

Linux Containers

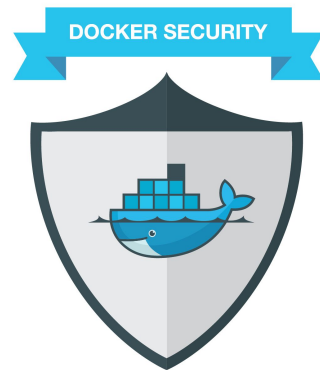
Everything you need to know about Containers Security

Track Containers

José Manuel Ortega

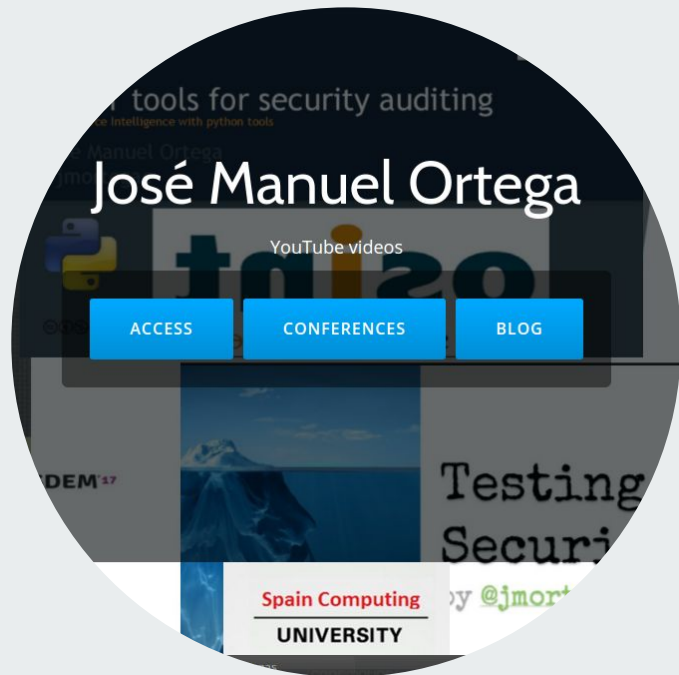


FOSDEM 2018





@jmortegac

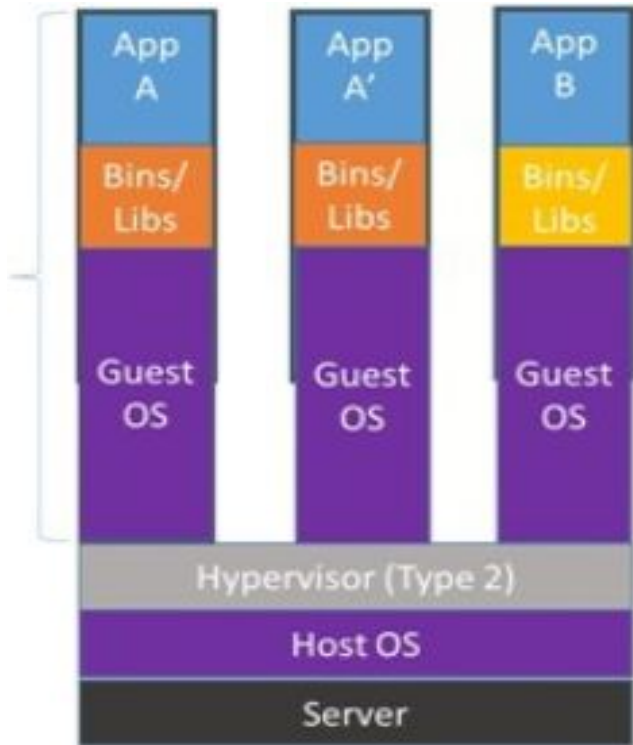




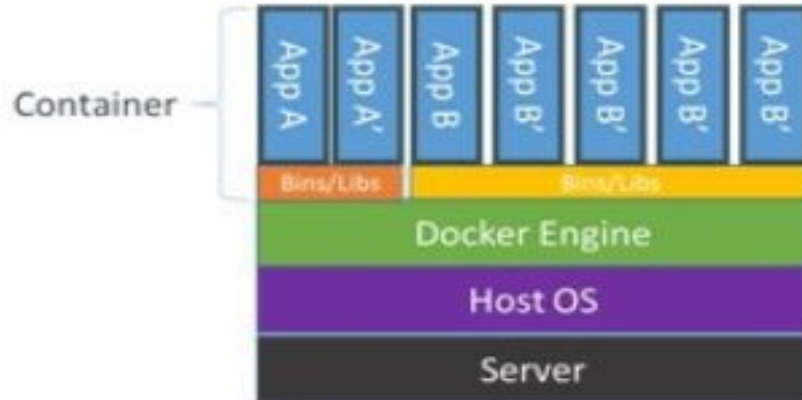
Agenda

- **Introduction to containers security**
- **Linux Containers(LXC)**
- **Docker Security**
- **Security pipeline && Container threats**
- **Tools for auditing container images**

Virtualization vs containers



Containers are isolated, but share OS and, where appropriate, bins/libraries



Virtualization vs containers



different kernels/OS

emulation of devices

many fs caches

limits per machine

legacy consolidation

single kernel

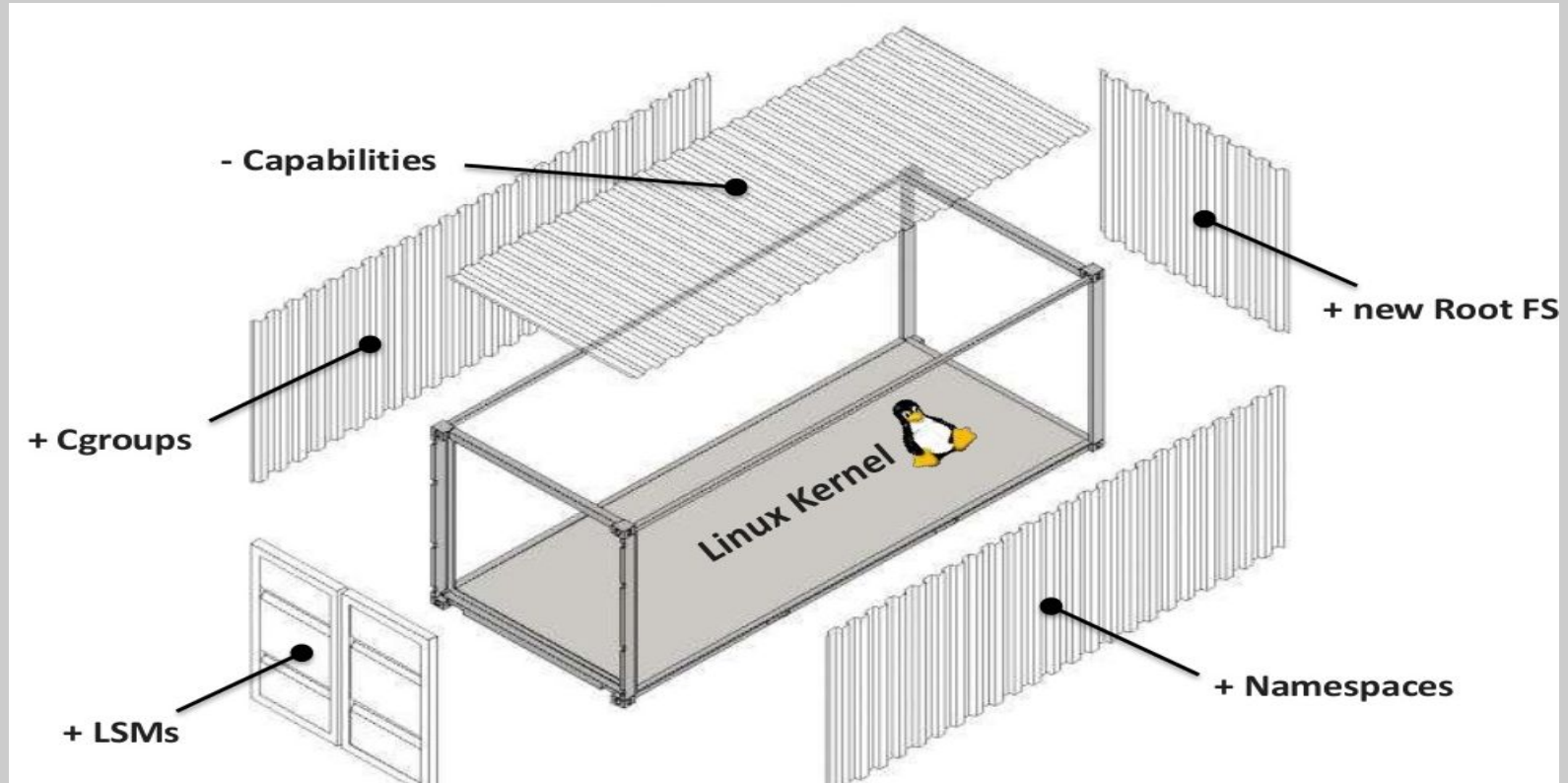
acl+syscall

single fs cache

limits per process

service deployment

Security mechanisms




Namespaces



- Provides an isolated view of the system where processes cannot see other processes in other containers
- Each container also gets its own network stack.
- A container doesn't get privileged access to the sockets or interfaces of another container.

Cgroups & capabilities



- **Cgroups:** kernel feature that limits and isolates the resource usage (CPU, memory, network) of a collection of processes.
- **Linux Capabilities:** divides the privileges of root into distinct units and smaller groups of privileges

Linux Containers(LXC)



LXC



- **Lightweight virtual machines**
- **VMs without the hypervisor**
- **Kernel namespaces**
- **Apparmor and SELinux profiles**
- **Seccomp policies**
- **Kernel capabilities and Control groups**

LXC

- Start single process in container
 - `lxc-execute -n container -- /bin/bash`
- Whole operating system
 - Mounting filesystems, etc from config file
 - Application is `/bin/init`
 - `lxc-start -n container`
 - `lxc-console -n container`
 - `lxc-stop -n container`

LXC:limit resources

- Cores
 - `lxc.cgroup.cpuset.cpus=1,2,3`
- CPU share
 - `lxc.cgroup.cpu.shares=1024 # default`
- Memory usage (!Debian)
 - `lxc.cgroup.memory.limit_in_bytes = 256M`
 - `lxc.cgroup.memory.memsw.limit_in_bytes = 1G`
- Disk (blkio)
 - Disk space – standard LVM, quota...
 - `echo 100 > /cgroup/disk1/blkio.weight # XXX < 1000 !`
 - `echo "3:0 1048576" > /cgroup/disk1/blkio.throttle.read_bps_device`

LXC:limit resources

```
root@tryit-talented:~# free -m
```

	total	used	free	shared	buff/cache	available
Mem:	256	116	69	177	69	69
Swap:	0	0	0			

```
root@tryit-talented:~# lxc exec first -- free -m
```

	total	used	free	shared	buff/cache	available
Mem:	256	78	176	177	1	176
Swap:	0	0	0			

```
root@tryit-talented:~# lxc config set first limits.memory 128MB
```

```
root@tryit-talented:~# lxc exec first -- free -m
```

	total	used	free	shared	buff/cache	available
Mem:	128	77	48	177	1	48
Swap:	0	0	0			

Docker





- Kernel Feature
- Groups of Processes
- Control Resource Allocation
 - CPU, CPU Sets
 - Memory
 - Disk
 - Block I/O

- The real magic behind containers
- It creates barriers between processes
- Different Namespaces
 - PID Namespace
 - Net Namespace
 - IPC Namespace
 - MNT Namespace
- Linux Kernel Namespace introduced between kernel 2.6.15 – 2.6.26

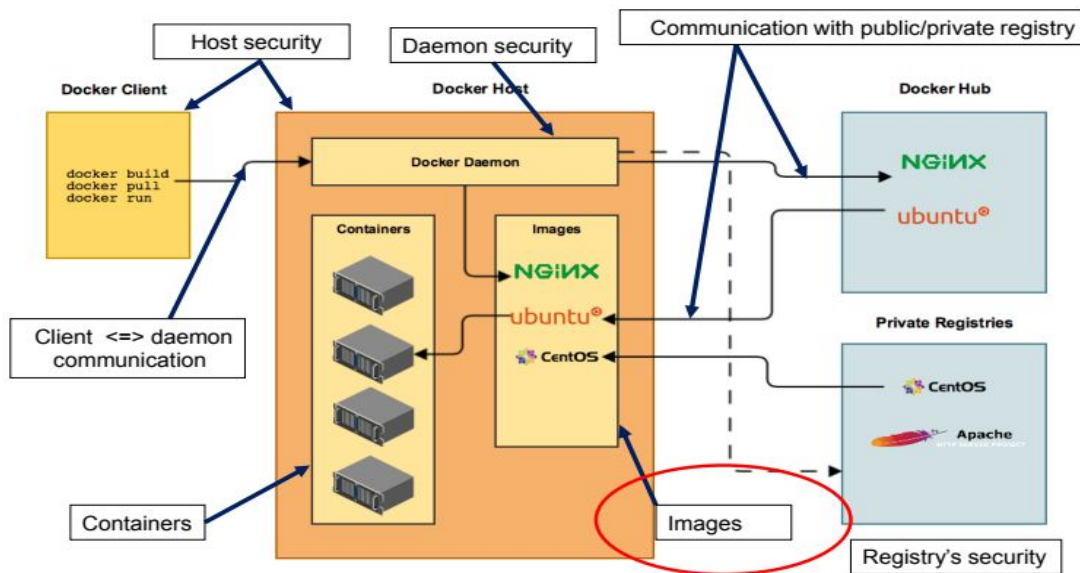
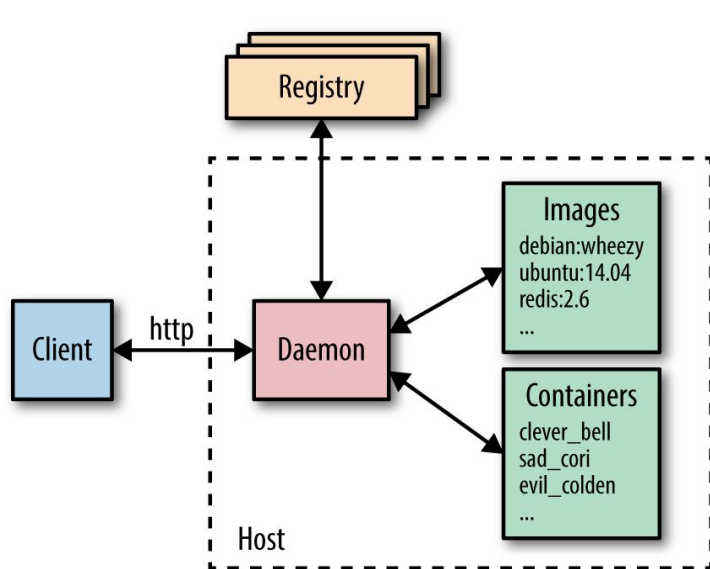
- Not a File System
- Not a VHD
- Basically a tar file
- Has a Hierarchy
 - Arbitrary Depth
- Fits into Docker Registry

docker run

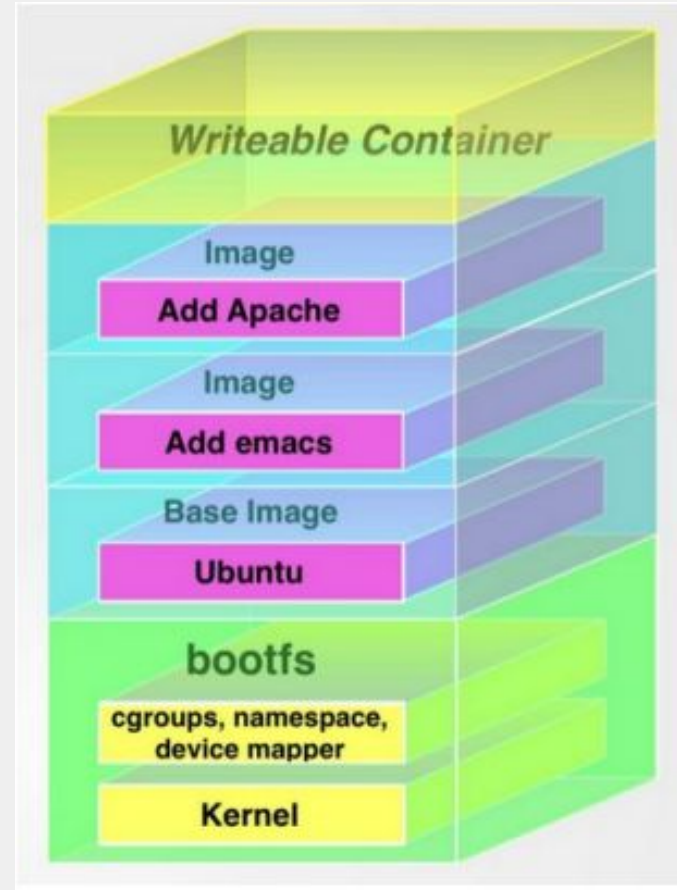
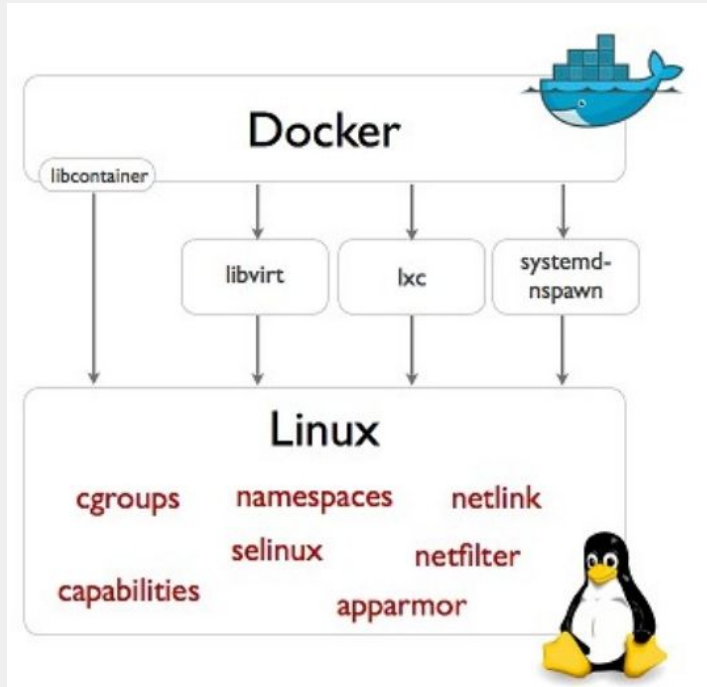


lxc-start

Container pipeline



Docker images



Docker security



- Isolation via kernel namespaces
- Additional layer of security Apparmor, SELinux, GRSEC
- Each container gets its own network stack
- Control groups for resources limiting
- Other interesting features...

Docker Content Trust



- We can verify the integrity of the image
- Checksum validation when pulling image from docker hub
- Pulling by digest to enforce consistent

```
$ docker pull debian@sha256:a25306f3850e1bd44541976aa7b5fd0a29be
```

```
$ export DOCKER_CONTENT_TRUST=1
[node1] (local) root@192.168.0.28 ~
$ docker pull python
Using default tag: latest
Pull (1 of 1): python:latest@sha256:59d8481f4b2d21f2ac6623e986b4e91fa704
112df3e7d9dddbbe7315d4a153ef5
sha256:59d8481f4b2d21f2ac6623e986b4e91fa704112df3e7d9dddbbe7315d4a153ef5:
  Pulling from library/python
85b1f47fba49: Pull complete
ba6bd283713a: Pull complete
817c8cd48a09: Pull complete
47cc0ed96dc3: Pull complete
4a36819a59dc: Pull complete
db9a0221399f: Pull complete
7a511a7689b6: Pull complete
1223757f6914: Pull complete
Digest: sha256:59d8481f4b2d21f2ac6623e986b4e91fa704112df3e7d9dddbbe7315d4
a153ef5
Status: Downloaded newer image for python@sha256:59d8481f4b2d21f2ac6623e
986b4e91fa704112df3e7d9dddbbe7315d4a153ef5
Tagging python@sha256:59d8481f4b2d21f2ac6623e986b4e91fa704112df3e7d9dddb
```

Usage: docker trust COMMAND

Manage trust on Docker images (experimental)

Options:

Management Commands:

key	Manage keys for signing Docker images (experimental)
signer	Manage entities who can sign Docker images (experimental)

Commands:

inspect	Return low-level information about keys and signatures
revoke	Remove trust for an image
sign	Sign an image
view	Display detailed information about keys and signatures

Docker Capabilites



- A capability is a unix action a user can perform
- Goal is to restrict “capabilities”
- Privileged process = all the capabilities!
- Unprivileged process = check individual user capabilities
- Example Capabilities:
 - CAP_CHOWN
 - CAP_NET_RAW

[node1] (local) root@192.168.0.13 ~

```
$ docker run --rm -it python sh -c 'apk add -U libcap; capsh --print'
```

Unable to find image 'python:latest' locally

latest: Pulling from library/python

85b1f47fba49: Pull complete

ba6bd283713a: Pull complete

817c8cd48a09: Pull complete

47cc0ed96dc3: Pull complete

4a36819a59dc: Pull complete

db9a0221399f: Pull complete

7a511a7689b6: Pull complete

1223757f6914: Pull complete

Digest: sha256:59d8481f4b2d21f2ac6623e986b4e91fa704112df3e7d9dddbe7315d4a153ef5

Status: Downloaded newer image for python:latest

sh: 1: apk: not found

Current: = cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sys_chroot,cap_mknod,cap_audit_write,cap_setfcap+eip

Bounding set =cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_s

```
$ docker run -it --cap-drop NET_RAW python sh
Unable to find image 'python:latest' locally
latest: Pulling from library/python
85b1f47fba49: Pull complete
ba6bd283713a: Pull complete
817c8cd48a09: Pull complete
47cc0ed96dc3: Pull complete
4a36819a59dc: Pull complete
db9a0221399f: Pull complete
7a511a7689b6: Pull complete
1223757f6914: Pull complete
Digest: sha256:59d8481f4b2d21f2ac6623e986b4e91fa704112df3e7d9dddbe7315d4a
153ef5
Status: Downloaded newer image for python:latest
# ping 8.8.8.8
ping: Lacking privilege for raw socket.
```


Containers security is about limiting and controlling the **attack surface** on the kernel.

Least privilege principle



- Do not run processes in a container as **root** to avoid root access from attackers.
- Enable User-namespaces
- Run filesystems as read-only so that attackers can not overwrite data or save malicious scripts to file.
- Cut down the kernel calls that a container can make to reduce the potential attack surface.

Read only containers & volumes

```
latest: Pulling from library/python
85b1f47fba49: Pull complete
ba6bd283713a: Pull complete
817c8cd48a09: Pull complete
47cc0ed96dc3: Pull complete
4a36819a59dc: Pull complete
db9a0221399f: Pull complete
7a511a7689b6: Pull complete
1223757f6914: Pull complete
Digest: sha256:59d8481f4b2d21f2ac6623e986b4e91fa704112df3e7d9dddbe7315d4
a153ef5
Status: Downloaded newer image for python:latest
# touch file
touch: cannot touch 'file': Read-only file system
# exit
[node1] (local) root@192.168.0.28 ~
$ docker run -it -v $(pwd)/secrets:/secrets:ro python sh
# touch /secrets/file
touch: cannot touch '/secrets/file': Read-only file system
```

Seccomp



- **Restricts system calls based on a policy**
- **Block/limit things like:**
 - **Kernel manipulation (init_module, finit_module, delete_module)**
 - **Executing mount options**
 - **Change permissions**
 - **Change owner and groups**

```
$ docker run --rm -it --security-opt seccomp:policy.json alpine sh
```

```
Unable to find image 'alpine:latest' locally
```

```
latest: Pulling from library/alpine
```

```
b56ae66c2937: Pull complete
```

```
Digest: sha256:d6bfc3baf615dc9618209a8d607ba2a8103d9c8
```

```
Status: Downloaded newer image for alpine:latest
```

```
/ # mkdir newdir
```

```
mkdir: can't create directory 'newdir': Operation not
```

```
/ # chown root:root bin
```

```
chown: bin: Operation not permitted
```

```
/ # chmod +x /etc/resolv.conf
```

```
chmod: /etc/resolv.conf: Operation not permitted
```

```
1  
2 "defaultAction": "SCMP_ACT_ALLOW",  
3 "syscalls": [  
4 {  
5 "name": "mkdir",  
6 "action": "SCMP_ACT_ERRNO"  
7 },  
8 {  
9 "name": "chmod",  
10 "action": "SCMP_ACT_ERRNO"  
11 },  
12 {  
13 "name": "chown",  
14 "action": "SCMP_ACT_ERRNO"  
15 }  
16 ]  
17
```

Docker bench security



- Auditing docker environment and containers
- Open-source tool for running automated tests
- Inspired by the CIS Docker 1.11 benchmark
- Runs against containers currently running on same host
- Checks for AppArmor, read-only volumes, etc...

<https://github.com/docker/docker-bench-security>

Docker bench security



- The host configuration
- The Docker daemon configuration
- The Docker daemon configuration files
- Container images and build files
- **Container runtime**
- Docker security operations

```
[INFO] 5 - Container Runtime
[PASS] 5.1 - Ensure AppArmor Profile is Enabled
[WARN] 5.2 - Ensure SELinux security options are set, if applicable
[WARN] * No SecurityOptions Found: mypython
[PASS] 5.3 - Ensure Linux Kernel Capabilities are restricted within containers
[PASS] 5.4 - Ensure privileged containers are not used
[PASS] 5.5 - Ensure sensitive host system directories are not mounted on containers
[PASS] 5.6 - Ensure ssh is not run within containers
[PASS] 5.7 - Ensure privileged ports are not mapped within containers
[NOTE] 5.8 - Ensure only needed ports are open on the container
[PASS] 5.9 - Ensure the host's network namespace is not shared
[WARN] 5.10 - Ensure memory usage for container is limited
[WARN] * Container running without memory restrictions: mypython
[WARN] 5.11 - Ensure CPU priority is set appropriately on the container
[WARN] * Container running without CPU restrictions: mypython
[WARN] 5.12 - Ensure the container's root filesystem is mounted as read only
[WARN] * Container running with root FS mounted R/W: mypython
[PASS] 5.13 - Ensure incoming container traffic is binded to a specific host interface
[WARN] 5.14 - Ensure 'on-failure' container restart policy is set to '5'
[WARN] * MaximumRetryCount is not set to 5: mypython
[PASS] 5.15 - Ensure the host's process namespace is not shared
```


Lynis



- <https://github.com/CISOfy/lynis-docker>
- Lynis is a Linux, Mac and Unix security auditing and system hardening tool that includes a module to audit Dockerfiles.
- **lynis audit system**
- **lynis audit dockerfile <file>**

[+] Containers

- Docker
 - Docker daemon [RUNNING]
 - Docker info output (warnings) [4]
- Containers
 - Total containers [19]
 - Running containers [1]
 - Unused containers [18]
- File permissions [OK]

[+] Security frameworks

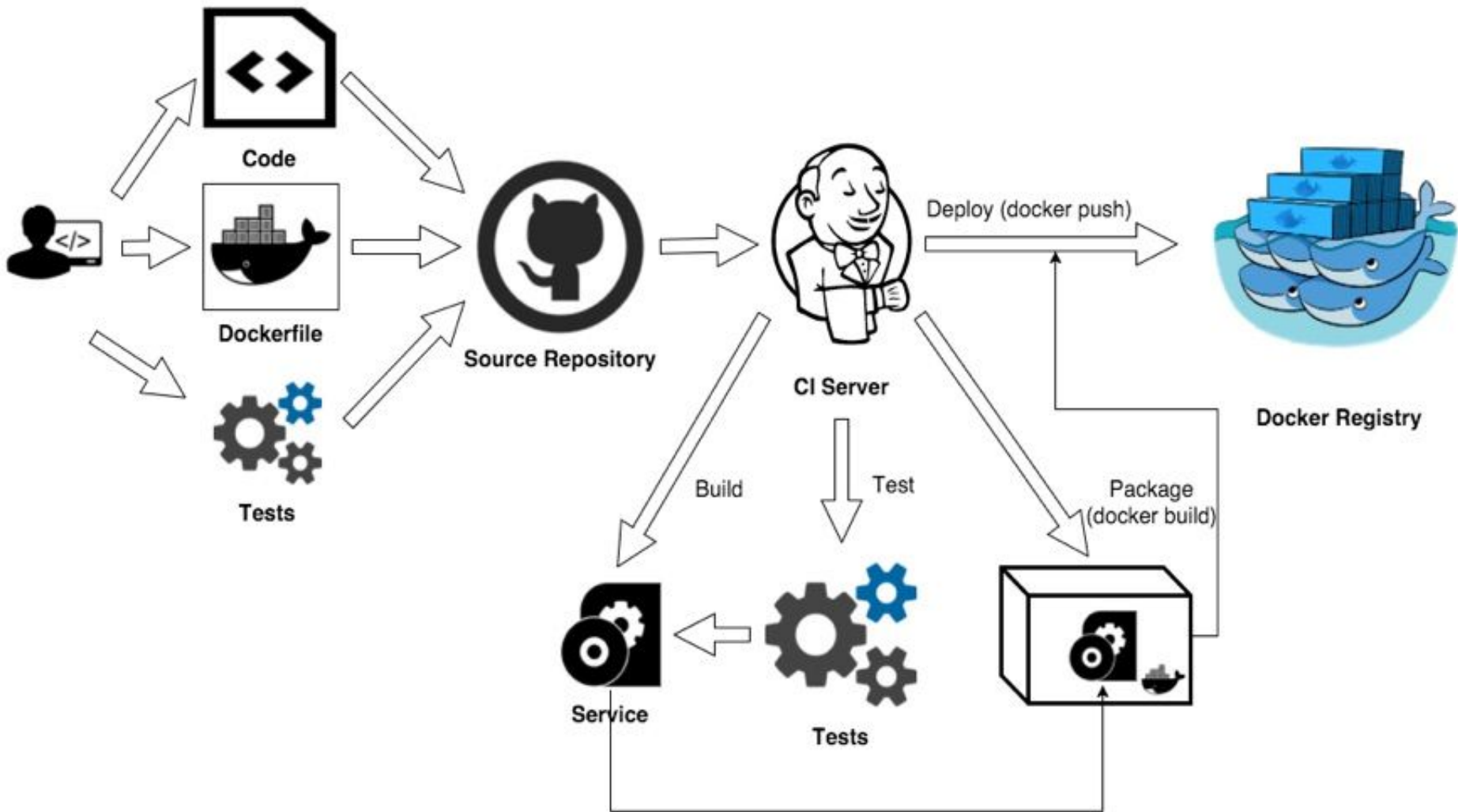
- Checking presence AppArmor [NOT FOUND]
- Checking presence SELinux [NOT FOUND]
- Checking presence grsecurity [NOT FOUND]
- Checking for implemented MAC framework [NONE]

[+] Software: file integrity

- Checking file integrity tools
- Checking presence integrity tool [NOT FOUND]

Security Pipeline







Jenkins

[Download from Github](#)



TeamCity

[Download from Github](#)



Bamboo

[Download from Github](#)



TFS

[Download from Github](#)



Travis CI

[Download from Github](#)



CircleCI

[Download from Github](#)



GitLab CI

[Download from Github](#)



Team Services

[Download from Github](#)

Container threats

- 
- **Kernel Exploits(Dirty Cow exploit)**
 - **Vulnerabilities like the glibc buffer overflow**
 - **SQL injection attacks**
 - **MongoDB and Elasticsearch ransomware attacks**

Remember



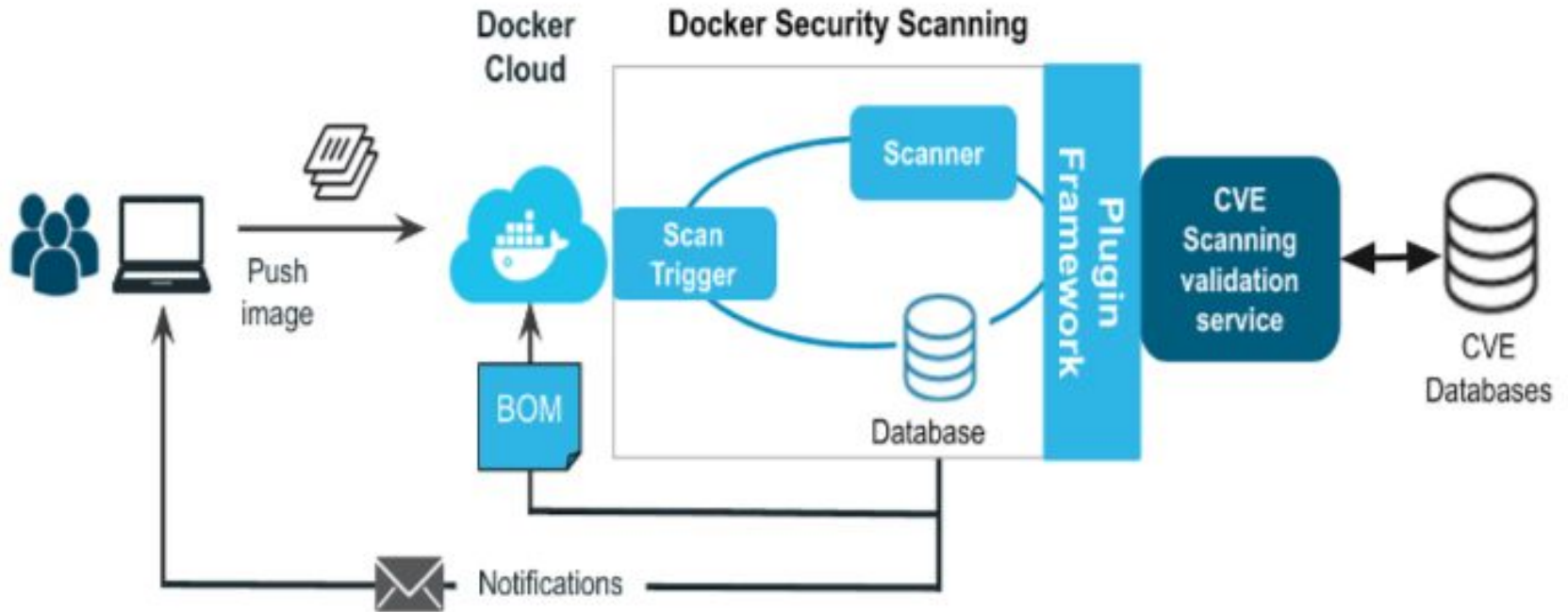
- **Don't run containers as root**
- **Drop all capabilities and enable only needed**
- **Enable user namespaces**
- **Use seccomp for limit syscalls for avoid kernel exploits**
- **Keep the host kernel updated with last patches**
- **Mount volumes with read only**

Audit Container Images



- You can scan your images for known vulnerabilities
- Find known vulnerable binaries
 - Docker Security Scanning
 - Anchore Cloud
 - Dagda
 - Tenable.io Container Security

Docker security scanning



Docker security scanning

24 of 143 components are vulnerable

Scanned 19 hours ago

[Provide Feedback](#)

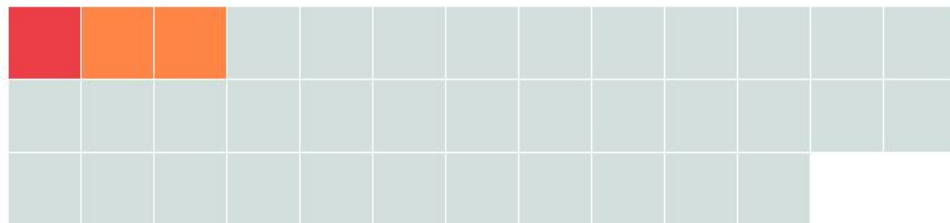
Layers

Components

1 [ADD file:a71e077a4299...03cc487124b1f70 in /](#)

Compressed size: 43.0MB

3 vulnerable components Base Layer



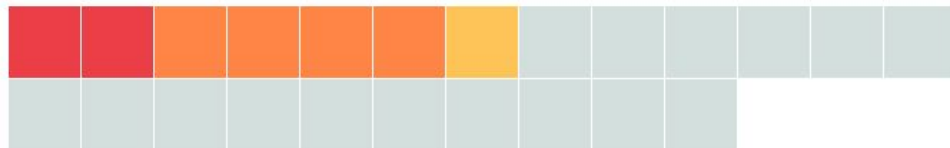
2 [CMD \["bash"\]](#) Compressed size: 0.0

No components in this layer Base Layer

3 [/bin/sh -c apt-get up... /var/lib/apt/lists/*](#)

Compressed size: 10.6MB

7 vulnerable components Base Layer



Layers

1 [ADD file:a71e077a4299...03cc487124b1f70 in /](#)

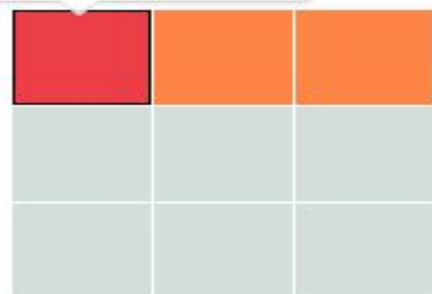
Compressed size: 43.0MB

COMPONENT	VULNERABILITY	SEVERITY
glibc 2.24-11+deb9u1 LGPL:Lgpl License	CVE-2017-15670 CVE-2017-15804 CVE-2017-15671	Critical Critical Major
berkeleydb 5.3.28-12+deb9u1 sleepycat:Copyleft License	CVE-2016-0694 CVE-2016-3418 CVE-2016-0682 CVE-2016-0689 CVE-2016-0692	Major Major Major Major Major
systemd 232-25+deb9u1 LGPL:Lgpl License	CVE-2017-15908	Major
acl 2.2.52-3+b1 LGPL:Lgpl License	No known vulnerabilities	N/A

glibc 2.24-11+deb9u1

2 Critical Vulnerabilities

1 Major Vulnerability



Anchore

anchore

SERVICES ▾

SOLUTIONS ▾

BLOG

OPEN SOURCE

| LOGIN

SIGNUP

Anchore Open Source Engine

An Open Container Certification Platform

 Install Anchore

 Documentation

PUBLIC REPOSITORY

anchore/cli ☆

Last pushed: 6 days ago

Repo Info

Tags

Short Description

Anchore Container Image Scanner

Full Description

Anchore

Anchore is a container inspection and analytics platform to enable operators to deploy containers with confidence. The Anchore toolset in this repository provides the ability to inspect, reason about, and evaluate policy against containers present on the local Docker host.

Docker Pull Command



```
docker pull anchore/cli
```

Owner



anchore

Anchore

CVE ID	Severity	*Total Affected	Vulnerable Package	Fix Available	Fix Images	Rebuild Images	URL
CVE-2017-8804	High	1	multiarch-su	None	c2b44478417f	None	https://security-tracker.debian.org/tracker/CVE-2017-8804
CVE-2017-8804	High	1	libc6-2.24-1	None	c2b44478417f	None	https://security-tracker.debian.org/tracker/CVE-2017-8804

[< Previous Image](#)

Image ID: 79e1dc9af1c1d276a3cae7c1c56290f2101c0aedbf70cbbb322e1a6b16149bd5

Repo/Tag: library/python:latest

library/python

latest

[Overview](#)[Policy](#)[Contents](#)[Security](#)

Image Summary

Repository / Tag:

[library/python](#)

latest

Registry:

[Docker Hub](#)

Image Created:

3 days ago on Sat, 04 Nov 2017 at 23:18 CET

Tag Scanned At:

3 days ago on Sun, 05 Nov 2017 at 00:01 CET

OS / Version:

Debian / 8

Size:

659 MB

Layers:

16

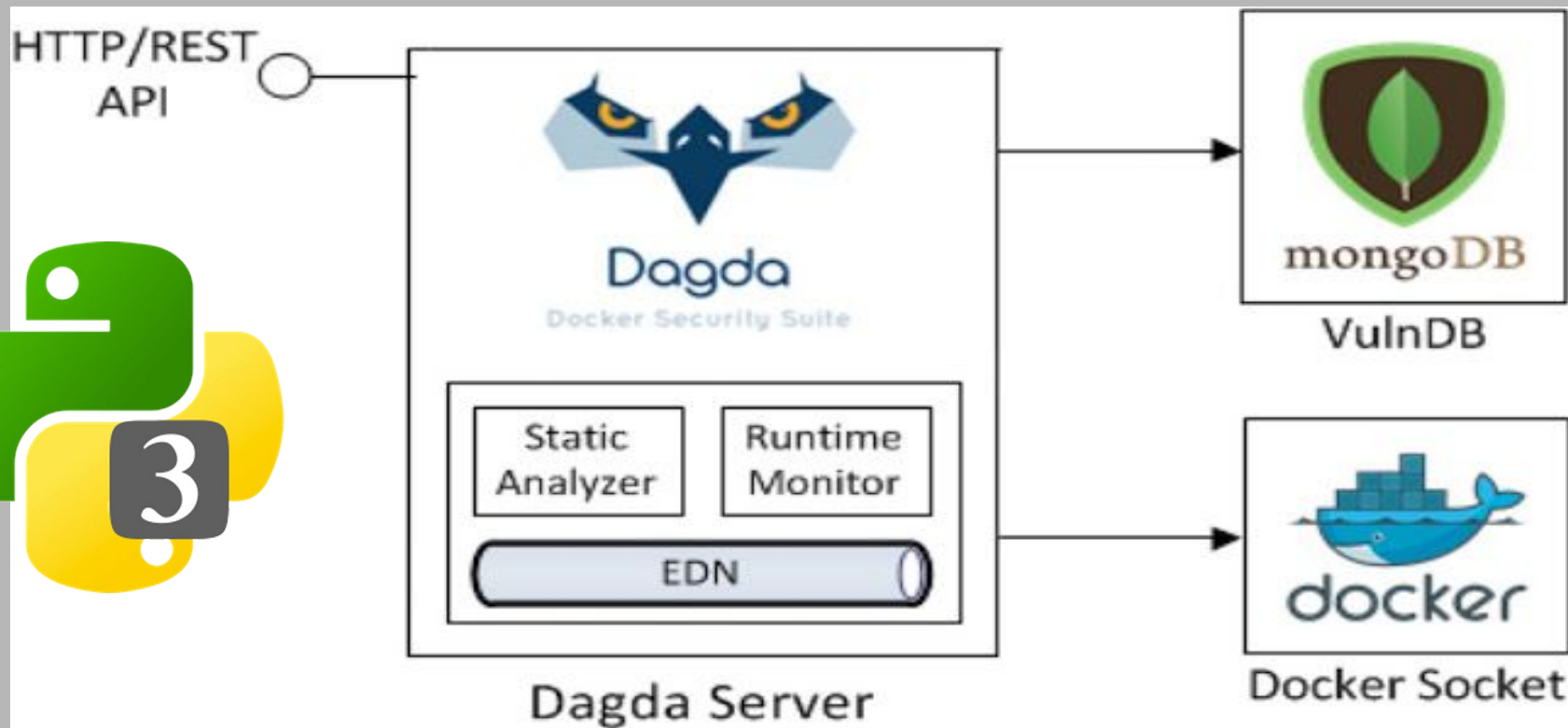
Associated Tags

latest

	Removed
Files	0
Packages	0

3.6-jessie

Dagda



Tenable.io container security



Tenable.io Vulnerability Management

Bring clarity to your security posture through a fresh, asset-based approach that provides maximum coverage of your evolving assets and vulnerabilities in ever-changing environments.

[Learn More.](#)

Free Trial

Buy Now

Tenable.io Container Security

Provide comprehensive visibility into the security posture of container images as they are developed, enabling vulnerability assessment, malware detection, policy enforcement and remediation prior to container deployment. [Learn More.](#)

Free Trial

Buy Now

Tenable.io Web Application Scanning

Gain visibility into the security of web applications with safe vulnerability scanning, complete with high detection rates to ensure you understand the true risks in your web applications.

[Learn More.](#)

Free Trial

Buy Now

Moderate risk items were discovered when inspecting the container image. Tenable.io Container Security discovered 40 unique vulnerabilities. 0 malware was detected. The vulnerabilities should be remediated in accordance to organizational policy.

40 vulnerabilities detected

0 pieces of malware detected

Container Image Type

Docker

Uploaded Date

2017-11-23T19:31:55.650Z

Last Scanned Date

2017-12-02T19:45:37.300Z

Digest

sha256:7344b8fbaf9241c38f3b6d4b2db553c1e44f0f3

Operating System (OS)

LINUX_DEBIAN

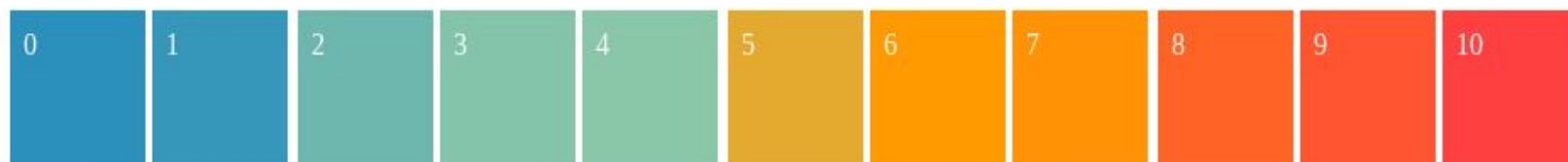
Operating System (OS) Version

8.9

Architecture

AMD64

Risk Scoring Framework Tenable.io Container Security measures the organizational risk using the Tenable.io Container Security Risk Scoring Framework



INFORMATIONAL

LOW RISK

MODERATE RISK

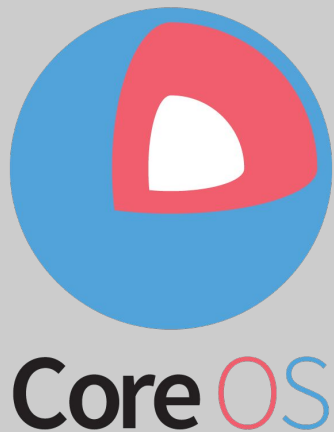
HIGH RISK

Informational results only.
No risks discovered.

Small low risk items discovered, such as
vulnerabilities in dead code.

Moderate risk items discovered, such as
numerous medium severity vulnerabilities.

High severity vulnerabilities or malicious
code discovered.



Csysdig "Containers" View

CPU Used by Container

Count of Processes in Container

Count of Threads in Container

Virtual memory assigned

Resident memory assigned

Total container file I/O in bps

Total container network I/O in bps

Container type (docker, rkt, lxc etc)

Container Identification
(Image, ID, Name)

Filter applied to data

Viewing: Containers For: whole machine

Source: alert-capture-b5eb4fcl-9244-466a-a88b-7b06e5b67290.scap (164849 evts, 8.67s) Filter: container.name != host

CPU	PROCS	THREADS	VRT	RES	FILE	NET	ENGINE	IMAGE	ID	NAME
0.00	0	0	1M	4K	0	0.00	docker	gcr.io/google_containers/paus	2276ceab8b58	k8s_P00_d8dbel6c_kube-proxy-ip-10
0.00	0	0	1M	4K	0	0.00	docker	gcr.io/google_containers/paus	a0afeca66f31	k8s_P00_956305ba_mongo-886875792-
0.00	0	0	1M	4K	0	0.00	docker	gcr.io/google_containers/paus	03c8c044a7dc	k8s_P00_96e7050b_javaapp-29377701
0.00	0	0	3G	218M	694	20.53	docker	d39dc18f6149	k8s_P00_d8dbel6c_sysdig-agent-c21	
0.00	0	0	287M	78M	25K	9.39K	docker	ltagliamonte/counterapp	591135d67903	k8s_javaapp.102b3dcb_javaapp-2748
0.00	0	0	3G	253M	449K	7.54K	docker	wongo	66f24c30196d	k8s_mongo.e19437dd_mongo-80687579
0.00	0	0	8G	6G	41K	44.93	docker	sysdig/agent:latest	1962e05e0707	k8s_sysdig-agent.9a5bctc6_sysdig-
0.00	0	0	735M	33M	99K	00.97	docker	ltagliamonte/demo-mongo-stats	8f8797830756	k8s_mongo-statsd.5aaf19fb_mongo-8
0.00	0	0	3G	229M	2K	61.62	docker	ltagliamonte/recurling	e69a1a716067	k8s_client.2f5844e1_jclient-35658
0.00	0	0	258M	32M	0	5.54K	docker	ltagliamonte/counterapp	4b26a99ba288	k8s_javaapp.5d683f88_javaapp-2937
0.00	0	0	1M	4K	0	0.00	docker	gcr.io/google_containers/hype	061c7fce675c	k8s_kube-proxy.3afec6db_kube-prox
0.00	0	0	1M	4K	0	0.00	docker	gcr.io/google_containers/paus	3b8f0b3550e5	k8s_P00_956305ba_mongo-886875792-
0.00	0	0	1M	4K	0	0.00	docker	gcr.io/google_containers/paus	dad6dcadfd28e	k8s_P00_96e7050b_javaapp-29377701
0.00	0	0	3G	222M	1K	5.82K	docker	ltagliamonte/counterapp	0a948489b27d	k8s_javaapp.5d683f88_javaapp-2937
0.00	0	0	1M	4K	0	0.00	docker	gcr.io/google_containers/paus	8103380ec520	k8s_P00.e1000589_redis-3547843244
0.00	0	0	5G	4G	29K	44.93	docker	ltagliamonte/demo-mongo-stats	6d3d52b05066	k8s_mongo-statsd.ce1719a0_mongo-8
0.00	0	0	1M	4K	0	0.00	docker	gcr.io/google_containers/paus	90fe4d4ed87d	k8s_P00_d8dbel6c_jclient-35650673
0.00	0	0	1M	4K	0	0.00	docker	gcr.io/google_containers/paus	7b4694c30e46	k8s_P00_d8dbel6c_client-129300300
0.00	0	0	1M	4K	0	0.00	docker	gcr.io/google_containers/paus	64c66d1aadff	k8s_P00.96e7050b_javaapp-27485018
0.00	0	0	1M	4K	0	0.00	docker	gcr.io/google_containers/paus	3d11d23aa950	k8s_P00.2225036b_kubernetes-dashb
0.00	0	0	3G	8M	25K	5.25K	docker	redis:2.8.19	7ac5f1d36109	k8s_redis.bc3c3ecf_redis-35478432
0.00	0	0	179M	18M	68K	7.01K	docker	ltagliamonte/recurling	dc6430e42e39	k8s_client.3637a3b6_client-129300
0.00	0	0	49M	31M	611	95.05	docker	gcr.io/google_containers/kube	e26cc225bddc	k8s_kubernetes-dashboard.0041cd97
0.00	0	0	291M	82M	28K	4.03K	docker	wongo	4f8ad1dflc7a	k8s_mongo.550b3782_mongo-88687579

References



- <https://docs.docker.com/engine/security>
- <http://www.oreilly.com/webops-perf/free/files/docker-security.pdf>
- http://container-solutions.com/content/uploads/2015/06/15.06.15_DockerCheatSheet_A2.pdf
- Docker Content Trust
https://docs.docker.com/engine/security/trust/content_trust
- Docker Security Scanning
- <https://docs.docker.com/docker-cloud/builds/image-scan>
- <https://blog.docker.com/2016/04/docker-security>
- <http://softwaretester.info/docker-audit>

O'REILLY®

Docker Security

Using Containers Safely in Production



Adrian Mouat



Community Experience Distilled

Securing Docker

Learn how to secure your Docker environment and keep your environments secure irrespective of the threats out there

Scott Gallagher

[PACKT] open source*
PUBLISHING



Thanks!

Contact:

[@jmortegac](https://twitter.com/jmortegac)

jmortega.github.io

about.me/jmortegac

