DevSecOps: Injecting Security into DevOps

AFCEA/GMU Critical Issues in C4I Symposium
21-22 May 2019

Gilliam E. Duvall, PhD, Engr
Data Security Strategies, LLC

You need more than a security plan, you need a Strategy...
Agenda

• What is DevOps?

• How Does DevSecOps Work?
  • Role of Culture, Process & Technology
  • Use of metrics & KPIs

• DoD Software Development
  • Current State
  • Desired State
  • DevSecOps example

• DIB Recommendations: for Future DevOps & DevSecOps
What is DevOps?

“DevOps is the process of continuously improving software products through rapid release cycles, global automation of integration and delivery pipelines and close collaboration between teams.”

- The goal of DevOps is to **shorten the time** and **reduce the cost** of transforming an idea into a product that customers use.
- DevOps makes heavy use of automated processes to speed up development and deployment.
- An organization able to build software four times faster than its competitor has a **significant competitive advantage**.
- History has shown that customers value innovative products that may be incomplete at first, but improve quickly and steadily.
- Organizations adopt DevOps to reduce the cost and latency of development cycles, and **answer their customer’s demands**.

https://freecontent.manning.com/where-security-meets-devops-test-driven-security/
DevOps Concepts

• Combining Development & Operations processes
  • Best solution combining speed & agility to managing rapid change of coding (code change velocity) in business applications
• Coding evolutions happen in sprints, with cadence; fast application delivery to customers with business improvements in an unconstrained software change process
• The ability to compete more effectively than using legacy IT methods
  • An evolutionary successor in the world of software development life cycle (SDLC) models (e.g., Waterfall and Agile)
  • High performing DevOps - automated code development/testing/delivery
  • Infrastructure-as-code (IaC) – refresh cloud environment using machine readable code, vice physical hardware configuration
• Continuous integration/continuous delivery (CI/CD)
Traditional Build SDLC vs a DevOps Loop

DoD Instruction 5000.02, with Change 3, 2017

‘As Is’ state: Slow & rigid Build Schedules

‘To Be’ state: Fast & agile CI/CD
# Stages of DevOps Evolution

<table>
<thead>
<tr>
<th>Defining practices* and associated practices</th>
<th>Practices that contribute to success</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stage 0</strong></td>
<td></td>
</tr>
<tr>
<td>• Monitoring and alerting are configurable by the team operating the service.</td>
<td>• Build on a standard set of technology.</td>
</tr>
<tr>
<td>• Deployment patterns for building applications or services are reused.</td>
<td>• Put application configurations in version control.</td>
</tr>
<tr>
<td>• Testing patterns for building applications or services are reused.</td>
<td>• Test infrastructure changes before deploying to production.</td>
</tr>
<tr>
<td>• Teams contribute improvements to tooling provided by other teams.</td>
<td>• Source code is available to other teams.</td>
</tr>
<tr>
<td>• Configurations are managed by a configuration management tool.</td>
<td></td>
</tr>
<tr>
<td><strong>Stage 1</strong></td>
<td></td>
</tr>
<tr>
<td>• Application development teams use version control.</td>
<td>• Deployment patterns for building applications and services are reused.</td>
</tr>
<tr>
<td>• Teams deploy on a standard set of operating systems.</td>
<td>• Rearchitect applications based on business needs.</td>
</tr>
<tr>
<td></td>
<td>• Put system configurations in version control.</td>
</tr>
<tr>
<td><strong>Stage 2</strong></td>
<td></td>
</tr>
<tr>
<td>• Build on a standard set of technology.</td>
<td></td>
</tr>
<tr>
<td>• Teams deploy on a single standard operating system.</td>
<td></td>
</tr>
<tr>
<td><strong>Stage 3</strong></td>
<td></td>
</tr>
<tr>
<td>• Individuals can do work without manual approval from outside the team.</td>
<td>• Individuals can make changes without significant wait times.</td>
</tr>
<tr>
<td>• Deployment patterns for building applications and services are reused.</td>
<td>• Service changes can be made during business hours.</td>
</tr>
<tr>
<td>• Infrastructure changes are tested before deploying to production.</td>
<td>• Post-incident reviews occur and results are shared.</td>
</tr>
<tr>
<td><strong>Stage 4</strong></td>
<td></td>
</tr>
<tr>
<td>• System configurations are automated.</td>
<td>• Teams build on a standard set of technologies.</td>
</tr>
<tr>
<td>• Provisioning is automated.</td>
<td>• Teams use continuous integration.</td>
</tr>
<tr>
<td>• Application configurations are in version control.</td>
<td>• Infrastructure teams use version control.</td>
</tr>
<tr>
<td>• Infrastructure teams use version control.</td>
<td></td>
</tr>
<tr>
<td><strong>Stage 5</strong></td>
<td></td>
</tr>
<tr>
<td>• Incident responses are automated.</td>
<td>• Security policy configurations are automated.</td>
</tr>
<tr>
<td>• Resources available via self-service.</td>
<td>• Application developers deploy testing environments on their own.</td>
</tr>
<tr>
<td>• Rearchitect applications based on business needs.</td>
<td>• Success metrics for projects are visible.</td>
</tr>
<tr>
<td>• Security teams are involved in technology design and deployment.</td>
<td>• Provisioning is automated.</td>
</tr>
</tbody>
</table>

* The practices that define each stage are highlighted in bold font.
Agenda

• What is DevOps?

• How Does DevSecOps Work?
  • Role of Culture, Process & Technology
  • Use of metrics & KPIs

• DoD Software Development
  • Current State
  • Desired State
  • DevSecOps example

• DIB Recommendations: for Future DevOps & DevSecOps
What is DevSecOps?

- Integrates security with CI/CD into daily mission/business application development
  - ‘Secure & safe’ practices are injected into each of the seven ‘Fast & agile’ phases
- DevSecOps concepts integrate well with enterprise objectives to incorporate:
  - Cost savings
  - Automation
  - Cloud adoption
- Studies* have indicated DevSecOps high performers spend 50% less time remediating security issues

*Puppet | 2016 State of DevOps Report
DevSecOps Principles

- Successful implementation involves
  - Culture (people)
  - Processes (communication, feedback)
  - Technology (to deliver security at developer’s speed)
- “Moves security to the left” by empowering developer teams to ‘do’ security
- Integrates security & QA teams into the development process sooner
- DevSecOps principles to follow:
  1. Automate security in all phases (esp. testing, monitoring, audit & response)
  2. Allow developers to fail quickly (Test Driven Security)
  3. No false alarms (threshold mgmt.)
  4. Build security champions (within the developer community)
  5. Process transparency (communicates “normal”)
DevSecOps Cultural Change

Collaboration (teaming) between Developers, Operators & the cybersecurity SMEs

<table>
<thead>
<tr>
<th>NEW RESPONSIBILITIES</th>
<th>NEW SKILL REQUIREMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable developers to find and fix security-related code defects</td>
<td>Ability to provide remediation coaching and guidance on security-related code defects</td>
</tr>
<tr>
<td>Govern the use of open source components</td>
<td>Basic understanding of application development and why and how third-party components are used</td>
</tr>
<tr>
<td>Implement developer training on secure coding</td>
<td>Understanding of the basics of software development</td>
</tr>
<tr>
<td>Manage and report on application security policy, KPIs and metrics</td>
<td>The ability to measure meaningful metrics at each point in the SDLC process</td>
</tr>
<tr>
<td>Understand the requirements for security testing solutions in a DevSecOps environment — including the need for immediacy and accuracy of results to avoid impacting the delivery cycle — and enable dev to use these solutions</td>
<td>Basic understanding of developer role and tools, and the operation of a modern software delivery pipeline/factory</td>
</tr>
<tr>
<td>Create developer security champions</td>
<td>Be empathetic and consultative</td>
</tr>
</tbody>
</table>

VERACODE GUIDE - THE SECURITY PROFESSIONAL’S ROLE in a DevSecOps World
DevSecOps Process Elements

Processes to improve: Communication, Collaboration, Reporting, Measurements, Concept Integration

Collaboration: People + Process: Every participant in the process must understand the entire process and their contribution.

Automation: Process + Technology. Technology must support the process and eliminate repetitive or tedious tasks

Analysis: Technology + People. Technology must improve workflow and the analysis of bottlenecks in order to improve results with cross-functional skills.

Success = People + Process + Technology: With a successful implementation, people are able to collaborate effectively and drive results efficiently, thus overcoming the “silo effect.”

JIDO, SecDevOps CONOPS, Ver 1.0, 2017
DevSecOps Technology Stack (example)
Continuous monitoring is comprised of metrics and Key Performance Indicators (KPIs)

**Elements of a Logging Pipeline**

- **Collect**: Log events are recorded from various components of the infrastructure.
- **Stream**: Log records are captured and routed to the corresponding Layer.
- **Analyze**: Log records are inspected in order to detect anomalies and raise alerts.
- **Store**: Log records are stored in short & long-term storage facilities.
- **Access**: Log administrative console to access and review logs/alerts.

- The ideal Logging Pipeline is **automated** and allows analysis of types of traffic, application-level security metrics & security incidents in real-time
- **KPIs reflect the performance of a DevSecOps program**

Ref: ISC²
DevSecOps KPI Monitoring & Testing

Some typical metrics & KPIs
- Availability
- Change Failure
- Change Lead Time
- Change Volume
- Customer Issue Resolution Time
- Customer Issue
- Defect Burn Rate
- Defect Density
- Deployment Frequency
- Logging Availability
- Mean Time Between Failures (MTBF)
- Mean Time to Failure (MTTF)
- Mean Time to Recovery (MTTR)
- Number of Functional/Acceptance Tests
- Number of Passed/Failed Security Tests
- Number of Unit/Integration Tests
- Security Benchmark Deviation
- Security Controls
- Test Coverage
- Time to Patch
- Time to Value
- Vulnerability Patching Frequency
- Vulnerability Patching Lead Time

Interactive App Sec Testing (IAST) / Run-time App Sec Protection (RASP)*
- Continuous security services using embedded agents
- Real-time integrated testing, monitoring & protection

* www.softwaresecured.com

Ref: ISC²

Based on mission/business needs and compliance requirements
Derived DevSecOps Performance Metrics

**Defect Density**: the number of bugs identified divided by the codebase of an application. Used to set goals & measure progress within teams and within specific applications or services.

**Defect Burn Rate**: amount of time to fix vulnerabilities in an application. Focus less on the quantity of defects and instead turn to how quickly those defects are addressed by the team.

**Top Vulnerability Types and Top Recurring Bugs**: security teams track top vulnerability types will be in a much better position to help developers make long-term improvements in the way they code.

**Number of Adversaries per Application**: security teams that want to improve their developer's risk IQ should be asking them how many adversaries they think an application actually has.

**Adversary Return Rate**: this metric gets developers invested in thinking about how applications are being attacked and how often an adversary is using the same tactics, techniques and procedures.

**Time to Value**: Time between a feature request (user story creation) and realization of business value from that feature.

Ref: https://businessinsights.bitdefender.com/seven-winning-devsecops-metrics-security-should-track
## 2017 DevOps Research & Assessment (DORA) Report

<table>
<thead>
<tr>
<th></th>
<th>High performance</th>
<th>Medium performance</th>
<th>Low performance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deployment frequency</strong></td>
<td>On demand</td>
<td>Between once per week and once per month</td>
<td>Between once per week and once per month</td>
</tr>
<tr>
<td>How often does your organization deploy code?</td>
<td>(multiple deploys per day)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Lead time for changes</strong></td>
<td>Less than one hour</td>
<td>Between one week and one month</td>
<td>Between one week and one month*</td>
</tr>
<tr>
<td>What is your lead time for changes (i.e., how long does it take to go from code-commit to code successfully running in production)?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mean time to recover (MTTR)</strong></td>
<td>Less than one hour</td>
<td>Less than one day</td>
<td>Between one week and one day</td>
</tr>
<tr>
<td>How long does it generally take to restore service when a service incident occurs (e.g., unplanned outage, service impairment)?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Change failure rate</strong></td>
<td>0-15%</td>
<td>0-15%</td>
<td>31-45%</td>
</tr>
<tr>
<td>What percentage of changes results either in degraded service or subsequently requires remediation (e.g., leads to service impairment, service outage, requires a hotfix, rollback, fix forward, patch)?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Low performers were lower on average (at a statistically significant level), but had the same median as the medium performers (2017 DevOps Report)
Agenda

• What is DevOps?

• How Does DevSecOps Work?
  • Role of Culture, Process & Technology
  • Use of metrics & KPIs

• DoD Software Development
  • Current State
  • Desired State
  • DevSecOps example

• DIB Recommendations: for Future DevOps & DevSecOps
Current state – the problem:

- Software is ubiquitous and U.S. national security relies on software
  - The ability to acquire and deploy software is central to national defense and integrating with allies.
- The threats the U.S. faces change rapidly,
  - DoD’s ability to adapt and respond is now determined by its ability to develop and deploy software to the field
- **The current approach to software development is a leading source of risk to DoD**
  - It takes too long, is too expensive & exposes warfighters to unacceptable risk
  - Software is **not** being used to enable a more effective force, strengthen our ability to work with allies, and improve the business processes of the Department
- Nothing is changing: most of this has been said before - 1987 DSB report on military software

“Software is Never Done: Refactoring the Acquisition Code for Competitive Advantage” -- Defense Innovation Board, 3 May 2019
DoD DevOps Desired State

• **Speed and cycle time are the most important metrics for managing software**
  • DoD needs to deploy and update software that works for its users at the speed of (mission) need
  • Execute inside the OODA loop of our adversaries to maintain advantage

• **Software is made by people and for people, so digital talent matters**
  • DoD’s current personnel processes and culture will not allow its military and civilian software capabilities to grow nearly enough to meet its needs.
  • New mechanisms are required.

• **Software is different than hardware (and not all software is the same)**
  • Hardware can be developed, procured, and maintained
  • Software is an enduring and evolving capability that must be supported and continuously improved throughout its lifecycle

From “Software is Never Done: Refactoring the Acquisition Code for Competitive Advantage” -- Defense Innovation Board, 3 May 2019
Defense Innovation Board  
Ten Commandments of Software

1. Make computing, storage, and bandwidth abundant to DoD developers and users.

2. All software procurement programs should start small, be iterative, and build on success – or be terminated quickly.

3. The acquisition process for software must support the full, iterative life cycle of software.

4. Adopt a DevSecOps culture for software systems.

5. Automate testing of software to enable critical updates to be deployed in days to weeks, not months or years.

6. Every purpose-built DoD software system should include source code as a deliverable.

7. Every DoD system that includes software should have a local team of DoD software experts who are capable of modifying or extending the software through source code or API access.

8. Only run operating systems that are receiving (and utilizing) regular security updates for newly discovered security vulnerabilities.

9. Security should be a first-order consideration in design and deployment of software, and data should always be encrypted unless it is part of an active computation.

10. All data generated by DoD systems - in development and deployment - should be stored, mined, and made available for machine learning.

DoD must develop/deploy software as fast or faster than adversarial tactics -- building on commercially available tools and technologies for the four software types.

DIB Software Acquisition and Practices (SWAP) study, May 2019

Copyright Data Security Strategies, LLC – All Rights Reserved
DoD DevSecOps Metrics

• Traditional metrics within DoD is that software complexity/productivity is often estimated based on number of source lines of code (SLOC)

• While easily measured, it is not necessarily predictive of cost, schedule, or performance
  • Obsolete metrics are irrelevant at best and, at worst, could be misleading

• The process for software DevSecOps to manage travel is different from what is required to manage the software on an F-35 – suggesting a taxonomy with four types of software requiring four different approaches:
  • **Type A**: commercial (“off-the-shelf”) software with no DoD-specific customization required
  • **Type B**: commercial software with DoD-specific customization needed
  • **Type C**: custom software running on commodity hardware (in data centers or in the field)
  • **Type D**: custom software running on custom hardware (e.g., embedded software)
DoD DevSecOps Metrics – cont.

Alternatively, measures useful for DoD to track DevSecOps performance and drive improvement in cost/schedule, performance & security include the following:

<table>
<thead>
<tr>
<th>#</th>
<th>Metric</th>
<th>Target value (by software type)</th>
<th>Typical DoD values for SW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Time from program launch to deployment of simplest useful functionality</td>
<td>COTS apps: &lt;1 mo, Custom SW: &lt;3 mo, COTS HW/OS: &lt;6 mo, Real-time HW/SW: &lt;1 yr</td>
<td>3-5 yrs</td>
</tr>
<tr>
<td>2</td>
<td>Time to field high priority fcn (spec → ops) or fix newly found security hole (find → ops)</td>
<td>COTS apps: &lt;1 wk, Custom SW: &lt;1 wk, COTS HW/OS: &lt;1 wk, Real-time HW/SW: &lt;1 wk</td>
<td>1-18 m</td>
</tr>
<tr>
<td>3</td>
<td>Time from code committed to code in use</td>
<td>COTS apps: &lt;1 wk, Custom SW: &lt;1 hr, COTS HW/OS: &lt;1 wk, Real-time HW/SW: &lt;1 wk</td>
<td>1-18 m</td>
</tr>
<tr>
<td>4</td>
<td>Time req’d for full regression test (automat’d) and cybersecurity audit/penetration testing</td>
<td>COTS apps: &lt;1 da, Custom SW: &lt;1 da, COTS HW/OS: &lt;1 da, Real-time HW/SW: &lt;1 da</td>
<td>2 yrs</td>
</tr>
<tr>
<td>5</td>
<td>Time required to restore service after outage</td>
<td>COTS apps: &lt;1 hr, Custom SW: &lt;6 hr, COTS HW/OS: &lt;1 day, Real-time HW/SW: N/A</td>
<td>?</td>
</tr>
<tr>
<td>6</td>
<td>Automated test coverage of specs/code</td>
<td>COTS apps: N/A, Custom SW: &gt;90%, COTS HW/OS: &gt;90%, Real-time HW/SW: 100%</td>
<td>?</td>
</tr>
<tr>
<td>7</td>
<td>Number of bugs caught in testing vs field use</td>
<td>COTS apps: N/A, Custom SW: &gt;75%, COTS HW/OS: &gt;75%, Real-time HW/SW: &gt;90%</td>
<td>?</td>
</tr>
<tr>
<td>8</td>
<td>Change failure rate (rollback deployed code)</td>
<td>COTS apps: &lt;1%, Custom SW: &lt;5%, COTS HW/OS: &lt;10%, Real-time HW/SW: &lt;1%</td>
<td>?</td>
</tr>
<tr>
<td>9</td>
<td>% code avail to DoD for inspection/rebuild</td>
<td>COTS apps: N/A, Custom SW: 100%, COTS HW/OS: 100%, Real-time HW/SW: 100%</td>
<td>?</td>
</tr>
<tr>
<td>10</td>
<td>Number/percentage of functions implemented</td>
<td>COTS apps: 80%, Custom SW: 90%, COTS HW/OS: 70%, Real-time HW/SW: 95%</td>
<td>?</td>
</tr>
<tr>
<td>11</td>
<td>Usage and user satisfaction</td>
<td>COTS apps: TBD, Custom SW: TBD, COTS HW/OS: TBD, Real-time HW/SW: TBD</td>
<td>?</td>
</tr>
</tbody>
</table>

Defense Innovation Board Metrics for Software Development, 3 May 2019
### DoD DevSecOps Metrics – cont.

#### Program Management, Assessment, and Estimation Metrics

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Metric</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Complexity metrics</td>
<td>#/type of specs, structure of code, # programmers</td>
<td>Partial/manual tracking</td>
</tr>
<tr>
<td>13</td>
<td>Development plan/environment metrics</td>
<td>#/type of platforms, #/type deployments</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>“Nunn-McCurdy” threshold (for any metric)</td>
<td>1.1X, 1.25X, 1.5X, 1.5X each effort, 1.25X Total $</td>
<td></td>
</tr>
</tbody>
</table>

12. Structure of specifications, code, and development and execution platforms
13. Structure and type of development & operational environment
14. Tracking software program progress
   - 25% unit cost growth and 50% total program cost growth thresholds often will not make sense for continuously developed software programs
DevOps on a Hardware Platform

Lessons Learned
• Culture change has been the biggest hurdle
• The program must recognize and accept that things will go wrong.
• Security controls limit flexibility and communication
Agenda

• What is DevOps?

• How Does DevSecOps Work?
  • Role of Culture, Process & Technology
  • Use of metrics & KPIs

• DoD Software Development
  • Current State
  • Desired State
  • DevSecOps example

• DIB Recommendations: Future DoD DevOps & DevSecOps
Future DoD DevOps & DevSecOps Concept

OSD & Congress

OSD & Services

Services & OSD

DoD & Industry

A. Refactoring Statutes, Regulations, and Processes
  - Rec A1
  - Rec A2
  - Recs A3-A7

B. Creating Digital Infrastructure
  - Rec B1
  - Rec B2
  - Rec B3
  - Recs B4-B7

C. Creating New Paths for Digital Talent
  - Rec C1
  - Rec C2
  - Recs C3-C4

D. Changing the Practice (DevSecOps)
  - Rec D1
  - Rec D2
  - Rec D3
  - Recs D4-D8

Report Themes
Recommendations (bold = key rec)
Draft Implementation Plans

Description
Stakeholders
Background
Desired State
Role of Congress
Draft Implementation Plan
  - Action A1.1
  - Action A1.2

Subgroup Ideas
  - Idea 1
  - Idea 2

Related Recs
  From Previous Studies
  - Rac 1
  - Rac 2

DIB Software Acquisition and Practices (SWAP) study, May 2019

Copyright Data Security Strategies, LLC – All Rights Reserved
### Future DoD DevOps & DevSecOps Concepts – cont.

<table>
<thead>
<tr>
<th>Line of Effort A (Congress and OSD): Refactor statutes, regulations, and processes for software</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A1</strong> Establish one or more new acquisition pathways for software that prioritize continuous integration and delivery of working software in a secure manner, with continuous oversight from automated analytics</td>
</tr>
<tr>
<td><strong>A2</strong> Create a new appropriation category for software capability delivery that allows (relevant types of) software to be funded as a single budget item, with no separation between RDT&amp;E, production, and sustainment</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Line of Effort B (OSD and Services): Create and maintain cross-program/cross-Service digital infrastructure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>B1</strong> Establish and maintain digital infrastructure within each Service or Agency that enables rapid deployment of secure software to the field, and incentivize its use by contractors</td>
</tr>
<tr>
<td><strong>B2</strong> Create, implement, support, and use fully automatable approaches to testing and evaluation (T&amp;E), including security, that allow high-confidence distribution of software to the field on an iterative basis</td>
</tr>
<tr>
<td><strong>B3</strong> Create a mechanism for Authorization to Operate (ATO) reciprocity within and between programs, Services, and other DoD agencies to enable sharing of software platforms, components, and infrastructure and rapid integration of capabilities across (hardware) platforms, (weapon) systems, and Services</td>
</tr>
</tbody>
</table>

DIB Software Acquisition and Practices (SWAP) study, May 2019

<table>
<thead>
<tr>
<th>Line of Effort C (Services and OSD): Create new paths for digital talent (especially <em>internal</em> talent)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C1</strong> Create software development units in each Service consisting of military and civilian personnel who develop and deploy software to the field using DevSecOps practices</td>
</tr>
<tr>
<td><strong>C2</strong> Expand the use of (specialized) training programs for CIOs, SAEs, PEOs, and PMs that provide (hands-on) insight into modern software development (e.g., Agile, DevOps, DevSecOps) and the authorities available to enable rapid acquisition of software</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Line of Effort D (DoD and industry): Change the practice of how software is procured and developed</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D1</strong> Require access to source code, software frameworks, and development toolchains—with appropriate IP rights—for DoD-specific code, enabling full security testing and rebuilding of binaries from source</td>
</tr>
<tr>
<td><strong>D2</strong> Make security a first-order consideration for all software-intensive systems, recognizing that security-at-the-perimeter is not enough</td>
</tr>
<tr>
<td><strong>D3</strong> Shift from the use of rigid lists of requirements for software programs to a list of desired features and required interfaces/characteristics to avoid requirements creep, overly ambitious requirements, and program delays</td>
</tr>
</tbody>
</table>

DIB Software Acquisition and Practices (SWAP) study, May 2019
In Conclusion

We discussed:

• What is DevOps?

• How Does DevSecOps Work?
  • Role of Culture, Process & Technology
  • Use of metrics & KPIs

• DoD Software Development
  • Current State
  • Desired State
  • DevSecOps example

• DIB Recommendations: for Future DevOps & DevSecOps
Questions?

Dr. Gil Duvall
President & CEO
Data Security Strategies, LLC

e-mail: gil@datasecuritystrategies.com

website: www.datasecuritystrategies.com