Using SELinux with container runtimes

Because privileged containers are scary

Lukas Vrabec
Senior Software Engineer
Security Technologies
Who am I?

- Lukas Vrabec
- SELinux Evangelist
- Member of Security Technologies team at Red Hat
- RHEL & Fedora Contributor (selinux-policy, xguest, udica, netlabel_tools)
- lukas.selinux@redhat.com
- https://lukas-vrabec.com
- https://github.com/wrabcak
- https://twitter.com/mynamewrabcak
Why?
New cluster:

- SELinux enforcing by default
New cluster:

- SELinux enforcing by default
- 189 pods
New cluster:

- SELinux enforcing by default
- 189 pods
  - 618 containers
New cluster:

- SELinux enforcing by default
- 189 pods
  - 618 containers
    - 134 privileged containers
New cluster:

- SELinux enforcing by default
- 189 pods
  - 618 containers
    - 134 privileged containers
Privileged containers are scary
What are privileged containers anyway?
Why is SELinux important at all?
Quick SELinux introduction
TECHNOLOGY FOR **PROCESS ISOLATION** TO MITIGATE ATTACKS VIA PRIVILEGE ESCALATION
CONTAINER_T CONTAINER_FILE_T
ARE LABELS
ASSIGNED TO PROCESSES
ASSIGNED TO PROCESSES

ASSIGNED TO SYSTEM RESOURCES
ASSIGNED TO PROCESSES
ASSIGNED TO SYSTEM RESOURCES
BY SELINUX SECURITY POLICY
LABELS IN REALITY
STORED IN EXTENDED ATTRIBUTES OF FILE SYSTEMS - EXT2, EXT3, EXT4 ...
# getfattr -n security.selinux /etc/passwd
getfattr: Removing leading '/' from absolute path names
file: etc/passwd
security.selinux="system_u:object_r:passwd_file_t:s0"

# ls -Z /etc/passwd
system_u:object_r:passwd_file_t:s0 /etc/passwd
$ ps -eZ | grep container_t

system_u:system_r:container_t:s0:c435,c872 17864 pts/0 00:00:00 bash

system_u:system_r:container_t:s0:c236,c541 17865 pts/0 00:00:00 bash

system_u:system_r:container_t:s0:c123,c456 17866 pts/0 00:00:00 bash
SELINUX POLICY DESCRIBES AN INTERACTION BETWEEN PROCESSES AND SYSTEM RESOURCES
allow container_t container_file_t::file {getattr open read};
BY DEFAULT EVERYTHING IS DENIED AND YOU DEFINE POLICY RULES TO ALLOW CERTAIN REQUESTS.
Generic container
SELinux policy
Protects the host system from container processes
Protects the host system from container processes
Container processes can only read/execute /usr files
Protects the host system from container processes
Container processes can only read/execute /usr files
Container processes only write to container files.
Protects the host system from container processes
Container processes can only read/execute /usr files
Container processes only write to container files.

process type - `container_t`
file type - `container_file_t`
Every Container Runtime CVE container breakout was a file system breakout.

CVE-2019-5736 Execution of malicious containers allows for container escape and access to host filesystem

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 has contained them ALL.
What about containers attacking each other?
Multi Category Security
Based on MLS
(Multi Level Security)
• container_t:s0:c1,c2
  ○ container_file_t:s0:c1,c2
  ○ container_file_t:s0

• container_t:s0:c2,c3
  ○ container_file_t:s0:c2,c3
  ○ container_file_t:s0
• container_t:s0:c1,c2  
  ○ container_file_t:s0:c1,c2  
  ○ container_file_t:s0  

• container_t:s0:c2,c3  
  ○ container_file_t:s0:c2,c3  
  ○ container_file_t:s0  

Relabeling in container engines:

# podman run -d -v /var/lib/mydb:/var/lib/mariadb:Z rhel7-mariadb
• container_t:s0:c1,c2
  ○ container_file_t:s0:c1,c2
  ○ container_file_t:s0

• container_t:s0:c2,c3
  ○ container_file_t:s0:c2,c3
  ○ container_file_t:s0

Relabeling in container engines:
# podman run -d -v /var/lib/mydb:/var/lib/mariadb:Z rhel7-mariadb
# podman run -ti -v /home/lvrabec/shared:/home/lvrabec/shared:z fedora /bin/sh
# podman run -ti -v /home/lvrabec/shared:/home/lvrabec/shared:z fedora /bin/sh
Problems with SELinux Container Confinement
Default Container Type (container_t) too strict for certain use cases, e.g:
Default Container Type (container_t) too strict for certain use cases, e.g:

- Fedora SilverBlue project needs containers to read/write home directory
Default Container Type (container_t) too strict for certain use cases, e.g:

- Fedora SilverBlue project needs containers to read/write home directory
- Fluentd project needs containers to be able to read logs in /var/log directory
Default Container Type (container_t) too loose for certain use cases, e.g:
Default Container Type (container_t) too loose for certain use cases, e.g:

- No SELinux Network Controls
  - All container processes can bind to any network port
Default Container Type (container_t) too loose for certain use cases, e.g:

- No SELinux Network Controls
  - All container processes can bind to any network port
- No SELinux control on Linux Capabilities
  - All container processes can use all linux capabilities
Current Situation
# podman run -d -v /var/log:/var/log:Z fluentd

- BAD: Tells podman to set labels on /var/log directory to be container specific.
- Other confined tools will no longer be able to write their logs
# podman run -d -v /var/log:/var/log:Z fluentd

- **BAD**: Tells podman to set labels on /var/log directory to be container specific.
- Other confined tools will no longer be able to write their logs

# podman run -ti -v /home:/home --security-opt label:disabled fedora sh

- Turn off SELinux container separation for these use cases
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.
• Solutions
  ○ Write completely new SELinux policy for custom container
    ■ Best solution
    ■ Too difficult for system administrators
      ● SELinux expertise required
• **Solutions**
  ○ Write completely new SELinux policy for custom container
    ■ Best solution
    ■ Too difficult for system administrators
      ● SELinux expertise required
  ○ Add additional rules for container_t type
    ■ Not ideal still difficult for system administrators
    ■ Rules apply to all containers, not just specific container.
Solution: Udica Project
Udica ~ Fishing rod
Udica is a tool for generating SELinux security profiles for containers.
● Example container
  ○ Mounting /home as read/write
  ○ Mounting /var/spool as read only
  ○ Exposing port tcp/21
- Example container
  - Mounting /home as read/write
  - Mounting /var/spool as read only
  - Exposing port tcp/21

- Generic SELinux domain for container
  - Cannot read/write /home
  - Cannot read /var/spool
  - Exposes all ports
Let’s generate SELinux policy for example container!

Live Demo!

https://github.com/containers/Demos/tree/master/security/SELinuxUdica
Using udica to solve these issues

```bash
# podman run -v /home:/home:ro -v /var/spool:/var/spool:rw -p 21:21 -it fedora bash

# podman inspect -l | udica my_container

# semodule -i my_container.cil
/usr/share/udica/templates/{{base_container.cil,net_container.cil,home_container.cil}}

# podman run --security-opt label=type:my_container.process -v /home:/home:ro -v /var/spool:/var/spool:rw -p 21:21 -it fedora bash
```
Using udica to solve these issues

```bash
# ps -efZ | grep my_container.process

unconfined_u:system_r:container_runtime_t:s0-s0:c0.c1023 root 8837 5865 0 14:29 pts/0 00:00:00 podman run
--security-opt label=type:my_container.process -v /home:/home:ro -v /var/spool:/var/spool:rw -p 21:21 -it fedora bash

system_u:system_r:my_container.process:s0:c116,c171 root 8920 8909 0 14:29 pts/0 00:00:00 bash
```
Udica under the hood
• Concept based on "block inheritance" SELinux CIL language
- Concept based on "block inheritance" SELinux CIL language
- Udica creates policy combining rules from specified CIL blocks (templates)
  - Inspecting container JSON file
    - Mounts
    - Ports
    - Capabilities
- Concept based on "block inheritance" SELinux CIL language
- Udica creates policy combining rules from specified CIL blocks (templates)
  - Inspecting container JSON file
    - Mounts
    - Ports
    - Capabilities
  - Combines with default container template file
    - /usr/share/udica/templates/base_container.cil
Contains read/exec /usr & read /etc

Allows network access

Allows access homedirs

*Required for every container*

+ Allowing bind on ftp_port_t (21)

+ Add only read/write perms
base  net

home

my_container
- No block for /var/spool

- Udica will detect all labels what could be inside /var/spool and create allow rules in my_container policy.
Using the new type with container runtimes
# podman run --security-opt label=type:my_container.process -v /home:/home:rw -v /var/spool:/var/spool:ro -p 21:21 -it fedora bash
# docker run --security-opt label=type:my_container.process -v /home:/home:rw -v /var/spool:/var/spool:ro -p 21:21 -it fedora bash
# buildah bud --security-opt label=type:my_container.process -f Dockerfile .
apiVersion: v1
kind: Pod
metadata:
  name: udica-demo
spec:
  containers:
  - name: udica
    image: gcr.io/google-samples/node-hello:1.0
    securityContext:
      seLinuxOptions:
        type: "my_container.process"
## QUESTIONS?

<table>
<thead>
<tr>
<th></th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demo</td>
<td><a href="https://github.com/demos/SELinuxEscape">https://github.com/demos/SELinuxEscape</a></td>
</tr>
<tr>
<td>Udica</td>
<td><a href="https://github.com/containers/udica">https://github.com/containers/udica</a></td>
</tr>
<tr>
<td>Podman</td>
<td><a href="https://podman.io">https://podman.io</a></td>
</tr>
<tr>
<td>Generic SELinux policy</td>
<td><a href="https://github.com/containers/container-selinux">https://github.com/containers/container-selinux</a></td>
</tr>
<tr>
<td>Udica PoC</td>
<td><a href="https://github.com/fedora-selinux/container-selinux-customization">https://github.com/fedora-selinux/container-selinux-customization</a></td>
</tr>
</tbody>
</table>
Thank you

Red Hat is the world’s leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.