DevSecOps
A Systemic Approach for Secure Software Development

By Seetharaman Jeganathan

The author reviews how security processes can be effectively embedded in the DevOps model to increase the success of IT projects in an organization.

Abstract

DevSecOps (security DevOps) is relatively new in the field of information security. The fundamental idea aligns with the concept of having security as an integral part of the software development principles, processes, and methodologies. The DevOps model is rapidly being adopted by the technology industry in order to support the need for developing and releasing core business systems and applications to customers in much faster and reliable ways than the traditionally followed software development life cycle (SDLC) models. The security industry has adapted to the demand for DevOps by introducing relevant processes in the form of DevSecOps principles and methods without affecting the original intent of DevOps. The author will review how security processes can be effectively embedded in the DevOps model to increase the success of IT projects in an organization. However, this article is not meant to review how to adopt DevOps for its benefits when compared to traditional approaches.

DevSecOps is relatively new in the field of information security. There are several definitions for DevSecOps; however, the more relevant ones in my view are:

> *DevSecOps enables organizations to deliver inherently secure software at DevOps speed* – Stefan Streichsbier—CEO & Founder, GuardRails
>
> *DevOps + Security = DevSecOps*

Fundamentally, information security functions have been providing confidentiality, integrity, availability, and accountability services to information systems and infrastructure. These services are often referred to as primary goals for information security functions. The primary objective is to secure the overall IT systems and business functions to support the growth of the underlying business. But in traditional software development models, security is often viewed as an afterthought wherein security testing is mostly conducted during specific testing phases of the software development life cycle, which are usually planned far ahead (right) in the schedule; hence security comes into play at a later stage in the cycle. This paradigm is changed in the DevOps + Security model where security is pushed to the forefront as much as possible and advised to begin from the early stages of the software development cycle.

DevSecOps – a practical approach

DevSecOps expects change in the organization's culture, behavior, and job functions of at least three different teams—development, security, and operations—wherein they are required to work together during all the phases of software development. Unfortunately, there is no one-size-fits-all model for all organizations. Hence, it is imperative for each organization to figure out how to work together to adopt this evolving new practice, tools, and technologies and secure the overall software delivery model. In order to achieve this, an organization's key elements such as people, process, and technology must change to adapt this shift in the culture. A conceptual model describes people, process, and technology as pillars building DevSecOps, supported by enterprise, IT, and security governance processes followed by the organization (figure 1).

People factor

Changes at the people/organization level is the first step in the journey towards accomplishing DevSecOps. In the traditional software model, different teams such as development, testing, and operations get involved during the specified cy-

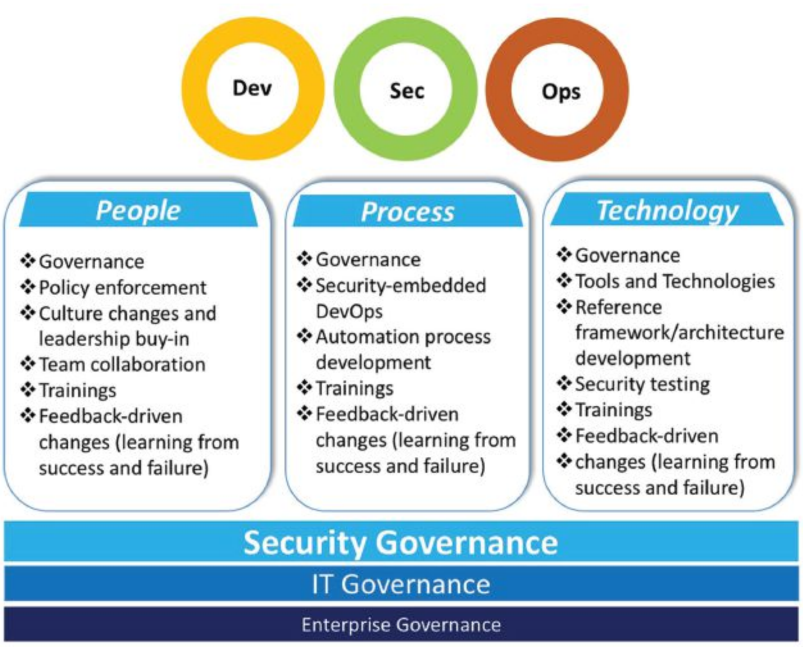
Dev
Sec
Ops
People
❖Governance
❖Policy enforcement
❖Culture changes and leadership buy-in
❖Team collaboration
❖Trainings
❖Feedback-driven changes (learning from success and failure)
Process
❖Governance
❖Security-embedded DevOps
❖Automation process development
❖Trainings
❖Feedback-driven changes (learning from success and failure)
Technology
❖Governance
❖Tools and Technologies
❖Reference framework/architecture development
❖Security testing
❖Trainings
❖Feedback-driven
❖changes (learning from success and failure)
Security Governance
IT Governance
Enterprise Governance

Figure 1 – DevSecOps conceptual model [3]

cles and perform their tasks in a sequential fashion. To adopt DevSecOps, we need to break these silos and make these teams collaborate from the very beginning of the development cycle.

Build a culture of security by educating the teams involved about the negative impacts of security breaches on the organization and develop the responsibility of adopting security as a foundational requirement for software development. Security cannot be compartmentalized as the responsibility of just a few security team members, but rather it is the collective responsibility of developers, testers, and operations *along with* the IT security team (figure 2).

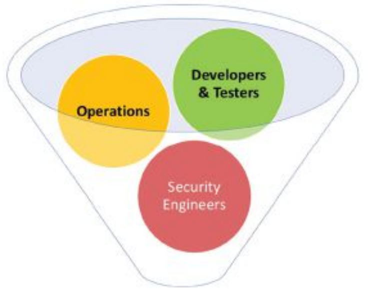

Figure 2 – DevSecOps team

This cultural shift is not going to be easy to make and succeed in its first attempt. It needs to begin with buy-in support from leadership by building an unambiguous business case with references to case studies, benefits, and ROI factors. Without leadership support, any cultural changes be in vain. Even after garnering the support, it is not advised to adopt the "go big or go home approach." For example, if an organization is running a mission-critical IT project to support its business functions, it is not advised to switch from traditional to this advanced model right in the middle of the project as it probably won't lead to success due to the various challenges involved. Instead, train the resources for this new model and run a smaller project to roll out the changes in the development processes. Learn from success and failures and reward the teams based on their performance and achieving desired outcomes. It is important to understand that DevSecOps adoption doesn't have to be at the entire organizational level from the beginning; instead it can begin with a focused group and slow rollout across several business units [4][5].

Process factor

Changes in IT & business processes come next and are vital for successful adoption of DevSecOps in an organization. Each organization is unique, depending on the IT environment complexity, architecture preferences, technology stack, risk tolerance levels, and operational maturity. Hence, changing and adopting any new processes, principles, and methods must go through careful planning, analysis, and iterative phases to successfully roll out the changes. Figure 3 shows the shift in the development process from Waterfall to Agile to DevSecOps and the reduction in the application development life time, which is a prime benefit of this change adoption.

In DevOps, everyone must focus on the customer, delivering the needs of the customers in a continuous integration and continuous delivery (CI/CD) fashion. But traditionally, security teams focus on security-centric goals to make the organization compliant with regulations and privacy laws. If security hinders the DevOps speed of CI/CD delivery model, then it will impact the success of the customer-centric delivery model. This is why security is expected to collaborate and become an integral part of the DevOps teams. Whether security adopts the concept of DevOps, or DevOps embraces security, the goal here is to deliver secure products/projects at DevOps speed. There is no hard and fast rule in defining the DevSecOps process for an organization. However, establishing a process model will lead to the next

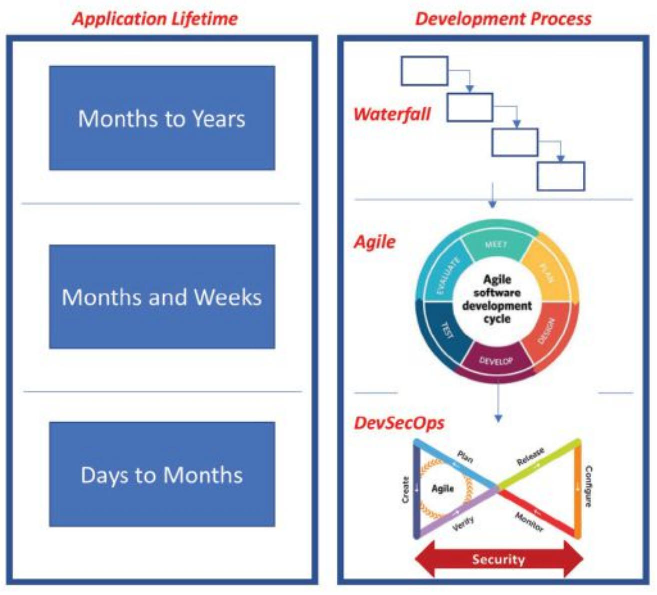

Figure 3 – Waterfall to DevSecOps [2]

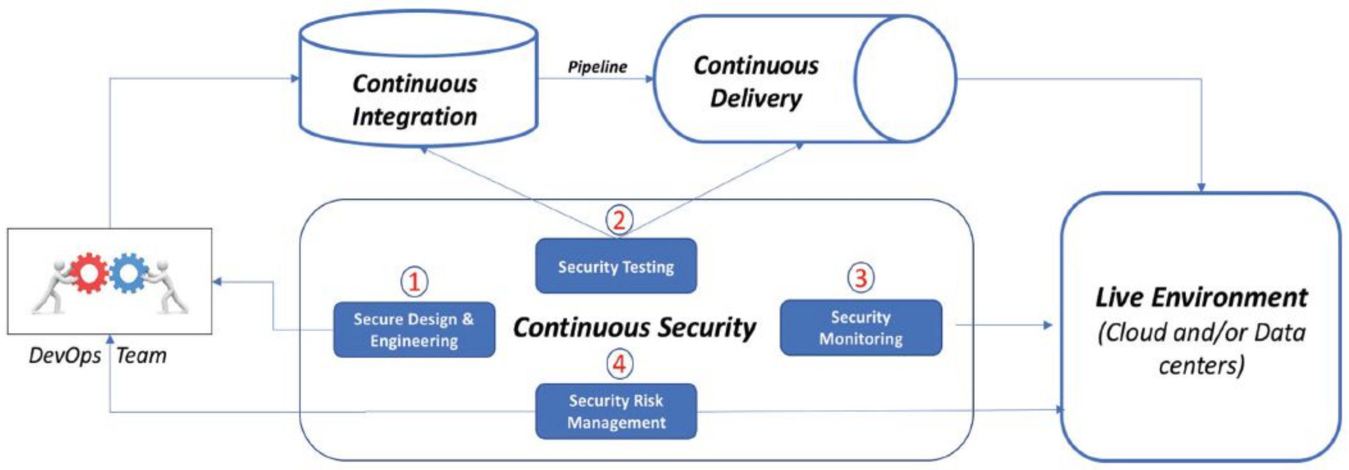

Figure 4 – DevSecOps continuous security model logical architecture [11]

step of selecting the tools and technologies to implement the process [11].

As security becomes a continuous part of the DevOps cycle, it can be referred to as a continuous security (CS) pipeline. Let's focus on this continuous security pipeline and build a process model (logical architecture) for implementing DevSecOps for any given organization. Figure 4 describes a conceptual DevSecOps process model. In effect, the process model has the following essential features:

- Continuous integration (CI)
- Continuous delivery (CD)
- Continuous security (CS)

In the proposed process model, the continuous security approach offers the following key services to the DevOps ecosystem.

(1) Security design and engineering

Security design and engineering services ensure that the products and services developed by the DevOps team comply with security best practices, regulations, standards, and laws and achieve data privacy and protection. Security requirements are carefully analyzed and properly designed during the requirements and design phase. In addition develop threat modeling (simple and complex) and implement controls during the coding stage, for example, implementing controls to prevent SANS 25 and OWASP Top 10 vulnerabilities for web-based applications.

Complex threat modeling is essential for business-critical systems to predict different possible attack vectors and to plan the system to be "fail safe" with no exposure of critical system data.

Secure coding is a development practice in which security-related weaknesses, defects, and integration errors are addressed by following the established simple and complex threat models, for example, applying controls for input validation, session management, user credential validation, user access control, data protection and privacy, logging, API security, detecting security misconfigurations, etc. while coding the software modules [3].

(2) Security testing

Security testing is the next critical facet of DevSecOps, where the software modules go through various testing cycles for quality assurance. Security testing should not only focus on the software modules but also the end-to-end pipeline, live production systems, infrastructure, databases, middleware components, and all integration points to protect from security attacks originating from any of them. When it comes to testing, it must differ from the traditional manual testing approach by adopting automation wherever possible.

The security team must collaborate with developers in testing the software modules for common vulnerability exposures such as SANS 25 and OWASP Top 10 to make sure that the security basics are followed and assured for quality. Over time these testing processes must be automated against CI/CD pipeline and all the defined test cases—functional and security—must pass before the code moves to the live production environment. Source code analysis or static application security testing (SAST) is a common process followed in analyzing the source code of software modules to detect commonly known security flaws and misconfigurations.

Developers and security teams must collaborate to integrate source code analysis into the integrated development environment (IDE) setup where coding is done to develop software modules. Likewise, dynamic application security testing (DAST) and interactive application security testing (IAST) are primarily focused on securing web applications against several know vulnerabilities before being released to the live environment. Developer, tester, and security teams must work together in implementing these mandatory testing cycles in an automated fashion in the DevOps pipeline [8][9].

The other two facets, security monitoring and security risk management are just not specific to DevSecOps but are commonly followed security principles that would add value in DevSecOps.

(3) Security monitoring

Security monitoring focuses on live and offline analysis of logs created in the live systems, infrastructure, and applications for known attacks and vulnerabilities and alerts the security team in order to respond to security incidents/loopholes. This also provides forensic capabilities when critical security incidents are being investigated. Intrusion detection, intrusion prevention, incident response, forensics analysis, etc. aren't any different for DevOps. But, they must be integrated into the process.

ISSA International Web CONFERENCE

ISSA Thought Leadership Series

The Persistent Pernicious Myths and Hidden Truths of Cybersecurity

60-minute Live Event: Wednesday, November 6, 2019

10 a.m. US-Pacific/1 p.m. US-Eastern/6 p.m. London

IT implementors are made less successful due to "Technical Debt." Cybersecurity suffers from "Myth Debt," where the same untrue tropes are repeated and hold us back. It takes experience to recognize these myths, but worse still is they can mask the valuable truths that lie within the myth. These never-dying misunderstandings spread outside cybersecurity and falsely inform IT and business leaders, making it harder still to stop bad things from happening.

So let's poke some holes in some myths, pick some or all:

- Insider threat is the biggest worry
- Great Pen Tests mean excellent security
- Any attacker motivated enough can hack you easily
- Security training and education of devs will get us secure code and apps
- The cloud is secure. The cloud is insecure
- Encrypting everything makes for strong security
- Spending more on security makes security better
- Excellent endpoint security means we no longer have to worry about network or other security
- You can't defend yourself against ransomware

Moderator: Jorge Orchilles, SANS Certified Instructor

Speakers:

- Greg Young, VP, Cybersecurity, Trend Micro
- Zane Lackey, Co-Founder, Chief Security Officer, Signal Sciences
- Dr. Cragin Shelton, DSc, CISSP

Generously supported by

TREND MICRO

Click HERE TO REGISTER.

For more information on these or other webinars:
ISSA.org => Events => Web Conferences

(4) Security risk management

Similarly, security risk management is a continuous process where security risks are analyzed and mitigated by applying cost-effective security controls. However, even risk management must work together and support the cycle at DevOps speed and not hinder the process. A lightweight approach or rapid risk-assessment (RRA) is preferred over the traditional approach for DevSecOps. Tailor the organization risk management process to DevOps and ensure the threat modeling process is handled effectively and that all applicable threats are addressed before the software module is released to the production environment. Where applicable, adopt some of the widely used threat modeling methods such as STRIDE (spoofing, tampering, repudiation, information disclosure, denial-of-service, elevation of privilege) and DREAD (damage potential, reproducibility, exploitability, affected users, discoverability) by Microsoft to support the threat modeling processes and quantify the risks to the organization for releasing vulnerable software to the live environment [6][10].

Technology factor

After the people and process pillars are defined, the focus should be on defining the technology stack to support the shift to DevSecOps adoption. We shall focus more on security considerations, tools, and technologies in this section. There are a good amount of options to choose from a wide variety of open source tools as well to support the DevSecOps technology stack. But it is important to compare and contrast the capabilities between open source and commercial products so as to make informed decisions on what is good and cost-effective for the organization. Figure 5 shows a typical DevSecOps pipeline in multiple stages until it is released to the live environment.

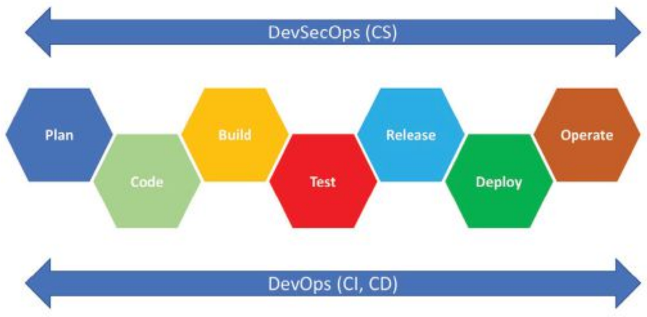

Figure 5 –DevSecOps pipeline stages (CI, CD, and CS) [9]

Security considerations for the plan through operate stages of DevSecOps are outlined below. These are not exclusive or limited to what is called out but just a reference to develop further. We have the opportunity to adopt industry standards such as ISO 27002, NIST SP800-53, or others as a starting point and tailor to what will best fit for the organization.

Plan

This is the stage where security requirements and design are done for software development and securing the entire DevOps life cycle.

- Start with a high-level rapid risk assessment for the new release and quantify the risks by evaluating the threat models.
- Plan and secure the DevOps lifecycle tool, typically web-based tools such as GitLab, Azure DevOps, etc.
    - For example, secure access points based on role- or attributes-based access control models.
    - Protect user logon by integrating with company federation (identity provider) and web-access management tools if exist, otherwise with a compensating control to meet the requirements.
    - Apply 2FA/MFA based on the criticality of the environment and systems.
- Ensure user access keys, privileged service accounts, API keys, etc. are protected properly with privileged account security tools if exist, otherwise with a compensating control to meet the requirements.
- Define infrastructure protection controls and enforce segregation of duties. For example, developers don't need access to the live environment, only the operations team [1][11].

Code

- Apply secure coding practices, integrate SAST tools (OWASP SonarQube, Fortify, Veracode, Checkmarx, etc.) in the IDE tools (Eclipse, IntelliJ, Visual Studio, etc.).
- Enforce industry-followed secure-coding practices (e.g., OWASP and CERT) at this stage.

- Train developers to adopt security principles such as confidentiality, integrity, availability, and accountability while coding software modules.
- Conduct peer code reviews wherever applicable.
- Design and develop unit test scripts focusing on security along with functional unit testing of the modules, for example, input validation to prevent SQL injection attacks.
- Eliminate the use of vulnerable components from the beginning. For example, if adopting open source technologies, understand if it is being supported by an active working group to address any known vulnerabilities. Otherwise it is not advised to adopt them in the development tasks [1][11].

Build

This is the stage where software modules are checked into the source code repository and made available to package and bundle for deployments into next environments such as QA, user acceptance testing, pre-production, and production.

- Ensure best practices are followed in segregating the repository by branching it to production vs. non- production environments.
- Apply access and separation of duties controls to make sure developers aren't changing code directly in any higher branches but changing in development and passing all the required functional and security testing.
- Adopt industry-followed automated tools for building and packaging software modules (e.g., MSBuild, Maven, Gradle, etc.). Avoid manual intervention as much as possible.
- Adopt native access controls offered by the source code repository tools to prevent accidental misconfigurations, deletion of source code, and dependency errors.
- If leveraging public/private-hosted repositories such as GitHub, Git, GitLab, BitBucket, etc., additional controls are required for user access controls. Scan for privileged credentials such as password and keys to avoid security mishaps. There are known security breaches due to security misconfigurations at this stage.
- Apply 2FA/MFA to protect unauthorized access to code repositories from all the access points (web layer).

Test

Security testing is a major difference between traditional and DevSecOps development methods. Even though unit-level security testing is done by developers, extensive system and integration testing occurs at this stage to prevent various security flaws in the software modules. Security teams at this stage are referred to as blue team and red team, where red team focuses on offensive testing and blue team focuses on preventing attacks from the red team.

- Web application security scanning—leverage known tools, open source and/or commercial—to scan the built modules for known web application vulnerabilities. This is commonly referred as DAST and IAST testing techniques. There are wide range of tools to leverage for application security scanning and recommendations to fix common vulnerability exposures (e.g., OWASP ZAP, IBM AppScan, Veracode, Qualys, etc.) [8][11].
- Fuzzing tools (Radamsa, AFL, Burp Suite, ZAP etc.) that follow fuzzing techniques for negative testing and validating the behavior of software modules [11].
- Penetration testing (red team) – this is typically done by an external party with legal understanding with the orga-

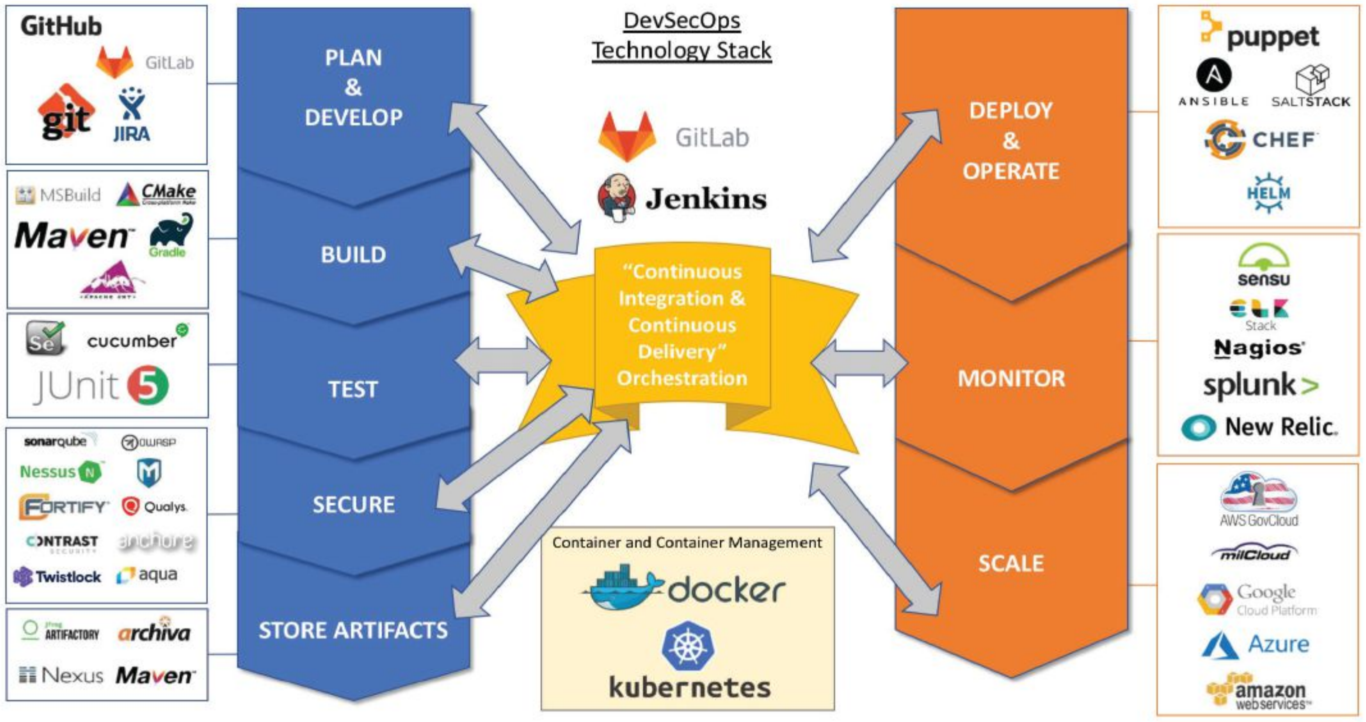

Figure 6 – DevSecOps technology stack sample reference
(Source - Nicolas-Chaillan-OSD-DoD-Enterprise-DevSecOps-Platform-DAU-Presentation-v1.3-7Mar2019)

nization to penetrate their systems and infrastructure to expose vulnerabilities and further help to fix the problems.

- Similarly, non web-based applications, APIs, data access layer, integration layer, and middleware components all must be scanned with appropriate vulnerability scanning tools and techniques for holistic security testing of the environment [3].

Release and Deploy

The release and deploy pipeline in DevSecOps is a set of processes, tools, and technologies where software modules are released to the lower and live environments on a defined schedule and/or polices that are applied for the specific release items. This is supported by various tools such as Puppet, Ansible, CHEF, etc. where scripts are written to automate the release items as per the configured policies.

If containers are used, extra caution is required in releasing the containerized modules to its storage such as Docker Hub.

Common security considerations that apply here:

- Ensure proper authentication controls are applied, prefer 2FA/MFA where applicable.
- User access controls. Grant access to users based on different DevOps roles, based on need-to-know principles.
- Protect privileged credentials by integrating with native or organization-level controls for privileged access security.
- Follow container security best practices for pushing and pulling container objects from storage. Manage the permissions in the pipeline so that accidental mishaps are avoided in the process. Adopt container signing practices for increasing the trust of containers before publishing to the storage hub [11].
- Ensure post release and deploy validation is automated to scan for common vulnerabilities such as credentials compromise, security misconfigurations, vulnerability scanning before and after release and deploy cycles. If a critical vulnerability occurs post new release, then roll back the changes immediately and fix the vulnerability before making it live.

Operate

This stage is last but not least. The operations team is heavily involved in this stage rather than the developers and testers. However, the security team is actively involved in continuous monitoring, analysis, and protection for the live-run environment, whether it is on the customer data centers or public/private cloud-based deployments.

All standard security considerations and operations controls apply here regardless of DevSecOps or any traditional method of software development cycle. Continuous security operations such as log analysis, incident response, forensic operations, intrusion detection and prevention, and post mortem analysis are some of the key considerations at this stage.

Technology factor – Selecting tools and technologies

DevSecOps is an emerging field with various tools and technologies available to support the process and principles. However, selecting the right tool sets could be a daunting task for any organization, and it heavily depends on the organization's technology adoption, training culture, and support from leadership. DevSecOps doesn't have to be an organization-wide initiative from the beginning but rather may be a small initiative from one or more of the business units and slowly expand to the entire organization based on the success and failure in the process adoption. Learn from mistakes and go through maturity cycles to figure out what would work better for your organization. As there is no one-size-fits-all, a sample is given in figure 6 of how various tools and technologies support the DevSecOps ecosystem.

Netflix's DevOps adoption case study is a personal inspiration and I am a fan of the "Simian Army" project where chaos techniques and tools are used to create havoc in the production environment and work backwards to prevent total shutdown. Adopt these principles in your journey as applicable [7].

Conclusion

As we have reviewed so far, DevSecOps is a cultural shift where people, process, and technology elements must embrace change and work together. Success or failure depends on how committed the teams are in the overall process. Starting from conceptualization through implementation, DevSecOps needs diligent effort from several teams to be successful. It is not a bad thing to fail but learn from mistakes and prevent them from re-occurring. By applying layered and continuous security for DevOps, it is mostly certain that any organization can successfully roll out this process and gain the envisioned benefits.

References

1. Allen, Ben, et al. "Continuous Security: Exploring the DevOps Toolchain," SANS, 11 Oct. 2018, https://blogs.sans.org/appsecstreetfighter/files/2018/10/DevSecOps_Exploring_Phase1-2.pdf.
2. Chaillan, Nicolas. "DoD Enterprise DevSecOps Platform," DAUAA – https://dauaa.org/wp-content/uploads/2019/03/Nicolas-Chaillan-OSD-DoD-Enterprise-DevSecOps-Platform-DAU-Presentation-v1.3-7Mar2019.pptx.
3. Elder, Michael, et al. "Security Considerations for DevOps Adoption." IBM – https://www.ibm.com/developerworks/library/d-security-considerations-devops-adoption/.
4. Janca, T. "DevSecOps: Securing Software in a DevOps World," DZone DevOps (2019, September 26) – https://dzone.com/articles/devsecops-securing-software-in-a-devops-world.
5. Lam, Thomas. "DoD Enterprise DevSecOps Reference Design" (2019, August 12) – https://dodcio.defense.gov/Portals/0/Documents/DoD_Enterprise_DevSecOps_Reference Design v1.0_Public Release.pdf?ver=2019-09-26-115824-583.
6. Mozilla. "Rapid Risk Assessment" – https://infosec.mozilla.org/guidelines/risk/rapid_risk_assessment.html.
7. Netflix Technology Blog. "The Netflix Simian Army," Medium, 20 Sept. 2018 – https://medium.com/netflix-techblog/the-netflix-simian-army-16e57fbab116.
8. OWASP. "Free for Open Source Application Security Tools." OWASP – https://www.owasp.org/index.php/Free_for_Open_Source_Application_Security_Tools.
9. Porter, Tom. "DevSecOps - A New Chance for Security," DZone DevOps, 19 July 2019 – https://dzone.com/articles/shifting-left-devsecops.
10. Shevchenko, Nataliya. "Threat Modeling: 12 Available Methods." Carnegie Mellon Univesity, 3 Dec. 2018 – https://insights.sei.cmu.edu/sei_blog/2018/12/threat-modeling-12-available-methods.html.
11. Vehent, Julien. *Securing DevOps: Security in the Cloud.* Manning Publications Co., 2018.

About the Author

Seetharaman Jeganathan, CISSP, has 17 years of experience in IT, security consulting, and project management. He focuses on information systems risk assessments, identity and access management, privileged account management, and cloud security consulting to his customers. He may be reached at seetharaman.jeganathan@gmail.com.