Getting in front of the DevSecOps challenge: Strategies that work

About.gitlab.com | @gitlab
Problem: Even Basic Application Security Testing is Hard

➢ Applications are a prime target of cyber attacks
➢ Lack of hygiene allows proven exploits to be reused
➢ App Sec tools are expensive and require integration of both technology and processes
➢ To shift left, workflows must target both dev and sec teams
➢ Security and developer teams lack the means to collaborate and scale across silos
Traditional App Sec creates a silo and friction

Diagram:
- Repository
  - Code commit
  - CI
  - Merge request
  - CD
  - Deploy
  - Test environment
  - Security
    - DAST / IAST report
  - Developer
    - Dismiss or Create issue
Do you test all code changes?

Solve for the obvious cases.
Is Security a Square Peg in a Round Hole of DevSecOps?

Established security tools were intended for a waterfall process at the end of SDLC and are incongruent with DevSecOps’s
- People
- Process
- Technology
Problem: Modern Software vs Legacy App Sec

➢ The iterative development process (Agile/MVC) is incongruent with full app security scans

➢ Code changes faster and faster, with more open source, more APIs, and microservices (mini apps)

➢ DevSecOps doesn’t scale without developer enablement, automation, and exception-based security

➢ New attack surfaces of composable application infrastructure make network security less relevant.
DevOps Complexity inhibits auditability and introduces more security risk

Business Problems

Solution Deployment

- 100s of Tools
- Multiple Data Models
- Complexity & Risk
- Lack of Transparency
Each app sec product must integrate with multiple DevOps tools

IDE
- Visual Studio
- eclipse
- IntelliJ IDEA

BUG TRACKING
- Bugzilla
- Micro Focus
- Jira Software

BUILD SYSTEMS
- Jenkins
- TeamCity
- Bamboo
- Bitbucket
- Circle CI
- Travis CI

GRC
- modsecurity
- splunk
- RSA Archer GRC

SIEM
- WAF
What if you could...

Scan all code, every time
Seamlessly for dev
Using FEWER tools
With Dev, Sec, and Ops on the same page
And happy compliance auditors
“Your most important security product won’t be a security product.”

CISO of VMWare
Seamlessly test for vulnerabilities within the developer workflow
Continuous Application Security = a United Workflow
Make your security scanning as iterative as your development

- Actionable
- Accountable
- Iterative
- Contextual

Focus on the point of code commit
When vulnerabilities are present in an MR, you can easily see and triage them before the MR moves forward.

Enabling Developer To Address Security Findings
Automate, allowing security to focus on exceptions
Providing Visibility Into Security Risk

Quickly understand your at risk projects with Project Security Grades

Manage Security Risk Globally
Advantage of this approach

- **Contextual**
  - Within CI/CD dev workflow - accountable person
  - MR pipeline for dev
  - Security dashboard for Security

- **Congruent with DevOps processes**
  - Iterative within dev, tests every code change
  - Immediate cause/effect of code changes

- **Integrated with DevOps tools**
  - Create issues
  - Auto remediation
  - Production feedback

- **Efficient and automated**
  - Eliminate work wherever possible
  - No context-switching
  - Less tracking/triaging and more value-added security

Simplicity and Integration Wins!
All this cloud native stuff is secure... right?

Lessons from the Cryptojacking Attack at Tesla

https://redlock.io/blog/cryptojacking-tesla

Kubernetes console entry to AWS S3 storage bucket to sensitive data

Major Docker vulnerability disclosed in January

https://neuvector.com/docker-security/runc-docker-vulnerability/
Theft of more than 100 million customer records, 140,000 Social Security numbers
Lone insider
Gained access to the company’s AWS server by exploiting a misconfiguration in one of Capital One’s application firewalls.

Misconfiguration is a huge risk...and it will get worse!
Misconfigurations will be on par with failure to patch.

Cindy Blake
Cloud requires shared security accountability

Responsibility:
- **Cloud Provider**
- **Security Vendors**
- **End-user**
Cloud Native requires greater accountability

Orchestration (K8s): access/authentication, runtime resources, network policies

Containers: Images, Registry, East/West traffic

- Application Threat Detection
- Application Access
- App Configuration and App Patching

Virtualization and root access

- System Threat Detection
- System Patching and System Access

Network segmentation and Perimeter security

- Network Threat Detection

Responsibility:

- Cloud Provider
- Security Vendors
- End-user
Three main building blocks of cloud native architecture

**Containers**
Hold a cloud native application’s libraries and processes. They share one operating system. They make the applications portable.

**Orchestrators**
Direct how and where containers run.

**Microservices**
Apps are broken down into smaller parts, or microservices, to make them easier to scale based on load.
You need to proactively protect cloud-native environments by providing context-aware technologies to reduce your overall security risk.

This can be achieved by a natural extension of your existing operations practices providing security visibility across the entire DevSecOps lifecycle. This empowers your organization to apply DevSecOps best practices with visibility from the first line of code written all the way through monitoring and protecting your applications deployed into operations.
Container Network Security (CNS) →
Provides pod-specific firewall protection inspecting traffic entering and leaving Kubernetes pods as well as:

- Can be used to limit communication between pods and the internet when combined with the Ingress controller
- Can protect against rogue pods in the same cluster
- Can be especially useful for large, multi-tenant clusters
10 Steps Every CISO Should Take to Secure Next-Gen Software