DevSecOps

Learn how ZolonTech bridges the gap between DevOps and Security

Forward

Today we live in a culture where we must have everything instantly from social media, news, instant messaging, and the web. Customers, whether they are commercial customers or Federal IT customers, all live in the same era of astonishing technology, instant information and rampant social networking. IT organizations have been traditionally set up to deal with a long tedious Software Development Lifecycle (SDLC) that has customers waiting for releases every quarter, bi-annually or annual updates to software that doesn't meet the end-user needs, and still has defects requiring additional updates.

To provide a secure rapid delivery method to production and disconnect between development and operation teams, Zolon Tech Inc. (ZTI) has incorporated a secure manifesto to already industry leading software development and delivery process tool in software development and operations (DevOps). Secure DevOps or DevSecOps addresses the collaboration, and automation between software development and operations team securely. Zolon Tech's Secure DevOps approach addresses agile development, continuous integration, continuous testing, continuous delivery, and continuous security through the use of automated tools, and streamlined processes. Our goal of safely distributing security decisions at speed and scale allows us to deliver incremental development continuously to production, which reduces defects, eliminates excess cycle time, provides continuous feedback and eliminates outage windows when deploying to production without sacrificing the safety required.

Implementing effective security measures within the DevOps process has found to be difficult due to the manner at which secure software is currently being developed. Often, having such quickly developing software is prone to have equally rapidly arising security issues, and the process of creating completely secure development codes that abide by most to all of the regulations is difficult, and is seen as potential roadblocks to the efficiency in terms of development and the delivery cycle. The current security methodologies are designed to cater to previous, traditional waterfall systems; this cultural discontinuity between the roles of security, development, and production, is one of the key issues within the current testing models. This whitepaper discusses how ZTI intends to solve this problem by developing a model that reduces the risk exposure of the software products from the DevOps system such that fewer vulnerabilities are coded, and any vulnerabilities are found closer to the introduction time, and managed as rapidly as possible before market deployment.

Contents

Security Challenges in DevOps

With the evolution of DevOps, automation of developing, testing and deploying the code has become a common norm in many modern organizations today. DevOps promises faster time to market, shortened release cycle and low rate of release failure. It is the best interest of any organization to develop an application that is secure, but securing the code early in the software development lifecycle has been a challenge ever since mankind started software development. Many organizations do have existing security measures in place, but most of their strategies cannot cope up with the rapid pace of automation in DevOps.

This lag between security and DevOps counters the basic idea of DevOps by slowing down the process. Traditional security measures involve threat modeling by analyzing security architecture and security requirements compilation, which are manual tasks that are time-consuming and result in significant delays in the process. And if separate security tools are implemented to automate this process, it leads to repetition with increase in frequency and consistency without compromising the product.

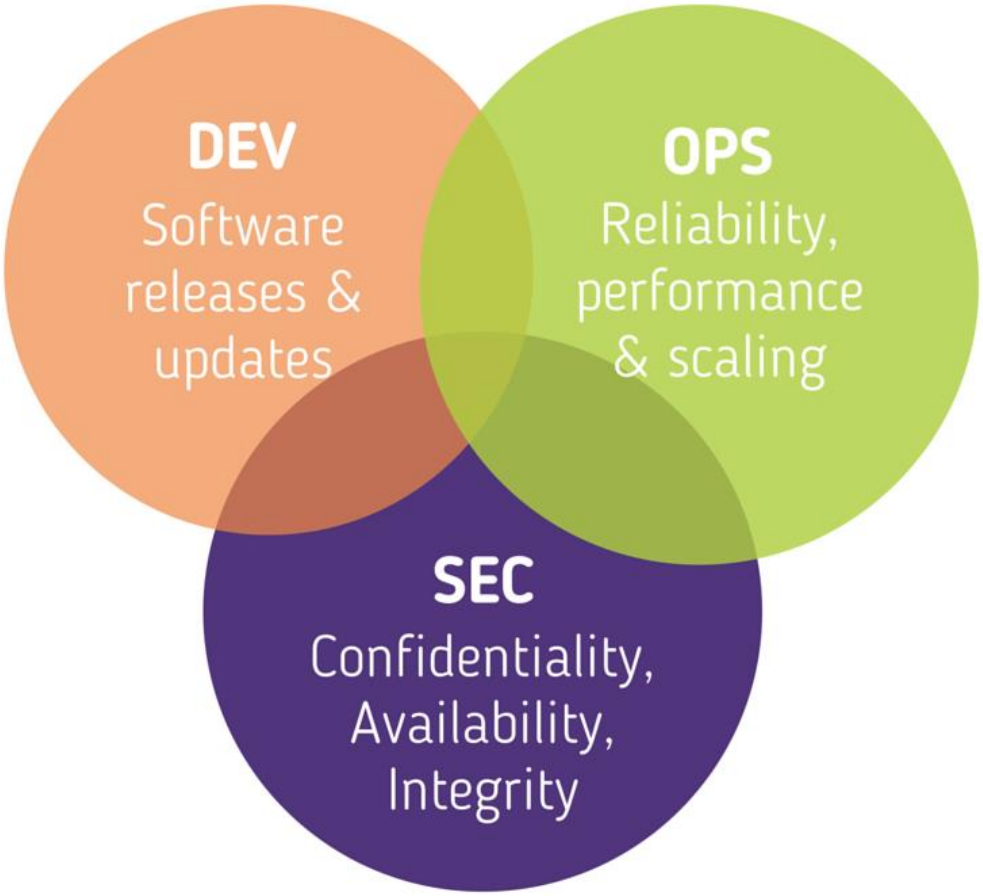
By having a separate security system, the security team is challenged with lack of communication with the security requirements from the development team. And mixing security requirements with developer requirements can slow down the process if the developers don't know how to handle them, or are not familiar with security principles. This process undermines a critical step in DevOps of efficient development and implementation.

To test application security, functional and non-functional, tools such as IBM AppScan and HP Fortify are developed to secure the development process, and they can easily be automated. Large number of error findings cannot be automated by implementing such scanning tools in a DevOps environment. The errors found at this stage of the SDLC are costly and time-consuming as each error must be addressed and fixed by the development team and the underlying architecture needs to be re-evaluated. This can be avoided by addressing earlier in the SDLC via security requirements management, which would result in far fewer findings at the testing stage and spending less time on remediation, for a faster release.

Therefore, the only best strategy to counter this disadvantage is by integrating security into the DevOps pipeline to provide a seamless automation between Security, Development and Operations. A recent finding from the 2016 State of DevOps Report shows that IT teams that engage in Secure DevOps spends 50% less time remediating security issues, because they're constantly finding ways to improve the code from the beginning of the development lifecycle.

DevSecOps Approach

In simple terms DevSecOps aims to ensure a better quality software by incorporating security principles into the software development process from the beginning of the software development lifecycle. Protecting your information has never been so critical with information and data being dubbed as the new "Gold". Anyone with intent to steal are finding new and innovative ways to find flaws within the system and one wrong code could lead an attacker to backdoors that could lead an organization to crisis. At ZTI, we are no strangers to the dangers that a bad code could possess, we are a company that strive on providing innovative and secure solutions to our federal and commercial customers. Being an IT company our solution scope is not limited to the IT industry; our Centre of Excellence - Solution Engineering (COE –SE) team are always on the constant lookout for new and emerging technologies along with exploring proven techniques and methodologies from a wide array of industries. Through this we have taken inspiration from a very popular proverb in the medical industry: "An ounce of prevention is worth a pound of cure". At ZTI, we protect our customers by providing a Secure DevOps approach that guarantees a secure and robust code, which would prevent all imminent dangers.



Our DevSecOps approach encompasses the entire SDLC. Our approach (see Figure 1) includes agile development, continuous integration, continuous testing, continuous delivery and three specific checkpoints for security implementations while providing continuous monitoring and feedback throughout every phase of the lifecycle. Automation is a critical component of a successful DevSecOps approach, and the tools in this space continue to rapidly advance. Our approach is agnostic to a particular toolset, and is customizable based on customer preference. The key steps for automation that enable our DevSecOps approach include:

1. **Daily Code Commit:** Developers on a daily basis check-in code into a central source code repository.

2. **Automated Asynchronous testing:** Initial Security check to monitor bugs and mistypes within the code before being integrated to the server.

3. **Automated Builds:** A Continuous Integration (CI) server is continually polling the source repository for changes, and when a change occurs the code is checked out of the repository and built. The built software is stored in a repository manager by the CI server.

4. **Synchronous Security Testing:** Active, rapid checks for the strength of the code utilizing Software Penetration Testing.

5. **Automated Testing:** The code is automatically unit tested, code quality tested, smoke and user interface (UI) tested, and performance tested.

6. **Asynchronous Security Implementation:** Automatic Testing before the code is deployed to the market. Ensuring that the initial security check was optimally effective.

7. **Automated Delivery:** The built version is deployed using provisioning tools that treat infrastructure as code.

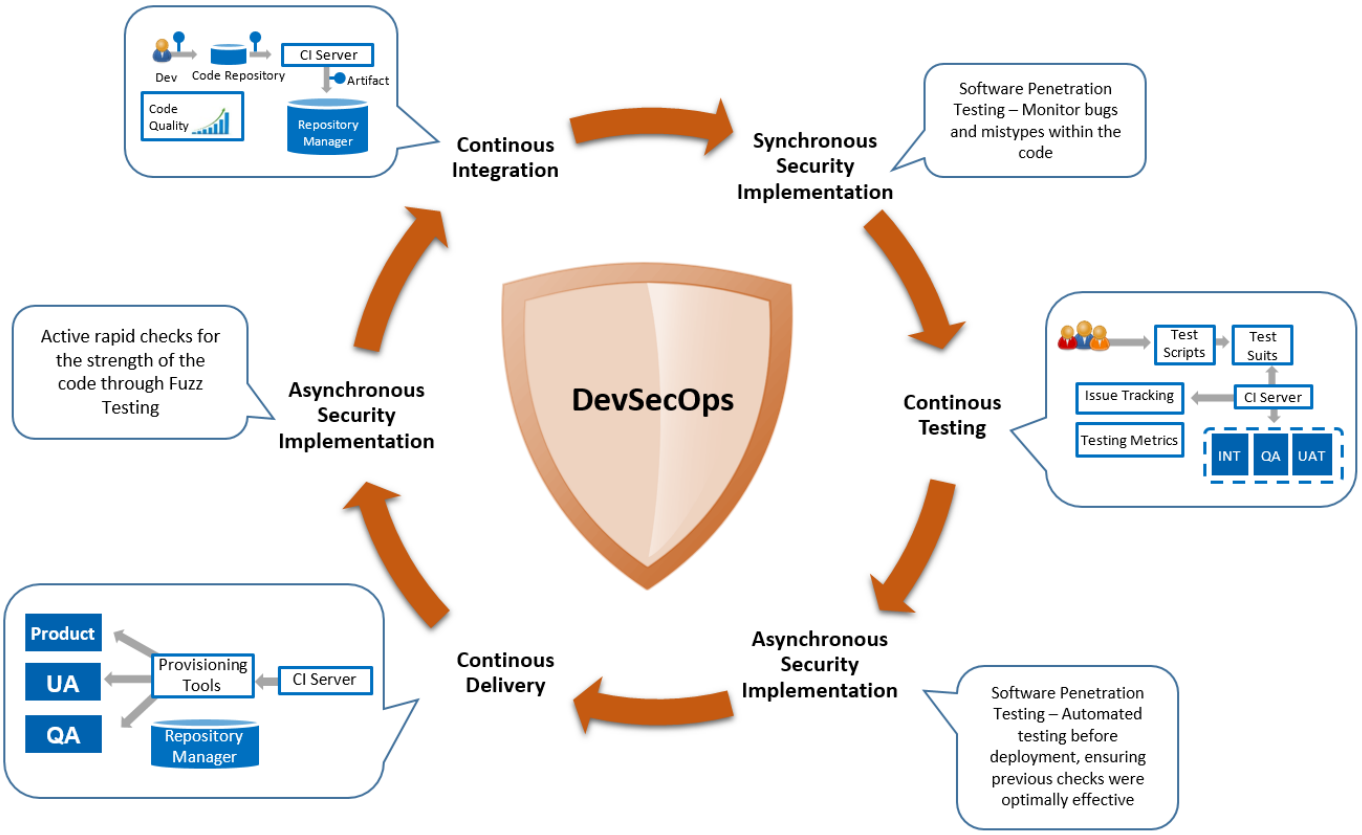


Figure 1 *Zolon Tech's DevSecOps Approach*

The entire pipeline, described in the next sections, is tailorable to add in manual validation at any phase. We increase efficiencies by automating the promotion of software code from development to testing and into production. Our DevSecOps approach forms collaborative teams, and reduces the number of issues occurring during development, deployment, and operations, resulting in faster delivery of applications/features, securely by reducing Operations & Maintenance (O&M) costs.

Continuous Integration (CI), and Continuous Delivery (CD)

Our Continuous Integration (CI) and Continuous Delivery (CD) approach is designed to create an automation environment for the entire end-to-end release process so that every change to the application results in a releasable version that is built automatically. With our approach applications are built from a very early stage in the development process at specific frequent intervals or on every change checked in by the developers. This effectively eliminates the need for integration testing because the code is incrementally being integrated on a daily basis. This removes the cost associated with developers spending time on this phase. The enablement of frequent incremental builds and mandating a comprehensive automated testing process also allows developers to detect problems early and as a result, ensure higher application quality.

To support rapid deployment and release we use tools that allow us to automate provisioning of infrastructure resources and platforms. The server configuration, packages installed, relationships with other servers are modeled with code, and is automated and has predictable outcomes, removing error-prone manual steps. Our approach uses software development best practices for the infrastructure code and stores the code in a Code Repository with tags and branches, and releases the code just as if it were applications software. Our infrastructure code is continuously integrated, tested and deployed right along the application software and is treated no differently.

We configure our CI server with build steps to check for coding style, coding standards, static analysis, and other features using tools such as SonarQube. We run unit tests from the CI server and deploy the code to the development integration environment and execute additional functional test scripts. We use tools like Selenium for smoke and UI testing. Performance testing is part of our automated testing using tools like JMeter for load testing. Using CI server, we produce artifacts that document the results of the build, unit testing, and deployment and functional testing along with the source version used for the build.

Our automated deployment provides a continuous delivery pipeline that automates deployments to test and production environments. Our approach significantly reduces the manual intensive tasks, resource lag time and errors prone from manual repetition. We provide automated deployment tools and processes reducing deployment risk, and giving the option of deploying code multiple times per day without any degradation in service. The releases are small which reduces the risk for system instabilities and customer user experience issues. Every change is easier to roll back and easier to test because the number of changes per release is very small.

Continuous Monitoring and Continuous Feedback

Continuous monitoring across all phases of the application development, testing and deployment is crucial for a successful DevOps implementation. Our approach lowers the costs of errors and changes by providing continuous feedback throughout each phase of the lifecycle. We offer a unique approach to monitoring using multiple tools to monitor the applications, environments, and systems. Using tools like Evolven for environment management provides real-time user tracking and work flows to manage the technical users that develop and operate the application on a daily basis. We use tools like Splunk for log analysis for developers and tools like New Relic to monitor the performance of the applications from the user's perspective such as page-load times, database-transactions, and systems monitoring to focus on CPU load, memory utilization, and disk space. These tools allow our teams to better understand issues and metrics, and ensures that we are optimizing resources to reduce operational expenditures.

Continuous Security

ZTI's unique Secure DevOps approach provides the means to develop a model that reduces the risk exposure of the software products from the DevOps system with fewer vulnerabilities in the code and rapid mitigation of any anomalies detected closer to the introduction time before market deployment. ZTI does this by addressing the aforementioned cultural discontinuity by allowing a complete integration of security practices within the model, and utilizing the in-built capabilities of DevOps (such as continuous development, continuous monitoring, and continuous testing) to the advantage of security developers instead of having it traditionally work against it. This model uses a philosophy of creating a system that "bends" but does not break after potential breaches, differing from the current security methodology that works towards creating preventative features. By continuously editing the code to effectively amalgamate the changes within the software with each oncoming update.

Adding automation as a component regarding security testing amongst key points within the widely popular pre-existing Zolon DevOps pipeline allows for this approach. We have added three Security Implementations to optimize the efficiency of security testing. First, it is important to define two abilities being used within these checkpoints: fuzz testing and software penetration testing. Fuzz testing is a technique where the code is tested through the implementation of large amounts of random data (hence it is dubbed, "fuzz") to determine the strength of the code while also determining any coding errors or security loopholes within the code. Software penetration testing also works towards determining weaknesses by utilizing reconnaissance and having members of the security team consciously find methods to infiltrate the software such that potential weaknesses can be altered within the code to avoid these issues later.

Writing Secure Code in Continuous Delivery:

At ZTI we take advantage of DevOps techniques in our security program, by implementing three specific security checkpoints as seen in figure 1. By implementing an automated asynchronous testing environment preceding the Continuous integration stage allows for an initial security check for minor bugs or mistypes within the coding before it is integrated to the server. By conducting these tests we are able to output good code that is bug-free and functions well when deployed. This also eliminates the risk of insider threats by increasing developer accountability. Depending on your application's complexity and requirements we use tools such as Gerrit, Phabricator or Atlassian Crucible to provide a platform that increases transparency to changes and ensure that a change can't be pushed out without at least one other person being aware of what was done and why it was done. This knowledge of reviewing one's work encourages developers to be more vigilant and careful in their work and anyone trying to introduce a logic bomb or a back door can be prevented.

Humans are always prone to errors and no one is correct every time, realizing this vulnerability ZTI has implemented another way to improves code security. We do this is by scanning the code for security vulnerabilities using automated static analysis software testing (SAST) tools. These tools can find subtle mistakes that reviewers will sometimes miss, and that might be hard to find through other kinds of testing. Developers take advantage of built-in checkers in their IDE, using plug-ins like FindBugs or Find Security Bugs, or commercial plug-ins from Coverity, Klocwork, HPE Fortify, Checkmarx, or Cigital's SecureAssist to catch security problems and common coding mistakes as they write code.

Secure Testing in Continuous Delivery:

To verify security as soon as changes are made, we moved security testing directly into continuous integration and continuous delivery. We added synchronous security implementation between the Continuous Integration stage and the Continuous Testing Stage which allows for there to be active, rapid checks of the strength of the code utilizing Software Penetration Testing to do this. This means wiring of application scanning and fuzzing integrated into the Continuous Delivery pipeline. Through this we are able to take advantage of work that the development team has already done to create an automated test suite, adding security checks into unit testing, and automating security attacks as part of integration and functional testing.

ZTI uses a technique that is important to optimize the efficiency of security testing by fuzzing. Fuzz testing is a technique where the code is tested through the implementation of large amounts of random data (hence it is dubbed, "fuzz") to determine the strength of the code while also determining any coding errors or security loopholes within the code. Fuzz testing tools such as WebScarab and WSFuzeerare from OWASP provide a framework for analyzing applications that communicate using the HTTP and HTTPS protocols and real-world manual SOAP pen testing. JBroFuzz is used for web applications.

And finally we use Gauntlt (a popular open source test framework written in Ruby) to describe simple security-related asserts or complex automated attack scenarios in a way that is easy for developers and auditors to follow. Gauntlt wraps pen-testing and security-testing tools, abstracting the details of how they work and making them more accessible, and controlling them so that you we create repeatable, deterministic steps with clear pass/fail results

Next Steps

Understanding DevSecOps and its benefits is just the beginning. ZTI's DevSecOps model will minimize the cultural discontinuity between the traditional security testing regulation model and the DevOps methodology by utilizing several synchronous and asynchronous security implementations. ZTI can help facilitate the adoption of DevSecOps in existing and new programs. We offer expertise to help clients align business process and technology to their business needs. We help clients develop a strategic roadmap for how DevSecOps technology may be used to meet mission and business requirements. Our approach helps clients understand the goals and IT strategies within the roadmap, and we execute projects to implement these DevOps strategies. Our DevOps experts assist clients in identifying the drivers, key considerations, and other important factors for migrating and modernizing IT Services. Our experts develop comprehensive IT Transformation plans that provide clear steps to increase the value of technology.

Conclusion

Implementing effective security measures within the DevOps process has found to be increasingly difficult due to the manner at which secure software is currently being developed. Often, having such quickly developing software is prone to have equally rapidly arising security issues, and the process of creating completely secure development codes that abide by most to all of the regulations is difficult, and is seen as potential roadblocks to the efficiency in terms of development and the delivery cycle.

Our DevSecOps approach addresses the entire SDLC. With the implementation of automation, security and repeatable processes, we securely deliver continuous integration to production systems without outage windows, and unnecessary manual processes. Our approach reduces costs by providing environments that are automated thus removing the need for staff to spend time with manual processes. We make processes repeatable which allows for more frequent and less error-prone releases. Our vendor-agnostic approach drives increased efficiencies through improved development and operational processes, transparency via continuous monitoring and feedback, improved quality from continuous integration and testing, and less risk due to an automated environment and a combination of asynchronous and synchronous threat detection and mitigation in the code. We have the vision, experience, and technical knowhow to assist Federal agencies to take full advantage of DevSecOps.

13921 Park Center Road, Suite 500

Herndon, VA 20171

www.zolontech.com