Security and Compliance Theater
“The Seventh Deadly Disease”
John Willis
@botchagalupe
botchagalupe@gmail.com
DEVOPS AUTOMATED GOVERNANCE REFERENCE ARCHITECTURE

Attestation of the Integrity of Assets in the Delivery Pipeline
You can’t Lean, Agile, SAFE, Devops or even SRE your way around a bad organizational culture.
Organizational Anthropology
“What Abraham Wald found was a logical error known as Survival Bias”
Complexity

system A

system B
HTTP Request with curl containing Content-Type:

curl http://127.0.0.1:8900/struts2-showcase

Former Equifax employee who 

The company is still 

The Equifax Data Breach

Majority Staff Report
115th Congress

December 2018
Conway’s Law

An adage named after computer programmer Melvin Conway, who introduced the idea in 1967. It states that "organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations."
Richard Smith
Chief Executive Officer

David Webb
Chief Information Officer

John Kelley
Chief Legal Officer

Graeme Payne
SVP & CIO for Global Corporate Platforms

Susan Mauldin
Chief Security Officer
Webb asked Mauldin whether she would support moving the CSO back under the CIO. Webb testified:

A. I actually did have a conversation one time with Susan Mauldin about whether she thought it was a better option.

Q. And what was her response?

A. I think she was comfortable with where it was.

Mauldin testified about her knowledge of the origin of the particular organizational structure. She stated:

[T]hat structure was in place . . . at the time I arrived at Equifax. It was the structure that was there with the person that was my predecessor. And I knew that it was that structure going in. I didn’t question it. I was okay with it. And so it was just what was there, and so it continued with what it had been.
Based on information confirmed on July 31 by the lead forensic analyst, Mauldin stated “I felt like I knew at that point that PII had been involved in this incident.”\textsuperscript{247} She reported this to John Kelley on July 31, but did not inform David Webb.\textsuperscript{248} Mauldin testified:

Q. Is there any particular reason why you did not report to the CIO your belief that PII may have been exfiltrated in connection with the security incident we have been discussing?

A. I don’t remember a particular reason about that . . . . I just don’t remember thinking about that.\textsuperscript{249}
Capital One Data Breach Compromises Data of Over 100 Million

Organizational Anthropology

- 10 to 20 Pre-Assessment Calls
- 30 to 50 Assessment Meetings
- Interview 150-200 People
- Over 400 Pages of Notes
- 300 Summarized Observations
Two Days With Leadership

What are the Five things XYZ Corp is not doing that you should be doing?

<table>
<thead>
<tr>
<th>PARTICIPANT</th>
<th>UNITS SOLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>15</td>
</tr>
<tr>
<td>Capacity</td>
<td>14</td>
</tr>
<tr>
<td>Direction</td>
<td>11</td>
</tr>
<tr>
<td>Consistency</td>
<td>9</td>
</tr>
<tr>
<td>Transparency</td>
<td>8</td>
</tr>
<tr>
<td>Empowerment</td>
<td>6</td>
</tr>
</tbody>
</table>

Top Categories:

- Communication: 15
- Capacity: 14
- Direction: 11
- Consistency: 9
- Transparency: 8
- Empowerment: 6

Percentage:

- Communication: 24%
- Capacity: 22%
- Direction: 17%
- Consistency: 14%
- Transparency: 13%
Common Top Three

• Toil
• Risk
• Inconsistency
General Toil

- Downstream Dependancies
- ITIL Processing Toil
- A Lot of Signoff’s
- Only High Priority Get Fixed
- High Technical Debt
- Product and Team Silos
General Risk

- Permitter Based Risk Models
- Subjective Governance Models
- Inconsistent Policies for Dev/Test/QA
- Low Attestation Efficacy
- Configuration Blind Spots
General Consistency

• Inconsistent Environments
• Unclear Roles and Responsibilities
• CD Anti-Patterns
• Cross Function Chaos
The Deadliest Disease!
Security and Compliance Theater
Work, Toil, Misalignment, Knowledge, Org Design, Complexity
Devops (Shift Left Auditors)

- Review Boards (ARB, PRB, CAB)
- Check Box Compliance
- Workarounds and Hidden Work
- Auditor Workarounds
- Vulnerability Theater
- Negative Risk RIO
- Policy Theater
**DevSecOps**

- **Preventative**
  - Requirements & Design
    - Application Risk Classification
    - Security Requirement Definition
    - Threat modeling
  - Development
    - Static Analysis/IDE
    - Secure Libraries
    - Secure Coding Standards
  - CI
    - SCM
    - Static Analysis (CI)
    - Open Source Governance (CI)
    - Container Security Compliance (CI)
  - Interval Trigger Assessment
    - Dynamic Assessments
    - Threat-Based Pen Test
  - Production
    - Perimeter Assessment
    - Web Application Firewalls
    - Automated Attack/Bot Defense
    - Container Security Management

**Security Mavens (Security-Trained Developers and Operations)**

**Role Based Software Security Training**

**Continuous Monitoring, Analytics and KPI Gathering**
DevSecOps Operational Tips

• Work with and educate your auditors
• Move Subjective Attestation to Objective Attestation
• Ruthlessly eliminate false positives to Developers
• Explain the vulnerabilities in business impact terms
• Devops the vulnerability (JIRA, backlog, Kanban)
• Open the code base to everyone in the organization
• Educate on how to fix
Changing subjective attestation into objective attestation
Devops Automated Governance

• Attestation of the integrity of assets in the delivery pipeline
  
  • Automated Attestation in CI/CD
  • Transform CAB (Change Advisory Board)
  • Reduce Effort w/ Compliance Activities - “Continuous Compliance”
Creating Better Pipelines

So how do we design, measure, and improve our pipelines to avoid the above?

Pipeline Design

At Capital One, we design pipelines using the concept of the “16 Gates”. These are our guiding design principles and they are:

- Source code version control
- Optimum branching strategy
- Static analysis
- >80% code coverage
- Vulnerability scan
- Open source scan
- Artifact version control
- Auto provisioning
- Immutable servers
- Integration testing
- Performance testing
- Build deploy testing automated for every commit
- Automated rollback
- Automated change order
- Zero downtime release
- Feature toggle

These gates are used to understand each and every product’s progress through the DevOps process.
The Delivery Pipeline

Source Code Repo → Build → Package → Non Prod Deploy → Prod Deploy

Dependency Mgmt ← Artifact Repo

Figure 1: Delivery Pipeline
Constructing an Attestation
Attestation Database

Grafeas

Universal artifact metadata

Store, query, and retrieve critical metadata about all of your software artifacts, regardless of their type and where they are stored. Get 360-degree visibility across a variety of environments, including on-premises, private, and public cloud clusters.

An open artifact metadata API to audit and govern your software supply chain
Basic Governance Model

![Diagram: Basic Governance Model]

Figure 3: Basic Governance Model
Source Code Repository Stage

**Risks**
1. Unapproved changes
2. Untested changes
3. Unapproved 3rd party dependency
4. Information (secrets) leakage
5. Low quality code sent to production

**Controls**
1. Peer review
2. Unit test coverage
3. Clean dependency
4. Scan for sensitive information
5. Static code analysis/lintering

**Aactors**
1. Code author
2. Code reviewer

**Actions**
1. Commit
2. Change request (Pull request, merge request)
3. Review, merge

I/P
Request for Change

O/P
New Version
Build Stage

Risks
1. Inaccurate, unapproved build configuration
2. Build information is missing, modified, or inconsistent
3. Unapproved 3rd party dependency
4. Build output is untested
5. Build output is low quality
6. Build output has security vulnerability

Controls
1. Build configuration in Source Control: peer reviewed
2. Immutable build and O/P
3. Upstream approved dependency store
4. Unit test
5. Linting
6. Static security analysis

I/P
1. New version of code
2. Dependencies
3. Build definition

Build

O/P
1. Artifacts
2. Build log
3. BOM

Actors
1. System

Actions
1. Build
Dependency Management Stage

**Risks**
1. Unknown and potentially vulnerable dependencies are being used
2. Dependencies may not have proper licensing
3. Dependencies may have security licensing
4. Dependencies may be of low quality
5. Unapproved versions of dependencies being used

**Controls**
1. Download only from approved external sources
2. License check
3. Security check
4. Library quality check (age, community)
5. Approved versions

**I/P**
1. External artifact
2. Internal shared artifact
3. Enterprise usage policy

**Build**

**O/P**
1. Artifact

**Actors**
1. Legal
2. Security
3. Architects
4. Developer/engineer

**Actions**
1. Legal scan
2. Security scan
3. Manage usage policy
Package Stage

**Risks**
1. Unapproved, vulnerable 3rd party dependencies are packaged for production deployment.
2. Software package contains vulnerable components
3. Integrity and security of software package
4. Software configuration contains vulnerability
5. Untraceable software changes
6. Unreliable package metadata

**Controls**
1. Package only from pipeline and/or trusted dependency mgmt
2. Vulnerability scan of application
3. Digital signing (trusted authority)
4. Vulnerability scan of configuration
5. Unique versioning
6. Truthful metadata

**I/P**
1. Artifacts
2. Dependency
3. Configuration
4. Runtime

**Package**

**O/P**
1. Artifact

**Actors**
1. Engineers
2. System (automation)

**Actions**
1. Package
2. Scan
3. Upload to artifact repo
Artifact Stage

**Risks**
1. Untrusted packaging source
2. Artifact modified after packaging and before production deploy
3. Loss of previously deployed software—needed for legal audit purposes

**Controls**
1. Only allow upload from trusted packaging source
2. Immutable artifact
3. Retention policy

**I/P**
1. Artifact
2. Metadata
3. Usage policy

**O/P**
1. Artifact

**Artifacts Repo**

**Actors**
1. Engineers
2. System (automation)

**Actions**
1. Upload
2. Download
Prod Stage

Risks
1. Deployment from untrusted sources
2. Production systems have unapproved system configurations
3. Production systems lack vulnerability detection mechanisms
4. Low quality software is promoted to production systems
5. Lack of ability to detect and resolve production system issues quickly
6. Unauthorized changes to production systems
7. Unauthorized access to production systems
8. Lack of strategy around production system changes causing unpredictable behavior

Controls
1. Only from trusted source (artifact repo/packaging)
2. Allowed configuration for production (connectivity, encryption, secrets, input tokenization, costs, ...)
3. Security (intrusion detectors, identity, threats)
4. Evaluation of testing/promotion/quality gates
5. Monitoring/logging/alerting
6. Drift management/change order
7. Production access controls (and break glass)
8. Deployment strategy enforced

I/P
1. Artifacts
2. Environment config
3. Rules for exposure and progression (aka deployment strategy)

Non Prod Deploy

O/P
1. Service availability

Actors
1. Engineering
2. Product owner
3. Business
4. Security
5. Customer/users
Creating Trust in the Deployment Pipeline

With the advent of “DevOps” more and more of the delivery pipeline is being automated and decentralized. However, with these new automated and decentralized models organizations need to ensure common validation and trust mechanisms throughout the continuous update process. In other words an optimum process would create a signed output process to certify the process. Approved signatures would be part of the automated pipeline process. This would give an organization assurances that the automated continuous updates are certified by a known authority. The goal would be to create trust in an organizations delivery pipeline.\(^6\)

\(^6\) Adapted from - [https://liquidsoftware.com/](https://liquidsoftware.com/)