

DOCKER SECURITY: USING CONTAINERS SAFELY

Charlas CiberSeguridad UTFSM

Monday 12th June, 2017

Maximiliano Osorio

Mail: <mosorio@inf.utfsm.cl>

Universidad Técnica Federico Santa María

WHO AM I?

- Maximiliano Osorio
- M.Sc. Computer Science Student, UTFSM
- Informatics Engineer, UTFSM
- Red Hat Certified Engineer
- Red Hat Certified System Administrator
- Unemployed :)
- <https://mosorio.me>

DOCKER CONTAINERS

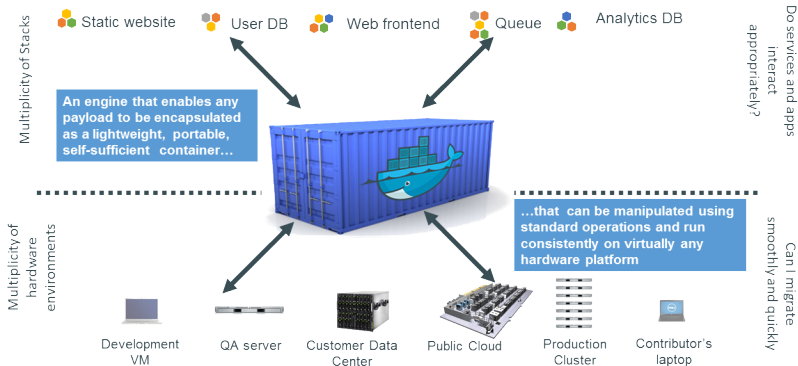


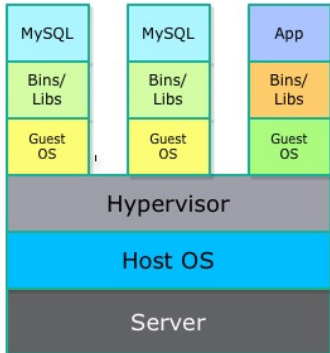
Image: [docker.com](https://www.docker.com)

- Docker is an open platform for developing, shipping, and running applications.
- Docker allows you to package an application with all of its dependencies into a standardized unit for software development.

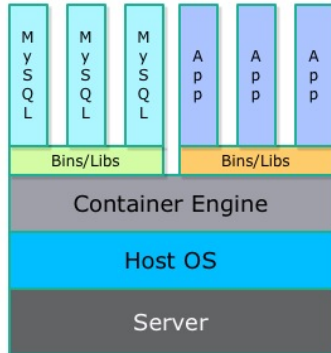
DOCKER BENEFITS

- Fast (deployment, migration, restarts)
- Secure
- Lightweight (save disk & CPU)
- Open Source
- Portable software
- Microservices and integrations (APIs)
- Simplify DevOps
- Version control capabilities

Virtual Machines



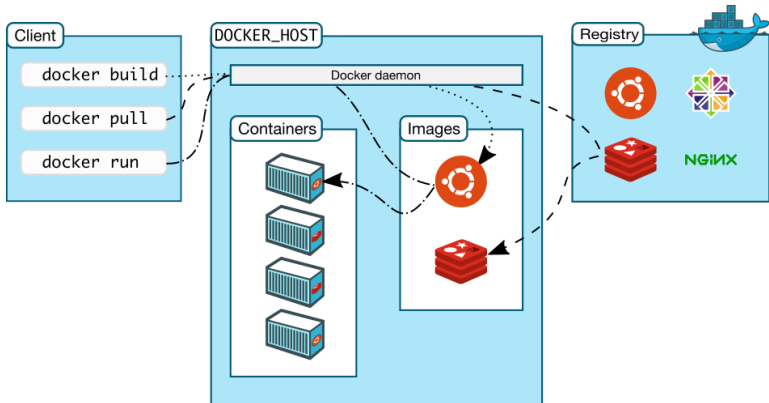
Containers



- Provide processes with their own view of the system.
- Multiple namespaces: pid, net, mnt, uts, ipc, user

Docker: Namespaces provide the first and most straightforward form of isolation: processes running within a container cannot see, and even less affect, processes running in another container, or in the host system.

DOCKER REGISTRY



<http://cloud.51cto.com/>

CONTAINERS DON'T CONTAIN

WHY DON'T CONTAINERS CONTAIN?

Dan Walsh ¹ meant that not all resources that a container has access to are namespaces.

- UID (by default) If a user is root inside a container and breaks out of the container, that user will be root on the host.
- The kernel keyring
- The kernel itself and any kernel modules
- The system time `SYS_TIME`
- Namespaces have vulnerabilities!

¹Mr. SELinux

- The linux kernel is responsible for managing the uid and gid space
- the uid and gid that created the process are examined by the kernel to determine if it has enough privileges to modify the file

The username isn't used here, the uid is used!

```
1 [mosorio@host ~]$ docker run -d ubuntu:
  latest sleep infinity
```

```
1 root          30636    0.0    0.0    4372    364 ?
          Ss      20:43    0:00          \_ sleep
infinity
```

Does root inside the container == root outside the container?

- Yes, because, as I mentioned, there's a single kernel and a single, shared pool of uids and gids.

```
1 [mosorio@host ~]$ docker run -it -u $(id -u $  
   USER):$(id -g $GROUP) centos:7 bash  
2 bash-4.2$ id  
3 uid=1002 gid=1002 groups=1002
```

As of Docker 1.10, you can enable user namespacing by starting the kernel with the `-userns-remap`.

- WARNING: Using Docker volumes becomes more complex as the changed UIDs affect access privileges.

```
1 root@container:/# id
2 uid=0(root) gid=0(root) groups=0(root)
```

```
1 root@host:/# ps -eaf | grep bash
2 231072 4080 4040 ... bash
```

userid 0 in container is mapped to 231072 in host

```
1 root@host:/# cat /proc/4080/uid_map
2 0 231072 65536
```

Any user who can start and run Docker containers effectively has root access to the host. For example, consider that you can run the following:

```
1 usermod -aG docker dummyuser
```

```
1 [mosorio@host ~]$ docker run -v /:/homeroot -  
  it debian bash  
2 root@b33b601c262c:/# ls /homeroot/root/.ssh  
3 authorized_keys  config  id_rsa  known_hosts
```

REMOVE SETUID/SETGID BINARIES

Chances are that your application doesn't need any setuid or setgid binaries.

```
1 find / -perm +6000 -type f -exec chmod a-s  
    {} \;
```

LIMIT MEMORY AND CPU

Limiting memory protects against both DoS attacks and applications with memory leaks

```
1  docker run -m 128m --memory-swap 128m ubuntu:
    latest
2  docker run -d --name load1 -c 2048
3  docker run -d --restart=on-failure:10 my-
    flaky-image
```

Also, apply resource limits (ulimits)!

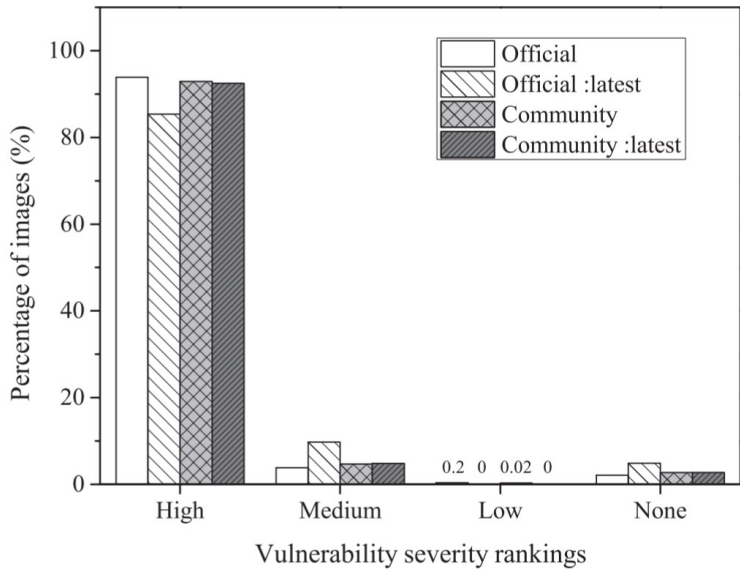
The Linux kernel defines sets of privileges—called capabilities—that can be assigned to processes to provide them with greater access to the system.

```
1 $ docker run --cap-drop all debian chown 100  
   /tmp  
2 chown: changing ownership of '/tmp':  
   Operation not permitted
```

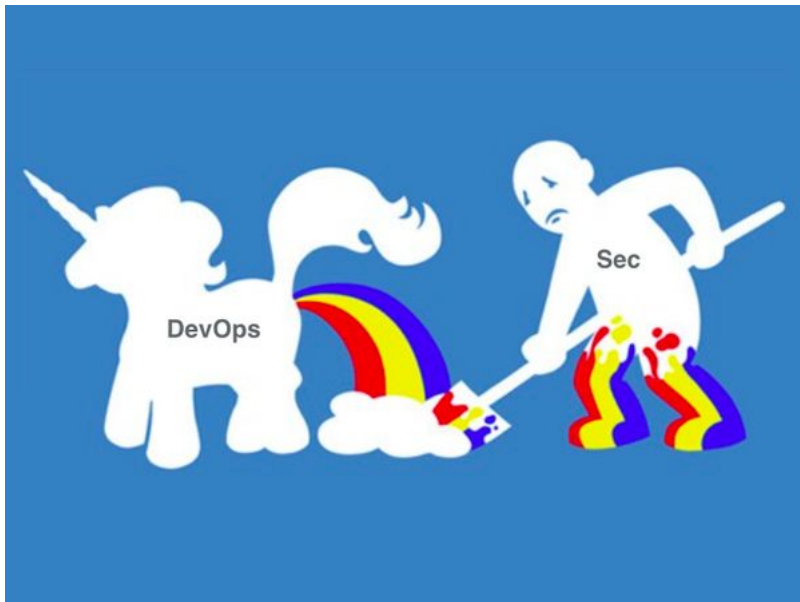
Table 3: Number of Vulnerabilities per Image.

Image Type	Total Images	Number of Vulnerabilities				
		Mean	Median	Max	Min	Std. Dev.
Community	352,416	199	158	1,779	0	139
Community :latest	75,533	196	153	1,779	0	141
Official	3,802	185	127	791	0	145
Official :latest	93	76	76	392	0	59

[A Study of Security Vulnerabilities on Docker Hub](#)



DOCKER IS ABOUT RUNNING RANDOM
CRAP FROM THE INTERNET AS ROOT
ON YOUR HOST.



<http://devops.tumblr.com>

APPLYING UPDATES

The ability to quickly apply updates to a running system is critical to maintaining security, especially when vulnerabilities are disclosed in common utilities and frameworks.

- Identify images that require updating
- Get or create an updated version of each base image
- For each dependent image, run `docker build` with the `nocache` argument. Again, push these images
- Restart
- Clean

SELINUX

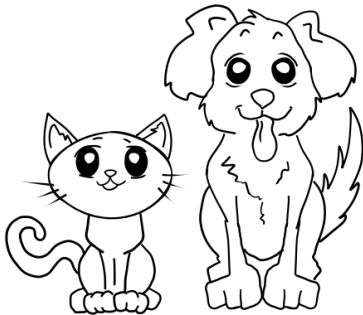
The SELinux, or Security Enhanced Linux, module was developed by the United States National Security Agency (NSA) as an implementation of what they call mandatory access control (MAC)

Type Enforcement

PROCESS TYPES

The SELinux primary model of enforcement is called type enforcement. Basically this means we define the label on a process based on its type, and the label on a file system object based on its type.

Imagine a system where we define types on objects like cats and dogs. A cat and dog are process types.



CAT

DOG



CAT_CHOW



DOG_CHOW

RedHat



ALLOW



CAT



CAT_CHOW:FOOD



EAT



ALLOW



DOG

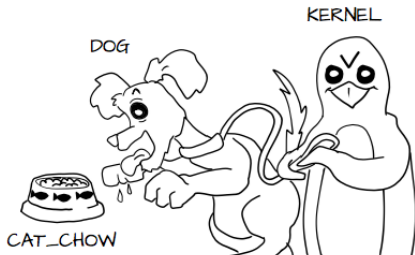


DOG_CHOW:FOOD

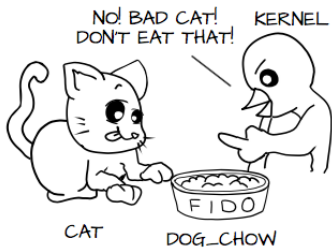


EAT

RedHat






Likewise cats would not be allowed to touch dog food.



RECOMMENDATIONS

- Only run container images from trusted parties. Container applications should drop privileges or run without privileges whenever possible.
- Make sure the kernel is always updated with the latest security fixes; the security kernel is critical.
- Do not disable security features of the host operating system.
- Examine your container images for security flaws and make sure the provider fixes them in a timely manner.
- Further security can be provided at the kernel level by running hardened kernels and using security modules such as AppArmor or SELinux.

-  A. Mouat, *Using Docker: Developing and Deploying Software with Containers*.
" O'Reilly Media, Inc.", 2015.
-  M. Hayden and R. Carbone, "Securing linux containers," *GIAC (GCUX) Gold Certification, Creative Commons Attribution-ShareAlike 4.0 International License*, 2015.
-  R. Shu, X. Gu, and W. Enck, "A study of security vulnerabilities on docker hub," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, pp. 269–280, ACM, 2017.