Cloud Security Playbook Volume 2

February 11, 2025

Version 1.0

Approved By

MARTIN.CHARLES.L.1171456467 Digitally signed by MARTIN.CHARLES.L.1171456467 Date: 2025.02.21 15:03:23 -05'00'	February 21, 2025
Charles L. Martin	Date
Cybersecurity Lead Cloud and Software Modernization Directorate Department of Defense Office of the Chief Information Officer-Information Enterprise	
LAMB.GEORGE.WILLIE.IV.1506724321 Digitally signed by LAMB.GEORGE.WILLIE.IV.1506724321 Date: 2025.02.21 14:59:15 -05'00'	February 21, 2025
George Lamb	Date
Director Cloud and Software Modernization Directorate DoD CIO, DCIO-Information Enterprise	

Trademark Information

Table of Contents

List of Figures

List of Tables

Introduction

This is the second volume of the Cloud Security Playbook. The first volume provides some rationale for why cloud security is so important. It includes 18 plays and several appendices.

Volume 2 discusses some more sophisticated plays that do not apply to all systems hosted in a cloud. For example, this volume discusses containers, microservices, defending DevSecOps pipelines, and securing Artificial Intelligence (AI) systems.

Audience

The Cloud Security Playbook is intended for Mission Owners (MOs), Software Development Managers, developers, and organizations that are developing software (or who have acquired software) to host in a cloud, including those using cloud native services.

Purpose

This document was created to make it easy to improve the cybersecurity of applications hosted in a cloud.

Like Volume 1, this volume also points to useful documents so that MOs know where to find more details.

There are many threats and vulnerabilities related to cloud security. Each play in this playbook contains an actions section that describes several actions that mission owners should take to mitigate these cloud vulnerabilities to reduce cybersecurity risk to their systems and missions.

Play Reading Order

This is the second volume of the Cloud Security Playbook. Readers should start with the first volume before moving to this one.

Plays may be read in any order, but some plays use concepts described in earlier plays. It is not necessary to implement the plays in order. Indeed, the implementation of many plays may be accomplished in parallel. However, some plays rely on earlier plays having been accomplished.

Play 19. Secure Containers and Microservices

This play discusses two concepts that work well together: microservices and containers.

Microservices

Microservices, and using a microservice architecture, is an approach to application development in which a large application is built as a collection of modular, loosely coupled components or services. A microservices architecture uses fine-grained services, and lightweight protocols. Each microservice typically runs inside a software container.

The design of a microservice is based on the following drivers from NIST SP 800-204, *Security Strategies for Microservices-based Application Systems* [1].

- Each microservice must be managed, replicated, scaled, upgraded, and deployed independently of other microservices.
- Each microservice must have a single function and operate in a bounded context (i.e., have limited responsibility and dependence on other services).
- All microservices should be designed for constant failure and recovery and must therefore be as stateless as possible.
- Reuse existing hardened services (e.g., databases, caches, directories) for state management.
- Some benefits of decomposing an application into different smaller microservices include:
    - Improved modularity.
    - Makes the application easier to understand, develop, and test, since each component is smaller.
    - Makes the application more resilient to architecture erosion.
    - Parallelizes development by enabling small autonomous teams to develop, deploy and scale their respective services independently.

One benefit of a microservices approach is that each microservice can evolve independently from other microservices. Other benefits include faster scaling on demand, upgrades that do not impact users, more precise cyber hardening at a per-service level, graceful degradation, and improved support to quickly recover from failure.

For more information on securing microservices, see NIST SP 800-204A, *Building Secure Microservices-based Applications Using Service-Mesh Architecture* [2], and NIST SP 800-204, *Security Strategies for Microservices-based Application Systems* [1].

Containers

A modular open system approach (MOSA) is an acquisition and design strategy consisting of a technical architecture that adopts open standards and supports a modular, loosely coupled and highly cohesive system structure. U.S. Code Title 10 Section 2446a, and DoD Instruction 5000.02 require MOSA. A modern software architecture predicated upon microservices, and software containers meets MOSA requirements.

A container is a light-weight, standalone, executable package of software that includes everything needed to run a microservice or mission service except the Operating System (OS); it includes compiled code, runtime (e.g., the Jave Runtime Environment), system tools, system libraries and configuration settings. Containers run in isolated processes from one another, so several containers can run in the same host OS without conflicting with one another.

Containers are lighter weight than Virtual Machine (VM) Images, and they start much more rapidly than a VM, since the operating system is already running. Thus, they can scale up and down faster than a VM-based architecture.

Containers can readily move from one cloud or local environment to another. Containers that do not use any cloud services should move easily between CSPs. However, if the software inside the container uses any cloud services, that container will not move easily between CSPs, since the code in the container cannot run in an environment that lacks the cloud services it requires. Such containers must be refactored for the new CSP.

All containers must be Open Container Initiative (OCI) compliant.

A **container image** is a reusable, shareable file used to create containers. A container is a runtime instance of a container image.

Container images should be immutable. That is, they should be built, then stored in an artifact repository and not modified. When they are deployed in an environment (development, test, staging, production, etc.), the container image is deployed as it was built, without modification, though it may accept parameters (e.g., IP addresses) on installation. The container image should not be modified. If there is an issue, a change request is submitted, the container is re-built and re-tested, and if it passes appropriate checks or control gates, this new immutable container can then be deployed.

Note that immutable containers may be stateless or stateful. Stateless containers allow automatic horizontal scaling in a cloud, so they are preferred when possible, and are often used in the middle-tier of a web application. However, some software is inherently stateful (e.g., a database), so some immutable containers will be stateful.

Poisoned Containers

A container image can be poisoned. This can happen when an attacker embeds malicious software inside a container image, for example, for a publicly available open-source container. The Poisoned Containers part of the Actions in this play offer suggestions to help mitigate this risk. In addition, a secure software supply chain helps; for more on that, see Play 20 and Play 21.

Hardening Containers

Containers should be hardened. Generally, a container is built and tested for cybersecurity vulnerabilities. If some are found, they are either fixed or mitigated with the mitigation described in a Plan of Action and Milestones (POA&M). Once it meets the risk threshold of acceptable risk, it is considered hardened. Using hardened immutable containers improves cybersecurity.

DISA's *Container Hardening Process Guide*, 2022 [3] includes a set of cybersecurity requirements for DoD hardened containers that include the following.

- ☐ Comply with initial and ongoing DOD Cybersecurity accreditation regulations/ frameworks.
  - If a Security Technical Implementation Guide (STIG) is available, the container base OS image must be STIGed.
  - NIST 800-53v5 moderate controls plus FedRAMP+ IL 6 controls.
  - Risk Management Framework (RMF) process and required documentation.
  - Containers must be compliant with DISA STIGs that exist for container technology and consistent with NIST SP 800-190 [4].
- ☐ Generate and automate necessary documentation for Risk Management.
  - RMF Controls
  - Data Flows
- ☐ Enable TLS on all PaaS tools that have a user interface or send data. Redirect HTTP and use Federal Information Processing Standard (FIPS) 140.2 encryption.
- ☐ Whenever possible, prohibit processes and containers from running as root.

Kubernetes

Proper deployment and management of containers also requires a container orchestrator. A **container orchestrator** performs tasks such as checking for new versions of containers, deploying the containers into the appropriate environment (development, test, or production), self-healing, rolling updates, security checks, and performing post-deployment validation tests.

Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications. When using K8s, the DoD requires Cloud Native Computing Foundation (CNCF) certified Kubernetes, as specified in the *DoD Enterprise DevSecOps Reference Design: CNCF Kubernetes* [5]; for brevity, this playbook refers to CNCF certified Kubernetes as CNCF K8s.

The key benefits of adopting Kubernetes include:

- Multimodal Environment: code runs equally well in a multitude of compute environments, benefitting from the K8s API abstraction.
- Resiliency: self-healing of unstable or crashed containers.
- Adaptability: containerized microservices create composable ecosystems.
- Automation: fundamental support for a GitOps model and IaC speeds process and feedback loops.
- Scalability: application elasticity to appropriately scale and match service demand.

Sidecar Security Container (SSC)

K8s packages containers into **pods**. Each pod may contain several containers, and containers within pods can share disk and network resources. A **sidecar container** is a container used to extend or enhance the functionality of an application container without strong coupling between two.

The use of pods makes it possible to create a **Sidecar Security Container (SSC)** and automatically deploy an instance of it in each pod alongside each application container, as depicted in the figure below. This sidecar security container helps to build security into the application without requiring any action from the application development team.

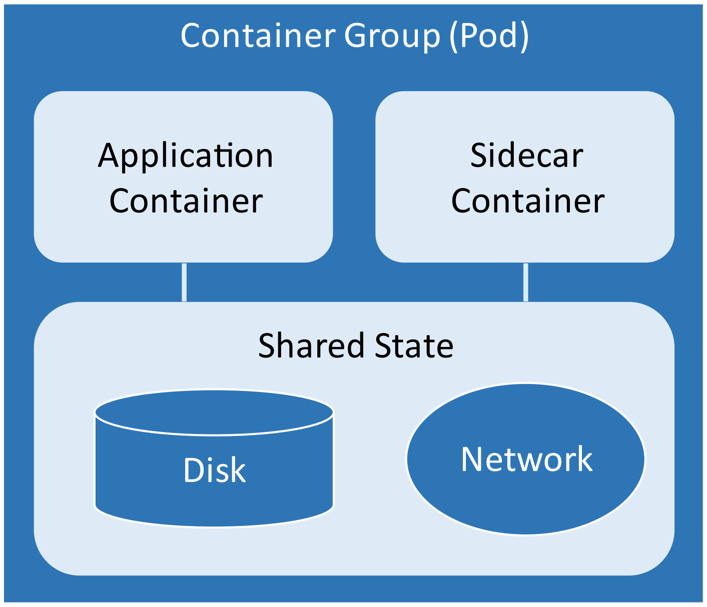

Figure 1. Pod with Sidecar Container

There are two key benefits to the SSC approach: first, application container developers do not need to modify it, and second, decoupling it from the main container makes it easy to rapidly deploy updates to the security sidecar without any need to rebuild the main container. So, the SSC can evolve independently of the application container. The security sidecar can include several services, including the following.

- Host Intrusion Detection System (HIDS) agent for signature-based continuous scanning using Common Vulnerabilities and Exposures (CVEs)
- HIDS agent for Runtime behavior analysis
- Centralized logging and telemetry that includes Extract, Transform, and Load (ETL) capabilities to normalize log data
- Robust east/west network traffic management (whitelisting)
- Role-Based Access Control
- Container policy enforcement
- Automated STIG compliance that complies with the Security Content Automation Protocol (SCAP).
- Service mesh proxy to tie into the service mesh described in the next subsection.

DoD sidecar containers are validated operationally in a threat-representative operational environment to verify that they provide the security services as designed.

For more on the SSC approach, see the *DoD Enterprise DevSecOps Reference Design: CNCF Kubernetes* [5]. Some components have developed an SSC, for example, the Air Force's Platform One has developed an SSC.

Service Mesh

A **service mesh** is "a dedicated infrastructure layer with a set of deployed infrastructure functions that facilitate service-to-service communication through service discovery, routing and internal load balancing, traffic configuration, encryption, authentication and authorization, metrics, and monitoring. It provides the capability to declaratively define network behavior, microservice instance identity, and traffic flow through policy in an environment of changing network topology due to service instances coming and going offline and continuously being relocated." – NIST SP 800-204A, *Building Secure Microservices-based Applications Using Service-Mesh Architecture* [2].

A service mesh can create a network of deployed services including load balancing, service discovery, service-to-service authentication and authorization, encryption, monitoring, and support for the circuit breaker pattern. The circuit breaker pattern identifies instances having trouble (e.g., slow to respond to requests), isolates them by stopping further requests from going to them, monitors them, and only routes new requests to them if the instance recovers.

If an application uses a microservice architecture, it should use a service mesh.

For more information, see NIST SP 800-204A, *Building Secure Microservices-based Applications Using Service-Mesh Architecture* [2], and NIST SP 800-204, *Security Strategies for Microservices-based Application Systems* [1].

Part of a service mesh implementation includes providing a sidecar container proxy for each service instance. The mesh routes service requests between the application's microservices through these proxies. These sidecar container service mesh proxies handle East-West (inter-service) traffic, logging and some security controls. Figure 2 depicts an example of a service mesh for an application with several microservices. A service mesh proxy is part of the SSC discussed in the previous section.

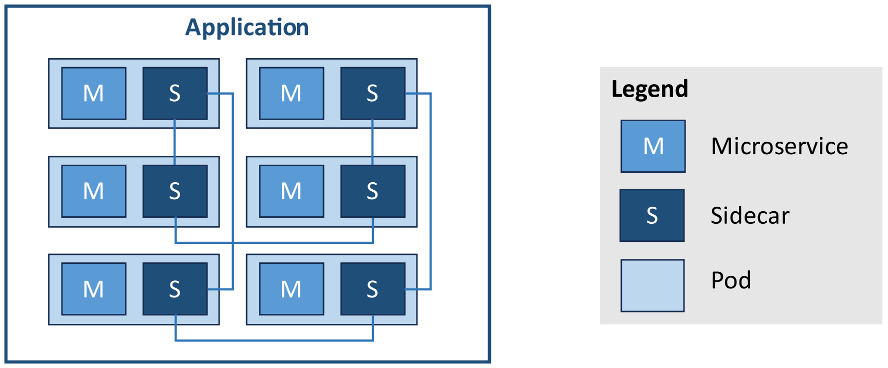

Figure 2. Service Mesh

As mentioned above, one aspect of a service mesh is to abstract inter-service communication (East-West traffic) from an application's microservices. This allows developers to focus on adding new mission features, while the operations team operates the application and manages the service mesh.

Microservice based applications that lack a service mesh must add code to each microservice to handle inter-service communication, thus causing developers to spend less time on mission goals. In addition, it is difficult to debug communications issues, since developers must look at the communications code in each affected service. With a service mesh, that logic is in one layer, making it easier to track down issues. A service mesh includes distributed logging and distributed tracing. It also captures inter-service communication performance metrics, which teams can use to improve application performance.

CNCF Kubernetes offers transport layer (Open Systems Interconnection (OSI) layer 4) load balancing. But a service mesh includes application layer (OSI layer 7) load balancing, which offers more advanced capabilities for load balancing, zero trust, access control, and routing.

Ambient Mesh

A recent alternative to using sidecars is an ambient mesh, or an ambient mode for a service mesh. This approach does not require a sidecar container. Compared to a sidecar approach, an ambient mesh moves the service mesh proxy from the sidecar to the Kubernetes node for mutual Transport Layer Security (mTLS) and identity. Since each node can run several pods, this approach "reduces the number of proxies to manage, slashing service mesh costs by reducing the compute and memory requirements per node."[1]

Consider an ambient mesh as an alternative to using a sidecar security container.

Actions

- ☐ Research and understand the benefits of a microservices architecture.
- ☐ Only adopt CNCF Certified Kubernetes to ensure software conformance of required APIs.
- ☐ Leverage a secure repository for hardened containers and other software artifacts.
- ☐ Use or create a sidecar security container or use an ambient mesh.

[1] Source: Ambient Mode - Simplify Operations of the Istio Service Mesh (solo.io).

- ☐ If using a sidecar, always inject the Sidecar Container Security Stack to maximize runtime security.
- ☐ If using microservices, adopt a service mesh or ambient mesh to further secure east-west network traffic.
- ☐ If using microservices, be aware of recommendations from these sources:
  - NIST SP 800-204A, *Building Secure Microservices-based Applications Using Service-Mesh Architecture* [2], and
  - NIST SP 800-204, *Security Strategies for Microservices-based Application Systems* [1].
- ☐ Package software in the form of containers.
- ☐ All containers must be OCI compliant.
- ☐ Scan containers for cybersecurity issues.
- ☐ Harden containers to improve cybersecurity.
- ☐ Use immutable containers.
- ☐ Create an artifact repository for hardened containers and their assessments.
- ☐ Implement the use of CNCF Kubernetes to orchestrate and manage containers.
- ☐ Use Kubernetes to deploy the sidecar security container with each container it deploys.
- ☐ Remember that while the container package moves easily between environments, containers that use cloud services will not run in other environments that lack those services.

Actions from NIST 800-190

NIST Special Publication 800-190, Application Container Security Guide [4] offers guidance on securing containers. Some of that guidance is summarized here, but more can be found in the original source.

- ☐ Use container-specific host OSs instead of general-purpose ones to reduce the attack surface.
- ☐ Only group containers with the same purpose, sensitivity, and threat posture on a single host OS kernel to allow for additional defense in depth.
- ☐ Adopt container-specific vulnerability management tools and processes for container images to prevent compromises.
- ☐ Consider using hardware-based countermeasures to provide a basis for trusted computing.

- ☐ Use container-aware runtime defense tools. Deploy and use a dedicated container security solution capable of preventing, detecting, and responding to threats aimed at containers during runtime.
- ☐ Avoid embedded clear text secrets. Secrets should be stored outside container images and provided dynamically at runtime as needed. Container orchestrators, such as Kubernetes, include native secrets management.
- ☐ Use only trusted container images. To mitigate these risks, organizations should take a multilayered approach that includes:
    - The capability to centrally control what container images and container registries are trusted in their environment.
    - Discrete identification of each image by cryptographic signature, using a NIST-validated implementation.
    - Enforcement to ensure that all hosts in the environment only run images from these approved lists.
    - Validation of image signatures before image execution to ensure images are from trusted sources and have not been tampered with.
    - Ongoing monitoring and maintenance of these repositories to ensure images within them are maintained and updated as vulnerabilities and configuration requirements change.
- ☐ Organizations should control the egress network traffic sent by containers. Specifically, app-aware tools should provide the following capabilities:
    - Automated determination of proper container networking surfaces, including both inbound ports and process-port bindings;
    - Detection of traffic flows both between containers and other network entities, over both 'on the wire' traffic and encapsulated traffic; and
    - Detection of network anomalies, such as unexpected traffic flows within the organization's network, port scanning, or outbound access to potentially dangerous destinations.

Avoiding Poisoned Containers

Mitigations for poisoned containers from [4] include:

- ☐ Ensure that only vetted, tested, validated, and digitally signed images are allowed to be uploaded to an organization's registries.
- ☐ Ensure that only trusted images are allowed to run, which will prevent images from external, unvetted sources from being used.
- ☐ Automatically scan images for vulnerabilities and malware, which may detect malicious code such as rootkits embedded within an image.

- ☐ Implement runtime controls that limit the container's ability to abuse resources, escalate privileges, and run executables.
- ☐ Use container-level network segmentation to limit the “blast radius” of what the poisoned image might do.
- ☐ Validate that container runtimes follow least-privilege and least-access principles.
- ☐ Build a threat profile of the container's runtime. This includes, but is not limited to, processes, network calls, and filesystem changes.
- ☐ Validate the integrity of images before runtime by leveraging hashes and digital signatures.
- ☐ Restrict images from being run based on rules establishing acceptable vulnerability severity levels.

Play 20. Defend DevSecOps Pipelines

Attacks on the software supply chain have become more common. Also, many organizations that either develop or contract development of software have moved to using a DevSecOps approach. Such an approach is also popular at most top-tier software companies. DevSecOps is an evolution of DevOps, combining development, security and operations. Typically, software produced with DevSecOps processes is developed using a Continuous Integration (CI) / Continuous Delivery (CD) pipeline, sometimes called a DevSecOps pipeline to emphasize the security aspect. This pipeline is a series of tools and processes that are automated and orchestrated to produce software, as it passes through various phases in the DevSecOps lifecycle, as depicted in Figure 3.

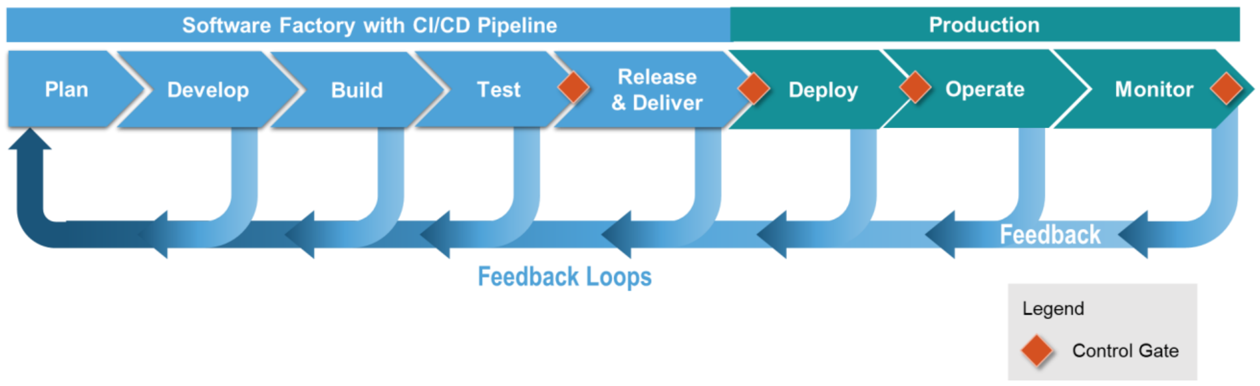

Figure 3. DevSecOps Lifecycle

The phases are iterated as much as necessary. For example, a team may perform a dozen iterations of Plan-Develop-Build-Test before moving to the Deliver phase. The pipeline tools along with various environments (e.g., Development, Test, Staging and Production) are typically deployed in a cloud, which is why they warrant mention in this Playbook.

Security must be built-in to each phase. For example, the Develop phase should include tools integrated into the development environment that check for security coding errors as the code is written, while the Test phase must include both static and dynamic application security testing. Control gates determine whether a piece of software passes to the next phase or not. These gates can ensure that the software passes various security tests at appropriate points in the process.

Furthermore, the pipeline should include automatic creation of a Software Bill of Materials (SBOM) and perform Software Composition Analysis (SCA) to help mitigate risk to the software supply chain. To learn more about DevSecOps, see the DoD CIO Library, which has several papers on that topic. Start with the *DoD Enterprise DevSecOps Fundamentals* [6].

Key Terms

Here are definitions of a few key terms related to DevSecOps in the DoD.

DevSecOps pipeline – "a collection of DevSecOps tools, upon which the DevSecOps process workflows can be created and executed." – *DoD Enterprise DevSecOps Fundamentals* [6].

DevSecOps Platform (DSOP) – "the set of tools and automation that enables a software factory. It includes the ability to create DevSecOps pipelines with control gates, and to deploy software into development, test, and staging/pre-production environments. It may also deploy into production, depending on the production environment." – *DevSecOps Continuous Authorization Implementation Guide* [7].

DSOPs are typically hosted in a cloud.

Software Factory – "a DSOP combined with the people and processes that support the DSOP, as well as a hosting environment such as a cloud; it includes at least development, test and staging/pre-production environments, and it may include a production environment, as well as other environments such as integration." – [7].

The Software Factory should be based on one of the *DoD Enterprise DevSecOps Reference Designs (RD)* to be found in the DoD CIO Library.

Pipeline Threats

Since these DevSecOps pipelines produce multiple applications and services, so they are prime targets for Malicious Cyber Actors (MCAs) [8].

> "These pipelines make valuable targets for Malicious Cyber Actors (MCAs) as a successful compromise of a CI/CD pipeline could impact both infrastructure and applications. Organizations should follow best practices in securing their organization's CI/CD pipelines, such as strong IAM practices, keeping tools up to date, auditing logs, implementing security scanning, and properly handling secrets." – NSA [8].

> "The CI/CD pipeline is a distinct and separate attack surface from other segments of the software supply chain. MCAs can multiply impacts severalfold by exploiting the source of software deployed to multiple operational environments." – *Defending Continuous Integration/Continuous Delivery (CI/CD) Environments*, NSA, Cybersecurity and Infrastructure Security Agency, 2023, [9].

The next figure, from [9], illustrates some threats to a CI/CD pipeline.

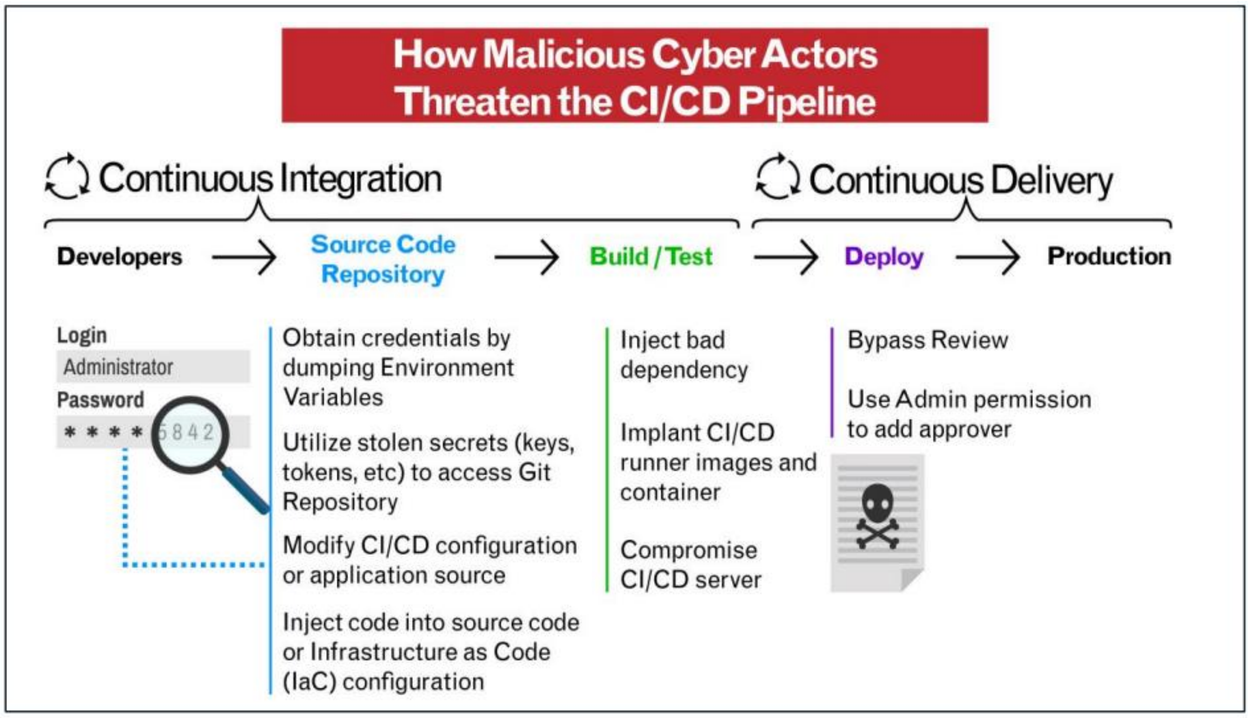
How Malicious Cyber Actors Threaten the CI/CD Pipeline
Continuous Integration
Continuous Delivery
Developers
Source Code Repository
Build / Test
Deploy
Production
Login
Administrator
Password
* * * * 6842
Obtain credentials by dumping Environment Variables
Utilize stolen secrets (keys, tokens, etc) to access Git Repository
Modify CI/CD configuration or application source
Inject code into source code or Infrastructure as Code (IaC) configuration
Inject bad dependency
Implant CI/CD runner images and container
Compromise CI/CD server
Bypass Review
Use Admin permission to add approver

Figure 4. Threats to the Pipeline

Top Ten Pipeline Risks and Mitigations

Another good source of information is the *Open Web Application Security Project® (OWASP) Top 10 CI/CD Security Risks* [10]. That is the source for the next table of top 10 CI/CD security risks with mitigations.

Table 1. Top 10 CI/CD Security Risks with Mitigations

Risk	Evidence of Mitigation
Insufficient Flow Control Mechanisms	Evidence of properly configured pipeline control gates. Evidence of use of GitOps pull, but no push.
Inadequate Identity and Access Management	DSOP and processes verify proper IdAM, including use of signed artifacts and GitOps pull.
Dependency Chain Abuse	Secure the chain via DSOP, processes and training
Poisoned Pipeline Execution (PPE)	Use GitOps. Pull, do not allow a push
Insufficient Pipeline-Based Access Controls (PBAC)	Different Impact Levels (IL) require separate environments; limit permissions; revert execution node to pristine state after each execution; separate context for installation scripts
Insufficient Credential Hygiene	Verify proper credential processes are in place. Verify that team is properly trained on credentials.

Risk	Evidence of Mitigation
Insecure System Configuration	Use IaC and GitOps; least privilege principle
Ungoverned Usage of 3rd Party Services	Governance processes in place, Visibility over ongoing usage
Improper Artifact Integrity Validation	Code signing with external authority, artifact verification automation, configuration drift detection automation
Insufficient Logging and Visibility	Verify that DSOP provides logging and log analysis tools, as well as a dashboard to display them

Continuous Authorization to Operate

In the DoD, the gold standard for pipeline security is obtaining a Continuous Authorization to Operate (cATO) for the software factory that includes the pipeline (see the Glossary for a definition). Developing software in a DoD Software Factory with a Continuous Authorization to Operate (cATO) is normally the fastest way to achieve authorization to deploy an application into production in a cloud.

Using a DoD Software Factory with a Continuous Authorization to Operate (cATO) is normally the fastest way to achieve authorization

More on cATO can be found in the DoD CIO Library, including a set of evaluation criteria that help to mitigate risks to the pipeline. Start with the *DevSecOps Continuous Authorization Implementation Guide* [7] to learn about the concept, then peruse the *DevSecOps Continuous Authorization to Operate (cATO) Evaluation Criteria* [11], which includes many details on how to secure the pipeline and prepare a submission for a cATO.

Actions

Zero Trust Mitigation

- ☐ Use a zero-trust approach. Assume no user, endpoint device or process is fully trusted.

The next set of mitigations to employ come from [9].

Authentication and Access Mitigations

- ☐ Use encryption with a FIPS 140-2 approved algorithm.

- ☐ Minimize use of long-term credentials. To authenticate people, use identity federation and phishing-resistant security tokens to obtain temporary SSH and other keys.
- ☐ Implement secure code signing to establish trust within the pipeline. See *Security Considerations for Code Signing*, NIST [12], which contains several recommendations.
- ☐ Use two-person rules for code, at least one other developer must approve code (or IaC, or Policy as Code) before it can be promoted to the main branch. Some organizations may require more than one reviewer.
- ☐ Implement least-privilege policies for access to the pipeline. Developers should only have access to components they need for their tasks, not the entire environment.
- ☐ Secure user accounts
- ☐ Secure secrets
- ☐ "Implement network segmentation and traffic filtering Implement and ensure robust network segmentation between networks and functions to reduce the spread of malware and limit access from other parts of the network that do not need access. Define a demilitarized zone that eliminates unregulated communication between networks. Filter network traffic to prohibit ingress and egress communications with known malicious IP addresses"

Development Environment Mitigations

- ☐ Keep all software up to date and patched, including operating systems and pipeline tools.
- ☐ Remove unnecessary applications.
- ☐ Implement Endpoint Detection and Response (EDR) tools.

Development Process Mitigations

- ☐ Integrate security testing into the pipeline, including Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST).
- ☐ Pull artifacts such as containers and libraries only from a trusted artifact repository in which the artifacts have been scanned.
- ☐ Analyze committed code for security vulnerabilities.

- ☐ Remove any temporary resources, such as those created in testing.
- ☐ Keep audit logs that include who committed, reviewed and deployed, what they deployed, when and where.
- ☐ Implement the generation of a Software Bill of Materials (SBOM) and perform software composition analysis (SCA). Do this both for the pipeline tools and the software moving through the pipeline. The SBOM must be in a standard SBOM format and must include all third-party and open-source components. The SBOM must be compared with known vulnerabilities to determine if any components increase the risk beyond acceptable thresholds.
- ☐ Build a resilient pipeline and test its resiliency.

Play 21. Mitigate Third Party Risk

> "Recent cyberattacks such as those executed against SolarWinds and its customers and exploits that take advantage of vulnerabilities such as Log4j, highlight weaknesses within software supply chains, an issue which spans both commercial and open-source software and impacts both private and Government enterprises. Accordingly, there is an increased need for software supply chain security awareness and cognizance regarding the potential for software supply chains to be weaponized by nation state adversaries using similar tactics, techniques, and procedures (TTPs)." – *Securing the Software Supply Chain* [13].

Here are some examples of common threats that can occur "during the software development lifecycle:

1. Adversary intentionally injecting malicious code or a developer unintentionally including vulnerable code within a product.
2. Incorporating vulnerable third-party source code or binaries within a product either knowingly or unknowingly.
3. Exploiting weaknesses within the build process used to inject malicious software within a component of a product.
4. Modifying a product within the delivery mechanism, resulting in injection of malicious software within the original package, update, or upgrade bundle deployed by the customer." – [13].

Secure Software Supply Chain

To mitigate third party risk, secure the software supply chain. As discussed in Play 20, the pipeline should include automatic creation of a Software Bill of Materials (SBOM) and perform Software Composition Analysis (SCA) to help mitigate risk to the software supply chain. Consider using a DoD software factory with a cATO, as this enables pipelines that incorporate features to help secure the software supply chain.

The Enduring Security Framework (ESF) is a public-private cross-sector group that addresses risks to critical infrastructure and National Security Systems. "ESF is chartered by the Department of Defense, Department of Homeland Security (DHS), Office of the Director of National Intelligence (ODNI), and the Information Technology (IT), Communications and Defense Industrial Base Sector Coordinating Councils. NSA serves as the Executive Secretariat of ESF."[2]

[2] Source: https://www.nsa.gov/About/Cybersecurity-Collaboration-Center/Enduring-Security-Framework

The ESF has created several documents related to this play, particularly the following.

- *Securing the Software Supply Chain: Recommended Practices Guide for Developers*, 2022 [13].
- *Securing the Software Supply Chain: Recommended Practices Guide for Customers*, 2022 [14].
- *Securing the Software Supply Chain: Recommended Practices for Managing Open-Source Software and Software Bill of Materials*, 2023 [15].
- *Securing the Software Supply Chain: Recommended Practices for Software Bill of Materials Consumption*, 2023 [16].

Actions

- ☐ Secure the Software Supply Chain.
- ☐ Consider using a DoD software factory with a cATO.
- ☐ Enable automatic creation of a Software Bill of Materials (SBOM) for software produced. The SBOM must be in a standard SBOM format and must include all third-party and open-source components.
- ☐ The SBOM must be compared with known vulnerabilities to determine if any components increase the risk beyond acceptable thresholds.
- ☐ Perform Software Composition Analysis (SCA) to help mitigate risk to the software supply chain.
- ☐ Read the appropriate ESF documents.
  - *Securing the Software Supply Chain: Recommended Practices Guide for Developers*, 2022 [13].
  - *Securing the Software Supply Chain: Recommended Practices Guide for Customers*, 2022 [14].
  - *Securing the Software Supply Chain: Recommended Practices for Managing Open-Source Software and Software Bill of Materials*, 2023 [15].
  - *Securing the Software Supply Chain: Recommended Practices for Software Bill of Materials Consumption*, 2023 [16].

Play 22. Move Towards Zero Trust (ZT)

Zero Trust is "a security model, a set of system design principles, and a coordinated cybersecurity and system management strategy based on an acknowledgement that threats exist both inside and outside traditional network boundaries. Zero Trust repeatedly questions the premise that users, devices, and network components should be implicitly trusted based on their location within the network. Zero Trust embeds comprehensive security monitoring; granular, dynamic, and risk-based access controls; and system security automation in a coordinated manner throughout all aspects of the infrastructure … to focus specifically on protecting critical assets (data) in real-time within a dynamic threat environment. This data-centric security model allows the concept of least privileged access to be applied for every access decision, where the answers to the questions of who, what, when, where, and how are critical for appropriately allowing or denying access to resources." – *Embracing a Zero Trust Security Model*, NSA, Feb 2021, [17].

One DoD strategic objective from the *Fulcrum Information Technology Advancement Strategy*, DoD CIO, June 2024 [18] is to "Implement ZT across DoD networks and compute fabric: Secure networks and compute fabric with ZT to increase resiliency against threats across the full range of conflict."

The next figure is from the *DoD Zero Trust Strategy*, 2022 [19]; it illustrates the pillars of ZT.

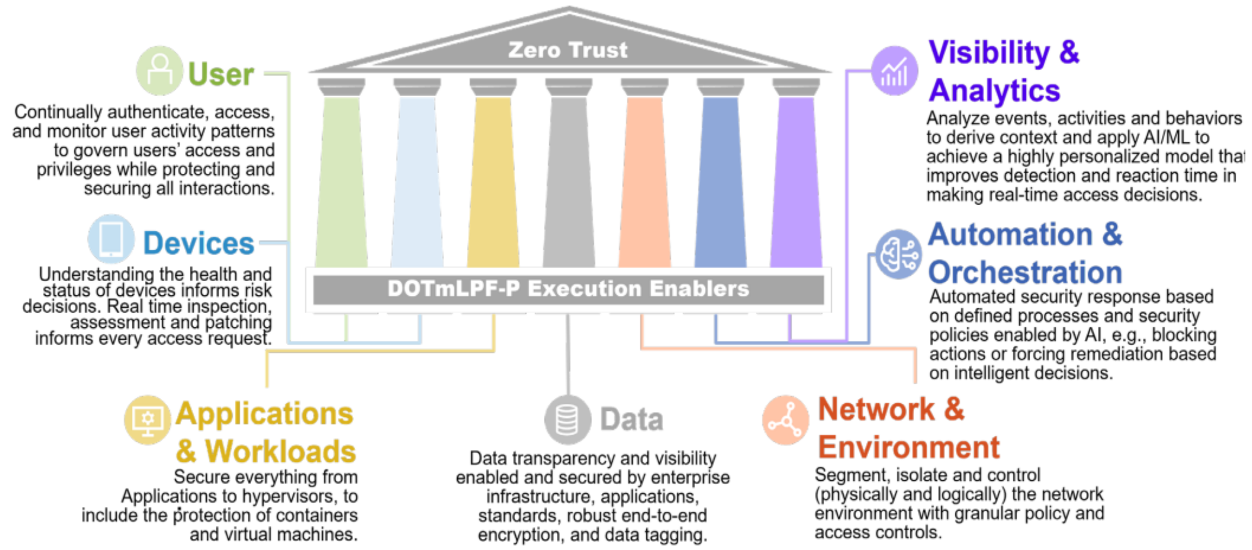
Zero Trust
DOTmLPF-P Execution Enablers
User
Continually authenticate, access, and monitor user activity patterns to govern users' access and privileges while protecting and securing all interactions.
Devices
Understanding the health and status of devices informs risk decisions. Real time inspection, assessment and patching informs every access request.
Applications & Workloads
Secure everything from Applications to hypervisors, to include the protection of containers and virtual machines.
Data
Data transparency and visibility enabled and secured by enterprise infrastructure, applications, standards, robust end-to-end encryption, and data tagging.
Network & Environment
Segment, isolate and control (physically and logically) the network environment with granular policy and access controls.
Automation & Orchestration
Automated security response based on defined processes and security policies enabled by AI, e.g., blocking actions or forcing remediation based on intelligent decisions.
Visibility & Analytics
Analyze events, activities and behaviors to derive context and apply AI/ML to achieve a highly personalized model that improves detection and reaction time in making real-time access decisions.

Figure 5. DoD Zero Trust Pillars

A Zero Trust approach treats every user and device as untrusted, regardless of their network location. It requires conditional access for every request for data or resources.

A complete discussion of how to implement Zero Trust is outside the scope of this Playbook, but here are some resources to guide the transition to zero trust.

- *DoD Zero Trust Strategy*, 2022 [19]
- *DoD Zero Trust Capability Execution Roadmap (COA1)*, DoD CIO, 2023 [20]
- *DoD Zero Trust Reference Architecture*, Version 2.0 July 2022, Defense Information Systems Agency (DISA) and NSA, [21]
- *Embracing a Zero Trust Security Model*, NSA, Feb 2021, [17]
- https://zerotrust.cyber.gov/
- *Zero Trust Maturity Model*, CISA, 2023 [22]
- *Advancing Zero Trust Maturity Throughout the User Pillar*, NSA, 2023 [23]
- *Cloud Security Technical Reference Architecture*, CISA, the U.S. Digital Service, and FedRAMP, 2022 [24]
- *Special Publication 800-207: Zero Trust Architecture*, National Institute of Standards and Technology, 2020 [25].

Major CSPs are moving towards offering Zero Trust solutions. Some example resources include: Microsoft Zero Trust, Microsoft Entra Suite, and Zero Trust on AWS.

Actions

- [ ] Read these DoD papers on ZT:
    - *DoD Zero Trust Strategy*, 2022 [19]
    - *DoD Zero Trust Capability Execution Roadmap (COA1)*, DoD CIO, 2023 [20]
    - *DoD Zero Trust Reference Architecture*, Version 2.0 July 2022, Defense Information Systems Agency (DISA) and NSA, [21]
- [ ] Implement ZT for the mission application.
    - Consider CSP-provided ZT solutions.

Play 23. Secure Artificial Intelligence (AI) Systems

Artificial Intelligence (AI) and Machine Learning (ML) systems have become popular. These systems are most often developed and deployed in a cloud, due to the need for massive compute power and storage. These systems are targets for malicious actors. Moreover, there are new vulnerabilities associated with such systems.

“The rapid adoption, deployment, and use of AI capabilities can make them highly valuable targets for malicious cyber actors. Actors, who have historically used data theft of sensitive information and intellectual property to advance their interests, may seek to co-opt deployed AI systems and apply them to malicious ends. Malicious actors targeting AI systems may use attack vectors unique to AI systems, as well as standard techniques used against traditional IT. Due to the large variety of attack vectors, defenses need to be diverse and comprehensive. Advanced malicious actors often combine multiple vectors to execute operations that are more complex. Such combinations can more effectively penetrate layered defenses. Organizations should consider the following best practices to secure the deployment environment, continuously protect the AI system, and securely operate and maintain the AI system.” – *Deploying AI Systems Securely*, Ver. 1.0, NSA, 2024 [26].

“Securing AI systems requires us to protect the entire AI development lifecycle, an extension of secure software development practices we have today in cybersecurity. Specifically, we need to protect the training data, training frameworks, models, model abilities, and the machine learning (ML) development operations lifecycle.” – NSA Artificial Intelligence Security Center.

The NIST *AI 100-1, Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, 2023 [27] enumerates numerous AI-specific risks that are new or increased. Following are some risks that relate to cybersecurity of AI systems.

- Difficulty in performing regular AI-based software testing, or determining what to test, since AI systems are not subject to the same controls as traditional code development.
- Privacy risk due to enhanced data aggregation capability for AI systems.
- Underdeveloped software testing standards and inability to document AI-based practices to the standard expected of traditionally engineered software for all but the simplest of cases.
- The data used for building an AI system may not be a true or appropriate representation of the context or intended use of the AI system, and the ground truth may either not exist or not be available. Additionally, harmful bias and other data

quality issues can affect AI system trustworthiness, which could lead to negative impacts.

Actions

These actions are best practices for deploying secure and resilient AI systems from [26]. More information can be found there.

Secure the Deployment Environment

- ☐ Manage deployment environment governance.
- ☐ Ensure a robust deployment environment architecture.
- ☐ Harden deployment environment configurations.
- ☐ Protect deployment networks from threats.

Continuously Protect the AI System

- ☐ Validate the AI system before and during use.
- ☐ Secure exposed APIs.
- ☐ Actively monitor model behavior.
- ☐ Protect model weights.

Secure AI Operation and Maintenance

- ☐ Enforce strict access controls.
- ☐ Ensure user awareness and training.
- ☐ Conduct audits and penetration testing.
- ☐ Implement robust logging and monitoring.
- ☐ Update and patch regularly.
- ☐ Prepare for High Availability (HA) and Disaster Recovery (DR).
- ☐ Plan secure delete capabilities.

CSP-Specific Actions for AI

- ☐ If hosting on AWS, consider using the AWS Cloud Adoption Framework for Artificial Intelligence, Machine Learning, and Generative AI.

Play 24. Secure Application Programming Interfaces

Secure Application Programming Interfaces (APIs) and use an API gateway.

Any service provided by a CSP, or the MO will have an API. Moreover, web applications are typically built with one or more backend services, each of which has an API, and one or more client applications (such as a browser or a mobile application) that call these APIs. An API provides a standard interface to interact with the service. Using an API helps decouple the implementation of the backend services from the interface, so that the services can change without necessarily changing the API. Naturally, APIs have become major targets for MCAs.

It is important to secure these APIs. API Security "focuses on strategies and solutions to understand and mitigate the unique vulnerabilities and security risks of Application Programming Interfaces (APIs)."[3]

Be aware of vulnerabilities unique to APIs. Appendix A includes the top ten vulnerabilities related to APIs according to the OWASP API Security Top 10 for 2023. Here is a brief list.

- API1:2023 - Broken Object Level Authorization
- API2:2023 - Broken Authentication
- API3:2023 - Broken Object Property Level Authorization
- API4:2023 - Unrestricted Resource Consumption
- API5:2023 - Broken Function Level Authorization
- API6:2023 - Unrestricted Access to Sensitive Business Flows
- API7:2023 - Server-Side Request Forgery
- API8:2023 - Security Misconfiguration
- API9:2023 - Improper Inventory Management
- API10:2023 - Unsafe Consumption of APIs

Major CSPs have CSOs that help manage and secure APIs. One service they offer is an API gateway, which is a managed service that simplifies maintaining, monitoring, and securing APIs.[4]

Requests from a client application to a backend (CSP or MO) service is routed to the API gateway, which then forwards them to the appropriate service. The API gateway acts as a façade to the backend services, offering a layer of abstraction to enable backend services

[3] Source: https://owasp.org/www-project-api-security/

[4] Source: https://aws.amazon.com/api-gateway/

to evolve without impacting the client applications. The API gateway “enables consistent configuration of routing, security, throttling, caching, and observability.”[5]

The gateway:

- Acts as a façade to backend services
- Verifies API keys and other credentials such as tokens and certificates presented with requests
- Enforces usage quotas and rate limits
- Optionally transforms requests and responses as specified in policy statements
- If configured, caches responses to improve response latency and minimize the load on backend services
- Emits logs, metrics, and traces for monitoring, reporting, and troubleshooting [6]

Actions

- ☐ Enable an API gateway to help manage and secure APIs.
- ☐ Consider using other CSOs related to APIs that are offered by the selected CSP.

[5] Source: https://learn.microsoft.com/en-us/azure/api-management/api-management-key-concepts

[6] Ibid.

Conclusion

This volume of the Cloud Security Playbook has discussed several important topics, including securing containers and microservices, defending DevSecOps pipelines, and securing AI systems.

Together with volume 1 the playbook provides numerous actions that mission owners can take to significantly improve their security in a cloud.

References

[1] R. Chandramouli, "NIST Special Publication 800-204, Security Strategies for Microservices-based Application Systems," August 2019. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-204.

[2] R. Chandramouli and Z. Butcher, "NIST Special Publication 800-204A, Building Secure Microservices-based Applications Using Service-Mesh Architecture," May 2020. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-204A.

[3] DISA, "Container Hardening Process Guide, Version 1, Release 2," 24 August 2022. [Online]. Available: https://dl.dod.cyber.mil/wp-content/uploads/devsecops/pdf/Final_DevSecOps_Enterprise_Container_Hardening_Guide_1.2.pdf.

[4] M. Souppaya, J. Morello and K. Scarfone, "NIST Special Publication 800-190, Application Container Security Guide," September 2017. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-190.

[5] DoD CIO, "DoD Enterprise DevSecOps Reference Design: CNCF Kubernetes, Version 2.1," Sep 2021. [Online]. Available: https://dodcio.defense.gov/Library/.

[6] DoD CIO, "DoD Enterprise DevSecOps Fundamentals," September 2021. [Online]. Available: https://dodcio.defense.gov/Library/.

[7] DoD CIO, "DevSecOps Continuous Authorization Implementation Guide," March 2024. [Online]. Available: https://dodcio.defense.gov/Library/.

[8] NSA, "NSA's Top Ten Cloud Security Mitigation Strategies," 2024. [Online]. Available: https://media.defense.gov/2024/Mar/07/2003407860/-1/-1/0/CSI-CloudTop10-Mitigation-Strategies.PDF.

[9] NSA, Cybersecurity and Infrastructure Security Agency (CISA) , "Defending Continuous Integration/Continuous Delivery (CI/CD) Environments," June 2023. [Online]. Available: https://media.defense.gov/2023/Jun/28/2003249466/-1/-1/0/CSI_DEFENDING_CI_CD_ENVIRONMENTS.PDF.

[10] Open Web Application Security Project® (OWASP), "Open Web Application Security Project® (OWASP) Top 10 CI/CD Security Risks," [Online]. Available: https://owasp.org/www-project-top-10-ci-cd-security-risks/.

[11] DoD CIO, "DevSecOps Continuous Authorization to Operate (cATO) Evaluation Criteria," May 2024. [Online]. Available: https://dodcio.defense.gov/Library/ .

[12] NIST, "Security Considerations for Code Signing," January 2018. [Online]. Available: https://doi.org/10.6028/NIST.CSWP.5.

[13] NSA, CISA, ODNI, "Securing the Software Supply Chain: Recommended Practices Guide for Developers," 1 September 2022. [Online]. Available: https://media.defense.gov/2022/Sep/01/2003068942/-1/-1/0/ESF_SECURING_THE_SOFTWARE_SUPPLY_CHAIN_DEVELOPERS.PDF.

[14] ESF, "Securing the Software Supply Chain: Recommended Practices Guide for Customers," 17 November 2022. [Online]. Available: https://www.nsa.gov/About/Cybersecurity-Collaboration-Center/Enduring-Security-Framework/.

[15] ESF, "Securing the Software Supply Chain: Recommended Practices for Managing Open-Source Software and Software Bill of Materials," 11 December 2023. [Online]. Available: https://www.nsa.gov/About/Cybersecurity-Collaboration-Center/Enduring-Security-Framework/.

[16] ESF, "Securing the Software Supply Chain: Recommended Practices for Software Bill of Materials Consumption," 9 November 2023. [Online]. Available: https://www.nsa.gov/About/Cybersecurity-Collaboration-Center/Enduring-Security-Framework/.

[17] NSA, "Embracing a Zero Trust Security Model," Feb 2021. [Online]. Available: https://media.defense.gov/2021/Feb/25/2002588479/-1/-1/-/CSI_EMBRACING_ZT_SECURITY_MODEL_UOO115131-21.pdf.

[18] DoD CIO, "Fulcrum Information Technology Advancement Strategy," 6 June 2024. [Online]. Available: https://dodcio.defense.gov/Portals/0/Documents/Library/FulcrumAdvStrat.pdf.

[19] DoD CIO, "DoD Zero Trust Strategy," 2022.

[20] DoD CIO, "DoD Zero Trust Capability Execution Roadmap (COA1)," 6 Jan 2023. [Online]. Available: https://dodcio.defense.gov/Portals/0/Documents/Library/ZTCapabilitiesActivities.pdf.

[21] Defense Information Systems Agency (DISA) and National Security Agency (NSA), "DoD Zero Trust Reference Architecture Version 2.0," July 2022. [Online]. Available: https://dodcio.defense.gov/Portals/0/Documents/Library/(U)ZT_RA_v2.0(U)_Sep22.pdf.

[22] CISA, "Zero Trust Security Maturity Model," April 2023. [Online]. Available: https://www.cisa.gov/sites/default/files/2023-04/zero_trust_maturity_model_v2_508.pdf.

[23] NSA, "Advancing Zero Trust Maturity Throughout the User Pillar," March 2023. [Online]. Available: https://media.defense.gov/2023/Mar/14/2003178390/-1/-1/0/CSI_Zero_Trust_User_Pillar_v1.1.PDF.

[24] Cybersecurity and Infrastructure Security Agency, United States Digital Service, and the Federal Risk and Authorization Management Program, "Cloud Security Technical Reference Architecture, version 2.0," June 2022. [Online]. Available: https://www.cisa.gov/sites/default/files/2023-05/Cloud%20Security%20Technical%20Reference%20Architecture%20v2.pdf.

[25] National Institute of Standards and Technology, "Special Publication 800-207: Zero Trust Architecture," 2020. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-207/final.

[26] NSA, "Deploying AI Systems Securely, Ver. 1.0," April 2024. [Online]. Available: https://media.defense.gov/2024/Apr/15/2003439257/-1/-1/0/CSI-DEPLOYING-AI-SYSTEMS-SECURELY.PDF.

[27] NIST, "AI 100-1, Artificial Intelligence Risk Management Framework (AI RMF 1.0)," January 2023. [Online]. Available: https://doi.org/10.6028/NIST.AI.100-1.

[28] U.S. Congress, "National Artificial Intelligence Initiative Act of 2020 (enacted as Division E of the William M . (Mac) Thornberry National Defense Authorization Act for Fiscal Year 2021 (Public Law 116-283), Section 5002(3)," 2021. [Online]. Available: https://www.congress.gov/bill/116th-congress/house-bill/6216/text#toc-H41B3DA72782B491EA6B81C74BB00E5C0.

[29] DISA, "Cloud Service Provider (CSP) Security Requirements Guide (SRG), Version 1, Release 1," 14 June 2024. [Online]. Available: https://public.cyber.mil/dccs/dccs-documents/.

[30] NIST, "NIST SPECIAL PUBLICATION 1800-19, Trusted Cloud: Security Practice Guide for VMware Hybrid Cloud Infrastructure as a Service (IaaS) Environments," April 2022. [Online]. Available: https://doi.org/10.6028/NIST.SP.1800-19.

[31] National Cyber Security Centre (NCSC), "The Near-term Impact of AI on the Cyber Threat, NCSC," 24 January 2024. [Online]. Available: https://www.ncsc.gov.uk/report/impact-of-ai-on-cyber-threat.

[32] United States Department of Homeland Security, "Cyber Safety Review Board Releases Report on Microsoft Online Exchange Incident from Summer 2023," 2 April 2024. [Online]. Available: https://www.dhs.gov/news/2024/04/02/cyber-safety-review-board-releases-report-microsoft-online-exchange-incident-summer.

Appendix A. Glossary

Term	**Definition**
Artificial Intelligence (AI)	AI is a machine-based system that can, for a given set of human-defined objectives, make predictions, recommendations or decisions influencing real or virtual environments. Artificial intelligence systems use machine and human-based inputs to a) perceive real and virtual environments; b) abstract such perceptions into models through analysis in an automated manner; and c) use model inference to formulate options for information or action. (Source: National Artificial Intelligence Initiative Act of 2020 (Public Law 116-283) Section 5002(3) [28]).
Cloud Service Offering (CSO)	A CSO is a service offered by a CSP. Each CSP provides many different CSOs.
Cloud Service Provider (CSP)	A CSP is an entity that offers one or more cloud services in one or more deployment models. Each CSP provides many CSOs. – (Source: *Cloud Service Provider (CSP) Security Requirements Guide (SRG)*, Version 1, Release 1, DISA,14 June 2024 [29].
Cloud workload	A logical bundle of software and data that is present in, and processed by, a cloud computing technology. (Source: NIST SP 1800-19 [30]).
Continuous Authorization to Operate (cATO)	Continuous Authorization to Operate (cATO) is the state achieved when the organization that develops, secures, and operates a system has demonstrated sufficient maturity in their ability to maintain a resilient cybersecurity posture that traditional risk assessments and authorizations become redundant. This organization must have implemented robust information security continuous monitoring capabilities, active cyber defense, and secure software supply chain requirements to enable continuous delivery of capabilities without adversely impacting the system's cyber posture. (Source: *DevSecOps Continuous Authorization Implementation Guide* [7]).

DevSecOps pipeline	A collection of DevSecOps tools, upon which the DevSecOps process workflows can be created and executed. (Source: *DoD Enterprise DevSecOps Fundamentals* [6]).
DevSecOps Platform (DSOP)	The set of tools and automation that enables a software factory. It includes the ability to create DevSecOps pipelines with control gates, and to deploy software into development, test, and staging/pre-production environments. It may also deploy into production, depending on the production environment. (Source: *DevSecOps Continuous Authorization Implementation Guide* [7]).
Generative AI	AI that can generate new content, such as text, images or video. Large Language Models (LLMs) are an example of generative AI. (Source: *The near-term impact of AI on the cyber threat*, 2024 [31]).
Infrastructure as a Service (IaaS)	The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls). (Source: NIST Glossary).
Large Language Model (LLM)	A large language model (LLM) is a specialized type of artificial intelligence (AI) that has been trained on vast amounts of text to understand existing content and generate original content. (Source: Gartner Glossary)
Machine Learning (ML)	Machine Learning is an application of artificial intelligence that is characterized by providing systems the ability to automatically learn and improve on the basis of data or experience, without being explicitly programmed. (Source: National Artificial Intelligence Initiative Act of 2020 (Public Law 116-283) Section 5002(3) [28]).
Platform as a Service (PaaS)	The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers,

	operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment. (Source: NIST Glossary)
Software as a Service (SaaS)	The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings. (Source: NIST Glossary).
Software Factory	A DSOP combined with the people and processes that support the DSOP, as well as a hosting environment such as a cloud; it includes at least development, test and staging/pre-production environments, and it may include a production environment, as well as other environments such as integration. (Source: *DevSecOps Continuous Authorization Implementation Guide* [7]).
Threat	Any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service. Also, the potential for a threat-source to successfully exploit a particular information system vulnerability. (Source: NIST Glossary).
Vulnerability	A weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source. (Source: NIST Glossary)

Appendix B. Acronyms

Acronym	Definition
3PAO	Third Party Assessment Organization
ACAS	Assured Compliance Assessment Solution
AD	Active Directory
AI	Artificial Intelligence
AO	Authorizing Official
API	Application Programming Interface
APT	Advanced Persistent Threat
AST	Application Security Testing
ATO	Authorization to Operate
ATT&CK	Adversarial Tactics, Techniques & Common Knowledge
AWS	Amazon Web Services
BCAP	Boundary Cloud Access Point
BCD	Boundary Cyberspace Defense
BOM	Bill of Materials
C-ITP	Cloud Information Technology Project
CAC	Common Access Card
CAO	Connection Approval Office
CAP	Cloud Access Point
CATC	Cloud Authorization to Connect
CAVEaT	Cloud Adversarial, Vectors, and Threats
CC	Cloud Computing
CC SRG	Cloud Computing Security Requirements Guide
CD	Continuous Delivery
CDR	Cloud Detection and Response
CERT	Computer Emergency Readiness Team
CI	Continuous Integration
CI/CD	Continuous Integration / Continuous Delivery
CIEM	Cloud Infrastructure Entitlement Management
CIO	Chief Information Officer
CISA	Cybersecurity and Infrastructure Security Agency
CNAP	Cloud Native Access Point
CNAPP	Cloud-Native Application Protection Platform
CNCF	Cloud Native Computing Foundation
CND	Computer Network Defense
CNDSP	Computer Network Defense Service Provider
CNSA	Commercial National Security Algorithm
CNSS	Committee on National Security Systems
CNSSI	Committee on National Security Systems Instruction
CNSSP	Committee on National Security Systems Policy
COA	Course of Action

Acronym	Definition
COOP	Continuity of Operations
CPTC	Cloud Permission to Connect
CPTs	Cyber Protection Teams
CPU	Central Processing Unit
CRL	Certificate Revocation List
CRT	Continuous Risk Treatment
CS	Cybersecurity
CSA	Cloud Security Alliance
CSO	Cloud Service Offering
CSP	Cloud Service Provider
CSPM	Cloud Security Posture Management
CSSP	Cybersecurity Service Provider
CUI	Controlled Unclassified Information
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
CWP	Cloud Workload Protection
D3FEND	Detection, Denial, and Disruption Framework Empowering Network Defense
DAST	Dynamic Application Security Testing
DB	database
DCAS	DoD Cloud Authorization Services
DCAT	DoD Cyber Assessment Team
DCD	DODIN Cyberspace Defense
DCO	Defensive Cyberspace Operations
DCRT	DoD Cyber Red Team
DevSecOps	Development Security Operations
DFARS	Defense Federal Acquisition Regulation Supplement
DHS	Department of Homeland Security
DISA	Defense Information Systems Agency
DISN	Defense Information Systems Network
DMZ	demilitarized zone
DoD	Department of Defense
DoDI	Department of Defense Instruction
DODIN	DoD Information Network
DR	Disaster Recovery
DSAWG	DOD Security/Cybersecurity Authorization Working Group
DSOP	DevSecOps Platform
DSS	DISN Subscription Service
DTM	Directive-type Memorandum
EaC	Everything as Code
EDR	Endpoint Detection and Response
eMASS	Enterprise Mission Assurance Support Service
ESF	Enduring Security Framework

Acronym	Definition
ETL	Extract, Transform, and Load
FE	Federated Entity
FedRAMP	Federal Risk and Authorization Management Program
FIDO	Fast IDentity Online
FIPS	Federal Information Processing Standard
HA	High Availability
HIDS	Host Intrusion Detection System
HSM	Hardware Security Module
HTTP	Hypertext Transfer Protocol
IA	Information Assurance
IaaS	Infrastructure as a Service
IaC	Infrastructure as Code
IAM	Identity and Access Management
IAP	Internet Access Point
IAST	Interactive Application Security Testing
IATT	Interim Authorization to Test
ICAM	Identity, Credential, and Access Management
ID	Identification
IdAM	Identify and Access Management
IDM	Internal Defensive Measures
IDS	Intrusion Detection System
IE	Information Enterprise
IL	Impact Level
IMDS	Instance Metadata Service
IoT	Internet of Things
IP	Internet Protocol
IPS	Intrusion Prevention System
IT	Information Technology
JFHQ	Joint Force Headquarters
JIT	Just-in-Time
JWCC	Joint Warfighter Cloud Capability
KM	Key Management
KMS	Key Management System
LLM	Large Language Model
MCA	Malicious Cyber Actor
MCD	Mission Cyberspace Defense
MFA	Multi-Factor Authentication
ML	Machine Learning
MO	Mission Owner
MOSA	Modular Open System Approach
MPE	Mission Partner Environment
mTLS	mutual Transport Layer Security

Acronym	Definition
NCSC	National Cyber Security Centre
NIDS	Network Intrusion Detection System
NIPRNet	Non-classified Internet Protocol Router Network
NIST	National Institute of Standards and Technology
NPE	Non-Person Entity
NSA	National Security Agency
NSS	National Security Systems
OCI	Open Container Initiative
OCSP	Online Certificate Status Protocol
ODNI	Office of the Director of National Intelligence
OS	Operating System
OSI	Open Systems Interconnection
OWASP	Open Web Application Security Project
PA	Provisional Authorization
PaaS	Platform as a Service
PaC	Policy as Code
PAW	Privileged Access Workstation
PBAC	Pipeline-Based Access Controls
PE	Person Entity
PIN	Personal Identification Number
PK	Public Key
PKI	Public Key Infrastructure
POA&M	Plan of Action and Milestones
PoLP	Principle of Least Privilege
PPE	Poisoned Pipeline Execution
RD	Reference Design
RME	Risk Management Executive
RMF	Risk Management Framework
SaaS	Software as a Service
SAST	Static Application Security Testing
SBOM	Software Bill of Materials
SCA	Software Composition Analysis
SCAP	Security Content Automation Protocol
SDN	Software Defined Network
SDP	Software Defined Perimeter
SIEM	Security Information and Event Management
SIPRNet	Secret Internet Protocol Router Network
SLA	Service Level Agreement
SNAP	System Network Approval Process
SOAR	Security Orchestration, Automation, and Response
SOC	Security Operations Center
SP	Special Publication

Acronym	Definition
SQL	Structured Query Language
SRG	Security Requirements Guide
SSC	Sidecar Security Container
SSH	Secure Shell
SSL	Secure Sockets Layer
SSP	System Security Plan
SSRF	Server-Side Request Forgery
STIG	Security Technical Implementation Guide
TAG	Technical Advisory Group
TLS	Transport Layer Security
TTP	Tactics, Techniques, and Procedures
U.S.	United States
UEBA	User and Entity Behavior Analytics
URI	Uniform Resource Identifier
US	United States
US-CERT	United States - Computer Emergency Readiness Team
VDMS	Virtual Datacenter Managed Service
VDSS	Virtual Datacenter Security Stack
VM	Virtual Machine
VNet	Virtual Network
VPC	Virtual Private Cloud
VPN	Virtual Private Network
WAF	Web Application Firewall
XDR	Extended Detection and Response
ZT	Zero Trust
ZTA	Zero Trust Architecture
ZTNA	Zero Trust Network Access