS

While experienced security specialists are valuable, few agile teams have dedicated security specialists. This means that most of the time, security must be done by the same team that does the rest of the software development. Continuous Integration/Continuous Deployment (CI/CD) can't wait for an external security review before the code moves forward on the assembly line. Security must be an integrated part of our everyday development and testing of code. Our team has to own security the same way it owns the user experience, reliability, performance, and all the other non-functional requirements. An important part of this principle is that security is not someone else's job.

SECURITY IS—FIRST AND FOREMOST—THE SOFTWARE TEAM'S JOB.

Security specialists function in agile development like specialists of any other discipline (performance, reliability, user experience, etc.). They help our team understand how it can achieve our security goals. Security specialists can also take on tasks that require knowledge or experience, like integrating security tools into the development tool chain and the CI/CD environment. Security specialists need to be involved as we elaborate user stories and prioritize our backlog.

2 SECURE WHILE WE WORK MORE THAN AFTER WE'RE DONE

If an agile team is working well and producing good software, it probably needs very little change to produce secure software. If the existing process is not producing good software, then adding security activities will not fix that. Assuming our team is happy with their current agile process, we must make security activities a part of that process. We don't want security activities that cause us to stop what we are doing, go think about security for a while, and then come back to what we were doing.

INSTEAD, WE SHOULD INTEGRATE SECURITY ACTIVITIES INTO WHAT WE ARE ALREADY DOING.

An important embodiment of this principle appears in the way security specialists interface with development. The security specialists can do their work however they like. But their interface to our development team has to use the tools the developers already use. If our development team uses a whiteboard, sticky notes, an online tool, or something else, then the security priorities have to be embodied right there. For example, we could use gold padlock stickers on the whiteboard, different color sticky notes, or security tags in the online tool. Fundamentally, the planning tool that our team uses for normal software development must also contain our security priorities.

3 IMPLEMENT FEATURES SECURELY MORE THAN ADDING ON SECURITY FEATURES

Small, iterative changes evolve the security of our software over time. We should focus on business value of software and achieving the software's mission, not adding or implementing security features. As much as possible, this should be intrinsic to what we do through secure-by-default frameworks, rather than explicit calls to security libraries or custom implementations of security functions. The implementation of security features such as authentication controls, password storage schemes, and cryptographic algorithms is best left to experienced professionals whose day-to-day work focuses on these particular topics.

WE FAVOR USING TRIED AND TRUE SECURE IMPLEMENTATIONS
INSTEAD OF BUILDING IT OURSELVES.

Security features are aspects of the software that expose the inner workings of security to the user. They include turning on encryption, saving passwords, choosing how to handle trust failures, etc. When security needs are identified, we should prioritize changing the code in a way that makes the feature secure instead of exposing some security mechanism to the user.

4 MITIGATE RISKS MORE THAN FIX BUGS

It is tempting to hunt for all the security vulnerabilities we can find, mitigate what we identify, and call the software "secure" afterwards. To build secure software we need to consider the risks to the business, the users, the data, etc. We try to minimize those risks by building the software securely. Sometimes all we have to do is fix a bug. But often it is more complicated. Risk management is about figuring out the right way to deal with a risk. Maybe we write some code; maybe we monitor and react to bad behavior; maybe we use legal and contractual controls instead of technical controls. We favor a holistic view of things that could go wrong, instead of distilling "security" down to a giant list of individual bugs that need to be squashed.

Agile teams often embrace the principle of "do the simplest thing that could possibly work." Sometimes, in security, we see this distorted to "write the least amount of code that makes the security report go away." It is harder to misapply this principle if we take a holistic view of risk. In that view, the simplest thing that could work might not involve any code at all, if there is something else (a business process, monitoring, legal contracts, etc.) that addresses the risk to the business.

WE SHOULD FAVOR ADDRESSING THE RISK,
NOT JUST FIXING SOME BUGS.

Applying the Four Principles

The purpose of these principles is to guide and inspire us, but they are not rigid rules. The Agile Manifesto says to favor "Working software over comprehensive documentation;" they are not abandoning documentation entirely. Likewise, the Agile Security Manifesto says we favor risks over bugs; but, that doesn't mean we abandon finding and fixing bugs. These principles give us a preference or a priority given other things being equal.

Building secure software in an agile way is fundamentally just building software in an agile way.