Everything you need to know about Containers Security

Track Containers

José Manuel Ortega
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.
<table>
<thead>
<tr>
<th>Virtualization vs Containers</th>
<th>Virtualization vs Containers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Different kernels/OS</td>
<td>Single kernel</td>
</tr>
<tr>
<td>Emulation of devices</td>
<td>ACL+syscall</td>
</tr>
<tr>
<td>Many fs caches</td>
<td>Single fs cache</td>
</tr>
<tr>
<td>Limits per machine</td>
<td>Limits per process</td>
</tr>
<tr>
<td>Legacy consolidation</td>
<td>Service deployment</td>
</tr>
</tbody>
</table>
Security mechanisms

-Capabilities

+Cgroups

+LSMs

+new Root FS

+Namespaces

Linux Kernel
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.cpu.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.memsww.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@sha256:59d8481f4b2d21f2ac6623e986b4e91fa704112df3e7d9dddbe7315d4a153ef5
sha256:59d8481f4b2d21f2ac6623e986b4e91fa704112df3e7d9ddbbe7315d4a153ef5:
  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: sha256:59d8481f4b2d21f2ac6623e986b4e91fa704112df3e7d9ddbbe7315d4a153ef5
Status: Downloaded newer image for python@sha256:59d8481f4b2d21f2ac6623e986b4e91fa704112df3e7d9ddbbe7315d4a153ef5
Tagging python@sha256:59d8481f4b2d21f2ac6623e986b4e91fa704112df3e7d9ddbbe7315d4a153ef5
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 Capabilities

- 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
$ 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: sha256:59d8481f4b2d21f2ac6623e986b4e91fa704112df3e7d9ddbbe7315d4a153ef5
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+ep
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: sha256:59d8481f4b2d21f2ac6623e986b4e91fa704112df3e7d9dddbe7315d4a153ef5
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-namespac
- 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: sha256:59d8481f4b2d21f2ac6623e986b4e91fa704112df3e7d9ddbe7315d4a153ef5
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:d6bfc3baf615dc9618209a8d607ba2a8103d9c
Status: Downloaded newer image for alpine:latest

/ # mkdir newdir
mkdir: can't create directory 'newdir': Operation not permitted

/ # chown root:root bin
chown: bin: Operation not permitted

/ # chmod +x /etc/resolv.conf
chmod: /etc/resolv.conf: Operation not permitted
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
5 - Container Runtime

5.1 - Ensure AppArmor Profile is Enabled

5.2 - Ensure SELinux security options are set, if applicable
   * No SecurityOptions Found: mypython

5.3 - Ensure Linux Kernel Capabilities are restricted within containers

5.4 - Ensure privileged containers are not used

5.5 - Ensure sensitive host system directories are not mounted on containers

5.6 - Ensure ssh is not run within containers

5.7 - Ensure privileged ports are not mapped within containers

5.8 - Ensure only needed ports are open on the container

5.9 - Ensure the host's network namespace is not shared

5.10 - Ensure memory usage for container is limited
   * Container running without memory restrictions: mypython

5.11 - Ensure CPU priority is set appropriately on the container
   * Container running without CPU restrictions: mypython

5.12 - Ensure the container's root filesystem is mounted as read only
   * Container running with root FS mounted R/W: mypython

5.13 - Ensure incoming container traffic is binded to a specific host interface

5.14 - Ensure 'on-failure' container restart policy is set to '5'
   * MaximumRetryCount is not set to 5: mypython

5.15 - Ensure the host's process namespace is not shared
Lynis

- [https://github.com/CISOfy/lynis-docker](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 [NOT FOUND]
- Checking presence integrity tool [NOT FOUND]
Security Pipeline
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

<table>
<thead>
<tr>
<th>Layers</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ADD file:a71e077a4299...03cc487124b1f70 in /</td>
<td>3 vulnerable components</td>
</tr>
<tr>
<td>Compressed size: 43.0MB</td>
<td>Base Layer</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>2 CMD [&quot;bash&quot;] Compressed size: 0.0</td>
<td></td>
</tr>
<tr>
<td>No components in this layer</td>
<td>Base Layer</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>3 /bin/sh -c apt-get up... /var/lib/apt/lists/*</td>
<td>7 vulnerable components</td>
</tr>
<tr>
<td>Compressed size: 10.6MB</td>
<td>Base Layer</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>VULNERABILITY</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>glibc 2.24-11+deb9u1</td>
<td>CVE-2017-15670</td>
</tr>
<tr>
<td></td>
<td>CVE-2017-15804</td>
</tr>
<tr>
<td></td>
<td>CVE-2017-15671</td>
</tr>
<tr>
<td>berkeleydb 5.3.28-12+deb9u1</td>
<td>CVE-2016-0694</td>
</tr>
<tr>
<td></td>
<td>CVE-2016-3418</td>
</tr>
<tr>
<td></td>
<td>CVE-2016-0682</td>
</tr>
<tr>
<td></td>
<td>CVE-2016-0689</td>
</tr>
<tr>
<td></td>
<td>CVE-2016-0692</td>
</tr>
<tr>
<td>systemd 232-25+deb9u1</td>
<td>CVE-2017-15908</td>
</tr>
<tr>
<td>acl 2.2.52-3+b1</td>
<td>No known</td>
</tr>
<tr>
<td></td>
<td>vulnerabilities</td>
</tr>
</tbody>
</table>
Anchore Open Source Engine
An Open Container Certification Platform

_install Anchore | Documentation_
Anchore Container Image Scanner

docker pull anchore/cli

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.
<table>
<thead>
<tr>
<th>CVE ID</th>
<th>Severity</th>
<th><strong>Total Affected</strong></th>
<th>Vulnerable Package</th>
<th>Fix Available</th>
<th>Fix Images</th>
<th>Rebuild Images</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2017-880</td>
<td>High</td>
<td>1</td>
<td>multiarch-su</td>
<td>None</td>
<td>c2b44478417f</td>
<td>None</td>
<td>https</td>
</tr>
<tr>
<td>4 security-debi</td>
<td></td>
<td></td>
<td>pport-2.24-1</td>
<td></td>
<td></td>
<td>(jmortegac/1)</td>
<td>://se</td>
</tr>
<tr>
<td>E-2017-track</td>
<td></td>
<td></td>
<td>1+deb9u1</td>
<td></td>
<td></td>
<td>linux_tweet_a</td>
<td>track</td>
</tr>
<tr>
<td>CVE-2017-880</td>
<td>High</td>
<td>1</td>
<td>libc6-2.24-1</td>
<td>None</td>
<td>c2b44478417f</td>
<td>None</td>
<td>https</td>
</tr>
</tbody>
</table>
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
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.

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.

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.
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.

Risk Scoring Framework


- **0** = INFORMATIONAL
  - Informational results only. No risks discovered.
- **1**
- **2**
- **3**
- **4**
- **5** = LOW RISK
  - Small low risk items discovered, such as vulnerabilities in dead code.
- **6** = MODERATE RISK
  - Moderate risk items discovered, such as numerous medium severity vulnerabilities.
- **7**
- **8**
- **9**
- **10** = HIGH RISK
  - High severity vulnerabilities or malicious code discovered.
### Csystsdig “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)**
- **Filter applied to data**
- **Container Identification (Image, ID, Name)**

#### Viewing: Containers (For whole machine)

<table>
<thead>
<tr>
<th>Container Name</th>
<th>CPU</th>
<th>MEM</th>
<th>DISK</th>
<th>SWAP</th>
<th>VIRT</th>
<th>RES</th>
<th>FILE</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>docker</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rkt</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lxc</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>kubernetes</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Source:** netcaptection @beatrice-korner-066a-088b-7b0865679290 
**Filter:** container.name != host

---

**Legend:**
- `CPU`: CPU Used by Container
- `MEM`: Count of Processes in Container
- `DISK`: Count of Threads in Container
- `SWAP`: Virtual memory assigned
- `VIRT`: Resident memory assigned
- `FILE`: Total container file I/O in bps
- `TIME`: Total container network I/O in bps
- `CONTAINER TYPE`: Container type (docker, rkt, lxc etc)

---

**Filter applied to data:**
- `FILTERS`: container.name != host

---

**Help:**
- `Help`:
- `Views`:
- `Filter`:
- `Echo`:
- `Stats`:
- `Legend`:
- `Action`:
- `Sort`:
- `Toggle`:
- `CTRL`:
- `Search`:
- `Pause`:

---

1/24 (4.2%)
References

- https://docs.docker.com/engine/security
- 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
Thanks!

Contact:
@jmortegac
jmortega.github.io
about.me/jmortegac