The Ultimate Guide of Orchestrating Security and DevOps
The Ultimate Guide of Orchestrating Security and DevOps

Organizations are transitioning from traditional software development to DevOps and Continuous Integration/Continuous Delivery (CI/CD) practices to reap the benefits of increased productivity and faster delivery. However, DevOps is changing the way we need to think about security.

While DevOps is not a technology, it is supported by a huge variety of tools that enable the transition to continuous integration and continuous delivery (CI/CD) – from container orchestration platforms like Kubernetes to CI/CD management tools like Jenkins. Gartner predicts that by 2022 more than 75% of organizations will be running containerized applications in production, which marks a significant increase from what we are witnessing today.

Why is DevOps security so different?

In the traditional, sequential “waterfall” software development model, security testing was happening only during the last stages of the process. This could result in several changes to close security gaps until the product complies with the various security recommendations imposed by regulations, policies, and standards. Fixing errors and closing security gaps at the end of the software development lifecycle made the development process longer and more expensive.

Implementing a DevOps model requires collaboration between teams throughout the software development lifecycle. Changes are an integral part of the development process, which results in producing secure products faster. Continuous Integration/Continuous Deployment (CI/CD) pipeline makes automation a critical part of DevOps, with being implemented at set intervals.

Security should be integrated into every stage of the software development lifecycle (SDLC) resulting in what is known as DevSecOps. DevSecOps is technology agnostic and organizations can use a combination of technologies, policies, and procedures to secure the DevOps pipeline. DevSecOps relies on collaboration between departments, who share the responsibility for establishing and enforcing security practices at every step of the SDLC. Development teams should ensure that their products are reliable, data is protected, and they must comply with regulatory and governance protocols.
DevOps security challenges

A variety of factors impact application security. However, security challenges in DevOps are often rooted in the “cultural conflict” between developers and security teams. Developers’ goal is to release their software products as quickly as possible, while security teams emphasize the elimination of security gaps, which can delay development.

DevOps hectic pace

Security teams are gasping to keep up with the hectic pace of DevOps. DevOps is about speed, with very short software development cycles. Reviewing code can take much longer than to develop it. As a result, security is often sacrificed, allowing misconfigurations, vulnerabilities, and other security gaps to remain, leaving the application open to breaches or malfunctions.

DevOps teams are negligent about security

A challenging barrier to overcome is the cultural resistance to security and testing. This stems from the developers’ stance that considers security as an inconvenience that disrupts and slows down the development process. However, the truth is that retroactive bug fixes will ultimately require more time and effort. Addressing and integrating security in the early phases of the pipeline reduces disastrous flaws and is worth the initial delay in the SDLC.

Inherent risks in development tools

DevOps often relies on cloud infrastructure, containers, microservices, and open-source components and tools. While these platforms and tools increase productivity, flexibility, and scalability, they also introduce potential security risks for DevOps environments. For example, the lack of visibility into containers makes it difficult to scan them for vulnerabilities.

Inadequate controls expand the attack surface

DevOps environments require robust controls for privileged access management. Developers and computing tools should leverage mechanisms like API access tokens and certificate-based credentials to establish and maintain code integrity, data confidentiality, and authenticity. Failure to adequately manage secrets, and weak access controls expand the corporate threat surface, providing attackers with the opportunity to exploit these vulnerabilities to steal or compromise data, disrupt operations, and gain control of the business IT infrastructure.
How to establish a DevSecOps environment

Organizations need to secure product development with DevSecOps capabilities. They have to integrate security into areas such as build automation, test automation, deployment automation, monitoring, environment management, and others. To achieve that they need to follow a modular route by assessing their existing DevOps and security strength and then orchestrating a tailored plan.

The plan should encompass data security checks for test automation, build automation, deployment, environment management, and monitoring. Further, they need to codify security, compliance, control, and governance requirements and then map them with their CI/CD pipeline. This will help their infrastructure to operate at a tightly screwed, desired security standards, at levels of SDLC. Finally, they need to ensure that security checks are automated and embedded at the server build level to cleanse the entire SDLC pipeline.

The following table summarizes the steps organizations should take to orchestrate security and DevOps processes. These steps are further analyzed in the following paragraphs.

<table>
<thead>
<tr>
<th>Assessment &amp; Implementation</th>
<th>Testing &amp; Analysis</th>
<th>Managed Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ DevOps &amp; Cloud Assessment</td>
<td>▪ Static Application Security Testing SAST</td>
<td>▪ Managed DAST</td>
</tr>
<tr>
<td>▪ CI/CD Plan &amp; Implementation</td>
<td>▪ Dynamic Application Security Testing DAST</td>
<td>▪ Managed SAST</td>
</tr>
<tr>
<td>▪ DevOps Security Training</td>
<td>▪ Interactive Application Security Testing</td>
<td>▪ Managed Vulnerability and Penetration Testing</td>
</tr>
<tr>
<td></td>
<td>▪ Application Composition Analysis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▪ Secure Code Reviews</td>
<td></td>
</tr>
</tbody>
</table>

Assessment and Implementation

DevOps and Cloud Assessment

Assessing and reviewing your existing architecture helps development and engineering teams validate the security-related design features of their application or service before starting the development phase. This allows organizations to identify and fix potential vulnerabilities before they can be exploited and before the fix requires a substantial re-engineering effort.

CI/CD Plan and Implementation

The next step is to produce an architecture and design that has considered all security risks and challenges to help improve the overall environment and the product’s security posture so that everyone can benefit from it. The cost and effort of retrofitting security after development is too high. Special attention must be given to the design approaches adapted for those areas that most commonly exhibit vulnerabilities.
DevOps Security Training

Developer security training is foundational to all the security tracks highlighted in this procedure. Without this training, together with experience and a security mindset, it will not be possible to do threat modeling, write accurate security user requirements, or evaluate Static Application Security (SAST) or Dynamic Application Security (DAST) Testing results. While most enterprises have their own SDLC curriculums or training courses, developers can benefit from free training provided by such organizations as SAFECode.

Testing and Analysis

Static Application Security Testing (SAST)

Static application security testing (SAST), also known as white-box testing, is an application testing methodology that analyzes source code to find security vulnerabilities that leave an organization’s applications open to exploitation and cyber-attacks. SAST scans an application before the code is compiled.

SAST takes place early in the software development life cycle (SDLC) since it does not require a working application or the execution of code. It helps developers identify vulnerabilities in the initial stages of development and quickly resolve security issues. SAST tools give developers real-time feedback as they code, preventing security-related issues from being considered an afterthought.

Dynamic Application Security Testing (DAST)

Dynamic Application Security Testing (DAST) is a security checking process that uses penetration tests on applications while they are running. DAST is a black-box security testing methodology that does not require a view into the source code. It essentially uses the same tactics and techniques that an attacker would use to find potential weaknesses. DAST testing looks for a broad range of vulnerabilities, including input/output validation issues that could leave an application vulnerable to cross-site scripting or SQL injection.

Interactive Application Security Testing (IAST)

Interactive Application Security Testing (IAST) analyzes code for security vulnerabilities while the application is run by an automated or manual test, or any activity “interacting” with the application’s functionality. This testing methodology reports vulnerabilities in real-time, without adding any extra time to the CI/CD pipeline.

IAST differs from both static analysis (SAST) and dynamic analysis (DAST) because it works inside the application. This type of testing tests only the part of the application that is exercised by the functional test. Many functional API tests are automated, making IAST a good fit for teams building in microservices, etc. Interactive application security testing solutions help organizations identify and manage security risks associated with vulnerabilities discovered in running web applications using runtime testing techniques.
Application Composition Analysis

Application Composition Analysis deals with managing open-source component use. Associated tools perform automated scans of an application’s codebase, including related artifacts such as containers and registries, to identify all open-source components, their license compliance data, and any security vulnerabilities. In addition to providing visibility into open-source use, some tools also help fix open-source vulnerabilities through prioritization and automated remediation.

Secure Code Reviews

A secure code review is a specialized task involving manual and/or automated review of an application's source code to identify security flaws in the code. A secure code review does not attempt to identify every issue in the code but instead looks to provide insight into what types of problems exist and to help the application developers understand the types of open issues. The goal is to arm the developers with information to help them make the application's source code more robust and secure.

Managed Services

If an organization’s security team lacks the resources or the skills to implement effectively all application testing methodologies, they may elect to use managed services offered by highly skilled and experienced security teams.

Managed DAST

Managed DAST enables organizations and development teams to implement on-demand dynamic analysis to systematically find and eliminate vulnerabilities in applications. Managed DAST services offer flexibility, scalability, consistency, and enablement to help security teams develop a remediation plan tailored to their risk environment.

Managed SAST

Managed SAST enables organizations to quickly and cost-effectively implement and scale static analysis to discover and eliminate security vulnerabilities in source code. Managed SAST provides on-demand expert static analysis for Java, C#, PHP, SQL, or Python based code to pinpoint, prioritize, and remediate vulnerabilities and other defects in the code.

Managed Vulnerability and Penetration Testing

This sort of testing uncovers more complex security flaws that may not be caught by DAST or SAST tools. Even when you have followed all the above steps, there may still be a possibility of a defect or vulnerability
making it to the final gate prior to going into production. To that end, penetration testing provides a final validation that secure code is going into production.

The penetration test assessment method is defined by the main attack vectors and test scenarios to be carried out based on the information provided by the requestors. The goal of the managed pen test is not to have comprehensive coverage of code scanning or finding all security defects that the technology has, but rather focus on areas where there may be higher business risks. For example, the scenarios may target to identify security defects that may expose critical data, compromise credentials, or have a potential reputational impact on the business. A pen tester will use various scanning tools, as well as manual testing, to identify security vulnerabilities that have a high business impact. The length of each pen test varies based on the scope size of the technology and the resources assigned.

How Ampcus Cyber can help you

DevOps is causing a major cultural change in security. Businesses need to embrace this change if they want to keep competitive and thrive in a shifting business environment. Security needs to be an ever thought and not an afterthought in DevOps. Failure to “bake” security into software lifecycle processes will result in producing insecure applications. Adversaries are always looking for the easiest way to break into corporate networks, and an app with security gaps will make their life easier.