State of Container Security

Dan Walsh @rhatdan

Please Stand

Please read out loud all text in RED

I Promise

To say
Make a copy
Rather than
Make a Xerox

I Promise

To say

Tissue

Rather than

Kleenex

I Promise

To say

Container Registries

Rather than

Docker registries

I Promise

To say

Container Images

Rather than

Docker images

Je promets

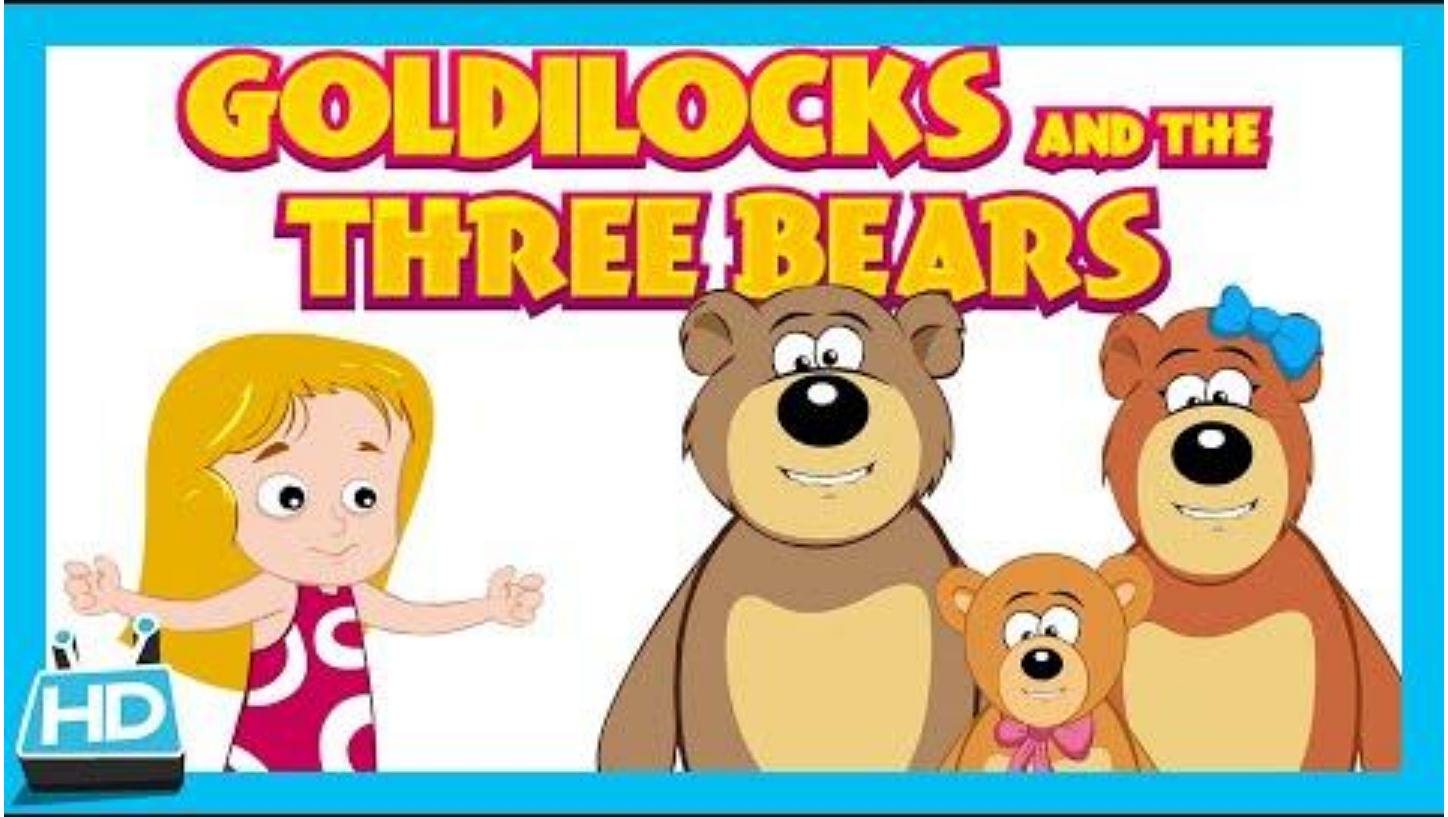
Dire
Les conteneurs
Plutôt que
Docker Containers

Sit Down


Dan Quiote

boucle d'or
&
les trois bières

Goldylocks
&
The Three Beers


GOLDILOCKS AND THE THREE BEARS
HD


TOO
HARD


TOO
SOFT
TOO
HARD


TOO SOFT
JUST RIGHT
TOO HARD

No one turns up security

No one turns up security

- How many of you have ever done
  - podman run --cap-drop capability ...

No one turns up security

- How many of you have ever done
  - podman run --cap-drop capability ...
- How many of you have ever done
  - podman run --privileged …

No one turns up security

- How many of you have ever done
  - podman run --cap-drop capability ...
- How many of you have ever done
  - podman run --privileged …
- People turn down security… Sadly **setenforce 0**

No one turns up security

- How many of you have ever done
  - podman run --cap-drop capability ...
- How many of you have ever done
  - podman run --privileged …
- People turn down security… Sadly **setenforce 0**
- How do I get users to move from ...

Seriously, stop disabling SELinux.

Learn how to use it before you blindly shut it off.

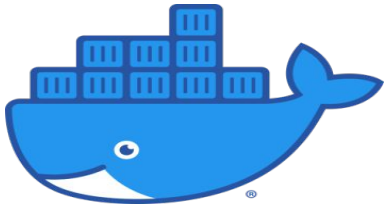
Every time you run `setenforce 0`, you make Dan Walsh weep.

Dan is a nice guy and he certainly doesn't deserve that.

#nobigfatdaemons

OCI Images format

Container Engines



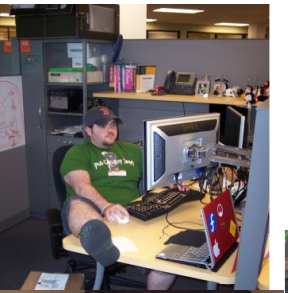
OCI Images format

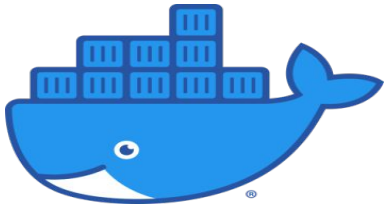
Humans & Orchestators

Container Engines

OCI Images format

kubernetes





Humans & Orchestators

kubernetes

Container Engines

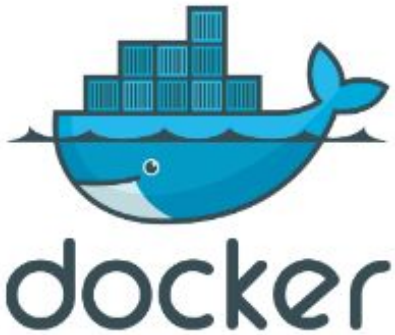
cri-o

buildah

podman

containerd

OCI Images format



cri-o

podman

skopeo

buildah

Just say no to root (in containers)

Even smart admins can make bad decisions.

29 Mar 2018 | Daniel J Walsh (Red Hat) | 76 | 5 comments

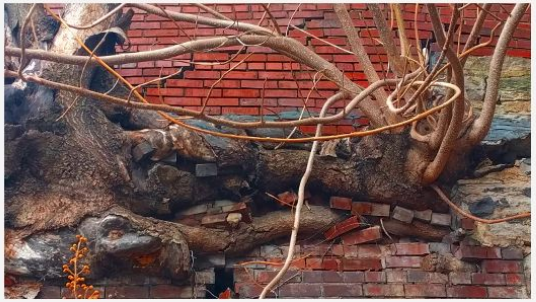


Image credits : *Rikki Endsley. CC BY-SA 4.0*

I get asked all the time about the different security measures used to control what container processes on a system can do. Most of these I covered in previous articles on Opensource.com:

- Are Docker containers really secure?
- Bringing new security features to Docker
- Docker security in the future

Almost all of the above articles are about controlling what a privileged process

Limiting the power of root: Capabilities

- Allow 14 out of 37 Linux Capabilities by default.

Limiting the power of root: Capabilities

- Allow 14 out of 37 Linux Capabilities by default.
- Originally defined by upstream Docker Project

Limiting the power of root: Capabilities

- Allow 14 out of 37 Linux Capabilities by default.
- Originally defined by upstream Docker Project
- Do you know what they are?

Limiting the power of root: Capabilities

AUDIT_WRITE, CHOWN, DAC_OVERRIDE, FOWNER, FSETID, KILL, MKNOD,

NET_BIND_SERVICE, NET_RAW, SETFCAP, SETGID, SETPCAP, SETUID,

SYS_CHROOT

Limiting the power of root: Capabilities

AUDIT_WRITE, CHOWN, DAC_OVERRIDE, FOWNER, FSETID, KILL, MKNOD,

NET_BIND_SERVICE, NET_RAW, SETFCAP, SETGID, SETPCAP, SETUID,

SYS_CHROOT

Limiting the power of root: Capabilities

AUDIT_WRITE, CHOWN, DAC_OVERRIDE, FOWNER, FSETID, KILL, MKNOD,

NET_BIND_SERVICE, NET_RAW, SETFCAP, SETGID, SETPCAP, SETUID,

SYS_CHROOT

Limiting the power of root: Capabilities

AUDIT_WRITE, CHOWN, DAC_OVERRIDE, FOWNER, FSETID, KILL, MKNOD,

NET_BIND_SERVICE, NET_RAW, SETFCAP, SETGID, SETPCAP, SETUID,

SYS_CHROOT

Limiting the power of root: Capabilities

AUDIT_WRITE, CHOWN, DAC_OVERRIDE, FOWNER, FSETID, KILL, MKNOD,

NET_BIND_SERVICE, NET_RAW, SETFCAP, SETGID, SETPCAP, SETUID,

SYS_CHROOT

Limiting the power of root: Capabilities

AUDIT_WRITE, CHOWN, DAC_OVERRIDE, FOWNER, FSETID, KILL, MKNOD,

NET_BIND_SERVICE, NET_RAW, SETFCAP, SETGID, SETPCAP, SETUID,

SYS_CHROOT

DEMO

New Idea:
Image Developer Specifies Capabilities

Allow images to specify Capabilities as Image Annotations/Labels

Annotation/Label

LABEL “io.containers.capabilties=SETUID,SETGID”

New Idea:
Image Developer Specifies Capabilities

Allow images to specify Capabilities as Image Annotations/Labels

Annotation/Label

LABEL “io.containers.capabilties=SETUID,SETGID”

Defaults: AUDIT_WRITE, CHOWN, DAC_OVERRIDE, FOWNER, FSETID, KILL, MKNOD, NET_BIND_SERVICE, NET_RAW, SETFCAP, SETGID, SETPCAP, SETUID, SYS_CHROOT

New Idea:
Image Developer Specifies Capabilities

Allow images to specify Capabilities as Image Annotations/Labels

Annotation/Label

LABEL “io.containers.capabilities=SETUID,SETGID”

Defaults: AUDIT_WRITE, CHOWN, DAC_OVERRIDE, FOWNER, FSETID, KILL, MKNOD, NET_BIND_SERVICE, NET_RAW, SETFCAP, SETGID, SETPCAP, SETUID, SYS_CHROOT

Container engine launches container with only SETGID, SETUID

the SELinux COLORING BOOK

"It's raining cats and dogs!"

written by DAN WALSH

illustrated by MÁIRÍN DUFFY

Every Container Runtime CVE container breakout was a file system breakout.

CVE-2015-3629 Symlink traversal on container respawn allows local privilege escalation

SELinux Blocked

CVE-2015-3627 Insecure opening of file-descriptor 1 leading to privilege escalation

SELinux Blocked

CVE-2015-3630 Read/write proc paths allow host modification & information disclosure

SELinux Blocked

CVE-2015-3631 Volume mounts allow LSM profile escalation

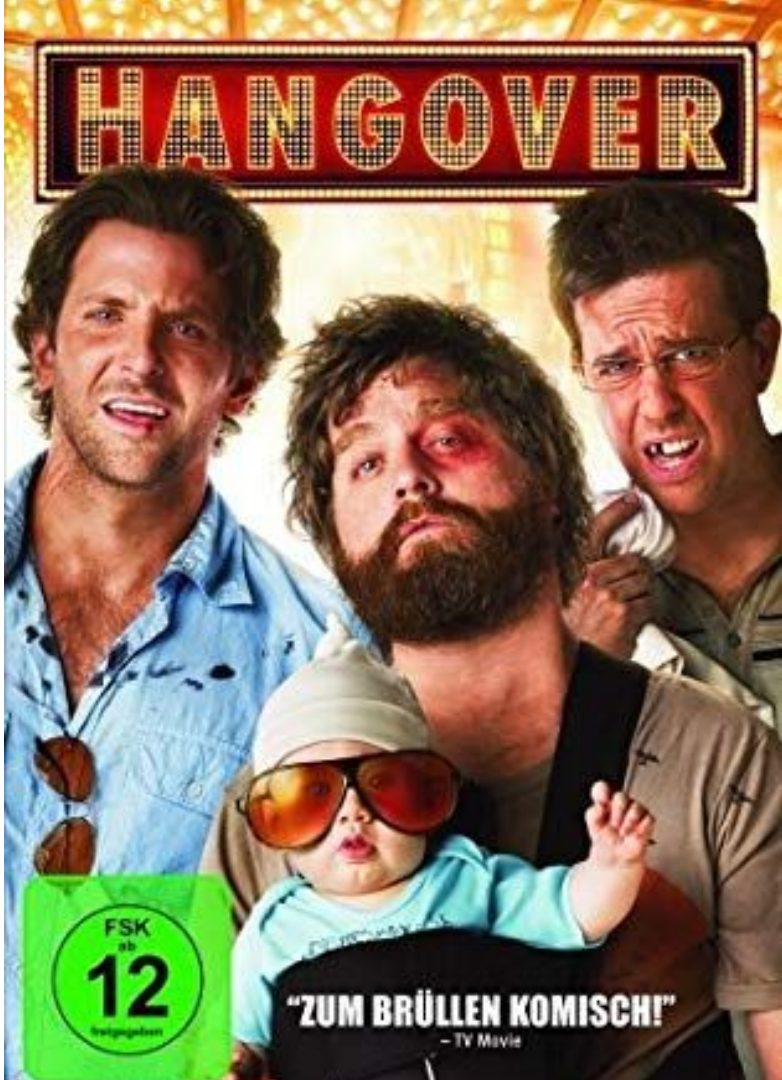
SELinux Blocked

CVE-2016-9962 RunC Exec Vulnerability

SELinux Blocked

SELinux Goldilocks

What happens In Vegas stays in Vegas!



SELinux Confinement

- SELinux has blocked almost every Docker breakout so far
- Best tool to protect the file system from container escape.
- Allow container all access within container
  - Allow all capabilities
    - Let Linux capabilities controls them
  - Allow all network access
    - Let VPN and Firewall rules control

Problems with SELinux Confinement

- Volumes
  - Expose parts of OS Into Containers
  - Relabel content “z”, “Z”
    - podman run -v /var/lib/db:/var/lib/mariadb:Z mariadb
    - podman run -v /var/log:/var/log:Z fluentd
      - Bad idea, host apps will break
      - podman run --security-opt label=disabled

Moving toward Mama Bear without disabling SELinux Separation

udica

https://github.com/containers/udica

1. Examines container configuration
2. Generate SELinux policy
    - Allowing access volume types

Devconf.CZ Talk

Custom SELinux container policies in OpenShift

Sunday, January 26 • 11:00am - 11:55am

Moving toward Papa Bear

udica

https://github.com/containers/udica

1. Enables SELinux capability controls
2. Enables Network controls

Limiting the communications with the Kernel

Processes communicate with the kernel via SYSCALLS

SECCOMP Filters protect

/usr/share/containers/seccomp.json

- Allows 300 Linux Syscalls out of approximately of 450
- Eliminates all 32 bit syscalls
- Can we do better?

Limiting the communications with the Kernel

“The high number of available syscalls is essential to support as many containers as possible but according to Aqua Sec, most containers require only **40 to 70 syscalls**.“

https://podman.io/blogs/2019/10/15/generate-seccomp-profiles.html

Limiting the communications with the Kernel

- **Oci-seccomp-bpf-hook**
  - https://github.com/containers/oci-seccomp-bpf-hook
  - Generate seccomp profile, tracing syscalls made by container.

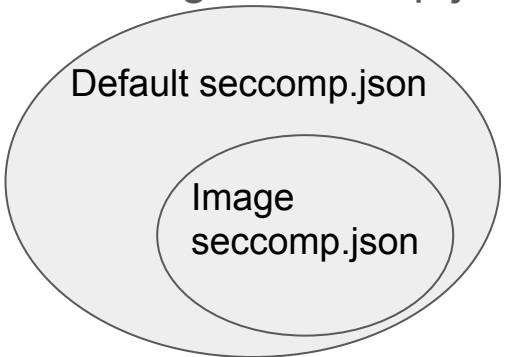
Devconf.CZ Talk
Generate seccomp profiles for containers using bpf
Saturday, January 25 • 5:00pm - 5:25pm

DEMO

How do we ship/use generated seccomp rules by default?

New Idea:

- Devloper of container image writes seccomp.json
  - Package it up into container image
- LABEL “io.containers.seccomp=/seccomp.json”
- Iff container image seccomp.json is subset of default seccomp.json
  - Container engine applies image seccomp.json automatically.



User Namespace Security

- Allows us to run containers as non-root
  - Rootless Podman
  - Rootless Buildah
- Rootless Builds inside container launched by Podman or CRI-O/Kubernetes
- Issue network distibution of /etc/subuid & /etc/subgid
  - We are making progress on this, potential sssd solution.

User Namespace Security

- Sadly still no one uses it for container separation

podman run --usermap 0:100000:5000 …
podman run --usermap 0:200000:5000 …

- Guarantee different user namespace for each container
- Still no Kubernetes support
- Lack of file system support
  - We are getting better with chown
    - Parallel chown shows promise
  - Shifting file system is moving forward

User Namespace Security

Possible Solution

- podman run --userns=auto
  - Podman automatically picks different User Namespace per Container, guaranteeing uniqueness.
  - Similar to what we do with SELinux
  - Allow administrator to turn this on by default
- Add similar feature to Kubernetes/CRI-O
  - Still have difficulty or chowning volumes to match User Namespace

Containers.conf

- Allow distributions/Administrators & Users to set default settings for containers.
  - /usr/share/containers/containers.conf
  - /etc/containers/containers.conf
  - $HOME/.config/containers.conf
- Including Default Capabilities.
  - Eliminate questionable Capabilities
- Default to allowing ping within your containers with sysctl
  - Default_sysctls

Recommended Container Engine Talks

Fri 12:30pm - Container Security BOF - what's next?
Fri 1:30pm - Understanding Container Engines by Demo™ y
Sat 10:30am - Podman, Buildah, Skopeo, and CRI-O A Year Later
Sat 12:30pm - Finding, Building, Sharing & Deploying Containers
Sat 2:30pm - Building multi-arch container images with buildah
Sat 4:30pm - From Terminal to Container: Tracing Podman Run
Sat 5:00pm - Generate seccomp profiles for containers using bpf
Sun 10:00am - Kubernetes BOF Josh Berkus
Sun 11:00am - Custom SELinux container policies in OpenShift
Sun 12:00pm - OCP+Fedora+VirtualKubelet+RPI3+Podman = Fun^2!
Sun 1:00pm - Containers Birds of a Feather