Case Study

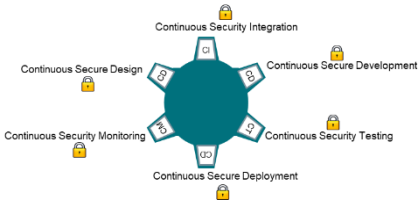
Public Sector
Embracing Digital Transformation

Securing the DevOps Pipeline

Recent cyber attacks have focused on the build process vulnerabilities. Find out how to secure the DevOps pipeline to protect from these attacks.

Sidebar header

In this episode Steve Orrin and Darren Pulsipher discuss why and how the DevOps pipeline must be secured. The only way to deliver solid, resilient, and secure code is if security is built in, and the earlier the better.



Video: Youtube Channel

Podcast: SoundCloud

A study from over 20 years ago on the return of security investment showed the earlier security is built into the development cycle, the cheaper it is than waiting until the end of the cycle. Although we have known this for two decades, it's still a work in progress.

Most development companies have security as part of their overall development process, so there has been a lot of headway, but it's a journey, not a sprint. It's about understanding all of the different exposure points and weaknesses and being able to provide the right security processes to those problems.

The Attack Vectors

Oftentimes people think of attacks as vulnerabilities of employees or packages, for example, and forget the process part of the story. On the operation side is the time it takes to close a vulnerability. On the other side is in the development and delivery of products. There are various break points along that chain, and those have been exploited recently in the latest stages of the build process. As far as customers were concerned, the code that was updated was legitimate because it came right from the source. So we need to think carefully about where to put security into the process.

Agile, CI/CD, DevOps,…Where is Security?

While we tend to look at needing security at the transitions from developer to QA, and from QA to operations or deployment, security should really be injected into the whole build cycle, not just at a few checkpoints. The approach should be continuous security.

Make Security Part of Every Phase

Continuous security is challenging. Most developers and QA aren't security trained; this is an uphill battle. The industry tried this approach in the early 2000s,but ran into three problems. First, the turnover is too high. Second, the security landscape changes too rapidly to keep them up to date, and third, since it's not their day job, the right behavior was not incentivized.

How do we, then, integrate security into the process, automate the key things we want to do, and get out of the way of developers so they can do their jobs, which is to build, test, and deploy the code? The security process can really shine by building it into those automations that you are already doing in DevOps such as automated unit testing, automated quality and regression testing, automated build, and automated deployment. This will not solve all the problems, but it will raise the bar significantly so you can focus on the hard challenges around security

Means Security is Baked in, not a Single Step or Stage

Some common tools already provide some automated security that points out vulnerabilities. For example, GitHub will run security checks on projects using Node.js code and all the included packages. This can be helpful, but it is too late; the security should be built into the pipeline before it gets checked in.

How Do We Get There?

Current security breaches highlight that security must be injected at every stage of the process, including between build and production, and right before the script runs to build the application. In addition to injecting security into the build process, we need to secure the build process itself; it's been a gaping hole for a long time.

Many companies that do internal development are now taking a closer look at their build process because of the recent breaches. This is good, but it can't stop with these knee-jerk reactions to each attack. We need to think holistically and not wait for the next weak link in the chain.

Some practical ways to secure the process are to treat the build server as a critical asset in the overall infrastructure and apply the same rules and controls to that server as you would for your core systems. Proper credentials, firmware secure boot, verifying code, auditing and logging the system, etc. throughout its life is then building into the DevOps process when someone clicks the button.

From Solutions to Services and Beyond

Many people don't think about the script itself as a target. It doesn't matter how many good modules are included if the script itself isn't protected. A few ways to protect the script is to run a checksum, and it should be versioned, checked and signed. This adds complexity for DevOps, but there are tools that can help.

Build Once, Deploy Everywhere

Just like we automate the development process, we can build in the automation for implementing these controls and checks. Automation prevents another person from potentially messing with your builds, but we do also want to make sure there's a human receiving results and verifying audits.

The tools you are already using can be extended to add security automation and checks such as those to do continuous development integration for the Agile cycle, or automation tools in the Linux world.

Organizations can also distribute their security people throughout the business development teams so when things go wrong, security people are already embedded in the process. Two places you want to make sure you have security people are in infrastructure to support, for example, your Agile process, and in product management to get security requirements for the product requirement definition phase before it even gets to a developer.

There is always a shortage of enough trained and capable security people and also funding to hire the right people because of high demand. A few options are to train the people you already have and give them the necessary tools. You do not need a crypto guru at every step of the process. Another possibility is instead of having each coder be responsible for coding authentication, credentials, and protocol in a secure build in an infrastructure library, have a team build modules that are in your languages and your environments that do all the security functions. The coder can pull the module, and it handles the hard work. That way, you build once and deploy everywhere.

We are seeing companies provide SaaS security tools, cloud-based services that can be consumed for your application and your runtime environment. This is a great step in the process. There are companies that provide security injection points such as application security in a fast style environment. These application checks such as input sanitation and input validation can be embedded into your functionalist environment, but that's still waiting to the end. Remember that the earlier in the process you start security, the cheaper and less painful it becomes.

All of this does, of course, require more integration work. Developers can be wary of the work involved, but if a framework with built-in security exists (and there are prototypes out there such as Ruby on Rails and certain cloud infrastructures), it can save many hours. You still have to make sure, however, that you don't rely only on the platform for security, as it could be a single point of failure.

Automation Sill Set You Free

The security breaches in the last six months have been profound. Here are some key points of advice:

Security should be integral in the whole lifecycle from requirements forward. Security must be in the DevOps cycle itself, not just in coding and testing, but also in the infrastructure that drives that process.

When building security tools and objects through modules, build once, make it modular, and deploy everywhere.

Leverage services that let you rely on someone else's expertise to augment your own, underfunded, cyber team.

Automation will set you free. Automate as much as you can to make security easier and faster and reduce friction for your developers and testers. With automation, you can eliminate 80 percent of what we call the stupid stuff so you can spend your limited resources on the hard problems.

Intel® technologies may require enabled hardware, software, or service activation.
No product or component can be absolutely secure.
Your costs and results may vary.
Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.