A DevOps State of Mind: Continuous Security with DevSecOps + Containers

Chris Van Tuin
Chief Technologist, NA West / Silicon Valley
cvantuin@redhat.com
SECURITY BREACH: BILLION DATA RECORDS

http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/
WHAT IS THE GREATEST SECURITY RISK?

- 36% - Employees not taking proper security measures
- 32% - Outside breach
- 14% - Unpatched or unpatchable
- 11% - Internal attack by an employee
- 4% - Shadow IT
- 3% - Bring your own device/mobile

Source: Techvalidate/Red Hat
“Only the paranoid survive”

- Andy Grove, 1996
SECURITY MUST EVOLVE & KEEP UP
HYBRID CLOUD ENVIRONMENTS

ANY COMBINATION, WHETHER TRADITIONAL OR CONTAINERIZED

LEGACY APPS
(1,000+)

BARE METAL

VIRTUAL

PRIVATE CLOUD

PUBLIC CLOUD

PRODUCTION

DEV/TEST
DISTRIBUTED APPLICATIONS

ON-PREMISE
- BARE METAL
- VIRTUAL
- PRIVATE CLOUD

OFF-PREMISE
- PUBLIC CLOUD
SECURITY IS AN AFTERTHOUGHT

“Patch?
The servers are behind the firewall.”

- Anonymous (far too many to name), 2005 - …
DEVSECOOPS

Culture

Process

Technology

End to End Security

Culture:
- DEV
- QA
- OPS

Process:
- App dev environment
- APP
- CI/CD
- APP
- Monitor
- Manage
- Dev feedback loop
- IaaS
- Orchestration
- Source Control Management
- Collaboration
- Build and Artifact Management
- Testing
- Frameworks

Technology:
- Linux + Containers
- Open Source
DEVSECOPS

Reduce Risks, Lower Costs, Speed Delivery, Speed Reaction

Security Automation

Process Optimization

Continuous Security Improvement
CONTAINERS
CONTAINERS AT SCALE
MORE THAN CONTAINERS…

- Scheduling
- Lifecycle & health
- Discovery
- Monitoring
- Scaling
- Persistence
- Aggregation
- Security
DEVSECOPS
End to End Security
Web
replicas=2, role=web

Database
replicas=1, role=db

ORCHESTRATION
Deployment, Declarative

Controller Manager & Data Store (etcd)
ORCHESTRATION
Schedule + Provision Pods (Compute/Storage/Network)

Web
replicas=2,
role=web
ReplicaSet

Pods
role=web
role=web

Nodes

Internet

Image Registry
ORCHESTRATION
Schedule + Provision Pods (Compute/Storage/Network)

**Web**
- replicas=2
- role=web
  - ReplicaSet

**Database**
- replicas=1
- role=db
  - StatefulSet

Pods
- role=web
- role=db

Nodes
- role=web

Image Registry

Internet
ORCHESTRATION

Service (Load Balancer)

- Internet
- **Controller Manager & Data Store (etcd)**

Services

- **Web**
  - replicas=2, role=web
- **Database**
  - replicas=1, role=db

Pods

- role=web
- role=db

Nodes

- role=web
HEALTH CHECK

Monitoring & Logging

Services

Pods
role=web
role=db

Nodes

db

Web
replicas=2,
role=web

Database
replicas=1,
role=db

role=web

role=db
HEALTH CHECK

Services

Pods
role=web
role=db

Nodes

Controller Manager & Data Store (etcd)

Internet

Web
replicas=2,
role=web

Database
replicas=1,
role=db

role=web
AUTO-SCALE

Controller Manager & Data Store (etcd)

Pods
- role=web
- role=db

Nodes
- role=web
- role=db

Services
- Web
  - replicas=3
  - role=web
- Database
  - replicas=1
  - role=db

Internet

50% CPU
CONTAINER SECURITY
SECURING CONTAINERS

- Builds
- Images
- Registry
- CI/CD
- Container host

- Network isolation
- Storage
- API & Platform access
- Monitoring & Logging
- Federated clusters
CONTAINER BUILDS
- Are there known vulnerabilities in the application layer?
- Are the runtime and OS layers up to date?
- How frequently will the container be updated and how will I know when it’s updated?
CONTAINER BUILDS

Build

FROM fedora:1.0
CMD echo “Hello”

Build file

Best Practices

• Treat as a Blueprint
• Specify a user, defaults to root
• Don’t login to build/configure
• Version control build file
• Be explicit with versions, not latest
• Each Run creates a new layer
A CONVERGED SOFTWARE SUPPLY CHAIN
CONTAINER IMAGE SECURITY
64% of official images in Docker Hub contain high priority security vulnerabilities

examples:
- ShellShock (bash)
- Heartbleed (OpenSSL)
- Poodle (OpenSSL)

SECURITY IMPLICATIONS

What’s inside matters…
TREAT CONTAINERS AS IMMUTABLE

- code
- config
- data

Container image

Kubernetes configmaps
secrets

Traditional data services,
Kubernetes persistent volumes
CONTAINER REGISTRY SECURITY
PRIVATE REGISTRY

Image

Description: These are images in the test project, more description data goes here.
Source: http://project.example.com/test
Author: Stef Walter <stefw@redhat.com>
Built: 3 days ago
Digest: sha256:91e54dfb11794fad694460162bf0c0b0a4fa710cfa3f60979c177
Tags: project/test:latest, project/test:0.5

$ sudo docker pull 172.30.0.0/project/test:latest

Configuration

# /bin/registry "/etc/docker/registry/config.yml"

Run as: root
Working Directory: /
Stop with: SIGTERM
Architecture: amd64

Ports: 5000 (TCP)
Volumes: /var/lib/registry

Metadata

Image Layers

Labels: INSTALL docker run -v /srv/registry:/var/lib/registry --privileged IMAGE /install
Annotations: openshift.blah: Blah blah blah
Docker Version: 1.7.4
IMAGE SIGNING
Validate what images and version are running

Image Manifest Digest → Encrypted With → Private Key → Generates → Signature Claim

Signature Claim → Decrypted With → Public Key → Generates → Image Manifest Digest
CONTINUOUS INTEGRATION WITH CONTAINERS
CONTINUOUS INTEGRATION + SECURITY

Diagram showing the continuous integration and security process, with roles such as Developer, SCM, Ops, Artifact Repository, QA Manager, Release Manager, and JENKINS involved in the process.
CONTINUOUS INTEGRATION WITH SECURITY SCAN
CONTINUOUS DELIVERY WITH CONTAINERS
CONTINUOUS DELIVERY WITH CONTAINERS

With Container based model, the artifact the passes across environments is a Container Image that includes application and its dependent libraries.
CONTINUOUS DELIVERY + SECURITY
CONTINUOUS DELIVERY: DEPLOYMENT STRATEGIES
CONTINUOUS DELIVERY DEPLOYMENT STRATEGIES

DEPLOYMENT STRATEGIES

- Recreate
- Rolling updates
- Blue / Green deployment
Recreate
RECREATE WITH DOWNTIME

Version 1.2

Tests / CI

Version 1

Version 1

Version 1
RECREATE WITH DOWNTIME

Tests / CI

Version 1.2
RECREATE WITH DOWNTIME

**Use Case**
- Non-mission critical services

**Cons**
- Downtime

**Pros**
- Simple, clean
- No Schema incompatibilities
- No API versioning

Version 1.2

Version 1.2

Version 1.2
Rolling Updates
ROLLING UPDATES with ZERO DOWNTIME

Tests / CI

Version 1.2

Version 1

Version 1

Version 1
Deploy new version and wait until it’s ready…

Health Check: readiness probe e.g. tcp, http, script
Each container/pod is updated one by one

50%

Version 1  V1  V1.2  Version 1.2
Each container/pod is updated one by one

Use Case
- Horizontally scaled
- Backward compatible API/data
- Microservices

Cons
- Require backward compatible APIs/data
- Resource overhead

Pros
- Zero downtime
- Reduced risk, gradual rollout w/health checks
- Ready for rollback
Blue / Green Deployment
BLUE / GREEN DEPLOYMENT

Route

Version 1

BLUE
BLUE / GREEN DEPLOYMENT

Version 1

BLUE

Version 1.2

GREEN
BLUE / GREEN DEPLOYMENT

Version 1

Version 1.2

Tests / CI

BLUE

GREEN
BLUE / GREEN DEPLOYMENT

Route

Version 1

Version 1.2

BLUE

GREEN
BLUE / GREEN DEPLOYMENT

Use Case
• Self-contained microservices (data)

Cons
• Resource overhead
• Data synchronization

Pros
• Low risk, never change production
• No downtime
• Production like testing
• Rollback

Route

Rollback

Version 1

Version 1.2

BLUE

GREEN
RAPID INNOVATION & EXPERIMENTATION
"only about 1/3 of ideas improve the metrics they were designed to improve."

Ronny Kohavi, Microsoft (Amazon)
CONTINUOUS FEEDBACK LOOP
A/B TESTING USING CANARY DEPLOYMENTS
CANARY DEPLOYMENTS

100%

Route

Version 1

25% Conversion Rate

Version 1.2

Tests / CI

?! Conversion Rate
CANARY DEPLOYMENTS

50%  Route  50%

Version 1  Version 1.2

25% Conversion Rate  30% Conversion Rate
CANARY DEPLOYMENTS

Route

Version 1
25% Conversion Rate

Version 1.2
30% Conversion Rate

100%
CANARY DEPLOYMENTS

100%

Route

Rollback

Version 1
25% Conversion Rate

Version 1.2
20% Conversion Rate
CONTAINER HOST SECURITY
CONTAINERS ARE LINUX

Containers

Container CLI

SYSTEMD

Docker Image

Unit File

Cgroups

Namespaces

SELinux

Drivers

seccomp

Read Only mounts

Kernel

Hardware (Intel, AMD) or Virtual Machine
CGROUPS - RESOURCE ISOLATION
NAMESPACES - PROCESS ISOLATION

Namespaces

Mount  UTC  IPC  PID  Network
## SECCOMP - DROPPING PRIVILEGES

<table>
<thead>
<tr>
<th>PRIVILEGE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAP_SETPCAP</td>
<td>Modify process capabilities</td>
</tr>
<tr>
<td>CAP_SYS_MODULE</td>
<td>Insert/Remove kernel modules</td>
</tr>
<tr>
<td>CAP_SYS_RAWIO</td>
<td>Modify Kernel Memory</td>
</tr>
<tr>
<td>CAP_SYS_PACCT</td>
<td>Configure process accounting</td>
</tr>
<tr>
<td>CAP_SYS_NICE</td>
<td>Modify Priority of processes</td>
</tr>
<tr>
<td>CAP_SYS_RESOURCE</td>
<td>Override Resource Limits</td>
</tr>
<tr>
<td>CAP_SYS_TIME</td>
<td>Modify the system clock</td>
</tr>
<tr>
<td>CAP_SYS_TTY_CONFIG</td>
<td>Configure tty devices</td>
</tr>
<tr>
<td>CAP_AUDIT_WRITE</td>
<td>Write the audit log</td>
</tr>
<tr>
<td>CAP_AUDIT_CONTROL</td>
<td>Configure Audit Subsystem</td>
</tr>
<tr>
<td>CAP_MAC_OVERRIDE</td>
<td>Ignore Kernel MAC Policy</td>
</tr>
<tr>
<td>CAP_MAC_ADMIN</td>
<td>Configure MAC Configuration</td>
</tr>
<tr>
<td>CAP_SYSLOG</td>
<td>Modify Kernel printk behaviour</td>
</tr>
<tr>
<td><strong>CAP_NET_ADMIN</strong></td>
<td>Configure the network:</td>
</tr>
<tr>
<td></td>
<td>- Setting the hostname/domainname</td>
</tr>
<tr>
<td></td>
<td>- mount(), unmount()</td>
</tr>
<tr>
<td></td>
<td>- nfsservctl</td>
</tr>
<tr>
<td><strong>CAP_SYS_ADMIN</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- ....</td>
</tr>
</tbody>
</table>
READ ONLY MOUNTS

/sys
/proc/sys
/proc/sysrg-trigger
/proc/irq
/proc/bus
CONTAINER HOST SECURITY

Best Practices
- Don’t run as root
- Limit SSH Access
- Use namespaces
- Define resource quotas
- Enable logging
- Apply Security Errata
- Apply Security Context and seccomp filters

SECURING CONTAINERS

- Builds
- Images
- Registry
- CI/CD
- Container host

- Network isolation
- Storage
- API & Platform access
- Monitoring & Logging
- Federated clusters
NETWORK ISOLATION
NETWORK ISOLATION

**Multi-Environment**

**Multi-Tenant**

Network Namespace provides resource isolation
NETWORK POLICY

description:
all pods in namespace ‘project-a’ allow traffic from any other pods in the same namespace.”
STORAGE SECURITY
STORAGE SECURITY

Local Storage Quota

Security Context Constraints

Node’s /var/lib/origin

mySQL
mongoDB
postgres
API & PLATFORM ACCESS
API & PLATFORM ACCESS

Authentication via OAuth tokens and SSL certificate

Authorization via Policy Engine checks User/Group Defined Roles
Aggregate platform and application log access via Kibana + Elasticsearch
MONITORING

Historical CPU and Memory usage
FEDERATION
FEDERATED CLUSTERS
Roles & access management (in-dev)
MICROSERVICES
SERVICE MESH

Monitoring & Metrics
- prometheus (logs)
- grafana (visual)

Access Control & usage policies
- mixr (policy decisions)

Traffic routing
- pilot
- circuit breaker
- a/b testing
- traffic mirroring

Encryption & Auth
- citadel
- service 2 service
- user auth

Fault injections
- envoy
corner cases: abort & delays
DEVSECOPS METRICS

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compliance Score</td>
<td></td>
</tr>
<tr>
<td>Deployment Frequency</td>
<td></td>
</tr>
<tr>
<td>Lead Time</td>
<td></td>
</tr>
<tr>
<td>Deployment Failure Rate</td>
<td>404</td>
</tr>
<tr>
<td>Mean Time to Recover</td>
<td>99.999</td>
</tr>
<tr>
<td>Service Availability</td>
<td></td>
</tr>
</tbody>
</table>
THANK YOU

linkedin: Chris Van Tuin
email: cvantuin@redhat.com
twitter: @chrissvantuin