Security Practices in OpenShift

as experienced @ Amadeus

Nenad Bogojević
Amadeus S.A.S.
Diogenes Rettori
Red Hat
2017
Amadeus

In one slide

Provides IT services for travel industry

Operates e-commerce web sites, payment processing, b2b services in travel

Using OpenShift 3 since 2 years
  • In own datacenters, in public clouds
Why security

And not the one like in picture

_Protecting assets
  • computing capacity, data

_Personal information
  • General Data Protection Regulation (GDPR)

_e-Commerce & payment processing
  • PCI/DSS
How?
To be better than the one like in picture

OpenShift & Containers
- Lot of things are changing
- Old rules may not be applicable
- Risks are still out there
OpenShift & Containers

- Lot of things are changing
- Old rules may not be applicable
- Risks are still out there

EVERYONE ON BOARD
OpenShift Architecture

In one slide
Preparing infrastructure
And security

Use OpenStack on our hardware
Or public cloud providers

_Pre-constructed VM images_
- mirrored repositories & registries
- scanned using OpenSCAP

_Network design_
- Where are DMZ and layered protection?
- OpenStack – security groups

_Access control (bastion server)_

_Upgrade policy_
- Rebuild vs rolling
- Bi-weekly/monthly
OpenShift Security Architecture

Different kind of network zones
OpenShift
Let’s login!
Let’s login!

oc login -u system:admin
Let’s login!
OpenShift Security Introduction

Membership

The role “edit” was granted to “jane.smith”.

Users (4)

<table>
<thead>
<tr>
<th>Name</th>
<th>Roles</th>
<th>Add another role</th>
</tr>
</thead>
<tbody>
<tr>
<td>thomson.dupond</td>
<td>operator</td>
<td></td>
</tr>
<tr>
<td>nenad (you)</td>
<td>admin</td>
<td></td>
</tr>
<tr>
<td>jane.smith</td>
<td>edit</td>
<td></td>
</tr>
<tr>
<td>tompson.dupont</td>
<td>view</td>
<td></td>
</tr>
</tbody>
</table>

Add another role

- Select a role
- Add
OpenShift Secrets
Decoupling sensitive information from applications

Way of managing & distributing sensitive information

• keys, certificates, passwords, usernames

Separate sensitive information management from application pods

• Secured delivery to nodes (TLS)
• Only present in memory on openshift nodes
• Centralized management
• Easy access from application
  • Environment variables
  • Volumes
Using Secrets
Security as code

apiVersion: v1
kind: Pod
metadata:
  name: use-secret-pod
spec:
  containers:
  - name: secret-test-container
    image: myapp
    env:
      - name: SECRET_USERNAME
        valueFrom:
          secretKeyRef:
            name: top-secret
            key: username
  restartPolicy: Always

apiVersion: v1
kind: Secret
metadata:
  name: top-secret
data:
  username: bmVuYWQ=
  password: aWtuZXd5b3V3b3VsZHRyeXRoaXM=
OpenShift Secrets – „Less Great Things“
Handbrake for certification

_STORED IN (ALMOST) CLEAR

- in etcd on masters
- on tmp storage on nodes
- accessible through API

_How about vaults?
Some solutions
It’s not show-stopper

You already have big issue if someone compromised your infrastructure

Encrypt disks

Store in vault, with decryption service
  • Side-car or init containers
  • Security as a service

Compensating controls
OpenShift Audit Log

OpenShift provides log of activities that have affected system by individual users, administrators, or other components of the system.

Activate on master `/etc/origin/master/master-config.yaml`

```
auditConfig:
  enabled: true
```

```
AUDIT: id="5c3b8227-4af9-4322-8a71-542231c3887b" ip="127.0.0.1"
method="GET" user="nenad" as="<self>" namespace="someproject"
uri="/api/v1/namespaces/someproject/secrets"
AUDIT: id="5c3b8227-4af9-4322-8a71-542231c3887b" response="401"
```
auditd introduction

- `auditd` rules:
  ```
  -a always,exit -S <syscall> -w <filename>
  ```

- Audit daemon
- System Call Processing
- Kernel
- User
- Task
- Exit
- Exclude

- Audit log
- rsyslog
- Alerting system
auditd rules for masters

Monitoring etcd

_OpenShift master - know if someone plays with etcd

-a always,exit -F arch=b64 -S creat -S open -S openat 
-S open_by_handle_at -S truncate -S ftruncate 
-F dir=/var/lib/etcd 
-k openshift_etcd
..and on nodes

Monitoring secret

_Secrets mounted as tmpfs inside /var/lib/opesnift.

_When new secret is mounted add it to auditd rules
  • When new secret is unmounted remove it from auditd rules

_All monitorable secrets must have certain string in name
  • (e.g. secret~example)

_If you open or close secrets often, it may generate a lot of messages

```
findmnt --list --noheadings --types tmpfs --poll --output ACTION,TARGET |
grep secret~example |
awk '{$1 == "mount" { print $2 }}' |
xargs -L 1 -i auditctl --a always,exit -F arch=b64 -S creat -S open -S openat -S open_by_handle_at -S truncate -S ftruncate -F dir={} -k openshift_secret
```

```
findmnt --list --noheadings --types tmpfs --poll --output ACTION,TARGET |
grep secret~example |
awk '{$1 == "unmount" { print $2 }}' |
xargs -L 1 -i auditctl --d always,exit -F arch=64 -S creat -S open -S openat -S open_by_handle_at -S truncate -S ftruncate -F dir={} -k openshift_secret
```
More use of auditd

With the help of openscap
Secure communication inside or outside your cluster

Service annotated with
service.alpha.openshift.io/serving-cert-secret-name=name

Certificate automatically generated and provided as a secret to pod

Clients can rely on automatically mounted CA
/var/run/secrets/kubernetes.io/serviceaccount/service-ca.crt
Containers
We want to empower developer

Let’s be agile!
  • Run as root
    • Privileged containers – hostpath
    • port < 1000
  • Running old containers
    • FROM httpd:2.4.12
  • There’s this cool blackhat/jboss container on docker hub, let’s pull it
Containers & Developers

- We want to empower developer
- Let’s be agile!
  - Run as root
    - Privileged containers – hostpath
    - port < 1000
  - Running old containers
    - FROM httpd:2.4.12
  - There’s this cool blackhat/jboss container on docker hub, let’s pull it

When will they learn!
Support arbitrary user ids

• Use root group
  
  chown -R someuser:root /app && chmod -R g+rwX /app

Your application needs to listen on port 80?

• Can’t you change it?

Use SCC (Security Context Constraint)

• privileged containers, host paths, user id, FS Groups, selinux, capabilities

seccomp if you want to restrict even more
Image control

Secured source

- All images come from internal registry
- Using RHEL as base images
  - RedHat repository mirrored into internal
- Other images must be built internally from source code
- No automatic access to docker hub from build machines
- Production access it’s own repository with only validated images
Old images and security vulnerabilities

_image-inspector_

Can we run security scan on image before it runs?
- image-inspector
- oscap-docker

Run OpenSCAP on a docker image and serve result

docker run -ti --rm --privileged -p 8080:8080
-v /var/run/docker.sock:/var/run/docker.sock
openshift/image-inspector --image=some-application:20
--path=/tmp/image-content --serve 0.0.0.0:8080 --scan-type=openscap

_used during build process_
Guiding thoughts

Platform can be secured from container vulnerabilities
- containers do bring risk, but it can be managed

Platform will not solve application vulnerabilities
- but it can help
- true multitenancy is complex

Start with the principle of least access
- grant new capabilities to applications only when needed
What we miss
This might be roadmap

Encryption of Secrets!

Network policies – internal and egress

Generic/pluggable image-inspector?

More fine-grained RBAC.
Thank you!