DevOps Anti-Patterns

Time to fire the Ops team!

Have the Ops team deal with it.

Let’s hire a DevOps unit!
Anti-Pattern: “Throw it Over the Wall”

Development

Operations
Anti-Pattern: “DevOps Team Silo”
Anti-Pattern: “NoOps” Approach
Anti-Pattern: “Ops Will Handle it”
Anti-Pattern: “Ops Will Handle it”
Development and Operations Collaboration
Dev and Ops Fully Shared Responsibilities
DevOps-as-a-Service

- Development
- Operations
- DevOps
Collaboration Across Teams
Cross-Training
Support Business Agility
Breaking Down Silos
Automation of Repetitive Tasks
“Everything” as Code
Test, Measure, and Monitor

Process

People

Tools
We want to turn this...
Into this!
“DevSecOps is the process of incorporating and enforcing meaningful security controls without slowing down deployment velocity.”
Enabling DevSecOps Through People
Break Down the Silos
Build a Team Beyond Yourself
Be Approachable
Train Others
Radical Transparency
Communication and Collaboration

- Critical piece to the DevOps puzzle
- A culture of trust and empowerment makes for a healthy workplace
- Move towards shipping software faster and more confidently
- Embrace cross-team communication and training
- Feedback available from each step of the pipeline
- Security is a great fit in modern DevOps cultures
Case Study: The "Two Pizza" Team
Enabling DevSecOps Through *Process*
DevSecOps Pipelines
DevOps Processes

- Automate **building** the dev and production environment
- Automate software **testing** (including security)
- Automate **deploying** software and services
- Automate **monitoring** and **alerting**
- **Tune** your tools to become more automated and hands-off
- Build the pipeline **slowly** and don’t fear failure!
- Be careful with **sensitive areas** which are difficult to automate (access control, biz logic, complex actions)
60% OF THE TIME

DEPLOYMENTS WORK EVERYTIME!
Key Goals of DevSecOps Pipelines

- Optimize the critical resource: Security personnel
- Automate things that don’t require a human brain
- Drive up consistency
- Increase tracking of work status
- Increase flow through the system
- Increase visibility and metrics
- Reduce any dev team friction with application security
Pipeline Security

Continuous Integration

Code Committed

Automated test suite

Peer review

Repository

Continuous Deployment

Infrastructure QA Testing (performance, load, etc.)

Build Automation

Configuration management, artifact creation, db migrations, etc.

Production

Post-deploy checks

IaaS, PaaS, On-Prem, etc.

Monitoring and alerting
Development (Pre-Commit)

- Developer laptops are the first line of defense in a DevSecOps pipeline
- Moving security to the left prevents costly mistakes and vulnerabilities later
- Required Git pre-commit hooks can offer a simple, effective feedback loop
  - Static analysis scans in the IDE
  - Peer review from security engineers
  - Lightweight, threat modeling in sensitive areas
Git-Secrets

https://github.com/awslabs/git-secrets
Brakeman Static Scanning (Git Pre-Commit Hook)
Continuous Integration (Commit Stage)

- Basic automated testing is performed after a commit is made
- Must be quick and offer instant feedback
- Key place to include security checks that run in parallel with integration tests, unit tests, etc.
  - Identify risk in third-party components
  - *Incremental* static security scanning
  - Alerting on changes to high-risk areas
  - Digital signatures for binaries
Continuous Integration (Commit Stage)

- CI server may include a dedicated security worker
- Third-party dependency checking performed in CI
  - OWASP Dependency Check
  - Node Security Project
  - Bundler-Audit
  - SRC:CLR
- Custom alerts set on repositories and sent to “on-call” security teams
  - Is someone changing pw hashing algorithm?
  - Is a new password policy enabled?
Continuous Deployment (Acceptance)

- Triggered by successful commit and passing build
- Utilize parallel, out-of-band processes for heavyweight security tasks
- IaaS and Config Management should provision latest, known-good environment state (as close to production as possible)
- Security checks during acceptance:
  - Comprehensive fuzzing
  - Dynamic Scanning (DAST)
  - Deep static analysis
  - Manual security testing
Continuous Deployment (Acceptance)

- Zap Baseline scan incorporated into CI stage of the deployment pipeline
- Runs a basic scan scan from a simple Docker run command
- By default will output all results of passive scan rules
- Highly configurable but still struggles in certain areas

https://github.com/zaproxy/community-scripts/tree/master/api/mass-baseline
Production (Post-Deployment)

- After all security checks have passed and deployment is complete
- Security teams’ job does not stop here:
  - Monitoring and Alerting
  - Runtime Defense (RASP)
  - Red Teaming
  - Bug Bounties
  - External Assessments
  - Web Application Firewalls
  - Vulnerability Management

Post-deploy checks

IaaS, PaaS, On-Prem, etc.
Enabling DevSecOps Through Technology
Where are we going?

- Infrastructure as a Service
- Secrets Storage
- Logging and Monitoring
- Containers and Microservices
Infrastructure-as-a-Service (IaaS)

- Delivery of a complete computing foundation
  - Servers (virtualized, physical, or “serverless”)
  - Network
  - Storage
- Infrastructure is exposed to operators using a service
  - Programming network and infrastructure through APIs vs. buying and building physical hardware
- Can be operated by a third-party, hosted in-house (K8s), or a hybrid model
Infrastructure-as-a-Service

Managed by Ops

IaaS Provider

Network Hardware

Auto Scaling
Amazon DynamoDB
Amazon EC2

Physical VM Hosts
IaaS Security Considerations

- “The Cloud” doesn’t *do* security for you – this is your responsibility
- Network and Data Security
- Auditing Capabilities
- Compliance Requirements
  - SOC, PCI, HIPAA, etc.
- Encryption Capabilities
- Third-Party Certificates and Audits
- Secrets Storage, Built-in Security Features, etc.
The Cloud Won’t Protect You

Security
Dow Jones index – of customers, not prices – leaks from AWS repo
S3 bucket was set to authenticate all AWS users, not just Dow Jones users

Verizon confirmed on Wednesday the personal data of 6 million customers has leaked online.

The security issue, uncovered by research from cybersecurity firm UpGuard, was caused by a misconfigured security setting on a cloud server due to "human error."

The error made customer phone numbers, names, and some PIN codes publicly available online. PIN codes are used to confirm the identity of people who call for customer service.

198 million Americans hit by 'largest ever' voter records leak

Personal data on 198 million voters, including analytics data that suggests who a person is likely to vote for and why, was stored on an unsecured Amazon server.
Distributing Secrets

- Software systems often need access to a shared credential to operate:
  - Database password
  - Third-Party API key
  - Microservices

- Secret management is full of opinions and could be a course itself

- Many options exist – Choose your own adventure!
Commandments of Sane Secret Management

- Secrets should not be written to disk in cleartext
- Secrets should not be transmitted in cleartext
- Access to secrets should be recorded
- Operator access to secrets should be limited
- Access control to secrets should be granular
- Secrets distribution infrastructure should be mutually authenticated
- Secrets should be version-controlled
HashiCorp’s Vault
Which is the most secure way to pass secrets to an app running in a container?

1. Pass secrets as an environment variable
2. Mount volume in container that has secrets in a file
3. Build the secrets into the container image
4. Query a "Secrets API" over your network
5. Other