

# State of Container Security



**Urvashi Mohnani**  
**@umohnani8**

**Sally O'Malley**  
**@somalley108**



# Goldilocks and The Three Bears



# The Different Levels

Too Hard



# The Different Levels

Too Soft



Too Hard



# The Different Levels

Too Soft



Just Right



Too Hard



# No One Turns Up Security



- How many of you have ever done
  - `podman run --cap-add capability ...`
- How many of you have ever done
  - `podman run --privileged ...`
- How many of you have turned down security
  - `setenforce 0`
- How do we 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.**

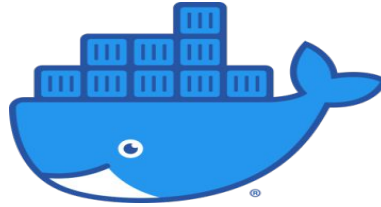


# OCI Images format





# Container Engines



# OCI Images format



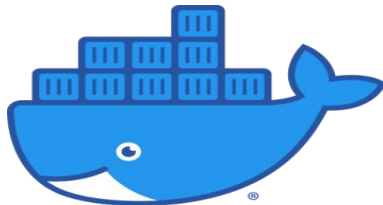
## Humans & Orchestrators



**kubernetes**



## Container Engines



## OCI Images format



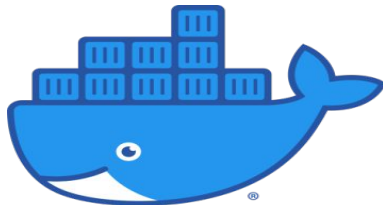
## Humans & Orchestrators



**kubernetes**

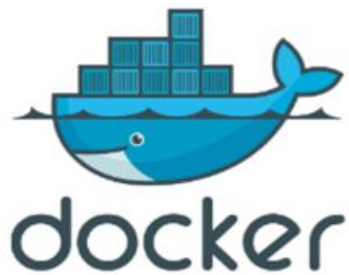


## Container Engines



## OCI Images format





**cri-o**



**podman**



**skopeo**



**buildah**

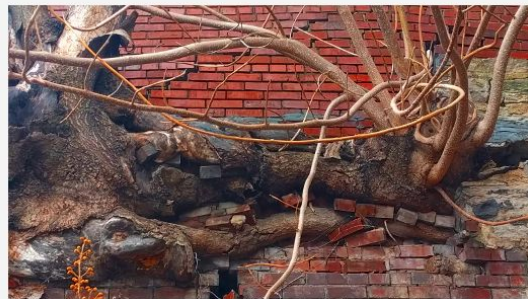


**OPEN** CONTAINER  
INITIATIVE

# 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](#)

When it comes to security, if it's easier to disable a feature than it is to configure it, chances are it will get disabled.

<https://www.grant.pizza/>



# Limiting the Power of Root: Capabilities

- Allow 14 out of 37 capabilities by default



# Limiting the Power of Root: Capabilities

- Allow 14 out of 37 capabilities by default
- Originally defined by upstream Docker Project





# Limiting the Power of Root: Capabilities

- Allow 14 out of 37 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

Demo!



# New Idea: Image Developer Specifies Capabilities

- Allow images to specify Capabilities as Image Annotations/Labels





# New Idea: Image Developer Specifies Capabilities

- Allow images to specify Capabilities as Image Annotations/Labels
- Example Label:
  - LABEL “io.containers.capabilities=**SETUID,SETGID**”



# New Idea: Image Developer Specifies Capabilities

- Allow images to specify Capabilities as Image Annotations/Labels
- Example 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



# New Idea: Image Developer Specifies Capabilities

- Allow images to specify Capabilities as Image Annotations/Labels
- Example 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 **SETUID, SETGID**



# New Idea: Image Developer Specifies Capabilities

Demo!



# Limiting the Communications with the Kernel

- How can we limit 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)

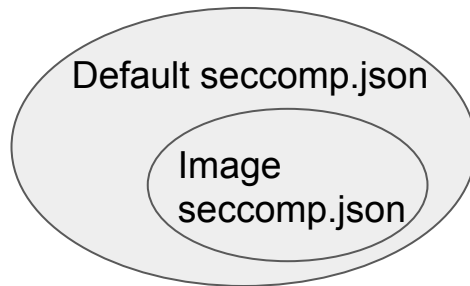
- <https://github.com/containers/oci-seccomp-bpf-hook>
- Generate seccomp profile by tracing container syscalls

Demo!



# New Idea!

- Can we ship/use generated seccomp rules by default?
- Container image developer generates seccomp.json
  - Package seccomp.json file into container image
- LABEL “io.containers.seccomp=/seccomp.json”
- If image seccomp.json is subset of default seccomp.json
  - Container engine applies image seccomp.json automatically.





the  
**SELINUX**  
COLORING BOOK

"It's raining cats and dogs!"

LEARN  
as you  
COLOR!



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 Confinement

- SELinux has blocked almost every container 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 control them
  - Allow all network access
    - Let VPN and Firewall rules control



# Problems with SELinux Confinement

- Volumes
  - Expose parts of OS Into Containers



# 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`

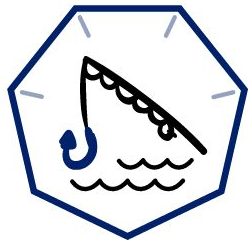


# Problems with SELinux Confinement

- 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 towards Mama Bear without Disabling SELinux Separation

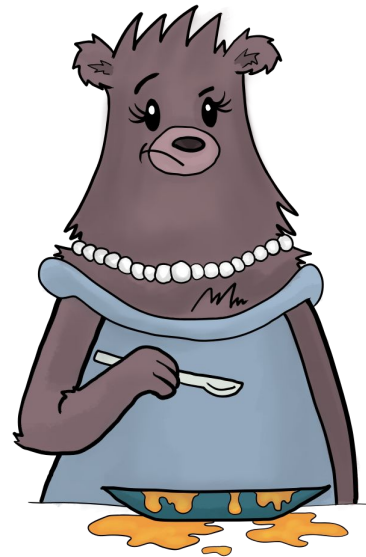


udica

<https://github.com/containers/udica>

- Examines container configuration
- Generate SELinux policy
  - Allowing access volume types

Demo!



# Moving towards Papa Bear



udica

<https://github.com/containers/udica>

- Enables SELinux capability controls
- Enables Network controls

Demo!





# User Namespace Security

- Allows us to run containers as non-root
  - Rootless Podman
  - Rootless Buildah



# User Namespace Security

- Allows us to run containers as non-root
  - Rootless Podman
  - Rootless Buildah
- Sadly still no one uses it for container separation



# User Namespace Security

So, we could guarantee a different user namespace for every container...

- Still no Kubernetes support
  - Still have difficulty or chowning volumes to match User Namespace



# User Namespace Security

So, we could guarantee a different user namespace for every container...

- Still no Kubernetes support
  - Still have difficulty or chowning volumes to match User Namespace
- 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 --usersns=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

Demo!



# 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`



# 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



# 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`



Demo!



# Additional Resources

- Demo Scripts: [github.com/containers/Demos](https://github.com/containers/Demos)
- Oci-bpf-hook: [github.com/containers/oci-seccomp](https://github.com/containers/oci-seccomp)
- [podman.io](https://podman.io)
- [buildah.io](https://buildah.io)
- Coloring books
  - [https://github.com/mairin/selinux-coloring-](https://github.com/mairin/selinux-coloring-book)
  - <https://github.com/mairin/coloringbook-container-commandos>

# Thank You!



**Urvashi Mohnani**

**@umohnani8**



**Sally O'Malley**

**@somalley108**



[linkedin.com/company/red-hat](https://linkedin.com/company/red-hat)



[youtube.com/user/RedHatVideos](https://youtube.com/user/RedHatVideos)



[facebook.com/redhatinc](https://facebook.com/redhatinc)



[twitter.com/RedHat](https://twitter.com/RedHat)