21st July 2023

OWASP Bangkok Chapter

2600 Thailand Monthly Meeting

Top 10

CI/CD Security Risks

Natworpong Loyswai

Chief Technology Officer

SHIPS
シ ッ プ ス

21st July 2023

WHOAMI

Natworpong Loyswai

Chief Technology Officer
@Shippusu Co., Ltd.

SHIPS
シ ッ プ ス

- Former Microsoft Learn Student Ambassador
- B.Sc. IT @KMUTT

21st July 2023

SHIPS
シ ッ プ ス

SHIPS

Worawit Makkasan

Senior Developer

Boonnithi Jiaramaneepinit

Chief Executive Officer

Sakulthip Rassameecharoentham

Chief Sales Officer

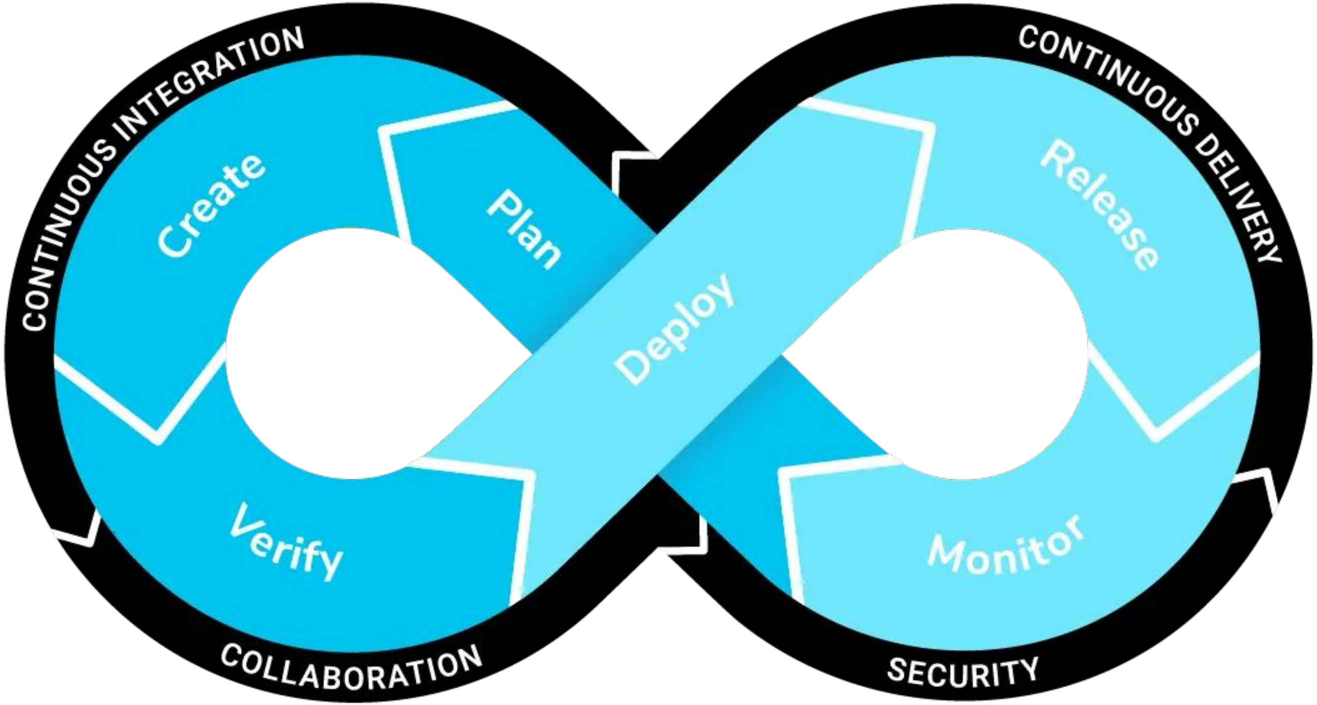
TODAY I'M HERE TO...

- Push myself to learn new things
- Sharing about security risks in your CI/CD pipeline
  - What are CI/CD major security risks?
  - How are they being used for exploitation?
  - What are impacts?
  - Recommendation about how to protect yourself and prevent them?

Continuous Integration / Continuous Delivery

(CI/CD)

DevOps loop

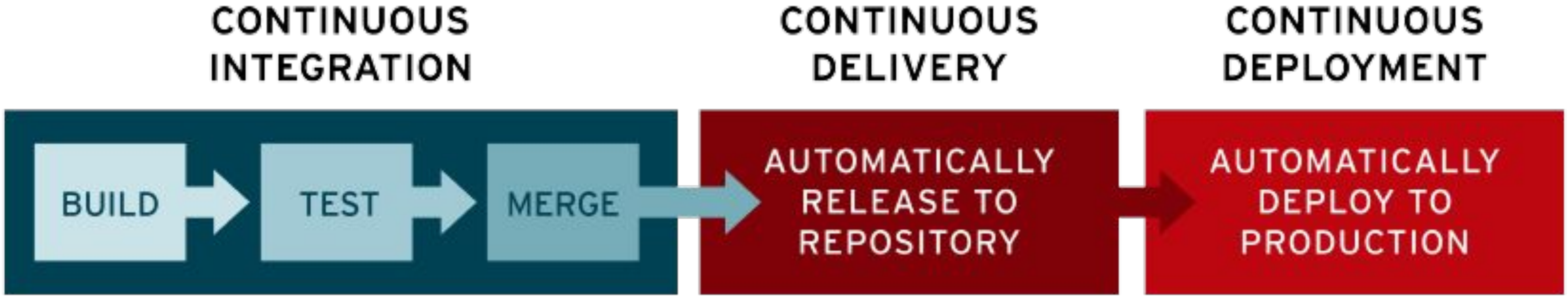

CONTINUOUS INTEGRATION
Create
Plan
Deploy
Release
CONTINUOUS DELIVERY
Verify
Monitor
COLLABORATION
SECURITY

HOWEVER, CI/CD IS NOT A PIPELINE

HOWEVER, CI/CD IS NOT A PIPELINE

A set of best practices and techniques to automate and streamline the process of building, testing, and deploying application.

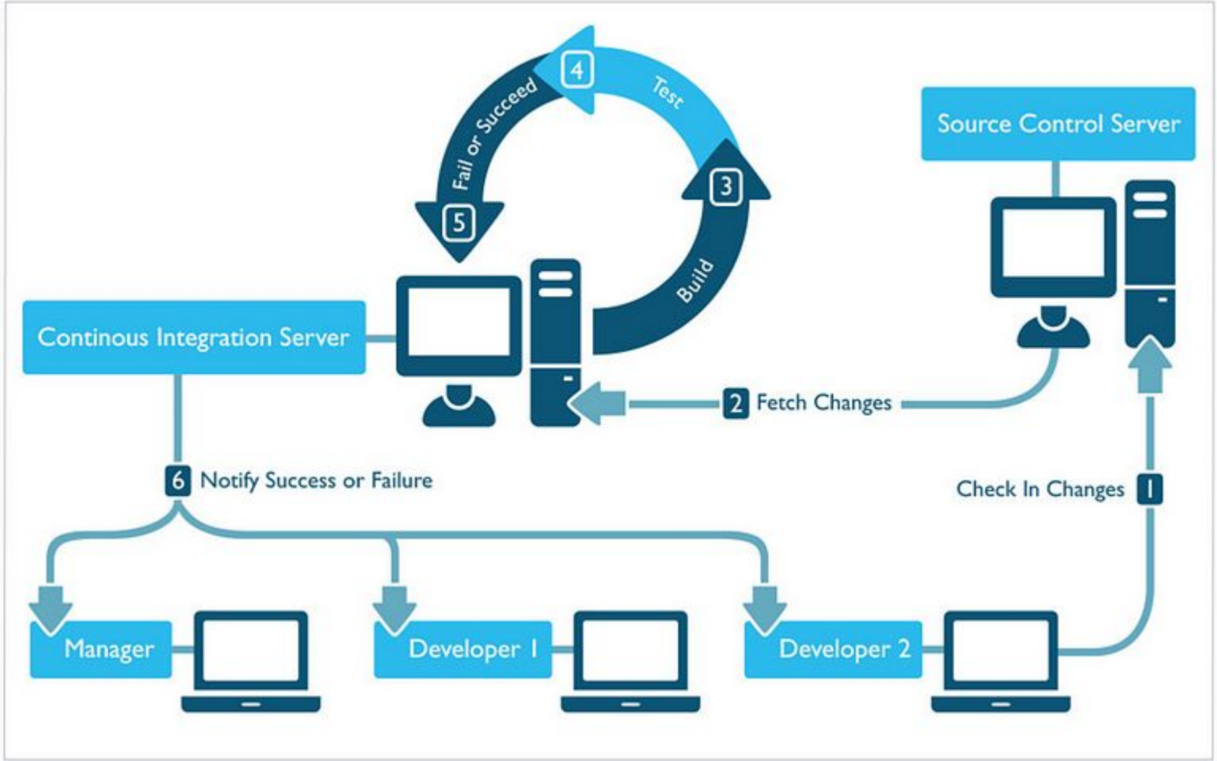
CI/CD Pipeline



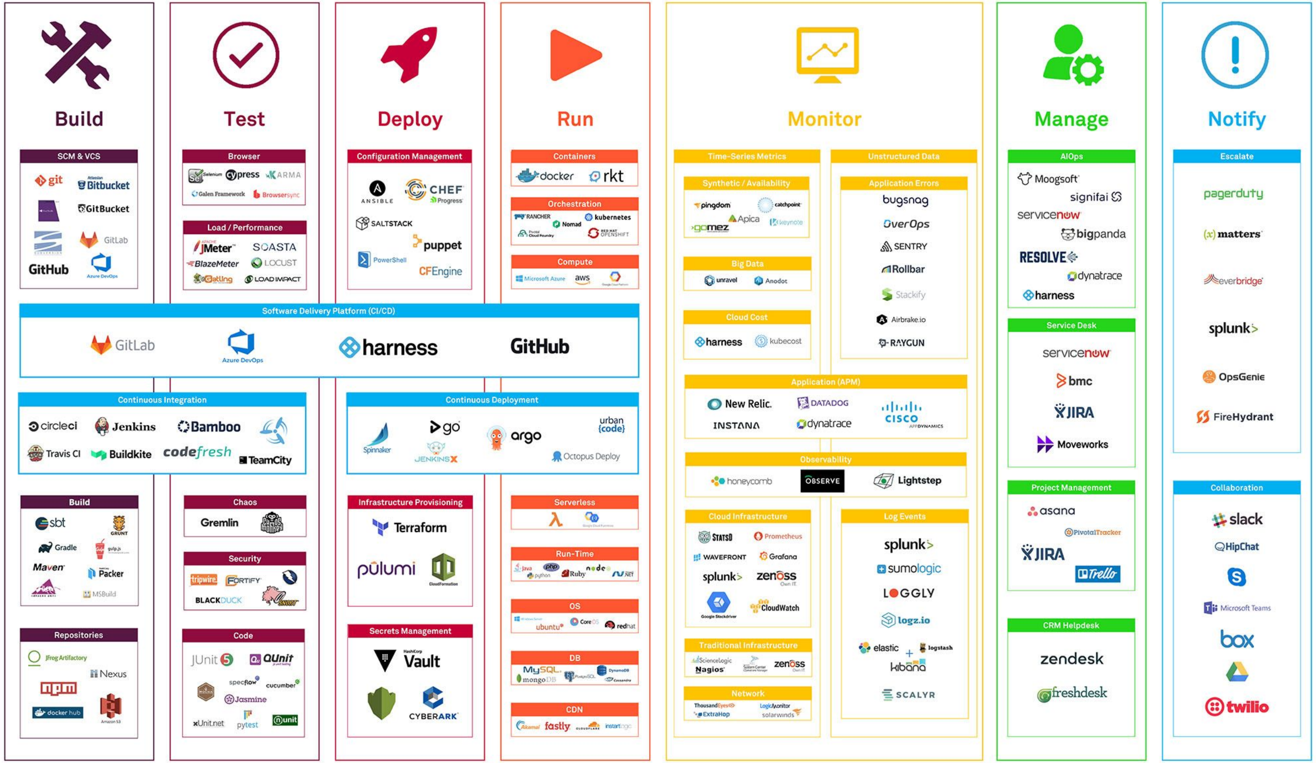
BEFORE CI/CD

FOR HERA'S SAKE, WHO KNEW DELIVERING THESE SERVICES WOULD BE SO HARD?
WHY ISN'T MY CODE UP YET??
OUR UPDATE CYCLES ARE TOO SLOW!
WE CAN'T SCALE UP FAST ENOUGH!
GAME'S GONNA GET STALE, BOSS...
SORRY TO CALL AT 3:00AM, BUT—
OH, CRAP— POWER'S OUT AGAIN!
WHY CAN'T IT JUST WORK ALL THE TIME?!

Continuous Integration



DevOps Tools Ecosystem 2021

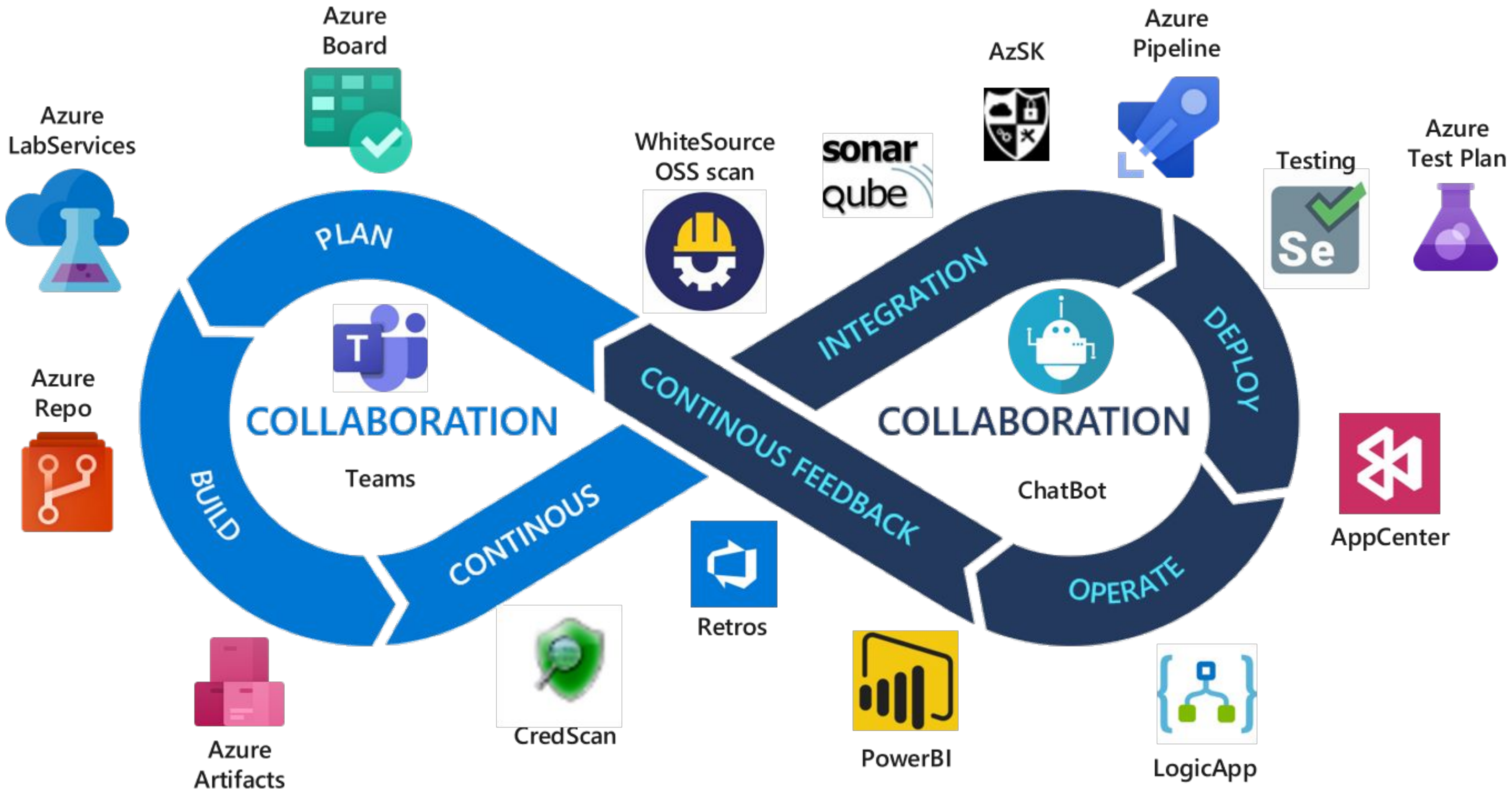


harness

The Platform for Software Delivery
www.harness.io

Copyright © Harness Inc. 2021

SHIPS
シ ッ プ ス



Microsoft Azure

CI/CD PITFALL

There’s more but one example is…

CI/CD PITFALL

There’s more but one example is…

Limited environment challenges

CI/CD PITFALL

There’s more but one example is…

Limited environment challenges

- Limited resources offered to testers.

CI/CD PITFALL

There’s more but one example is…

Limited environment challenges

- Limited resources offered to testers.
- Shared testing environment.

CI/CD PITFALL

There’s more but one example is…

Limited environment challenges

- Limited resources offered to testers.
- Shared testing environment.
- Try Isolation/Containerization.

Top 10 CI/CD Security Risks

Cider

OWASP

paloalto®
NETWORKS

CICD-SEC-1: Insufficient Flow Control Mechanisms

CICD-SEC-2: Inadequate Identity and Access Management

CICD-SEC-3: Dependency Chain Abuse

CICD-SEC-4: Poisoned Pipeline Execution (PPE)

CICD-SEC-5: Insufficient PBAC (Pipeline-Based Access Controls)

CICD-SEC-6: Insufficient Credential Hygiene

CICD-SEC-7: Insecure System Configuration

CICD-SEC-8: Ungoverned Usage of 3rd Party Services

CICD-SEC-9: Improper Artifact Integrity Validation

CICD-SEC-10: Insufficient Logging and Visibility

Case Study

Mercedes-Benz onboard logic unit (OLU) source code leaks online

Mercedes-Benz onboard logic unit (OLU) source code leaks online

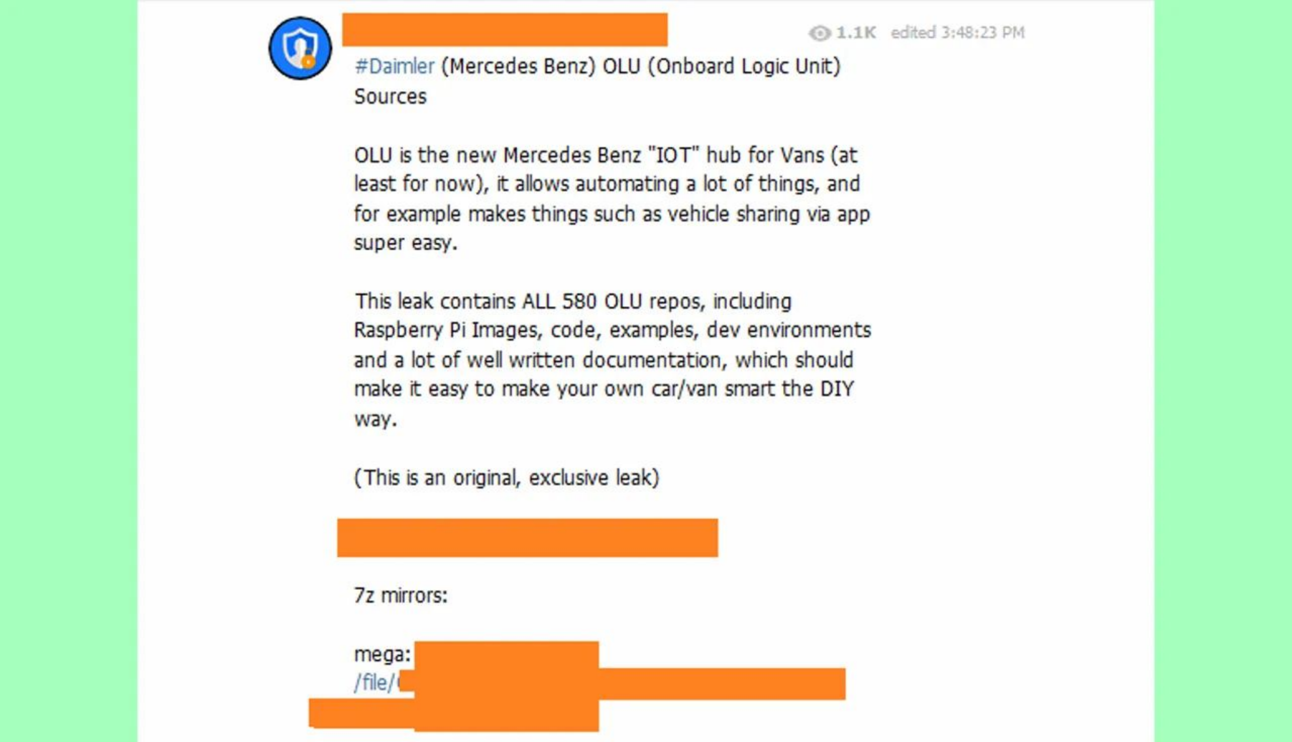
OLU is a component that sits between the car's hardware and software, and "connects vehicles to the cloud." which "simplifies technical access and the management of live vehicle data" and allows third-party developers to create apps that retrieve data from Mercedes vans.

Mercedes-Benz onboard logic unit (OLU) source code leaks online

Unsecured GitLab installation leaks OLU code

- Using something as simple as Google dorks.
- Kottmann says Daimler failed to implement an account confirmation process, which allowed to register an account on the company's official GitLab server using a non-existent Daimler corporate email.
- Threat intelligence firm Under the Breach which also reviewed the data, told ZDNet they discovered passwords and API tokens for Daimler's internal systems.

Mercedes-Benz onboard logic unit (OLU) source code leaks online



[redacted] 1.1K edited 3:48:23 PM

#Daimler (Mercedes Benz) OLU (Onboard Logic Unit) Sources

OLU is the new Mercedes Benz "IOT" hub for Vans (at least for now), it allows automating a lot of things, and for example makes things such as vehicle sharing via app super easy.

This leak contains ALL 580 OLU repos, including Raspberry Pi Images, code, examples, dev environments and a lot of well written documentation, which should make it easy to make your own car/van smart the DIY way.

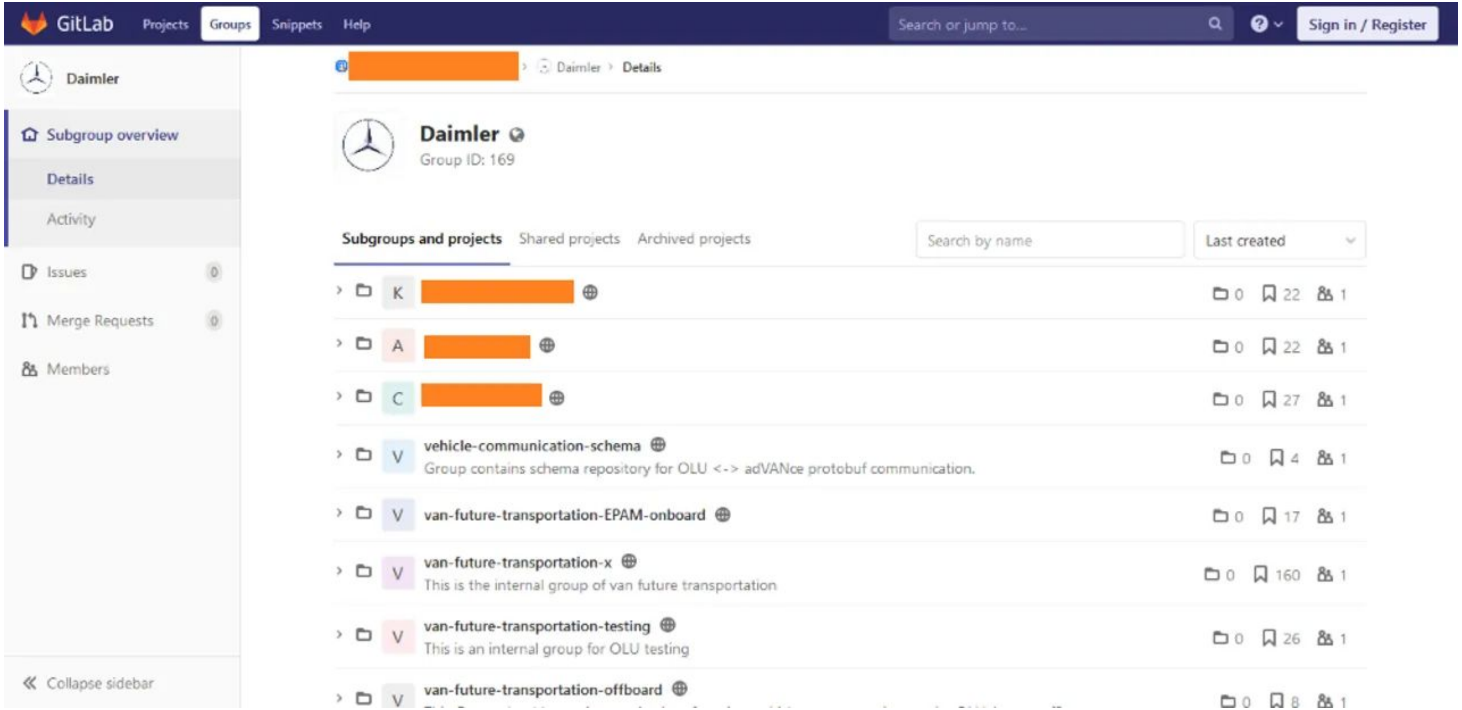
(This is an original, exclusive leak)

[redacted]

7z mirrors:

mega:
/file/[redacted]

Mercedes-Benz onboard logic unit (OLU) source code leaks online



GitLab Projects Groups Snippets Help Search or jump to... Sign in / Register

[redacted] › Daimler › Details

Daimler

- Subgroup overview
  - Details
  - Activity
- Issues 0
- Merge Requests 0
- Members

« Collapse sidebar

Daimler

Group ID: 169

Subgroups and projects | Shared projects | Archived projects — Search by name — Last created

K	[redacted]	0	22	1
A	[redacted]	0	22	1
C	[redacted]	0	27	1
V	vehicle-communication-schema Group contains schema repository for OLU <-> adVANce protobuf communication.	0	4	1
V	van-future-transportation-EPAM-onboard	0	17	1
V	van-future-transportation-x This is the internal group of van future transportation	0	160	1
V	van-future-transportation-testing This is an internal group for OLU testing	0	26	1
V	van-future-transportation-offboard	0	8	1

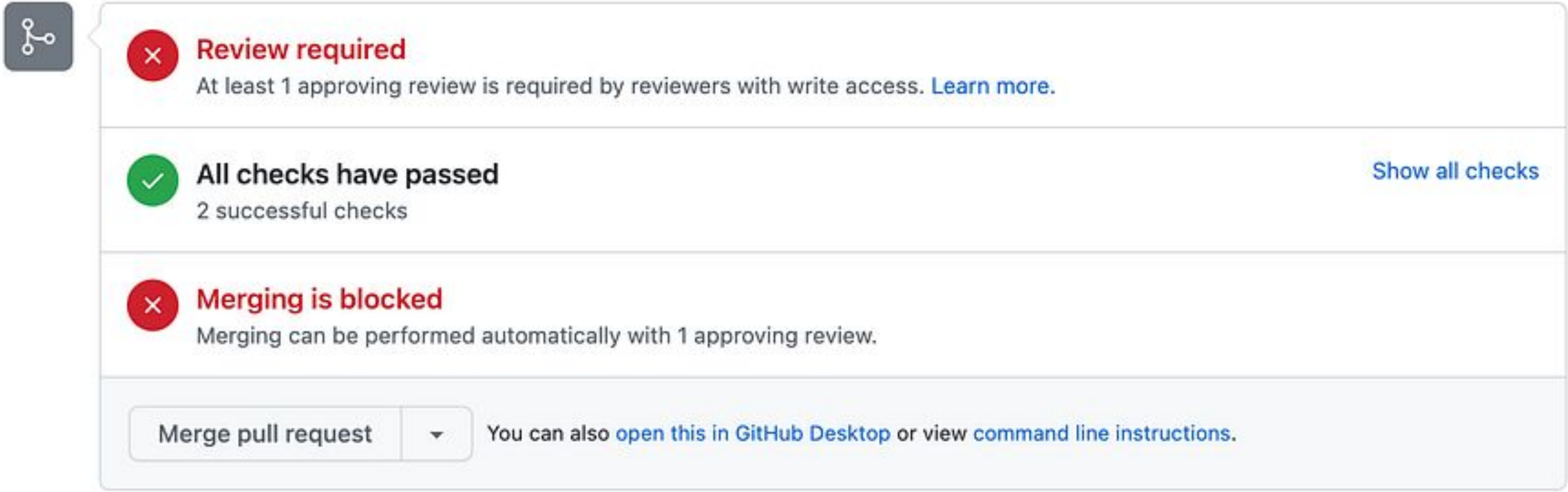
Case Study

Bypassing required reviews using GitHub Actions

Bypassing required reviews using GitHub Actions

Bypassing required reviews using GitHub Actions

Bypassing required reviews using GitHub Actions



✕ **Review required**

At least 1 approving review is required by reviewers with write access. Learn more.

✓ **All checks have passed**

2 successful checks

Show all checks

✕ **Merging is blocked**

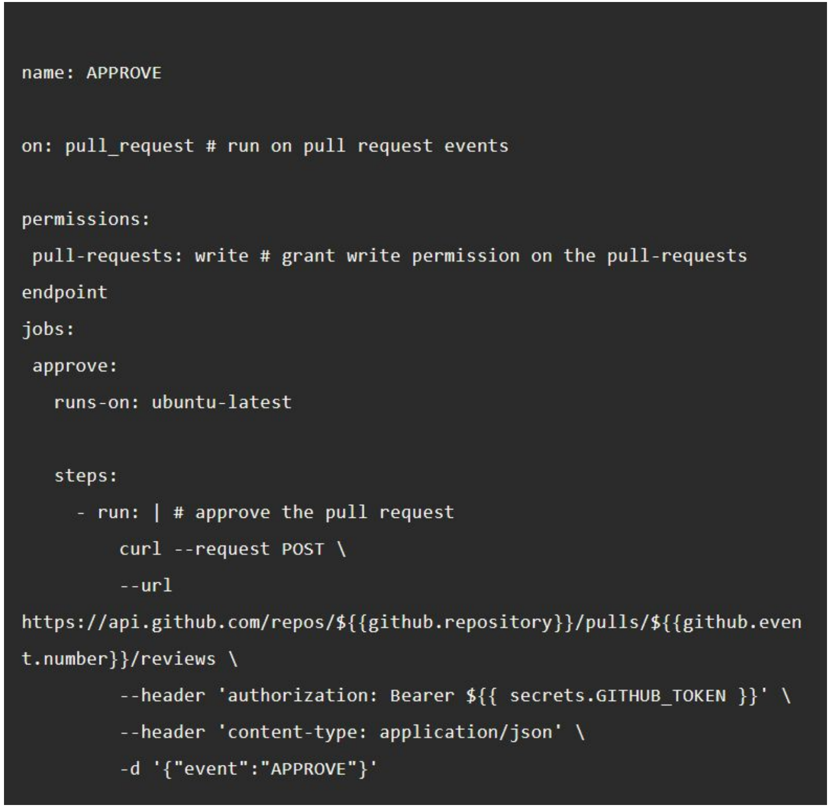
Merging can be performed automatically with 1 approving review.

You can also open this in GitHub Desktop or view command line instructions.

Bypassing required reviews using GitHub Actions

- GitHub Actions is installed by default on any GitHub organization, and on all of its repositories.
- Any user that can push code to the repo (Write permissions or higher), can create a workflow that runs when code is pushed.
- With each workflow run, GitHub creates a unique GitHub token (GITHUB_TOKEN) to use in the workflow to authenticate against the repo.
- Anyone with write access to a repository can modify the permissions granted to the GITHUB_TOKEN, adding or removing access as required, by editing the permissions key in the workflow file.

Bypassing required reviews using GitHub Actions



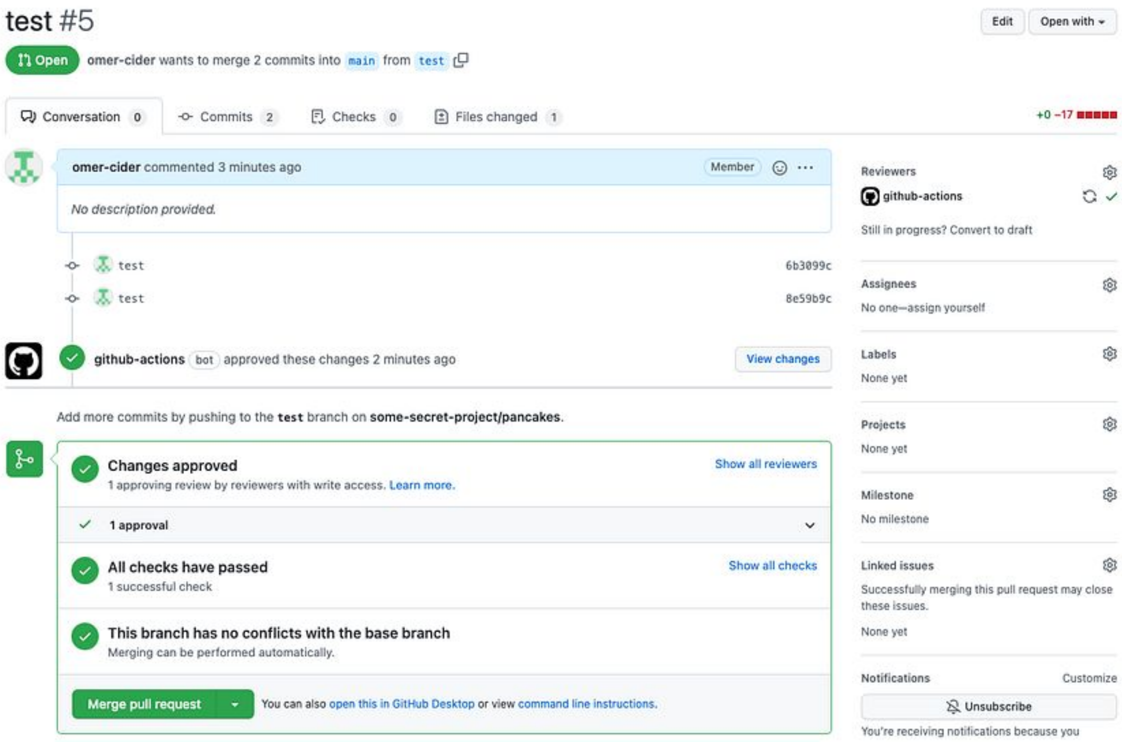
```
name: APPROVE

on: pull_request # run on pull request events

permissions:
 pull-requests: write # grant write permission on the pull-requests
endpoint
jobs:
 approve:
   runs-on: ubuntu-latest

   steps:
     - run: | # approve the pull request
         curl --request POST \
         --url
https://api.github.com/repos/${{github.repository}}/pulls/${{github.even
t.number}}/reviews \
         --header 'authorization: Bearer ${{ secrets.GITHUB_TOKEN }}' \
         --header 'content-type: application/json' \
         -d '{"event":"APPROVE"}'
```

Bypassing required reviews using GitHub Actions



test #5

Open · omer-cider wants to merge 2 commits into `main` from `test`

Conversation 0 · Commits 2 · Checks 0 · Files changed 1 · +0 −17

omer-cider commented 3 minutes ago · Member

No description provided.

- test 6b3099c
- test 8e59b9c

github-actions bot approved these changes 2 minutes ago · View changes

Add more commits by pushing to the **test** branch on **some-secret-project/pancakes**.

✓ **Changes approved** · Show all reviewers
1 approving review by reviewers with write access. Learn more.

✓ 1 approval

✓ **All checks have passed** · Show all checks
1 successful check

✓ **This branch has no conflicts with the base branch**
Merging can be performed automatically.

Merge pull request · You can also open this in GitHub Desktop or view command line instructions.

Reviewers: github-actions ✓

Still in progress? Convert to draft

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: No milestone

Linked issues: Successfully merging this pull request may close these issues. None yet

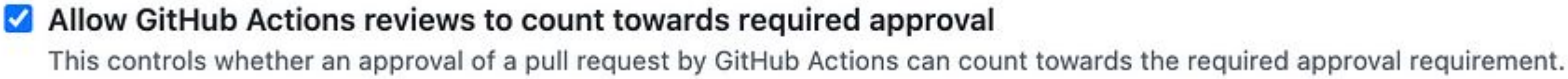
Notifications · Customize

Unsubscribe

You're receiving notifications because you

Bypassing required reviews using GitHub Actions

- Mitigation
  - Update: January 2022
  - Organization admins can now disallow GitHub Actions from approving pull requests.
  - Organization-wide setting default to allows Actions to approve pull requests in existing organizations, and disallows it in newly created orgs.
  - Disable under Organization Settings.



☑ **Allow GitHub Actions reviews to count towards required approval**

This controls whether an approval of a pull request by GitHub Actions can count towards the required approval requirement.

CICD-SEC-1

Insufficient Flow Control Mechanisms

CICD-SEC-1: Insufficient Flow Control Mechanisms

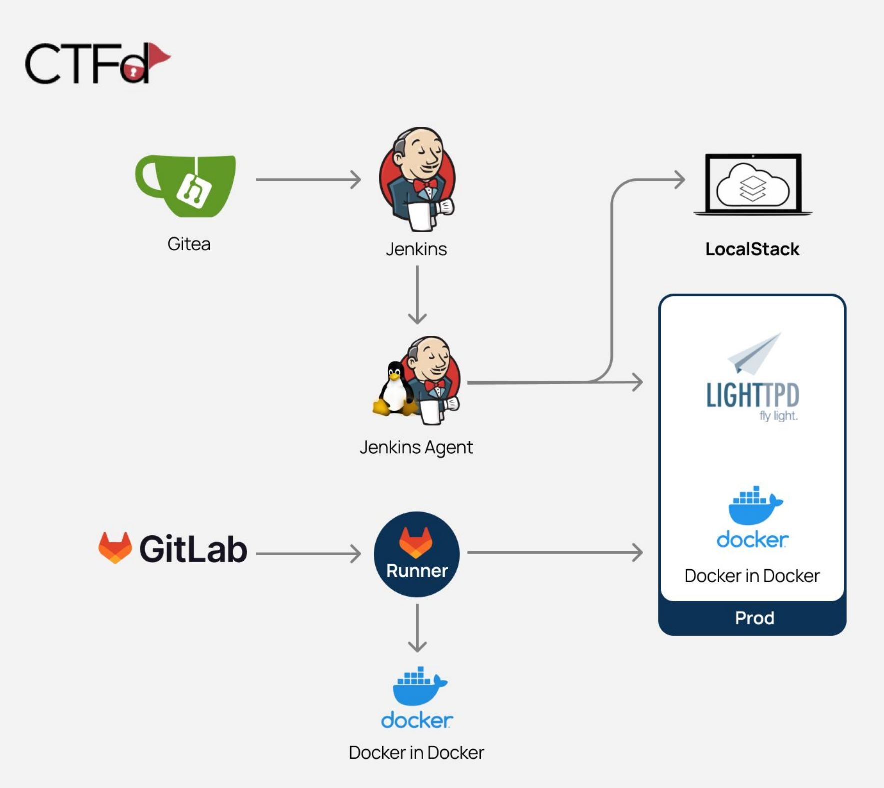
The ability of an attacker that has obtained permissions to a system within the CI/CD process (SCM, CI, Artifact repository, etc.) to single handedly push malicious code or artifacts down the pipeline, due to a lack in mechanisms that enforce additional approval or review.

CICD-SEC-1: Insufficient Flow Control Mechanisms

Example

CICD-SEC-1: Insufficient Flow Control Mechanisms

CI/CD Goat architecture



CICD-SEC-1: Insufficient Flow Control Mechanisms

Example

Dodo

CICD-SEC-1: Insufficient Flow Control Mechanisms

- Impact
  - The attacker can ship malicious artifact through the pipeline - potentially all the way to production - without any approval or review.
  - Push code which get automatically deployed through the pipeline.
  - Push code and then manually trigger a pipeline.
  - Directly push code to a library, which is used by a production system.
  - Abuse an auto-merge rule in the CI.
  - Abuse insufficient branch protection rules.

CICD-SEC-1: Insufficient Flow Control Mechanisms

- Recommendation
  - Establish pipeline flow control mechanisms to ensure that no single entity is able to ship sensitive code and artifacts through the pipeline without external verification or validation.
  - Configure branch protection rules and avoid exclusion of entities.
  - Limit the usage of auto-merge rules and ensure that they are applicable to the minimal amount of contexts.
  - Where applicable, prevent triggering production build and deployment pipelines without additional approval or review.
  - Prevent artifacts that have been uploaded by other accounts from flowing through the pipeline without secondary review and approval.

CICD-SEC-2

Inadequate Identity and Access Management

CICD-SEC-2: Inadequate Identity and Access Management

Stem from the difficulties in managing the vast amount of identities spread across the different systems in the engineering ecosystem and the existence of poorly managed identities - increases the potential and the extent of damage of accounts compromise.

CICD-SEC-2: Inadequate Identity and Access Management

Overly permissive identities: Ensuring each identity has been granted only the permissions required and only against the actual repositories it needs to access.

Stale identities: Identities that are not active or no longer require access but have not had account against all CI/CD systems deprovisioned.

Local identities: Local accounts create challenges in enforcing consistent security policies (e.g. password policy, lockout policy, MFA) as well as properly deprovisioning access across all systems.

CICD-SEC-2: Inadequate Identity and Access Management

External identities: Account with address from a domain not owned or managed by organization.

Self-registered identities: In systems where self-registration is allowed, it is often the case that a valid domain address is the only prerequisite for self-registration and access to CI/CD.

Shared identities: Identities shared between human users / applications / both humans and applications increase the footprint of their credentials as well as create challenges having to do with accountability in case of a potential investigation.

CICD-SEC-2: Inadequate Identity and Access Management

Example

Hearts

CICD-SEC-2: Inadequate Identity and Access Management

- Impact
  - Lack of strong identity and access management practice, leads to a state where compromising nearly any user account on any system.
  - Overly permissive accounts.
  - Could grant powerful capabilities to the environment.
  - Create investigation challenges.

CICD-SEC-2: Inadequate Identity and Access Management

- Recommendation
    - Conduct a continuous analysis and mapping of all identities across all systems within the engineering ecosystem.
    - Map the identity provider, level of permissions granted and level of permissions actually used.
    - Refrain from allowing users to self-register to systems, and grant permission on an as-needed basis.
    - Avoid creating local user accounts. Instead, create and manage identities using a centralized organization component (IdP).

CICD-SEC-2: Inadequate Identity and Access Management

- Recommendation
  - Avoid creating local user accounts. Instead, create and manage identities using a centralized organization component (IdP).
  - Prevent employees from using any address which belongs to a domain not owned and managed by the organization.
  - Avoid using shared accounts. Create dedicated accounts for each specific context, and grant the exact set of permissions required for the context in question.

CICD-SEC-3

Dependency Chain Abuse

SHIPS
シップス

CICD-SEC-3: Dependency Chain Abuse

The ability of an attacker to abuse flaws relating to how engineering workstations and build environments fetch code dependencies, results in a malicious package inadvertently being fetched and executed locally when pulled.

CICD-SEC-3: Dependency Chain Abuse

Dependency confusion: Publication of malicious packages in public repositories with the same name as internal package names.

Dependency hijacking: Obtaining control of the account of a package maintainer on the public repository, in order to upload a new, malicious version of a widely used package.

Typosquatting: Publication of malicious packages with similar names to those of popular packages in the hope that a developer will misspell a package name and unintentionally fetch the typosquatted package.

Brandjacking: Publication of malicious packages in a manner that is consistent with the naming convention or other characteristics of a specific brand’s package.

CICD-SEC-3: Dependency Chain Abuse

Example

Twiddledum

CICD-SEC-3: Dependency Chain Abuse

- Impact
    - Executing malicious code (RCE) on a host pulling the package, can be leveraged many ways such as credentials theft and lateral movement within the environment.

CICD-SEC-3: Dependency Chain Abuse

- Recommendation
    - Any client pulling code packages should not be allowed to fetch packages directly from the internet or untrusted sources.
    - Ensure all packages are pulled through an internal proxy rather than directly from the internet.
    - Enable checksum verification and signature verification for pulled packages.
    - Avoid configuring clients to pull the latest version of a package.

CICD-SEC-3: Dependency Chain Abuse

- Recommendation
  - Ensure that a separate nonsensitive context exists for scripts executing as part of a package installation.
  - Ensure that internal projects always contain configuration files of package managers (for example .npmrc in NPM) to override any insecure configuration.
  - Ensure that an appropriate level of focus is placed around detection, monitoring and mitigation.

CICD-SEC-4

Poisoned Pipeline Execution (PPE)

CICD-SEC-4: Poisoned Pipeline Execution (PPE)

The ability of an attacker with access to source control systems and manipulate the build process by injecting malicious code/commands into the build pipeline configuration, then run malicious code as part of the build process.

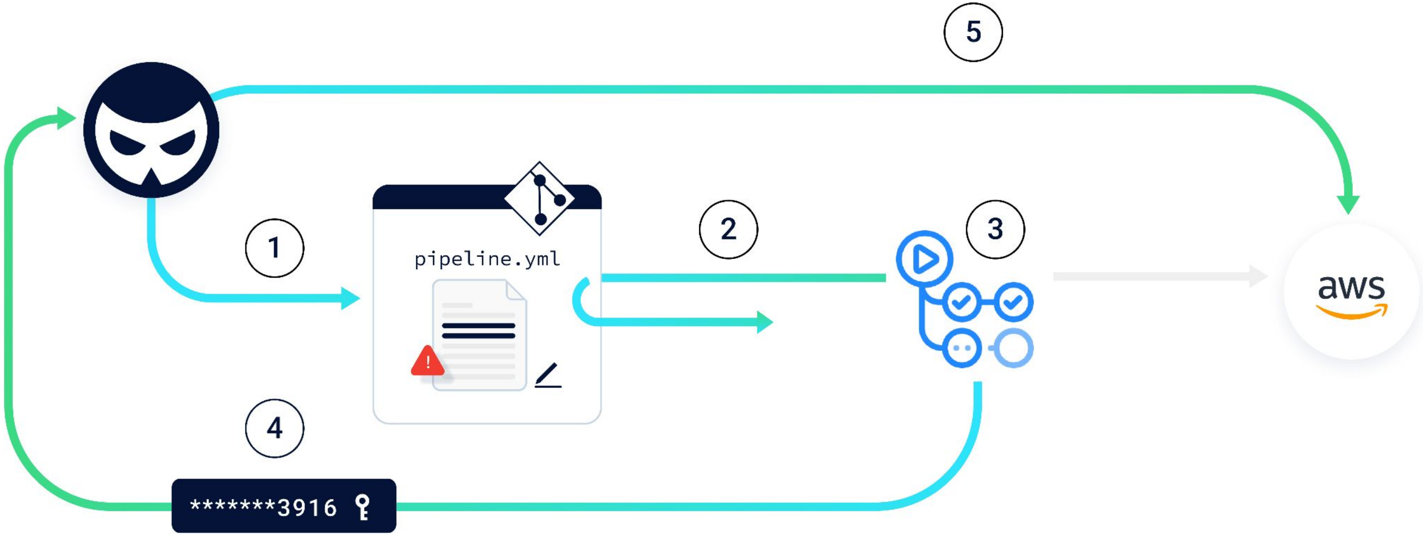
CICD-SEC-4: Poisoned Pipeline Execution (PPE)

Direct PPE (D-PPE): Attacker modifies the CI config file in a repository they have access to, either by pushing the change directly to an unprotected remote branch on the repo, or by submitting a PR with the change from a branch or a fork.

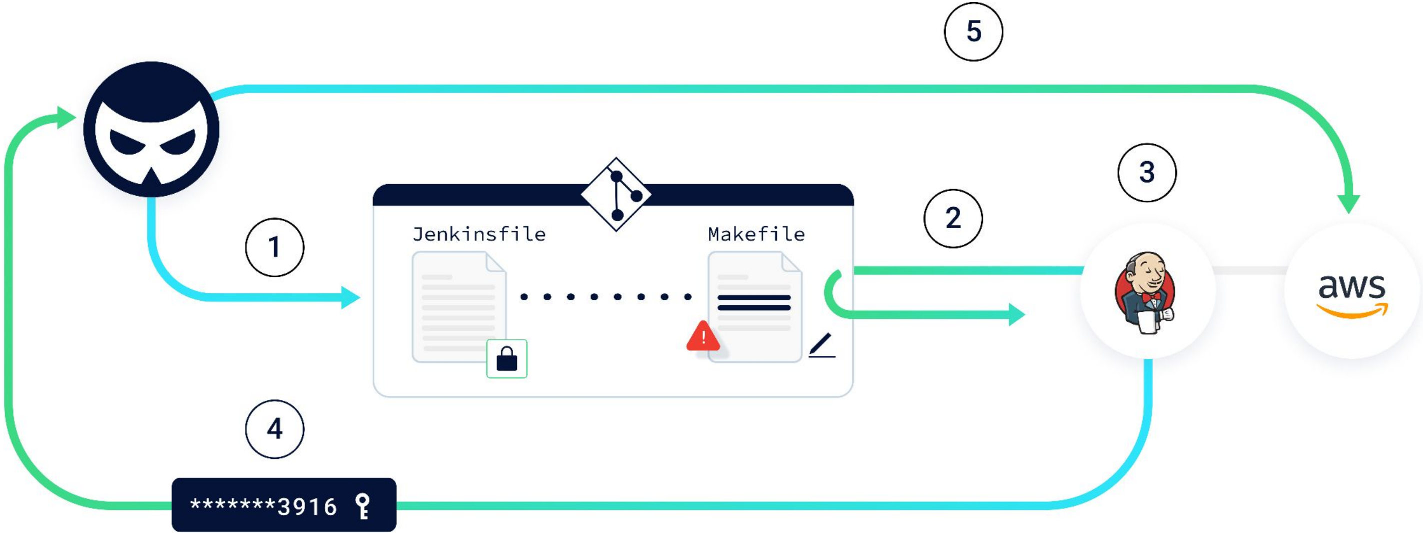
Indirect PPE (I-PPE): Attacker poison the pipeline by injecting malicious code into files referenced by the pipeline configuration file.

Public-PPE (3PE): Attacker has public access and can contribute to repository while CI pipeline runs unreviewed code suggested by anonymous users.

CICD-SEC-4: Direct Poisoned Pipeline Execution (D-PPE)



CICD-SEC-4: Indirect Poisoned Pipeline Execution (I-PPE)



CICD-SEC-4: Public Poisoned Pipeline Execution (3PE)

Example

Caterpillar

CICD-SEC-4: Poisoned Pipeline Execution (PPE)

- Impact
    - Provides the attacker with the same abilities and level of access as the build job.
    - Access to any secret available to the CI job
    - Access to external assets the job node has permissions to
    - Ability to ship code and artifacts further down the pipeline
    - Ability to access additional hosts and assets in the environment of the job node

CICD-SEC-4: Poisoned Pipeline Execution (PPE)

- Recommendation
  - Preventing and mitigating the PPE attack vector involves multiple measures spanning across both SCM and CI systems
  - Unreviewed code are executed on isolated nodes, not exposed to sensitive environments
  - Refrain from running pipelines originating from forks
  - Has a correlating branch protection rule
  - CI configuration file must be reviewed before the pipeline runs
  - Each pipeline should only have access to the credentials it needs to fulfill its purpose

CICD-SEC-5

Insufficient PBAC (Pipeline-Based Access Controls)

CICD-SEC-5: Insufficient PBAC (Pipeline-Based Access Controls)

The attacker’s ability to abuse the overly permission granted to the pipeline for moving laterally within or outside the CI/CD system.

CICD-SEC-5: Insufficient PBAC (Pipeline-Based Access Controls)

Example

Cheshire Cat

CICD-SEC-5: Insufficient PBAC (Pipeline-Based Access Controls)

- Impact
  - Provides the attacker with the access to secrets, underlying host and connect to any of the systems the pipeline in question has access to
  - Leads to exposure of confidential data
  - Lateral movement within the CI environment
  - Potentially accessing servers and systems outside the CI environment
  - Deployment of malicious artifacts down the pipeline

CICD-SEC-5: Insufficient PBAC (Pipeline-Based Access Controls)

- Recommendation
  - Do not use a shared node for pipelines with different levels of sensitivity / that require access to different resources.
  - Shared nodes should be used only for pipelines with identical levels of confidentiality.
  - Ensure that pipeline to have access to only the secrets it requires.
  - Revert the execution node to its pristine state after each pipeline execution.
  - Granted OS permissions on the execution node according to the principle of least privilege

CICD-SEC-5: Insufficient PBAC (Pipeline-Based Access Controls)

- Recommendation
    - Ensure the execution node is appropriately patched.
    - Ensure network segmentation in the environment which allow the execution node to access only the resources it requires within the network.
    - CI/CD pipeline jobs should have limited permissions on the controller node.
    - Where applicable, run pipeline jobs on a separate, dedicated node.

CICD-SEC-6

Insufficient Credential Hygiene

CICD-SEC-6: Insufficient Credential Hygiene

The attacker’s ability to obtain and use various secrets and tokens spread throughout the pipeline due to flaws having to do with access controls around the credentials, insecure secret management and overly permissive credentials.

CICD-SEC-6: Insufficient Credential Hygiene

Code containing credentials being pushed to one of the branches of an SCM repository: Credentials are exposed to anyone with read access to the repository, and even if deleted from the branch it was pushed into

Credentials used insecurely inside the build and deployment processes: Insecure used to access code repositories, read from and write to artifact repositories, and deploy resources and artifacts to production environments.

Credentials in container image layers: Credentials that were only required for building the image, still exist in one of the image layers.

CICD-SEC-6: Insufficient Credential Hygiene

Credentials printed to console output: Credentials exposed in clear-text in logs, available to anyone with access to the build results to view. These logs can potentially flow to log management systems, expanding their exposure surface.

Unrotated credentials: Failing to rotate credentials results in a constantly growing amount of people and artifacts that are in possession of valid credentials. (“If it isn’t broken, don’t fix it”)

CICD-SEC-6: Insufficient Credential Hygiene

Example

Duchess

CICD-SEC-6: Insufficient Credential Hygiene

- Impact
  - Credentials are the most sought-after object by adversaries, putting high-value resources of many organizations at the risk of compromise due the exposure of their credentials.
  - Deploying malicious code and artifacts through the pipeline.
  - Lateral movement through environment or out of CI/CD context.

CICD-SEC-6: Insufficient Credential Hygiene

- Recommendation
    - Establish procedures to continuously map credentials found across the different systems in the engineering ecosystem - from code to deployment.
    - Ensure each set of credentials follows the principle of least privilege.
    - Avoid sharing the same set of credentials across multiple contexts.
    - Prefer using temporary credentials over static credentials.
    - Periodically rotate all static credentials and detect stale credentials.
    - Configure usage of credentials to be limited to predetermined conditions (like scoping to a specific source IP or identity
    - Detect secrets pushed to and stored on code repositories.
    - revent secrets from being printed to console outputs of future builds.
    - Verify that secrets are removed from any type of artifact.

CICD-SEC-7

Insecure System Configuration

CICD-SEC-7: Insecure System Configuration

Stem from flaws in the security settings, configuration and hardening of the different systems across the pipeline (e.g. SCM, CI, Artifact repository)

CICD-SEC-7: Insecure System Configuration

- Potential hardening flaws:
  - Using an outdated version or lacking important security patches.
  - Overly permissive network access controls.
  - Using default set of configurations which is not secure and requires optimization.

CICD-SEC-7: Insecure System Configuration

Example

Mercedes-Benz onboard logic unit (OLU) source code leaks online

CICD-SEC-7: Insecure System Configuration

- Impact
  - A security flaw in one of the CI/CD systems may be leveraged by an adversary to obtain unauthorized access to the system or worse.
  - Compromise the system and access the underlying OS.
  - Abused by an attacker to manipulate legitimate CI/CD flows.
  - Obtain sensitive tokens and potentially access production environments.
  - Move laterally within the environment and outside the context of CI/CD systems.

CICD-SEC-7: Insecure System Configuration

- Recommendation
  - Maintain an inventory of systems and versions in use, including mapping of a designated owner for each system.
  - Continuously check for known vulnerabilities in these components. If a security patch is available, update the vulnerable component.
  - Consider removing/disabling the vulnerable component/system when security patch is unavailable.
  - Ensure network access to the systems is aligned with the principle of least access.
  - Establish a process to periodically review all system configurations.
  - Ensure permissions to the pipeline execution nodes are granted according to the principle of least privilege.

CICD-SEC-8

Ungoverned Usage of 3rd Party Services

CICD-SEC-8: Ungoverned Usage of 3rd Party Services

The extreme ease with which a 3rd party service can be granted access to resources in CI/CD systems, effectively expanding the attack surface of the organization.

CICD-SEC-8: Ungoverned Usage of 3rd Party Services

- SCM - GitHub SAAS - 3rd party applications can be connected through one or more of these 5 methods
  - GitHub Application
  - OAuth application
  - Provisioning of an access token provided to the 3rd party application
  - Provisioning of an SSH key provided to the 3rd party application.
  - Configuring webhook events to be sent to the 3rd party.

CICD-SEC-8: Ungoverned Usage of 3rd Party Services

Example

Dormouse

CICD-SEC-8: Ungoverned Usage of 3rd Party Services

- Impact
  - Compromise of a single 3rd party can be leveraged to cause damage far outside the scope of the system the 3rd party is connected to.
  - 3rd party with write permissions on a repository, can be leveraged by an adversary to push and run malicious code on the build system.
  - Organizations are only as secure as the 3rd parties they implement.
  - Lack of governance, visibility, RBAC, least privilege around 3rd party implementations.

CICD-SEC-8: Ungoverned Usage of 3rd Party Services

- Recommendation
  - Governance controls around 3rd party services should be implemented within every stage of the 3rd party usage lifecycle.
  - Approval: Establish vetting procedures to ensure granted access to resources are approved and the level of permission is aligned with least privilege.
  - Integration - Introduce controls and procedures to maintain continuous visibility over all 3rd parties integrated to CI/CD systems.
  - Visibility over ongoing usage - Ensure each 3rd party is limited and scoped to the specific resources it requires access to.
  - Deprovisioning - Periodically review all 3rd parties integrated and remove those no longer in use.

CICD-SEC-9

Improper Artifact Integrity Validation

SHIPS
シ ッ プ ス

CICD-SEC-9: Improper Artifact Integrity Validation

Due to insufficient mechanisms for ensuring the validation of code and artifacts, allows an attacker with access to one of the systems in the CI/CD process to push malicious code or artifacts down the pipeline.

CICD-SEC-9: Improper Artifact Integrity Validation

Example

Dormouse

CICD-SEC-9: Improper Artifact Integrity Validation

- Impact
    - Adversary with a foothold within the software delivery process to ship a malicious artifact through the pipeline.
    - Execute malicious code on systems within the CI/CD process or worse - in production.

CICD-SEC-9: Improper Artifact Integrity Validation

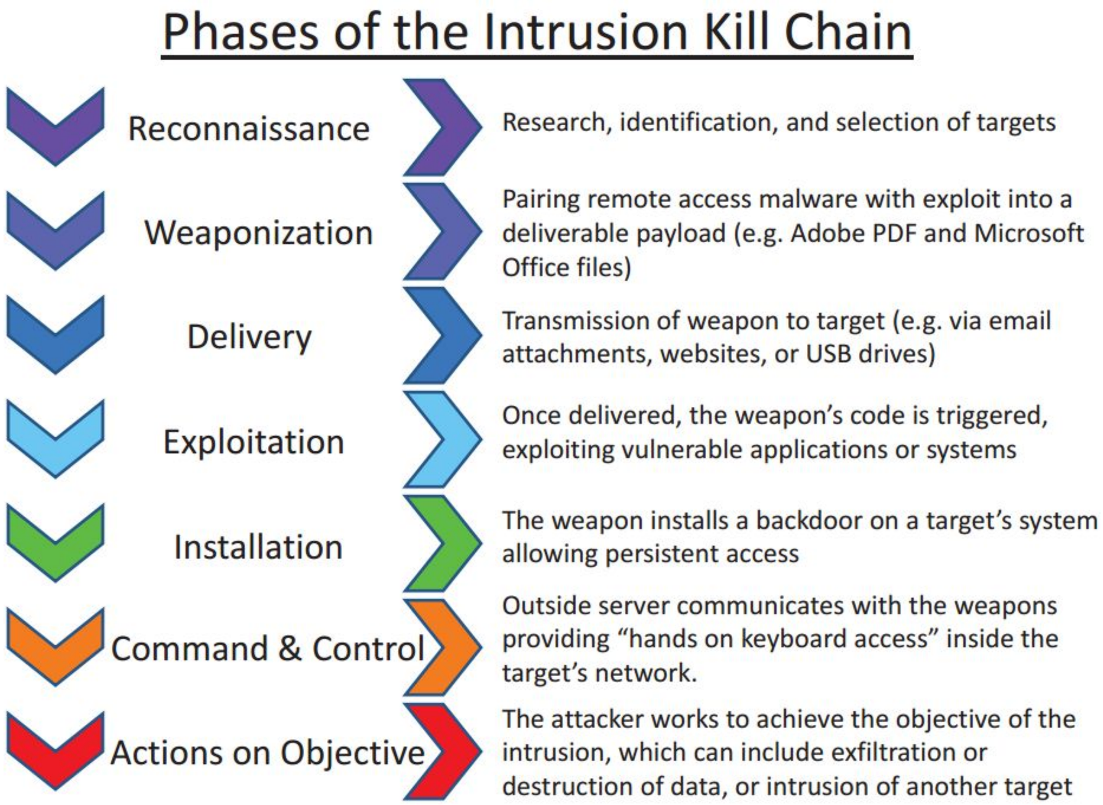
- Recommendation
  - Code signing - SCM solutions provide the ability to sign commits using a unique key for each contributor.
  - Artifact verification software - Prevent unverified software from being delivered down the pipeline.
  - Configuration drift detection - Measures aimed at detecting configuration drifts (e.g. resources in cloud environments which aren’t managed using a signed IAC template), potentially indicative of resources that were deployed by an untrusted source or process.

CICD-SEC-10

Insufficient Logging and Visibility

CICD-SEC-10: Insufficient Logging and Visibility

Allow an adversary to carry out malicious activities within the CI/CD environment without being detected during any phase of the attack kill chain, including identifying the attacker's TTPs (Techniques, Tactics and Procedures) as part of any post-incident investigation.



Phases of the Intrusion Kill Chain

Reconnaissance

Research, identification, and selection of targets

Weaponization

Pairing remote access malware with exploit into a deliverable payload (e.g. Adobe PDF and Microsoft Office files)

Delivery

Transmission of weapon to target (e.g. via email attachments, websites, or USB drives)

Exploitation

Once delivered, the weapon's code is triggered, exploiting vulnerable applications or systems

Installation

The weapon installs a backdoor on a target's system allowing persistent access

Command & Control

Outside server communicates with the weapons providing "hands on keyboard access" inside the target's network.

Actions on Objective

The attacker works to achieve the objective of the intrusion, which can include exfiltration or destruction of data, or intrusion of another target

CICD-SEC-10: Insufficient Logging and Visibility

- Impact
  - The existence of all relevant data sources in a centralized location may be the difference between a successful and devastating outcome in an incident response scenario.
  - Minimal investigative capabilities.
  - Failing to detect a breach.
  - Facing great difficulties in mitigation/remediation.

CICD-SEC-10: Insufficient Logging and Visibility

- Recommendation
  - Mapping the environment - Strong visibility capabilities cannot be achieved without an intimate level of familiarity with all the different systems involved in potential threats.
  - Identifying and enabling the appropriate log sources - Ensuring that all relevant logs are enabled
  - Shipping logs to a centralized location - Support aggregation and correlation of logs between different systems for detection and investigation.
  - Creating alerts to detect anomalies and potential malicious activity - both in each system on its own and anomalies in the code shipping process

THAT’S IT

SHIPS
シ ッ プ ス

The CI/CD Goat just got wilder!

Gryphon

"That's the reason they're called lessons," the Gryphon remarked: *"because they lessen from day to day."*

How long will it take you to crack the Gryphon challenge?

This time, you've compromised GitLab! Use your user account to capture flag11. Do you have what it takes?

💡Good to know: you can click the "Explore projects" button to view public projects as well.

Reference

- What is CI/CD? (redhat.com)
- OWASP Top 10 CI/CD Security Risks | OWASP Foundation
- CI/CD Goat – A deliberately vulnerable CI/CD environment - Cider Security Site
- Malicious code analysis: Abusing SAST | Cider Security
- Shared/Defenders think in lists. Attackers think in graphs. As long as this is true, attackers win
- Bypassing required reviews using GitHub | Cider Security
- Bash Uploader Security Update - Codecov

Q&A

SHIPS
シップス

"Seems SAVING the game really is impossible.

...

But...

Maybe, with what little power you have...

You can SAVE something else."

- Undertale -

21st July 2023

FOR ALL YOUR ATTENTION

THANK YOU

Top 10 CI/CD Security Risks

Natworpong Loyswai

SHIPS
シップス