Scribe

A USE CASE OF USING SCRIBE TRUST HUB

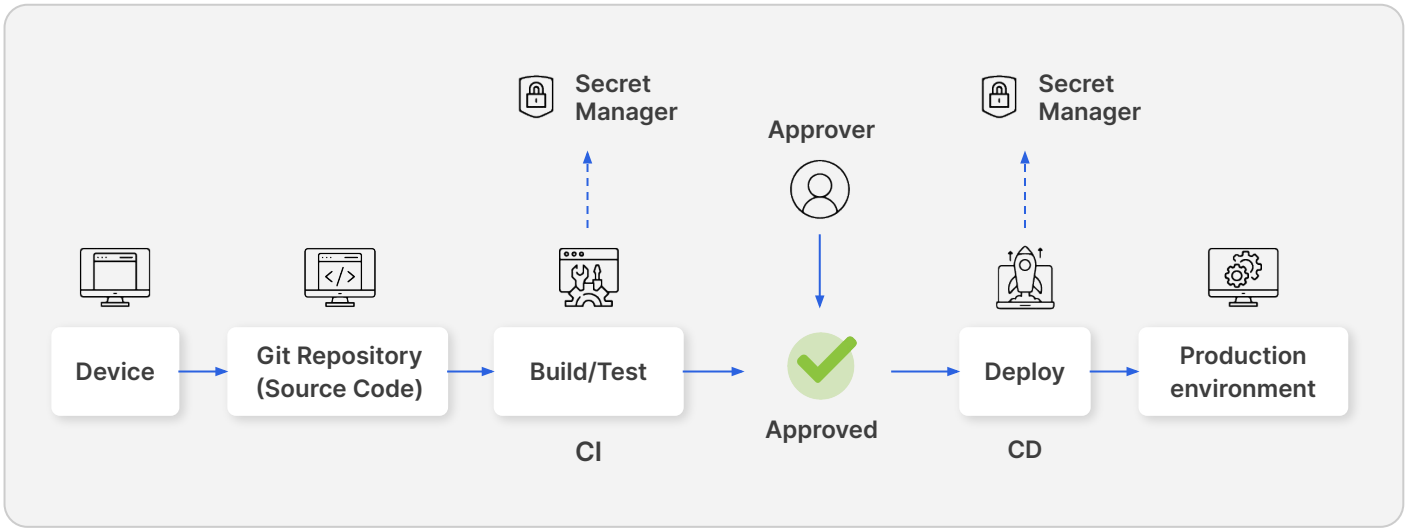
Securing Your CI/CD Pipelines

TABLE OF CONTENTS

BACKGROUND

Software-supply-chain attacks are on the rise and the need to respond to this attack vector has become crucial. But from where to start?

If we had the attacker's war-map, we could prioritize our efforts accordingly. We do not have a specific-attacker war-map, but we do have a blueprint of typical attacker war-maps - the CI\CD threat matrix based on MITRE att&ck framework:



Initial Access	Execution	Persistence	Privilege Escalation	Defence Evasion	Credential Access	Lateral Movement	Exfiltration	Impact
Supply Chain Compromise on CI/CD	Modify CI/CD Configuration	Compromise CI/CD Server	Get credential for Deployment (CD) on CI stage	Add Approver using Admin permission	Dumping Env Variables in CI/CD	Exploitation of Remote Services	Exfiltrate data in Production environment	Denial of Services
Valid Account of Git Repository (Personal Token, SSH key, Login password, Browser Cookies)	Inject code to IaC configuration	Implant CI/CD runner images	Privileged Escalation and compromise other CI/CD pipeline	Bypass Review	Access to Cloud Metadata	(Monorepo) Get credential of different folder's context	Clone Git Repositories	
Valid Account of CI/CD Service (Personal Token, Login password, Browser Cookie)	Inject code to source code	Modify CI/CD Configuration		Access to Secret Manager from CI/CD kicked by different repository	Read credentials file	Privileged Escalation and compromise other CI/CD pipeline		
Valid Admin account of Server hosting Git Repository	Supply Chain Compromise on CI/CD	Inject code to IaC configuration		Modify Caches of CI/CD	Get credential from CI/CD Admin Console			
	Inject bad dependency	Inject code to source code		Implant CI/CD runner images				
	SSH to CI/CD pipelines	Inject bad dependency						
	Modify the configuration of Production environment							
	Deploy modified applications or server images to production environment							

Source

Reading the table from left to right leads us through the attack cycle - the attacker needs to gain initial access, then to execute his code, reach persistence, build up his capabilities by privilege escalation, defense evasion, and credential access. Using these capabilities he can perform lateral movement in order to reach his prey, and then attack - either exfiltrate or cause other impact.

Each column of the table lists techniques the attacker may use. For example, the first column suggests that initial access could be gained by either a prior compromise of the CI\CD, or by weak access controls (valid account of a Git service, valid account of a CI\CD service, or valid admin account of a critical server).

SCRIBE'S APPROACH TO SECURING THE SUPPLY CHAIN

Scribe's evidence-driven approach to securing the supply chain is essentially simple: Trust an artifact only if it is associated with supporting evidence, signed evidence which are defined as digital attestations.

To implement this approach, Scribe supplies the software tools needed to collect and manage such evidence and evaluate the trustworthiness of the artifact according to policies and attestations (signed evidence).

Examples of policies supported by Scribe tools:

Security settings policies: assure that the security settings of the services used through the build process were up to a predefined standard. Examples include use of multi-factor authentication, tight permissions management, ephemeral builds, avoiding or secure usage of "dangerous" capabilities.

Source\File\Module modification policies: assure that source code, configurations, build script and IoC files have been modified according to a predefined standard, that defines the identities, processes, steps and states that are allowed to conduct such modifications.

Source\File\Module integrity policies: assure that source-code, files and modules are identical to predetermined allowed versions.

Dependency Trust Policies: assure dependencies used are up to a predefined standard (such as OSSF Scorecard, version-age, allow\banned list).

Vulnerabilities policies: assure that dependencies and other public-sourced (open or closed) components' vulnerabilities do not pose a high risk on the artifact's security.

Software Development Lifecycle Policies: assure code was reviewed by designated stakeholders, assure that review comments were resolved, assure testing and security testing has been completed successfully.

Pipeline behavior policies: assure that the pipeline behaves as planned. For example, require that an image is pushed as the output of the pipeline and not by some other process.

Examples of evidence Scribe collects - Continuously and automated for every build:

Fine grained SBOMs from the source SCM as well as final artifacts (containers)

Snapshots of the build environment

Snapshots of the source-repo

Security setting of the source-control and build environment

OS-level relevant events collected from the build machine (coming soon)

In Addition, the Scribe tools support collecting user-generated data and utilizing it as evidence.

Signed evidence is regarded as more trusted than plain data. Scribe tools' evidence are signed data artifacts, which assures they are tamper-proof and identifiable.

SCRIBE SOLUTION ON THE ATTACKERS MAP

INITIAL ACCESS

The following Scribe policies can reduce the risk:

- Secure settings policy assure that the source-control and CI access security settings are at a high standard: Validate that commits have multi-factor-authentication, limit high-permissions to a minimal user-group, manage access-keys, etc.
- When run continuously, the secure setting policy will assure policies are up-to-date, and that security-settings-modifications do not go untracked.
- Pipeline behavior policies can be used to assure that access to accounts follow known or predefined access patterns. For example, to assure that code modifications were done on the developer workstation, and not by directly accessing the source control platform.

EXECUTION

Most execution vectors are based on modifying some resource: a configuration, a source-code file, a source-code project file etc.

Scribe's security settings policies, source modification policies will assure that such execution vector was not utilized:

- Security settings policies will assure that only highly trusted personnel have permissions to conduct sensitive modifications.
- Source modification policies will assure that only trusted personnel and processes have changed sensitive files. For high-risk build pipelines, Scribe's OS-level sensor integration will support higher-coverage policies, capable of detecting source-modifications similar to those exercised in the infamous SolarWinds attack.
- Integrity and Dependency Trust policies assure that dependencies and other 3rd party software components do not include or import known-bad or suspected dependencies. This policy can reduce the risk of an attacker utilizing the "inject bad dependency" attack vector.

ATTACK SCENARIOS HANDLED BY SCRIBE SECURITY TECHNOLOGIES

1. Attacker supplies his artifact instead of the software producer.
2. Attacker tampers with the supplied artifact (before delivery or use by consumer).
3. Attacker modifies source code in source-control repo
4. Attacker modifies dependencies consumed by build pipeline
5. Attacker modifies build pipeline scripts
6. Attacker modifies build pipeline cache
7. Attacker modifies build machine and build pipeline tooling
8. Attacker modifies source control settings (as a step in gaining access or causing damage) - modification of authentication method, permissions and branch protection rules.
9. Attacker bypasses checks and tests (as a step in inserting vulnerabilities)
10. Attacker who stole credentials (example: the CircleCI attack) and bypasses the build pipeline (even if he has access to private keys!)

AFTER ATTACK:

Map blast-radius: which artifacts have been produced in a specific pipeline, which artifacts contain contaminated packages etc.

Map vulnerability and malicious package ramifications: which artifacts contains these packages.

Response: detect what was tampered with and where along the pipeline