Common Vulnerabilities and Exposures in containers

Quentin Machu

@Quentin__M  |  quentin.machu@coreos.com  |  coreos.com
CoreOS is Running the World’s Containers

OPEN SOURCE

90+ Projects on GitHub, 1,000+ Contributors
CoreOS.com - @coreoslinux - github/coreos

ENTERPRISE

Secure solutions, support plans, training + more
sales@coreos.com - tectonic.com - quay.io

CoreOS Linux, rkt, etcd, kubernetes, TECTONIC, QUAY
### Repositories

<table>
<thead>
<tr>
<th>REPOSITORY NAME</th>
<th>LAST MODIFIED</th>
<th>ACTIVITY</th>
<th>STAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>coreos / etcd</td>
<td>Last Friday at 8:56 PM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>coreos / flannel</td>
<td>09/23/2016</td>
<td></td>
<td></td>
</tr>
<tr>
<td>coreos / dnsmasq</td>
<td>08/19/2016</td>
<td></td>
<td></td>
</tr>
<tr>
<td>coreos / hyperkube</td>
<td>Last Tuesday at 11:36 PM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>coreos / clair</td>
<td>10/25/2016</td>
<td></td>
<td>⭐</td>
</tr>
<tr>
<td>coreos / registry</td>
<td>10/31/2016</td>
<td></td>
<td></td>
</tr>
<tr>
<td>coreos / registry-build-worker</td>
<td>Last Tuesday at 8:42 PM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>coreos / awscli</td>
<td>03/15/2016</td>
<td></td>
<td>⭐</td>
</tr>
<tr>
<td>quay / quay-builder-qemu-coreos</td>
<td>09/12/2016</td>
<td></td>
<td></td>
</tr>
<tr>
<td>coreos / dex</td>
<td>10/28/2016</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stackanetes / stackanetes-mariadb</td>
<td>10/20/2016</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stackanetes / stackanetes-rabbitmq</td>
<td>10/20/2016</td>
<td></td>
<td></td>
</tr>
<tr>
<td>coreos / klocksmith</td>
<td>10/14/2016</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Why is container security a problem?
Traditional deployment
Who’s responsible for the dependencies?
Containerized deployment

Ops

kernel
systemd
rkt | docker
sshd

Devs

java          appA
python        appB
nginx openssl appC
Containerized deployment

- kernel
- systemd
- rkt | docker
- sshd

Ops

Devs

- java
- appA
- python
- appB
- java
- appC
Containerized deployment

- Ops
  - kernel
  - systemd
  - rkt | docker
  - sshd

- Devs
  - java
  - appA
  - python
  - appB
  - java 8.0
  - appC
In practice...
Containers are in fact opaque
What is actually running?

Ops
- kernel
- systemd
- rkt | docker
- sshd

Devs
- ?
- appA
- ?
- appB
- ?
- appC

FROM golang:onbuild
EXPOSE 80 443 8080 6060 6161
Why is this an issue?
CVE-2015-0235
aka
GHOST

“GHOST is a buffer overflow bug affecting the gethostbyname() and gethostbyname2() function calls in the glibc library. This vulnerability allows a remote attacker that is able to make an application call to either of these functions to execute arbitrary code.”
CVE-2014-0160
aka
Heartbleed

“The TLS and DTLS implementations in OpenSSL do not properly handle Heartbeat Extension packets, which allows remote attackers to obtain sensitive information from process memory via crafted packets that trigger a buffer over-read.”
80K Vulnerabilities
How can we improve the situation?
Inspect our container images!
How do we do so?
A little more context
A little more context

Quay contains millions of images that could have vulnerabilities
A little more context

Quay contains **millions** of images that could have vulnerabilities

Running container images can be highly **insecure**
A little more context

Quay contains **millions** of images that could have vulnerabilities

Running container images can be highly **insecure**

Analyzing these images should be as **fast** as possible
And finally...
Open source is **key**
Open source project for the analysis of vulnerabilities in appc and docker containers.

github.com/coreos/clair
Clair is an open source system for analyzing container images for vulnerabilities.
Clair is an **open source** system for analyzing container images for vulnerabilities.

Clair makes use of **static analysis** for fast, reproducible results.
Clair is an open source system for analyzing container images for vulnerabilities.

Clair makes use of static analysis for fast, reproducible results.

Clair is extensible to allow for easy addition of new analyzers, reporters, image formats, etc.
So let’s see it!
go NewDemo("quay.io").Run()
How?
Getting Vulnerabilities

- Periodically, Pull from external databases
Getting Vulnerabilities

- Periodically, pull from external databases
- As first-party as possible
  - Close to the maintainers
  - Distribution-specific trackers
    - Knowledge about implementation details, backports, etc …
Getting Vulnerabilities

- Periodically, Pull from external databases

- As first-party as possible
  - Close to the maintainers
  - Distribution-specific trackers
    - Knowledge about implementation details, backports, etc …
Analyzing an image
Analyzing an image

- A client sends an image to Clair’s API
  - Layer by Layer (Reusability)
Analyzing an image

- A client sends an image to Clair’s API
  - Layer by Layer (Reusability)
- **Worker** passes layers to detectors
Analyzing an image

- A client sends an image to Clair’s API
  - Layer by Layer (Reusability)

- **Worker** passes layers to **detectors**
  - **Data**: Static content, depending on the **image format**
Analyzing an image

- A client sends an image to Clair’s API
  - Layer by Layer (Reusability)

- **Worker** passes layers to **detectors**
  - **Data**: Static content, depending on the **image format**
  - **Features**: Any elements that define the image
Analyzing an image

- A client sends an image to Clair’s API
  - Layer by Layer (Reusability)

- **Worker** passes layers to **detectors**
  - **Data**: Static content, depending on the **image format**
  - **Features**: Any elements that define the image
  - **Namespaces**: Context between features and vulnerabilities
A client sends an image to Clair’s API
- Layer by Layer (Reusability)

Worker passes layers to detectors
- Data: **Static content**, depending on the image format
- Features: **Any elements** that define the image
- Namespaces: **Context** between features and vulnerabilities
Taking action!

- Which vulnerabilities?
- What can be **fixed**?
- How?
Getting notified

Over 15 new vulnerabilities / day (+ updates!)
Over 15 new vulnerabilities / day (+ updates!)

- Redo analysis of millions of images
Over 15 new vulnerabilities / day (+ updates!)

- Redo analysis of millions of images
- Exploit immutable nature of images
  - What we knew about the static content of an image is still true
  - Match new vulnerability data with existing knowledge about container images
Over 15 new vulnerabilities / day (+ updates!)

- Redo analysis of millions of images
- Exploit immutable nature of images
  - What we knew about the static content of an image is still true
  - Match new vulnerability data with existing knowledge about container images
"Notification": {
  "Old": {
    "Vulnerability": {
      "Name": "CVE-SPOOKY",
      "NamespaceName": "debian:8",
      "Description": "Spooky CVE is spooky",
      "Severity": "High",
      "FixedIn": [
        { "Name": "grep", "NamespaceName": "debian:8", "Version": "None" }
      ]
    },
    "LayersIntroducingVulnerability": [
      "3b59c795b34670618fbca4d2c7a27c5ecec156812c9e2c90d3f4be1916b12d",
      "523ef1d23f22195488575f52a39c729c76a8c5630c9a194139cb246fb212da6"
    ]
  },
  "New": {
    "Vulnerability": {
      "Name": "CVE-SPOOKY",
      "NamespaceName": "debian:8",
      "Description": "Spooky CVE is spooky (but at least there’s a fix now!)",
      "Severity": "High",
      "FixedIn": ["grep", "debian:8", "2.25"
      ]
    },
    "LayersIntroducingVulnerability": [
      "523ef1d23f22195488575f52a39c729c76a8c5630c9a194139cb246fb212da6"
    ]
  }
}
"Old": {
  "Vulnerability": {
    "Name": "CVE-SPOOKY",
    "NamespaceName": "debian:8",
    "Description": "Spooky CVE is spooky",
    "Severity": "High",
    "FixedIn": [
      { "Name": "grep", "NamespaceName": "debian:8", "Version": "None" }
    ]
  },
  "LayersIntroducingVulnerability": [
    "3b59c795b34670618fbcace4dac7a27c5ecec156812c9e2c90d3f4be1916b12d",
    "523ef1d23f222195488575f52a39c729c76a8c5630c9a194139cb246fb212da6"
  ]
}
"New": {
  "Vulnerability": {
    "Name": "CVE-SPOOKY",
    "NamespaceName": "debian:8",
    "Description": "Spooky CVE is spooky (but at least there’s a fix now!)",
    "Severity": "High",
    "FixedIn": [
      { "Name": "grep", "NamespaceName": "debian:8", "Version": "2.25" }
    ],
  },
  "LayersIntroducingVulnerability": [
    "523ef1d23f222195488575f52a39c729c76a8c5630c9a194139cb246fb212da6"
  ]
}
Customizing it

Components in **Red** can be extended by anyone
type FeaturesDetector interface {
    GetRequiredFiles() []string
    Detect(map[string][]byte) ([]database.FeatureVersion, error)
}

type NamespaceDetector interface {
    GetRequiredFiles() []string
    Detect(map[string][]byte) *database.Namespace
}

type DataDetector interface {
    Supported(path string, format string) bool
    Detect(layerReader io.ReadCloser, toExtract []string, maxFileSize int64) (data map[string][]byte, err error)
}

type Fetcher interface {
    FetchUpdate(database.Datastore) (FetcherResponse, error)
    Clean()
}

type Notifier interface {
    Configure(config.NotifierConfig) (bool, error)
    Send(database.VulnerabilityNotification) error
}
type Datastore interface {
    ListNamespaces() ([]Namespace, error)

    InsertLayer(Layer) error
    FindLayer(name string, withFeatures, withVulnerabilities bool) (Layer, error)
    DeleteLayer(name string) error

    ListVulnerabilities(namespaceName string, limit int, page int) ([]Vulnerability, int, error)
    InsertVulnerabilities(vulnerabilities []Vulnerability, createNotification bool) error
    FindVulnerability(namespaceName, name string) (Vulnerability, error)
    DeleteVulnerability(namespaceName, name string) error
    InsertVulnerabilityFixes(vulnerabilityNamespace, vulnerabilityName string, fixes []FeatureVersion) error
    DeleteVulnerabilityFix(vulnerabilityNamespace, vulnerabilityName, featureName string) error

    GetAvailableNotification(renotifyInterval time.Duration) (VulnerabilityNotification, error)
    GetNotification(name string, limit int, page PageNumber) (VulnerabilityNotification, PageNumber, error)
    SetNotificationNotified(name string) error
    DeleteNotification(name string) error

    InsertKeyValue(key, value string) error
    GetKeyValue(key string) (string, error)

    Lock(name string, owner string, duration time.Duration, renew bool) (bool, time.Time)
    Unlock(name, owner string)
    FindLock(name string) (string, time.Time, error)

    Ping() bool
    Close()
}
That’s it?
No
A lightweight operating system designed for automation, scalability, reliability and security.
A lightweight operating system designed for automation, scalability, reliability and security.

- No package manager
- Read-only /usr
- Run an app? Run a container!
Updating CoreOS Linux
Updating CoreOS Linux

CoreOS Container Linux

VM / Bare Metal

A

B

OS Update
Updating CoreOS Linux

VM / Bare Metal
Linux Kernel Local Root Escalation

Released fix same day for CoreOS Linux
Majority of hosts running the patch within 24 hours

Oct. 20th 2016
Kubernetes Client Cert Validation Failure

Released fix 10 days after disclosure
Manual upgrade for users

Oct. 20th 2016
Updating Kubernetes

Kubernetes

Deployment

A

B

Application Container
Updating Kubernetes
Updating Kubernetes

Kubernetes

Deployment

A
B
A container engine for Linux designed to be *composable*, *secure*, and *built on standards*. 
rkt

A container engine for Linux designed to be **composable, secure, and built on standards**.

- No daemon
- Image signatures required by default
  - GPG detached
A container engine for Linux designed to be composable, secure, and built on standards.

- **stage1-fly**
  - host OS
  - rkt
    - chroot
    - user-app1

- **stage1-coreos (default)**
  - host OS
  - rkt
    - systemd-nspawn
      - systemd
        - user-app1

- **stage1-kvm**
  - host OS
  - rkt
    - lkvm
      - kernel
      - systemd
      - chroot
      - user-app1
Distributed Trusted Computing

- Verify integrity of the OS release
- Customer key embedded in the firmware
- Verify configuration state
- Verify images with trusted keys
- Only attested machines are allowed to join

Clusters:
- Kubernetes
  - Only attested machines are allowed to join

Containers:
- rkt
  - Verify images with trusted keys
  - Verify configuration state

OS:
- CoreOS Linux
  - Verify integrity of the OS release

Hardware:
- Firmware & TPM
  - Customer key embedded in the firmware

Tamper-evident audit log (TPM)
December 12 & 13 2016 - New York City

tectonic.com/summit - @TectonicSummit
Thank you!

github.com/coreos/clair

Quentin Machu

@Quentin__M  |  quentin.machu@coreos.com  |  coreos.com

We’re hiring in all departments!  Email: careers@coreos.com  Positions: coreos.com/ careers