FIRST STEPS TO FULL LIFECYCLE SECURITY WITH OPEN SOURCE TOOLS

INTRODUCTION

- Anaïs Urlichs
  - Open Source Developer Advocate @ Aqua
  - CNCF Ambassador
  - YouTube AnaisUrlichs
- Rory McCune
  - Cloud Native Security Advocate @ Aqua
  - **Twitter/Github** - @raesene

HIGH LEVEL WORKSHOP OBJECTIVES

- Scanning Containers and IaC in Development
- Testing in the CI/CD pipeline
- Security in Production

COURSE PRE-REQUISITES

- Ability to run a local Kubernetes cluster
  - KinD
  - minikube
  - microk8s
  - ...
- Ability to download and run binaries in :-
  - Linux
  - MacOS
  - FreeBSD?!

COURSE LOGISTICS

- Ground rules
- Materials
  - Slides - http://slides.pwndland.uk
  - commands - http://commands.pwndland.uk
  - setup notes - http://setup.pwndland.uk
- Questions? Just Ask!

SECURITY IN DEVELOPMENT

- Vulnerability Scanning
- IaC Mis-configuration Scanning

SECURITY SCANNING PROCESS

- Before using any third-party resources
- During Develpment
- Before Deployment

VULNERABILITY SCANNING

- Using a container image vulnerability scanning tool is a useful way of assessing base images and also built container images.
- We'll demonstrate this with Trivy

HOW DO CONTAINER VULNERABILITY SCANNERS WORK?

- Generally look at two types of information
  - OS packages (e.g.debian, alpine, RHEL)
  - Programming language packages (e.g. npm, rubygems)
- Assess whether there are known vulnerabilities in the installed versions

OPEN SOURCE CONTAINER VULNERABILITY SCANNERS

- Trivy
- Grype
- Clair
- Snyk **CLI Only**

INSTALLING TRIVY

- Several options on the install page
  - https://aquasecurity.github.io/trivy/v0.28.0/getting-started/installation/
- Homebrew for MacOS
- APT repository for Debian/Ubuntu
- YUM repo for RHEL/CentOS
- Let's install Trivy!

USING TRIVY TO SCAN IMAGES

```
trivy i ubuntu:20.04
```

```
trivy i public.ecr.aws/docker/library/ubuntu:20.04
```

IGNORE UNFIXED

```
trivy i --ignore-unfixed ubuntu:20.04
```

```
trivy i --ignore-unfixed public.ecr.aws/docker/library/ubuntu:2
```

LOOKING FOR HIGH/CRITICAL OS ISSUES

```
trivy image --severity HIGH,CRITICAL --vuln-type os postgres:10.
```

```
trivy image --severity HIGH,CRITICAL --vuln-type os public.ecr.a
```

LOOKING FOR HIGH/CRITICAL LIBRARY ISSUES

```
trivy image --severity HIGH,CRITICAL --vuln-type library node:1(
```

```
trivy image --severity HIGH,CRITICAL --vuln-type library public.
```

SCANNING GITHUB REPOSITORIES

```
trivy repo --vuln-type library https://github.com/raesene/sycamo
```

SCANNING THE FILESYSTEM

- Pick a project on your machine or

```
git clone https://github.com/raesene/sycamore
```

- Scan a whole directory

```
trivy fs ./sycamore/
```

- Scan a file

```
trivy fs ./sycamore/yarn.lock
```

JSON OUTPUT

```
trivy i --format json raesene/spring4shelldemo:latest
```

USING JQ TO FIND A SPECIFIC ISSUE

```
trivy -q i --format json raesene/spring4shelldemo:latest | jq '
```

CONFIGURATION SCANNING

- A good practice during development or when using 3rd party projects
- Can flag up where good security practices aren't being followed
- Rulesets vary by tool, although some can be based on standards (e.g. Kubernetes PSS, CIS Benchmarks)

CONFIGURATION SCANNING - DOCKER

```
git clone https://github.com/AnaisUrlichs/trivy-demo.git
cd trivy-demo
```

```
trivy config bad_iac/docker/
```

FIXING A DOCKER ISSUE

- Uncomment the USER line in the Dockerfile

```
trivy config bad_iac/docker/
```

CONFIGURATION SCANNING - KUBERNETES

```
trivy config bad_iac/kubernetes/
```

FIXING A KUBERNETES ISSUE

- Pick an issue from the ones flagged up and see if you can fix it, then re-scan

```
trivy config bad_iac/kubernetes/
```

CONFIGURATION SCANNING - TERRAFORM

```
trivy config bad_iac/terraform/
```

TRIVY SBOM

```
trivy sbom ubuntu:20.04
```

*also available as Docker Desktop Extension

SECURITY SCANNING IN CI/CD

- Applying the same checks as are available in development, in CI/CD pipelines provides an additional layer of security.
- Using GitHub Actions and SARIF, we can automate security checks either periodically or as code is checked in

USING A GITHUB ACTION TO BUILD AND SCAN A DOCKER IMAGE

- Here's an example https://github.com/raesene/sycamore/blob/main/.github/workflows/publish.yml
  - It's a modified version of GitHub's basic Docke

PERMISSIONS TO BUILD+SCAN AN IMAGE

```yaml
permissions:
    contents: read
    packages: write
    # This is used to complete the identity challenge
    # with sigstore/fulcio when running outside of PRs.
    id-token: write
    security-events: write   # To upload sarif files
```

RUNNING TRIVY

```
- name: Run trivy
        uses: aquasecurity/trivy-action@master
        with:
          image-ref: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}:$
          format: sarif
          output: 'trivy-results.sarif'
```

UPLOADING RESULTS TO GITHUB SECURITY

```
- name: Upload Trivy scan results to GitHub Security tab
      uses: github/codeql-action/upload-sarif@v1
      with:
        sarif_file: 'trivy-results.sarif'
```

CONFIG SCANNING IN CI/CD WITH TRIVY

```
    - name: Run Trivy in Config mode to generate SARIF
      uses: aquasecurity/trivy-action@master
      with:
        scan-type: 'config'
        hide-progress: false
        format: 'sarif'
        output: 'trivy-results.sarif'
```

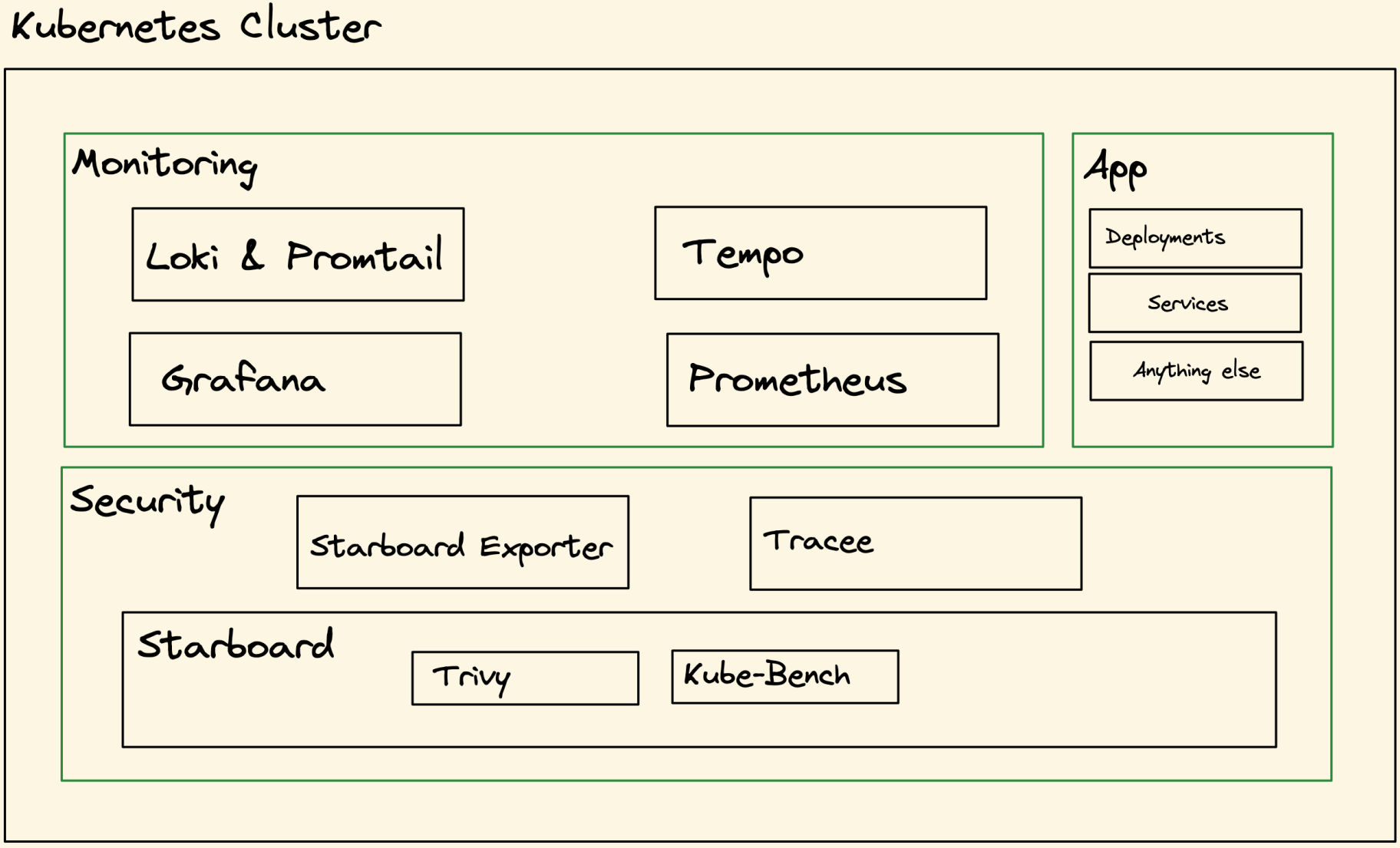
OPEN SOURCE SECURITY IN PRODUCTION

- Once our workloads are deployed to clusters we need on-going security
- Regular scans for compliance and assurance
- Runtime security to detect attacks

KUBERNETES OPERATOR

Automating human behaviour through controllers

https://www.cncf.io/wp-content/uploads/2021/07/CNCF_Operator_WhitePaper


Kubernetes Cluster
Monitoring
Loki & Promtail
Tempo
Grafana
Prometheus
App
Deployments
Services
Anything else
Security
Starboard Exporter
Tracee
Starboard
Trivy
Kube-Bench

INSTALLING STARBOARD OPERATOR

```
kubectl apply -f https://raw.githubusercontent.com/aquasecurity/
```

CRDS

- Starboard creates a number of CRDs to hold scan data.

```
ciskubebenchreports.aquasecurity.github.io
clustercompliancedetailreports.aquasecurity.github.io
clustercompliancereports.aquasecurity.github.io
clusterconfigauditreports.aquasecurity.github.io
configauditreports.aquasecurity.github.io
vulnerabilityreports.aquasecurity.github.io
```

CONFIRMING ALL IS WELL

- Checking the deployment of the operator tells us if the install was successful

```
kubectl get deployment -n starboard-system
```

CREATING A DEPLOYMENT

```
kubectl create ns app

kubectl apply -f https://raw.githubusercontent.com/AnaisUrlichs/
```

VULNERABILITYREPORT

Automatically scans the containers that are used inside of your cluster.

Deployment-scoped

```
kubectl get vulnerabilityreports -o wide -n app
```

Cluster-scoped

```
kubectl get clustervulnerabilityreports -o wide -n app
```

CONFIGURATION AUDITING

Kubernetes configurations are checked against built-in policies

Deployment-scoped

```
kubectl get configauditreports -o wide -n app
```

Cluster-scoped

```
kubectl get clusterconfigauditreports -o wide -n app
```

INFRASTRUCTURE SCANNING

- CIS benchmark for Kubernetes nodes provided by kube-bench.
- Penetration test results for a Kubernetes cluster provided by kube-hunter.

CISKUBEBENCHREPORT

Maps CIS Benchmarks against Kubernetes version

```
kubectl get nodes

kubectl get ciskubebenchreports -o wide

kubectl describe ciskubebenchreports/<insert report name>
```

OTHER TOOL TO PRODUCE CIS BENCHMARKS

- https://github.com/chen-keinan/kube-beacon

CLUSTER COMPLIANCE REPORT

NSA report

ClusterCompliance and ClusterComplianceDetail Report

```
kubectl apply -f https://raw.githubusercontent.com/aquasecurity/

kubectl get clustercompliancereport -o wide

kubectl describe clustercompliancereport nsa

kubectl get clustercompliancedetailreport -o wide
```

NSA REPORTS PRODUCED BY OTHER TOOLS

```
❯ kubescape scan framework nsa --submit
[info] ARMO security scanner starting
[warning] Kubernetes cluster nodes scanning is disabled. This is required to collect valuable data for certain controls. You can enable it using  the --enable-host-scan flag
[info] Downloading/Loading policy definitions
[success] Downloaded/Loaded policy
[info] Accessing Kubernetes objects
[success] Accessed to Kubernetes objects
[info] Scanning. cluster: do-fra1-demo-cluster
[success] Done scanning. cluster: do-fra1-demo-cluster

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Controls: 19 (Failed: 15, Excluded: 0, Skipped: 2)

+----------+-----------------------------------------+------------------+------------------+---------------+---------------+
| SEVERITY |                CONTROL NAME             | FAILED RESOURCES | EXCLUDED RESOURCES | ALL RESOURCES | % RISK-SCORE |
+----------+-----------------------------------------+------------------+------------------+---------------+---------------+
| Critical | Disable anonymous access to Kubelet service |        0         |         0        |       0       |    skipped*   |
| Critical | Enforce Kubelet client TLS authentication |        0         |         0        |       0       |    skipped*   |
| High     | Cluster-admin binding                   |        2         |         0        |      84       |       2%      |
| High     | Privileged container                    |        4         |         0        |       8       |      38%      |
| Medium   | Allow privilege escalation              |        7         |         0        |       8       |      79%      |
| Medium   | Allowed hostPath                        |        3         |         0        |       8       |      29%      |
| Medium   | Automatic mapping of service account    |        57        |         0        |      57       |      100%     |
| Medium   | Cluster internal networking             |        5         |         0        |       5       |      100%     |
| Medium   | Container hostPort                      |        1         |         0        |       8       |      10%      |
| Medium   | Exec into container                     |        2         |         0        |      84       |       2%      |
| Medium   | HostNetwork access                      |        5         |         0        |       8       |      48%      |
| Medium   | Ingress and Egress blocked              |        8         |         0        |       8       |      100%     |
| Medium   | Insecure capabilities                   |        2         |         0        |       8       |      19%      |
| Medium   | Linux hardening                         |        7         |         0        |       8       |      79%      |
| Medium   | Non-root containers                     |        8         |         0        |       8       |      100%     |
| Low      | Immutable container filesystem          |        7         |         0        |       8       |      79%      |
| Low      | Resource policies                       |        5         |         0        |       8       |      60%      |
+----------+-----------------------------------------+------------------+------------------+---------------+---------------+
|          |              RESOURCE SUMMARY          |        64        |         0        |      157      |     30.66%    |
+----------+-----------------------------------------+------------------+------------------+---------------+---------------+
FRAMEWORK NSA
```

https://github.com/armosec/kubescape

CUSTOM POLICIES

- Trivy/Starboard: Write custom policies using Rego
- tfsec: Custom policies in JSON/YAML

```
trivy config --policy ./custom-policies --namespaces user ./mani
```

WHAT IS NEXT?

- Integrate Starboard metrics into your observability stack
- Try out Trivy's functionality in your own projects
- Let us know what use cases you would like to see

LINKS

Repository https://github.com/AnaisUrlichs/trivy-demo

Aqua GitHub https://github.com/aquasecurity

Rory's Twitter https://twitter.com/raesene

Anais' Twitter https://twitter.com/urlichsanais

THANK YOU!

- Rory's Twitter/GitHub: @raesene
- Anais' Twitter: @urlichsanais