Defending Continuous Integration/Continuous Delivery (CI/CD) Environments

Executive summary

The National Security Agency (NSA) and the Cybersecurity and Infrastructure Security Agency (CISA) are releasing this cybersecurity information sheet (CSI) to provide recommendations and best practices for improving defenses in cloud implementations of development, security, and operations (DevSecOps). This CSI explains how to integrate ***security*** best practices into typical software development and operations (DevOps) Continuous Integration/Continuous Delivery (CI/CD) environments, without regard for the specific tools being adapted, and leverages several forms of government guidance to collect and present proper security and privacy controls to harden CI/CD cloud deployments. As evidenced by increasing compromises over time, software supply chains and CI/CD environments are attractive targets for malicious cyber actors (MCAs). Figure 1 provides a high-level representation of threats to various parts of the CI/CD pipeline.

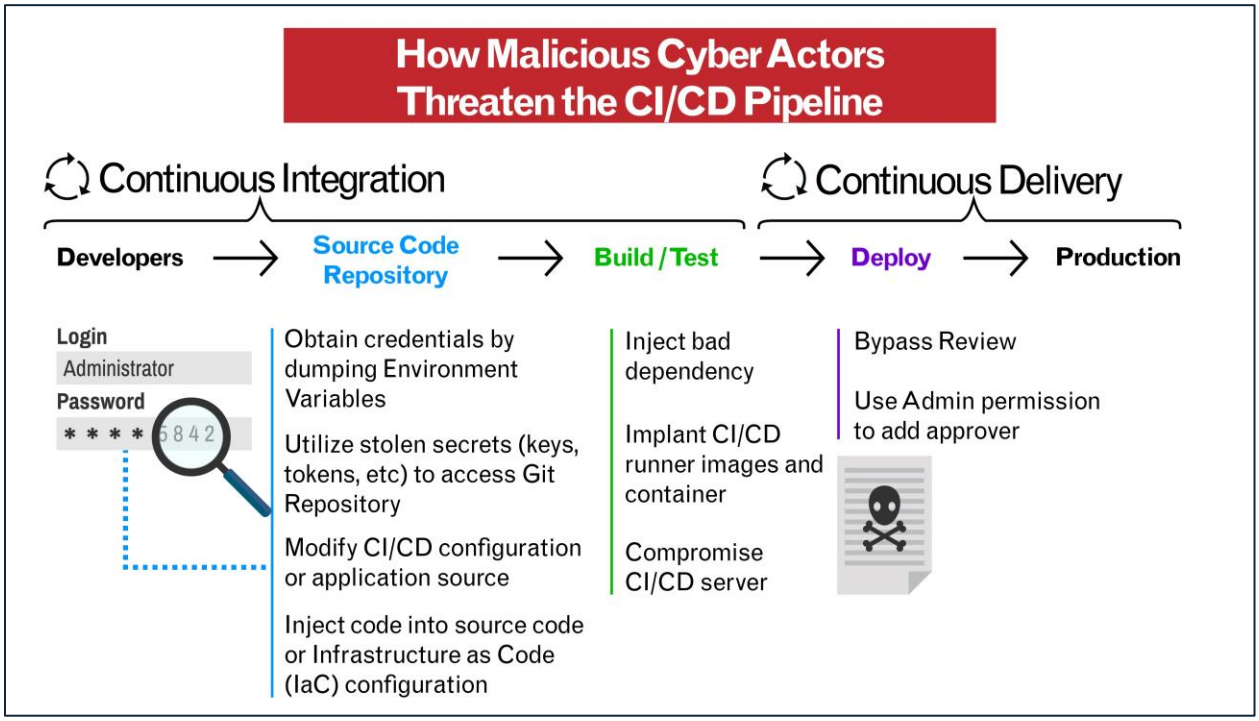

Figure 1: Threats to the CI/CD pipeline

Contents

Figures

Introduction

Continuous Integration/Continuous Delivery (CI/CD) is a development process for quickly building and testing code changes that helps organizations maintain a consistent code base for their applications while dynamically integrating code changes. CI/CD is a key part of the development, security, and operations (DevSecOps) approach that integrates security and automation throughout the development lifecycle. CI/CD pipelines are often implemented in commercial cloud environments because of the cloud's role in IT modernization efforts. Organizations are constantly leveraging CI/CD-focused tools and services to securely streamline software development and manage applications and clouds' programmable infrastructure. Therefore, CI/CD environments are attractive targets for malicious cyber actors (MCAs) whose goals are to compromise information by introducing malicious code into CI/CD applications, gaining access to intellectual property/trade secrets through code theft, or causing denial of service effects against applications.

NSA and CISA authored this CSI to provide recommendations and best practices for hardening CI/CD pipelines against MCAs to secure DevSecOps CI/CD environments, regardless of the tools being adapted. It outlines key risks for CI/CD deployments, using the MITRE ATT&CK® framework to enumerate the most significant potential CI/CD vulnerabilities based on known threats. See Appendix A for details on the tactics, techniques, and countermeasures for the threats mapped to ATT&CK and D3FEND™. NSA and CISA encourage organizations to implement the proposed mitigations to harden their CI/CD environments and bolster organizational DevSecOps. By implementing the proposed mitigations, organizations can reduce the number of exploitation vectors into their CI/CD environments and create a challenging environment for the adversary to penetrate.

This CSI utilizes several government guides to collect and present the proper security and privacy controls to harden CI/CD environments.

Key terms

Continuous delivery (**CD**) is the stage after continuous integration where code changes are deployed to a testing and/or staging environment after the build stage.

Continuous deployment is similar to continuous delivery except that the releases happen automatically, and changes to code are available to customers immediately after they are made. The automatic release process may in many instances include A/B testing to facilitate slow rollout of new features, thereby mitigating the impact of failure resulting from a bug or error. [1]

Continuous integration (CI) involves developers frequently merging code changes into a central repository where automated builds and tests run. Build is the process of converting the source code to executable code for the platform on which it is intended to run. In the CI/CD pipeline software, the developer's changes are validated by creating a build and running automated tests against the build. This process avoids the integration challenges that can happen when waiting for release day to merge changes into the release branch. [1]

A **CI/CD pipeline** is a component of a broader toolchain that entails continuous integration, version control, automated testing, delivery, and deployment. It automates the integration and delivery of applications and enables organizations to deploy applications quickly and efficiently. [2]

Development operations (DevOps) is a set of practices that combines software development and information technology (IT) operations. It aims to shorten the systems development lifecycle and provide continuous delivery with high software quality.

DevSecOps (DevSecOps) is an approach that integrates development (Dev), security (Sec), and delivery/operations (Ops) of software systems to reduce the time from a recognized need to capability availability and provide continuous integration and continuous delivery (CI/CD) with high software quality.

A **software supply chain** is composed of the components, libraries, tools, and processes used to develop, build, and publish a software artifact. Software vendors often create products by assembling open source and commercial software components. Software supply chains are made up of software components, such as

open source packages and infrastructure as code (IaC) templates, as well as underlying delivery pipelines, such as version control systems and CI/CD pipelines.

Software composition analysis is an automated process that identifies the software in a code base. This analysis is performed to evaluate security, license compliance, and code quality.

CI/CD security threats

Software supply chains and CI/CD environments are attractive targets for MCAs. CI/CD pipeline compromises are increasing. Recognizing the various types of security threats that could affect CI/CD operations and taking steps to defend against each one are critical to securing a CI/CD environment. Common examples of risks in CI/CD pipelines are listed here. For a more comprehensive description, refer to the OWASP Top 10 CI/CD Security Risks. [3]

- **Insecure first-party code**: Code that is checked in by authorized developers but that contains security-related bugs that are not detected by either the software developers or by security tooling.
- **Insecure third-party code**: Insecure code that is compiled into a CI/CD pipeline from a third-party source, such as an open source project.
- **Poisoned pipeline execution**: Exploitation of a development/test/production environment that allows the attacker to insert code of its choosing.
- **Insufficient pipeline access controls**: Unauthorized access to source code repositories or build tools.
- **Insecure system configuration**: Various infrastructure, network, and application configurations vulnerable to known exploitation techniques.
- **Usage of insecure third-party services**: Using services created by an external individual or organization that intentionally or negligently includes security vulnerabilities.
- **Exposure of secrets**: Security key compromise and insecure secrets management within the pipeline, such as hardcoding access keys or passwords into infrastructure as code (IaC) templates.

Attack surface

An insecure CI/CD pipeline can easily lead to an insecure application. Some of the attack surfaces that MCAs can exploit are as follows:

Insecure code

Insecure code within a CI/CD pipeline can include code defects from the authorized developers, open source components, or third-party integrations that are not effectively evaluated or vetted. Rapid development without proper security can introduce vulnerabilities and expose the pipeline to critical risks. Integration of third-party code and lack of code scanning of source code components can introduce vulnerabilities into a CI/CD pipeline. Not following code security best practices can significantly increase the vulnerable attack surface. Common code vulnerabilities include buffer overflows, format string vulnerabilities, and improper error handling.

Poisoned pipeline execution

Poisoned pipeline execution (PPE) is a technique that MCAs use to poison the CI pipeline. This technique allows MCAs to abuse permissions in source code management repositories to manipulate the build process. During this type of compromise, an MCA injects malicious code or commands into the build pipeline configuration, poisoning the pipeline to run malicious code during the build process. [3]

Insufficient pipeline access controls

An MCA might abuse the lack of access control permissions to pivot in a CI/CD pipeline, which could allow them to inject malicious code into an application. CI/CD pipeline-based access controls (PBAC) grant or deny access to resources and systems inside and outside the execution environment. Pipeline execution nodes use these resources to perform various actions. See OWASP CICD-Sec-5: Insufficient PBAC for more detail.

Insecure system configuration

An MCA can exploit system misconfigurations in a CI/CD environment. A CI/CD system may contain various infrastructure, network, and application configurations. These configurations impact the security posture of the CI/CD pipeline and its susceptibility to exploitation. See OWASP CICD-Sec-7: Insecure System Configuration for more detail.

Usage of third-party services

Third-party services are often utilized in CI/CD pipelines. This integration facilitates rapid development and delivery. An MCA can take advantage of the improper usage of third-party services to introduce security weaknesses into the pipeline. Examples of third-party services are GitLab, GitHub, and Travis CI.

Exposure of secrets

MCAs have exploited CI/CD pipelines by using exposed secrets to gain initial access. Secrets (e.g., private keys and passwords) are required for authentication between tools and in the build and deployment process to ensure deployed resources have access. Cloud native CI/CD tools employ numerous secrets to gain access to many sensitive resources, such as databases and codebases.

Threat scenarios

The following are three potential threat scenarios to consider and their corresponding mitigations. These scenarios are not all-inclusive, so consider other threat scenarios as well that are relevant to a particular CI/CD environment based on its attack surface.

- **Scenario 1:** MCAs acquire a developer's credential to access a Git repository service (e.g., stolen personal token, SSH key, browser cookie, or login password). Typically, an MCA will go after 1) valid accounts for a source code repository, 2) valid accounts for a CI/CD Service, or 3) valid admin account of a server hosting a source code repository.
    - **Recommended mitigations:**
        - Minimize the use of long-term credentials.
        - Utilize two-person rules (2PR) for all code updates.
        - Secure user accounts.
        - Implement least-privilege policies for CI/CD access.
        - Implement network segmentation and traffic filtering [CPG 2.F].
- **Scenario 2:** Supply chain compromise of an application library, tool, or container image in a CI/CD pipeline that leads to a poisoned DevSecOps environment.
    - **Recommended mitigations:**
        - Restrict untrusted libraries and tools.
        - Analyze committed code.
        - Implement endpoint detection and response (EDR) tools and auditing.
        - Keep CI/CD tools up-to-date.
        - Maintain up-to-date software and operating systems.
- **Scenario 3:** Supply chain compromise of a CI/CD environment that 1) modifies the CI/CD configuration, 2) injects code into the IaC configuration, 3) injects code into the source code, or 4) injects a malicious or vulnerable dependency.
    - **Recommended mitigations:**

- Analyze committed code.
- Integrate security scanning as part of the CI/CD pipeline.
- Keep audit logs.
- Implement EDR tools.
- Add signature to CI/CD configuration and verify it.
- Implement software bill of materials (SBOM) and software composition analysis (SCA).

The following figure illustrates different attack vectors using the example of a CI/CD pipeline hosted in Amazon Web Services (AWS) and using common AWS services. Similarly, these attack vectors apply to other CI/CD pipelines generally, whether hosted in the cloud or on-premises.

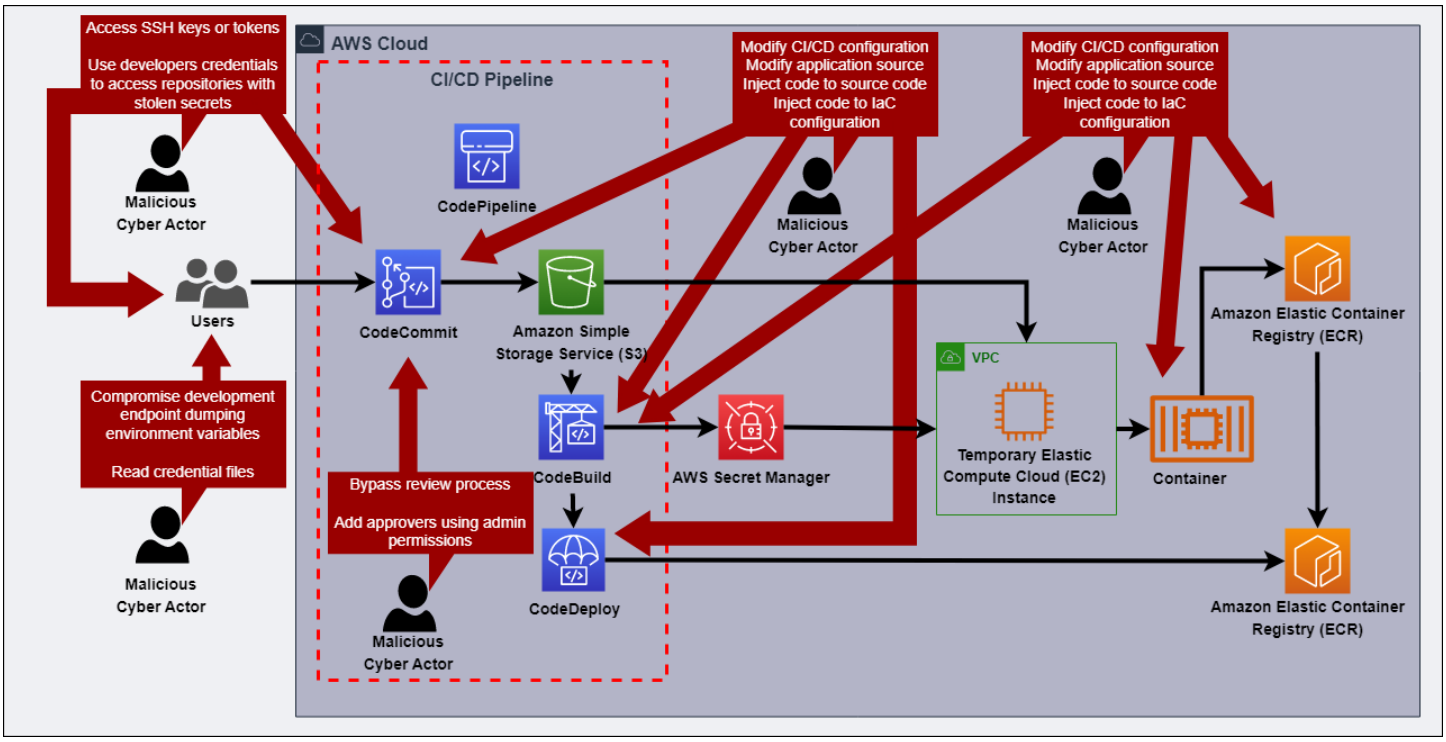

Figure 2: Different attack vectors in an AWS CI/CD pipeline

Active hardening

NSA and CISA recommend organizations implement the following mitigations to help secure CI/CD environments. A zero trust approach, where no user, endpoint device, or process is fully trusted, will help detect and prevent successful compromise of the environment. [4] Organizational transition to zero trust can be aided by referencing CISA's Zero Trust Security Maturity Model and NSA's Advancing Zero Trust Maturity Throughout the User Pillar.

Authentication and access mitigations

Use NSA-recommended cryptography

NSA and CISA recommend the implementation and configuration of strong cryptographic algorithms when configuring cloud applications and services. Proper implementation and configuration of these algorithms augments the protection of data, secrets, application programming interfaces (APIs), and keys generated across the CI/CD pipeline. The utilization of weak and outdated cryptographic algorithms poses a threat to CI/CD pipelines, which may result in sensitive data exposure, data leakage, broken authentication, and insecure sessions—violations that MCAs could exploit to circumvent the CI/CD pipeline and software supply chain.

National Security Systems (NSS) are required to use the algorithms in the NSA-approved Commercial National Security Algorithm (CNSA) Suite (see Annex B of Committee on National Security Systems Policy (CNSSP) 15).

Non-NSS U.S. Government systems are required to use the algorithms as specified by the National Institute of Standards and Technology (NIST), which includes the algorithms approved to protect NSS. NSA and CISA recommend using the cryptographic implementations that have undergone testing, such as Federal Information Processing Standards (FIPS) validation [CPG 2.K].

Minimize the use of long-term credentials

Use strong credentials that are resistant to stealing, phishing, guessing, and replaying wherever and whenever possible. For human authentication, always use identity federation and phishing-resistant security tokens to obtain temporary SSH and other keys. For software-to-software authentication, avoid using software-based long-term credentials as much as possible.

In cloud environments, take advantage of cloud-provided temporary and ephemeral credentials that are available for compute services. For non-cloud environments, where long-term credentials sometimes must be used (e.g., boot-strapping authentication based on public keys in x509 certificates), carefully manage and protect all associated private keys.

Add signature to CI/CD configuration and verify it

NSA and CISA recommend implementing secure code signing to establish digital trust within the CI/CD pipeline. Ensure code is continuously and properly signed and that the

signature is verified throughout the CI/CD process, regardless of the stage of development. If the signature does not validate, investigate the cause of the validation issue. It could be a minor configuration problem or a sign of a larger breach.

Code signing establishes the authenticity of new releases. If a code signing identity itself is compromised, it undermines trust. NIST's Security Considerations for Code Signing describes the concept of digitally signing code for data integrity and source authentication. It also explains features and architectural relationships of typical code signing solutions that are widely deployed today to support various use cases.

Utilize two-person rules (2PR) for all code updates

No single developer should be able to check in code without another developer reviewing and approving the changes. This practice not only increases code quality generally, it also means that the compromise of a single developer's credentials is much less likely to result in malicious code being successfully checked in.

Implement least-privilege policies for CI/CD access

The CI/CD pipeline should not be accessible by everyone in the organization. If personnel request access, they should not automatically receive access to all pipelines, but only limited access with certain privileges [CPG 2.E]. Developers should only have access to the pipelines they are tasking and the components they are updating. Separation of duties should be implemented. For example, developers checking in source code do not need the privilege for updating the build environment, and engineers in charge of builds do not need read-write source code access. For more detail on implementing security controls, see NIST SP 800-53.[1] Use well-authenticated, concurrent versioning systems and keep long histories of changes tagged to specific users.

Mitigate password risks by implementing multi-factor authentication (MFA). Enforce MFA for users within and outside the organization and complement it with role-based access control (RBAC), following the principle of the least privilege [CPG 2.H]. [5]

[1] NIST 800-53 rv5 Control NIST IDs that aligns to implementing security controls AC-2 (1), AC-2 (2), AC-2 (4), AC-2 (9), AC-2 (10), AC-03 (07), SC-07 (05), AC-17(2), SC-7, SC-8(1), AC-5, AC-6, AC-6 (1), AC-6 (2), AC-6 (3), AC-6 (4), AC-6 (5), AC-6 (9), AC-6 (10), and SC-23

Secure user accounts

Regularly audit administrative user accounts and configure access controls under the principles of least privilege and separation of duties. Audit logs to ensure new accounts are legitimate [CPG 2.E]. [6]

Secure secrets

Secure handling of secrets, tokens, and other credentials is crucial in a CI/CD pipeline. Never pass secrets in plaintext anywhere in the pipeline. Ensure secrets (e.g., passwords and private keys) are never left embedded in software where they can be reverse-engineered out. [7] Most modern CI/CD tools come with a secrets management solution, which means a CI/CD tool can securely store the secrets and pass them using an indirect reference within the pipeline [CPG 2.L]. [8]

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 [CPG 2.F].

Development environment mitigations

Maintain up-to-date software and operating systems

NSA and CISA recommend upgrading operating systems and software on all devices, including development systems and all CI/CD assets, to the latest stable versions supplied by the vendors. Upgrading the operating system may require additional hardware or memory upgrades, and obtaining a new software version may require a maintenance or support contract with the vendor. Consider using a centralized patch management system that includes a software integrity and validation process, ensuring that the software has not been maliciously altered in transit. For a list of centralized patch management system examples, visit Info-Tech Research Group's reviews on patch management.

Maintaining up-to-date operating systems and software protects against critical vulnerabilities and security issues that have been identified and fixed in newer releases. Devices running outdated operating systems or vulnerable software are susceptible to a

variety of known vulnerabilities, and exploiting these devices is a common technique used by MCAs to compromise a network [CPG 1.E].

Scan for vulnerabilities and keep software updated**.** Ensure antivirus/antimalware programs are updated with current signatures and set to conduct regular scans of network assets [CPG 1.E].

Keep CI/CD tools up-to-date

Update the CI/CD tools on a regular schedule. Like other software programs, CI/CD tools may contain bugs and vulnerabilities. Failure to update CI/CD tools leaves the pipeline vulnerable and allows an MCA to more easily exploit known vulnerabilities [CPG 1.E].

Remove unnecessary applications

Remove any application not deemed necessary for day-to-day operations.

Implement endpoint detection and response (EDR) tools

EDR tools provide a high degree of visibility into the security status of endpoints and can help effectively protect against MCAs.

Development process mitigations

Integrate security scanning as part of the CI/CD pipeline

Include security scanning early in the CI/CD process. The following tools should be employed to detect security flaws in CI/CD pipelines: [9]

- **Static application security testing (SAST):** Include a static code analysis tool in the build stage to check the code for common security vulnerabilities and compliance issues.
- **Registry scanning:** Scan every image pulled into the pipeline.
- **Dynamic analysis security testing (DAST):** Deploy an instance of the newly built application to a testing environment and run tests against the application.

No automated security scanning tool is perfect, so it is important to manually review the code and the CI/CD pipeline. By understanding how the pipeline works and what could go wrong, the team can ensure that the pipeline is as secure as possible.

Restrict untrusted libraries and tools

Only use software, tools, libraries, and artifacts from secure and trusted sources. Employing software from a trusted source helps minimize the threats posed to the CI/CD pipeline and prevent potential exploitation (i.e., code execution and backdoors) by MCAs. Adopting a “never trust/always verify” approach toward software reduces overall CI/CD attack surface.

Analyze committed code

Securing the CI/CD pipeline involves analyzing the code that is being committed, which can be achieved manually or by using automated tools. Automated tools can identify potential security vulnerabilities in the code and track changes over time. Analyzing the code ensures that only approved changes are made to the code base and that any potential security vulnerabilities are addressed before they can be exploited. [10], [11]

Remove any temporary resources

A CI/CD pipeline may also create temporary resources, such as virtual machines or Kubernetes clusters, to run tests. While test environments are usually always live, these temporary resources are meant to be created for a single test purpose and must be destroyed after the pipeline run. Failure to destroy these allocations can provide additional attack vectors to the system that can pose a security threat, placing the CI/CD pipeline at risk.

Keep audit logs

An audit log should provide clear information on who committed, reviewed, and deployed what, when, and where. If all previous measures fail, an audit log will at least help forensically reconstruct an incident post-compromise, so it can be quickly addressed [CPG 2.T, 2.U].

Implement software bill of materials (SBOM) and software composition analysis (SCA)

An SBOM and SCA can play a useful role in the software development lifecycle (SDLC) and in DevSecOps by helping to track all third-party and open source components in the codebase. A vulnerability management team will need to evaluate correlations of SBOM data to known common vulnerabilities and exposures (CVEs) [CPG 1.E]. SBOM should reflect vulnerabilities as they are found. It is recommended that SBOM results be ingested into a security information and event management (SIEM) solution, immediately making the data searchable to identify security vulnerabilities across a fleet

of products. This information can also be converted into a human-readable, tabular format for other data analysis systems. Once an SBOM is available for a given piece of software, it needs to be mapped onto a list of known vulnerabilities to identify the components that could pose a threat. NSA and CISA recommend connecting these two sources of information. See the National Telecommunications and Information Administration's Minimum Elements for an SBOM and CISA's SBOM pages for further reference. Beware, however, that malicious actors can manipulate the content of SBOMs as well as other artifacts in the pipeline, so SBOMs cannot be presumed accurate.

Plan, build, and test for resiliency

Build the pipeline for high availability, and test for disaster recovery periodically. Consider availability (in addition to confidentiality and integrity) threats to the pipeline during its threat modeling.

Ensure the CI/CD pipeline can elastically scale so that new artifacts can be built and deployed across all the compute instances quickly, as was necessary to address Log4Shell exposure in many environments. Consider including coverage for emergency patch updates in service level agreements (SLA).

Conclusion

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. By exploiting a CI/CD environment, MCAs can gain an entryway into corporate networks and access sensitive data and services.

As a subcomponent of DevSecOps, defending the CI/CD pipeline requires focused and intentional effort. Keep MCAs out by the following recommended guidance to secure and harden the CI/CD attack surface. This is essential for ensuring a strong cybersecurity posture for National Security Systems (NSSs); the Department of Defense (DoD); the Defense Industrial Base (DIB); federal, state, local, tribal, and territorial (SLTT) governments; and private sector information system owners.

Further guidance

Supplementary NSA guidance on ensuring a secure and defensible network environment is available at www.nsa.gov/cybersecurity-guidance. Of particular relevance are:

- NSA's Top Ten Cybersecurity Mitigation Strategies
- Defend Privileges and Accounts
- Continuously Hunt for Network Intrusions
- Segment Networks and Deploy Application-aware Defenses
- Transition to Multi-factor Authentication
- Actively Manage Systems and Configurations
- Performing Out-of-Band Network Management
- Hardening SIEM Solutions
- Mitigating Cloud Vulnerabilities
- Securing the Software Supply Chain for Developers
- Securing the Software Supply Chain: Recommended Practices Guide for Suppliers

Additional CISA guidance includes:

- Multifactor Authentication
- Implementing Phishing Resistant MFA
- CISA Releases Cloud Security Technical Reference Architecture
- CISA Releases SCuBA Hybrid Identity Solutions Architecture Guidance Document for Public Comment
- ESF Identity Hardening Guidance

Works cited

[1] Pittet S. (2021), "Continuous integration vs. continuous delivery vs. continuous deployment." https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

[2] National Institute of Standards and Technology March (2022), "Special Publication 800-204C: Implementation of DecSecOPs for a Microservices-based Application with Service Mesh." https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-204C.pdf

[3] OWASP Foundation (2023), "OWASP Top 10 CI/CD Security Risks." https://owasp.org/www-project-top-10-ci-cd-security-risks/

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

[5] National Institute of Standards and Technology (2020), "Special Publication 800-53: Security and Privacy Controls for Information Systems and Organizations." https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf
[6] Department of Defense (2020), "Identity, Credential, and Access Management (ICAM) Strategy." https://dodcio.defense.gov/Portals/0/Documents/Cyber/ICAM_Strategy.pdf
[7] Hill M. (2023), "Hard-coded secrets are up 67% as secrets sprawl threatens software supply chain." https://www.csoonline.com/article/3689892/hard-coded-secrets-up-67-as-secrets-sprawl-threatens-software-supply-chain.html
[8] National Institute of Standards and Technology (2022), "Key Management Guidelines." https://csrc.nist.gov/projects/key-management/key-management-guidelines
[9] National Institute of Standards and Technology (2022), "Special Publication 800-218: Secure Development Framework (SSDF) Version 1.1: Recommendation for Mitigating the Risk of Software Vulnerabilities." https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218.pdf
[10] Department of Defense (2021), "DevSecOps Playbook." https://dodcio.defense.gov/Portals/0/Documents/Library/DevSecOps%20Playbook_DoD-CIO_20211019.pdf
[11] Department of Defense (2019), "DoD Enterprise DevSecOps Reference Design." https://dodcio.defense.gov/Portals/0/Documents/DoD%20Enterprise%20DevSecOps%20Reference%20Design%20v1.0_Public%20Release.pdf

Disclaimer of endorsement

The information and opinions contained in this document are provided "as is" and without any warranties or guarantees. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement, recommendation, or favoring by the United States Government, and this guidance shall not be used for advertising or product endorsement purposes.

Purpose

This document was developed in furtherance of the authoring cybersecurity authorities' cybersecurity missions, including their responsibilities to identify and disseminate threats to information systems, and to develop and issue cybersecurity specifications and mitigations. This information may be shared broadly to reach all appropriate stakeholders.

Contact

Cybersecurity Report Feedback: CybersecurityReports@nsa.gov
General Cybersecurity Inquiries: Cybersecurity_Requests@nsa.gov
Defense Industrial Base Inquiries and Cybersecurity Services: DIB_Defense@cyber.nsa.gov
CISA's 24/7 Operations Center to report incidents and anomalous activity: Report@cisa.gov or (888) 282-0870
Media Inquiries / Press Desk:

- NSA Media Relations: 443-634-0721, MediaRelations@nsa.gov
- CISA Media Relations: 703-235-2010, CISAMedia@cisa.dhs.gov

Appendix A: CI/CD threats mapped to MITRE ATT&CK

The first step to securing a CI/CD pipeline is determining the threats and vulnerabilities within the build and deployment process that require additional security. Threat modeling can help map threats to the pipeline. Additionally, inventory all CI/CD connections and treat them as potential points of compromise. The MITRE ATT&CK for Enterprise framework is a globally accessible knowledge base of adversary tactics and techniques based on real-world observations.

MITRE's D3FEND provides a one-stop shop for understanding defensive cyber techniques and demonstrates the power of collaboration across the public and private sectors in countering malicious cyber activity.

The following figure and tables list the ATT&CK techniques an MCA may use to exploit CI/CD pipelines, as well as the D3FEND mitigations to counter these malicious activities.

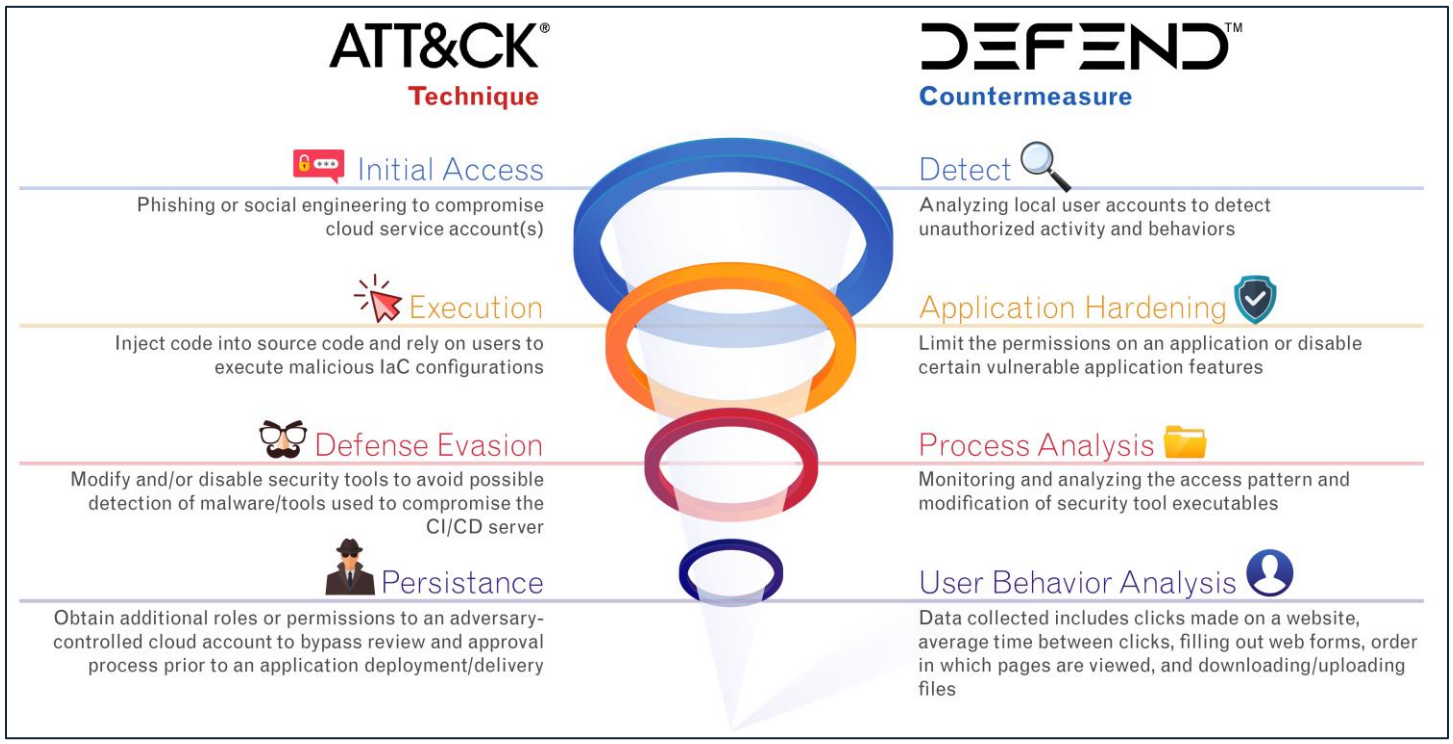

Figure 3: CI/CD attack vectors mapped to ATT&CK techniques and D3FEND countermeasures

Initial Access

From the perspective of a cybersecurity practitioner, begin by following the standard MITRE ATT&CK Matrix definition for Initial Access. Initial Access consists of techniques that MCAs use to gain an initial foothold within a network. For Initial Access, an MCA

may employ the following ATT&CK Tactics and Techniques/Sub-Techniques to exploit a DevSecOps CI/CD cloud environment:

ATT&CK Tactic	Technique
Initial Access	Supply Chain Compromise [T1195]
Initial Access	Compromise Software Supply Chain [T1195.002]
Initial Access	Valid Accounts [T1078]

D3FEND enumerates the following mitigations to counter these techniques:

D3FEND Tactic	Countermeasure
Application Hardening	Application Configuration Hardening [D3-ACH]
User Behavior Analysis	Local Account Monitoring [D3-LAM]

Execution

Execution consists of techniques that result in adversary-controlled code running on a blue space system. Techniques that run malicious code are often paired with techniques from any other tactics to achieve broader goals, such as exploring a network or stealing data. For Execution, an MCA may employ the following ATT&CK Tactic, Techniques/Sub-Techniques to exploit a DevSecOps CI/CD cloud environment:

ATT&CK Tactic	Technique
Execution	Container Administration Command [T1609]
Execution	Command and Scripting Interpreter [T1059]

D3FEND enumerates the following mitigations to counter these techniques:

D3FEND Tactic	Countermeasure
Application Hardening	Application Configuration Hardening [D3-ACH]
Execution Isolation	Executable Allow Listing [D3-EAL]

Persistence

Persistence consists of techniques that adversaries use to keep access to systems across restarts, changed credentials, and other interruptions that could cut off their access. Techniques used for Persistence include any access, action, or configuration changes that let them maintain their foothold on systems, such as replacing or hijacking legitimate code or adding startup code. For Persistence, an MCA may employ the

following ATT&CK Tactic, Techniques/Sub-Techniques to exploit a DevSecOps CI/CD cloud environment:

ATT&CK Tactic	Technique
Persistence	Create Account [T1136]
Persistence	Account Manipulation [T1098]
Persistence	Additional Cloud Credentials [T1098.001]
Persistence	Additional Email Delegate Permissions [T1098.002]
Persistence	Additional Cloud Roles [T1098.003]
Persistence	SSH Authorized Keys [T1098.004]
Persistence, Privilege Escalation	Create or Modify System Process [T1543]
Persistence, Privilege Escalation	Event Triggered Execution [T1546]
Persistence	Implant Internal Image [T1525]

D3FEND enumerates the following mitigations to counter these techniques:

D3FEND Tactic	Countermeasure
Platform Monitoring	Endpoint Health Beacon [D3-EHB]
User Behavior Analysis	Local Account Monitoring [D3-LAM]

Privilege Escalation

Privilege Escalation consists of techniques that adversaries use to gain higher-level permissions on a system or network. Adversaries can often enter and explore a network with unprivileged access but require elevated permissions to follow through on their objectives. Common approaches are to take advantage of system weaknesses, misconfigurations, and vulnerabilities. For Privilege Escalation, an MCA may employ the following ATT&CK Tactic, Techniques/Sub-Techniques to exploit a DevSecOps CI/CD cloud environment:

ATT&CK Tactic	Technique
Privilege Escalation	Cloud Accounts [T1078.004]

D3FEND enumerates the following mitigations to counter these techniques:

D3FEND Tactic	Countermeasure
Platform Monitoring	Endpoint Health Beacon [D3-EHB]

Defense Evasion

Defense Evasion consists of techniques that adversaries use to avoid detection throughout their compromise. Techniques used for Defense Evasion include uninstalling/disabling security software or obfuscating/encrypting data and scripts. Adversaries also leverage and abuse trusted processes to hide and masquerade their malware. For Defense Evasion, an MCA may employ the following ATT&CK Tactic, Techniques/Sub-Techniques to exploit a DevSecOps CI/CD cloud environment:

ATT&CK Tactic	Technique
Defense Evasion, Persistence, Privilege Escalation, Initial Access	Cloud Accounts [T1078.004]
Defense Evasion	Exploitation for Defense Evasion [T1211]
Collection	Data from Cloud Storage [T1530]
Persistence	Implant Internal Image [T1525]
Credential Access, Collection	Adversary-in-the-Middle [T1557]
Execution	Container Administration Command [T1609]
Defense Evasion, Execution	Deploy Container [T1610]
Privilege Escalation	Escape to Host [T1611]
Discovery	Container and Resource Discovery [T1613]
Defense Evasion, Lateral Movement	Use Alternate Authentication Material [T1550]
Defense Evasion	Impair Defenses [T1562]

D3FEND enumerates the following mitigations to counter these techniques:

D3FEND Tactic	Countermeasure
Platform Monitoring	Endpoint Health Beacon [D3-EHB]

Credential Access

Credential Access consists of techniques for stealing credentials, such as account names and passwords. Techniques used to get credentials include keylogging or credential dumping. Using legitimate credentials can give adversaries access to systems, make adversaries harder to detect, and provide adversaries with the opportunity to create more accounts to help achieve their goals. For Credential Access,

an MCA may employ the following ATT&CK Tactic, Techniques/Sub-Techniques to exploit a DevSecOps CI/CD cloud environment:

ATT&CK Tactic	Technique
Credential Access	Multi-Factor Authentication Interception [T1111]
Credential Access	Exploitation for Credential Access [T1212]
Credential Access	Steal Application Access Token [T1528]
Credential Access	SAML Tokens [T1606.002]
Credential Access	Private Keys [T1552.004]
Credential Access, Defense Evasion, Persistence	Modify Authentication Process [T1556]

D3FEND enumerates the following mitigations to counter these techniques:

D3FEND Tactic	Countermeasure
Application Hardening	Application Configuration Hardening [D3-ACH]
Credential Hardening	Multi-Factor Authentication [D3-MFA]
Credential Hardening	One-time Password [D3-OTP]
Harden	Credential Hardening [D3-CH]
Credential Hardening	User Account Permissions [D3-UAP]

Lateral Movement

Lateral Movement consists of techniques that adversaries use to exploit and control remote systems on a network. Achieving their primary objective often requires exploring the network to find their target and subsequently gaining access to it. Reaching their objective often involves pivoting through multiple systems and accounts. For Lateral Movement, an MCA may employ the following ATT&CK Tactic, Techniques/Sub-Techniques to exploit a DevSecOps CI/CD cloud environment:

ATT&CK Tactic	Technique
Execution	Command and Scripting Interpreter [T1059]
Privilege Escalation	Exploitation for Privilege Escalation [T1068]
Persistence	Account Manipulation [T1098]
Defense Evasion, Persistence, Privilege Escalation, Initial Access	Valid Accounts [T1078]

D3FEND enumerates the following mitigations to counter these techniques:

D3FEND Tactic	Countermeasure
Platform Monitoring	Endpoint Health Beacon [D3-EHB]

Exfiltration

Exfiltration consists of techniques that adversaries may use to steal data from the network. Once they have collected data, adversaries often package it to avoid detection while moving it back to their own systems. This can include compression and encryption. For Exfiltration, an MCA may employ the following ATT&CK Tactic, Techniques/Sub-Techniques to exploit a DevSecOps CI/CD cloud environment:

ATT&CK Tactic	Technique
Persistence	Create Account [T1136]
Exfiltration	Automated Exfiltration [T1020]
Exfiltration	Exfiltration Over C2 Channel [T1041]
Exfiltration	Exfiltration Over Alternative Protocol [T1048]
Exfiltration	Transfer Data to Cloud Account [T1537]

D3FEND enumerates the following mitigations to counter these techniques:

D3FEND Tactic	Countermeasure
Platform Monitoring	Endpoint Health Beacon [D3-EHB]