O bezpieczeństwie kontenerów linuksowych

DAAAAAD, I lost one of my microServices I can't play without it.
I told you he is not mature enough for it. We should have bought him a monolith.
I believe you.
DANIEL STORI {TURNOFF.US}

Kraków, 2019-05-29

Maciej Lasyk

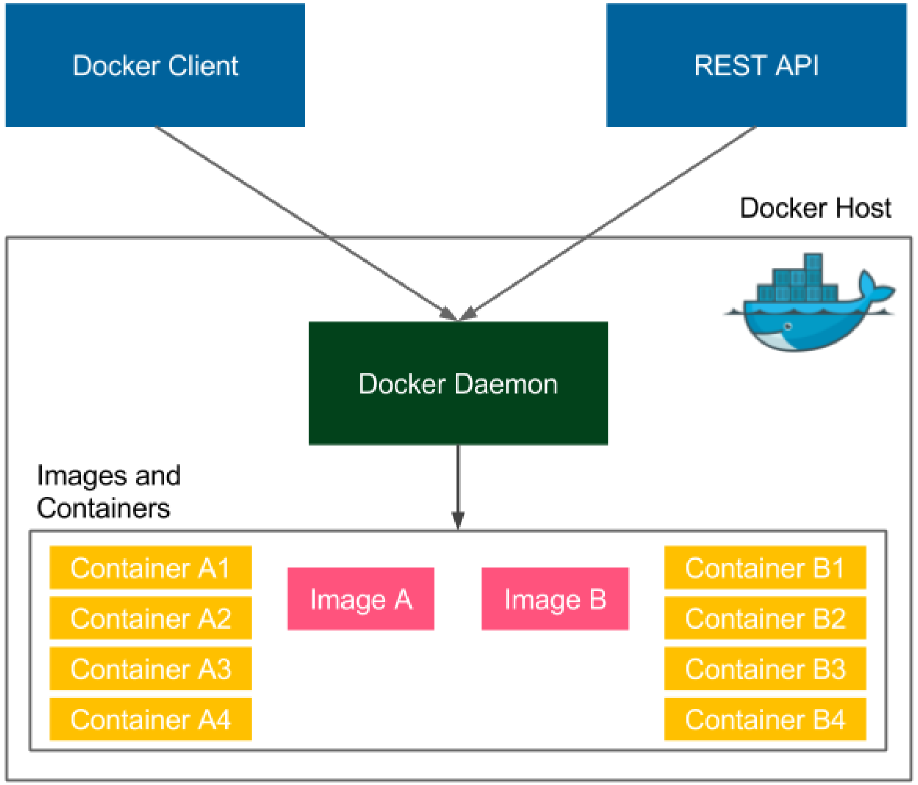
$ whois maciej.lasyk.info

- SRE / cloud operations @Codewise
- fedora project contributor
- @docent-net
- github.com/docent-net/
- maciej.lasyk.info
- dlugodystansowy.pl

Linux containers?

- Used for process containment
- Linux namespaces for providing users/FS/others view
- Cgroups v1/v2 for resources management
- Linux LSMs for providing confinement
- **By design not created for providing additional security layer**
- Some storage copy-on-write magic (not needed btw at all)
- Quo-vadis containers: https://www.youtube.com/watch?v=_GSLj-c_LMI

Docker architecture



Docker architecture

- Binary client ($ docker)
- REST API on docker.sock by default
- ...booring? Not rly
- $ docker run --privileged -v /:/host:rw
- (unless SELinux which by default denies socket access)

Docker security considerations

- docker run --user foo
    - executes the process in the container as non - root
    - dockerd, containerd, and runc still running as root

Docker security considerations

- docker run --user foo
  - executes the process in the container as non - root
  - dockerd, containerd, and runc still running as root
- USER in Dockerfile
  - same as above
  - you can't run dnf/yum/apt-get install whatever

Docker security considerations

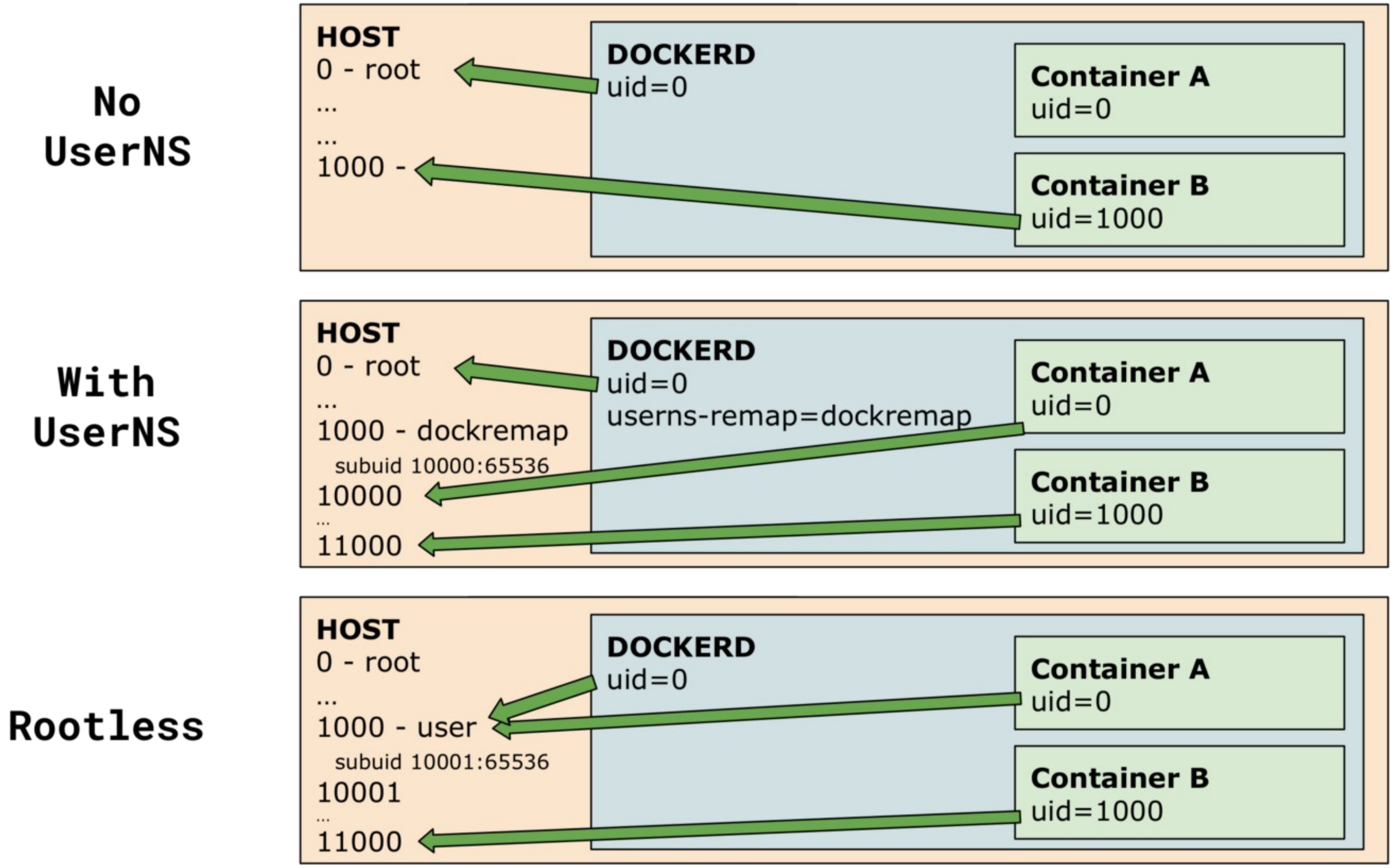
- docker run --user foo
  - executes the process in the container as non - root
  - dockerd, containerd, and runc still running as root
- USER in Dockerfile
  - same as above
  - you can't run dnf/yum/apt-get install whatever
- usermod -aG docker foo
  - allows non - root user to connect to docker.sock
  - remember docker run --privileged -v /:/host - DON'T

Docker - what are privileged containers?

- Basically Linux capabilities unlimited
- See man 7 capabilities
  - setuid
  - getcap / setcap
- Try: --cap-drop=ALL
- Read: runtime-privilege-and-linux-capabilities

Docker - rootless considerations

- https://docs.docker.com/engine/security/userns-remap/
- dockerd --userns-remap
  - executes containers as non - root (dockremap) using user namespaces
  - most similar to rootless, but still needs dockerd, containerd, runc to run from root


No UserNS
HOST
0 - root
...
...
1000 -
DOCKERD
uid=0
Container A
uid=0
Container B
uid=1000
With UserNS
HOST
0 - root
...
1000 - dockremap
subuid 10000:65536
10000
...
11000
DOCKERD
uid=0
userns-remap=dockremap
Container A
uid=0
Container B
uid=1000
Rootless
HOST
0 - root
...
1000 - user
subuid 10001:65536
10001
...
11000
DOCKERD
uid=0
Container A
uid=0
Container B
uid=1000

Rootless finally in Docker?

- Original issue: https://github.com/moby/moby/pull/38050
- https://engineering.docker.com/2019/02/experimenting-with-rootless-docker/
- Downsides:
  - w/out cgroups (so no resource management)
  - w/out apparmor and SELinux
  - w/out overlay networks
  - w/out ports exposing directly - needs socat
  - On Ubuntu overlayFS, rest VFS which is no good for production
- **So this is an experiment**

“Containers do not contain”

- Originally said by Dan Walsh: docker-security-selinux
- “I have heard people say Docker containers are as secure as running processes in separate VMs/KVM.”
- “I know people are downloading random Docker images and then launching them on their host.”
- “I have even seen PaaS servers (not OpenShift, yet) allowing users to upload their own images to run on a multi-tenant system.”
- “I have a co-worker who said: "Docker is about running random code downloaded from the Internet and running it as root.”

“Containers do not contain”

- **Containers were not created for/security by design!**
- Solaris zones were, and have great support directly from FS (see ZFS, Crossbow)
- See Containers do not contain

Docker & SELinux

- Stop disabling SELinux
- “Container security: frustration in the RedHat security team was high because of difficulties to integrate patches into the Docker product [...]” [source]
- See: Docker versus Systemd - Can't we just get along?

Docker & SELinux - do you really need LSM?

Major kernel subsystems are not namespaced like:

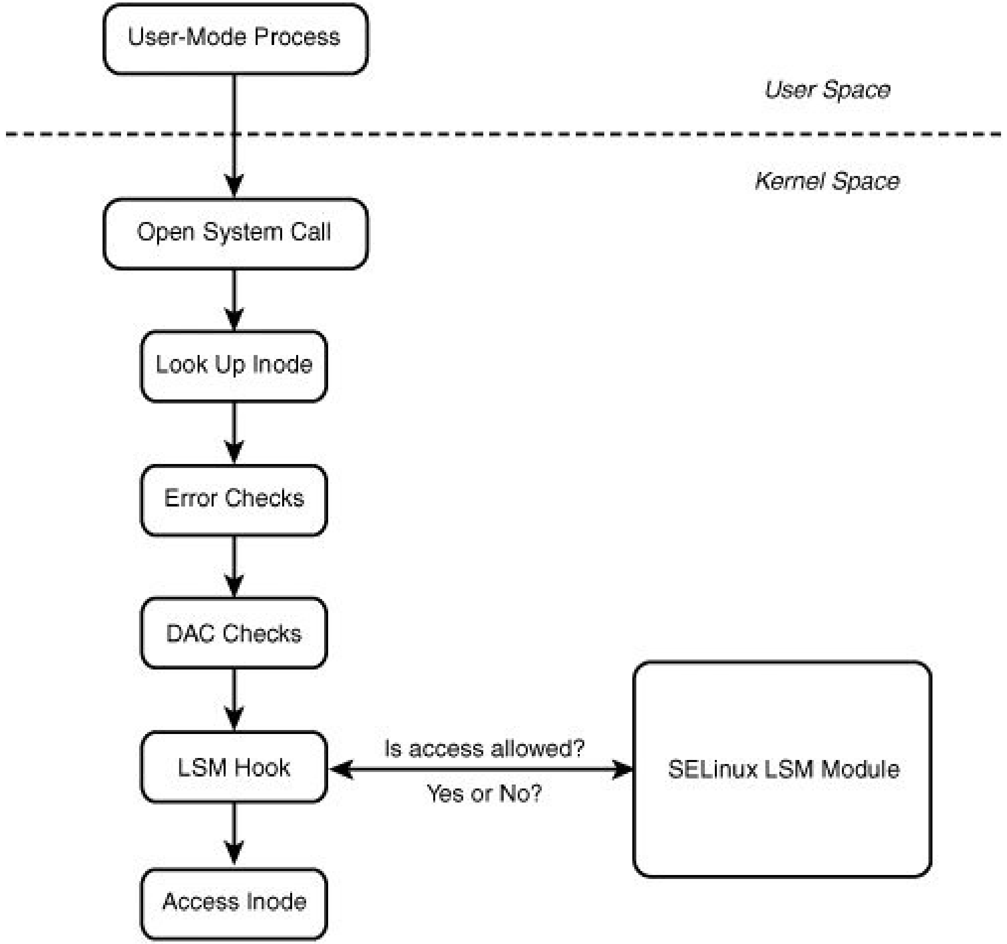
- Cgroups
- file systems under /sys
- /proc/sys, /proc/sysrq-trigger, /proc/irq, /proc/bus

Devices are not namespaced:

- /dev/mem
- /dev/sd* file system devices

Kernel Modules are not namespaced

If you can communicate or attack one of these as a privileged process, you can own the system.


User-Mode Process
User Space
Kernel Space
Open System Call
Look Up Inode
Error Checks
DAC Checks
LSM Hook
Is access allowed?
Yes or No?
SELinux LSM Module
Access Inode

Docker seccomp

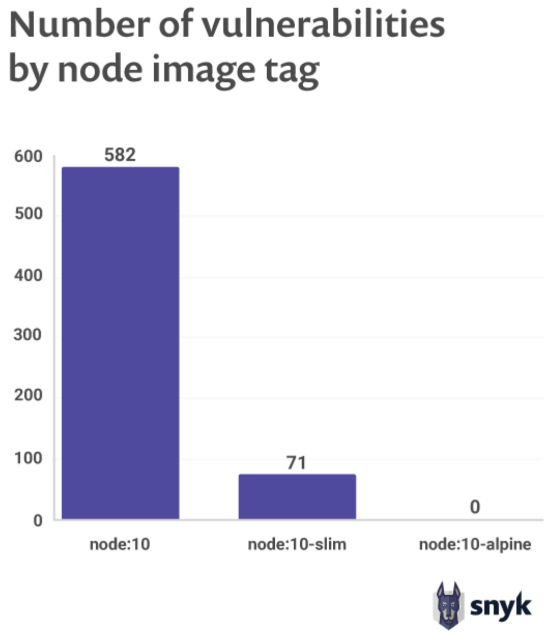
- Kernel w/seccomp
- Docker-engine w/seccomp
- Read: https://docs.docker.com/engine/security/seccomp/

Docker images

- Remember ““I have a co-worker who said: "Docker is about running random code downloaded from the Internet and running it as root.”?
- Read most-popular-docker-images-each-contain-at-least-30-vulnerabilities/

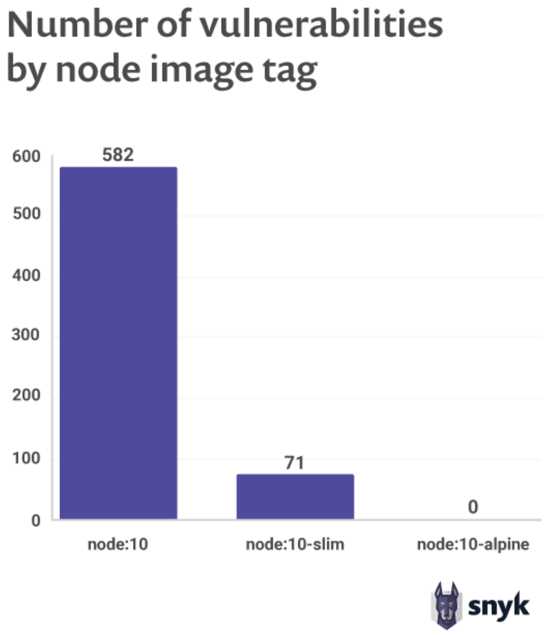
Docker images

- Remember ““I have a co-worker who said: "Docker is about running random code downloaded from the Internet and running it as root.”?
- Read most-popular-docker-images-each-contain-at-least-30-vulnerabilities/



Docker images

- Remember ““I have a co-worker who said: "Docker is about running random code downloaded from the Internet and running it as root.”?
- Read most-popular-docker-images-each-contain-at-least-30-vulnerabilities/



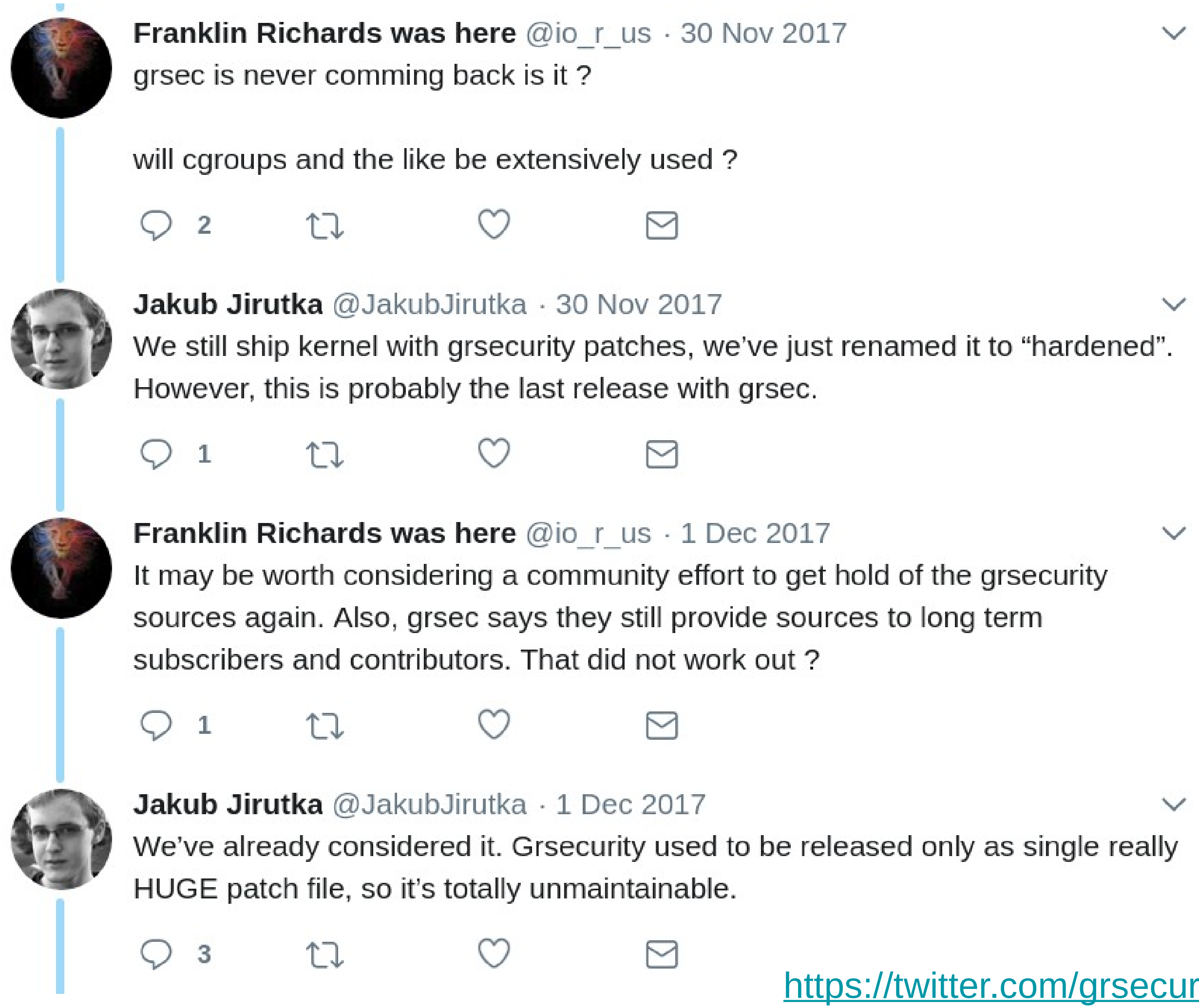
[...] Alpine Linux doesn’t maintain a security advisory program, which means that if a system library has vulnerabilities, Alpine Linux will not issue an official advisory about it [...]

Is Alpine images secure as they say?

- **Alpine Linux is a security-oriented, lightweight Linux distribution based on musl libc and busybox.**
- Top G results: **Alpine so secure, very fast, best, why use anything else?**
- APK - yet another packaging system
  - How much effort needs maintaining packaging system and packages?
  - https://news.ycombinator.com/item?id=17981452
  - 2 pplf for review(!): https://wiki.alpinelinux.org/wiki/Creating_an_Alpine_package#Code_review
  - **“To successfully have your package pass through code reviewers (as of Feb 18, 2018 are nmeum and jirutka on GitHub) and possible increased acceptance, the following conventions need to be followed:”**
  - Looks like npm install
  - Why not rpm or deb? (because no glibc!)
  - Last year no critical security problems with dnf/yum/apt; those are very stable and many, many ppl work on it; and review processes are thorough maintained by number of ppl

Is Alpine images secure as they say?

- Alpine has Kernel patched by unofficial grsecurity
- Unofficial because grsec is no more free
- Can you really maintain Kernel patches for free? NO



Franklin Richards was here @io_r_us · 30 Nov 2017

grsec is never comming back is it ?

will cgroups and the like be extensively used ?

2

Jakub Jirutka @JakubJirutka · 30 Nov 2017

We still ship kernel with grsecurity patches, we've just renamed it to "hardened". However, this is probably the last release with grsec.

1

Franklin Richards was here @io_r_us · 1 Dec 2017

It may be worth considering a community effort to get hold of the grsecurity sources again. Also, grsec says they still provide sources to long term subscribers and contributors. That did not work out ?

1

Jakub Jirutka @JakubJirutka · 1 Dec 2017

We've already considered it. Grsecurity used to be released only as single really HUGE patch file, so it's totally unmaintainable.

3

https://twitter.com/grsecurity/status/936422357757022209

Is Alpine images secure as they say?

- Alpine has Kernel patched by unofficial grsecurity
- Unofficial because grsec is no more free
- Can you really maintain Kernel patches for free? NO
- And it doesn’t matter – there is no “container Kernel”, just the host one!

Alpine: musl vs glibc

- How many of you can compile w/first and the second?
- Can u rly strace w/musl?
- Operational drama
- Glibc is huge as its support & ppl behind it (G, RH, Canonical, IBM, whatever)
- Some binaries will crash in corner cases w/musl
- Read: what_is_musl_and_glibc
- systemd will not work w/musl

Alpine: so why ppl use it?

- Because it’s small; few of MBs (6 or smt)
- “If it consists of just few libs it must be secure”
- Do you have any other ideas?

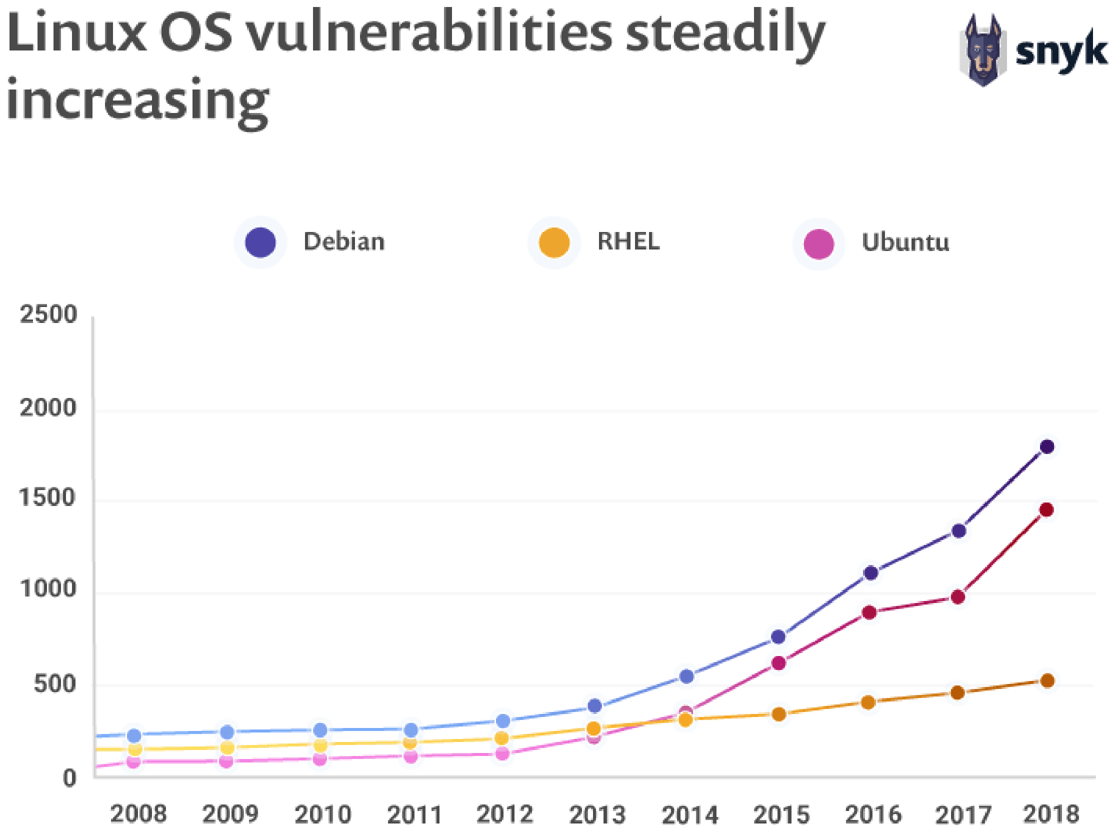
Alpine: so why ppl use it?

- Because it’s small; few of MBs (6 or smt)
    - We have currently layered FSes w/copy-on-write
    - You can really download 100mb image very fast
    - You don’t have to redownload it at all
- “If it consists of just few libs it must be secure”
    - Yeah, add more and pray that those are secure (remember they don’t have security advisory program!)
- Do you have any other ideas?

Alpine: history

- Created w/routers, small boxes etc in mind
- Why so high adoption in Docker?
  - Because Docker hub had gigantic performance problems these times, so little Alpine fixed it
  - Because back then storage drivers (aufs /n Debians and devicemapper on RHs) sucked a lot and layers were just too big to handle w/good performance [thx Marcin]

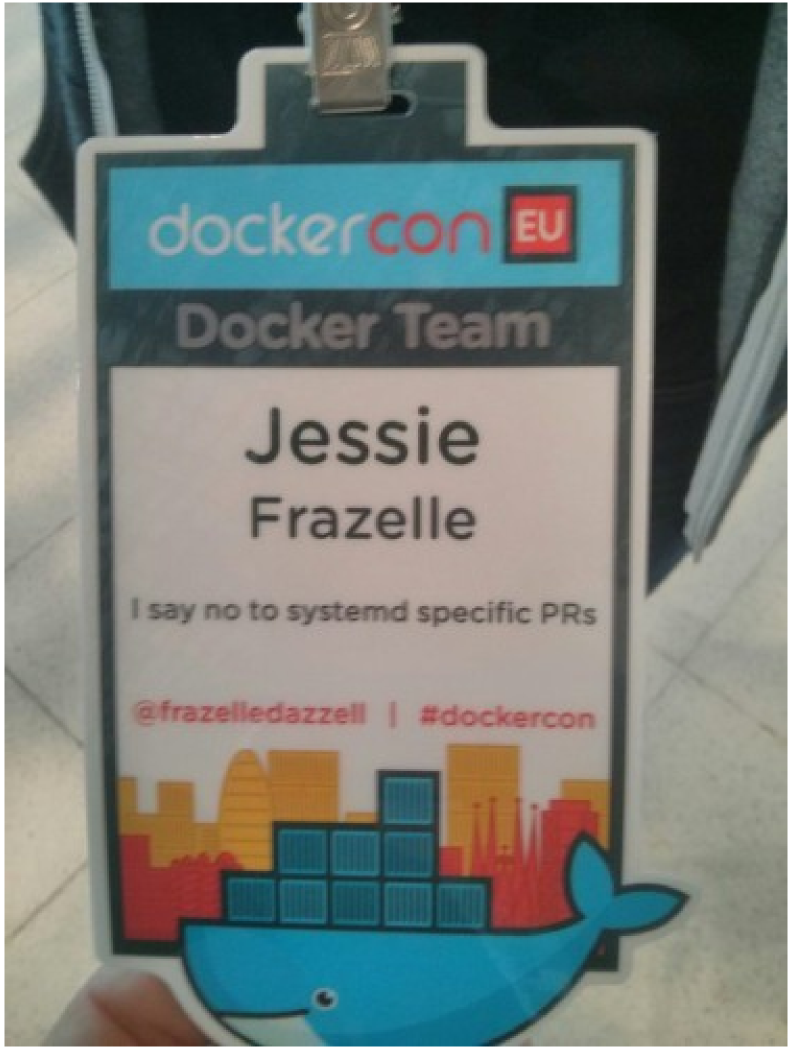
Which image?


Linux OS vulnerabilities steadily increasing
snyk
Debian
RHEL
Ubuntu
2500
2000
1500
1000
500
0
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018

Docker & systemd

"This is Lennart Poettering," said Walsh, showing a picture. "This is Solomon Hykes", showing another. "Neither one of them is willing to compromise much. And I get to be in the middle between them."

[source]

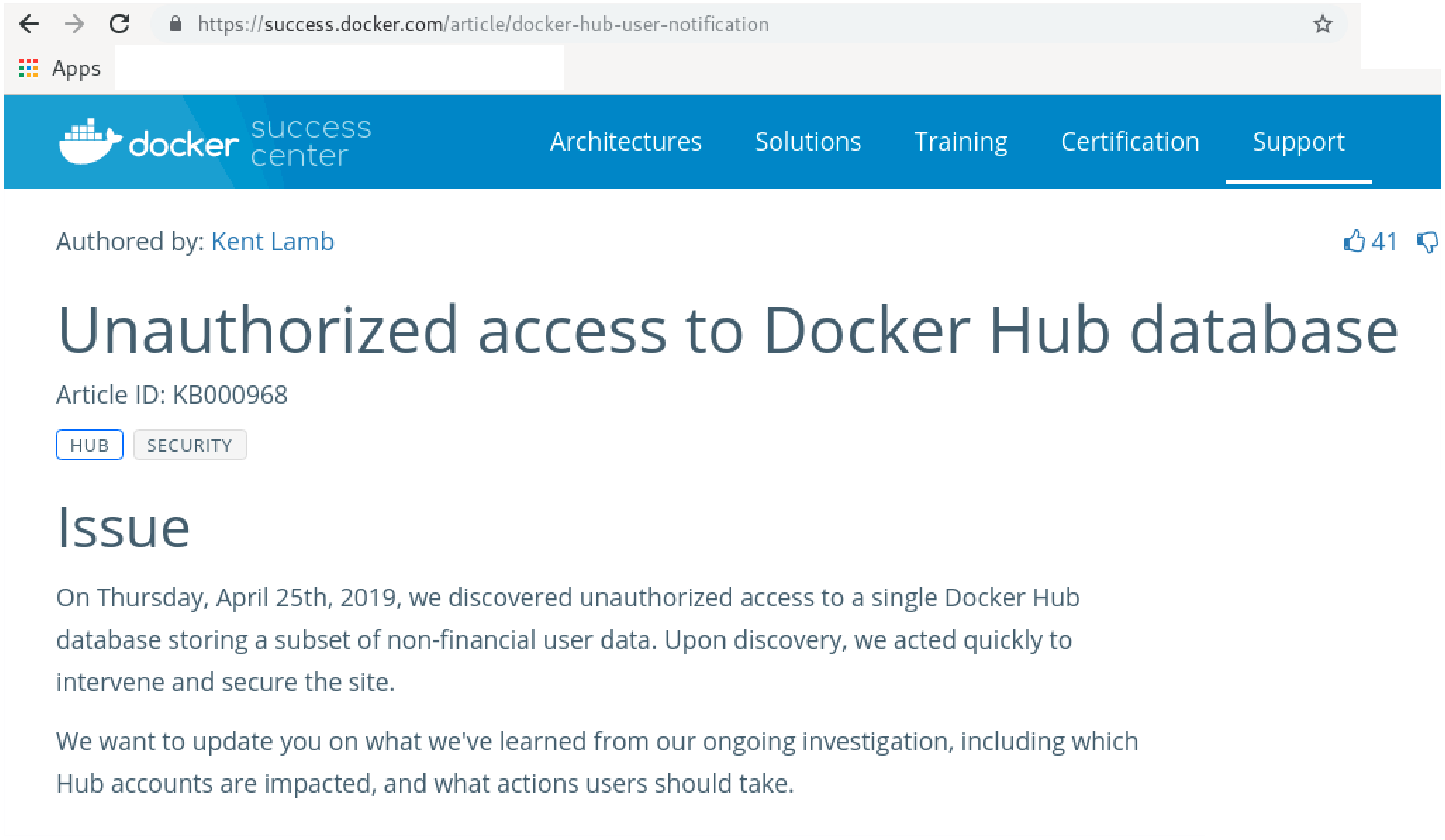

dockercon EU
Docker Team
Jessie
Frazelle
I say no to systemd specific PRs
@frazelledazzell | #dockercon

Docker & systemd

"According to Walsh's presentation, the root cause of the conflict is that the Docker daemon is designed to take over a lot of the functions that systemd also performs for Linux. These include initialization, service activation, security, and logging. "In a lot of ways Docker wants to be systemd," he claimed. "It dreams of being systemd.""

[source]

dockercon EU
Docker Team
Jessie
Frazelle
I say no to systemd specific PRs
@frazelledazzell | #dockercon



Authored by: Kent Lamb

Unauthorized access to Docker Hub database

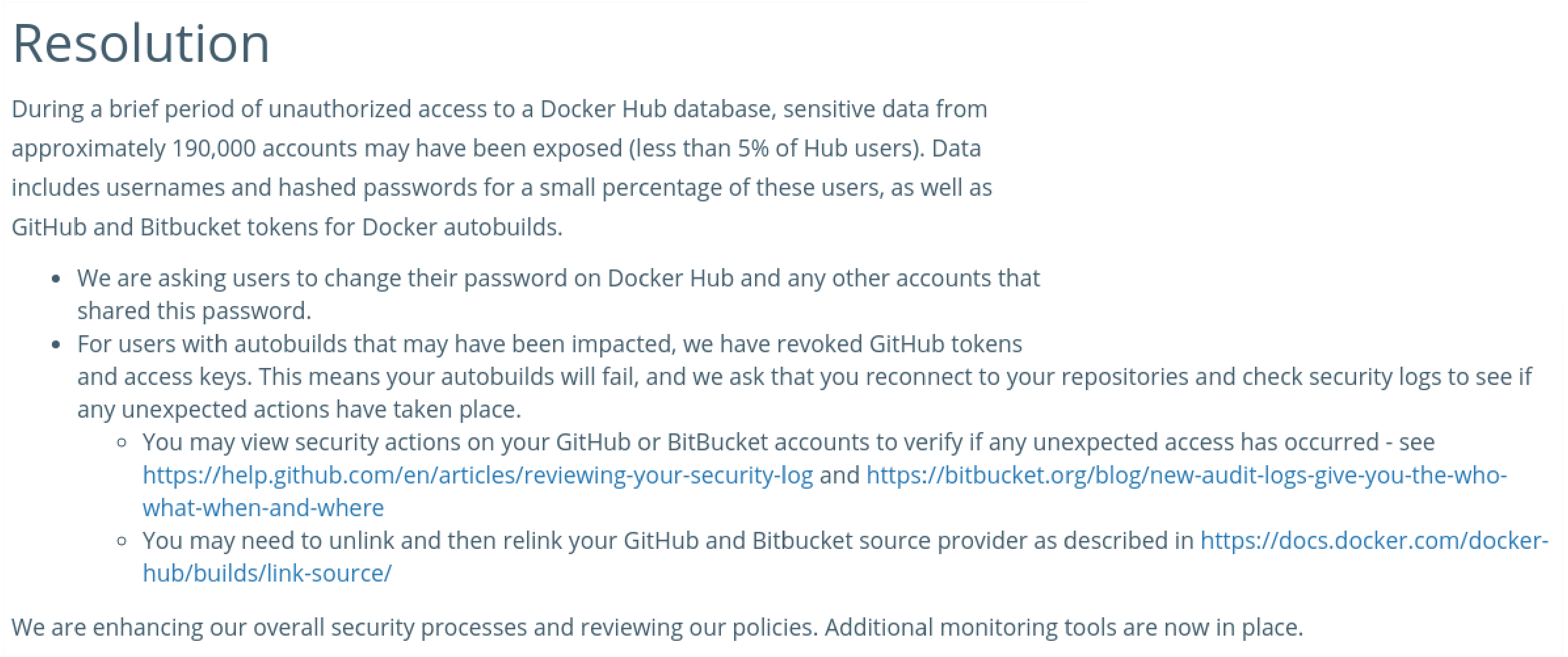
Article ID: KB000968

HUB SECURITY

Issue

On Thursday, April 25th, 2019, we discovered unauthorized access to a single Docker Hub database storing a subset of non-financial user data. Upon discovery, we acted quickly to intervene and secure the site.

We want to update you on what we've learned from our ongoing investigation, including which Hub accounts are impacted, and what actions users should take.



Resolution

During a brief period of unauthorized access to a Docker Hub database, sensitive data from approximately 190,000 accounts may have been exposed (less than 5% of Hub users). Data includes usernames and hashed passwords for a small percentage of these users, as well as GitHub and Bitbucket tokens for Docker autobuilds.

- We are asking users to change their password on Docker Hub and any other accounts that shared this password.
- For users with autobuilds that may have been impacted, we have revoked GitHub tokens and access keys. This means your autobuilds will fail, and we ask that you reconnect to your repositories and check security logs to see if any unexpected actions have taken place.
  - You may view security actions on your GitHub or BitBucket accounts to verify if any unexpected access has occurred - see https://help.github.com/en/articles/reviewing-your-security-log and https://bitbucket.org/blog/new-audit-logs-give-you-the-who-what-when-and-where
  - You may need to unlink and then relink your GitHub and Bitbucket source provider as described in https://docs.docker.com/docker-hub/builds/link-source/

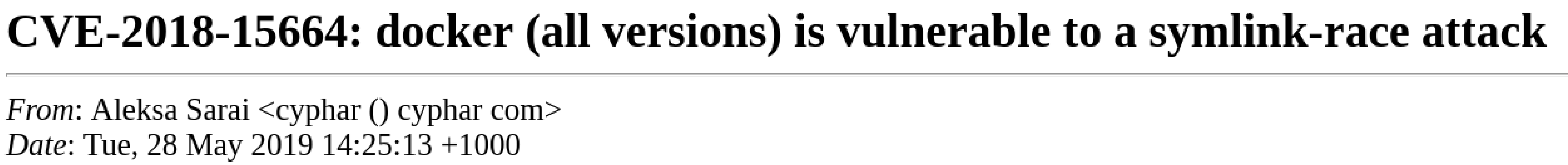
We are enhancing our overall security processes and reviewing our policies. Additional monitoring tools are now in place.

CVE-2018-15664: docker (all versions) is vulnerable to a symlink-race attack

---

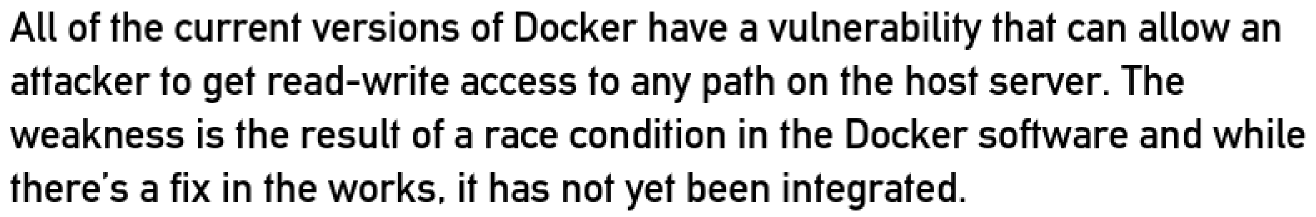
From: Aleksa Sarai <cyphar () cyphar com>
Date: Tue, 28 May 2019 14:25:13 +1000

https://seclists.org/oss-sec/2019/q2/131
https://duo.com/decipher/docker-bug-allows-root-access-to-host-file-system



CVE-2018-15664: docker (all versions) is vulnerable to a symlink-race attack

From: Aleksa Sarai <cyphar () cyphar com>
Date: Tue, 28 May 2019 14:25:13 +1000



All of the current versions of Docker have a vulnerability that can allow an attacker to get read-write access to any path on the host server. The weakness is the result of a race condition in the Docker software and while there's a fix in the works, it has not yet been integrated.

https://seclists.org/oss-sec/2019/q2/131
https://duo.com/decipher/docker-bug-allows-root-access-to-host-file-system

CVE-2018-15664: docker (all versions) is vulnerable to a symlink-race attack

---

From: Aleksa Sarai <cyphar () cyphar com>
Date: Tue, 28 May 2019 14:25:13 +1000

All of the current versions of Docker have a vulnerability that can allow an attacker to get read-write access to any path on the host server. The weakness is the result of a race condition in the Docker software and while there's a fix in the works, it has not yet been integrated.

"If an attacker can add a symlink component to the path *after* the resolution but *before*it is operated on, then you could end up resolving the symlink path component on the host as root. In the case of 'docker cp' this gives you read *and* write access to any path on the host."

https://seclists.org/oss-sec/2019/q2/131
https://duo.com/decipher/docker-bug-allows-root-access-to-host-file-system

CVE-2018-15664: docker (all versions) is vulnerable to a symlink-race attack

From: Aleksa Sarai <cyphar () cyphar com>
Date: Tue, 28 May 2019 14:25:13 +1000

All of the current versions of Docker have a vulnerability that can allow an attacker to get read-write access to any path on the host server. The weakness is the result of a race condition in the Docker software and while there's a fix in the works, it has not yet been integrated.

"If an attacker can add a symlink component to the path *after* the resolution but *before*it is operated on, then you could end up resolving the symlink path component on the host as root. In the case of 'docker cp' this gives you read *and* write access to any path on the host."

"The most likely case for this particular vector would be a managed cloud which let you (for instance) copy configuration files into a running container or read files from within the container (through "docker cp").," Sarai said via email.

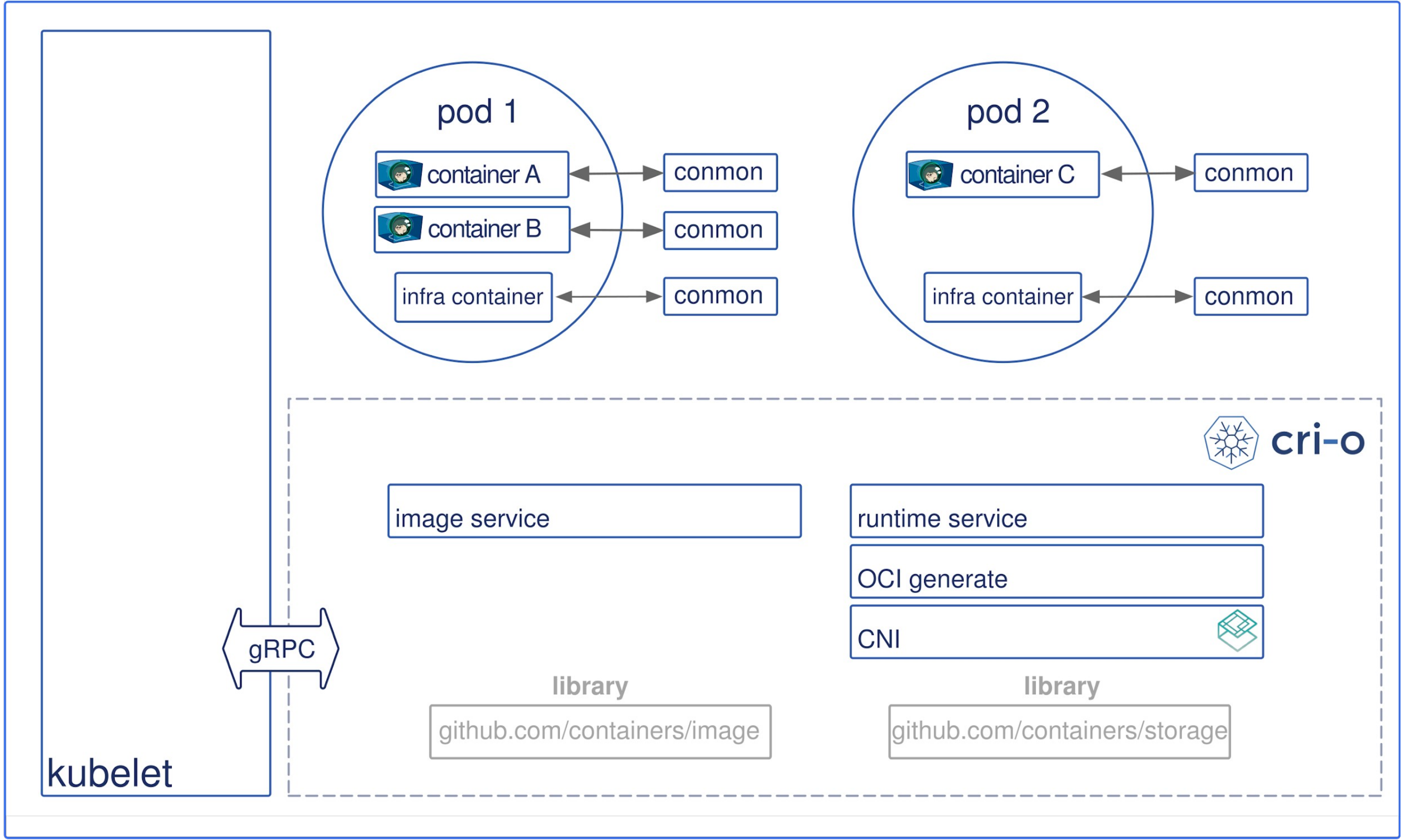
https://seclists.org/oss-sec/2019/q2/131
https://duo.com/decipher/docker-bug-allows-root-access-to-host-file-system

Remember what Dan said!

http://www.youtube.com/watch?v=o5snlP8Y5GY

Is there a world without Docker?

- Yeah, Podman (cr-engine) and CRI-O (cr-runtime)
- **“CRI-O owes a great deal of gratitude to the upstream Docker project. As Isaac Newton said “If I have seen further, it is by standing on the shoulders of giants.”**


pod 1
container A
container B
infra container
conmon
conmon
conmon
pod 2
container C
infra container
conmon
conmon
cri-o
image service
runtime service
OCI generate
CNI
gRPC
library
github.com/containers/image
library
github.com/containers/storage
kubelet

Podman - what is it?

- drop-in replacement for docker
- #nobigfatdaemons
- one process per container (supervised by init, e.g. systemd)
- systemd-cgroups: https://asciinema.org/a/182946
- user-namespaces
- rootless containers (in k8s pod share same user namespace)
- support for fuse (on newer Kernels w/out root)/overlays
- systemd-features:
    - automated start
    - dependencies between specified containers and other system services (or even containers)
    - socket-activation
    - sd-notify

Podman - howto

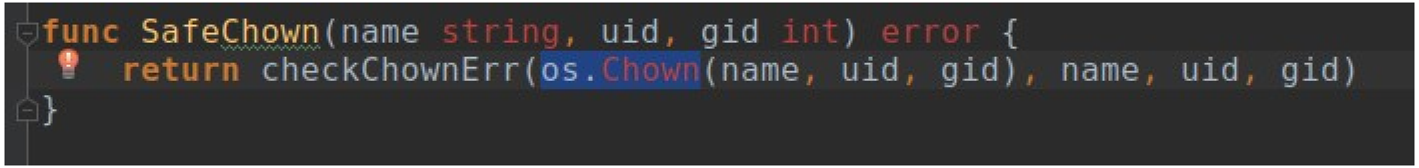
- dnf/yum install -y podman
- alias docker=podman
- https://media.ccc.de/v/ASG2018-177-replacing_docker_with_podman#t=74

Podman - user namespaces?

- Read podman-and-user-namespaces
- each container runs in own user namespace
- “Since the real UID=0 is not mapped into the container, any file owned by root will be treated as owned by nobody. Even if the process inside the container has CAP_DAC_OVERRIDE, it can't override this protection. DAC_OVERRIDE enables root processes to read/write any file on the system, even if the process was not owned by root or world readable or writable.”
- “Podman can use different user namespaces on the same image because of **automatic chowning** built into containers/storage by a team led by Nalin Dahyabhai. “

Podman - user namespaces?

- Read podman-and-user-namespaces
- each container runs in own user namespace
- “Since the real UID=0 is not mapped into the container, any file owned by root will be treated as owned by nobody. Even if the process inside the container has CAP_DAC_OVERRIDE, it can't override this protection. DAC_OVERRIDE enables root processes to read/write any file on the system, even if the process was not owned by root or world readable or writable.”
- “Podman can use different user namespaces on the same image because of **automatic chowning** built into containers/storage by a team led by Nalin Dahyabhai. “



```
func SafeChown(name string, uid, gid int) error {
    return checkChownErr(os.Chown(name, uid, gid), name, uid, gid)
}
```

Podman - user namespaces

```
$ sudo bash -c "echo Test > /tmp/test"
$ sudo chmod 600 /tmp/test
$ sudo ls -l /tmp/test
-rw-------. 1 root root 5 Dec 17 16:40 /tmp/test
```

```
$ sudo podman run -ti -v /tmp/test:/tmp/test:Z --uidmap 0:100000:5000 fedora sh
# id
uid=0(root) gid=0(root) groups=0(root)
# ls -l /tmp/test
-rw-rw----. 1 nobody nobody 8 Nov 30 12:40 /tmp/test
# cat /tmp/test
cat: /tmp/test: Permission denied
```

~~Docker~~ Podman security considerations

- podman run
  - executes the process in the container as current user
  - dockerd, containerd, and runc not running as **#nobigfatdaemons**
- USER in Dockerfile
  - same as above
  - you can run dnf/yum/apt-get install whatever
- usermod -aG docker foo
  - No usermod as no docker.socket

Podman rootless

- Read how-does-rootless-podman-work
- Watch: replacing_docker_with_podman
- Working out-of-the-box
- “The Podman tool is enabling people to build and use containers without sacrificing the security of the system; you can give your developers the access they need without giving them root.”

What if no Docker, no Podman - just Linux?

What if no Docker, no Podman - just Linux?

systemd FTW!

What if no Docker, no Podman - just Linux?

systemd FTW!

- systemd-run process confinement
- systemd portable services
- systemd-nspawn

Process confinement w/systemd-run

- See my systemd talks here
- **man systemd.resource-control, systemd.exec**
  - ProtectHome=true, ProtectSystem=Strict, ReadOnlyDirectories, InAccessibleDirectoreis, ReadWriteDirectories, PrivateTmp, TemporaryFileSystem, BindPath, BindReadOnlyPath
  - MemoryMax and others
  - CPUQuota and others
  - IPAddressDeny and others
- Read ip-accounting-and-access-lists-with-systemd

Process confinement w/systemd-run

- **systemd-run -p <param1> -p <param2> -t /bin/sh**
  - IPAddressDeny=any + IPAddressAllow=8.8.8.8 + IPAddressAllow=127.0.0.0/8
  - ProtectSystem=strict
  - ProtectHome=true
  - PrivateTmp=true
  - BindPaths=/mnt/sd-test
  - CPUQuota=20%
  - MemoryMax/Min

systemd-nspawn

- watch “systemd-nspawn is chroot on steroids” (Lennart Poettering)
- created for debugging boot process of Linux OS (by RedHat / Lennart & co)
- single process/service w/systemd as init
- quite low - level
- this was mainly for debugging init process when working on systemd
- steeper learning curve
- **man systemd-nspawn**

systemd portable services

- Watch: portable_services_are_ready_to_use
- Read:
    - walkthrough for Portable Services, walkthrough for Portable Services in Go
    - portable services
    - dynamic-users-with-systemd.html
- normal services w/optional chroot and some containment
- multiple sandboxing options
- leave no artifacts
- Own transient user database
- Builtin ready security profiles
- This is just a wrapper around systemd (portablectl)

systemd portable services - dynamic users

- nss-systemd (not using /etc/passwd at all)
- man 5 systemd.exec
- Setting DynamicUser=yes implies ProtectSystem=strict and ProtectHome=read-only and PrivateTmp=yes
- These sand-boxing options turn off write access to pretty much the whole OS directory tree, with a few relevant exceptions, such as the API file systems /proc, /sys and so on, as well as /tmp and /var/tmp.
- Setting DynamicUser=yes implies RemoveIPC=yes
- allocation of users cheap and ephemeral

Process confinement w/systemd-run

- **systemd-run -p <param1> -p <param2> -t /bin/sh**
  - IPAddressDeny=any + IPAddressAllow=8.8.8.8 + IPAddressAllow=127.0.0.0/8
  - ProtectSystem=strict
  - ProtectHome=true
  - PrivateTmp=true
  - BindPaths=/mnt/sd-test
  - CPUQuota=20%
  - DynamicUser=true (see id)
  - PrivateUsers=true (see ps, ls)

systemd-analyze security

- analyzes the security and sandboxing settings of one or more specified service units
- The command checks for various security-related service settings, assigning each a numeric "exposure level" value, depending on how important a setting is
- It then calculates an overall exposure level for the whole unit, which is an estimation in the range 0.0…10.0 indicating how exposed a service is security-wise

Sources, urls, ppl

- https://rootlesscontaine.rs/
- https://snyk.io/blog/top-ten-most-popular-docker-images-each-contain-at-least-30-vulnerabilities/
- https://media.ccc.de/v/ASG2018-177-replacing_docker_with_podman
- https://opensource.com/article/19/2/how-does-rootless-podman-work
- https://opensource.com/article/18/12/podman-and-user-namespaces
- https://opensource.com/article/18/10/podman-more-secure-way-run-containers
- https://www.youtube.com/watch?v=-MvKe5TFW7g
- https://www.projectatomic.io/blog/2018/02/reintroduction-podman/
- https://learning.oreilly.com/library/view/continuous-delivery-with/9781787125230/5ef77ae7-ce0c-4f85-92a6-a336bbfe8c29.xhtml
- https://www.certdepot.net/death-of-docker/
- https://opensource.com/business/14/7/docker-security-selinux

Special thanks to Dan Walsh, Lennart Poettering and Marcin Skarbek <3

O bezpieczeństwie kontenerów linuksowych

SO/DO

DAAAAAD, I lost one of my microServices I can't play without it.
I told you he is not mature enough for it. We should have bought him a monolith.
I believe you.
DANIEL STORI {TURNOFF.US}

Kraków, 2019-05-29

Maciej Lasyk