Red Hat

Freedom From Interference (FFI)

Douglas Schilling Landgraf
Container On Wheels Engineer @ Red Hat
QM Maintainer
<dougsland@redhat.com>

Yariv Rachmani
Principal Quality Software Engineer @ Red Hat
QM Maintainer
<yrachman@redhat.com>

Boston Thu Aug 15 2024

Red Hat

Red Hat

Agenda

- What's Freedom From Interference?
- Automotive Distribution and FFI
- Podman, cgroups, bluechi and systemd
- Demo

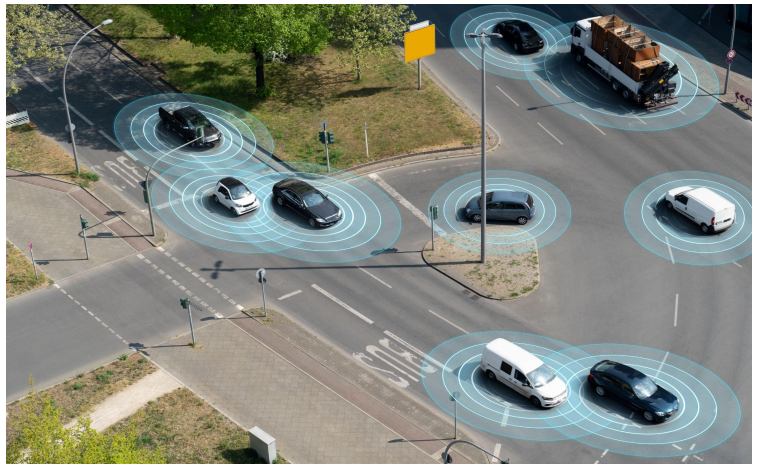
Red Hat

Quick start:

Why FFI is important for for critical systems?

"Ensuring that safety-critical components operate independently and are not disrupted by faults or unintended interactions from other system components"

How about real examples of standards that incorporates FFI?



ISO 26262 *is an international functional safety standard for the development of electrical and electronic systems in road vehicles. It defines guidelines to minimize the risk of accidents and ensure that automotive components perform their intended functions correctly and at the right time.*

DO-178C *is to ensure that safety-critical software in airborne systems is developed to a high level of safety and reliability to reduce the risk of accidents or incidents caused by software failures.*

ARP4754*, Aerospace Recommended Practice ARP4754B, is a guideline from SAE International, dealing with the development processes which support certification of Aircraft systems, addressing "the complete aircraft development cycle, from systems requirements through systems verification*

IEC 62304 *is an international standard that specifies life cycle requirements for the development of medical software and software within medical devices.*

How complex it can be to create a car OS?

Government Regulations

↓

Functional Safe (FuSa)

General safety principles applicable across various industries

↓

ISO 26262 (Automotive)

↓

Operational System

Automotive Safety Integrity Level (ASIL) **Quality Management (QM)**

An accredited third-party entity validates and certifies whether the operating system complies with all safety standards.

Inspectors in the datacenter

Safety requirements

How complex it can be to create a car OS?

Tire pressure
Updates on the air
Cruise Control
Sensors
Cameras
Radio AM/FM
News
PodCast
Spotify
Car Battery
Google Assistant
Trailer Lights
Apple Car
Android Auto

User profile
3rd vendor software integration
Wifi Client
bluetooth
Wireless charger
Google Maps
Climate
YouTube
Safety security settings

Car Manufacturer's End to end vehicle software platform

Platform running on: Automobile Operational System based on Linux

podman
systemd
SELinux

composefs crun qm bluechi glibc kernel cgroups

Red Hat

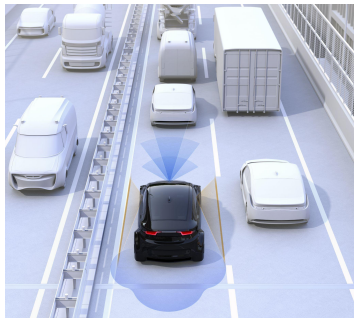
Safety Requirements: ASIL

What's Automotive Safety Integrity Level (ASIL)?

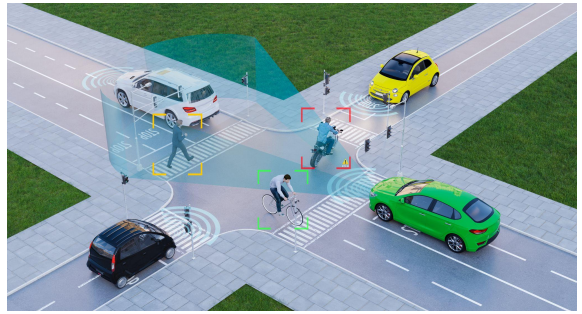
ASIL A	ASIL B	ASIL C	ASIL D
Lowest safety Failure could cause only minor injuries .	Moderate safety Failure could lead to significant but non-life-threatening injuries.	High safety Failure could cause severe injuries or possibly death.	Highest safety Failure is likely to result in life-threatening injuries or fatalities

Examples of ASIL Services

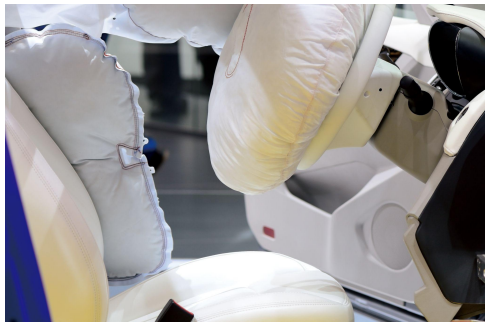
Collision Warning Systems



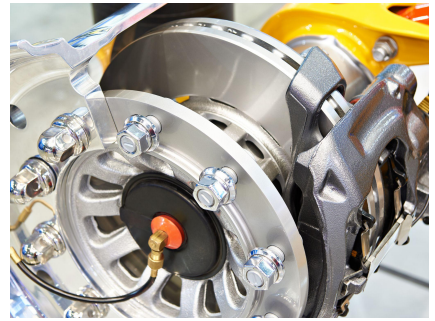
Blind Spot Detection Systems



Airbag Control Systems



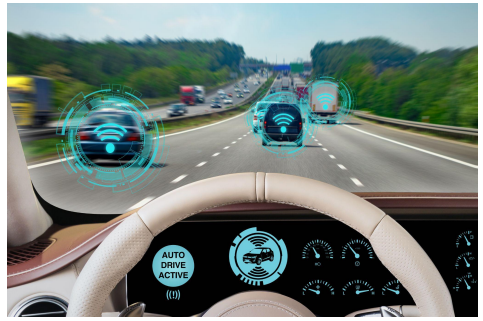
Brakes Systems



Driver Drowsiness Detection Systems

Rear-View Camera and Parking Assistance

Advanced Driver Assistance Systems



Tyre Pressure Systems

Red Hat

Safety Requirements: QM

What's Quality Management?

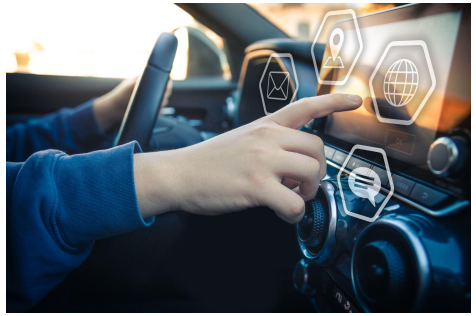
Red Hat

Examples of QM Services

Navigation Systems

Climate Control Systems

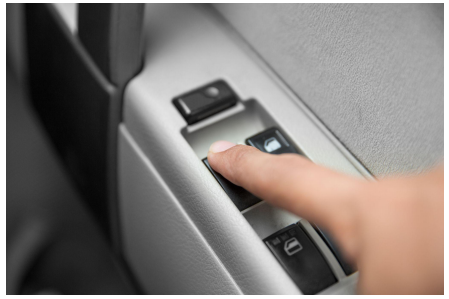
Infotainment Systems



Power Seats

Interior Lighting Systems

Power Window



Red Hat

How complex it to have Linux based car OS?

<irony>
It seems simple as rocket science, doesn't it?
</irony>

Red Hat

Automotive Distro and FFI

CentOS

Automotive Special Interest Group (SIG)

- It is Linux
- Openness and community support
- Freedom From Interference
- Simple configuration

Red Hat

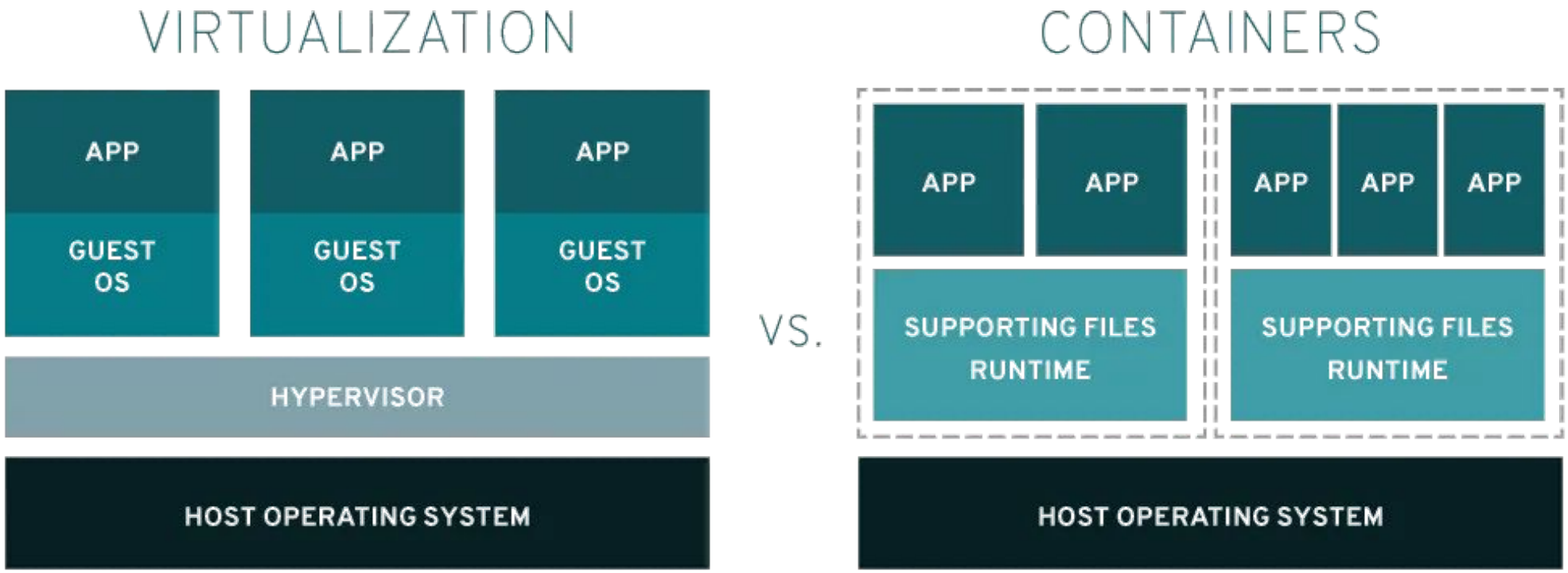
Automotive Distro and FFI

CentOS
Automotive SIG

- No forms to fill out or signatures required for a truly open-source distro!
- Code **IS available** in gitlab, github repos
- ALL in ONE Image
  - Download nightly images
  - Build your own images

Red Hat

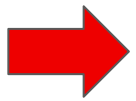
FFI: Linux isolation models

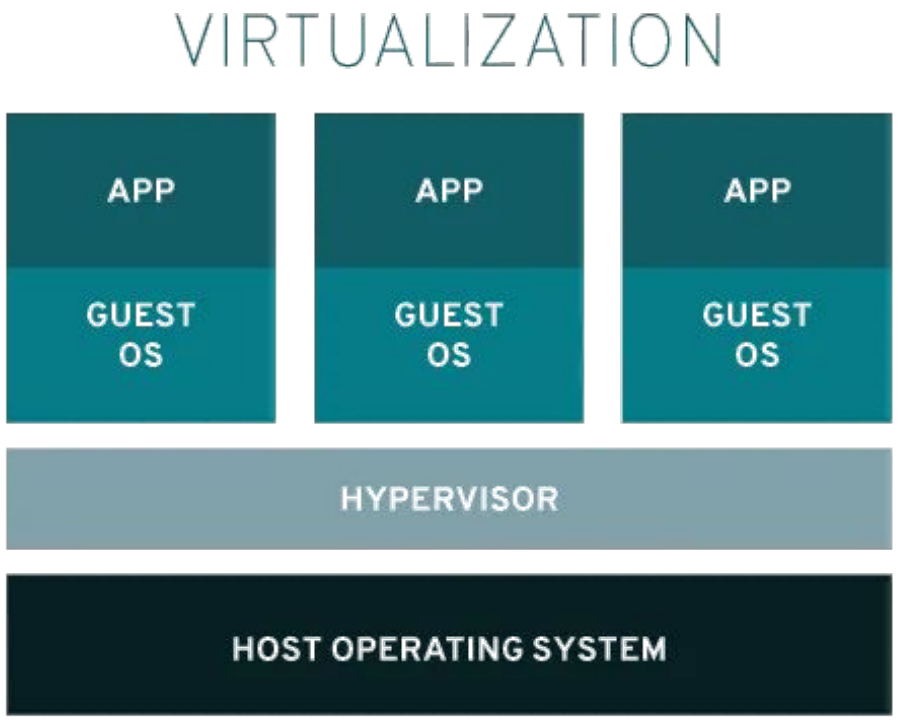


Red Hat

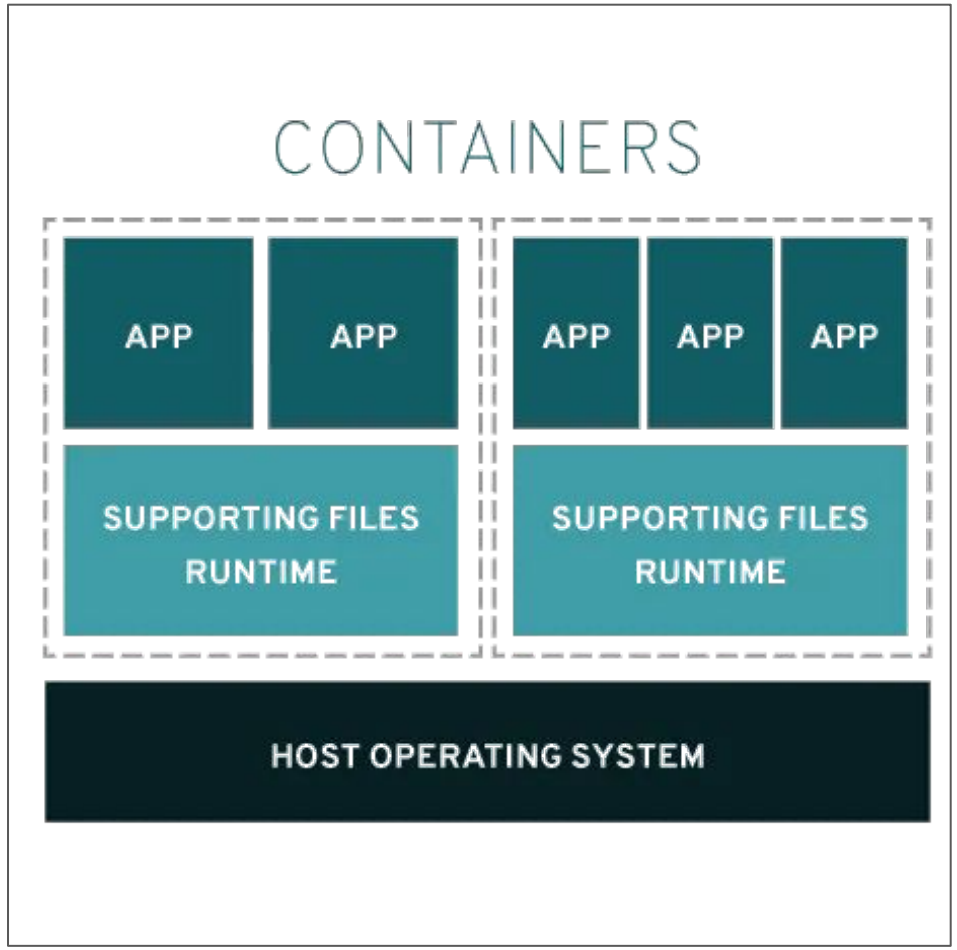
FFI: Linux isolation

- **Additional layer**
- **Hardware Emulation**





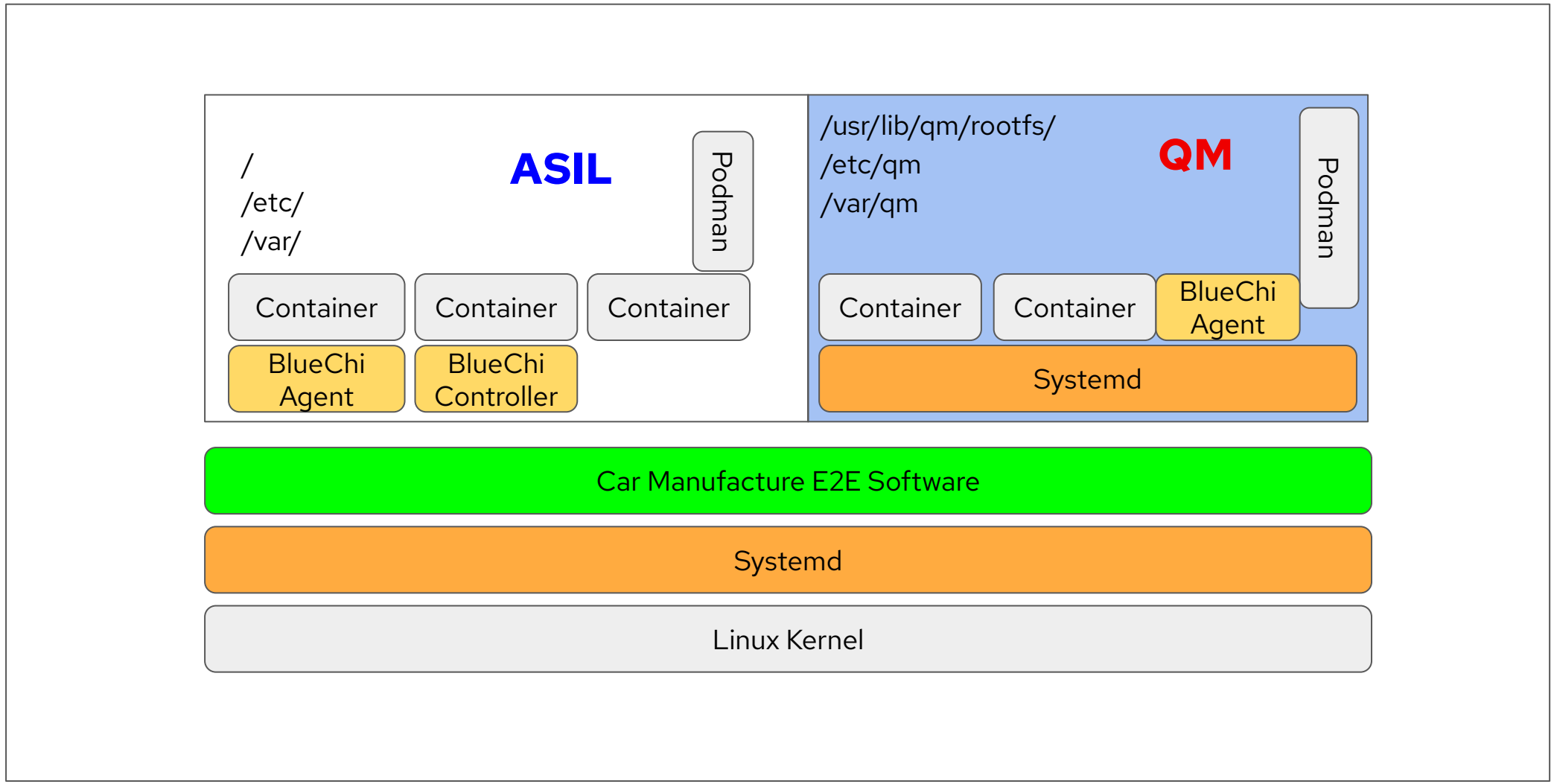
VS.



Red Hat

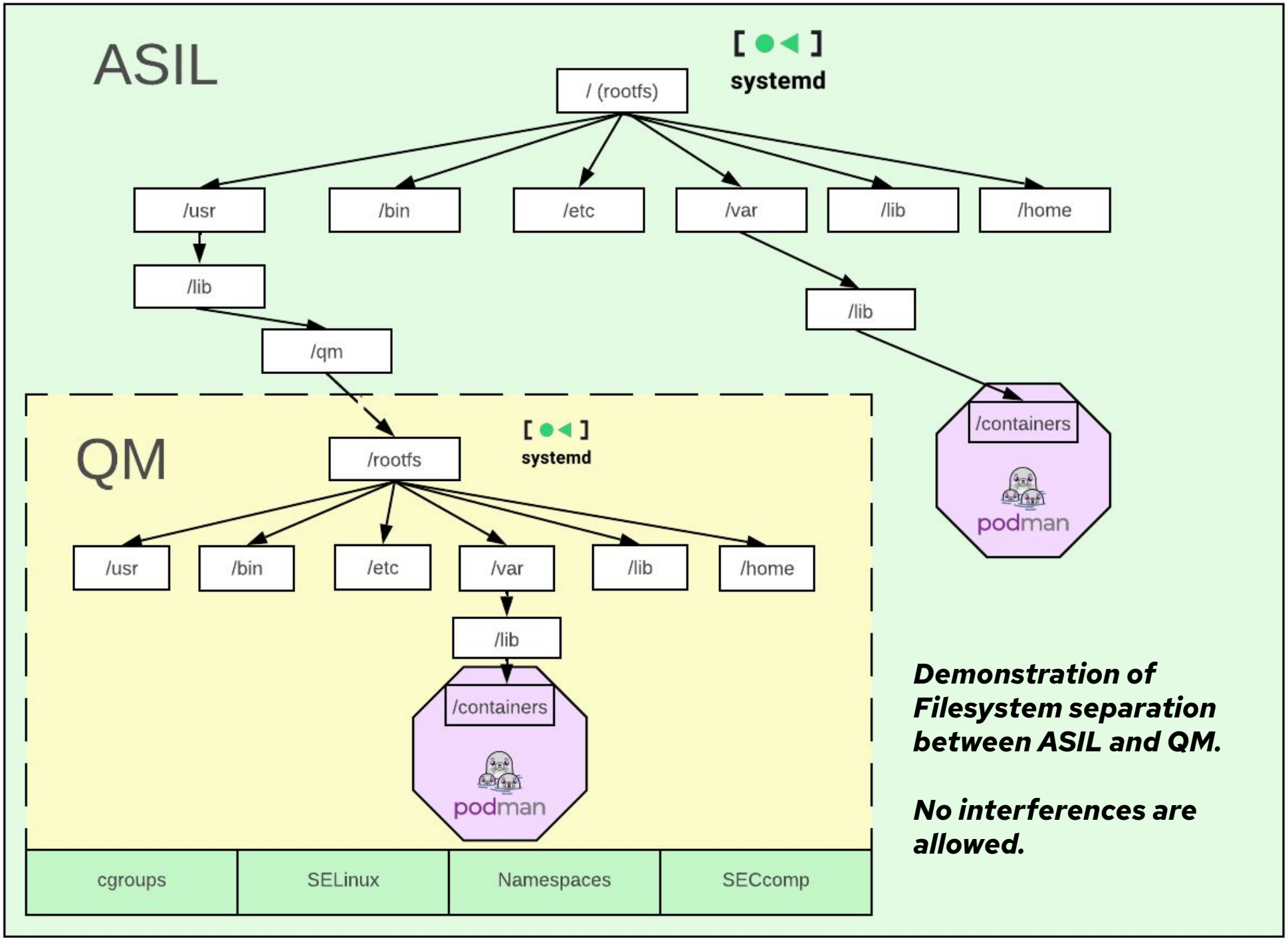
If there is no OpenShift, what's the Architecture?

FFI: Componentes view



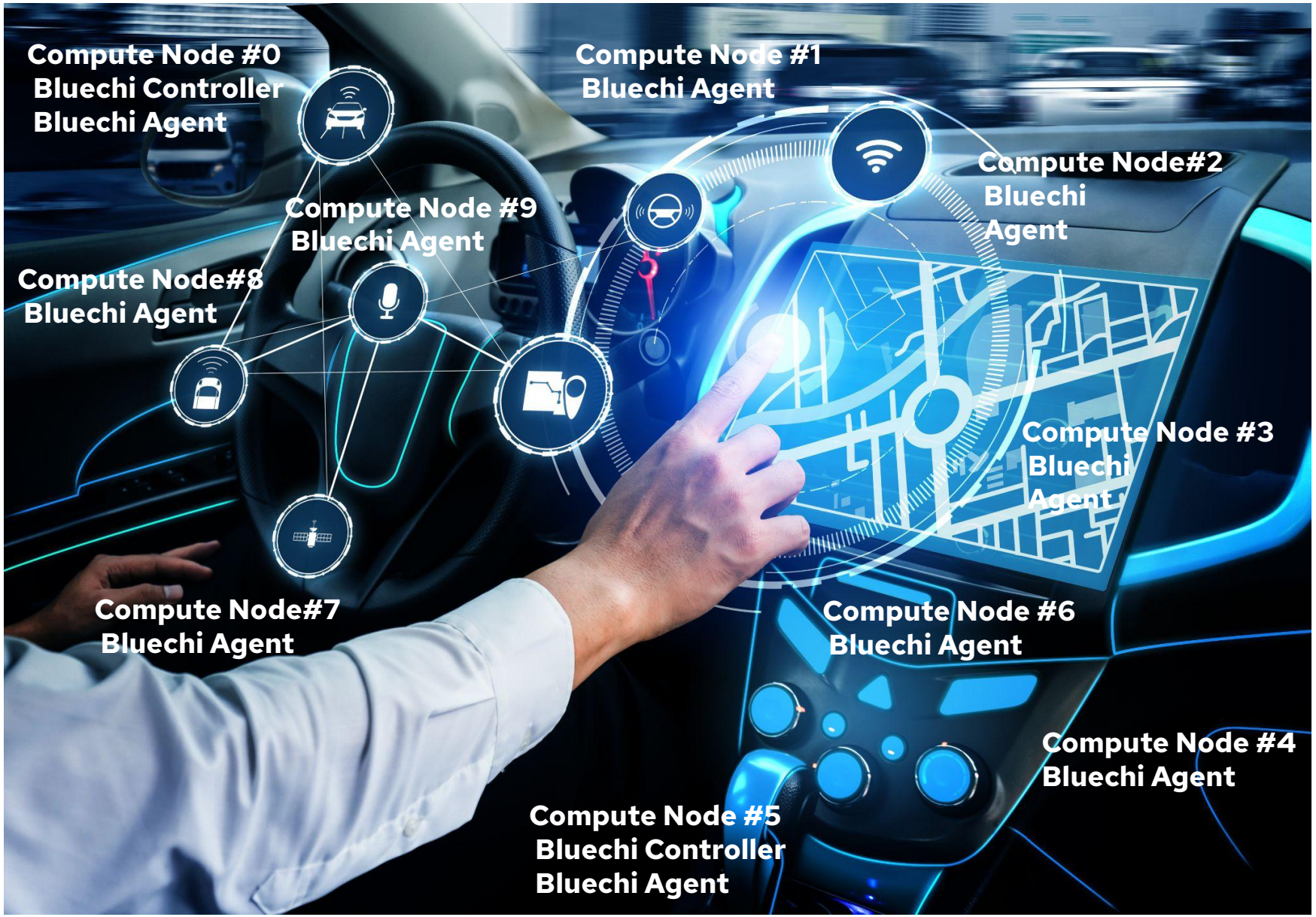
Design can be adjusted according to car manufacturer requirements

Red Hat


ASIL
systemd
/ (rootfs)
/usr
/bin
/etc
/var
/lib
/home
/lib
/qm
/lib
/containers
podman
QM
systemd
/rootfs
/usr
/bin
/etc
/var
/lib
/home
/lib
/containers
podman
cgroups
SELinux
Namespaces
SECcomp
Demonstration of Filesystem separation between ASIL and QM.
No interferences are allowed.

Red Hat

Bluechi – Node's Controller and Agent for Services



Red Hat

FFI: Linux Systemd (Phase 1)

initramfs boot

Systemd loading services

i.e (bluechi controller, bluechi agent)

Containers started via **Podman Quadlet**

Red Hat

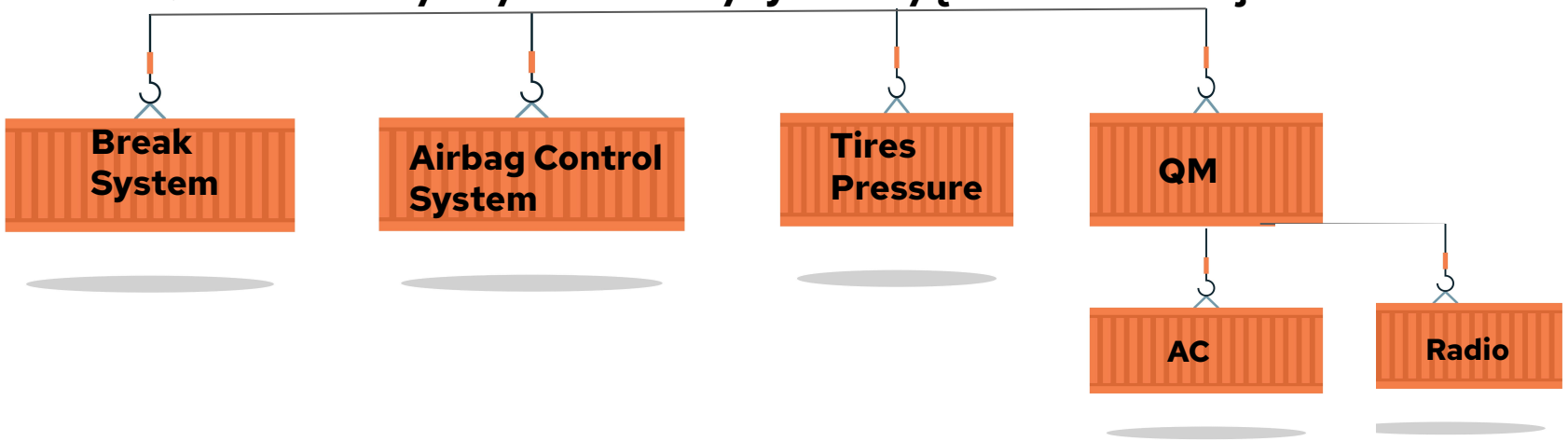
FFI: Linux Resource Mmt (Phase 2 – podman's view)

2.1 **podman run:** systemd reads quadlet files and initialize a new process for the container.

podman

Selects Container Runtime for run the containers: ***crun***

Quadlet files: /etc/containers/systemd/{files.container}



Red Hat

FFI: Linux Resource Mmt (Phase 2 – podman's view)

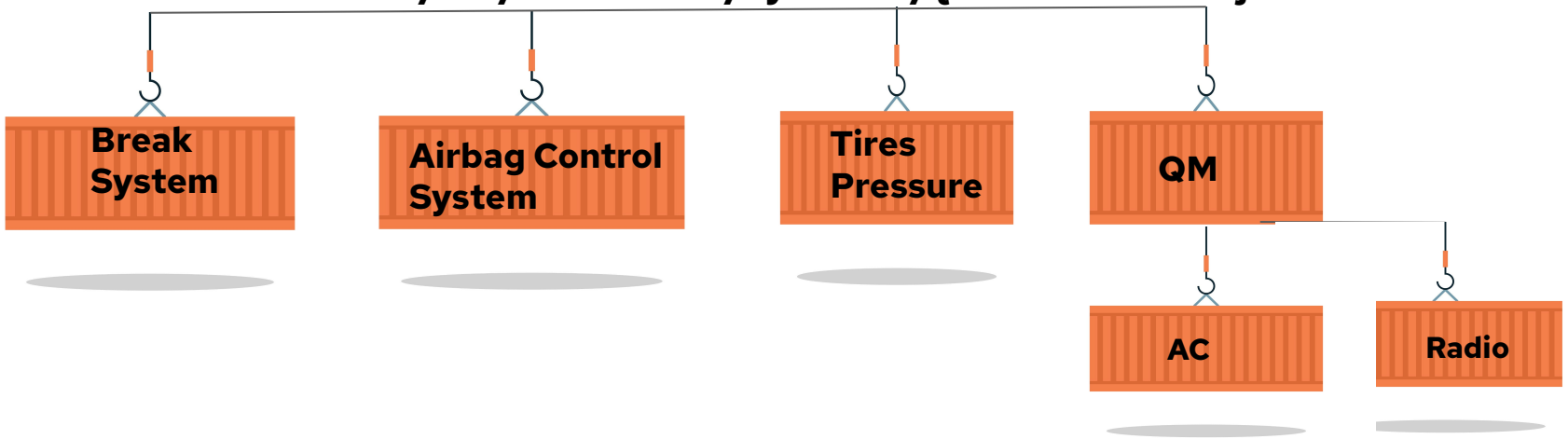
podman

Selects Container Runtime for run the containers: ***crun***

2.1 **podman run:** systemd reads quadlet files and initialize a new process for the container.

2.2 **podman requests kernel services:** Podman uses system calls (`clone`, `unshare`, etc.) to request the Linux kernel to create a new process and place it in specific namespaces (PID, network, mount) for isolation.

Quadlet files: /etc/containers/systemd/{files.container}



Red Hat

FFI: Linux isolation (Phase 3 – kernel's simplest view)

3.1

The kernel's **`namespace`** and **`cgroup`** subsystems handle the process's assignment to the appropriate namespaces and cgroups, ensuring isolation and resource limits.

Cgroup v2 (Control group) →

Namespaces
IPC Namespace (Isolate Interprocess)
PID Namespace (Process Isolation)
Mount Namespace (Isolate FileSystem)
UTS Namespace (Isolate Hostname)
Network Namespace (Isolate Network)
User Namespace (Isolate Users IDs & Permissions)

erly

Red Hat

FFI: Linux isolation (Phase 3 – kernel's simplest view)

3.1 The kernel's **`namespace`** and **`cgroup`** subsystems handle the process's assignment to the appropriate namespaces and cgroups, ensuring isolation and resource limits.

3.2 **SELinux** applies mandatory access control policies to the process, restricting its permissions based on security contexts.

Cgroup v2 (Control group)

Let me see what's going on in these syscalls, pointers, structs...

Security Layer

SELinux

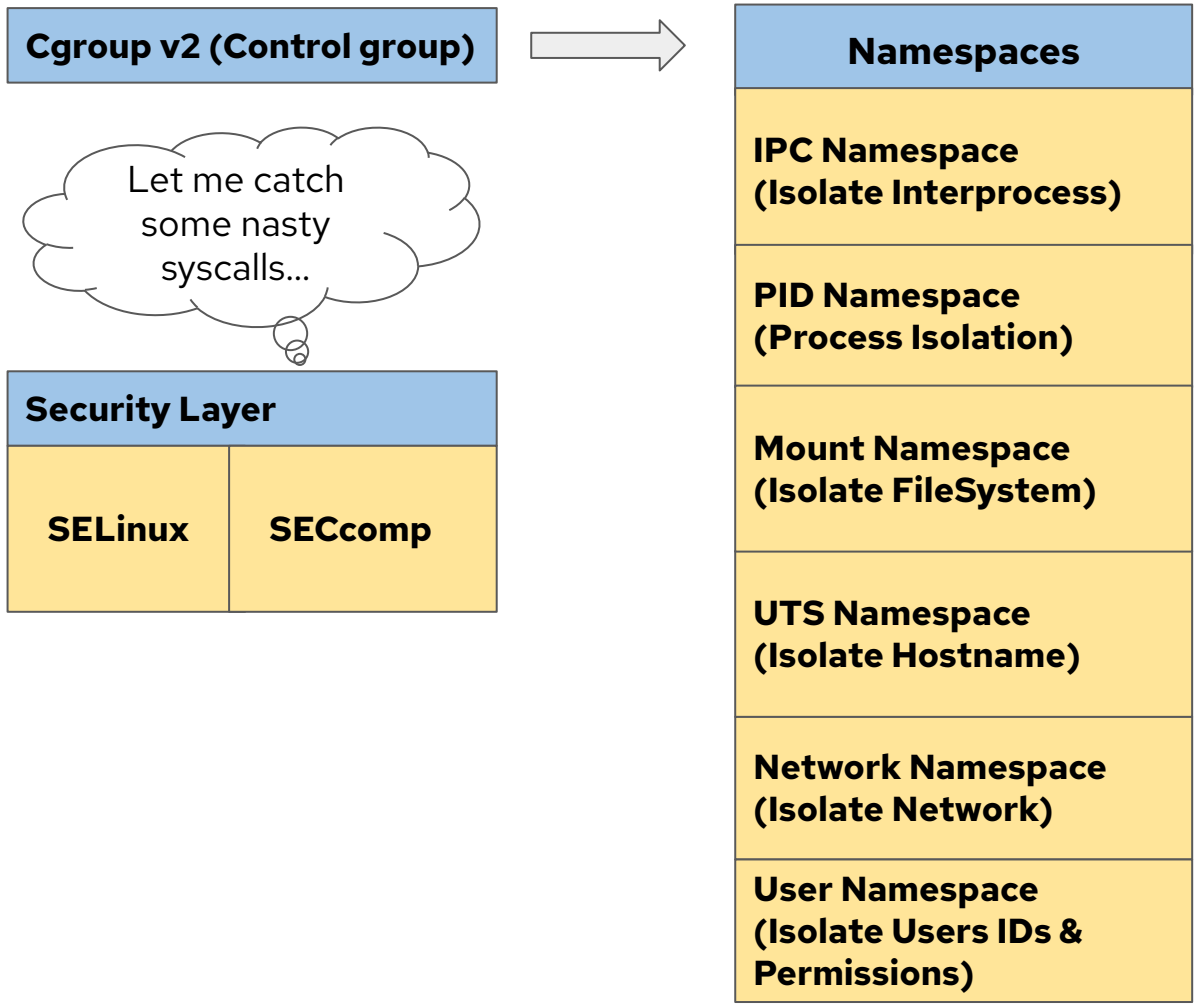
Namespaces
IPC Namespace (Isolate Interprocess)
PID Namespace (Process Isolation)
Mount Namespace (Isolate FileSystem)
UTS Namespace (Isolate Hostname)
Network Namespace (Isolate Network)
User Namespace (Isolate Users IDs & Permissions)

FFI: Linux isolation (Phase 3 – kernel's simplest view)

3.1 The kernel's **`namespace`** and **`cgroup`** subsystems handle the process's assignment to the appropriate namespaces and cgroups, ensuring isolation and resource limits.

3.2 **SELinux** applies mandatory access control policies to the process, restricting its permissions based on security contexts.

3.3 **seccomp** filters the system calls the process can make, enhancing security by blocking potentially dangerous operations.



erly

Red Hat

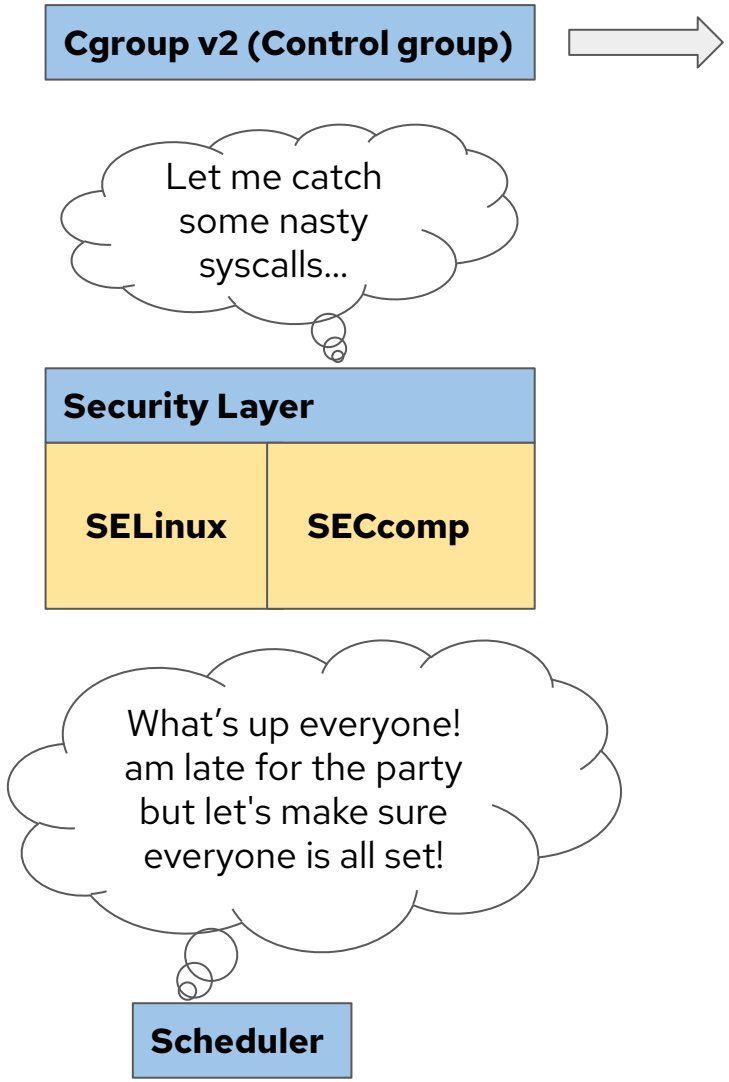
FFI: Linux isolation (Phase 3 – kernel's simplest view)

3.1 The kernel's **`namespace`** and **`cgroup`** subsystems handle the process's assignment to the appropriate namespaces and cgroups, ensuring isolation and resource limits.

3.2 **SELinux** applies mandatory access control policies to the process, restricting its permissions based on security contexts.

3.3 **seccomp** filters the system calls the process can make, enhancing security by blocking potentially dangerous operations.

3.4 **Kernel scheduler** enforces resource limits from cgroup .



Namespaces
IPC Namespace (Isolate Interprocess)
PID Namespace (Process Isolation)
Mount Namespace (Isolate FileSystem)
UTS Namespace (Isolate Hostname)
Network Namespace (Isolate Network)
User Namespace (Isolate Users IDs & Permissions)

erly

Red Hat

Is Linux tested to satisfy and mitigate risk analysis for automotive?

podman

Red Hat

Sample Risk Analysis according to ISO 26262:

SELinux + QM Blocking nested containers attack from "stealing CPU priority"

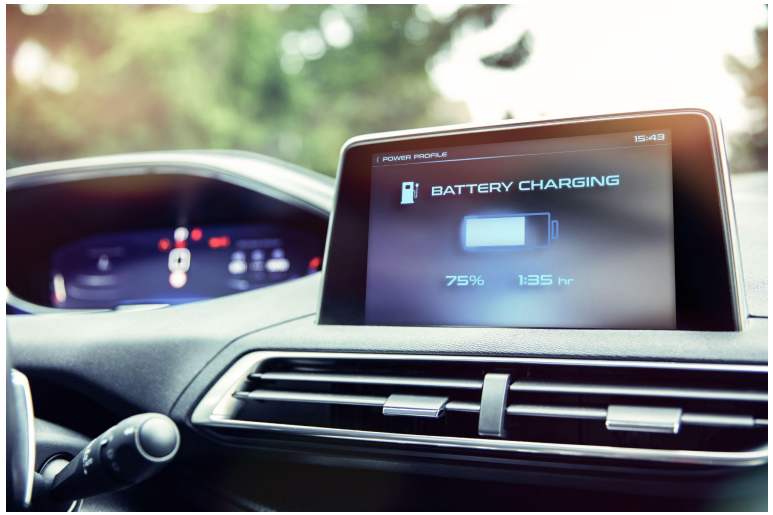
Let's image the hacker is smart enough and is able to break the initial security layers and it's ALSO able to connect to a nested container **as root**....

Red Hat

Sample Risk Analysis according to ISO 26262:

SELinux + QM Blocking nested containers attack from "stealing CPU priority"

Next step: Deploy the crypto miner and steal all CPUs priority to mine while the car is in charge mode and send it to his/her digital wallet. (from 9PM until 5AM - owner is sleeping)




```
# subZer0> ./make-me-rich
10:24:45 - reading the system .......
10:24:46 - Setting make-me-rich as daemon and hiding files ....
10:24:47 - collecting current OS scheduler  ........
10:24:48 - waiting car be in charge mode .....
.......
21:55:51  - Car is now connected to be charge ...
21:55:52 - +++ make-me-rich mode starting +++
21:56:53 - +++ reading the current scheduler policy +++
21:56:54 - +++ Setting priority scheduler policy to make-me-rich...
FAILED, unable to access Operational System system call
```

Sample Risk Analysis according to ISO 26262:

SELinux + QM Blocking nested containers attack from "stealing CPU priority"

Lets understand what just happened.....

Nested Container

subZer0> ./make-me-rich
10:24:45 - reading the system .....
10:24:46 - Setting make-me-rich as daemon and hiding files ....
10:24:47 - collecting current OS scheduler .....
10:24:48 - waiting car be in charge mode .....
.....
21:55:51 - Car is now connected to be charge ...
21:55:52 - +++ make-me-rich mode starting +++
21:56:53 - +++ reading the current scheduler policy +++
21:56:54 - +++ Setting priority scheduler policy...
FAILED, unable to access Operational System system call

ASIL host (side)

journalctl -r
<SNIP>
SELinux is preventing **make-me-rich** from map access on the file /usr/lib64/ld-linux-x86-64.so.2.
... avc: denied { map } <----- HERE
....avc: denied { read } <----- HERE

Red Hat

Sample Risk Analysis according to ISO 26262:

SELinux + QM Blocking nested containers attack from "stealing CPU priority"

But guess what, let's keep with our imagination....

For some reason, the trainee Disabled SELinux in that car model for tests and all car models got updated from the cloud image... OH NO! :-/

Let's simulate this situation setting the the car OS to permissive mode

```
[root@RHIVOS-carOS ~]# setenforce 0
```

Sample Risk Analysis according to ISO 26262:

SELinux + QM Blocking nested containers attack from "stealing CPU priority"

```
# subZer0> ./make-me-rich

21:55:51  - Car is now connected to charged ...
21:55:52 - +++ make-me-rich mode starting +++
21:56:53 - +++ reading the current scheduler policy +++
21:56:54 - +++ Setting priority scheduler policy to
make-me-rich: steal_cycles_sched_deadline failed to
boost pid 0: Operation not permitted
```

Sample Risk Analysis according to ISO 26262:

SELinux + QM Blocking nested containers attack from "stealing CPU priority"

```
# subZer0> ./make-me-rich

21:55:51  - Car is now connected to charged ...
21:55:52 - +++ make-me-rich mode starting +++
21:56:53 - +++ reading the current scheduler policy +++
21:56:54 - +++ Setting priority scheduler policy to
make-me-rich: steal_cycles_sched_deadline failed to
boost pid 0: Operation not permitted
```

BUT WHO SAVED THE DAY?

Sample Risk Analysis according to ISO 26262:

SELinux + QM Blocking nested containers attack from "stealing CPU priority"

ME ?

Red Hat

Sample Risk Analysis according to ISO 26262:

SELinux + QM Blocking nested containers attack from "stealing CPU priority"

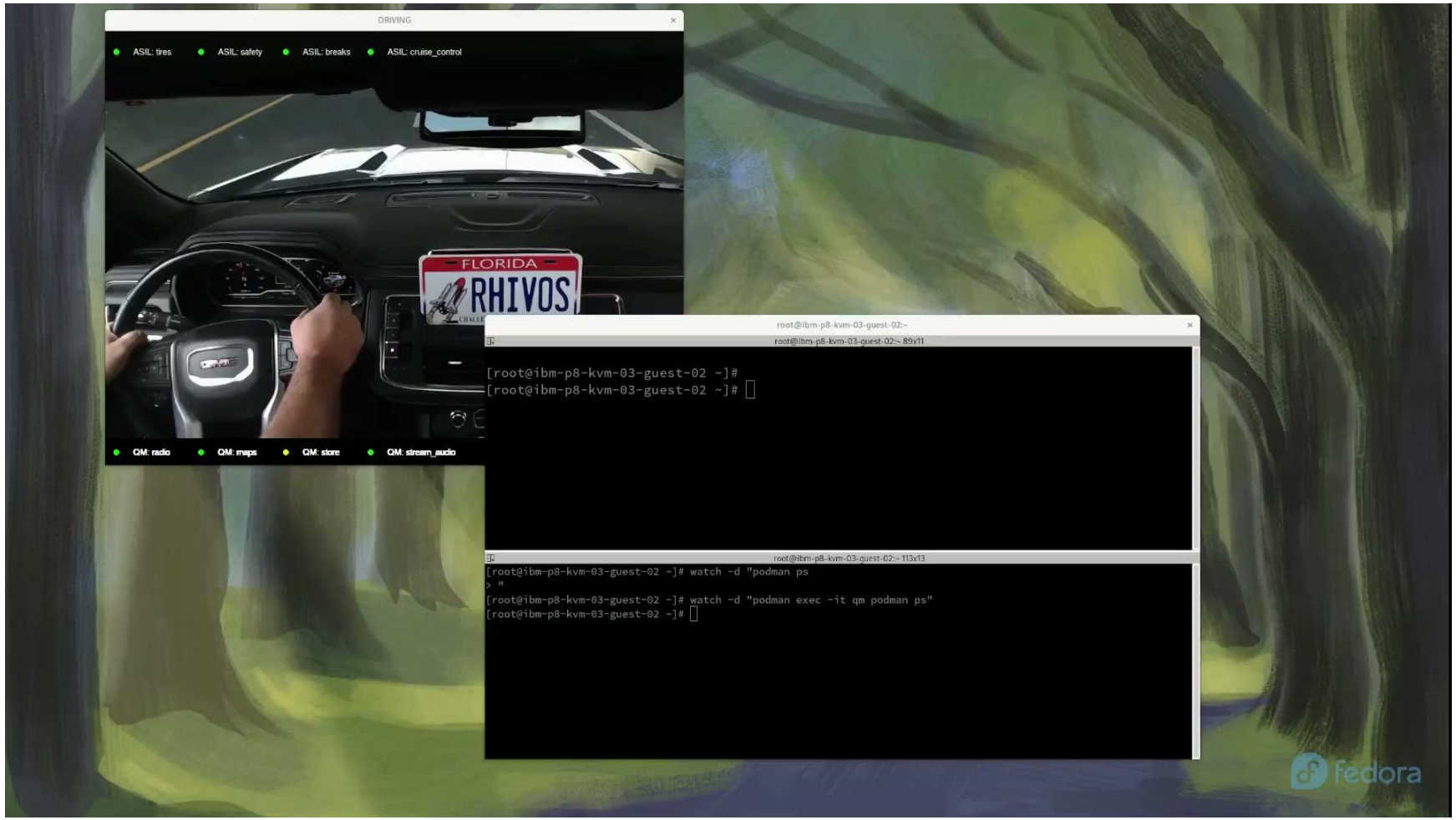
"Several layers of security..."

+ SECComp

podman

Seccomp is a Linux kernel feature that provides a way to filter and limit the system calls available to a process. By using seccomp, Podman enhances the security of containers by minimizing the attack surface and reducing the risk of malicious activities.

DEMO TIME


DRIVING
ASIL: tires
ASIL: safety
ASIL: breaks
ASIL: cruise_control
FLORIDA
RHIVOS
QM: radio
QM: maps
QM: store
QM: stream_audio
root@ibm-p8-kvm-03-guest-02:~
root@ibm-p8-kvm-03-guest-02:~ 89x11
[root@ibm-p8-kvm-03-guest-02 ~]#
[root@ibm-p8-kvm-03-guest-02 ~]#
root@ibm-p8-kvm-03-guest-02:~ 113x13
[root@ibm-p8-kvm-03-guest-02 ~]# watch -d "podman ps
> "
[root@ibm-p8-kvm-03-guest-02 ~]# watch -d "podman exec -it qm podman ps"
[root@ibm-p8-kvm-03-guest-02 ~]#
fedora

Red Hat

Thank you all

To the Audience
To DevConf.US Organizers & Volunteers
To Open Source contributors
To Red Hat and Automotive team

Red Hat

Q&A

- DevConf2023 Containers in a car by Dan Walsh
- https://sig.centos.org/automotive/getting_started/
- github.com/containers/podman Podman project
- github.com/containers/qm QM project
- github.com/eclipse-bluechi/bluechi Bluechi project
- github.com/containers/engine-stressor Engine Stressor
- https://github.com/containers/Demos/qm/devconf/README.md

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat