AUTOMATING SECURITY COMPLIANCE AT SCALE

11.18.2019 - Kubecon

Ravi Devineni
Michael Pereira

Northwestern Mutual®

NORTHWESTERN MUTUAL

- Financial Services Company - HQ - Milwaukee, WI
- Locations in NYC, Minneapolis, Phoenix
- Fortune # 104 – 2018
- America's best Employers # 367 – 2019
- Best employers for Women # 131 – 2019
- Best employers for Diversity # 33 – 2019
- Veteran Friendly Workplace Award (USO Wisconsin, 2018)
- Best Place to Work for LGBT Equality (Perfect Score, Human Rights Campaign Corporate Equality Index, 2015 – 2019)
- 50 Best Companies for Diversity (Black Enterprise Magazine, 2018)
- Top Companies for Women Technologists (AnitaB.org, 2017)

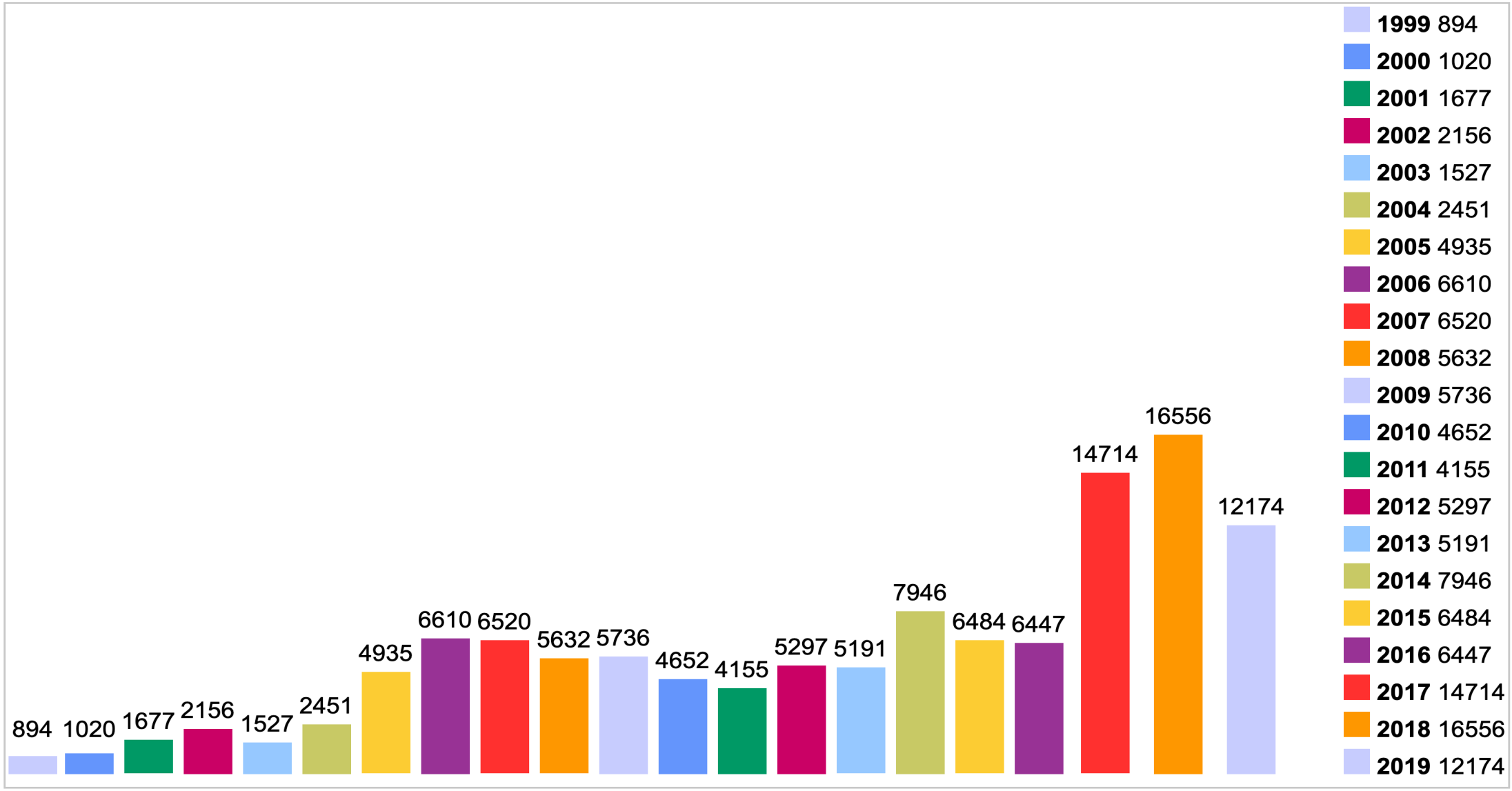
AGENDA

- CICD PIPELINES AND SECURITY
- SECURITY OF THE PIPELINE
- SECURITY IN THE PIPELINE
- Q&A

CICD PIPELINES

WHY SECURITY AND COMPLIANCE?

Vulnerabilities By Year



WHY SECURITY AND COMPLIANCE?

shhgit

shhgit live! v0.3

69 matches | 0 filters

- High entropy string 27
- Username and password in U 10
- Django configuration file 6
- Google Cloud API Key 6
- Log file 6
- PHP configuration file 4
- NPM configuration file 3
- Environment configuration file 3
- Google OAuth Key 3
- Potential Ruby On Rails datab. 1
- Carrierwave configuration file 1
- SQLite3 database file 1
- Shell profile configuration file 1

☑ Interesting file extensions ☑ High entropy strings ☐ Notify on match

@darkp0rt | blog | Connected

Read the corresponding blog post that inspired this tool.

	Found	Signature Name	Matches	File	★
	10:52:00 AM	Log file	–	/Logs/Packages-Update.log	1
	10:51:42 AM	Username and password in URI	mongodb://dbUser:dbPassword1@ds249623.mlab.com:49623/getir-case-study';	/test.js	0
	10:51:40 AM	Google Cloud API Key	AIzaSyCNnfqtppF_25UEmRc_ezyHX4IdyauoR0c	/Lato/src/main/java/com/mycompany/lato/model/Treasurer.java	0
	10:51:39 AM	Google Cloud API Key	AIzaSyCNnfqtppF_25UEmRc_ezyHX4IdyauoR0c	/Lato/src/main/java/com/mycompany/lato/control/AuthController.java	0
	10:51:37 AM	Username and password in URI	mongodb://dbUser:dbPassword1@ds249623.mlab.com:49623/getir-case-study';	/app.js	0
	10:51:32 AM	Log file	–	/log.log	0
	10:50:41 AM	Google Cloud API Key	AIzaSyAoEVTYuEF64ssgJHlG4NXycNEgjpt0vF4	/yuka_flutter/ios/Runner/GoogleService-Info.plist	-1
	10:50:41 AM	Username and password in URI	http://proxy_user:proxy_pass@example.com:8080';	/web/sites/default/settings.php	-1

Messages

CI/CD SECURITY – WHAT AND WHY?

Find security issues as early as possible

- Late detection increases cost, especially for security.
- Average Security Breach cost: $3.92 M (https://www.ibm.com/security/data-breach)

No company is exempt. - Equifax, Yahoo, Verizon, Target etc.

SECURITY BREACHES

Equifax Breach

- 150 Million people’s personal data exposed
- $1.4 Billion
- Open source; Apache Struts CVE-2017-5638

Verizon Breach

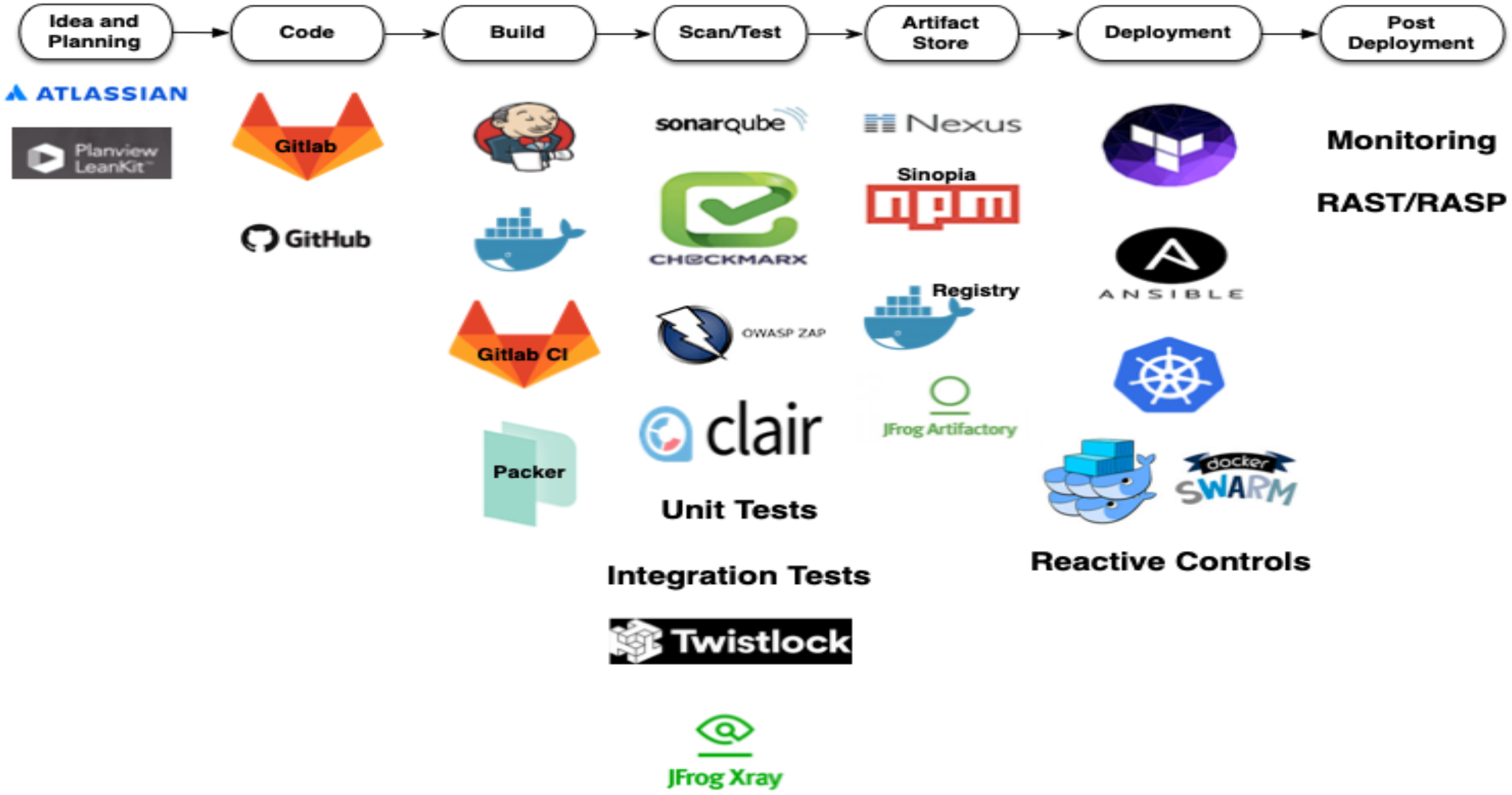
- Public S3 bucket
- 14 Million Customer records exposed

CONCEPTS

Security OF the pipeline

Security IN the pipeline

CICD PIPELINE AND TOOLS



SECURITY OF THE PIPELINE

- Secure all the tools used in the CICD Pipeline
- Principles
  - *Principle of Least privilege*
  - *Minimize attack surface and hence impact radius*
  - *Ensure* all communications are encrypted
  - Underlying OS is secure

SECURITY OF THE PIPELINE - EXAMPLES

Ensure repository settings are compliant

- Merge Requests Approvals configured
- Default branch protected

Authentication and Authorization

- All projects have the right amount of access and no more
- SSO is enabled and no open access

Security of the Pipeline Tools

- S3 buckets private/encrypted
- Encryption at rest and in-transit

CICD AUDITOR

Security of the Pipeline

as of Nov 18, 2019

Overall Audit Status	Failed Check Count
Fail	3

Individual Audit Checks

Check	Status
Check Public Projects	Pass
Check Users Email	Fail
Docker Registry: Data Encyrpted	Fail
Gitlab: Data Encrypted at Rest	Pass
Gitlab: Runner AMI has no vulnerabilities	Pass
Gitlab: S3 Buckets Encrypted	Pass
Gitlab: S3 Buckets Private	Pass
Jenkins: Image has no vulnerabilities	Fail
Jenkins: SSH access to the agents disabled	Pass
Nexus: Data Encyrpted	Pass
SSO Enabled in Gitlab	Pass
Twistlock: Image has no vulnerabilities	Pass

SECURITY IN THE PIPELINE

Concepts of Security Scanning

How to include it in the pipeline

Continuous Security of Production Environments

SECURITY SCANNING

Static security scanning

- OS vulnerabilities (containers)
- Dependencies vulnerabilities
- Code vulnerabilities
- Code quality

Dynamic security scanning

- Web app vulnerabilities

OS VULNERABILITIES

```
stage ('Container Scan') {
    runContainerScan imageName: 'myregistry:5000/mycontainer:latest'
}
```

OS VULNERABILITIES

```yaml
Clair:
  stage: Clair
  image: $CI_REGISTRY/lvcd/gitlabci-image-scanner:latest
  script:
    - scan-image
```

DEPENDENCY VULNERABILITIES

```yaml
Dependencies Scanning:
  stage: Security
  image: $CI_REGISTRY/lvcd/gitlabci-nexusiq-cli:latest
  script:
    - check-deps
  artifacts:
    paths:
    - $CI_PROJECT_DIR/nexusiq_results.json
    - $CI_PROJECT_DIR/nexusiq_results.pdf
    expire_in: 1 month
```

CODE VULNERABILITIES

```
stage ('Checkmarx Scan') {
    checkmarxScan()
}
```

CODE VULNERABILITIES

```yaml
Checkmarx:
  stage: Checkmarx
  image: $CI_REGISTRY/lvcd/gitlabci-checkmarx-cli:latest
  script:
    - check-code
  artifacts:
    paths:
    - $CI_PROJECT_DIR/results.xml
    - $CI_PROJECT_DIR/results.pdf
    expire_in: 1 month
  when: always
```

CODE QUALITY

```
stage ('Sonarqube Analysis') {
    runSQAnalysis()
}
```

CODE QUALITY

```yaml
SonarQube:
  stage: SonarQube
  image: $CI_REGISTRY/lvcd/sonar-scanner:latest
  script:
    - check-quality
  allow_failure: true
```

PIPELINE OVERLOAD

STATIC ANALYSIS - JENKINS

```
stage ('Security scans') {
    staticSecurityScan()
}
```

STATIC ANALYSIS - GITLAB

```yaml
include:
  - 'https://gitlab.company.com/gitlab/gitlabci-template/raw/master/post-build/.static-security-template.yml'
```

PRODUCTION IMAGES

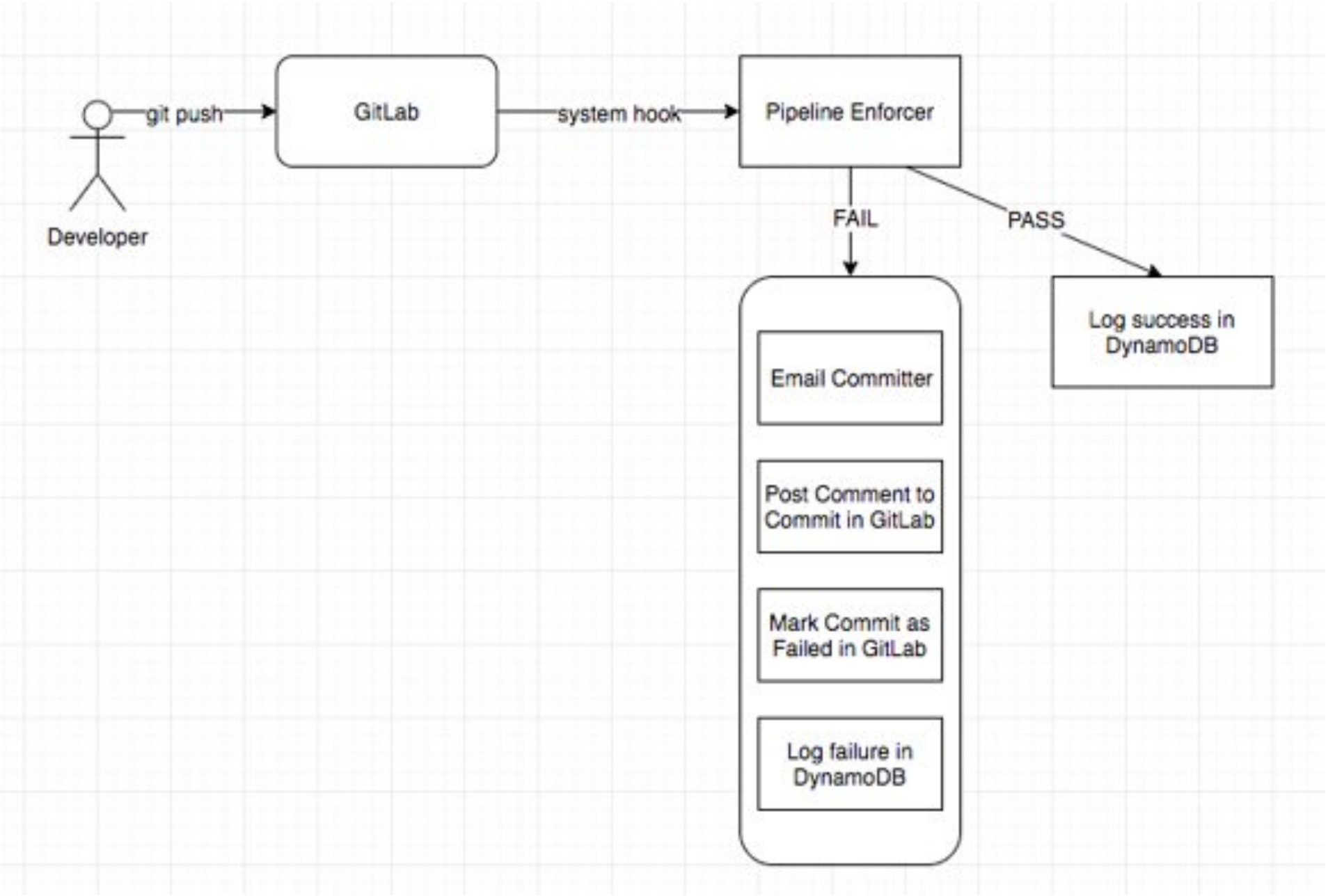
- A weekly process that runs a scan on all the production images.
- The scan job obtains a list of all the images and sequentially runs a container and dependencies scan on these images. Scheduled to run on every Sunday at 4AM UTC
- All the images with High vulnerabilities are then analyzed and the security team is notified

ENFORCING SECURITY IN THE PIPELINE

- Pipeline Enforcer
- Secrets Detector

PIPELINE ENFORCER

Checks build pipelines for required security and linting scans.


Developer
git push
GitLab
system hook
Pipeline Enforcer
FAIL
PASS
Email Committer
Post Comment to Commit in GitLab
Mark Commit as Failed in GitLab
Log failure in DynamoDB
Log success in DynamoDB

SECRET SAUCE

```
def string_is_in_file_contents(file_contents: str, patterns: List[str],
comment_token: str) -> bool:

    if not(file_contents):
        return False
    lines = file_contents.split('\n')
    for line in lines:
        line = remove_comment_from_line(
            line=line, comment_token=comment_token)
        for pattern in patterns:
            if pattern in line:
                return True
    return False
```

ENFORCEMENT

Automated email to the code committer with link to documentation

Pipeline Enforcer

This message was triggered by a commit you made to https:// .com/cicd/transform-gitlab-url-for-jenkins/tree/master. The following issues were found with the build pipeline defined in .gitlab-ci.yml on this branch.

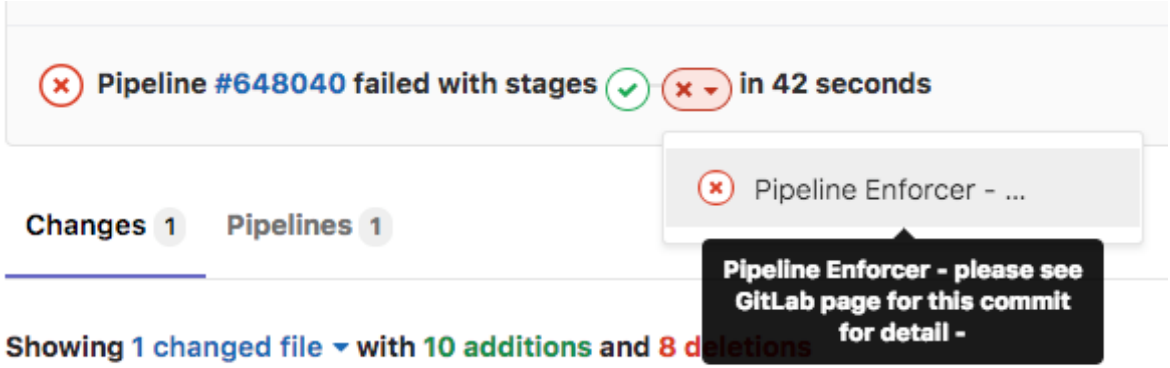
- Your pipeline is building a Docker Image but is missing a mandatory step to security scan the image.
Please see https:/ .com/display/AWS/Clair+Container+Scanning for more information.

CICD Engineering

NM Slack: #cicd-support

ENFORCEMENT

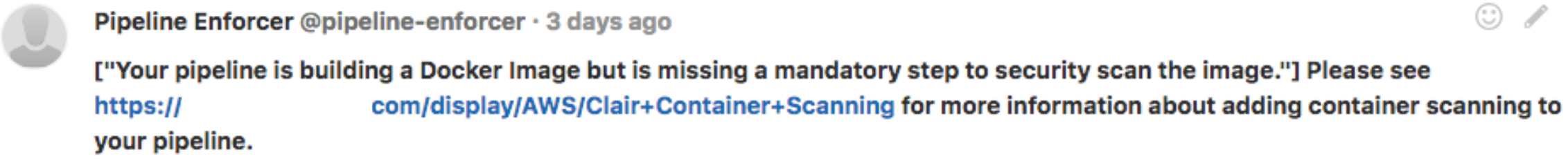
Mark commit as Failed in GitLab



Block merge if repo setting enabled.

ENFORCEMENT

Comment on commit in GitLab with link to documentation



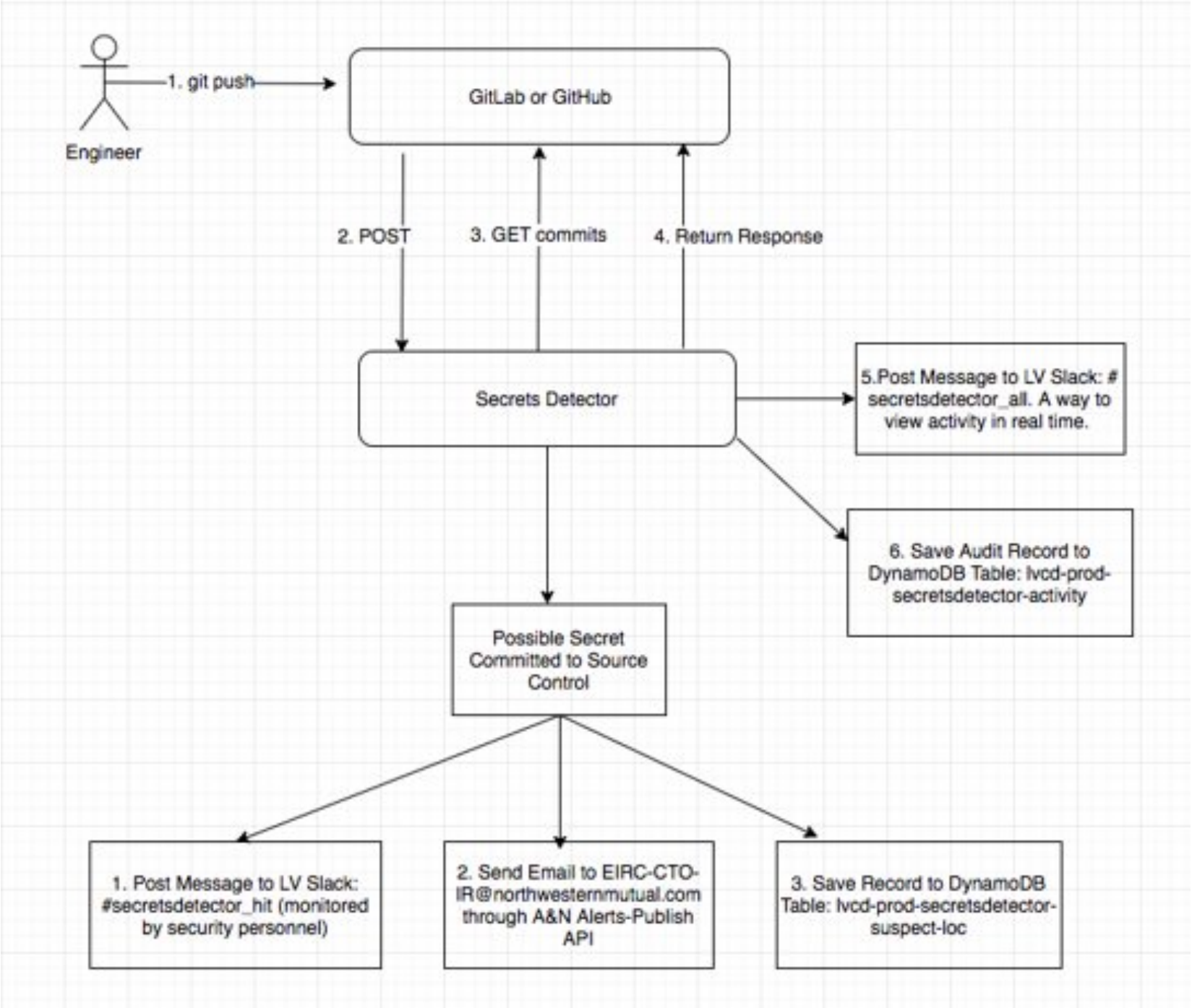
Enforcement is taken only for required scans.

For optional scans, system records the information for analysis.

SECRETS DETECTOR

Secrets Detector is a system designed to catch secrets committed to source control in real time and to alert security personnel.

System processes every commit made to SCM.


Engineer
1. git push
GitLab or GitHub
2. POST
3. GET commits
4. Return Response
Secrets Detector
5.Post Message to LV Slack: # secretsdetector_all. A way to view activity in real time.
6. Save Audit Record to DynamoDB Table: lvcd-prod-secretsdetector-activity
Possible Secret Committed to Source Control
1. Post Message to LV Slack: #secretsdetector_hit (monitored by security personnel)
2. Send Email to EIRC-CTO-IR@northwesternmutual.com through A&N Alerts-Publish API
3. Save Record to DynamoDB Table: lvcd-prod-secretsdetector-suspect-loc

WHY BUILD OUR OWN TOOL?

☐ **Prevent committing secrets to Git**

GitLab will reject any files that are likely to contain secrets. The list of file names we reject is available in the documentation.

- Detecting and Mitigating Secret-Key Leaks in Source Code Repositories
- https://github.com/dxa4481/truffleHog
- https://github.com/michenriksen/gitrob/
- https://github.com/awslabs/git-secrets
- https://github.com/zricethezav/gitleaks
- https://github.com/techjacker/diffence

DETECTING AND MITIGATING SECRET-KEY LEAKS IN SOURCE CODE REPOSITORIES

Detecting and Mitigating Secret-Key Leaks in Source Code Repositories

Vibha Singhal Sinha, Diptikalyan Saha, Pankaj Dhoolia, Rohan Padhye, Senthil Mani
IBM Research
{vibha.sinha, diptsaha, pdhoolia, ropadhye, sentmani}@in.ibm.com

***Abstract*—Several news articles in the past year highlighted incidents in which malicious users stole API keys embedded in files hosted on public source code repositories such as GitHub and BitBucket in order to drive their own work-loads for free. While some service providers such as Amazon have started taking steps to actively discover such developer carelessness by scouting public repositories and suspending leaked API keys, there is little support for tackling the problem from the code sharing platforms themselves.**

In this paper, we discuss practical solutions to detecting, preventing and fixing API key leaks. We first outline a handful of methods for detecting API keys embedded within source code, and evaluate their effectiveness using a sample set of projects from GitHub. Second, we enumerate the mechanisms which could be used by developers to prevent or fix key leaks in code repositories manually. Finally, we outline a possible solution that combines these techniques to provide tool support for protecting against key leaks in version control systems.

I. Introduction

Many web and mobile based applications interact with external services hosted by providers such as Facebook, Google and Amazon through Web APIs. The mechanism for

10,000 AWS secret access keys carelessly left in code uploaded to GitHub

By Shawn Knight on March 25, 2014, 1:00 PM

Exclusive: The co-founder of One More Cloud explains how an old AWS API key was used to take down the company's services, and the hard lessons learned.

Ryan Hellyer's AWS Nightmare: Leaked Access Keys Result in a $6,000 Bill Overnight

AWS urges developers to scrub GitHub of secret keys

Powered by SC Magazine SC

By Munir Kotadia on Mar 24, 2014 10:18 AM
Filed under Security

Dear AWS Customer,

Your security is important to us. We recently became aware that your AWS Access Key (ending with 3KFA) along with your Secret Key are publicly available on github.com . This poses a security risk to you, could lead to excessive charges from unauthorized activity or abuse, and violates the AWS Customer Agreement.

WHY BUILD OUR OWN TOOL?

☐ **Prevent committing secrets to Git**

GitLab will reject any files that are likely to contain secrets. The list of file names we reject is available in the documentation.

- Detecting and Mitigating Secret-Key Leaks in Source Code Repositories
- https://github.com/dxa4481/truffleHog
- https://github.com/michenriksen/gitrob/
- https://github.com/awslabs/git-secrets
- https://github.com/zricethezav/gitleaks
- https://github.com/techjacker/diffence

SECRET SAUCE?

```
def acme_key(blob: str) -> Tuple[int, List[str]]:
    assert isinstance(blob, str)

    tokens = list(set(re.findall(regex.RE_ACME_KEY_PATTERN, blob)))
    valid_tokens = [t for t in tokens if util.is_valid_acme_credential(t)]

    return (AlertLevel.CRITICAL, valid_tokens)
```

REAL SECRET SAUCE

```
def is_valid_acme_credential(acme_token: str) -> bool:

    headers = {"Authorization": f"token {acme_token}"}

    r = requests.get(
        "https://acme.com/api/v3/user",
        headers=headers,
        verify='/usr/local/share/ca-certificates/’)

    return r.status_code == 200
```

CAPTURED SECRETS

Name	Regex	Level
AWS Secret Credentials	`(?<![A-Za-z0-9/+])[A-Za-z0-9/+=]{40}(?![A-Za-z0-9/+=])AKIA[0-9A-Z]{16}`	CRITICAL
DSA Key	`-----BEGIN DSA PRIVATE KEY-----(?:[a-zA-Z0-9\+\=\/"']|\s){225,}-----END DSA PRIVATE KEY-----`	CRITICAL
Dropbox Link	`https:\/\/www.dropbox.com\/[(?:s|l)\/\S]+`	WARNING
Ec Key	`-----BEGIN (?:EC|ECDSA) PRIVATE KEY-----(?:[a-zA-Z0-9\+\=\/"']|\s)+?-----END (?:EC|ECDSA) PRIVATE KEY-----`	CRITICAL
Encrypted DSA Key	`-----BEGIN DSA PRIVATE KEY-----\s.*,ENCRYPTED(?:.|\s){225,}-----END DSA PRIVATE KEY-----`	CRITICAL
Encrypted Ec Key	`-----BEGIN (?:EC|ECDSA) PRIVATE KEY-----\s.*,ENCRYPTED(?:.|\s)+?-----END (?:EC|ECDSA) PRIVATE KEY-----`	CRITICAL
Encrypted Plaintext Key	`-----BEGIN ENCRYPTED PRIVATE KEY-----(?:.|\s)+?-----END ENCRYPTED PRIVATE KEY-----`	CRITICAL
Encrypted RSA Key	`-----BEGIN RSA PRIVATE KEY-----\s.*,ENCRYPTED(?:.|\s){225,}-----END RSA PRIVATE KEY-----`	CRITICAL
Github Key	`\b[a-zA-Z0-9]{40}\b`	CRITICAL
Ms Azure Oauth	`(?:https:\/\/login\.microsoftonline\.com\/(?:.*)\/oauth2\/v2.0\/token|https:\/\/login\.windows\.net\/(?:.*)\/oauth2\/token)`	WARNING
OPENSSH Key	`-----BEGIN OPENSSH PRIVATE KEY-----(?:[a-zA-Z0-9\+\=\/"']|\s){225,}-----END OPENSSH PRIVATE KEY-----`	CRITICAL
PGP Key	`-----BEGIN PGP PRIVATE KEY BLOCK-----(?:.|\s)+?-----END PGP PRIVATE KEY BLOCK-----`	CRITICAL
Plaintext Key	`-----BEGIN PRIVATE KEY-----(?:.|\s)+?-----END PRIVATE KEY-----`	CRITICAL
Putty Keys	`PuTTY-User-Key-File-2: ssh-.*\nEncryption: .*`	WARNING
RSA Key	`-----BEGIN RSA PRIVATE KEY-----(?:[a-zA-Z0-9\+\=\/"']|\s){225,}-----END RSA PRIVATE KEY-----`	CRITICAL

PRE COMMIT HOOK

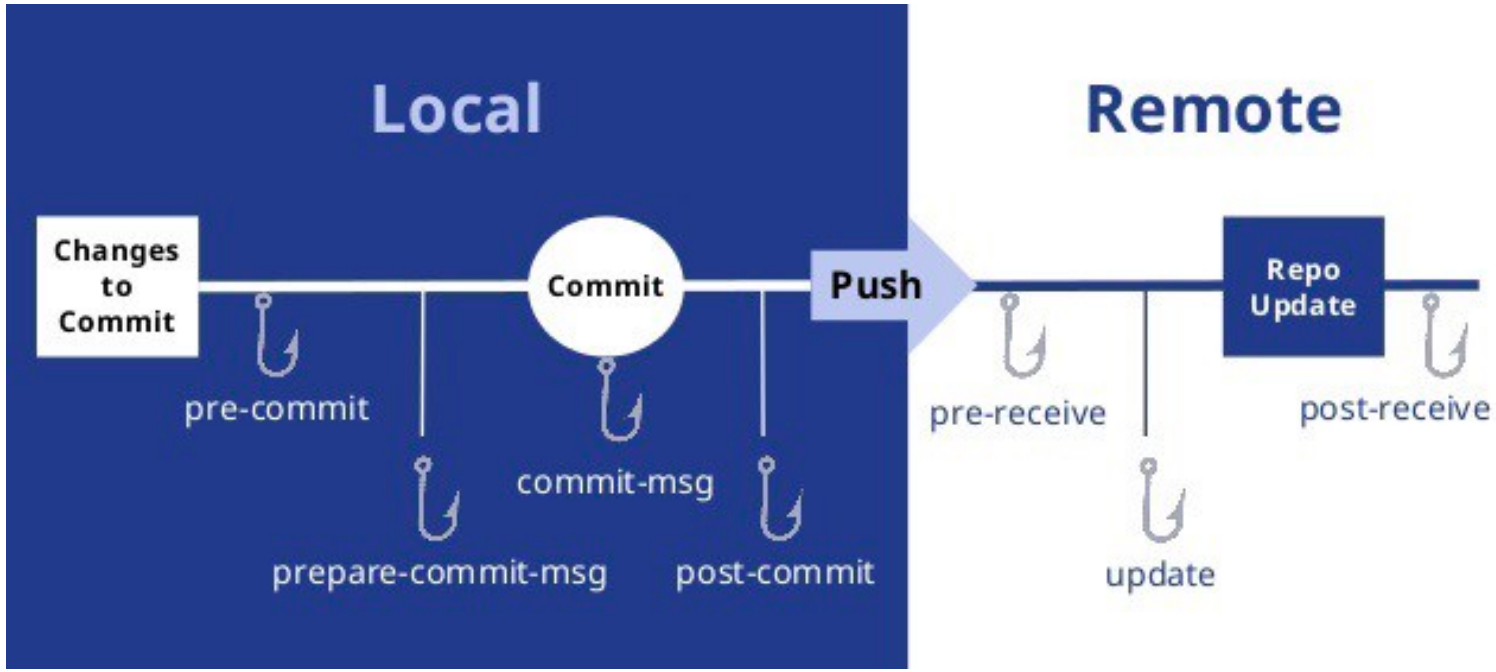
```
$ git commit -m ‘YOLO'
Secrets Detector Image is being updated, please stand by.
Image Updated
This commit has been stopped due to a potential secret being found.

vaulted_encryption_phase:
$ANSIBLE_VAULT;1.1;AES256
------------------------
github_key:
55*********
-----------------------
...
-----------------------
If you believe this is a false-positive, continue with the commit (y/n)?:n

The changes were not committed. Please look at its content (`git diff --
cached`) and unstage any unwanted changes (`git reset HEAD <file>`)
```

NEXT STEPS

Gitlab pre-receive server-side hook



https://medium.com/@suthagar23/git-hooks-keep-the-code-quality-119e6feb511e

QUESTIONS ?

THANK YOU

Northwestern Mutual®