Application Security for Agile Projects

Agile gave us a set of principles that allowed us to build projects in an iterative fashion and respond to change. The benefits of using agile practices over waterfall practices are well known and well documented. However, despite agile projects existing for years, the approach to security has remained the same as it was when waterfall was the way to deliver software. Security and penetration testing is still big bang, often taking place just one week before the release of the project.

There are some fundamental issues with this approach to application security. The most common is leaving penetration testing until right before a release. This leads the developers and product owners to find workarounds for the vulnerabilities in a rush to meet the deadlines, instead of patching them properly throughout development. A hastily added workaround for one vulnerability is a lot more likely to make another component vulnerable. This can lead to multiple unpatched vulnerabilities down the road.

Leaving requirements like security until the end can be detrimental, but making certain decisions too early can also cause problems. Software architecture covers many of the cross-functional requirements of your application like performance, scale or security. These requirements are often discussed and decided upon before you have written a line of code. This is when you know the least. Architectural decisions made at the very beginning of an engagement can lead to security issues, because we don't have all the information we need to make the right decision. By the time security vulnerabilities are uncovered, it might be too late to change the architecture.

Thinking about security at the start is too early and at the end is too late, so what approach should you take? Both these extremes drive out the need to integrate security into each step of the delivery process. This begs the

question: **What would it take to do application security in an agile manner?**

Based on our experiences, here are 5 practices your team can incorporate to move towards a more agile approach to security:

1. Your developers know how to write "secure code"

Most security problems can be avoided if developers know how to write secure code. Every time they write code to accept user input, they should think about validation. Every time they need to store customer data in a database, they should think about the sensitivity of the data and the need for encryption. Knowing about OWASP Top 10 vulnerabilities is a must. There are some great resources online for developers to learn about secure coding practices.

2. Have a "security expert" on the project

Let's agree that no matter how much a developer tries to learn about secure coding practices, there will always be something in the code that only an experienced eye can catch. Hence, it is advisable for each team to have at least one designated security expert who is reasonably experienced in the field of application security. The required experience level can vary based on the complexity, number, and type of "sensitive points" of the system, but the more experienced the security expert, the better. In some cases, a senior person with an eye for security can also play this role.

The responsibility of the security expert include:

- Pairing with other developers when developing security sensitive stories.
- Uncovering and conveying different ways to break the system to the developers. This also means fixing the problems before the code is even checked-in into version control system.
- Taking part in the discussions on choosing libraries, deciding

contracts with external systems, deciding public APIs, etc.
- Helping Quality Analysts to test security sensitive stories.

In addition to these tasks, the security expert should be sharing their knowledge and helping to develop secure coding and testing practices in the team.

3. Determine “security sensitive” stories

We want people on the team to pay special attention to the stories when it can have impact on security of the application. Tagging stories with a tag like “security sensitive” is one way to do so. At the beginning of the each iteration, a security expert should use agile planning meetings to mark such stories which involve security decisions factored into the design as “security sensitive”. Some examples that could be marked as security sensitive are:

- Introducing a new user input field (text, dropdown, etc.)
- Storing user input in a database
- Invoking external processes or system commands
- Processing data received from an external system
- Dealing with an authentication or authorization process

4. Have “abuse cases” for each security sensitive story

Every story has acceptance criteria. For security sensitive stories, Business Analysts (with the help of a security expert) should write acceptance criteria from a security point of view. Similar to Use Cases, Security experts can write “abuse cases” around the functionality. Though abuse-cases are usually tagged with use-case style, you can often find ways to convert them into given-when-then style. This will help Quality Analysts write test cases from a security testing point of view.

5. Carry out security testing before a story is accepted by business

When a security sensitive story moves to “in testing”, Quality Analysts must test the story for abuse cases to check the expected system behavior. A security specialist can help the Quality Analyst with this. This testing should be done in conjunction with the functional testing of the story. If a vulnerability is found during testing, the story must be moved back to “ready for development”. This allows developers to fix the vulnerability and move the story back into to “in testing”. This cycle can repeat as many times as necessary. Once a story is tested for all the functionality and no vulnerability and loss of functionality is reported, the story can be moved to the “accepted” status.

Automation or no automation?

Automation of security testing is a very controversial point of discussion in the field. Some believe there is no replacement for human testing, whereas some believe there should be a full automation suite. Thankfully there is a middle ground. A security tester needs to find the vulnerabilities in the system. But once found, security testers can write automated tests so that they will never need to test for the same vulnerability again in the same part of the system. Tools like OWASP Zed Attack Proxy can be helpful in writing automated tests.

*

Security breaches and attacks on applications are increasing. It is no longer acceptable to build security around your application. We should build it into the application in the same way we do with unit testing. There is unfortunately no silver bullet which can be applied to all projects to make your applications more secure. The whole point of being agile is that it gives us the opportunity to change or modify our approach, so that it is best suited for the system under development.

Check out further resources for your Agile project management, tracking and planning needs.