RCE-as-a-Service:
Lessons Learned from 5 Years of Real-World CI/CD Pipeline Compromise

Iain Smart & Viktor Gazdag

WhoAreWe

Iain Smart (@smarticu5)

NCC Group Containerisation Practice Lead

6 years Container & Cloud Pentesting

Viktor Gazdag (@wucpi)

Jenkins Security MVP, Cloud Research Group Lead

7 years Pentest - Internal & External Team

Terminology

CI – Continuous Integration

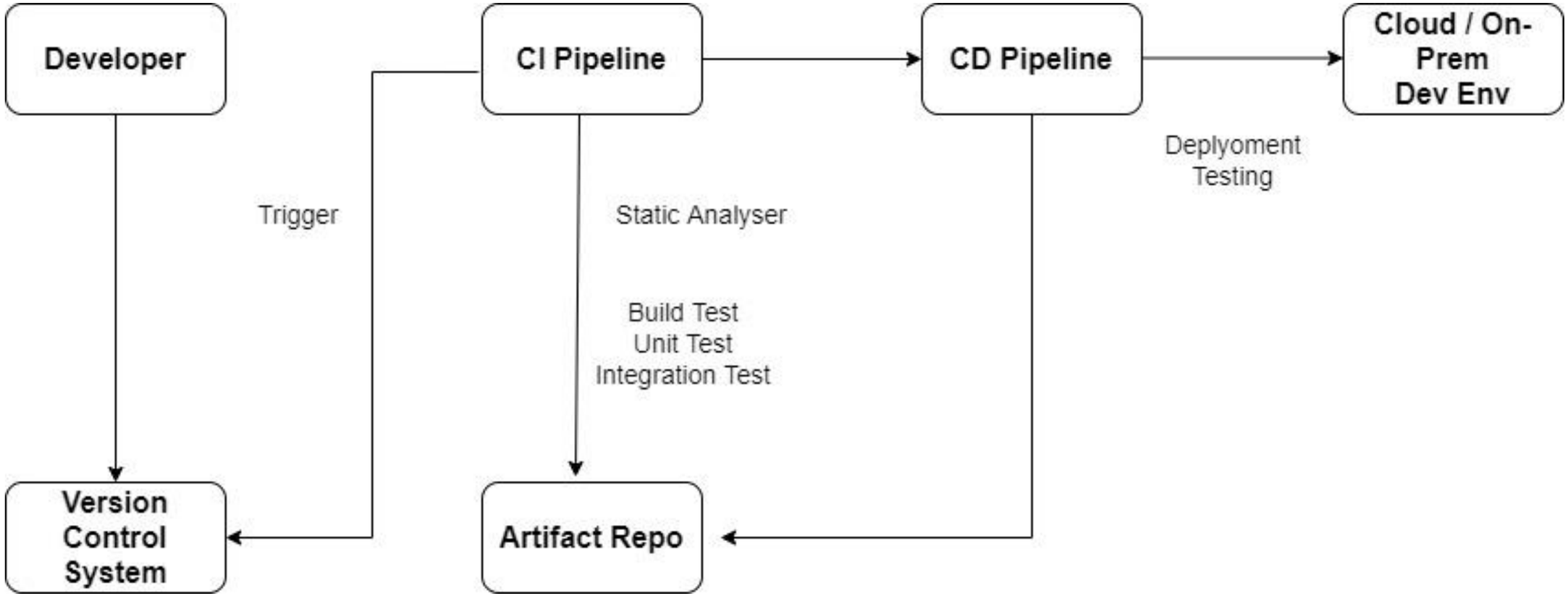
CD – Continuous Development/Deployment

Pipeline – A task or series of tasks performed automatically

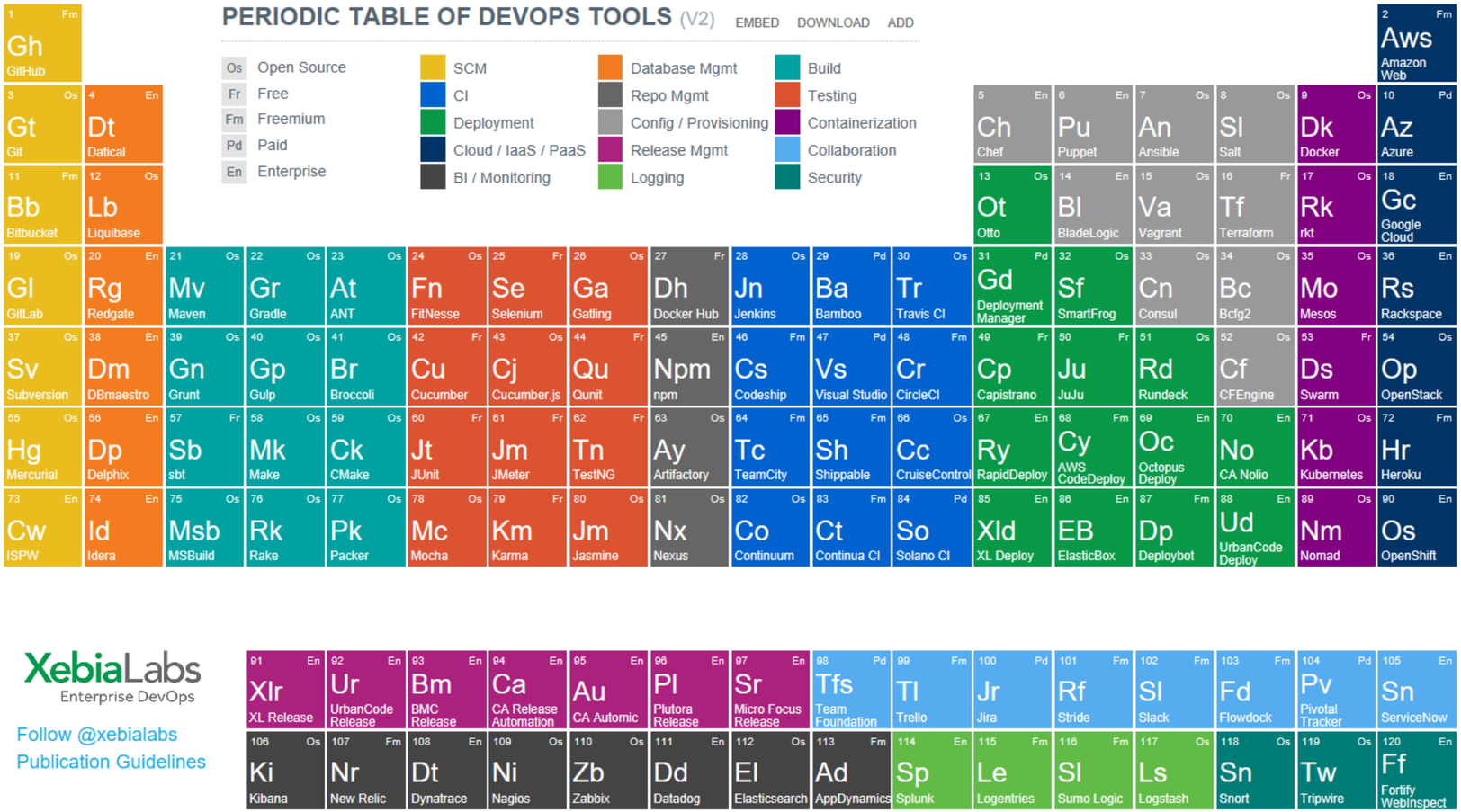
Secrets – Something sensitive you wouldn't want to be made public (e.g. a password)

RCE – Remote Code Execution

CI/CD Overview



CI/CD Overview



Attacker's View

Code goes in, apps come out

Pipelines have network access to a range of environments

Pipelines have credentials to pull dependencies and push artifacts

Free compute resources

Scenarios

Developers Access

Git Access

Granted developer access to an internal Bitbucket repo

Git Access > Load Dependencies

Apache Maven Project

Could modify project dependencies

Hosted dependency on an attacker-controlled URL

Git Access > Load Dependencies > Jenkins Runner Shell

Meterpreter shell from Jenkins runner

Limited environment means easy recon

Found keys

Git Access > Load Dependencies > Jenkins Runner Shell > Pivot

Network Recon – SSH

File system access – SSH key

Put them together? It just makes sense.

Git Access > Load Dependencies > Jenkins Runner Shell > Pivot > Root

SSH access to Jenkins control nodes

Secrets for all projects

Kubeconfig file

Lessons Learned

Build/Credential hygiene

Network filtering

Dependency validation and sources

Confusing Wording

GitHub Authorization

GitHub Authorization Plugin

"Grant READ Permissions to All Authenticated Users"

- Not “All Authenticated Users in this Organisation”

GitHub Authorization > Jenkins Access with Gmail Account

Register an arbitrary account in GitHub to log in

Log in to Jenkins

Access to build jobs and history

No monitoring and alerting

Lessons Learned

Read the description and test

Least privilege principal

Separation of duties

Monitoring and alerting

Build Output

Jenkins Login

Authenticated Users had Overall/Administrator role in Jenkins

Infra as Code pipeline with terraform cli within the steps

No monitoring and alerting

Jenkins Login > Job Logs

Terraform cli output in build output

Output did not mask any secrets

Showed all users AWS API keys

Lessons Learned

Least privilege principal

Separation of duties

Dedicated plugin with output masking

Monitoring and alerting

FSAS Engagement

Entry

Red team compromised a developer account

Git access to a range of repositories

Entry > Pipeline Access

Pipeline modifications

Pipeline prevented displaying of env variables

printenv | base64

Entry > Pipeline Access > Credentials

Service account with Domain Admin access

Lessons Learned

Restrict secrets to specific branches

Don’t run anything as Domain Admin

Web Application LFI

Web App

Internal infrastructure assessment

Identified web application SSRF/LFI vulnerabilities

Application was deployed for testing mid-pipeline

Web App > Kubernetes

Read service account token through LFI

Use to communicate with Kubernetes API Server

Permissions to edit Configmaps (AWS EKS authentication)

Gained cluster admin on the build cluster

Web App > Kubernetes > Container Registry

Stole AWS IAM credentials to deploy to ECR

Web App > Kubernetes > Container Registry > Kubernetes

Production workloads used pull-based CI

Overwrote images in ECR, new images were automatically pulled

Lessons Learned

Network isolation

RBAC – least privilege

From Cloud Web App to On-Prem Server

Web App

Started of WordPress with some custom pages

Application had directory listings enabled S3 bucket linked as sitemap

File responsible to push code to GitHub

Web App > Git

File had hardcoded credentials in S3 allowed access to VCS

Developer access to multiple repositories with read and write

No alert on signing in with the account, but had logs

Web App > Git > Jenkins

Same credentials allowed access to Jenkins, as an administrator

14 cluster with 200 build servers or agents

Web App > Git > Jenkins > Lateral Movement

Cluster admin on 14 Jenkins Master and 200 Jenkins build servers

Dump credentials from Jenkins: 200+ API Tokens, SSH Keys

Lessons Learned

Credential hygiene

Secret management

Least privilege principal

Lack of monitoring and alerting

Branched-Based Secrets

Developer Access

Developers had no direct access to production cloud environments

Only main branch deployed to prod

Main branch protected, and required merge approval

Developer Access > Circle CI

Pipelines configured using CircleCI YAML files

Configuration files in the same repo as application code

Secrets specified per branch

Developer Access > Circle CI

```
- name: Do Dev things
  image: registry.customer.com/terraform:v0.12
  environment:
    DEV_AWS_ACCESS_KEY_ID:
      from_secret: DEV_AWS_ACCESS_KEY_ID
    DEV_AWS_SECRET_ACCESS_KEY:
      from_secret: DEV_AWS_SECRET_ACCESS_KEY
  commands:
  - terraform apply
  when:
    branch:
    - feature/dev*
```

Developer Access > Circle CI

```
- name: Do Prod things
  image: registry.customer.com/terraform:v0.12
  environment:
    PROD_AWS_ACCESS_KEY_ID:
      from_secret: PROD_AWS_ACCESS_KEY_ID
    PROD_AWS_SECRET_ACCESS_KEY:
      from_secret: PROD_AWS_SECRET_ACCESS_KEY
  commands:
  - terraform apply
  when:
    branch:
    - main
```

Developer Access > Circle CI > Privilege Escalation

Developers could modify pipeline configuration file in non-Main branches

```
- name: Do Hacky things
  image: registry.customer.com/terraform:v0.12
  environment:
    PROD_AWS_ACCESS_KEY_ID:
      from_secret: PROD_AWS_ACCESS_KEY_ID
    PROD_AWS_SECRET_ACCESS_KEY:
      from_secret: PROD_AWS_SECRET_ACCESS_KEY
  commands:
  - printenv
  when:
    branch:
    - *
```

Lessons Learned

Validate secret management/protection

Assume every developer is malicious (or trust them)

Log and audit credential use

Least privilege RBAC

Privilege Escalation

SSO Credentials

Login with SSO Credentials

Testing 3 separated roles: read, build, admin

Locked down and documented roles

SSO Credentials > Privilege Escalation

Build user with Build/Replay permission

Code execution with Groovy

SSO Credentials > Privilege Escalation > Credentials Dump

Credentials dump

Lessons Learned

Least privilege principal

Common Themes

Service hardening

Network segmentation

Monitoring & alerting

Patch management

RBAC (is still difficult)

Preventions

Threat modelling

Network segmentation

Patch management

Monitoring & alerting

Secrets management

RBAC – Least privilege

Conclusion

CI/CD is beneficial, but also complex

None of these problems are new

Environments are rarely threat modelled

Little focus on the pipeline and infra

Questions?

@smarticu5 – iain.smart@nccgroup.com

@wucpi – viktor.gazdag@nccgroup.com