INSTITUTE FOR DEFENSE ANALYSES

Cybersecurity and DoD System Development: A Survey of DoD Adoption of Best DevSecOps Practice

Rachel Kuzio de Naray, Project Leader
George "Lee" Kennedy
Ryan Wagner
Steven Wartik

September 2021

IDA Document P-22749
Log: H 2021-000267

INSTITUTE FOR DEFENSE ANALYSES
4850 Mark Center Drive
Alexandria, Virginia 22311-1882

About This Publication

This work was conducted by the IDA Systems and Analyses Center under contract HQ0034-19-D-0001, Project AX-1-3100, "Technical Analysis for the Director, Developmental Test, Evaluation and Assessments (D(DTE&A))". The views, opinions, and findings should not be construed as representing the official position of either the Department of Defense or the sponsoring organization.

Acknowledgments

The authors are very grateful to the IDA committee, Dr. John S. Hong (chair), Dr. Davy Y. Lo, Dr. Jessica G. Swallow, and Dr. Karen D. Gordon, for providing technical review of this effort.

For More Information

John S. Hong, Project Leader
jhong@ida.org, 703-845-2564

Stephen M. Ouellette, Director, SED
souellet@ida.org, 703-845-2443

INSTITUTE FOR DEFENSE ANALYSES

IDA Document P-22749

Cybersecurity and DoD System Development: A Survey of DoD Adoption of DevSecOps Practice

Rachel Kuzio de Naray, Project Leader

George “Lee” Kennedy

Ryan Wagner

Steven Wartik

Executive Summary

Introduction

Department of Defense (DoD) software-intensive systems have a long and unfortunate history as expensive acquisitions. Delivered late, full of bugs, or with difficult and clumsy interfaces—or a combination of all three—they satisfy neither their customers nor operators. Attempts to fix DoD software acquisition have been an active topic of research for over half a century.

In the 1970's, DoD model of software acquisition was encapsulated by the Waterfall Model. The thinking was that a customer would present a developer with a set of requirements. The developer would eventually respond with working software to install and use. This was similar to how bridges were built. Span this chasm, said the requirements. Here's your bridge, said the contractor. Unfortunately, process models that worked in engineering physical systems didn't work for software. Requirements were never right the first time. Their implementations didn't work well either. Even when software worked, the Waterfall Model didn't allow for improvement incorporating feedback and experience on how using the software changed the workplace paradigm.

Alternate software development models emerged in the 1980's. Many emphasized automation: formally defined processes supported by tools, comprehensive documentation including executable models, and extensive, complex planning tools that would allow management to track progress.

The Agile Manifesto, introduced in 2001, was a response to this state of practice. The Agile methodology emphasizes delivery in increments. Increments deliver bug fixes, but also improvements based on customer feedback. The feedback is not from formally delivered reports, but collaboration between customers and developers. Agile's proponents never claimed their methodology was a cure-all, only that it was better suited to meet the needs and issues that arise in modern software-oriented projects. To judge from its widespread adoption, much of the software development community agrees.

Some Agile critics say it emphasizes collaboration with the wrong community. Customers are not necessarily users. This realization led to the DevOps methodology,[1] which encourages collaboration not between developers and customers, but between

[1] DevOps is a portmanteau of "Development" and (Information Technology) "Operations."

developers and operators. With these collaborations, the thinking goes, increments will better reflect real needs. DevOps has been called, “Agile done right.”

The danger of this collaboration is the potential for overlooking security. Everyone acknowledges the need for security, but for most operators security is mostly an extra step. Operators want new features, not the hassle of two-factor authentication. Furthermore, few developers enjoy implementing security. In the pressure to deliver on time, security can get shunted aside and promised in a future increment.

The DevSecOps methodology tries to balance quick delivery against the need for security (the “Sec” addition to DevOps). DevSecOps adds a security team to development. The security team is responsible for ensuring security is considered early and never neglected. A security team functions efficiently when it collaborates with developers, testers, and operators to establish an infrastructure that assists in incorporating security without excessively hindering progress.

In 2020, Sarah Standard[2] tasked The Institute for Defense Analyses (IDA) with studying the adoption of DevSecOps in DoD. This report discusses uses of DevSecOps, covers successes and problem areas, and makes recommendations for actions DoD can take to improve its use of DevSecOps.

Approach

IDA sought to understand why projects are increasingly opting to use DevSecOps when there is no DoD-wide mandate. The IDA team began by reviewing policies, practices, technologies, and standards related to DevSecOps and security. It reviewed DoD materials, paying close attention to other government standards, in particular those from the National Institute of Standards and Technology (NIST), and took care to consider commercial best practices, technologies, and standards as well. IDA identified several works considered foundational to DevOps and DevSecOps as a body of practice. Of particular interest was *The DevOps Handbook* [4], a handbook of DevOps principles and best practices.

Because this handbook is well known and widely read, the IDA team assumed DevSecOps practitioners would follow its recommendations. The team prepared a survey to discover how projects were using DevSecOps. How were interactions established between teams? What automated security scans were being performed? Had their experiences been positive or negative? What difficulties were experienced? What were the lessons learned? The literature frequently states that adopting DevSecOps requires an organization-wide mindset change. The team wanted to gauge the degree to which this change occurred, and to understand what actions DoD could take to promote it in the future.

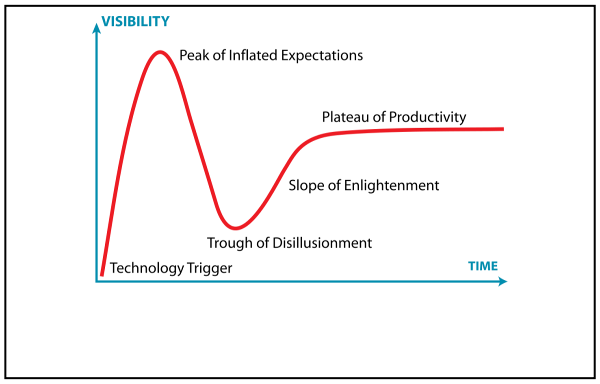
[2] Ms. Standard is the Cybersecurity/Interoperability Technical Director, Office of the Undersecretary of Defense Research and Engineering, Director, Developmental Test, Evaluation and Assessments.

The project's sponsor helped the IDA team identify and contact individuals within DoD using DevSecOps or non-Waterfall software development approaches. The IDA team sent out surveys, or conducted telephone interviews, during February and March 2021 and analyzed the responses to identify commonalities and differences. The team also looked for practices and technologies that worked or didn't work, and areas for improvement.

Findings

The survey and interview results were useful for describing how select DoD projects were using Agile, DevOps, or DevSecOps approaches. In lieu of conducting a comprehensive, DoD-wide survey, the team broadened its perspective by searching existing literature. A 2020 IDA report, *DevSecOps: State of the Art and Relevance to the Department of Defense* [5], and several products from Gartner Group were enlightening. The reports contained two important analyses that helped lend context to this report:

- Extensive Application Security Testing (AST) is necessary for successful DevSecOps practice, and there are 14 important commercial players who develop AST tools and the infrastructure to support them. Of these 14, five can be considered industry leaders. The others are attempting to catch up, exploring exciting, but unproven technologies or filling niches.
- The Gartner "hype cycle" illustrates a common paradigm for technology adoption: initial wild, unrealistic promises give way to disillusionment, which is followed by gradual realization of a functionally viable subset of the original technology and its slow, but steady assimilation into the workplace. Gartner's 2021 forecast predicts DevSecOps maturity between 2023 and 2026.



Gartner Group's Hype Cycle

The IDA team found through surveys and interviews several technologically focused commonalities that are characteristic of DevSecOps adoption. These include:

- ***Use of Static Application Security Testing (SAST) tools.*** Eight respondents reported using Fortify or SonarQube (or both), two commercial industry-leading SAST tools. Fortify was developed by Micro Focus, one of the companies Gartner identified as an industry leader.
- ***Use of container technology.*** Containers, which provide the capability to run an application (or a suite of applications) in an isolated environment, are increasingly seen as a solution to many modern-day problems that were once the purview of

virtual machines (VMs). Compared with VMs, containers can be deployed quicker, scaled more easily, and started, stopped, and restarted quicker. Containers are well-suited to cloud-based deployment, where the cloud-based processor itself is a VM.

- ***Consistent use of different environments for development, testing, staging, and deployment.*** The DevSecOps methodology advocates these different environments as a means to isolate activities and avoid conflicts. Development will never interfere with testing, and vice versa. More important is the ability to set up an automated process whereby any artifact in development *must* pass through testing prior to staging, and *must* pass through staging prior to deployment. Every change must be tested in different environments before being placed into a production system.
- ***Consistent use of repositories.*** Developers and testers are storing their artifacts in git repositories, meaning the artifacts are version-controlled, configuration-managed, easily viewed, and automatically subjected to scans and tests.

Automated scans and tests are achieved by implementing pipelines. A pipeline is a description of the process by which an artifact progresses from development through testing, into staging, and on to deployment. To the extent possible, pipelines are automated. Some environments automate pipelines so fully that any change submitted by a developer, no matter how small, moves into production automatically and almost immediately. This kind of pipeline is the epitome of Continuous Delivery (also known as Continuous Deployment, both abbreviated “CD”), wherein increments have negligible overhead.

IDA’s results show DoD pipelines have yet to reach this level of efficiency. Developing a pipeline so robust that most changes can be trusted is no simple undertaking. DoD’s situation is complicated because development and testing often take place on unclassified systems, whereas deployment is often on classified systems. These distinct environments have different policies and run different software. Software cannot be transferred from an unclassified to a classified system automatically, so fully automated pipelines are impossible. Classified artifacts cannot ever be transferred from a classified to an unclassified system, complicating debugging of problems found during operation.

The Survey and Interview Result Summary table below summarizes the IDA team’s survey and interview results.[3] Each row lists something recorded in more than one response (the third column lists the number).

[3] IDA received 18 written or virtual interview responses to the survey. Eight were from the developmental test community, four from the operational test community and six from the development community. Organizational break out: six from the Air Force, and four more from Space Force, four from the Army, and three from the Navy that also included input from the Marine Corps. DoD. DoD DSO COP input was also included.

Survey and Interview Result Summary

Success Stories		**# of Responses**
1	Enculturation of DevSecOps is foundational	8
2	Incorporate test processes and environments up front	6
3	Fully leverage automation for testing, pipeline, and builds	8
4	Results can be validated	4
5	DevSecOps can be adapted to physically isolated environments	2
Problem Areas		**# of Responses**
1	The current DoD acquisition model does not lend itself to a DevSecOps development methodology	3
2	The current DoD Authority To Operate (ATO) process is not fully compatible with DevSecOps	4
3	Many DoD systems may be incompatible with the DevSecOps process	6
4	There is no adequate standard definition of DevSecOps	4
5	The role of Developmental Testing (DT) within a DevSecOps methodology is unclear	6
6	Forming teams proved more challenging than expected	4
7	Organizations had trouble allocating resources	4
8	The difficulties of classified versus unclassified DevSecOps development are unresolved	5
Lessons Learned		**# of Responses**
1	Starting with DevSecOps is easier than switching to DevSecOps	6
2	DevSecOps is easier on small projects than big ones	5
3	DevSecOps must be adapted to the operational environment	6
4	Automation is key	8
5	Security requirements need to be more mission-focused	7
6	Leverage good architecture and design to support security	6

Conclusions and Recommendations

Every DevSecOps practitioner has learned that inculcating DevSecOps into one’s organization takes time and commitment. It involves introducing new, previously unconsidered or ill-defined roles and responsibilities into one’s organizational structure. These actions require patience, training, and commitment. Respondents said DevSecOps has higher start-up overhead than some traditional projects. An unwillingness to fully invest is perhaps why some interviewees said they had to “fail back to Waterfall.”

Other organizations have studied switching to DevSecOps, and proposed maturity models to measure commitment and progress. IDA found three:

- The Naval Information Warfare Center Atlantic has a nine-level maturity model based on an organization's practices.
- DoD has created a DevSecOps Maturity Review, a list of questions designed to help understand an organization's approach to DevSecOps and to propose improvements.
- The Open Web Application Security Platform (OWASP) has proposed the DevSecOps Maturity Model (DSOMM), a four-level security-focused model to ascertain the degree to which security is built into an application.

These models lack the rigor of a Capability Maturity Model Integration (CMMI)-style maturity model, but they still give a useful qualitative assessment of progress. The Chief Science Officer of the United States Air Force (USAF) is encouraging CMMI-like maturity assessments for DevSecOps. If he is successful, organizations will be able to judge their progress quantitatively.

IDA makes 12 recommendations for DoD based on the results and analyses in this report that, taken together, will ease DevSecOps adoption on DoD projects.

Recommendations to Ease DevSecOps Adoption by DoD

1	Define and propagate an adequate standard definition of DevSecOps, specifically of integrating the traditional role of DT in DevSecOps methodology.
2	Create top down understanding, commitment, and application of DevSecOps concepts in the development organization and culture.
3	Adopt a DevSecOps Maturity Model to measure and improve development program security effectiveness.
4	Understand the cultural and resource challenges in converting Waterfall programs to DevSecOps, scaling up from small projects, and cross domain development.
5	Identify mission focused security requirements and incorporate test processes and environments up front.
6	Identify, continually evaluate, and manage to objective security metrics to ensure the DevSecOps environment produces software that meets mission security requirements.
7	Commit sufficient operational and security staff in addition to developers to ensure teams produce software that meets operational and security requirements.
8	Ensure the development methodology is appropriate for the program requirements and can be adapted, especially for physically isolated or highly sensitive systems.
9	Incorporate good architecture and design in the development environment and pipeline to support security.
10	Work to identify and integrate testing that cannot be automated in a disciplined repeatable DevSecOps process.
11	Fully leverage automation including virtualization, containerization, and Infrastructure as Code (IaC) for testing, pipeline and builds.
12	Use DoD enterprise code repositories whenever possible to reduce rework and minimize supply chain threat.

Contents

(This page is intentionally blank.)

Figures

Tables

(This page is intentionally blank.)

1. Introduction

A. Purpose

Department of Defense (DoD) organizations are attempting to balance the desire for quick delivery with the need for security in software development by the rapid adoption of DevSecOps. DevSecOps is a portmanteau combining development, security, and operations. (The operations are usually information technology [IT] operations.) DevSecOps evolves the Agile and DevOps methodologies by integrating security experts into the development team in an attempt to shift security concerns "left," earlier in the software development process.[4]

The Cybersecurity/Interoperability Technical Director, Office of the Undersecretary of Defense (OUSD) Research and Engineering (R&E), Director, Developmental Test, Evaluation, and Assessments (D,DTE&A), tasked the Institute for Defense Analyses (IDA) with studying the adoption of DevSecOps in DoD. As such, IDA reviewed current general practices for employing and securing DevSecOps as a developmental methodology, with a focus on current practices in DoD to identify ways to better integrate cybersecurity (including Developmental Test, Evaluation, and Assessments [DTE&A] cybersecurity requirements and guidance) with modern, secure software development practices (e.g., DevSecOps and other Agile-like practices) throughout the secure software development life cycle. The goals were to identify representative processes, tools, and artifacts; evaluate their sufficiency as part of the current body of practice; determine modifications that would integrate cybersecurity more fully into the development process; and finally, provide recommendations for implementing those modifications.

B. Scope

This study encompasses the DTE&A goals of examining the degree to which DoD organizations employed both current and best policies, practices, technologies, and standards. As such, IDA researched and analyzed the spectrum of current and planned practices to apply and enhance cybersecurity in DoD software development, and compared and contrasted those practices with the private sector. IDA performed a review of current general practices for employing and securing DevSecOps as a developmental methodology, with a focus on current practices in DoD, and identified areas where improvements could be made.

[4] Process workflow models are usually drawn left to right, with early activities depicted on the left.

C. Outline

This report is organized into the following sections:

- Introduction (this section) describes the motivation for this report and provides the reader with the general scope of the problem area.
- Approach describes the approach IDA used to address the problem.
- DevSecOps Overview and State of Practice presents a general overview of DevSecOps and the state of practice in DoD.
- Findings presents the findings from the analyses performed in the course of this task.
- DevSecOps Maturity Models introduces maturity models for assessing DevSecOps organizational practices.
- Conclusion and Recommendations lists conclusions from the findings, and gives recommendations for actions to be taken to improve cybersecurity in DoD-developed or -funded systems.

D. Intended Audience

The target audience consists of individuals interested in improving DoD system cybersecurity software development methodologies. The report includes recommendations on policies, practices, standards, and technologies to institute, adopt, follow, and acquire. These recommendations should be useful for the following concerns:

- DoD, Service, and Agency representatives looking to influence and standardize software and system development approaches throughout their organizations.
- Program managers wishing to assess the state of, and potentially mature their programs; and program managers developing new programs.
- Individual architects, designers, coders, and testers seeking insight on recent technological advances in achieving cybersecurity.

2. Approach

This chapter describes the research methodology IDA followed to derive its conclusions and recommendations. The methodology attempted breadth insofar as time and resources permitted. IDA wished to uncover the full spectrum of current and planned practices as concerns cybersecurity in DoD software development, and to compare and contrast those practices with the private sector. IDA also analyzed academic research to uncover longer-term trends.

A. Review Current Policies, Practices, Technologies, and Standards

IDA performed a review of current general practices for employing and securing DevSecOps as a developmental methodology, with a focus on current practices in DoD. The DoD DevSecOps methodology currently resides in a phase of rapid adoption and evolution. Of necessity, any review will be a snapshot of what is in place at a given point in time. IDA team members used the monthly DoD DevSecOps Community of Practice (COP) meetings as a jumping-off point to collect resources on current DoD DevSecOps[5] implementations to include discussion of policies, practices, technologies, and standards. These meetings, organized by the DoD Chief Information Officer (CIO), provided an effective way to ensure DoD organizations can keep abreast of developments in the DoD DevSecOps community, and specifically of efforts by the United States Air Force (USAF) Chief Software Officer (CSO) who chairs the meetings and whose organization has been designated as the executive agent for DoD enterprise DevSecOps efforts.

The USAF CSO, at the time of this writing Nicolas Chaillan, also hosts an extensive repository of documents at the Air Force Software site.[6] In addition to the numerous white papers, briefings, minutes of working group meetings, architecture, service offering, and process documents, there are training videos, reading lists of references, and links to

[5] The DevSecOps COP monthly meeting is currently conducted in the DoD Teams Collaboration Virtual Resource (CVR) at this link https://teams.microsoft.com/l/meetup-join/19%3a4d92705edf2e40e2a54a7163f0b24d05%40thread.skype/1588014888844?context=%7b%22Tid%22%3a%2221acfbb3-32be-4715-9025-1e2f015cbbe9%22%2c%22Oid%22%3a%2244e424f7-0bcc-4c72-97d7-22160ced090d%22%7d. The files section and the associated site hosted by the USAF CSO at https://software.af.mil/ contain numerous briefings and background documents on all aspects of DevSecOps adoption in DoD.

[6] Ibid.

external resources such as DoD secure software repositories for developers.[7] This is a living archive and material is added as new resources are developed, so the library grows in a DevSecOps-like fashion, building from “minimum viable product” through continuous improvement.

IDA also obtained documents from the Navy, Marine Corps, Army, and Undersecretary of Defense (USD) (R&E). Of particular note is the Naval Information Warfare Center (NIWC) Atlantic Marine Corps Business Operations Support Services (MCBOSS) brief on DevSecOps Maturity and Business Value Driven Goals,[8] with its included discussion on maturity models, and the draft DevSecOps Test & Evaluation Guidebook. [21]

Finally, IDA reviewed the DoD Enterprise DevSecOps Reference Design. [33] This document is DoD’s evolving reference for “a logical description of the key design components and processes to provide a repeatable reference design that can be used to instantiate a DoD DevSecOps software factory.”

B. Identify Best Policies, Practices, Technologies, and Standards

The available DoD sources contained more documentation than could be reasonably reviewed within the time allocated for the study, so the IDA team narrowed its focus to most-cited references. At the core, four books were seen as foundational to Agile and DevOps (and consequently to DevSecOps) as a body of practice.

1. A Seat at the Table [1], by Mark Schwartz, published in 2017, with a more general focus on leadership’s place in Agility in software development;
2. The Phoenix Project [2], by Gene Kim, Kevin Behr, and George Spafford, published in 2013;
3. The Unicorn Project [3], by Gene Kim, published by IT Revolution Press in 2019;
4. The DevOps Handbook [4], by Gene Kim, Jez Humble, Patrick Debois, and John Willis, published in 2016 and undergoing update.

Several respondents emphasized the importance of these foundational texts, and were consistent in the observation that a lack of understanding by leadership and developers of the underlying Agile and DevSecOps principles and practices was a key factor in the failure of Agile/DevSecOps projects, and specifically, the failure of legacy projects to transition

[7] https://software.af.mil/dsop/services/ available 07/16/2020.

[8] Naval Information Center Atlantic Brief: DevSecOps Maturity and Business Value. Derived from [4].

successfully from Waterfall methodology. This lack of proper grounding was also seen as the source of unrealistic expectations for resourcing and results.

C. Develop Cybersecurity and Software Development Survey

IDA sought to understand the degree to which organizations employed both current and best policies, practices, technologies, and standards. It's not uncommon for organizations to pick and choose aspects of the Agile model to implement rather than implement all aspects of Agile, so IDA developed questions intended to help understand exactly how DoD organizations implemented Agile approaches. IDA also wished to explore what literature frequently states: the major impediment to adopting Agile and DevSecOps is a change of mindset. Had developers adopted a new mindset, and to what degree? Furthermore, many variations of the Agile methodology are practiced, and IDA wanted to learn which ones DoD uses. IDA developed a survey, included in Appendix A, which asks questions designed to uncover individuals' perceptions of their organization's approach to software development. Its questions fall into five categories.

1. Context questions, which ask about the respondent's organization, their role, their projects, and their organization's general involvement with DevSecOps.
2. Developmental Testing and Evaluation (DT&E) integration questions, asking whether efforts to integrate DevSecOps into Developmental Testing (DT) have succeeded: what's worked, hasn't worked, and lessons learned.
3. Team and Environment questions, covering the DevSecOps roles established and the computing environments used to facilitate those roles.
4. Development Process questions, asking how new and changed artifacts are propagated throughout an organization, and the actions, both automated and manual, that are triggered as a consequence of propagation.
5. Test Process questions, asking about tests run and the motivation for running those tests (usually described through references to standards), and also about how test results are logged and the uses of those logs.

(The categories do not match exactly the five groupings in Appendix A. The team found it convenient to deviate from the categories in order to maintain a natural flow of conversation.)

IDA decided survey questions should be detailed to get a picture of how Agile and DevSecOps were (or were not) being employed. IDA created questions targeted more to performers than managers.

D. Distribute Surveys and Conduct Interviews

Beginning in January 2021, project sponsor Sarah Standard used email to reach out to 10 individuals she knew to be involved in programs using Agile, DevOps, or DevSecOps development. She asked for recommendations of people IDA could interview, as well as people willing to complete the survey.

The 10 individuals responded throughout January with names and contact information—usually others, occasionally themselves. IDA, as stated in Develop Cybersecurity and Software Development Survey above, preferred performers, specifically developers and testers, to managers, and tried to identify those among the respondents. The surveys were ultimately sent to 45 individuals, sometimes directly from IDA, sometimes through managers. IDA received 10 written survey responses, one of which had "N/A" for every question and was discarded.

IDA preferred interviews to completed surveys, believing the ability to ask and probe would yield higher quality information than simple written responses. When a respondent indicated openness to supporting the task, IDA attempted to set up an interview. The project took place during the COVID-19 pandemic, so interviews were conducted through teleconference or videoconference software; no in-person meetings occurred, and IDA researchers did not have the opportunity to visit developer or tester facilities. The interviews, which occurred during February and March of 2021, usually lasted one hour. IDA conducted 11.[9] They were conducted by leading the subject through each of the survey questions. IDA researchers listened to the responses, took notes, and asked follow-up questions as they saw fit.

E. Analyze and Document Results

The analysis process began once IDA finished conducting the interviews and collected the surveys. IDA compared and contrasted respondents' results to all questions. The objectives were

- ***To identify commonalities.*** IDA wished to uncover DevSecOps trends in DoD software development, thereby indicating department-wide directions (without necessarily passing judgement on those directions).
- ***To identify differences.*** If IDA identified a single project with unique practices, IDA wanted to determine whether and how these practices were contributing to its success or failure. Similarly, if practices could be grouped into sets, it would be useful to study the compatibility of practices across

[9] Some respondents both returned a survey and participated in interviews. IDA contacted a total of 18 distinct individuals.

these sets. Such a study, conducted now, might forestall later gross incompatibilities across communities within DoD.

- ***To leverage respondents' knowledge of successes and shortfalls.*** Individual perspectives on what has and has not worked can be invaluable in charting directions and making recommendations.
- ***To identify areas for improvement.*** IDA wanted to understand how DoD practices compared to an idealized set of capabilities. Any gaps in this area—e.g., incomplete training, inadequate tooling, insufficient management support—could be the basis for important recommendations on changes to DoD DevSecOps policies and practices.

F. Limitations

Although the IDA researchers attempted to review the broadest possible range of available material on current DoD DevSecOps practices, there were some strict limiting factors on what they could cover. The available DoD sources contained more documentation than the IDA team could reasonably review within the time allocated for the study, so once the surveys began the team focused on references identified by the survey responses. In addition to the relatively compressed span for the study, which reduced the time available for a more extensive literature survey, there were limitations imposed by the COVID-19 pandemic. Not only did this preclude in-person meetings (teleconferences or videoconferences were substituted), but IDA researchers did not have the opportunity to visit developer or tester facilities. Further, due to the demands of telework imposed by the pandemic, respondents were limited in the time they could devote to the effort. The interviews, which occurred during February and March of 2021, usually lasted one hour with follow-up as available. IDA received a total of 18 written or virtual interview responses to the survey, which could have been better balanced given greater time to identify and access practitioners. Nonetheless, IDA was able to include responses from all the Services, and from the operational and development test community, as well as developers who were working in DevSecOps environments. DoD DevSecOps Community of Practice input was also included to provide a broader perspective. While there was sufficient diversity to provide significant insight into the current state of DevSecOps practice in DOD, further study would allow for greater granularity in specific findings and more prescriptive recommendations for improvement.

(This page is intentionally blank.)

3. DevSecOps Overview and State of Practice

A. Overview

Software development proved difficult to discipline almost from its beginning. One of the first efforts to provide a framework for managing software development was the Waterfall Model. Although it dates back to the 1950’s [16], the model is often associated with a 1970 paper by Winston Royce, which contained a graphical representation (see Figure 3-1) that became canonical [17]. In the Waterfall Model, software occurs in a linear sequence of phases.

1. Software Requirements, in which the necessary capabilities of the system are described.
2. Architectural Design, in which the major components of the system are established.
3. Detailed Design, in which the components are refined and design decisions regarding the implementation of components are established.
4. Coding, in which the detailed design is expressed in a formal, executable language.
5. Testing, in which the behavior of the code is verified against the requirements.

Many variations of this model have been proposed, but these five phases establish the basic sequence.

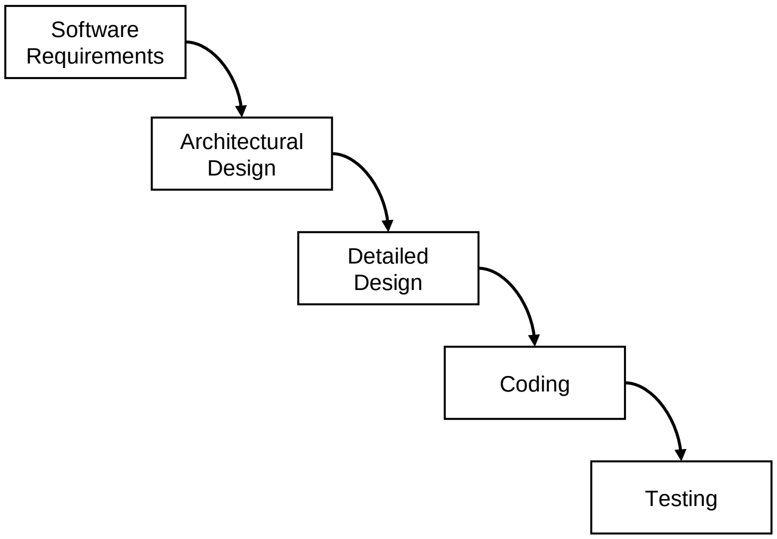


Figure 3-1. The Waterfall Model of the Software Life cycle

Royce, interestingly, didn't like the waterfall model even as he popularized it. (His paper contains a revealing quote: "I believe in this concept, but the implementation … is risky and invites failure.") He particularly criticized leaving testing until the end. No engineer would behave this way;[10] an engineer's models were rigorous enough to admit to formal evaluation as soon as they were developed. The first three Waterfall Model phases, by contrast, simply yielded unstructured documents containing natural language or pictures. Furthermore, development was never strictly linear. Architectural designers noticed problems in the requirements. Detailed designers found problems in requirements and architectural design. Coders found problems in work products from all three phases, and testers in all four. Software development was invariably more cyclic than the Waterfall Model implied. (Royce recognized this too, but the cycle-incorporating pictures he drew were more complicated and didn't catch on.)

This led to a rebellion in the 1980's, with practitioners publishing papers with such titles as "Stop the life-cycle, I want to get off," [18] Boehm introducing the Spiral Model, [19] and Parnas and Clements arguing that, however software development proceeded, the real value of the Waterfall Model was as an after-the-fact description. [20] The rebellion found a new expression in 2001 with the publication of the Agile Manifesto.[11] This manifesto, signed by experienced software researchers and practitioners, stated that their many years of experience had taught them to value:

1. Individuals and interactions over processes and tools

[10] Here, "engineer" is used in the traditional sense and refers to such disciplines as electrical and mechanical engineering.

[11] See http://agilemanifesto.org/

2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

The signatories were careful to state they did not consider processes, tools, documentation, etc. unimportant, only less important. Well thought-out processes and good tools were important in software development, but skilled and talented software developers and frequent interactions among them were (and are) more important.

The third item was particularly significant. Rather than developing to a fixed set of customer-provided requirements, it encouraged developers interacting with customers to determine whether the requirements actually satisfied their needs. This could best be done by developing incrementally, at first delivering small products that satisfied a small, but crucial set of needs and expanding on those. This allowed developers time to listen to how customers felt about the delivered products during the process and respond accordingly.

Most of the software development community agrees that, in the current milieu of software development, Agile development is preferable to Waterfall. No one claims Agile development is a cure-all, only that it is better suited to the needs and issues that arise in modern software-oriented projects. That said, Agile development did not address one of the Waterfall Model’s shortcomings. In both, the developing organization interacts with a customer, and that customer is more likely to be a contracting representative than an end user.[12] The resulting system may satisfy customer needs without considering whether the operator likes it: whether it is easy to use, has adequate help, provides necessary feedback, etc.

Out of this gap arose DevOps. DevOps complements Agile by recognizing that many systems—especially systems of interest to DoD—have an operations team. The operations team will spend time in front of computers and have the most immediate interest in a well-run, well-constructed, well-thought-out working system. The DevOps philosophy believes the most important interactions are between the operations and development teams. Only through these kinds of interactions will a system be built that satisfies its users. DevOps is sometimes referenced as “Agile done right.”[13]

DevOps prioritizes delivery, perhaps even more than Agile. Operators are often the ones who find bugs and suggest improvements. If these can be conveyed directly to developers rather than going through customers (as in Agile), a step has been eliminated and time (and probably money) has been saved.

---

[12] This statement applies to contractually developed software, which is characteristic of much government-procured software. It is less true of commercial software.

[13] See https://www.slideshare.net/TomasRiha/devops-its-just-agile-done-right, for example.

One danger of such close interaction is the potential for not adequately considering security. Operators acknowledge the need for security, but view it as a secondary concern and, often, an inconvenience. Operators want to do their job as quickly as possible, and they want tools to help them do their job as soon as possible. Neither of these desires encourages security. An enlightened operator tolerates two-factor authentication because they understand the risk of password-based authentication. Tolerance doesn't make for a happy user experience. In developers' initial urge to create a friendly system, is it no surprise that security concerns are shunted aside.

Some organizations try to balance the desire for quick delivery and the need for security by using DevSecOps. DevSecOps adds a security team to the development organization that is responsible for ensuring the codebase contains software that addresses security issues. Security is "shifted left:" security concerns start early in the software development process. This is in contrast to a surprisingly large portion of software development where security is an afterthought. There can be too much security; however, and the security team is responsible for finding the right balance.

B. State of Practice

Given the advocacy of DoD toward adopting DevSecOps, a quick look at the state of the practice is warranted.[14] This report is concerned specifically with DoD practices. Gartner conducted relevant research in the area across the entire software development industry.

In 2019, Gartner published a report with the following estimates: [22]

> By 2021, DevSecOps practices will be embedded in 60% of rapid development teams as opposed to 20% in 2019. By 2023, more than 70% of enterprise DevSecOps initiatives will have incorporated automated security vulnerability and configuration scanning for open-source components and commercial packages, which is a significant increase from fewer than 30% in 2019.

As will be discussed in detail in the next chapter, IDA's interviews and surveys suggest this may have been optimistic, at least within DoD. 50% of respondents reported using either DevOps or DevSecOps; only 20% reported using DevSecOps specifically. These numbers do not necessarily contradict Gartner's estimates: IDA's sample size was small and should not be taken as indicative of overall DoD trends. However, the participants were selected because they were involved in non-Waterfall projects. IDA asked participants which methodologies they were using; Figure 3-2 shows, for each methodology on the x-axis, the number of respondents who reported using that

[14] Much of the material in this section derives from [5].

methodology. [15] DoD may be moving away from the Waterfall model, but it is still far from fully converting to DevSecOps.

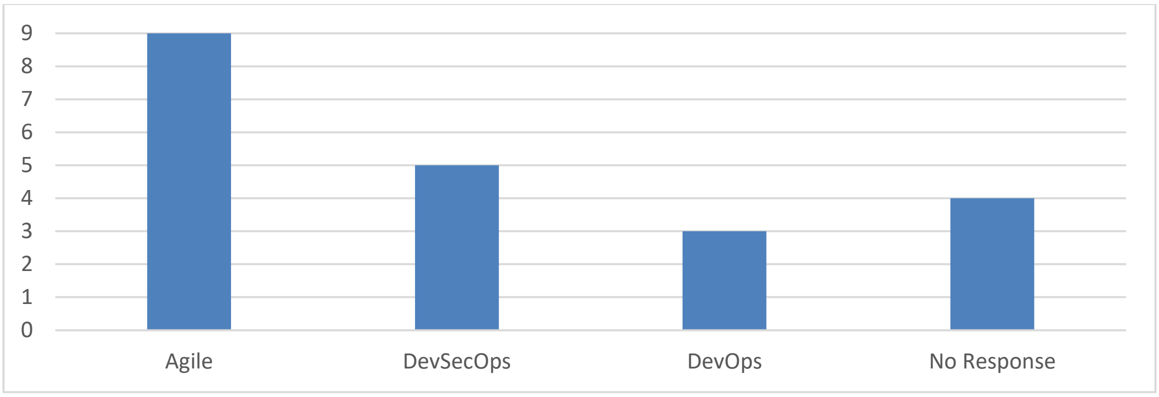


Figure 3-2. Number of projects using depicted methods

Whatever the trends in DoD projects may be, many open source software providers and commercial vendors have devoted considerable effort to developing tools able to contribute to automating testing. Automated testing being fundamental to DevSecOps, this can only be seen as a sign that the software development community is pushing the technological innovations necessary to make DevSecOps practical. A Gartner report from 2021 identified several companies with Application Security Testing (AST) products in varying stages of maturity. [23] This report, presented using Gartner's Magic Quadrant format,[16] groups companies into four categories, shown in Figure 3-3. The results showed five leaders, three challengers, three visionaries, and three niche players.

[15] The count of methodology uses exceeds the number of respondents because several respondents reported using multiple methodologies.

[16] See https://www.gartner.com/en/research/methodologies/magic-quadrants-research

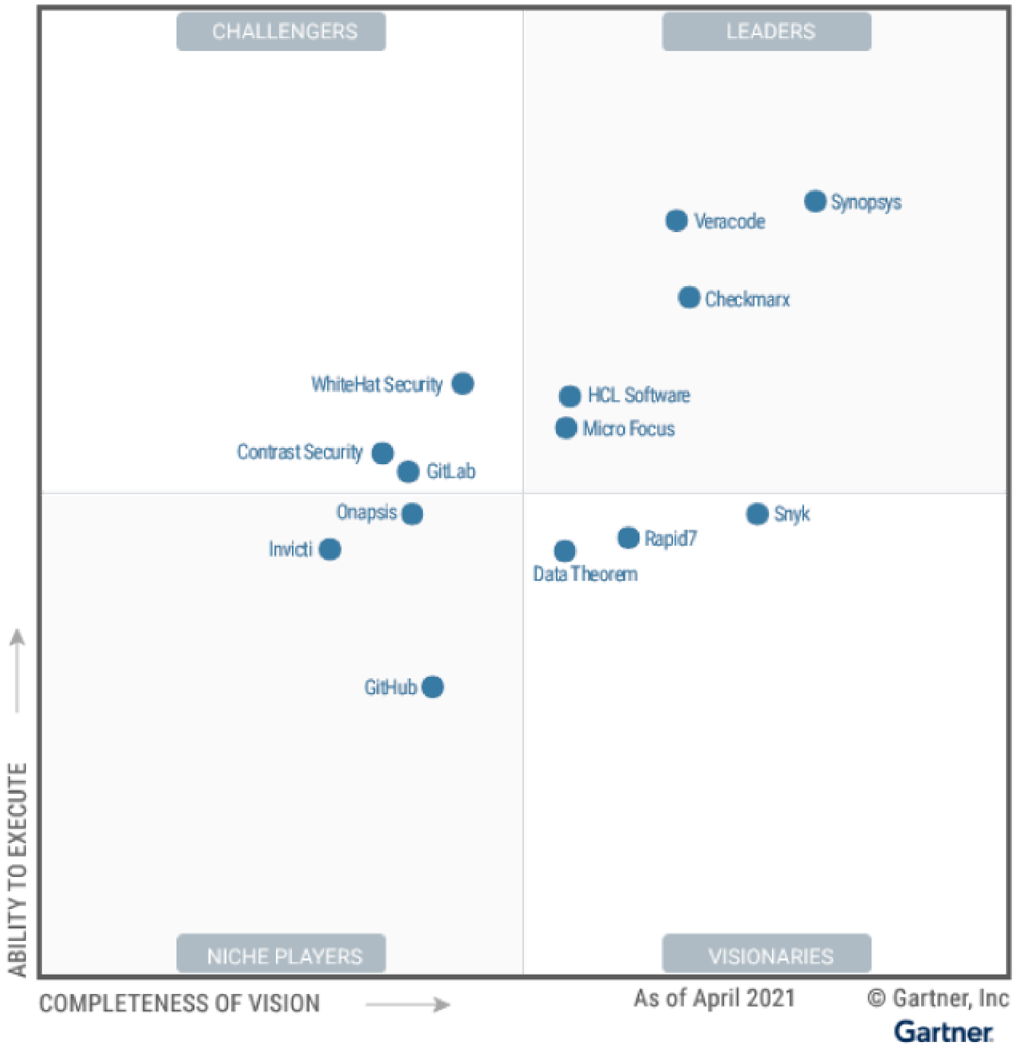


Figure 3-3. Magic Quadrant

The Gartner report did not cover every automated testing tool, nor did it intend to. It had strict criteria for inclusion. Moreover, being something other than a leader is not necessarily a flaw. Challengers may have a long-term strategy worth following; visionaries may focus on a hitherto-unexplored area; niche players have a narrow focus that may be totally appropriate for a given need. The conclusion was that in 2021 there were 14 AST products well worth considering.

Gartner uses a "hype cycle" graph (Figure 3-4) to illustrate technology trends.[17] It shows some event that triggers initial excitement—too much, peaking in inflated expectations. These expectations give way to disillusionment, followed by eventual realization of what the technology is actually good for, as well as how and why the

[17] The image is from Wikipedia: https://upload.wikimedia.org/wikipedia/commons/thumb/9/94/Gartner_Hype_Cycle.svg/1024px-Gartner_Hype_Cycle.svg.png

technology is good. These realizations culminate in a plateau of productive use, with visibility below original expectations, but nevertheless significant.

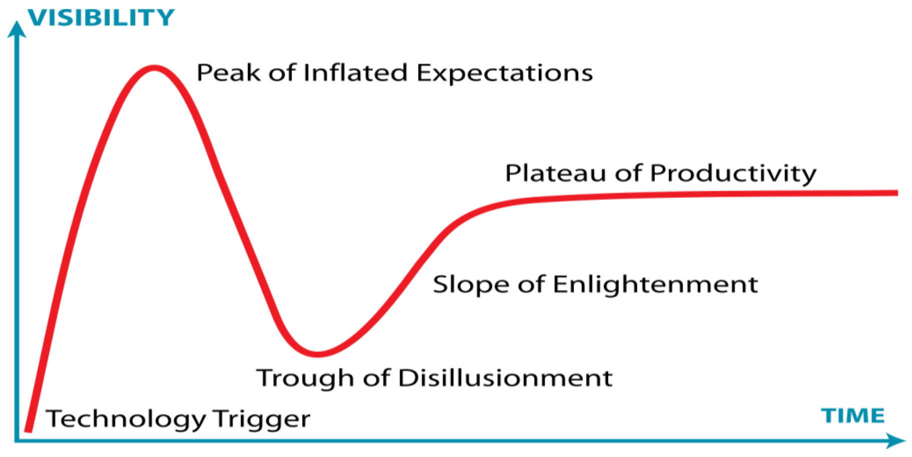


Figure 3-4. Hype Cycle

Gartner produced a hype cycle graph for application security in 2021, shown in Appendix C. [24] It shows DevSecOps midway up the slope of enlightenment and predicts its maturity in two to five years. Interestingly, the graph shows some foundational DevSecOps technologies further behind on the slope of enlightenment. Neither DevOps Test Data Management nor API Security Testing has reached the Peak of Inflated Expectations. Container Security is just beginning its rise up the slope of enlightenment. Undoubtedly, some areas of DevSecOps need further research, development, and adoption before the entire field can achieve maturity.

(This page is intentionally blank.)

4. Findings

As described in 2.C and Cybersecurity and Software Development Survey IDA developed a set of interview/survey questions covering five categories: (1) context questions, (2) DT&E integration questions, (3) team and environment questions, (4) development process questions, and (5) test process questions. IDA received 18 written or virtual interview responses to the survey. The number was constrained by challenges imposed on the work environment due to COVID, which prevented any in-person contact. Eight responses were from the developmental test (DT) community and four from the operational test (OT) community. The remaining six responses were from the development side of the house. (IDA promised non-attribution, so this report deliberately obscures the organizations involved.) This skewed balance was not surprising due to the survey emphasis on integration of developmental test into Continuous Integration/Continuous Deployment (CI/CD) methodologies. Survey respondents were also weighted to the Air Force with six from the Air Force proper, and four more from Space Force. We received four responses from the Army, and three from the Navy that also included input from the Marine Corps. Again, this proportion reflected the Air Force center of gravity as the DoD-designated lead in bringing DevSecOps to the community. The survey responses were further rounded out by documentation received from the DevSecOps Community of Practice monthly meeting files,[18] the Air Force CSO DevSecOps document repository, and the Navy.[19] The following section presents our findings in light of the state of DevSecOps practice in DoD. We highlight success stories from the responses, identify problem areas, and then close by discussing lessons learned.

[18] The DevSecOps COP monthly meeting is currently conducted in the DoD Teams Collaboration Virtual Resource (CVR) at this link https://teams.microsoft.com/l/meetup-join/19%3a4d92705edf2e40e2a54a7163f0b24d05%40thread.skype/1588014888844?context=%7b%22Tid%22%3a%2221acfbb3-32be-4715-9025-1e2f015cbbe9%22%2c%22Oid%22%3a%2244e424f7-0bcc-4c72-97d7-22160ced090d%22%7d. The files section and the associated site hosted by the USAF CSO at https://software.af.mil/ contain numerous briefings and background documents on all aspects of DevSecOps adoption in DoD.

[19] Navy Information Warfare Center Atlantic Brief: DevSecOps Maturity and Business Value.

A. DevSecOps Pipeline Practices

The traditional software development model involves delivering updates at fixed intervals. For example, a company will schedule delivery once every quarter, and provide whatever updates, enhancements, and bug fixes it has completed during that time.

As with DevOps, DevSecOps' philosophy states a product will undergo CI/CD. Changes are released into production as soon as they have been approved, no matter how large or small. Instead of issuing a release once a quarter, DevSecOps projects commonly release new versions many times each day.

This kind of schedule requires automating much of the development process. The traditional model calls for manual reviews and often prescribes supporting documentation (e.g., test results) to be presented at those reviews. It simply isn't practical for an organization to hold these reviews if it wants to release changes quickly. Too much scheduling and coordination are required. The only alternative is to automate as much as possible.

DevOps emerged from the realization that much of the testing, integration, and deployment could be automated once the code completion was formulated. Developers were asked to estimate completion, and reviews were scheduled accordingly. In DevOps, the act of completion triggers the automated reviews, wasting no time.

DevSecOps adds to DevOps by incorporating security-related assessments and incorporating them early. Whereas in DevOps security was (and is) relegated to the final stages of the development pipeline, DevSecOps goes to great lengths to ensure security is considered early. Some of this consideration necessarily involves human interaction, not automation. In particular, during planning stages the development, security, and operations teams interact to establish the nature and scope of security-related testing. With that accomplished, the objective is to design a pipeline, or enhance an existing pipeline, to evaluate security as early as possible, and as automatically as possible. This lowers the likelihood of a common DevOps problem: discovering a security flaw during a late life cycle activity such as penetration testing, thereby forcing developers to fix bugs that could have been detected much earlier. (And bug fixes, depending on a bug's nature, can require reconsidering significant aspects of the entire software architecture.)

IDA's interviews and surveys showed:

- Widespread use of Fortify[20] and SonarQube,[21] two security-focused commercial Static Application Security Testing (SAST) tools. (SonarQube is available in four configurations, one free and open source, the others

[20] https://www.microfocus.com/en-us/cyberres/application-security

[21] https://www.sonarqube.org/

proprietary. IDA did not inquire which configurations were being used.) Both tools can and were designed to be integrated into a pipeline. This allows developers' code to be scanned early in the software life cycle, prior to testing or deployment into staging environments.

- There were no references to Dynamic Application Security Testing (DAST) tools. DAST, unlike SAST, requires compiling, building, and running source code, and is typically harder to specify, implement, and perform. No respondents reported using DAST tools (e.g., fuzzers[22]), let alone integrating DAST into pipelines.
- Some use of containers. Three subjects reported using Docker.[23] Two subjects reported using Kubernetes[24] together with Tanzu[25] to manage multiple communicating containers.
- Consistent use of multiple environments (development, staging, release). Multiple environments are fundamental to DevSecOps: They ensure each developer, each tester, and each operator can work without concern that an action by someone else will affect them. Development is typically done in Impact Level 2 (IL2), production in IL4 or higher.[26] In other words, systems that are deployed in classified environments can be developed in unclassified environments, giving developers and testers access to a broader range of tools than are available in IL4.
- Consistent use of repositories, including GitHub, GitLab, Bitbucket, IronBank, and Cloud One. That is to say, everyone uses repositories, but different projects use different repositories. Some projects use multiple repositories, others use a single repository. The projects that used multiple repositories did so because they develop in unclassified environments, but deploy to classified environments. In such cases, there are equivalent repositories in each.

Interview and survey results indicate DoD projects are in the process of building automated pipelines, but have more work to do. The following paraphrased example

[22] https://owasp.org/www-community/Fuzzing

[23] https://www.docker.com/

[24] https://kubernetes.io/

[25] https://tanzu.vmware.com/tanzu

[26] For a discussion of impact levels, see [25], a 2015 IDA report covering the topic.

responses (these apply to individual projects, not all DoD programs surveyed) illustrate this

- Although several responses indicated the use of SAST tools by name, some respondents indicated they were not yet performing static testing with automated tools.
- In another case, only the security pipeline, not the coding pipeline, had been automated.
- Continuous penetration testing does not appear to have been performed.[27]
- The amount of non-functional testing and security performed depends on the program.
- Most testing is performed at the unit level.

On a positive note, one respondent said they do "all the cool automated things" in development and deployment pipelines. Clearly, that person believes their project is practicing DevSecOps, although other answers cast some doubt.

B. DevTest and Secure Coding Practices

DevSecOps in DoD is tightly coupled with adoption of cloud development environments. Many potential cloud technologies can be employed in development and fielding of defense systems, including public, private, community, multi-cloud, and hybrid tenancy solutions.[28] Although strictly speaking these offerings fall outside the scope of this discussion, cloud service offerings (CSOs) are part of the underlying development and fielding environment, and require some level of cyber DT&E. Adding to the complexity of the assessment problem is the use of different potential service level models, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS), and their different permutations. With few exceptions, DoD mission systems will be developed and fielded in environments certified to host sensitive or classified national security data due to limitations imposed by Federal Risk and Authorization Management Program (FedRAMP)[29] and the Defense Information Systems Agency (DISA) Cloud Computing Security Reference Guide (SRG) [30] under the Defense Acquisition Regulation

[27] Penetration testing requires specialized skills, which perhaps were not available.

[28] See https://www.abacusnext.com/blog/whats-difference-between-public-private-hybrid-and-community-clouds/ for a discussion of different cloud types.

[29] FedRAMP was established by a December 8, 2011 memorandum, from the Federal Chief Information Officer, titled *Security Authorization of Information Systems in Cloud Computing Environments*. See https://www.fismacenter.com/fedrampmemo.pdf.

Supplement (DFARS)[30,31] for hosting the specific data impact levels of DoD data. Offerings by only a handful of cloud service providers have passed the review at this level.[32]

The commercial cloud cyber assessment process required by FedRAMP and DISA Provisional Authorization (PA) as dictated by the Cloud SRG is heavily Risk Management Framework (RMF) [10] controls-focused and prohibits in-depth technical validation, especially of the consumer configured portions of a service offering. Additionally, DT&E requirements fall outside the provided guidance unless those requirements duplicate RMF activities.[33] The CSO used to host the DevSecOps environment provides the outer boundary of the set of nested code-based objects. This includes the Infrastructure as Code (IaC)-defined environment, pipeline, and virtualized elements such as containers that define the final software product and provide inheritance for risk management. Each of these must be assessed in turn to provide the foundational level of cyber assurance to support the next layer up in the development and production chain. Conceptually, if adequate processes and mechanisms are in place, such as guardrails, secure repositories, and architectural hoisting [27] to provide appropriate testing and evaluation of each inherited level, the DevSecOps development team can focus on applying secure coding and testing practices to the incremental sprints in the DevSecOps cycle.

With a well-constructed foundational infrastructure, sufficient guardrails can minimize the risks of vulnerabilities introduced during the development process. Additional techniques can further reduce vulnerabilities during development. Unfortunately, our interview respondents were primarily not developers, and the responses failed to address these mitigations in detail. However, they are known within the computer science community. Type safe programming languages can diagnose some vulnerabilities during the compilation process.[34] The use of memory-safe languages like Java, Rust, or Python can avert vulnerabilities stemming from errors like buffer overflows.[35] Within

[30] DFARS Clause 252.239-7010, *The Contractor shall implement and maintain administrative, technical, and physical safeguards and controls with the security level and services required in accordance with the Cloud Computing Security Requirements Guide (SRG)* – “most recent version.”

[31] DFARS SUBPART 239.7602-1, Cloud Computing, *A Cloud Service Provider (CSP) must have a DoD Provisional Authorization (PA) at the appropriate Information Impact Level (IIL) before contract award.*

[32] DISA Cloud Service Catalog, Available https://www.disa.mil/-/media/Files/DISA/Services/Cloud-Broker/Authorized-DoD-Cloud-Services-Catalog-SEPT2018.ashx?la=en&hash=4392BC616B34E381F52FC7E9CDB53231672E05A3 Accessed: October 23, 2018.

[33] See [26] for more detailed discussion of testing in the cloud versus testing of the cloud.

[34] https://en.wikipedia.org/wiki/Type_safety

[35] https://owasp.org/www-community/vulnerabilities/Buffer_Overflow

specific languages, there are libraries and functions that can be used (or not) to avoid common errors. For example, one can avoid the use of C’s malloc function, or use a wrapper for that memory allocation command in which the wrapper ensures specific guarantees of safety are met.[36] In Java, a stored procedure can be used rather than a traditional SQL (Structured Query Language) query, avoiding SQL injection errors.[37] When all else fails, pair programming is used in some Agile settings: two developers program together side by side, one coding while the other checks the code for vulnerabilities.[38] All of these can be used in conjunction with a DevSecOps approach.

Unfortunately, the survey responses do not instill confidence that the above techniques will be selected as needed. A number of responses hinted that security requirements generation—particularly at the big picture level—is lacking. A top-down, risk-based approach is needed to architecting a secure development process. This is the architecture for the architecture (and subsequent development). For example, a high-performance system in a domain such as avionics may be unable to tolerate the overhead of a memory-safe language like Java. However, a memory-safe subset of the C language could be used in conjunction with pair programming to mitigate the likelihood of some vulnerabilities. These trade-offs require competent program management that understands the options available and can select the trade-offs that best meet the needs of the system to be developed. Ideally, the rationales for key decisions such as security requirements sources and secure development choices are clearly articulated, documented and traceable to their origins. Documenting and maintaining this kind of information may seem against the Agile principle of valuing working code over documentation. But Agile does not say documentation is unimportant. An organization must make an up-front decision what information it needs to ensure security and commit to that decision throughout development.

Others have compiled sources of secure coding practices. A paper published by National Institute of Standards and Technology (NIST) [28] is an excellent starting point.

C. Reported Successes, Problem Areas, & Lessons Learned

This section discusses the survey results, splitting them into success stories, problem areas, and lessons learned. The survey asks respondents to state what is working well and what is not working well, and to share why, so it was straightforward to separate successes from problems, and to extract lessons learned. Table 4-1 summarizes the results.

---

[36] https://owasp.org/www-community/vulnerabilities/Memory_leak

[37] https://owasp.org/www-community/attacks/SQL_Injection.

[38] https://en.wikipedia.org/wiki/Pair_programming

Table 4-1. Survey Results

Topic number	Commonalities Reported by Respondents	Number of Responses
	What is Working Well (Success Stories)	
1	Enculturation of DevSecOps is foundational	8
2	Incorporate test processes and environments up front	6
3	Fully leverage automation for testing, pipeline, and builds	8
4	Results can be validated	4
5	DevSecOps can be adapted to physically isolated environments	2
	What is Not Working Well (Problem Areas)	
1	The current DoD acquisition model does not lend itself to a DevSecOps development methodology	3
2	The current DoD Authority To Operate (ATO) process is partially incompatible with DevSecOps	4
3	Many DoD systems may be incompatible with the DevSecOps process	6
4	There is no adequate standard definition of DevSecOps	4
5	The role of DT within a DevSecOps methodology is unclear	6
6	Forming teams proved more challenging than expected	4
7	Organizations had trouble allocating resources	4
8	The difficulties of classified versus unclassified DevSecOps development are unresolved	5
	Lessons Learned	
1	Starting with DevSecOps is easier than switching to DevSecOps	6
2	DevSecOps is easier on small projects than big ones	5
3	DevSecOps must be adapted to the operational environment	6
4	Automation is key	8
5	Security requirements need to be more mission-focused	7
6	Leverage good architecture and design to support security	6

1. Success Stories

Respondents provided a variety of responses on what worked well, but they clustered around a similar set of themes, which can be contrasted with the responses on what caused pain points. Many of the advantages can be directly attributed to the maturity of the specific DevSecOps environments, with better results for those organizations that invested in developing the needed skills, experience, and acculturation.

1. **Aggressively establishing and promoting a DevSecOps culture.** This ensures everyone understands the system from a user's perspective, the importance of

implementing security early, and the tools and processes necessary to do so. DevSecOps process training pays off for the entire team: leadership, PM, product leads developers, testers, and operators. Leadership learns the culture and process of DevSecOps, including the limitations and reasonable expectations. The same is true of the rest of the team. Through constant interaction, all the team members have the same understanding and expectations. All are involved in creating the user stories. Developers understand and accept the importance of building in feature security from the beginning. Testers embed with developers and are continuously involved. Testers are aware of the environment and workflow and able to successfully tailor, automate, and integrate tests into the pipeline. Operators play an essential role as the final arbiters of the success of the features added during a sprint.

2. **Planning and incorporating testing processes and environments up front.** If testing is defined as part of the requirements process, instead of an afterthought to development, the same incremental improvements may be leveraged as in the coding pipeline process, speeding up the testing process when it occurs. Another advantage is the target of opportunity provided for line testing. In traditional DT body of practice, the resources for a dedicated developmental test event must be made available. That includes mature code, a suitable test environment, and an available properly skilled team. Conversely, DevSecOps requires DT shift as far left as possible. Since the DevSecOps environment must be architected to allow for continuous testing, waiting for availability of a separate dedicated test environment is not an issue. Throughout the series of sprints, as the system is spun into production, the automated tests are executed, and coding defects are discovered as part of a continuously repeating process that provides real-time reporting and feedback.

3. **Fully leveraging automation for testing, pipeline, and builds.** Increased automated scanning helps reveal coding errors early and automated builds enable remediation. Automated scanning and continuous authorization of containers via proactive remediation of findings in the development pipeline reflect a fundamental change in how applications are secured. Vulnerabilities are removed from production applications faster, as much of the work to rebuild base containers has been automated. Once secured, they remain secure as long as they are monitored for drift and rebuilt frequently to return them to the secure baseline state. Only one of the respondents referenced use of a DoD repository of previously assessed and approved code objects (as opposed to widespread use of repositories in general) such as Iron Bank. Iron Bank[39] or formally, the DoD

[39] https://software.af.mil/dsop/services/ available 07/16/2020

Centralized Artifacts Repository (DCAR), is a DoD repository of digitally signed, binary container images including both Free and Open-Source software (FOSS) and Commercial off-the-shelf (COTS) sponsored by the Air Force. All artifacts are hardened according to the DISA Container Hardening Guide, and containers accredited in Iron Bank have DoD-wide reciprocity across classifications. Although the benefits for speed, security, compliance, and cost seem self-evident, the respondent's organization did not use it in favor of its own repository. (The respondent indicated that their organization preferred a repository they could control themselves.) Nonetheless, even regarded purely as a remediation against supply chain attacks for open source components, the development of these repositories is a clear success story for DoD.

4. **Objectively demonstrable results.** Several respondents report these improvements can be objectively verified through metrics. Although the challenge of developing cybersecurity metrics was also identified by respondents, the reduction of discreet common vulnerabilities and exposures (CVEs)[40] in production scanning was considered an indicator of the effectiveness of DevSecOps in reducing vulnerabilities. This effectiveness is compared with a legacy development and production environment, where the same CVEs may live in production for several months until the next periodic scan results in a Plan of Action and Milestones (POA&M) to remediate them. And that is only the lead time to generate the POA&M; the actual fix will need to go through the next production cycle. With DevSecOps, the cycle time needed to address the same vulnerabilities is greatly decreased.

5. **Adapting DevSecOps to physically isolated environments.** Survey results indicate Agile methodologies have also been effectively adapted to support code development for air frames, classified systems, and other types of applications, where the development and production environments are physically isolated from each other. These environments present challenges for employing automation to build, test, and promote across boundaries, but workable technical solutions exist and have been successfully leveraged, according to respondents. These observations were counterbalanced by examples from other respondents where these problems had not been overcome. Mixed mode environments require a greater degree of flexibility to succeed, but also a greater commitment to process discipline.

[40] https://cve.mitre.org/ available 07/11/2021

2. Problem Areas

Interviewees had plenty to report on problems using DevSecOps. Indeed, measured by the many responses, they had more to say about difficulties than successes. This should not be taken as a failure on the part of DevSecOps. It is still relatively new and certainly not as mature as some other, more familiar models. Every new employee requires more than training, also inculcation into the DevSecOps mindset.

Nevertheless, the reported problem areas bear enumeration and examination. Some have obvious, if not necessarily quick, fixes. We expect others will require more detailed study, involving data gathering and analysis, to consider alternatives and provide empirical evidence about what solutions are superior and the circumstances under which they can be employed.

Each respondent had their own unique way of expressing problems using DevSecOps, but an examination of the entire set revealed some common themes.

1. **The current DoD acquisition model does not lend itself to a DevSecOps development methodology.** Instead, requirements and milestones are agreed upon up front, and the development is more baked-in than would be optimal for DevSecOps, in which the development process allows for more fluid sequencing of backlog items.
2. **The current DoD Authority to Operate (ATO) process is partially incompatible with DevSecOps.** Although the dream of Continuous ATO (CATO) is alive and well, there is no consensus on the details. Roles and responsibilities of personnel involved in the CATO process are unclear, even by key personnel. There is legitimate skepticism by some of the respondents of the security value of a CATO (other than as a compliance exercise) without grounding in a broader, more in depth periodic assessment of the maturity and consistency of the development environment and practices of each specific software factory, comparable to the Capability Maturity Model Integration[41] (CMMI) certification program (CMMC)[42] being fielded for assessing the Defense Industrial Base.
3. **Many DoD systems may be incompatible with the DevSecOps process.** Systems are currently tested just prior to deployment into production environments, and the concepts of A/B testing and continuous deployment—particularly into classified or air gapped environments—are often contraindicated.

[41] https://acqnotes.com/acqnote/careerfields/capability-maturity-model-integration-cmmi

[42] https://acqnotes.com/wp-content/uploads/2014/09/CMMC-Securing-the-DoD-Supply-Chain-Overview-Nov-2019.pdf

4. **There is no adequate standard definition of DevSecOps.** Existing guidance is still high-level and not necessarily expressed in a form that can be tailored to meet a particular project's needs. One respondent noted that the Department of Defense Instruction (DoDI) 8500 series of documents has not been amended to enable DevSecOps and automated pipelines, leaving local organizations responsible for tailoring and reinterpreting traditional compliance in environments that implement DevSecOps. Then too, some respondents did not understand the differences between DevSecOps, DevOps, and Agile. They used tools potentially helpful for promoting CI/CD, but paid insufficient attention to DevSecOps' tenet of addressing security early. It remains unclear how thoroughly and accurately system architectures are documented and maintained; similar concerns exist regarding how carefully feature roadmaps are sequenced to avoid the implementation cross-cutting changes in later stages of development.

   This lack of a standard definition has kept training materials and documentation project- or organization-specific, rather than widespread and available. Several respondents pointed out the lack of training and consequent difficulty of bringing members onto a team. Compounding the issue, implementation guidance is lacking for DoD-specific use cases, in which, for example, workarounds would be provided for inability to implement CI/CD. The fact that DevSecOps is not fully part of the culture yet does not help. One respondent said his organization has a playbook and checklists, but no procedural or technical way to ensure teams are using either. Presumably he would not have made the same comment about Waterfall-based development. His organization hasn't figured out how to implement analogous feedback and review mechanisms for DevSecOps.

5. **The traditional role of DT within a DevSecOps methodology is unclear.** As stated earlier, it is unclear how DT fits within DevSecOps. In DT as currently practiced, tests are performed at a later stage in development rather than at times of code check-ins or during CI and CD. Tools used and artifacts generated during DevSecOps testing may be incompatible with – or contractually unavailable to – the DT team. It is possible to shoehorn in Developmental Testers during the DevSecOps process to guide the testings' compatibility with the final DT, but there is still a "big bang" DT that is inconsistent with how the DevSecOps process works.

6. **Forming teams proved more challenging than expected.** This was partially due to COVID-19: developers suddenly started working at home, complicating pairs programming and the developer-tester-user interactions that are so important to DevSecOps. Organizations also encountered difficulty for other reasons. For example, development and testing shops are often physically distinct organizations. Having developers and testers work together means bringing one

or the other to a new site for several months, raising quality-of-life and manpower issues. One operational testing organization had a contract with the USAF 47th Cyberspace Test Squadron to do continuous testing, but for political reasons each wanted its own red team.[43] Bringing team members together was not the whole problem. One respondent stated their test sites were in remote locations where recruitment was difficult, and bemoaned the lack of talent. Conway's Law states, "Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure."[44] In this case, the separation of testers into separate organizations can easily lead to incompatibilities in testing approaches, separate tooling, mismatched timelines, and unaligned goals. It is important to recognize that these problems will continue when developers, testers, and users are part of different organizations—DoD, contractors, and subcontractors, from the program level on down.

7. **Organizations had trouble allocating resources.** This may reflect unfamiliarity with setting up DevSecOps, in which case the problem can be expected to go away, or at least lessen, in the future. Nevertheless, it is worth noting the following responses:

   a. More up-front resources are required to implement DevSecOps. Developers, designers, and security experts get involved earlier than in traditional projects, and they quickly purchase and provision computing environments that previously would have been unnecessary for months or even years. One respondent particularly stressed the need to rely more on contractors and to invest more in the DT team. Given the challenge of finding sufficient security testers with the necessary skill sets for DT, the requirement to add dedicated security experts to the development teams compounds the problem.

   b. The community has not settled on approved tools for practicing DevSecOps. Projects implement what their team members know, so the tools for issue reporting, CI/CD, version control, containerization, etc. are still unsettled. This decreases integrability, or at least increases the effort required to achieve it.

   c. One respondent, familiar with Agile, complained that Agile programs own their code, but from lack of resources often struggle to keep it secure, operationally suitable, and operationally effective.

8. **The difficulties of classified versus unclassified development in DevSecOps are unresolved.** Several respondents spoke to how software developed in

[43] The reasons are not stated because they might reveal the respondent's identity.

[44] https://www.melconway.com/Home/Conways_Law.html

unclassified environments was transferred to classified environments for operation. This is a perennial problem, and certainly not specific to DevSecOps. DevSecOps does introduce its own challenges. The infinity loop diagram showing the process flow doesn't work when some high-side information cannot, by law, be transferred to a low-side environment. Automated cross domain transfer between a low side development environment and a high side production environment also presents a challenge. At best, the extra data flow is time-consuming. At worst, classified tests and high-side simulations can't exist on the low side, and the DevSecOps paradigm requires significant alteration to develop a workaround.

3. Lessons Learned

This section covers some lessons IDA gleaned from the surveys and interviews.[45] Characterizing them was difficult. The surveys and interviews yielded diverse results, each participant having their own story to tell. Our conclusion was that their experiences reflect their environment, their projects, their resources, and their familiarity with DevSecOps and its antecedents (Agile and DevOps). This is a lesson in and of itself: successful and widespread use of DevSecOps will require more standardization, resources, and training. There were also more specific lessons learned, and they are the focus of this section.

1. **Starting with DevSecOps is Easier than Switching to DevSecOps.** Some respondents attempted to switch legacy projects from a Waterfall process to DevOps or DevSecOps. Respondents agreed this was more difficult than if the project had begun with DevSecOps. Of course, they had no choice, but their consensus emphasizes the cultural changes that need to occur for DevSecOps to succeed,[46] and that change of culture was especially hard when introduced midstream, when the usual pressures to deliver were already present. In the Waterfall Model, teams often work in siloes; DevSecOps assumes close collaboration. Geographically distant teams were suddenly required to be together. The logistical challenges were obvious. So were the environmental challenges, where everyone must adapt to a new approach to obtaining artifacts and submitting changes. DevSecOps, or more precisely CI and CD, requires more automation than a Waterfall-based process. Setting up the CI/CD pipeline takes time and resources, as does learning to work with it.

[45] These lessons learned are not to be confused with survey question 7. They include answers to that question and to others as well.

[46] Many have written that successful adoption of approaches like Agile, DevOps, and DevSecOps require cultural change. See, for instance, https://www.agileconnection.com/article/7-ways-change-culture-devops-success and https://www.bmc.com/blogs/devops-culture/

2. **DevSecOps is Easier on Small Projects than Big Ones.** Respondents had difficulty sticking to DevSecOps as project size grew. Bigger projects tended to "fail back to Waterfall"—that is, large projects perceived that adhering to DevSecOps principles risked failure and, although claiming to be practicing DevSecOps, were mainly using Waterfall-based processes as far as the developers and testers were concerned. IDA did not have the opportunity to explore the reasons for reversion. It is easy to speculate. Larger projects require more training: more individuals who must undergo a cultural shift. Larger projects are more complex, and introducing DevSecOps only adds to that complexity until the cultural shift is complete. Larger projects have more potential communication channels; as DevSecOps assumes more communication than Waterfall, more of these channels must be used, increasing the amount of information everyone has to process. DevSecOps consumes more resources up front (several respondents noted this). Furthermore, the security-related problems DevSecOps finds aren't traceable to functional requirements, and it can be difficult to convince cost-conscious management (especially management that hasn't made the cultural shift) that the extra resources are justified.

   These challenges are not endemic to DevSecOps. Netflix, certainly a company that develops and deploys many large and complex systems, has long used DevOps successfully.[47] Then again, DoD's contracting model, and the resulting oversight and scrutiny, complicates any kind of change. IDA hopes the problems DoD has encountered when using DevSecOps on larger systems will diminish once DoD and its contractors gain experience. IDA recognizes that overcoming some challenges may require modifying existing contracting law.

3. **DevSecOps Must Be Adapted to the Operational Environment.** Respondents had projects in their portfolio that ranged from weapons systems to space C2 systems to IT systems. Operator interactions with each system category differ greatly: in authentication and authorization, involvement (a spectrum from reactive to proactive), and monitoring. Effective DevSecOps practice requires collaboration between developers and operators, and consequently an understanding by developers of their target operational environment. The different environments preclude a one-size-fits-all approach to DevSecOps. The easiest illustration of this is the difficulties of developing software systems in unclassified environments (which most respondents preferred) that are operated in classified environments. Staging and production environments, even if they are containerized, must operate according to the different constraints imposed by classified systems. There is also the matter of transferring products from

[47] See https://netflixtechblog.com/tagged/devops?gi=53aa556e6afa

unclassified to classified systems, something that regulations make difficult to automate fully.

4. **Automation Is Key.** Individuals and interactions are valued over processes and tools, according to the Agile Manifesto values, but CI/CD does not happen without automation. Much of CI/CD is rote compilation, assembly, and distribution of artifacts. These actions need no human involvement, and indeed, were humans to be involved, would likely be error-prone, humans being less adept than computers at following directions exactly. IDA's results show DevSecOps practitioners recognize this, and strive to identify mechanical parts of their processes that can be automated. Several respondents made extensive use of automation in setting up and provisioning environments through containers and virtual machines (VMs) (and collections of containers: Docker swarms, Kubernetes pods). Respondents also performed code reviews automatically, using such scanning tools as Fortify and SonarQube. As with the operational environment (lesson learned #3), a project can expect to tailor its automation based on the pipeline it implements. The set of tools will vary based on the product, project, and environments used in DevSecOps.

5. **Security Requirements Need to Be More Mission-Focused.** Respondents, when asked about how they obtained and tested to security requirements, mentioned such standards as the RMF[48] and Open Web Application Security Platform (OWASP). Although these are excellent works, they are generic guidance not tailored to the needs of individual projects. They do not focus on Mission Assurance: the need to develop security requirements starting from mission objectives. In effect, respondents indicated security requirements came from working backward. They started with (say) RMF controls, identified those relevant to their project, then determined how to fit each control into their development and testing process. This approach assumes the standard(s) they chose addressed every security-related aspect of their mission. It would be better to start with mission requirements, derive an operational perspective, then analyze that perspective to identify potential security flaws and lapses. This is not to say that RMF should be ignored, but RMF as currently practiced in DoD must be adapted to address the continuous nature of DevSecOps.[49] Using an operational perspective helps identify and eliminate operational flaws, but not necessarily

[48] RMF results from a NIST effort to provide a process for managing security risks. See https://csrc.nist.gov/projects/risk-management/

[49] For example, Air Force use of a git repository for security package artifacts in support of automated DevSecOps processes rather than Enterprise Mission Assurance Support Service (eMASS), the DoD RMF workflow and knowledge system. Both the Air Force and the Navy have published their own Continuous RMF Playbooks.

implementation flaws. Both the mission and system implementation must be analyzed.

6. **Leverage Good Architecture and Design to Support Security.** Following the Agile Manifesto does not preclude architectural documentation. However, advance planning is necessary to create sufficient architectural documentation along with a plan for how to implement various aspects of the development process in a secure manner. Interview results gave an impression that some developers lacked a mission-based perspective of the systems they were developing, leading to myopic and misdirected development efforts. And, as mentioned previously, the security requirements were often untraceable to mission-based risk assessments. It is possible to leverage Agile development with DevSecOps in a manner that anticipates and mitigates risks, but it requires subtle changes from current approaches. First, architecture is still important to develop and document. However, comprehensive architectural documentation may be overkill. Software development researcher George Fairbanks advocates for "just enough software architecture." [29] In the context of risk, he states:

   …you should use risk to help you decide how much architecture planning to do. Basically, do enough architecture until the risk of technical failure (eg [sic] from not thinking through the problem well enough, from not coordinating teams on a common engineering task) is lower than non-technical risks (eg [sic] time to market, risk of building the wrong thing).

   Said simply, it seems like common sense. In practice, teams tend to do WAY too much up-front work (see 1980's waterfall processes) or WAY too little (see 2010's agile processes). The risk-driven model gives more guidance than simply "Big Design Up Front" or "You Ain't Gonna Need It."[50]

   Architecture and design do not simply encompass what the fully-built system will look like. Instead, when properly done, they provide temporal guidance on how to sequence the development in smaller phases (e.g., through the development of a "walking skeleton"[51] to demonstrate limited end-to-end capabilities). A carefully thought out sequencing of development can reduce the likelihood that architecturally cross-cutting features (e.g., security) are not bolted on mid-way in

[50] https://www.georgefairbanks.com/software-architecture/risk-driven-model/

[51] A walking skeleton was described by Cockburn as "a tiny implementation of the system that performs a small end-to-end function. It need not use the final architecture, but it should link together the main architectural components. The architecture and the functionality can then evolve in parallel." See http://alistair.cockburn.us/Walking+skeleton.

development in a way that is disruptive to the development process and risks the security of the system.

(This page is intentionally blank.)

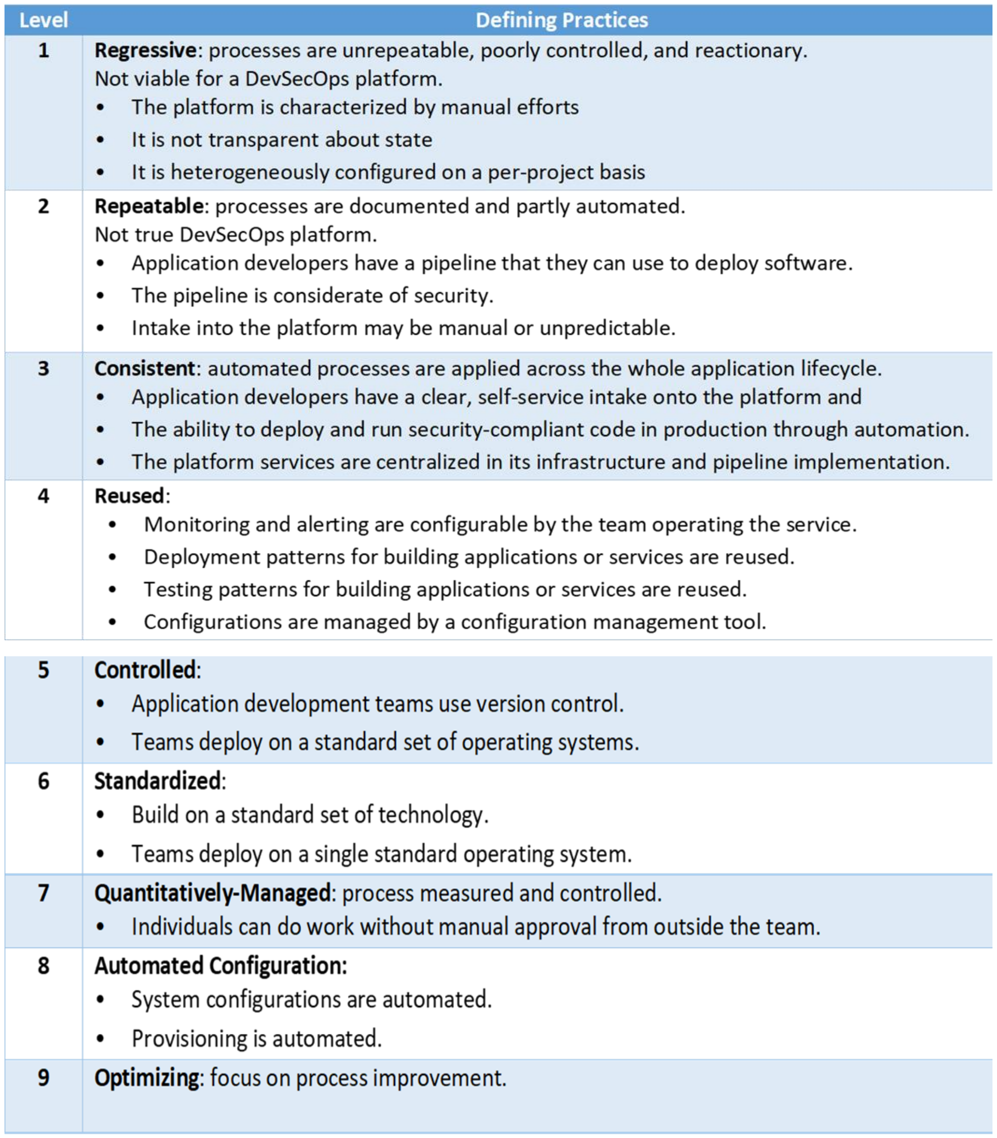
5. DevSecOps Maturity Models

Interviews consistently reinforced the importance of the maturity level of the individual DevSecOps program as a key indicator for both success of the project and the ability to insert good security practices into the DevSecOps pipeline. Programs that failed to achieve a level of maturity through education, discipline, and leadership commitment struggled with similar issues and in some cases were characterized as having “failed back to Waterfall.” Based on the scope of this study, the IDA team focused on how this lack of maturity affected secure coding and test practices, but the issues were cross cutting and particularly noted in delivery of software that did not meet the operational requirements. Although the interviews were concerned with integration of Security into the DevSecOps construct, lack of good integration of Operations into the CI/CD practice was also identified as a key impediment to meeting the development goal of rapidly providing operational capabilities to system users.

NIWC Atlantic[52] puts forward a nine-level maturity model for DoD DevSecOps efforts based on defining practices rather than metrics (see Figure 5-1). This model is useful not only for assessing the maturity of a particular DevSecOps implementation, but for assessing whether a legacy Waterfall-based development effort is a good candidate to convert to DevSecOps.

[52] Naval Information Center Atlantic Brief: DevSecOps Maturity and Business Value. The model is derived from these sources:

1. U.S. General Services Administration, *DevSecOps Guide*. See https://tech.gsa.gov/guides/#API+Agile+Design+DevSecOps+Development+Team.
2. Puppet Labs, “*Puppet State of DevOps Report 2018*, Puppet Labs, 2018. Available for download at https://puppet.com/resources/report/2018-state-devops-report/.
3. Reference [32].



Level	Defining Practices
1	**Regressive**: processes are unrepeatable, poorly controlled, and reactionary. Not viable for a DevSecOps platform. • The platform is characterized by manual efforts • It is not transparent about state • It is heterogeneously configured on a per-project basis
2	**Repeatable**: processes are documented and partly automated. Not true DevSecOps platform. • Application developers have a pipeline that they can use to deploy software. • The pipeline is considerate of security. • Intake into the platform may be manual or unpredictable.
3	**Consistent**: automated processes are applied across the whole application lifecycle. • Application developers have a clear, self-service intake onto the platform and • The ability to deploy and run security-compliant code in production through automation. • The platform services are centralized in its infrastructure and pipeline implementation.
4	**Reused**: • Monitoring and alerting are configurable by the team operating the service. • Deployment patterns for building applications or services are reused. • Testing patterns for building applications or services are reused. • Configurations are managed by a configuration management tool.
5	**Controlled**: • Application development teams use version control. • Teams deploy on a standard set of operating systems.
6	**Standardized**: • Build on a standard set of technology. • Teams deploy on a single standard operating system.
7	**Quantitatively-Managed**: process measured and controlled. • Individuals can do work without manual approval from outside the team.
8	**Automated Configuration:** • System configurations are automated. • Provisioning is automated.
9	**Optimizing**: focus on process improvement.

Figure 5-1. NIWC DevSecOps Maturity Model

Two other relevant DevSecOps maturity assessments are also available.

1. The DoD Enterprise DevSecOps Maturity Review. [31]
2. The Open Web Application Security Project (OWASP) DevSecOps Maturity Model (DSOMM)[53]

The DoD Enterprise DevSecOps Maturity Review, authored by the USAF CSO, is a questionnaire that leads an organization through a review of its DevSecOps "software factory." It appears the review is designed for onboarding organizations into one of the DoD Enterprise DevSecOps service offerings. The review doesn't allow for any scoring or

[53] OWASP DevSecOps Maturity Model DSOMM, https://owasp.org/www-project-devsecops-maturity-model/

self-assessment, with no criteria given for how responses align to greater or lesser maturity levels. However, better responses are strongly implied, many of them dependent on using the DoD enterprise services that the DoD DevSecOps COP champions.

The OWASP DSOMM has the advantage of being security focused. It is part of a project by the OWASP based on international standards to help systematically mature DevSecOps practices. The project is opensource with an accompanying git repository. There are four maturity levels with varying depth defined across five dimensions for characterizing an organization's DevSecOps practices.[54]

- Build and deployment
- Culture and organization
- Implementation
- Information gathering
- Test and verification

These dimensions correspond directly to the areas consistently identified by respondents to the IDA survey as integral for the success of programs employing DevSecOps, and specifically for inculcating security into DevSecOps based programs. Although only one of the respondents specifically called out use of the OWASP[55] model as an application security framework for development, many of the commonly used automated security tools adhere to the OWASP framework. Use of a model like the DSOMM builds on the security practices of the OWASP framework by providing a methodology to measure the maturity of the end to end development processes and supports leadership with evidence that the DevSecOps program can and does successfully deliver fully functional secure software.

Most responses concurred with the position of the USAF CSO [31] and the NIWC[56] on the need for adoption and application of a CMMC-like[57] DevSecOps maturity assessment, and with the need for ongoing effort to address where the current RMF practice falls short[58] in assessing and authorizing DevSecOps developed systems. Both of these initiatives underscore a need to adapt and update DT policy and practice to provide

---

[54] https://dsomm.timo-pagel.de/index.php

[55] https://owasp.org/www-project-application-security-verification-standard/

[56] Naval Information Center Atlantic Brief: DevSecOps Maturity and Business Value. See footnote 52.

[57] https://acqnotes.com/wp-content/uploads/2014/09/CMMC-Securing-the-DoD-Supply-Chain-Overview-Nov-2019.pdf

[58] https://media.dau.edu/media/Continuous+ATO/1_10jrntl6,and https://govtribe.com/file/government-file/fa877020r0518-air-force-continuous-ato-playbook-dot-pdf. Both the Air Force and the Navy have published their own CATO playbooks.

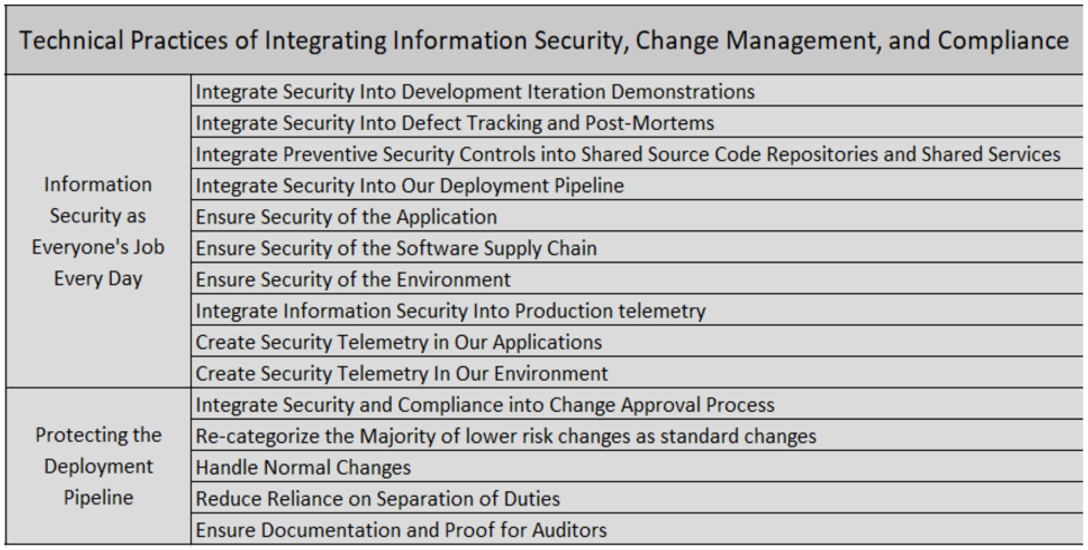
effective testing, evaluation, and assessment of software developed using a DevSecOps methodology and the DevSecOps environment itself. When the automated build process integrated in the pipeline is basically an extension of the software product, then testing must encompass the development environment, the automated processes instantiated as code, and the continuous development and delivery of the system software.

6. Conclusion and Recommendations

A. Conclusions

The highly technical, automated, and continuous cyclic nature of DevSecOps processes require a commensurate degree of knowledge, discipline, commitment, and oversight. If these elements are missing, the result is merely an ad hoc collection of practices that apply DevSecOps methodologies in name only. Interviews reinforced this observation with an acknowledgment that projects suffered where leadership and team members were working from assumptions of what DevSecOps and Agile meant, rather than knowledge achieved through actual training and experience.

Figure 6-1. Technical Security Practices[59] illustrates the information security, change management, and compliance practices required to support the goals of integrating information security continuously into every part of the development process, and also protect the development process itself. This duality of effort is particularly important in DevSecOps environments where use of EaC, Everything as Code, effectively makes the individual DevSecOps pipeline part of the final deployed software product.



Technical Practices of Integrating Information Security, Change Management, and Compliance	
Information Security as Everyone's Job Every Day	Integrate Security Into Development Iteration Demonstrations
	Integrate Security Into Defect Tracking and Post-Mortems
	Integrate Preventive Security Controls into Shared Source Code Repositories and Shared Services
	Integrate Security Into Our Deployment Pipeline
	Ensure Security of the Application
	Ensure Security of the Software Supply Chain
	Ensure Security of the Environment
	Integrate Information Security Into Production telemetry
	Create Security Telemetry in Our Applications
	Create Security Telemetry In Our Environment
Protecting the Deployment Pipeline	Integrate Security and Compliance into Change Approval Process
	Re-categorize the Majority of lower risk changes as standard changes
	Handle Normal Changes
	Reduce Reliance on Separation of Duties
	Ensure Documentation and Proof for Auditors

Figure 6-1. Technical Security Practices

Each of these individual practices is reflected in technical implementation in some part of the coding of either the development environment, the pipeline, or the product. Since most coders are not necessarily well versed in even the information security practices

[59] Naval Information Center Atlantic Brief: DevSecOps Maturity and Business Value. Derived from [4].

that apply to writing secure code, as reflected in the ongoing need for cyber DT, the necessity for development team members who are security practitioners is evident, in particular members who are knowledgeable testers. But this need for knowledge works both ways, the developmental security team members must be well versed in DevSecOps processes, and how to incorporate and automate the appropriate testing and monitoring into the pipeline and build processes to achieve an efficient consistent repeatable and secure result.

The repeated observations that maturity of an organization's implementation of DevSecOps methodology is the best indicator of how well security practices, and broadly any development practices, are being employed might seem self-evident. However, given the unique purpose-built nature of each DevSecOps ecosystem and the factories and pipelines within those ecosystems, metrics that provide consistent external measurement of the quality of processes across different ecosystems are difficult to achieve. Even in DoD where there are enterprise DevSecOps ecosystems available, much like in RMF, the amount and relevance of inheritance of good coding and security practices depends entirely on how much of what the enterprise ecosystem provides is actually used by the development team. The relationship can be compared to a Russian nesting doll, where each layer is dependent on the previous one that envelopes it. The development organization's direct responsibility begins with the layers it provides and controls. This reality is reflected in the recommendations for improving DevSecOps practice provided below.

B. Recommendations

The IDA team synthesized the following set of recommendations through a crosswalk between the DevSecOps success stories, problem areas, and lessons learned, captured in the survey responses, interviews, and the DevSecOps documentation available to us. IDA recommends DoD take the actions in Table 6-1 to promote widespread, consistent, and effective use of DevSecOps throughout defense-related projects, programs, and offices. Per the shading, there are three categories of recommendations: DoD-wide recommendations, per-program managerial recommendations, and per-program technical recommendations. Given the interdependencies of these efforts, the recommendations are not ranked or prioritized.

Table 6-1. Recommended Actions to Promote DevSecOps

Recommendations
Define and propagate an adequate standard definition of DevSecOps, and specifically of integrating the traditional role of DT in DevSecOps methodology
Create top down understanding, commitment, and application of DevSecOps concepts in the development organization and culture
Adopt a DevSecOps Maturity Model to measure and improve development program security effectiveness
Understand the cultural and resource challenges in converting Waterfall programs to DevSecOps, scaling up from small projects, and cross domain development
Identify mission focused security requirements and incorporate test processes and environments up front
Identify, continually evaluate, and manage development to objective security metrics to ensure the DevSecOps environment produces software that meets mission security requirements
Commit sufficient operational and security staff in addition to developers to ensure teams produce software that meets operational and security requirements
Ensure the development methodology is appropriate for the program requirements and can be adapted, especially for physically isolated or highly sensitive systems
Incorporate good architecture and design in the development environment and pipeline to support security
Work to identify and integrate testing that cannot be automated in a disciplined repeatable DevSecOps process
Fully leverage automation including virtualization, containerization, and IaC for testing, pipeline, and builds
Use DoD enterprise code repositories whenever possible to reduce rework and minimize supply chain threat

The IDA team also noted, in Problem Area #2, the lack of mature DoD guidance on how to implement Continuous ATO under DevSecOps. The team makes no specific recommendation on how to do so, believing the subject merits further study before anything concrete can be formulated. Accordingly, the team recommends a separate study on implementing Continuous ATO under DevSecOps. Another topic that, while seemingly apparent, is beyond the scope of this paper is the comparative value of using enterprise DevSecOps capabilities over “localized” implementations; the team recommends further study of this topic as well. Specific improvements to the DoD Cyber T&E Guidebook [34] based on these recommendations will need to be worked in conjunction with the DevSecOps T&E Guidebook.[60]

[60] In review at the time of this report.

Indeed, all of the DoD-wide recommendations seem sufficiently complex to merit future study. These studies would no doubt produce more focused granular recommendations of their own.

Appendix A.
Cybersecurity and Software Development Survey

Survey / Interview Questions:

Please focus on questions that apply to your role. No single responder is expected to know all the answers. Provide brief answers based on your immediate knowledge, not extended discussions or research. Mark questions N/A if you don't know the answer. Survey results will be non-attributable. We greatly appreciate your insights.

1. Development Processes

1. What development processes do you use (e.g., Agile, DevOps, DevSecOps, etc.)?
2. If you use DevSecOps:
    a. Who's on the team? (e.g., operations, developers, red team, blue team, security)
    b. What kinds of training (including certificates or certifications) do the team members have or is required of them?
    c. What are your organization's primary Development (DevSecOps) initiatives?

2. Roles

3. What is your role (developer, operations, security, testing, pipeline developer) in DevSecOps initiatives?
    a. In your role, describe your interaction with other team members; their roles and the nature and frequency of interaction (meetings, teleconferences, email, file shares, etc.).

3. General Observations

4. How have you integrated developmental testing (DT) into your DevSecOps initiatives?
5. Regarding DevSecOps and its integration with DT, **what is working well and why**? If possible, list things that have helped improve security in a DoD context.
    a. Has this been objectively demonstrated?
6. Regarding DevSecOps and its integration with DT, **what is not working well and why**? If possible, list things that are difficult or impossible in a DoD context.
    a. Has this been objectively demonstrated?
7. Regarding DevSecOps and its integration with DT, **do you have any lessons learned that are worth sharing with the community**? Do you measure software quality? How?

4. Development Practices

8. What environments do you have (dev, test, production, etc.)?
9. In each environment, what is instrumented (i.e., set up to log events) and how?

10. How do you arrive at the original specifications for the MVP and succeeding backlogs/sprints?
11. Who develops the specifications or requirements—including security-related ones—and how are they used?
    a. Where are they stored and in what format?
    b. When in the life cycle are the security requirements defined?
    c. How are they communicated to developers? To testers? To operators?
12. Are security requirements traceable to their implementation? How?
13. Which software development and/or developmental test specific DoD or NIST policies/guidance do you adhere to, and are these firm requirements or best practices for you?
14. Is there a monolithic source code repository, or are there multiple ones?
    a. Where are the repositories and who has access?
    b. What types of access exist?
    c. If there are multiple repositories, how are they interrelated?
    d. What security validation and monitoring are done against the repositories?
15. What is the branch strategy? How is it enforced?
16. What happens before a change is checked in?
17. What happens after a change is checked in (include any tools)?
    a. Does it trigger an automated build and test sequence?
    b. Is this check-in sequence fully logged and visible to the entire team?
    c. Does a failure at any state trigger a notification?
    d. Does a warning at any state trigger a notification and is a warning considered a failure?
18. How do changes transition from check-in to build to CI and unit testing, to functional and integration testing, and to staging? (Include any tools)
19. And then finally to production? (Include any tools)
    a. What is the deployment strategy (e.g., ring-based)?
    b. How is the software monitored in production?

5. Testing

20. What tests are run, and how do the tests and tools change with the system under test? (Include any tools)
    a. When in the life cycle are the test cases (including security test cases) defined and developed?
    b. For each testing category, who is accountable for defining and developing the tests (including security test cases)?
    c. Who is accountable for running the tests and who is accountable for addressing test results?
    d. Where are the test results logged? (Include any tools) and in what format?
    e. What is the source of the security tests?
        i. OWASP Top 10 or OWASP Proactive Controls?
        ii. Regression tests?
            1. Are all unit and integration tests always run or just selected ones?
            2. Are they automated as part of the pipeline?
        iii. Unit test sources?

                iv. Integration test sources?
                v. Static testing?
                    1. Automated tools and/or Code reviews?
                vi. Dynamic testing?
                    1. Automated tools, Fuzzing, Penetration testing, Fire drills?

21. What other tools are used, and for what purposes? (E.g., orchestration of containers)
22. What manual checks or reviews are performed and when?
23. How is this used for security and compliance purposes?
24. What is your logging strategy?
    a. Does your security testing include tracing through logs?
        i. What automated methods are used (including tools)?
        ii. What manual methods are used (including tools)?

(This page is intentionally blank.)

Appendix B.
Policy and Guidance Referenced by Respondents

The following is a list of policy and guidance documents respondents cited as guiding their organizational DevSecOps practices. Respondents consistently noted that they were forced to infer or seek additional definition for standards and guidance as the references were incomplete, not specific, or had not been updated to remain relevant with DevSecOps and other CI/CD development methodologies.

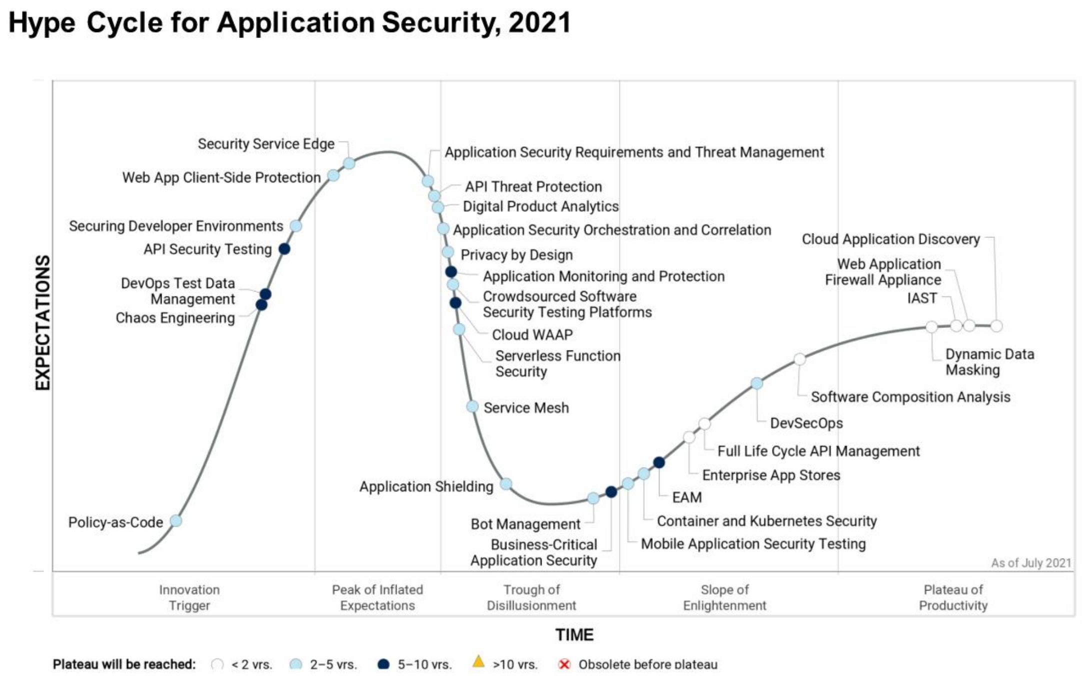
The IDA team tried to provide references to these documents, although sometimes a respondent did not supply IDA enough information to identify a document. Some respondents referred to a collection of documents. The list below notes the documents that could be identified.

- NIST SP 800-53 Rev. 5, Security and Privacy Controls for Information Systems and Organizations, December 10, 2020 [D-1].
- NIST SP 800-115, Technical Guide to Information Security Testing and Assessment, September 2008 [6].
- DoDI 5000.87, Operation of the Software Acquisition Pathway, October 2, 2020 [7].
- DoDI 5000.89, Test and Evaluation, November 19, 2020 [8].
- DoD Directive 8140.01, Cyberspace Workforce Management October 5, 2020 [9].
- DoDI 8510.01, Risk Management Framework (RMF) for DOD Information Technology (IT), May 24, 2016 [10].
- Director, Operational Test and Evaluation (DOT&E) Directive Memos. Not further specified in the response.
- Directive-Type Memorandum (DTM) 17-001: Cybersecurity in the Defense Acquisition System, January 11, 2017 [11].
- NAVAIRINST 3960.4B, Project Test Plan Policy and Guide for Testing Air Vehicles, Air Vehicle Weapons, And Air Vehicle Installed Systems, June 7, 2005 [12]. (This directive has been superseded and canceled by NAVAIRINST 3960.4C.)

- Department of the Air Force (DAF) Manual 63-119 Mission-Oriented Test Readiness Certification, April 15, 2021[13].
- Air Force Instruction (AFI) 99-103 Capabilities-Based Test and Evaluation, November 18, 2019[14].
- Air Force Operational Test and Evaluation Center (AFOTEC) Instruction 99-101 Conduct of Operational Test and Evaluation, May 1, 2007 [15].
- AO 177 IAS. Not further specified, and search results were inconclusive.
- System-Theoretic Process Analysis for Security (STPA-SEC).[61]
- Scaled Agile Framework (SAFe), version 5.0. See https://www.scaledagileframework.com/.
- Center for Internet Security guidance. See https://www.cisecurity.org/.
- Cloud Security Alliance guidance. See https://cloudsecurityalliance.org/.

[61] See https://psas.scripts.mit.edu/home/wp-content/uploads/2017/04/STAMP_2017_STPA_SEC_TUTORIAL_as-presented.pdf for a presentation discussing the concept.

Appendix C. Application Security Hype Cycle, 2021



Source: Gartner (July 2021)
747408

(This page is intentionally blank.)

Appendix D. References

[1] Schwartz, M. *A seat at the table: IT leadership in the age of agility.* IT Revolution Press, Portland, OR (2017). ISBN 9781942788119.

[2] Kim, G., K. Behr, and G. Spafford. *The phoenix project: A novel about IT, DevOps, and helping your business win.* IT Revolution Press, Portland, OR (2018). Third edition. ISBN 9781942788294.

[3] Kim, G. *The unicorn project: A novel about digital disruption, developers, and overthrowing the ancient powerful order.* IT Revolution Press, Portland, OR (2019). ISBN 9781942788799.

[4] Kim, G., P. Debois, J. Willis, J. Humble, and J. Allspaw. *The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations.* IT Revolution Press, Portland, OR (2016). ISBN 9781942788003.

[5] Agre, J. *DevSecOps: State of the Art and Relevance to the Department of Defense*. NS D-13119, Institute for Defense Analyses, Alexandria, VA (June 2020).

[6] *Security and Privacy Controls for Information Systems and Organizations*, NIST Special Publication 800-53, Revision 5. National Institute of Standards and Technology, Gaithersburg, MD (December 10, 2020). Available online at https://doi.org/10.6028/NIST.SP.800-53r5.

[6] Scarfone, K., M. Souppaya, A. Code, and A. Orebaugh. *Technical Guide to Information Security Testing and Assessment*, NIST Special Publication 800-115. National Institute of Standards and Technology, Gaithersburg, MD (September 2008). Available online at https://doi.org/10.6028/NIST.SP.800-115.

[7] DoD Instruction 5000.87, *Operation of the Software Acquisition Pathway*. October 2, 2020. Available online at https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500087p.PDF.

[8] DoD Instruction 500.89, *Test and Evaluation*. November 19, 2020. Available online at https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500089p.PDF.

[9] DoD Directive (DoDD) 8140.01, *Cyberspace Workforce Management*. October 5, 2020. Available online at https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodd/814001p.pdf.

[10] DoD Instruction 8510.01, *Risk Management Framework (RMF) for DOD Information Technology (IT)*, May 24, 2016. Available online at https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/851001p.pdf.

[11] Directive-Type Memorandum (DTM) 17-001: *Cybersecurity in the Defense Acquisition System*. January 11, 2017. Available online at https://www.hsdl.org/?view&did=797977.

[12] NAVAIRINST 3960.4B, *Project Test Plan Policy and Guide for Testing Air Vehicles, Air Vehicle Weapons, And Air Vehicle Installed Systems*. June 7, 2005. Available online at http://everyspec.com/USN/NAVAIR/NAVAIRINST_3960x4B_18247/.

[13] Department of the Air Force Manual 63-119, *Mission-Oriented Test Readiness Certification*. April 15, 2021. Available online at https://static.e-publishing.af.mil/production/1/saf_aq/publication/dafman63-119/dafman63-119.pdf.

[14] Air Force Instruction 99-103, *Capabilities-Based Test and Evaluation*. November 18, 2019. Available online at https://standards.globalspec.com/std/14358164/AFI%2099-103.

[15] Air Force Operational Test and Evaluation Center Instruction 99-101, *Conduct of Operational Test and Evaluation*. May 1, 2007. Available online at https://myclass.dau.edu/bbcswebdav/institution/Courses/Deployed/TST/TST204%20and%20TST204V/Archives/March%2017%20Student%20Files/Student%20CD/1%20References/04%20US%20Air%20Force%20Guidance/AFOTECI%2099-101.pdf.

[16] Bennington, H. "Production of Large Computer Programs." *IEEE Annals of the History of Computing* **5**:4, October 1983.

[17] Royce, W. "Managing the Development of Large Software Systems." Proc. IEEE WESCON **26**, August 1970.

[18] Gladden, G. "Stop the Life-Cycle, I Want to Get Off." *Software Engineering Notes* **7**:2, April 1982.

[19] Boehm, B. "A spiral model of software development and enhancement." *IEEE Computer* **21**:5, May 1988.

[20] Parnas, D. and P. Clements. "A Rational Design Process: How and Why to Fake it." *IEEE Transactions on Software Engineering* **SE-12**:2, January 1986.

[21] OUSD(R&E)/DOT&E, DEPARTMENT OF DEFENSE *DevSecOps Test & Evaluation Guidebook Version 1.0 Draft*, 23 October 2020.

[22] MacDonald, N. and D. Gardner. *12 Things to Get Right for Successful DevSecOps.* G00450792, Gartner, December 19, 2019. https://www.gartner.com/en/documents/3978490/12-things-to-get-right-for-successful-devsecops.

[23] Gardner, D., M. Horvath, and D. Zumerle. *Magic Quadrant for Application Security Testing.* G00733839, Gartner, May 27, 2021. https://www.gartner.com/en/documents/4001946-magic-quadrant-for-application-security-testing.

[24] Fritsch, J. *Hype Cycle for Application Security, 2021*. G00747408, Gartner, July 12, 2021. https://www.gartner.com/en/documents/4003469-hype-cycle-for-application-security-2021.

[25] Odell, L., R. Wagner, and T. Weir. *Department of Defense Use of Commercial Cloud Computing Capabilities and Services*, IDA P-5287, Institute for Defense Analyses, Alexandria, Virginia, November 2015.

[26] Ambroso, M. and G. Kennedy. *Ensuring Availability of Cybersecurity Developmental Test and Evaluation Data for Cloud Systems*, IDA P-10371, Institute for Defense Analyses, Alexandria, Virginia, August 2019.

[27] Fairbanks, G. "Architectural Hoisting," *IEEE Software* **31** (4), pp. 12–15, 2014, doi: 10.1109/MS.2014.82.

[28] Dodson, D., M. Souppaaya, and K. Scarfone. *Mitigating the Risk of Software Vulnerabilities by Adopting a Secure Software Development Framework (SSDF)*. National Institute of Standards and Technology, Gaithersburg, Maryland, April 23, 2020. Available online at https://csrc.nist.gov/publications/detail/white-paper/2020/04/23/mitigating-risk-of-software-vulnerabilities-with-ssdf/final.

[29] Fairbanks, G. *Just Enough Software Architecture: A Risk-driven Approach*. Marshall & Brainerd, Boulder, Colorado, ISBN 978-0984618101, August 2010.

[30] *Department of Defense Cloud Computing Security Requirements Guide*, Version 1, Release 3, March 6, 2017. Available online at https://rmf.org/wp-content/uploads/2018/05/Cloud_Computing_SRG_v1r3.pdf.

[31] Chaillan, N. *DoD Enterprise DevSecOps Maturity Review Version 1.6*, December 2019. Available online at https://software.af.mil/wp-content/uploads/2019/12/DoD-Enterprise-DevSecOps-Maturity-Review-v1.6.docx.

[32] Humble, J. and R. Russell. *The Agile Maturity Model Applied to Building and Releasing Software*, ThoughtWorks® Studios, September 2009. Available online at https://info.thoughtworks.com/rs/thoughtworks2/images/agile_maturity_model.pdf.

[33] Chaillan, N. *DoD Enterprise DevSecOps Reference Design Version 1.0*, Department of Defense, August 12, 2019. Available online at https://dodcio.defense.gov/Portals/0/Documents/DoD%20Enterprise%20DevSecOps%20Reference%20Design%20v1.0_Public%20Release.pdf.

[34] *Cybersecurity Test and Evaluation Guidebook, Version 2.0, Change 1*. Department of Defense, February 10, 2020. Available online at https://www.dau.edu/cop/test/DAU%20Sponsored%20Documents/Cybersecurity-Test-and-Evaluation-Guidebook-Version2-change-1.pdf.

(This page is intentionally blank.)

Appendix E. Acronyms and Abbreviations

AFI	Air Force Instruction
AFOTEC	Air Force Operational Test and Evaluation Center
AST	Application Security Testing
ATO	Authority to Operate
CATO	Continuous Authority to Operate
CD	Continuous Deployment (alternately, Continuous Delivery)
CI	Continuous Integration
CI/CD	Continuous Integration / Continuous Deployment
CIO	Chief Information Officer
CMMC	CMMI certification program
CMMI	Capability Maturity Model Integration
COP	Community of Practice
CSO	Chief Software Officer
CVE	Common Vulnerabilities and Exposures
DAF	Department of the Air Force
DAST	Dynamic Application Security Testing
DFARS	Defense Acquisition Regulation Supplement
DISA	Defense Information Systems Agency
DoD	Department of Defense
DODD	Department of Defense Directive
DODI	Department of Defense Instruction
DOT&E	Director, Operational Test and Evaluation
DSOMM	DevSecOps Maturity Model
DT	Developmental Testing
DT&E	Developmental Testing and Evaluation
DTM	Directive-Type Memorandum
EaC	Everything as Code
FedRAMP	Federal Risk and Authorization Management Program
IaC	Infrastructure as Code
IDA	Institute for Defense Analyses
IL	Impact Level
MCBOSS	Marine Corps Business Operations Support Services
NIWC	Naval Information Warfare Center
NIST	National Institute of Standards and Technology
OUSD	Office of the Undersecretary of Defense
OWASP	Open Web Application Security Project
POA&M	Plan of Actions and Milestones
RMF	Risk Management Framework
SAFe	Scaled Agile Framework

SAST	Static Application Security Testing
SRG	(Cloud) Security Reference Guide
STPA-SEC	System-Theoretic Process Analysis for Security
T&E	Test and Evaluation
USAF	United States Air Force
USD	Undersecretary of Defense
USD (R&E)	Undersecretary of Defense (Research and Engineering)
VM	Virtual Machine

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 09-2021	**2. REPORT TYPE** IDA Publication	**3. DATES COVERED** *(From - To)*
4. TITLE ANDSUBTITLE Cybersecurity and DoD System Development: A Survey of DoD Adoption of Best DevSecOps Practice		**5a. CONTRACT NUMBER** HQ0034-19-D-0001 **5b. GRANT NUMBER** _____ _____ _____ **5c. PROGRAM ELEMENT NUMBER** _____ _____ _____
6. AUTHOR(S) George L. Kennedy (ITSD); Steven P. Wartik (ITSD); Ryan R. Wagner (ITSD)		**5d. PROJECT NUMBER** AX-1-3100 **5e. TASK NUMBER** _____ _____ _____ **5f. WORK UNIT NUMBER** _____ _____ _____
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Institute for Defense Analyses 4850 Mark Center Drive Alexandria, Virginia 22311-1882		**8. PERFORMING ORGANIZATION REPORT NUMBER** 22749 H 2021-000267
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Ms. Sarah Standard Cybersecurity/Interoperability Technical Director Director, Developmental Test, Evaluation, and Assessments (D,DTE&A), OUSD R&E		**10. SPONSOR/MONITOR'S ACRONYM(S)** **11. SPONSOR/MONITOR'S REPORT NUMBER** _____ _____ _____

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES
_____ _____ _____

14. ABSTRACT
DoD is moving from the Waterfall Model of software development to modern methods such as Agile, DevOps, and especially DevSecOps, which emphasizes considering cybersecurity early. In 2020, OUSD/R&E tasked the Institute for Defense Analyses to study DoD organizations practicing DevSecOps and other non-Waterfall methodologies, to capture their successes and failures, to report actions organizations should take to adopt DevSecOps, and recommend DoD-wide actions to promote DevSecOps practice. IDA developed and distributed a survey, received 18 responses, and conducted follow-up telephone interviews. IDA heard many success stories, including increased up-front planning and incorporation of testing processes, and implementation of pipelines that lowered the time from coding to deployment; furthermore, several respondents reported their metrics objectively demonstrated improvement. At the same time, some respondents felt DoD's current acquisition model and ATO processes are not truly compatible with DevSecOps; that forming teams is difficult; and that the role of developmental testing is unclear within DevSecOps. Part of the problem is that DevSecOps is still new and lacks standard concepts and terminology. IDA recommends DoD take eleven actions to promote adoption of DevSecOps. These actions will clarify and help acculturate DevSecOps concepts throughout DoD. The actions will also simplify creating and using pipelines, lessening the up-front costs of a DevSecOps-based project.

15. SUBJECT TERMS
Agile, CI/CD, Cybersecurity, Developmental testing, DevOps, DevSecOps, Pipeline, Soft Acquisition

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** John Hong (SED)
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	Unlimited		**19b. TELEPHONE NUMBER** *(include area code)* (703) 845-2564

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI std. Z39.18