DevSecOps
What Why and How?

Anant Shrivastava
@anantshri
NotSoSecure Global Services
About: Anant Shrivastava

Director NotSoSecure Global Services

Sysadmin / Development / Security

Trainer / Speaker: BlackHat, Nullcon, RootConf, RuxCon, IPEXpo, C0c0n

Project Owner: Android Tamer, Code Vigilant

Contributor: null, G4H, OWASP and more

https://anantshri.info (@anantshri on social platforms)
● What is DevSecOps?
● Why do we need DevSecOps?
● How do we do DevSecOps?
● Integrate Security in DevOps Pipeline
● Tools of Trade
● Sample Implementation (On Prem and Cloud Native)
● Case Studies
• I will be listing a lot of tools, It’s not an exhaustive list
• I don't endorse or recommend any specific tool / vendor
• Every environment is different: Test and validate before implementing any ideas
Effort to strive for “Secure by Default”
- Integrate Security via tools
- Create Security as Code culture
- Promote cross skilling
DevOps moves at rapid pace, traditional security just can’t keep up

DevSecOps makes it easier to manage rapid pace of development & large scale secure deployments

DevSecOps allows for much smoother scaling of process

Security as part of process is the only way to ensure safety
Shifting Left saves cost & time

Developer
Source Code Repository
Build
CI/CD Server
Staging/QA
Production
Monitoring

Suite of Security Test
Shifting Left saves cost & time

- Fewer Man Day Effort
- No New Deployments
- Automated Source Code Review

1 SQL Injection
- Fewer Man Day Effort
- No New Deployments
How do we do DevSecOps?

- DevSecOps is Automation + Cultural Changes
- Integrate security tools into your DevOps Pipeline
- Enable cultural changes to embrace DevSecOps
Injecting Sec in DevOps

Developer
- Pre-Commit Hooks
- IDE Plugins

Code Repository
- Secrets Management

Pre-Build
- Static Application Security Testing (SAST)
- Source Composition Analysis (SCA)

Post-Build
- Dynamic Application Security Testing (DAST)

QA/Staging
- Manual Web Application Pentesting
- Business Logic Flaws

Production
- Security in Iaas
- Compliance as Code
- Alerting & Monitoring

CI/CD Server
- Build Artifacts versioning against code commits

Artifact Repository
- Vulnerability Management
• Sensitive information such as the access keys, access tokens, SSH keys etc. are often erroneously leaked due to accidental git commits
• Pre-commit hooks can be installed on developer’s workstations to avoid the same
• Work on pure Regex-based approach for filtering sensitive data
• *If developers want they can circumvent this step hence use it like a defense in depth but don't fully rely on it*
IDE Plugin's provide quick actionable pointer to developers

It is useful to stop silly security blunders

Work on pure Regex-based approach

*If developers want they can circumvent this step hence use it like a defense in depth but don't fully rely on it*
• Often credentials are stored in config files
• Leakage can result in abuse scenario
• Secrets Management allows you to tokenize the information
● We don’t write software's, we build on frameworks
● Biggest portion of software is now third party libraries
● Major languages provide module managements
  ● PIP, NPM, Gems, go get, perl cpan, php packager and more
● Software Composition Analysis performs checks to identify vulnerable/outdated 3rd party libraries
● White-box security testing using automated tools
● Useful for weeding out low-hanging fruits like SQL Injection, Cross-Site Scripting, insecure libraries etc
● Tool by default configured with generic setting, needs manual oversight for managing false-positives
- Black/Grey-box security testing using automated tools
- SAST may not get full picture without application deployment
- DAST will help in picking out deployment specific issues
- Results from DAST and SAST can be compared to weed out false-positives
- Tools may need prior set of configuration settings to give good results
Infrastructure as a code allows you to document and version control the infrastructure.

- It also allows you to perform audit on the infrastructure.
- Docker / K8s infra relies on base images.
- Environment is as secure as the base image.
- Base images need to be minimal in nature and need to be assessed to identify inherited vulnerabilities.
Compliance could be industry standard (PCI DSS, HIPAA, SOX) or org specific

Compliance is essentially a set of rules and hence can be converted into written test cases

Having written code format this can again be version controlled
Vulnerability Management

- All the tools discussed above result in report fatigue
- Every tool has a different style of presentation
- A central dashboard is required to normalize the data
- Vulnerability Management System can then be integrated to bug tracking systems to allow devs to work on items
Alerting and Monitoring

- Monitoring is needed for two end goals
  - Understand if our security controls are effective
  - What and where we need to improve

- To test Security control effectiveness:
  - When did an attack occur
  - Was it blocked or not
  - What level of access was achieved
  - what data was bought in and bought out
● With recent advancements assets now should include anything and everything where organization data resides.

● With rapid development & provisioning the asset inventory can't be a static inventory.

● We need to monitor the assets constantly both on premise and Cloud.

Reference: https://redhuntlabs.com/blog/redifining-assets-a-modern-perspective.html
A simplistic flow of DevSecOps Pipeline incorporating various stages

https://www.youtube.com/watch?v=7ILrzYLpr14
# Tools of The Trade

<table>
<thead>
<tr>
<th>Threat Modelling Tools</th>
<th>Pre-Commit Hooks</th>
<th>Software Composition Analysis</th>
<th>Static Analysis Security Testing (SAST)</th>
<th>IDE Plugins</th>
<th>Secret Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThreatSpec</td>
<td>Talisman</td>
<td>DEPENDENCY-CHECK</td>
<td>Bandit</td>
<td>Skim</td>
<td>Vault</td>
</tr>
<tr>
<td>Microsoft Threat Modeling Tool</td>
<td>git-secret</td>
<td>Requires.io</td>
<td>RAPS</td>
<td>CAT.net</td>
<td>Keywhiz</td>
</tr>
<tr>
<td></td>
<td>truffleHog</td>
<td>Retire.js</td>
<td>sonarqube</td>
<td></td>
<td>Confidant</td>
</tr>
</tbody>
</table>

Preference given to open-source tools; we don’t endorse any tool.
Tools of The Trade

- Vulnerability Management
- Dynamic Analysis Security Testing (DAST)
- Security in Infrastructure as Code
- Compliance as Code
- WAF

Preference given to open-source tools; we don’t endorse any tool.
To be or Not to Be in Pipeline

- API / command line access
- Execution start to final output should be 15 minutes max
- Tools should be Containerizable / scriptable
- Minimal licensing limitations (parallel scans or threads)
- Output format parsable / machine readable (no to stdout, yes to json / xml)
- Configurable to counter false negatives / false positives
Pipeline Optimization

- Pipeline to be tweaked based on Milestone (Initiative/Epic/Story)
- Remember initial onboarding is tedious
- Ensure dataset dependent tool get frequent data refresh
- Sample optimization
  - Only CSS Changes: no need for SCA
  - Only pom.xml or gradle changes: no need of SAST
  - If Infra as code has zero changes skip or fast track infra scan
- Ensure to run full (non optimized) pipeline periodically
Different programming languages need different tools for static analysis and software composition analysis.

Some tools support multiple languages like SonarQube.

Others are focused on one language.
Preference given to open-source tools; we don’t endorse any tool
DevSecOps Lab - PHP

Jenkins

SensioLabs Security Checker

phpcs-security-audit v2
DevSecOps Lab - Python
The Threat Landscape changes
- Identity and Access Management
- Asset Inventory
- Billing

Infrastructure as Code allows quick audit / linting

Focus more on:
- Security groups
- Permissions to resources
- Rogue / shadow admins
- Forgotten resources (compromises / billing)
Different Service Providers Approach Security Differently
All of them provide some of the ingredient in-house
Irrespective of Cloud provider some tools will still need to be sourced
- Static Code Analysis Tool
- Dynamic Code Analysis Tool
- Software Composition Analysis
- Vulnerability Management Tool
<table>
<thead>
<tr>
<th>Source Code Management</th>
<th>Conventional Infra</th>
<th>AWS</th>
<th>Azure</th>
<th>GCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitbucket, Github, Gitlab etc..</td>
<td>AWS CloudCommit</td>
<td>Azure Repos</td>
<td>Cloud Source Repositories</td>
<td></td>
</tr>
<tr>
<td>Infrastructure As a Code</td>
<td>Chef, Puppet, Ansible more..</td>
<td>Amazon CloudFormation</td>
<td>Azure DevTest Labs</td>
<td>Cloud Code</td>
</tr>
<tr>
<td>CI/CD Server</td>
<td>Jenkins, Bamboo, Gitlab, Travis CI, Circleci more</td>
<td>AWS CodeBuild AWS CodeDeploy AWS CodePipeline</td>
<td>Azure Pipelines, Azure Test Plans</td>
<td>Cloud Build, Tekton</td>
</tr>
<tr>
<td>Artifactory Repository</td>
<td>jFrog Artifactory, Sonatype Nexus, more..</td>
<td>Amazon S3</td>
<td>Azure Artifacts</td>
<td>Cloud Firestore</td>
</tr>
<tr>
<td>Stg/Prod Servers</td>
<td>VMWare, On-premises servers</td>
<td>EC2 ECS (Elastic Containers) EKS (Elastic Kubernetes)</td>
<td>Virtual Machines, Azure Lab Services, Azure Kubernetes Service (AKS)</td>
<td>Compute Engine, App Engine, Shielded VMs</td>
</tr>
<tr>
<td>Monitoring &amp; Alert</td>
<td>Nagios, Graphite, Grafana</td>
<td>AWS CloudWatch</td>
<td>Azure Monitor, Network Watcher</td>
<td>Access Transparency</td>
</tr>
<tr>
<td>Firewall</td>
<td>Modsecurity</td>
<td>AWS Firewall Manager, AWS WAF</td>
<td>Azure Firewall</td>
<td>Application Gateway</td>
</tr>
<tr>
<td>DLP</td>
<td>MyDLP, OpenDLP</td>
<td>Amazon Macie</td>
<td>Azure Information Protection</td>
<td>Cloud Data Loss Prevention</td>
</tr>
<tr>
<td>Threat Detection</td>
<td>Snort, Kismet</td>
<td>Amazon GuardDuty</td>
<td>Azure Advanced Threat Protection</td>
<td>Event Threat Detection (beta)</td>
</tr>
<tr>
<td>Vulnerability Scanning</td>
<td>OpenVAS, Nessus</td>
<td>Amazon Inspector</td>
<td>Azure Security Center</td>
<td>Cloud Security Scanner</td>
</tr>
<tr>
<td>Secrets Management</td>
<td>Hashicorp Vault, Docker Secrets</td>
<td>AWS Secrets Manager</td>
<td>Azure Key Vault</td>
<td>Secrets management</td>
</tr>
</tbody>
</table>
Cultural Aspect

- Automation alone will not solve the problems
- Encourage security mindset especially if outside sec team
- Cultivate/Identify common goals for greater good
- Build allies (security champions) in company
- Focus on collaboration and inclusive culture
- Avoid Blame Game

Security team should try to eliminate the need of dedicated security team
• Bridge between Dev, Sec and Ops teams
• Single Person per team
• Everyone provided with similar cross skilling opportunities
• Incentivize other teams to collaborate with Sec team
  ○ Internal Bug bounties
  ○ Sponsor Interactions (Parties / get-togethers)
  ○ Sponsor cross skilling trainings for other teams
## Security Enablers

### People
- Build relationships between teams, don’t isolate
- Identify, nurture security conscious individuals
- Empower Dev / ops to deliver better and faster and secure, instead of blocking.
- Focus on solutions instead of blaming

### Process
- Involve security from get-go (design or ideation phase)
- Fix by priority, don’t attempt to fix it all
- Security Controls must be programmable and automated wherever possible
- DevSecOps Feedback process must be smooth and governed

### Technology
- Templatize scripts/tools per language/platform
- Adopt security to devops flow don’t expect others to adopt security
- Keep an eye out for simpler and better options and be pragmatic to test and use new tools
Generic Case Study

DevOps

Manual Pentest
Zero Day
Zero Day Resolved

May | June | July | August

- Struts
- ShellShock

DevSecOps

May | June | July | August

- Struts
- ShellShock
DevSecOps @ Fannie Mae – The Strategy

Integrate with Culture
- Run as ONE (Security + DevOps as a single purpose team)
- Training development teams to develop secure code
  - OWASP Brown Bags and On Demand Training Courses
  - Secure Code Examples in GIT REPO show how to write secure code
- Empowering Developers/Engaging Business Partners
  - Verification of Fortify “Clean Scans”
  - Periodic “To-the-Right” Application Static and Dynamic Tests

Make Security Easy
- Tracking security issues in the same systems developers are using
  - Integrated Fortify with SonarQube
  - Integrated Fortify with SSC
  - Application Security Issues Defect Tracking (Jira)
- Integrating preventive security controls/tools in the development phase
  - HP-Secure Assist
  - Find Security Bugs
  - Sonatype IQ Plugin

Automate Everything
- Automating as many security tests as possible to run alongside other tests
  - Integrating SAST tools (HP-SA, Find Bugs, Find Security Bugs, Fortify)
  - Future: Use DAST tool
- Detecting when applications are relying on libraries that have known vulnerabilities
  - Integrating Sonatype with fortify to detect third party libraries that have known vulnerabilities

Case Studies – Fannie Mae

DevSecOps @ Fannie Mae – The Results

Delivering the Promise

- Average days to close a vulnerability improved by 74%
- Automated code quality scanning shows overall security code scores has increased by 10%
- More than 60% of application teams are performing security tests before release
- Critically vulnerable open source components (CVE 7.5+) downloaded has decreased from 18% to 6.25%
- ~ 55% of technical debt and security defects identified as a result of periodic testing have been dispositioned
- ~ 77% of older technical debt and security defects have been remediated, have a remediation plan in place, or have been addressed through managed retirements of assets

Average Days to Close a Security Vulnerability

Case Studies – ABN Amro

https://www.slideshare.net/derweeksglobal/abn-amro-devsecops-journey
Case Studies – ABN Amro

- Test environment uptime improved
- Improved code quality & secure coding
- Improved cooperation across stakeholders
- Improved time to market
- Improved development processes
- Increased velocity

From 4 Internet Banking releases to 18 releases per year

Core review times have been shortened and violations when merging are being prevented

We never thought it would be possible to develop, test and deploy something completely in one sprint

Private Banking International team reduced build from 5 hours to 5 minutes

First continuous deployment realised by identity access mgmt team

I-Markets doubled velocity after 1 sprint containing CICD improvements only

Changes are being rolled out as soon as they are available

Release times halved for teams using XL Release

x3 deployments to UT +20% successful Builds -100% Package creation time -75% Testing time x2,5 deployments to ET

https://www.slideshare.net/derweeksglobal/abn-amro-devsecops-journey
Negative Case Studies

Cloud Assets Misconfiguration

# Top defense contractor Booz Allen Hamilton leaks 60,000 files, including employee security credentials and passwords to a US government system.
# Verizon partner leaks personal records of over 14 million Verizon customers, including names, addresses, account details, and for some victims — account PINs.
# An AWS S3 server leaked the personal details of WWE fans who registered on the company's sites. 3,065,805 users were exposed.
# Another AWS S3 bucket leaked the personal details of over 10 million American voters. The database contained information from three data mining companies known to be associated with the Republican Party.
# Another S3 database left exposed only contained personal details of job applications that had Top Secret government clearance.
# Dow Jones, the parent company of the Wall Street Journal, leaked the personal details of 2.2 million customers.
# Omaha-based voting machine firm Election Systems & Software (ES&S) left a database exposed online that contained the personal records of 1.8 million Chicago voters.
# Security researchers discovered a Verizon AWS S3 bucket containing over 100 MB of data about the company's internal system named Distributed Vision Services (DVS), used for billing operations.
# An auto-tracking company leaked over a half of a million records with logins/passwords, emails, VIN (vehicle identification number), IMEI numbers of GPS devices and other data that is collected on their devices, customers and auto dealerships.

Prevention: Continuous monitoring and review of cloud assets and config
• Rite of passage by periodic pen test and continuous bug bounty

• It's not just important to get feedback but to also action on them

• Risk Acceptance Documentation should be the worst case scenario, not your first bet
Did we secure the security controls

**DevSecOops**: If attacker controls security tools / build chain it has limitless power

Ensure the same practice is followed back again for these tools

Security role doesn't mean you get to circumvent the rules

Follow basic security hygiene we always keep talking about
  * Secure configuration
  * Patching Policy
References


• https://snyk.io/opensourcesecurity-2019/

• https://scaling-threat-detection.awssecworkshops.com/

Security is everyone's responsibility.

Embrace security as an integral part of the process, use feedback to refine the process.

DevSecOps is not a one size fits all: your mileage will vary.