SHIFTING SECURITY LEFT

DevSecOps eGuide

TECHWELL™

Prioritizing security in our software is more important now than ever. DevSecOps is all about incorporating security practices into our DevOps process from the start, moving these activities "left" in the development lifecycle so they don't delay delivering value later on. This eGuide rounds up a collection of resources to help you build security into your existing deployment pipeline, so you can reduce risk, expose vulnerabilities early, and prioritize security in your software from day one.

In this DevSecOps eGuide

DevSecOps: Incorporate Security into DevOps to Reduce Software Risk

DevSecOps is a growing movement to incorporate security into DevOps practices in order to ensure flaws and weaknesses are exposed early on through monitoring, assessment, and analysis, so remediation can be implemented far earlier than traditional efforts. By failing fast with security testing, organizations reduce risk of a security incident and decrease the cost of rework.

4 Keys to Protecting Your Data in a DevOps World

It may seem like the desires for end-to-end DevOps and protection of sensitive data are in conflict, but if done correctly, they can be two sides of the same coin. DevOps processes such as version control and delivery automation introduce the very measures needed to properly protect production data. The key to keeping data safe while using it during your DevOps process is to focus on these four areas.

Using the Principles of the CIA Triad to Implement Software Security

If you're starting or improving a security program for your software, you probably have questions about the requirements that define security. Data need to be complete and trustworthy, and also accessible on demand, but only to the right people. The CIA triad defines three principles—confidentiality, integrity, and availability—that help you focus on the right security priorities.

DevOps and Security: 5 Principles for DevSecOps

With the trend toward a more continuous delivery and deployment process, late-lifecycle activities like security assurance present a significant hurdle to continuously delivering value to customers. DevSecOps addresses this by shifting security assurance activities, personnel, and automation closer to development.

A Definition of Done for DevSecOps

In DevOps, we have a software delivery pipeline that checks, deploys, and tests every build. The goal is to produce a viable candidate for production, so we have to look at many different aspects of quality, including security. To be sure we hit all the crucial marks, we should have a definition of done for DevSecOps.

Managing Security Testing in Agile Software Development

One of the biggest myths in the world of agile development is that there is not enough time to do security testing. Sanjay Zalavadia shows you the most efficient and cost-effective way of performing security testing in an agile environment: by rolling it into each sprint incrementally, from day one.

Insight from the Industry

Additional Resources

TECHWELL™

TECHWELL™

DevSecOps: Incorporate Security into DevOps to Reduce Software Risk

By Alan Crouch

By now, most organizations have heard of DevOps, and many have begun to adopt DevOps practices as a key enabler of software delivery. Organizations that employ an agile approach find DevOps practices a natural extension, and DevOps truly enables agile practices to flourish.

Organizations typically start with implementing continuous integration, test-driven development, and test automation early on. Agile delivery teams embrace small, iterative development and increased code quality, and with these practices in place, continuous delivery and continuous monitoring practices start taking hold. Teams focus on faster delivery with less human interaction, and successful organizations build more collaboration between development and operations teams as they start working toward a shared goal.

While all these practices provide strategic benefits like breaking down the traditional silos between development, test, and operations, it is unfortunate that many organizations stop there. In my experience, most organizations fail to integrate their security programs into their development efforts, resulting in lengthy security compliance activities and lots of vulnerabilities identified late in the delivery lifecycle. Does that sound like DevOps to you?

These issues are why the concept of DevSecOps is making such a large impact in the security community. DevSecOps is a growing movement to incorporate security into DevOps practices in order to ensure flaws and weaknesses are exposed early on through monitoring, assessment, and analysis, so remediation can be implemented far earlier than traditional efforts. By failing fast with security testing, organizations reduce risk of a security incident and decrease the cost of rework.

Exploring a DevSecOps Workflow

Here's an example of DevSecOps in action:

1. Developers create the code and tests, which are managed by a version control system like Git.
2. Changes are committed to the Git.
3. Jenkins pulls the code from the repository and builds and runs unit tests, as well as static code analysis to identify code quality bugs and security defects.
4. An infrastructure-as-code tool, like Chef, provisions an environment, deploys the application, and applies security configurations to the system.
5. Jenkins runs a test automation suite against the newly deployed application, including UI, back-end, integration, API, and security tests.
6. If the application successfully passes all tests, the application is deployed to production using the same infrastructure-as-code tool used in the previous environments.
7. The production environment is continuously monitored by tools like New Relic and Splunk to detect active cyber security threats.

DevSecOps provides a number of benefits among development, security, and operations. It eliminates silos, promotes collaboration and teamwork, and identifies vulnerabilities early while still providing better, faster delivery. DevSecOps also contributes business value through dollars and resources saved, improved operations, diminished security threats, reduction of rework, and increased quality through automated testing, as well as the delivery of projects and products early and often, with less cycle time to the customer.

In short, we can spend more time adding customer value to our software and less time and money fixing costly security vulnerabilities identified late in the delivery process or in production.

DevSecOps Means More Than Automation

When people think of DevSecOps, the first thing that comes to mind is automation. A strong DevSecOps environment should employ tools that automate continuous integration, delivery, testing, deployment, and monitoring.

While automation is certainly important, it's just as important (if not more) to build the mindset that everyone is responsible for security, with the goal of safely distributing security decisions to those with the highest level of context without sacrificing the quality, privacy, and safety required by the system. This type of model typically relies on "shifting security left," or engaging security earlier in the software development and operations processes.

Changing an organization's mindset and culture doesn't happen overnight, so don't expect integrating security into DevOps to be any different. It requires hard work, training, coaching across the entire organization, and patience.

While there is no one right way to change organizational culture, a few critical components are necessary for DevSecOps to take hold:

- Building a knowledge base: Raising a developer's security knowledge plays enormous dividends. Ongoing training and learning activities ensure not only that developers know how to be responsible for security, but also that they continue to stay on top of cyber security best practices.
- Promoting openness: Openness in communication encourages collaboration and continuous improvement between development and security. Transparency of information using metrics and dashboards can be an effective mechanism for communicating what's really going on in a quantitative way. By building trust and cooperation through openness, organizations can ensure security does not become reactionary.
- Creating cyber security champions: The lack of highly qualified security professionals can make the transition from DevOps to DevSecOps difficult. Savvy organizations identify individuals who understand security within traditional dev and ops groups and trust these individuals to coach DevSecOps teams and act as the security conscience of the team during the transition.

Implementing the DevSecOps Process

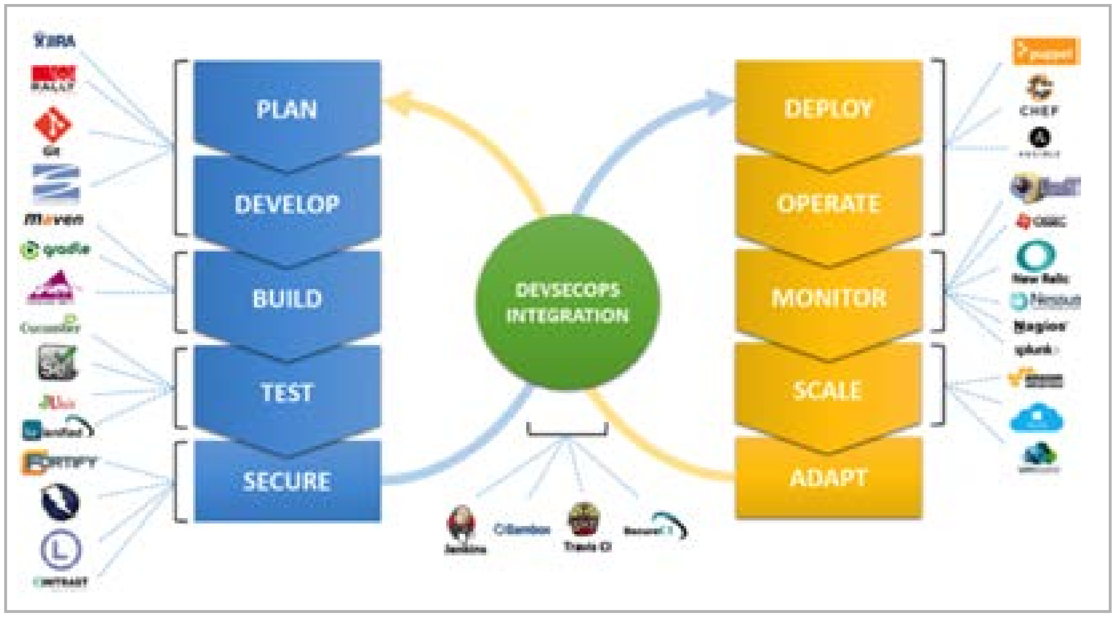
The primary goal of DevSecOps is to ensure security and operations team members are engaged and collaborating with development from the very beginning of a project or product development. In addition to cultural shifts, it demands a toolchain of technologies to facilitate collaborative change. It requires pushing past departmental lines for more effective planning, design, and release of secure products.

As organizations continue to build upon automated delivery, they find there are opportunities to test for issues beyond typical bugs:

potential security flaws, design defects, and code weaknesses. Imagine being able to identify and fix flaws earlier in delivery process, before they are exposed to the public.

Implementing DevSecOps in an organization requires building a single group of engineers (developers, admins, testers, and security engineers) that has end-to-end responsibility of the application, from requirements to deployment to monitoring and back to implementing new changes. This process forms a set of stages that can be carried out in a continuous loop until the desired product is achieved.

The diagram below shows the steps in a DevSecOps lifecycle, as well as some standard tools used in the toolchain.



Plan. All projects require planning. DevSecOps projects must plan user stories with more than just features descriptions. They should include functional and nonfunctional requirements (like security and performance), acceptance test criteria, UI and UX designs, and threat models. Security begins at planning before a single line of code is developed.

Develop. Generally, it is much less expensive to develop secure software than to correct security issues after the software package has been completed. Development teams should start by assessing the maturity of the practices, gaining sufficient resources to provide necessary guidance (like the OWASP secure development guide), and implementing code reviews of software design and implementation.

Build. Automated build tools do more than compile code. A tool like Gradle can be used to conduct test-driven development, enforce standards for release artifact generation, and ensure design and implementations comply with team coding standards and security best practices through static code analysis.

Test. Test automation in a DevSecOps environment is much more than UI-focused Selenium tests. Strong security testing practices should include unit, front-end, back-end, API, database, and passive security testing. Passive security testing can be completed with little to no effort by utilizing a robust testing framework, likeSelenified, with a security scanner in proxy mode.

Secure. Traditional security testing doesn't go away in DevSecOps organizations; we just anticipate identifying far fewer issues late in the development process. When we identify vulnerabilities with security scanning, we often have a greater context of the issues and can more confidently determine if the vulnerability is a potential exploitation or a false positive.

Deploy. Using an infrastructure-as-code tool, like Chef, automated provisioning and deployments can be utilized to expedite delivery of software and ensure more consistency in the development process. It also can be used to audit properties and configurations across the IT infrastructure, as well as to enforce secure configurations for all systems and services.

Operate. Routine maintenance and upgrades are an important component of any operations team. Zero-day vulnerabilities need to be patched rapidly to reduce exposure time. DevSecOps teams leverage infrastructure-as-code tools so that updates can be applied to the entire organization's infrastructure rapidly and consistently, with no human error.

Monitor. Implementing a strong continuous monitoring program makes it possible to obtain real-time evidence of how your system is performing and exploitations that may be taking place against your system or its data. This allows organizations to review whether controls and systems function as intended on an ongoing basis.

Scale. Virtualization and the cloud is an important piece of any modern IT infrastructure. The ability to scale infrastructure to the changing demands of its userbase, or even being able to completely replace a compromised environment in minutes, are now real-world problems that we can't successfully solve with traditional data-center operations.

Adapt. Continuous improvement is a hallmark of any strong agile practice. DevSecOps practices also must continuously improve and adapt as issues (whether usability, security, or performance) are identified. This informs decision-making, planning, and how teams improve the overall software development lifecycle.

Measuring DevSecOps Success

When utilizing DevSecOps practices, success should be measured quantitatively by the efficiency of continuous development, quality of code, threat detection, and release cycles. Establishing key metrics to determine success—and tracking an organization's progress against those metrics—is important.

Here are some frequently used DevSecOps metrics:

- Deployment frequency
- Lead time
- Test coverage
- Detection of threats, security defects, and flaws
- Mean time to repair
- Mean time to recovery

DevSecOps success relies on gradual changes to various concepts within your organization. Existing frameworks can be enhanced or replaced with new practices. With the rise of DevSecOps, we get to truly define how operations, engineering, and security can be brought together to achieve unparalleled success.

When utilizing DevSecOps practices, success should be measured quantitatively by the efficiency of continuous development, quality of code, threat detection, and release cycles.

TECHWELL™

4 Keys to Protecting Your Data in a DevOps World

By Tom Austin

The range, type, and volume of data that companies now handle is growing exponentially. Ensuring that you have an understanding of where your data is—and where it came from—has become vital, particularly in light of increasing legislation and expectations around data protection. Regulations like HIPAA, the EU–US Privacy Shield, and the General Data Protection Regulation (GDPR) are already in place, and the California Consumer Privacy Act will come into play on January 1, 2020.

At the same time, the increasing adoption of DevOps is enabling companies to move from infrequent, big-bang releases to a constant stream of small releases in order to get features from the keyboards of developers into the hands of customers faster. That includes the database, because changes to front-end applications often require the back-end database to be updated as well. If continuously updating the database is excluded from your DevOps process, it hinders the ability to continuously deliver new functionality.

In order to be able to create, accurately test, and deploy database updates quickly and seamlessly, however, developers often need to work with copies of the production database that contain the very data that customers now expect to be protected. In a survey of almost five hundred SQL Server professionals, 83 percent of respondents said they want to use production data in development and testing, but they are naturally restricted due to concerns around data sensitivity, storage, and regulatory requirements. Without access to production-like data for testing, DevOps processes won't produce releasable code.

It can appear that these two needs—a desire for end-to-end DevOps and the protection of sensitive data—are in conflict, but if done

correctly, they can be two sides of the same coin. DevOps processes such as version control and delivery automation introduce the very measures needed to properly control, audit, and protect sensitive production data.

The key to keeping production data safe while using it during your DevOps process is to focus on four areas.

1. Discovery: Understand Where Your Data Lives

This may sound easy, but data within organizations tends to leak into lots of different places, so if it's not protected during use, that can expose the organization to data privacy risks.

Use configuration management tools and DevOps automation to track where data resides, and set up access properly so data is not left unprotected.

All customer data begins in production, but look to see where else it exists within your systems. Consider using automated data discovery tools to scan your network and cloud providers to get a full picture of all your data and where it is located.

Bear in mind that you have to be able to know the stages along the journey—it isn't enough to know where data begins and where it ends up. For example, someone in the test team might set up a temporary testing database in Azure and then forget about it when they move on to something else. If this test database includes production data, it gives people inadvertent access to data in ways that haven't been thought about or protected against.

The ultimate answer is to create a record of every server and backup, copy, and legacy system in order to gain a real understanding of where the data flowing through your delivery process is stored and used, as well as who has access to it. Use configuration management tools and DevOps automation to track where data resides, and set up access properly so data is not left unprotected.

2. Classification: Determine the Data's Sensitivity

Once you've mapped out where your data resides, you need to understand its type. Is it sensitive material? Does it include personally identifiable information? By classifying your data by type, you can see what needs to be protected when it's shared with those in your development and delivery process.

A key method for protecting sensitive data while still providing access to those who need it is through masking, or replacing sensitive data with realistic, anonymized test data.

When it comes to masking, it's too time-consuming and complex to mask everything. Instead, you should drill down to the column level and work out what information must be masked for each purpose. For example, data on what has been ordered from a retailer can often be displayed, but customer names and personal information about the individual who made the purchase will need to be masked.

3. Protection: Understand Purpose and Needed Access

In most companies, different teams need access to different data in order to do their jobs. This means the approach to protecting that data may vary and needs to be controlled.

Development teams, for example, want realistic copies of the database to test their updates against so that breaking changes can be identified at the time they are made rather than later in the delivery pipeline. Here, all sensitive data should be masked, and access to this data must be tightly controlled. Once the masking rules for this type of database access have been decided, there are tools that can automate both the masking and provisioning of database copies with the appropriate access permissions, ensuring that the development team has what they need without exposing sensitive information to others.

Business intelligence teams, meanwhile, may want to analyze data for marketing, sales, or management requirements. Knowing what products have been bought, what other products are bought at the same time, how much customers are spending, and where customers are located would all be valuable information. In this instance, partial masking would be more appropriate so that information like the products that sell best in which zip codes can be seen, while individual customer names and addresses remain hidden.

4. Monitoring: Ensure Ongoing Protection

Database monitoring has typically focused on performance, as organizations want to know about issues that impact customer usability and overall performance, such as slower than usual queries, deadlocks and blocking processes, large file sizes, and growth trends.

New data privacy regulations mean database monitoring must be taken to an entirely new level. Organizations are now required to monitor and manage access, ensure data is available and identifiable, and report when any breaches occur. Organizations also need to know and have a record of which servers and what data are being managed, and they must be able to discover the reason for any issues quickly and accurately.

Should a data breach occur, it becomes even more crucial for appropriate monitoring to be in place, as organizations are obligated

Data privacy and protection are becoming legal requirements for many organizations, and a moral duty for others.

under privacy laws to disclose the nature of any breach, the categories and number of customers impacted, the likely consequence, and the measures taken to address the underlying issue.

All of these privacy concerns make an advanced monitoring solution coupled with strong configuration management a necessity. This enables organizations to both monitor availability of servers and databases containing personal data and be alerted to issues that could lead to a data breach before it happens.

Defending Data through DevOps

Data privacy and protection are becoming legal requirements for many organizations, and a moral duty for others. Regulations are obliging organizations to put controls and measures in place to protect personal data, and customers also are becoming more aware of their rights and freedoms.

As a direct consequence, organizations can no longer afford to ignore the threats to the personal data they hold, particularly when they are speeding up deployments through a DevOps process. Fortunately, DevOps practices such as automated configuration management and access control support the need to protect sensitive data. By identifying where data is and what it is, masking sensitive data appropriately, and monitoring databases in use, breaches and exposure can be minimized. Introducing DevOps processes to help protect your databases will both enable compliance and allow continuous delivery of value to customers.

Using the Principles of the CIA Triad to Implement Software Security

By Sylvia Killinen

Several years ago, I worked with my employer to start a software security program. We truly started from the ground up, with no dedicated security development team. Testing is a related discipline, with relevant skills in investigation, troubleshooting, and reporting bugs, so we started there. I'm still a tester, but now I focus on security first and foremost. With the new systems now in place, I took some time to look back on our process and lay out an example of what worked for us.

While we were building our team, I frequently heard two questions: Why do we need to care about security? And how do we even start thinking about it?

The first question is the easier to answer. No company wants to star in the next data breach headline or upset their user base with an embarrassing privacy slip. That sort of mistake costs big money, but even in the absence of an attention-grabbing breach, there are hidden costs to developing without security in mind that can be addressed by changing the way we think about what constitutes security.

This article is an attempt at answering the second question. There's a common perception that security is all about protecting your system from malicious users. That's a good start, but it isn't enough. Relying on the single question "Does an attacker care about this?" both underestimates the creativity of a hacker with a goal and, more importantly, limits the effectiveness of the security program.

What Does It Mean to Be Secure?

Security isn't a feature of a piece of software; it's a property of the entire system, including its users.

First, let's define the user. Human users are a good start, but that isn't sufficient for figuring out the security a system requires. I had a lightning bolt moment when, the day after a product failed, another product that relied on a file delivered by the downed system also errored out. My first thought was, "We've just failed our users!" No people were angry at us yet, but the second system was affected.

Broadly, anything relying on your software is a user, whether it's a machine or a human customer. I now rely on this definition when writing and executing my tests.

The definition of security was our next big hurdle. While everyone would like to claim that their product is secure, the truth is that perfect security isn't possible in the real world. No set of rules is unbreakable, and no software system is truly unhackable. Instead of "How do we get to perfection?" we asked, "How do we get secure *enough?*" Then, we had to figure out what "secure enough" meant. Some pieces were obvious, as they were required by regulators or our partners. Some were less obvious and came up only as we started to dig into what the most visible aspects actually meant in implementation.

We found right away that the big requirements had hidden under-pinnings. Keeping our users' data safe also meant knowing which data were important and how safe they ought to be. Being able to make decisions on that data meant it had to be complete and trustworthy, and also accessible on demand, but only to the right people. Fortunately, a framework exists to help define those baseline requirements.

To get closer to the true goals of security, we decided to model our measures on what's known as the CIA triad: confidentiality, integrity, and availability. The terms are simple, but their definitions do not match the common usage of the words; a quick redefinition is neces-sary to get the full benefit of the model.

Confidentiality: The Right Data Going to the Right Users

Confidentiality is about not just keeping information private, but also keeping the right information, whatever that may be, from being exposed to the wrong people. The "right" information is sensitive or crucial for system operation. For example, a revealed server name is unimportant to most users, but is a roadmap to an attacker, and

> *No set of rules is unbreakable, and no software system is truly unhackable.*

many forms of personally identifying or financial information can be used for profit.

The wrong users, then, are any people or systems not authorized to have access to the data. This is defined by the user's role. A trusted employee who isn't an administrator probably shouldn't have access to some data, a thief shouldn't have any confidential data at all, and a content scraper should only see public information. Privacy con-cerns nearly always map neatly to confidentiality problems.

On the other hand, not all data is the right data. Things the public can easily find out usually aren't confidential, although there are exceptions. This will depend on your system and what it handles, as well as any legal or regulatory requirements affecting your projects.

Integrity: Good Data from Trustworthy Sources

In order for the system to have integrity, the data must be valid, come from a trusted source, travel through secure means that don't allow it to be intercepted or tampered with, and be stored where it can't be viewed or altered by the wrong users. In effect, if your data can't be tampered with in motion or at rest but it came from who-knows-where, you probably still shouldn't trust it or make business decisions based on it.

This is a wider definition of integrity than is usual for the CIA model because it takes into account whether the data is trustworthy to begin with. Thus, it defines "Can I make a good decision on this?" as part of information security. Bad decisions cost resources, from the small example of having to make an extra phone call to the large examples that make the news every day.

Availability: Keeping the Data Flowing

Availability is probably the most straightforward measure used by this framework. A system is available when its data are accessible by the right people, when they need it. This can be easily expanded to include considerations of load: "When they need it" can also mean "as fast as needed, as often as needed." While this is a slight extension of the CIA framework, I feel it's a logical enough leap that it does not stretch the model past its original intent.

All three of these factors affect each other, and while the words are simple, the implementation is as complex as the system under test. The real value of the CIA framework lies in expanding the definition of "secure enough" to also include "Can the business trust this data to be valid and accessible for making critical decisions?"

The CIA Triad in Practice

We discovered several gaps in our development systems using the CIA model. Even though we wanted to have secure products, there were pieces missing from our requirements and, thus, from our development and tests. High load was a good example; we discovered the hard way that certain pieces of software would not stand up to much more traffic than they ordinarily received. As a software tester, one of the best tools I have to get a discussion started is to log a bug or a missed requirement.

Our information security team also started conversations with the product planning team about security requirements. When the questions started coming in, we found good external training resources on how to reduce vulnerabilities in code and gave them not only to our developers, but also to our testers, so that we could come at the problems from multiple angles.

The downside of the CIA framework is that it looks only at data. Stopping there doesn't do justice to the full complexity of any given system. A full list of aspects covered by other frameworks and how we addressed them is well out of scope for this writing, but the usual tools of gap analysis, good research and planning, and strong teamwork can overcome those obstacles as well.

When more people started using the CIA framework and asking questions about the software, we saw a sudden and dramatic increase in the quality of our work. If you need to implement or improve a security program for your software, think first about these three crucial principles: confidentiality, integrity, and availability.

> *The real value of the CIA framework lies in expanding the definition of "secure enough" to also include "Can the business trust this data to be valid and accessible for making critical decisions?"*

DevOps and Security: 5 Principles for DevSecOps

By Jeffery Payne

In one sense, it seems crazy to add additional terms to the DevOps portmanteau. A proper DevOps solution should include security, but we don't have "BusDevOps" to highlight the business part of DevOps, so why does DevSecOps even exist?

On the other hand, the security industry has mobilized around the term "DevSecOps" much more than they did DevOps, and that is helping to accelerate the integration of security best practices into continuous integration and delivery. It is also giving the security team an identity in the continuous world we now live in, and that is a positive development.

DevSecOps is the integration of security practices, principles, tooling, and knowledge into the software development, testing, and delivery process. Traditionally, security assurance of applications, environments, and production systems has typically been performed late in the software delivery lifecycle. With the trend toward a more continuous delivery and deployment process, late-lifecycle activities like security assurance present a significant hurdle to continuously delivering value to customers. DevSecOps addresses this challenge by shifting security assurance activities, personnel, and automation closer to development.

There are several competing DevSecOps manifestos out there that attempt to define DevSecOps values. Here are five of the themes I particularly like.

1. Build security in instead of bolting it on
Security controls like authentication, authorization, and encryption can help secure applications, but fundamentally, software security is about the inherent quality of the design and software we produce. To build secure applications, it is not enough to just add security controls. Threat modeling, defensive design, secure coding, and risk-based security testing are necessary.

2. Lean in over always saying no
Security has the reputation of being a hurdle instead of an enabler. DevSecOps strives to push security practices into the entire software lifecycle so security is assured as we go instead of at the end.

3. Rely on continuous learning instead of security gates
While it is important to identify and stop security vulnerabilities from getting into production, security at the speed of continuous delivery will only happen if root causes of vulnerabilities are evaluated, teams learn from the mistakes they have made, and the results are incorporated back into the software development process to avoid making the same mistakes again.

4. Encourage open collaboration over security requirements
It is as difficult to get security requirements right up front as it is for any other set of requirements. Therefore, a better approach is for security teams to be involved day to day in all activities associated with planning, implementation, and testing so that as requirements and needs change, security needs tune and adjust as well.

5. Share threat intelligence over keeping information to ourselves
It's an age-old security debate: Is it better to share knowledge of vulnerabilities and threats or hide them? Sharing allows others to proactively correct issues but gives hackers additional information to exploit those that don't keep pace. However, hiding information prolongs the exposure window for vulnerabilities. DevSecOps rightly believes it's better to share, learn, and improve security constantly to keep ahead of the bad guys.

A Definition of Done for DevSecOps

By Gene Gotimer

DevOps means different things to different people. To me, it is a culture of communication and collaboration across the entire team.

In DevOps, we have a software delivery pipeline that checks, deploys, and tests every build. The goal is to give us confidence that we are producing a viable candidate for production, so we have to look at many different aspects of software quality, including security. We also have to look into functionality, maintainability, performance, and many other characteristics of our software, but if we focus on security, then we often call that DevSecOps.

How do we remember to check all those factors for every story? We define a checklist of what we want to do and test for each feature. This definition of done is a minimum—there is a subset of tasks and checks that we want to do for all stories. We don't have unlimited time and resources to do every conceivable test before release. Instead, we have to have a baseline so we can trust that we have done enough.

Some of that baseline of trust will come from just making sure we write good, quality code. Static analysis tools will check that we are following style guidelines and best practices. Good unit tests set the expectations of how the code is intended to work. That makes maintenance easier and further builds our confidence in our software.

Our applications are made up of more than just the code we write ourselves. We rely on frameworks and libraries written by others, and we deploy on systems that are made up of packages and services. We can bolster those by making sure that they have the latest bug fixes and security patches.

And we wouldn't be confident in our code if we didn't at least do some security testing. We must make sure we aren't missing obvious security holes. We need to check that we don't allow users to do things or see data that they shouldn't, and we have to protect and monitor our systems.

But most importantly, we have to look at each feature and assess the associated risks. Is there something about this particular change that warrants a little extra attention? Are we dealing with sensitive data, like Social Security or credit card numbers? Could this feature be used to mislead or disrupt others using our system? Do we have some way to make sure data or credentials aren't fake or altered?

On each project, take some time to see if you have enough testing throughout your process—security or otherwise. Make sure your definition of done leads you to trust your software. A few extra checks can help you be more confident in the code you are developing.

TECHWELL™

Managing Security Testing in Agile Software Development

By Sanjay Zalavadia

One of the biggest myths in the world of agile development is that there is not enough time during each sprint to do security testing. However, many teams new to the agile methodology may have a lot of questions about security testing in their new environment.

When should you perform security testing? Is there any change in the way security testing is done? What are some of the tips for introducing security testing to an agile environment? This story will attempt to answer these common queries.

Agile methods recommend testing early and often. Irrespective of whether it is functional or nonfunctional testing, these duties should be done from the first sprint. At the end of each sprint, an increment of the product is created, and, if possible, should be deployed as well. If external security agencies or white-hat hackers should be involved, they should be available during each sprint as needed.

As such, the process of security testing remains the same as in traditional methods; modules are just security tested in small increments during each sprint.

Yes, it is a bit uncomfortable to imagine doing security testing or other nonfunctional testing from day one. However, this is the most beneficial and cost-effective way of testing a product for the enterprise. By bringing the process up front, most of the security issues are detected with enough time for the programmers to fix issues.

An important distinction in the agile test management method is that quality is everyone's responsibility, not just testers'. It's a good idea to ensure that all team members are trained in and aware of the security guidelines.

For those who are new to security testing in an agile environment, initial security-related stories could be identified during the planning phase. It is also recommended to keep at least two sprints' worth of backlog items ready at any point in time. This gives enough room to invite security experts for discussions as needed.

Penetration tests should be automated to reduce manual testing as much as possible.

When establishing the team's definition of done, it may be helpful to stipulate that no stories are complete without code being reviewed for security.Teams also should use "definition of ready" and "definition of done" checklists to keep security-related stories ready and items tested from a security perspective.

Here is a sample "definition of done" checklist focusing on security guidelines:

- Static code analysis done to check buffer overflows
- Peer review done by security experts per security coding checklist
- Threat modeling complete
- Penetration testing done and approved

"Fail fast and fail early" should be the mantra while implementing security testing in agile environments. You should consider security testing the same way you approach any other testing, and you should plan to complete it during every sprint.

There is definitely time in agile software development to perform security testing. You just need to be dedicated to building it into the sprints.

TECHWELL™

Insight from the Industry

"Who would buy a really fancy alarm system, monitoring, all that, and never check to see if the alarm sounds? That's where a lot of companies are. They have all the technology. Sometimes the people have been trained. But none of it has ever been tested in probably, I would say, about 80 percent of the cases. It's that false sense of security."
—Randy Rice

"To me, everybody in the testing community should be learning how to look for security vulnerabilities. They're that important, it ought to be integrated in with all of our other testing. It should not be something that is only done by some other group right at the end of the lifecycle. That doesn't work in DevOps, it doesn't work in agile. We've got to work to fix that."
—Jeffery Payne

"The recent spate of cyber attacks and data leaks mean cyber criminals are constantly changing their tactics. Everyone must be racing to beat hackers at their own game. Companies should keep up to date on the latest security threats and, if necessary, employ white-hat hackers to stay ahead of potential risks."
—Ryan Bronson

"As with any kind of testing, security testing for web applications should start with strategizing a complete plan and gauging the probable risks and attacks in order to formulate the most effective tests. While automating your security testing can ease efforts and make the process faster and much more efficient, there has to be a human touch to understand and anticipate the thought processes of potential hackers. Testers need to be creative in their test efforts to keep users safe when using their products."
—Ketan Sirigiri

"Unfortunately, shifting security left is easier said than done. Development and testing teams often lack application security expertise, and staffing and budgetary concerns limit IT security's capacity to do more frequent testing and ability to embed security analysts in software development teams. Having traditional testers perform some security testing efforts earlier is a great way of achieving a balanced approach to shifting left while being mindful of staffing and budgetary challenges."
—Alan Crouch

To me, everybody in the testing community should be learning how to look for security vulnerabilities.

"Current data indicates that many IoT devices have little, poor, or no UI security setup support. IoT producers seem happy in early adoption to let the user have the responsibility for security. This reminds me of the early days of PCs and the web. In the long run, the successful IoT producers will get better at device security."
—Jon Hagar

"In the event of a data compromise or breach, a security expert can be the difference between minimal damage and a widespread system breakdown. The advantage of a professional will be evident when your company expands operations, as the expert will be able to help you reduce the number of breachable contact points."
—Ray Parker

Additional Resources

MORE INFORMATION FOR SOFTWARE PROFESSIONALS

TECHWELL**HUB** A SLACK COMMUNITY

The TechWell Hub is a great resource to get your questions answered, help others with problems they're stuck on, and engage with experts in software. Follow channels like #DevSecOps, #Security, and more.

JOIN HERE

NARROW YOUR SEARCH TO A SPECIFIC TYPE OF RESOURCE:

AgileConnection Community

AgileConnection brings you the latest in agile and DevOps principles, practices, and technologies. Check out articles and interviews from experienced software professionals and thought leaders, and join the community to gain access to member-exclusive content such as conference presentations, weekly newsletter updates, Q&A discussions, and more.

DecSecOps Articles

AgileConnection is home to thousands of DevOps software resources, including articles, archived Better Software magazine articles, conference presentations, and interviews with industry notables. Check out these DevSecOps articles to read about how to reduce risk, protect data, and build security into your DevOps pipeline from the start.

TechWell Conference Presentations

Couldn't make it to a TechWell conference to sharpen your security and DevSecOps skills and knowledge? TechWell conference presentations are available to AgileConnection members soon after conferences end. **Click here** to join AgileConnection and access conference presentations related to security and DevSecOps.

Interviews

Each year, TechWell interviews dozens of software professionals, including well-known thought leaders, seasoned practitioners, and respected conference speakers. **Click here** to read, listen to, and watch interviews with DevOps and security experts.

Agile + DevOps Conferences

The Agile + DevOps conferences are full of keynotes, sessions, tutorials, and training classes covering project management, test automation, the agile development lifecycle, and more. Learn from experts in the field and network with your peers to get the most immersive agile conference experience possible.

DevSecOps Summit at Agile + DevOps East

The DevSecOps Summit will be a daylong series of first-person talks, giving an ideal perspective on how you and your team can enable faster application development with more rapid deployment to production while integrating security into your DevOps initiatives. Explore the program **here**.

coveros **Accelerate Delivery**

Our partner, Coveros, has significant DevSecOps experience and can help organizations implement DevOps with security in mind or integrate security capabilities into existing DevOps processes. Coveros offers more than a dozen courses on DevSecOps, DevOps, and security—all of which include best practices taught by industry leaders. Whether you're looking to get hands-on experience for yourself, your team, or your organization, Coveros has a learning solution for you.

DevOps & DevSecOps Courses | **Software Security Courses** | **Agile & DevOps Transformations** | **DevOps Engineering** | **DevSecOps**

TECHWELL™