ATTACKING AND DEFENDING KUBERNETES

Jay Beale, CTO and COO at InGuardians

@jaybeale and @InGuardians

December 7th, 2018

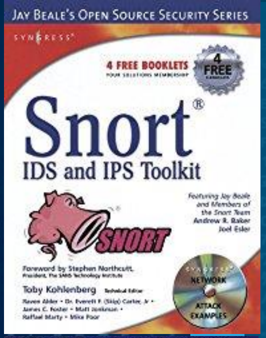
https://www.InGuardians.com

Jay Beale is a Linux security expert who has created several defensive security tools, including Bastille Linux/UNIX and the CIS Linux Scoring Tool, both of which were used widely throughout industry and government. He has served as an invited speaker at many industry and government conferences, a columnist for Information Security Magazine, SecurityPortal and SecurityFocus, and an author or editor of nine books, including those in his Open Source Security Series and the "Stealing the Network" series. He has led training classes on Linux Hardening and other topics at Black Hat, CanSecWest, RSA, and IDG conferences, as well as in private corporate training, since 2000. Jay is a co-founder, Chief Operating Officer and CTO of the information security consulting company InGuardians.

InGuardians is a leading information security consultancy with offices in Seattle, Boston, Chicago, Dallas, Atlanta and Washington, DC.

INGUARDIANS™

Graphical Bio


Jay Beale's Open Source Security Series

Syngress

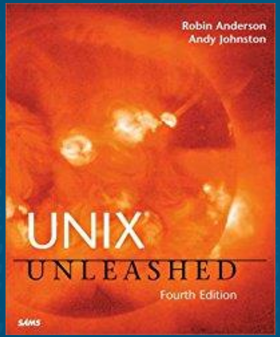
4 FREE BOOKLETS

Snort® IDS and IPS Toolkit

Featuring Jay Beale and Members of the Snort Team Andrew R. Baker Joel Esler

Foreword by Stephen Northcutt

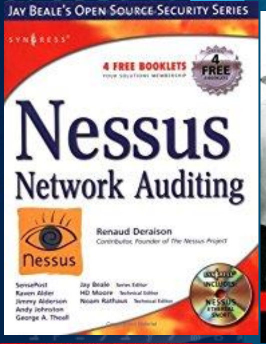
Toby Kohlenberg


Robin Anderson
Andy Johnston

UNIX UNLEASHED

Fourth Edition

SAMS

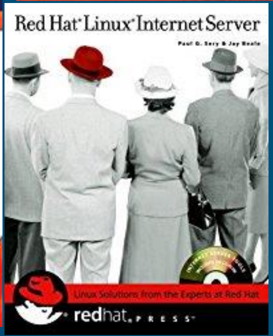

Jay Beale's Open Source Security Series

Syngress

4 FREE BOOKLETS

Nessus Network Auditing

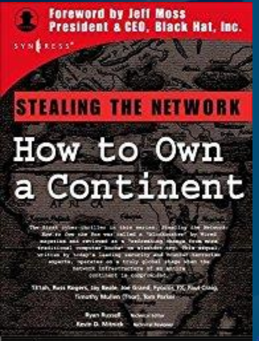
Renaud Deraison


Red Hat® Linux® Internet Server

Paul G. Sery & Jay Beale

Linux Solutions from the Experts at Red Hat

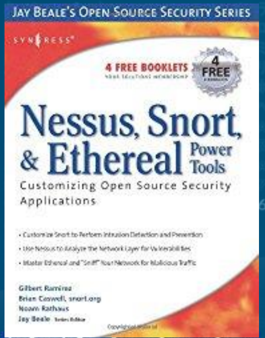
redhat. PRESS


Foreword by Jeff Moss
President & CEO, Black Hat, Inc.

Syngress

STEALING THE NETWORK

How to Own a Continent

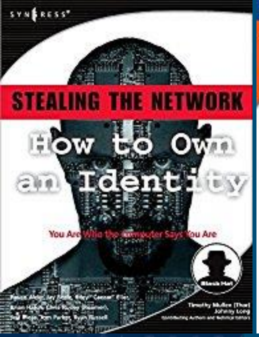

Jay Beale's Open Source Security Series

Syngress

4 FREE BOOKLETS

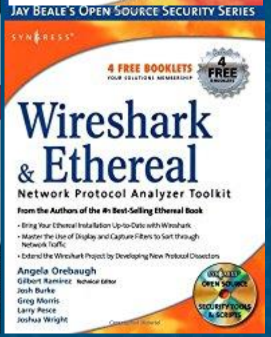
Nessus, Snort, & Ethereal Power Tools

Customizing Open Source Security Applications


Syngress

STEALING THE NETWORK

How to Own an Identity


Jay Beale's Open Source Security Series

Syngress

4 FREE BOOKLETS

Wireshark & Ethereal

Network Protocol Analyzer Toolkit

Angela Orebaugh

black hat®

AIKIDO ON THE COMMAND LINE - LINUX LOCKDOWN AND PROACTIVE SECURITY

JAY BEALE, INGUARDIANS | AUGUST 4-5 & AUGUST 6-7

CIS Center for Internet Security®

OVAL®

BASTILLE

http://bastille-linux.sourceforge.net/

INFORMATION SECURITY

Going the Distance

Reach beyond minimum standards with a strong data protection program and simplify compliance

SANS

DEFCON 11

toorcon

SecurityFocus™

CAN SEC WEST

INGUARDIANS™ RED TEAM

INGUARDIANS™

Talk Table of Contents

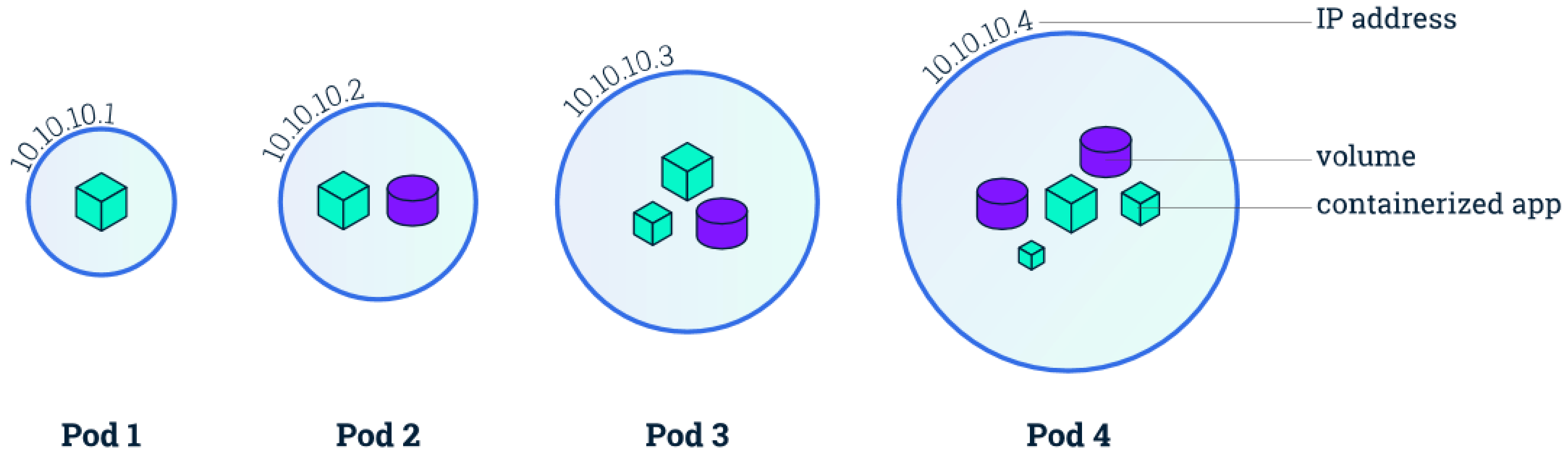
InGuardians™

What Does Kubernetes Do?

- Container Orchestration / software-defined datacenter
- Horizontal Scaling
- Automatic Binpacking
- Self Healing
- Automated Rollouts and Rollbacks
- Service Discovery and Load Balancing
- Storage Orchestration
- Secret and Configuration Management

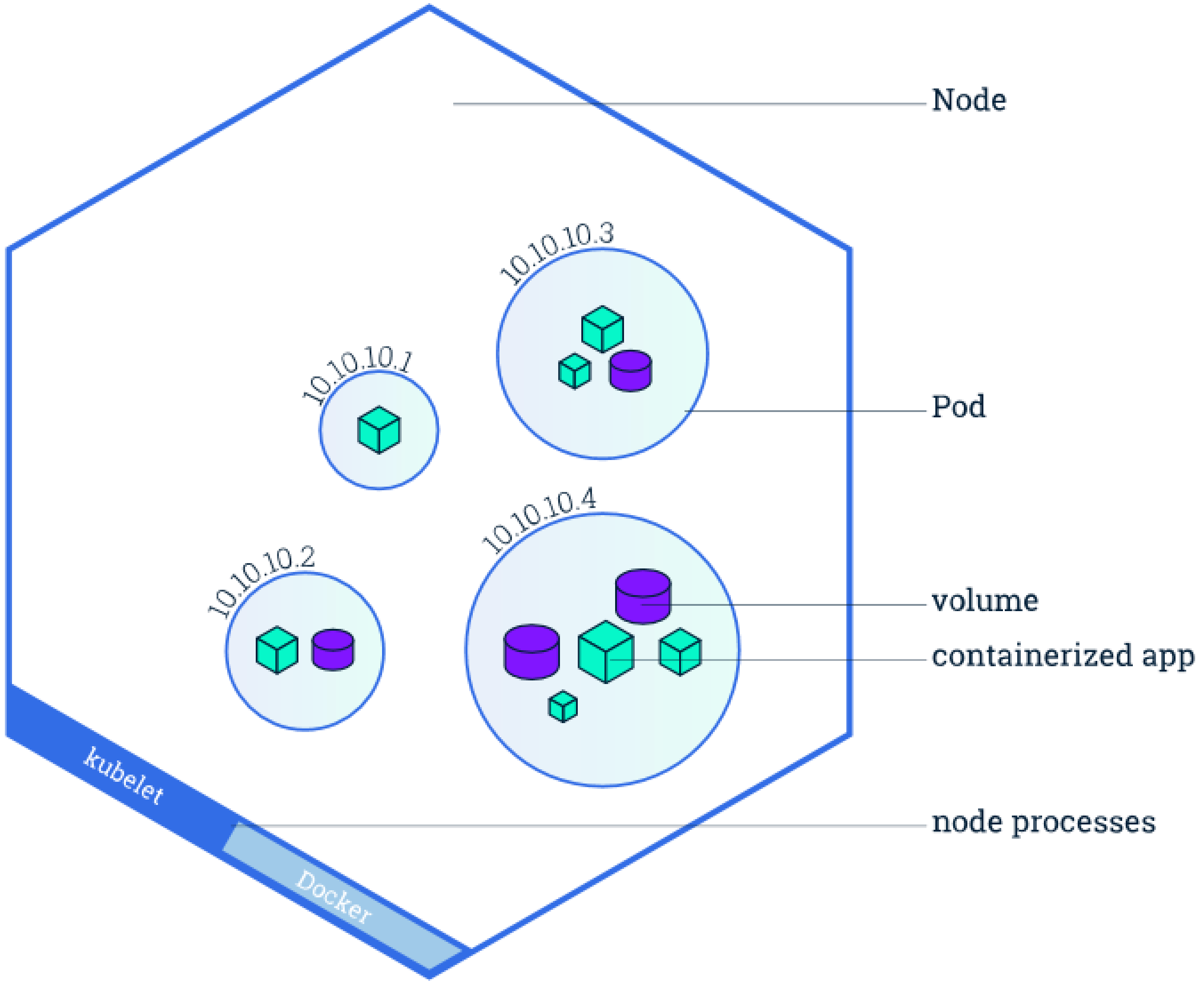
Kubernetes Pods

- A pod is a collection of one or more containers
- Pods are the smallest unit of work in Kubernetes.
- Pods always include a “pause” container.
- Containers in a pod share a single network kernel namespace
  - All containers in a pod have the same IP address
  - Programs across a pod must avoid binding to the same port numbers
- The "pod" construct expresses shares-a-host dependency between containers
  - If two programs need to be on the same host, their containers go into a pod together.


10.10.10.1
10.10.10.2
10.10.10.3
10.10.10.4
IP address
volume
containerized app
Pod 1
Pod 2
Pod 3
Pod 4

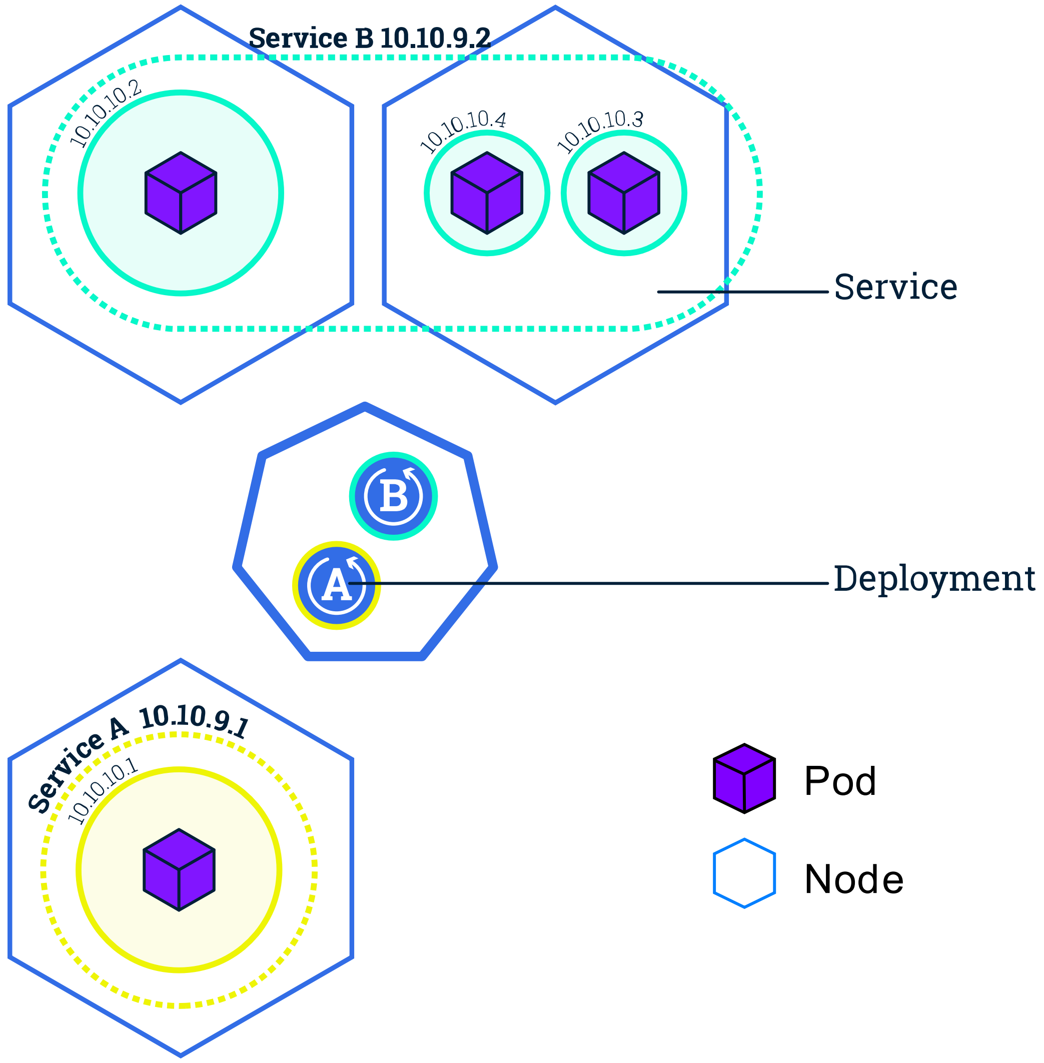
Kubernetes Nodes

- A node is a Kubernetes system where containers are staged
- A node runs a container runtime, like Docker, a Kubelet, and a Kube-proxy.
  - The Kubelet tells Docker what to create, destroy or configure.
  - The Kube-proxy configures iptables.
- For organizations' private clouds, the nodes may be physical machines.
- In public clouds, the nodes are generally virtual machines.


Node
10.10.10.3
10.10.10.1
Pod
10.10.10.4
10.10.10.2
volume
containerized app
kubelet
Docker
node processes

Kubernetes Services

- A service is a load balancer that creates a single IP address and port that maps to a specific set of pods, identified by their labels



INGUARDIANS™

Kubernetes Namespaces

- A namespace is a logical grouping for Kubernetes objects (pods, roles, …)
- Namespaces might separate departments, development groups, or tenants
- Every cluster starts with two namespaces:
  - All Kubernetes components run in the **kube-system** namespace
  - All user-added pods start in the **default** namespace
- K8S namespaces are not the same as Linux kernel / Docker namespaces

InGuardians™

Kubernetes Glossary

- Terms Reminder
  - Containers: Linux namespace-based lightweight VMs
  - Pods: collections of containers, smallest unit of work in K8S
  - Images: the persistent state of containers
  - Nodes: hosts on which the containers/pods run
  - Namespaces (Kubernetes): logical grouping, possibly by tenant, department or application

Kubernetes and YAML/JSON

- Prefers “declarative” rather than “imperative” usage.
- You tell Kubernetes that you’d like five (5) copies of this application running.
- Kubernetes takes responsibility for keeping five containers staged, spread out to as many as five nodes, watching for container or node failures.
- You build YAML files describing what you want, pass these to the API server, and let it take responsibility for effecting that declaration.

```
kubectl apply -f file.yaml
```

Vital Kubernetes Target Components (1 of 2)

- Kubernetes API Server
  - Accepts the declarative configurations and directs other components to take action.
- Kubelet
  - Runs on each node in the cluster, bridging the Kubernetes infrastructure to the container runtime (often Docker)
- Container Runtime/Docker
  - Pulls container images and instructs the kernel to start up containers

Vital Kubernetes Target Components (2 of 2)

- ETCD Server
  - Retains the state of the cluster
- Kubernetes Dashboard
  - Web interface that permits configuration and administration
- Metrics Components
  - Provide useful data about the target

Attacking Kubernetes Clusters

- An attack on Kubernetes generally starts from the perspective of a compromised pod.
  - The threat actor may have compromised the application running in one container in the pod.
  - The threat actor may have phished/compromised a person who had access to the pod.
  - The threat actor may be a user who is looking to escalate their privileges.

Threat Actor Actions (1 of 2)

- An attacker in a pod may:
  - Use the access provided by the pod to access other services
  - Attack other containers in their pod
  - Make requests to the API server or a Kubelet to:
    - Run commands (possibly interactively) in a different pod
    - Start a new pod with privilege and node filesystem/resource access
  - Gather secrets that Kubernetes provides to pods
  - Connect to the Kubernetes Dashboard to perform actions

Threat Actor Actions (2 of 2)

- An attacker in a pod may:
  - Interact with the etcd server to change the cluster state
  - Interact with the cloud service provider using the cluster owner’s account.

Demo: Kubernetes Attack - Owning the Nodes

We'll compromise a Kubernetes cluster, starting from a vulnerable application, running in a pod on the cluster.

https://youtu.be/ATpGJ_zgrHM

Watch the InGuardians website for a download link for the cluster.

InGuardians™

Dissecting the Own The Node Attack

- We got a Meterpreter running in the frontend pod.
- We interacted with the API server and tried to stage a custom pod.
- We moved laterally to a Redis pod, which had a better cluster role.
- We staged a custom pod with a hostPath mount onto a node, compromising it.
- We staged pods to every node using a Daemon Set, compromising every one.

Demo: Multitenant Kubernetes Attack

In this video demo, we'll attack a Kubernetes cluster that has a soft multitenancy setup, with a Marketing department and a Development department.

Multitenant Kubernetes Cluster Attack Demo:

https://youtu.be/wuSsyk_Y7_s

Dissecting the Attack Path

- Gained a Meterpreter in Marketing's Wordpress container. (Flag 1)
- Moved into Marketing's MySQL container (Flag 2)
- Used the MySQL container's unfettered network access to reach a Kubelet on the master node.
- Used the Kubelet's lack of authentication to invade Development's dev-web container. (Flag 3)
- Reasoning that the dev-sync container in this same pod might be used to synchronize content, gained the pod's secrets (SSH key and account).

Dissecting the Attack Path

- Authenticated to the high-value Developer machine. (Flag 4)
- Returned to the cluster, used the dev-web pod's placement on the master to gain control of the AWS account. (Bonus)

Defending Kubernetes

- Let's discuss the defenses available to Kubernetes cluster owners.
- Along the way, we'll demonstrate two of those defenses to break steps in the multitenant attack path.

Defense Overarching Note

- You must upgrade your Kubernetes cluster.
- Kubernetes development is moving very quickly, with many of the features we're about to discuss only moving out of alpha or beta in the last few major releases.
- Default settings in Kubernetes (and its third-party installers) continue to strengthen substantially.
- Support periods (patching) here resembles the world of smart phones far more than the world of desktop operating systems.

InGuardians™

Defense: RBAC and Authz

- Role-based Access Control (RBAC)
- Node Authorization
- Webhook to protect the Kubelet
- Removing default service account permissions

INGUARDIANS™

Defense: RBAC and Authz

- You can place restrictions on the API server via RBAC.
- Requests looks like:
  - Username (Subject)
    - Ex: [jay in group system:authenticated]
  - Verb
    - Ex: [in inguardians-ns, get pods]
- You provide the ability to do these things by creating:
  - Role/ClusterRole
  - RoleBinding

Create a Role and RoleBinding

```
kind: Role
apiVersion: …
metadata:
   name: ing-pod-getter
   namespace: inguardians-ns
rules:
- verbs: [“get”]
  apiGroups: [“”]
  resources: [“pods”]
```

```
kind: RoleBinding
apiVersion: …
metadata:
   name: jay-pod-getter
   namespace: inguardians-ns
roleRef:
  kind: Role
  apiGroup: …
  name: ing-pod-getter
Subjects:
- kind: User
  apiGroup: …
  name: jay
```

INGUARDIANS™

Demo: Creating the Role and Rolebinding

https://youtu.be/ZUiQLzWXz-I

Demo: Testing the Exec to Confirm Hardening

https://youtu.be/4o7mM8VjGh0

Creating Custom Service Accounts Automatically

- Jordan Liggitt wrote a tool called Audit2RBAC, similar to Audit2Allow for SELinux.
- https://github.com/liggitt/audit2rbac/
- Let's see a demo video.

Node Authorization

- Enable the NodeRestriction admission plugin to prevent a kubelet on a node from modifying other nodes.
- The API server must include `--authorization-mode=Node`.

- Reference:

https://kubernetes.io/docs/admin/authorization/node/

Webhook Authorization on the Kubelet

- Use Webhook to protect the Kubelet from disclosing information about itself on API paths, such as:
  - `/logs`
  - `/debug`
  - `/metrics`

- Reference:

  https://kubernetes.io/docs/admin/authorization/webhook/

Demo: Securing the Kubelet Against Anonymous Access

https://youtu.be/bd5qgcM5nFg

Network Policies

- Network policies let you set firewall rules, using label selection.
- You create one or more policies.
- Each policy names pods that it refers to via a podSelector.
- Rules are for ingress and/or egress.
- Once you create a network policy for a pod, you have a default deny for traffic for that pod in that direction.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: yourpolicy
  namespace: yourns
spec:
  podSelector:
  ingress:
  egress:
```

Network Policy Example

- This policy allows traffic **IN** to pods with labels:
  - **`app = myapp`**
  - **`role = api`**
- It permits traffic **only from** pods with label **`app`** set to **`myapp`**.
- These labels have no inherent meaning.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: api-allow
spec:
  podSelector:
    matchLabels:
      app: myapp
      role: api
  ingress:
  - from:
      - podSelector:
          matchLabels:
            app: myapp
```

INGUARDIANS™

Network Policy Tip

- Avoid this gotcha:

  In a podSelector, {} means “all pods”

whereas:

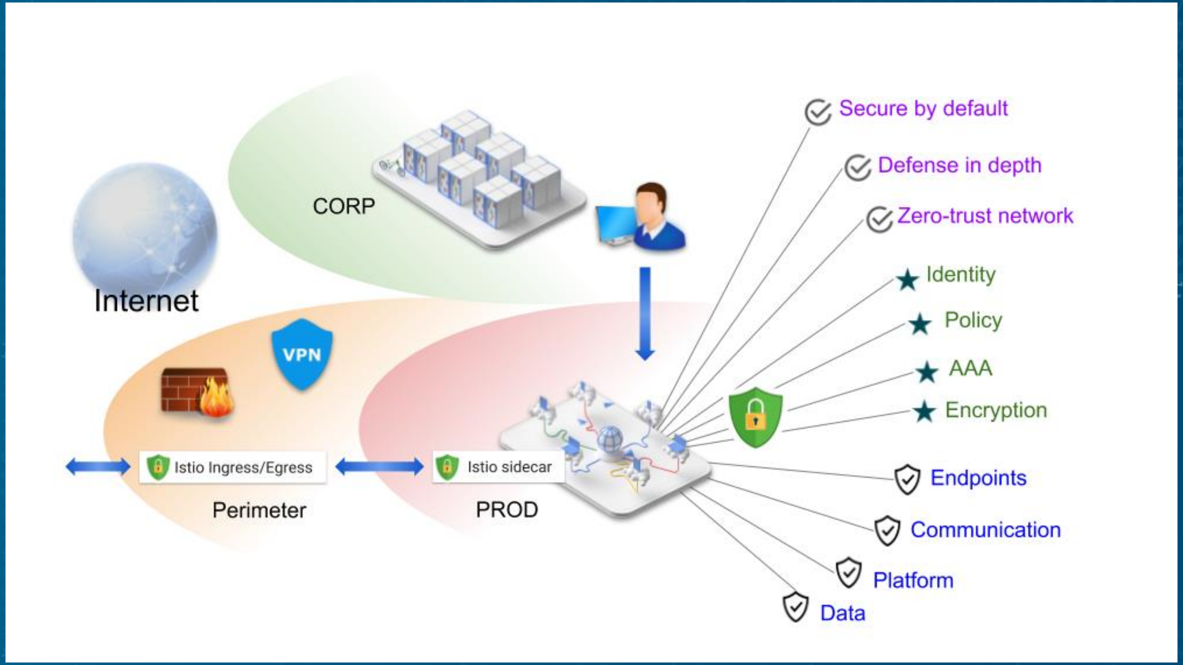
  In an ingress/egress list of traffic, [] means “nothing”

Network Policy Defense Against the Demo

- We can create a default deny egress policy for our pods, such than the attacker can't reach the Metadata API from those pods.
- We'll also discuss a few measures that bear less resemblance to a sledgehammer.

Defense: Service Meshes

- Service meshes bring encryption, service authentication, traffic control and observation to Kubernetes, among other features.
- Istio is one example, wherein each pod is given a sidecar proxy through which all network traffic will flow.



Reference and Image credit:

https://istio.io/docs/concepts/security/

Defense: Pod Security Policies (PSP)

- Pod Security Policies allow you to restrict the privilege with which a pod runs.
  - Volume white-listing / Usage of the node’s filesystem
  - Read-only root filesystem
  - Run as a specific (non-root) user
  - Prevent privileged containers (all capabilities, all devices, …)
  - Root capability maximum set
  - SELinux or AppArmor profiles – choose from a set
  - Seccomp maximum set

Pod Security Policy: Root Capability Supersets

- All of the "magic powers" that the root user has are named and numbered, codified in a POSIX standard called "capabilities."
- Here are a few of the most common ones that containers still maintain:
  - NET_BIND_SERVICE - Bind to TCP/UDP privileged ports (<1024).
  - DAC_OVERRIDE - Bypass file read, write & execute permission checks
  - CHOWN - Make arbitrary changes to file UIDs and GIDs
  - SETUID - Make arbitrary manipulations of process UIDs
  - KILL - Bypass permission checks for sending signals

Testing Out Root Capability Dropping

- In Docker, you can control what capabilities Docker retains by using the --cap-add and --cap-drop flags.
- The following tells Docker to drop all capabilities except net_bind_service, which lets us bind to a privileged (<1024) port.

```
docker run --cap-drop ALL --cap-add net_bind_service image /bin/bash
```

- Look up capability names and descriptions via the **capabilities** man page.

Pod Security Policy: Seccomp Filters

- You can restrict a container's available system calls (syscalls) with seccomp, locking the set of system calls to the ones the program used when uncompromised.
- This has two purposes:
  - Restrict what a compromised program can do
  - Reduce the kernel's attack surface
- Kubernetes can require that any pod running must a seccomp filter from a set that the cluster administrators vet.

Creating seccomp Filters

- Jessie Frazelle has led much of the container seccomp work. This blog post details her experience:

  https://blog.jessfraz.com/post/how-to-use-new-docker-seccomp-profiles/

- In my classes, we test this with a virtual machine, to which we've added a sample vulnerable service running in a Docker container.

Seccomp Step 1: Create a Dockerfile

```
# Start from a base container image
FROM centos:7
# Install strace and the program to profile
RUN yum update -y
RUN yum install -y strace vsftpd
# Configure the program to profile
COPY vsftpd.conf /etc/vsftpd/vsftpd.conf
RUN chmod go+rw /var/ftp/pub/
# Expose the network port and start the program
EXPOSE 21/tcp
ENTRYPOINT ["/usr/bin/strace","-ff","vsftpd"]
```

Seccomp Step 2: Build Docker Image

- Build a Docker image from that Dockerfile:

```
# docker build -t strace-vsftpd .
```

- Start a container based on the image.

```
# docker run -d  --security-opt seccomp=unconfined --name test-vsftpd strace-vsftpd
```

- Use docker-inspect to get the container’s IP address.

```
# docker inspect test-vsftpd
```

Seccomp Step 3: Exercise the Program

```
# ftp 172.17.0.2
…
Name (172.17.0.2:root): anonymous
Password: jay@harden-linux.com
230 Login successful.
…
ftp> cd /pub
250 Directory successfully changed.
```

```
ftp> ls
226 Directory send OK.
ftp> put Dockerfile
…
ftp> lcd ..
…
ftp> get Dockerfile
ftp> exit
```

Seccomp Step 4: Parse Logs to Profile

- Capture the strace output into vsftpd-strace.log:

```
# docker logs test-vsftpd > vsftpd-strace.log 2>&1
```

- Convert the strace output to a syscall profile:

```
# strace-to-seccomp vsftpd-strace.log >seccomp.json
```

- Try the new seccomp profile.

```
# docker run -d  --security-opt seccomp=seccomp.json strace-vsftpd
```

CIS Benchmark

- Additionally, you can find hardening steps for a Kubernetes cluster in the Center for Internet Security’s benchmark document for Kubernetes.

https://www.cisecurity.org/benchmark/kubernetes/

- Test each change to confirm that it fulfills your intent.

Metadata Concealment

- If you won't be cutting off a pod's access to the Metadata API with network policies or a service mesh, you have options.
- AWS: Kube2IAM

  https://github.com/jtblin/kube2iam
- GCE/GKE: GCE Metadata Proxy

  https://github.com/GoogleCloudPlatform/k8s-metadata-proxy
- On GKE, there's a beta feature for Metadata Concealment:

https://cloud.google.com/kubernetes-engine/docs/how-to/metadata-concealment

Image Safety

- It's critical that you understand the upstream source of your container images.
- Are your images cryptographically signed?
- Have you scanned them with CoreOS Clair?
- Who created them?
- References:

https://docs.docker.com/engine/security/trust/

https://docs.docker.com/engine/security/trust/content_trust/

InGuardians™

Further References

- Kubernetes Up and Running (O’Reilly)
- Kubernetes Cookbook (O’Reilly)
- Ahmet Balkan’s re-useable Network Policy recipes:

  https://github.com/ahmetb/kubernetes-network-policy-recipes
- Jordan Liggitt’s Audit2RBAC

  https://github.com/liggitt/audit2rbac
- Kubernetes Documentation

  https://kubernetes.io/docs/

Non-public Tool Demo

- InGuardians has been working on a number of Kubernetes security tools.
- One, called Peirates (greek for "Pirates") is an attack tool, written by:
  - Adam Crompton
  - Dave Mayer
  - Faith Alderson
  - Jay Beale

Easter Egg!

http://BustaKube.com

INGUARDIANS™

WEEKLY EXECUTIVE BRIEFING

- Latest InfoSec world news
- Future events and trainings
- We won't spam you!

https://www.inguardians.com/brief