10 Agile Software Development Security Concerns You Need to Know

Alex Babar

Agile software development is a type of methodology that centers around the core principle of *flexibility*. Agile development methods recognize that a cultural or mindset shift alongside other process changes may need to happen to deliver the best end products in the shortest amount of time. Despite widespread adoption since first being introduced in the early 2000s with the Agile Manifesto, in the last five years, DevOps has taken over as the standard methodology used to build great software products. Today though, DevSecOps is the gold standard in secure application development.

What Is Agile Software Development?

Agile software development is a type of methodology that centers around the core principle of *flexibility*. Creating an agile development environment offers a few key benefits. Modern business operates faster than ever before, so development teams must be able to react to market forces in a timely fashion while still prioritizing the efficient delivery of an application. Agile provides a framework that enables teams to move quickly and still build high-quality products thanks to techniques like ongoing testing *throughout* the development process. Finding more agile development methods allow for a more secure application development lifecycle from code-to-cloud. Successful deployment of security practices in tandem with agile software development allows for secure code development quickly, or DevSecOps.

Get to Know the Top 10 Agile Software Development Security Concerns

Even if you followed every agile development best practice perfectly, there are still risks in agile software development. Agile has proven that it offers more benefits than drawbacks, so it's important to understand the potential security risks so that you can mitigate them ahead of time.

#1 Lack of Process Documentation

Agile prioritizes working products over documentation. Some agile teams take this to the extreme and strive for zero documentation, but most typical, well-functioning agile teams focus on minimizing documentation and ensuring when something is documented, it adds value.

This practice, however, creates certain risks in software development. Agile prioritizes flexibility (which is a good thing), but this can lead to a rapid succession of quick changes that makes it easier to lose track of process documentation.

When you lose track of process documentation, you are not able to ensure that all security protocols are being followed because the reference documentation might not exist, or storing it in a consistent manner was not a priority. To ensure secure application development, it's ideal to have some reference documentation independent of the product. In this case, the improved agility associated with less documentation can increase your susceptibility to security breaches.

To help mitigate security concerns around a lack of documentation, focus on embedding security more heavily into the development process in the form of "security champions" within the development team or by automating some forms of documentation to create a historical record that can be referenced later. One major challenge CISOs and their security teams face today is understanding the application development pipelines they are tasked with securing because no documentation exists within the organization pertaining to which systems are being used by development. Modern security tools now exist, such as Legit Security's SaaS platform that can automatically discover and inventory SDLC assets – automatically creating documentation security teams can access as needed.

#2 Rapid Release Cycle

Agile methodology helps create an environment of rapid releases, but because so much priority gets placed on delivering product value within a sprint, security often takes a backseat to the development of core product functionality during earlier stages of the software development lifecycle (SDLC). Unfortunately, this has proven to be an anti-pattern. By deprioritizing security early in software development, you create security debt that will be addressed later when it will be much more resource-intensive to squash that security bug. In addition, the business has taken on more risk liability during that time if the security bug were to become an exploited vulnerability. Strive to integrate security into agile development methods.

Instead of simply prioritizing pushing out code, focus on ensuring that security vulnerability testing protocols are followed. This can help drive a "security-first" approach to agile and help you avoid potential security risks. In the spirit of "agile security," do your best to automate security checks and programmatically assess whether security protocols have been followed.

#3 Emergence of New Threats

Because agile development focuses on continuous improvement and bite-size chunks of work, releases can create new security risks that would have been avoided by thinking further ahead. Further, agile methodologies are sometimes misused as an excuse to avoid careful planning and preparation. Teams that are myopically focused on only the development of the next new release might not consider how the threat landscape may shift with the new release (i.e., developers are not security experts). As a result of this scenario, attackers can more easily exploit any hidden vulnerabilities in an agile development environment.

One widespread example of this in the software industry today is the use of hard-coded secrets in source code. This practice wasn't a large

concern ten years ago when the concept of cloud software delivered as-a-service was in the earlier stages of market adoption. However, over the past decade, most agile software development shops did not take a step back to assess the impact of SaaS and hard-coded secrets. This created a new attack vector where malicious actors seek to gain access to source code management systems and source code to gain further access to other systems (e.g., build servers) by stealing hard-coded secrets. By myopically focusing on each sprint, the software industry as a whole missed a glaring security issue that has now become a top concern.

When new threats emerge, tackle them in a two-pronged way:

1. Audit your existing attack surface to assess your current risk and prioritize the security fixes
2. Embed methods to iteratively audit your attack surface continually and automatically in a way that aligns with your existing development process to prevent further risk

In order to handle the emergence of new threats with this two-pronged approach, organizations need to implement an internal teams strategy for improved security and continuous education.

#4 Inadequate Security Awareness and Training

It's well-established that security often takes a back seat both on the part of internal teams as well as customers and third-party resources in favor of more directly "revenue generating" activities. In the era of software supply chain attacks this thinking has become dangerously outdated, however. To put it simply, the SolarWinds attack made it clear that a new type of threat exists that will directly result in lost customers because attackers are now seeking to attack your customers *through you*. That's why it's essential to emphasize and facilitate security awareness and training for all parties.

Instead of focusing only on delivery rates and timelines, set aside dedicated time to focus on security awareness education and security

protocol training. Strive to achieve agile security! Use modern software supply chain security tools like Legit Security that provide an aggregated "security score" by product line and development team. This way, you can apply more security training to the development teams that need it most without also burdening already secure development teams. With the gamification of security practices through security scores by product line, organizations can move into the hardening phase in agile security.

#5 Unlimited Access to Source Code Repositories & CI/CD Pipelines

It's not uncommon to simply give blanket permissions and authorizations to large swaths of the development team, especially in smaller organizations with less mature security practices. More users with access to code repositories, source code management systems (SCMs), build servers, artifact registries, and CI/CD pipelines equates to more risk to your SDLC. Each over-privileged user represents another weak point in your defenses that becomes a target for attackers. In addition, software development teams often outsource some portion of development to external, contracted development teams on a project basis. Often, these contractors maintain their permissions well past the conclusion of the project. The risks to agile software development are especially critical when administrative-level privileges are provided too freely.

Boost your agile security by following agile development best practices. Examine the permissions and authorization protocols for your development team and the systems they use to rapidly develop software. Use the principle of least privilege to ensure that each user has *only* the permissions needed to do their job.

If the number of developers with access to your development pipelines ranges from 0-50 in headcount, a manual approach might be possible and desired; however, for teams exceeding 50 developers, it's best to use a tool like the Legit Security platform to help you identify the current access risks present in your development pipelines before an attacker

compromises stale accounts you incorrectly assumed no longer had access to your pipelines.

#6 Failure to Manage Sensitive Data

Many development teams understand basic security principles, but most are not well-versed in how to layer security practices around data management in a way that aligns with agile development best practices.

The full list of security risks associated with managing sensitive data is long and beyond the scope of this blog, but some of the biggest challenges in DataOps are (1) adopting proactive data governance, (2) labeling sensitive data, and (3) educating teams on acceptable data practices.

Here are some tips for better sensitive data management:

- Centralize identity management
- Define role-based entitlements
- Mask sensitive data
- Move away from manual data security and *automate* whenever possible
- Continually scan code for sensitive data

As security and development teams become more aligned, it is advantageous to have a 'security champion' or someone on the development team that advocates and pushes for better overall security posture and sensitive data management. Embracing security as a part of development helps all teams overcome security hurdles.

#7 Poor Third-Party and Open-Source Management

Third-party and open-source code can play an essential role for most development teams by freeing up a lot of resources from needing to create every aspect of code in-house. However, that doesn't mean third-party and open-source code is maintenance-free. All this external code is

combined with in-house code to create a portion of your software supply chain that is highly susceptible to attack.

Easy ways to manage open-source and third-party software in your supply chain include:

- Establishing a policy on open-source use
- Establishing an open-source governance board
- Creating intertwined relationships with your suppliers so that if problems do arise, they are responsive and quick to take the necessary actions to remediate issues
- Regularly run Software Composition Analysis (SCA) scans
- Inventory your SDLC to understand which pipelines are most business-critical
- Scan your development pipelines for usage of SCA scans

It's important to call out that your software supply chain *is not just* the external code utilized her. Your software supply chain also consists of the developer tools and pipelines in your SDLC and any additional code, libraries, and resources utilized in the deployment of your software or tool pushed to your production environment.

#8 Not Using Third-Party Code to Your Advantage

Many teams place a high priority on developing internal security solutions, but there are several downsides to developing proprietary code and solutions. Due to this, sometimes a new, untried, untested, unsecured internal solution is used *out of habit* instead of simply evaluating third-party solutions that solve the problem in a more out-of-the-box, less resource-intensive way when compared to a DIY approach.

One hybrid approach is to invest in low-code and no-code security and automation solutions to integrate security into agile development methods. These types of tools help alleviate bottlenecks due to development team capacity constraints. Agile product owners don't always *truly* understand the security implications of their top features, so

it is difficult to plan ahead when drafting requirements. And not all teams are able to formulate requirements that address security issues without then dictating too many elements of the solution, which can lead to massive scope creep when attempting to build internal solutions.

In short, do not be afraid to use reputable and secure third-party code and solutions to your advantage because these solutions are well-tested and significantly decrease the time to deployment.

#9 Failing to Prioritize the Security Team

A security-first approach can seem counterintuitive to the pace of typical agile development for many agile teams. Security testing and procedures are often seen as roadblocks and speedbumps to delivering the finished product more quickly by agile team members focused on meeting the latest sprint deadline.

One way to boost the security in agile development is to drive a cultural change that makes secure development a non-negotiable priority. Spend time re-framing security procedures so that team members understand why a certain security practice is important. Integrate security and development teams so that security can be integrated earlier in the process by a subject matter expert. Discuss and establish reasonable testing frequencies and procedures to help shift towards an “agile security” culture. There is a strong relationship between adding too many unreasonable security checks in a process and development times inventing ways to bypass arduous processes. For example, it might be more secure to require 2+ code reviewers when pushing code, but it creates more than 2x the friction of requiring just one code reviewer. Security should be mindful of the impact on the developer experience when instituting policy that needs to be followed.

#10 Opting for DIY Security

It’s not uncommon for many teams to try and handle all their security needs in-house. There’s a common misconception that it can save you

time and money to manage security entirely with an internal team (especially given that companies are desperate for cybersecurity expertise.) Often, internal teams lack the comprehensive expertise required to keep *the entire* SDLC secure – that's where experts come in.

Outside expertise comes in two forms:

1. Cybersecurity service providers
2. Cybersecurity product vendors

Using external security expertise to manage your security provides some notable benefits:

- Frees up internal resources to focus more directly on the tasks that only they can do
- Fills specialization gaps that would be too difficult to hire an internal resource to provide

Now that more and more of development is being codified, the software industry has seen the emergence of "policy as code". SaaS solutions like the Legit Security Platform now exist that have codified hundreds of best practices in highly specialized security domains like software supply chain security. Instead of requiring an internal subject matter expert, you can leverage the expertise codified in a product. Additionally, SaaS is better suited to *governing* whether development teams are violating required policies or not when compared to an internal security resource. Ultimately, this helps achieve more security-driven development.

Let the Experts Help Protect Your Agile Development Process with Software Supply Chain Security

The agile software development methodology is one of the most tried and true development methods, but it doesn't come without its own set of safety and security risks in software development, like focusing so much on rapid releases that security takes a back seat. The good news is that

today's software builders and regulations are acknowledging the importance of secure software, and several frameworks are emerging to help fill the gap in subject matter expertise on your internal teams.

Legit secures your applications from code-to-cloud with automated SDLC discovery and analysis capabilities and a unified application security control plane that provides visibility, security, and governance over rapidly changing application development environments.

To learn more about how the Legit Security Platform was built to help application security teams keep up with fast-moving development teams at scale, schedule a product demo or learn more about our platform.