SANS DevOps Survey: Sneak Peek
Introduction

Frank Kim

• ThinkSec
  – Founder

• SANS Institute
  – Former CISO
  – Curriculum Lead
    ▪ Management and Application Security
  – Author & Instructor
    ▪ DEV540, DEV541
    ▪ MGT512, MGT514
Secure DevOps: Fact or Fiction?
- Sixth in a series of annual surveys on security in software development
- First to explicitly focus on DevOps
- Written by Jim Bird and Barb Filkens

Survey topics
- Previous survey covered how organizations balance speed of delivery against risk
- Current survey covers how security fits into DevOps
  - Where risks are found
  - How risks are managed
  - Success factors implementing SecDevOps
#1 DevOps Adoption
What percentage of your applications are cloud, container, or serverless?

- **Cloud**: 85%
- **Container**: 72%
- **Serverless**: 54%
Common security controls are applied to each trust boundary in the monolith architecture:

1. Security Controls
   - Web Application Firewall
   - HTTPS, Rate Limiting

2. Security Controls
   - Authentication, Authorization
   - Access control, Validation

3. Security Controls
   - System Authentication, TLS
   - Encryption at rest
Microservice Architecture

- How does this change in a microservice architecture?
Microservice Architecture Attack Surface

- Consider the attack surface in a modern microservice architecture:
• Delivery speed
  – 66% deploy changes more than once per month (vs. 60%)
  – 10% deploy changes on a continuous basis (vs. 5%)
• DevOps cycles through five key phases

**DevOps Pipeline**

- **PRE-COMMIT**
  - Activities before code is checked in to version control

- **COMMIT**
  - Automated build and Continuous Integration (CI) steps

- **ACCEPTANCE**
  - Automated acceptance and functional testing with Continuous Delivery (CD)

- **PRODUCTION**
  - Steps before, during, and after code is deployed to production

- **OPERATIONS**
  - Continuous monitoring, testing, audit, and compliance checks
DevOps Security Tools and Processes

- **PRE-COMMIT**
  - Threat Modeling
  - IDE Security Plugins
  - Pre-Commit Hooks
  - Peer Code Review

- **COMMIT**
  - Static Code Analysis
  - Security Unit Tests
  - Dependency Management
  - Container Security

- **ACCEPTANCE**
  - Dynamic Security Tests
  - Acceptance Tests
  - Infrastructure as Code
  - Config Management

- **PRODUCTION**
  - Security Smoke Tests
  - Security Configuration
  - Secrets Management
  - Server Hardening

- **OPERATIONS**
  - Blameless Postmortems
  - Continuous Monitoring
  - Penetration Testing
  - Threat Intelligence
• Testing speed
  - 24% test critical applications more than once per month (vs. 13%)
  - 25% test on a continuous basis (more than double the 12% from last year)
#2 Shifting Left
• Need to “Shift Left”
  – Move security earlier into the SDLC
  – Less than half (46%) include security in early phases of Inception and Requirements
SecDevOps Pre-Commit Phase

- Apply security controls before code is written and checked in

<table>
<thead>
<tr>
<th>Pre-Commit</th>
<th>Commit</th>
<th>Acceptance</th>
<th>Production</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDE Security Plugins</td>
<td>Security Unit Tests</td>
<td>Acceptance Tests</td>
<td>Security Configuration</td>
<td>Continuous Monitoring</td>
</tr>
<tr>
<td>Pre-Commit Hooks</td>
<td>Dependency Management</td>
<td>Infrastructure as Code</td>
<td>Secrets Management</td>
<td>Penetration Testing</td>
</tr>
<tr>
<td>Peer Code Review</td>
<td>Container Security</td>
<td>Config Management</td>
<td>Server Hardening</td>
<td>Threat Intelligence</td>
</tr>
</tbody>
</table>
Threat Modeling in DevOps

Iterative and lightweight threat modeling based on risk: early in design, or as major changes are made

Examine trust boundaries and assumptions in architecture

Ask these questions when you are making changes:

1. Are you changing the attack surface (new entry/exit points, new user role...)?
2. Are you changing the technology stack or application security controls?
3. Are you adding confidential/sensitive data?
4. Have threat agents changed – are we facing new risks?
Weaponizing the toolchain:

- **OWASP User Security Stories**
  - https://github.com/OWASP/user-security-stories

- **OWASP Application Security Verification Standards**

- **Mozilla's Rapid Risk Assessment (RRA)**

- **OWASP Threat Dragon**
Threat Modeling Example

Mozilla's rapid risk assessment guidance and Google Doc provide a blueprint for 30 minute RRAs:

RRA for <service name>

<table>
<thead>
<tr>
<th>Service Owner(s)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner’s Director</td>
<td></td>
</tr>
<tr>
<td>Service Data Classification</td>
<td></td>
</tr>
<tr>
<td>Highest Risk Impact</td>
<td></td>
</tr>
</tbody>
</table>

Service Notes

How does the service work? Do we have diagrams, demos, examples? Is the service in production yet? Can we break this service down per components?

RRA Request bug: Vendor questionnaire (if vendor):
IDE Security Plugins

Immediate, incremental scanning in each developer’s IDE catches security mistakes as code is being changed/saved by the developer

• Security becomes part of the engineering workflow
• Shifting as far left as possible in the kill chain
• Must have low false positive rates (important)
• Run high value rules and disable noisy rules that distract engineers
Weaponizing the toolchain:

- **FindSecurityBugs** plugin for Eclipse and IntelliJ
  - http://find-sec-bugs.github.io/
- **Puma Scan** plugin for Visual Studio
  - https://github.com/pumasecurity/puma-scan
- Microsoft’s **DevSkim** for VSCode, Sublime, Visual Studio
  - https://github.com/Microsoft/DevSkim
- **SonarLint** plugins for Visual Studio, IntelliJ, and Eclipse
  - https://www.sonarlint.org/

Note: IDE plugins are also available for most commercial SAST products
Secure Code Spell Checker
Pre-Commit Hooks

- Git Hooks automatically run scripts at different points in workflows
  - Local: **pre-commit**, prepare-commit, commit, post-commit, post-checkout, pre-rebase
  - Server-side: **pre-receive**, update, **post-receive**
- Implement team-wide workflow policies, or check code for problems
- **CAUTION:** Repo owner can alter/uninstall hooks – so hooks cannot be enforced
Weaponizing the toolchain:

• Open source frameworks to manage hooks for different languages + tools
  • Yelp pre-commit framework
  • Overcommit

• Pre-commit tools for scanning code:
  • AWS Labs git-secrets (https://github.com/awslabs/git-secrets)
  • Talisman (https://github.com/thoughtworks/talisman)
  • Auth0 repo-supervisor (https://github.com/auth0/repo-supervisor)
Pre-Commit Hook Example

AWS git-secrets blocking a commit that contains an access key and secret key id:

```bash
$ git commit -m "testing git-secrets"
Web/Licensing/appsettings.json:5:
  "AccessKey": "AKIAJNQ7C2FCRR6B4VWA",
Web/Licensing/appsettings.json:6:
  "SecretKey": "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"
[ERROR] Matched one or more prohibited patterns
```
#3  Moving Forward
• Marginal increase in ability to remediate in a satisfactory and timely manner
  – 51% vs. 50% repair more than 75% of vulnerabilities
  – Consistent 70% repair more than 50% of vulnerabilities
What are Your Top Three Challenges Implementing Secure DevOps?

- Major Secure DevOps Challenges
  - Organizational not technical
What Types of Apps Do You Protect Now and Expect to in Twelve Months?

- **Legacy apps**
  - Take majority of security team’s time and attention

- **Modern apps**
  - Attack surface of the future
Legacy Systems and Blast Radius

Where is the blast radius in this example?
What are the Top Three Factors That Have Contributed to Your Success?

- SecDevOps success relies on soft skills
  - Developing “security champions”
  - Management buy-in
  - Cross-functional teams
  - Improving communications
  - Sharing goals
Moving Forward with SecDevOps?

- What will happen when security does the following?

<table>
<thead>
<tr>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide the development team with a secure coding standard</td>
</tr>
<tr>
<td>Identify secure coding training that developers should attend</td>
</tr>
<tr>
<td>Provide budget for a dedicated resource to the development team to implement a secure SDLC</td>
</tr>
<tr>
<td>Supply working code that fixes the vulnerabilities</td>
</tr>
</tbody>
</table>
“Implementing a SecDevOps program allows us to deploy up to 25 times per day, reduces lead time for security changes and mean time to repair (MTTR) to one hour, and reduces change failure rate to 12%”
“By implementing a digital transformation, we performed 165 experiments in the peak tax season resulting in an increased conversion rate of 50% and an increase in revenue of $120 million.”

• Example modified from DORA research paper
  – Forecasting the Value of DevOps Transformations: Measuring ROI of DevOps
  – https://devops-research.com/roi
## Business Case Options

- Highlight trade-offs with business value, risk reduction, cost

<table>
<thead>
<tr>
<th></th>
<th>Option A</th>
<th>Option B</th>
<th>Option C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business value</td>
<td>✔</td>
<td>✔ ✔</td>
<td>✔ ✔ ✔</td>
</tr>
<tr>
<td>Risk reduction</td>
<td>✖️</td>
<td>✖️</td>
<td>✖️</td>
</tr>
<tr>
<td>Cost</td>
<td>$</td>
<td>$ $</td>
<td>$ $ $</td>
</tr>
</tbody>
</table>
In Summary

• Automate Everything
  – Cloud, containers, and serverless are here
  – Need to speed up security activities to keep up with delivery speed

• Shift Left
  – Challenge injecting security into the SDLC
  – Need to embed security activities earlier in the DevOps pipeline

• Focus on outcomes
  – Difficult to balance technical and organizational activities
  – Focus on the benefit to the business, not the technology itself
Thank you for attending!