SBA Research

Cars, CAN, Containers and Clouds

How eBPF could save the road

Bundesministerium
Klimaschutz, Umwelt,
Energie, Mobilität,
Innovation und Technologie

Bundesministerium
Arbeit und Wirtschaft

FFG
Forschung wirkt.

wirtschafts
agentur
wien

Für die
Stadt Wien

European
Commission

FWF Österreichischer
Wissenschaftsfonds

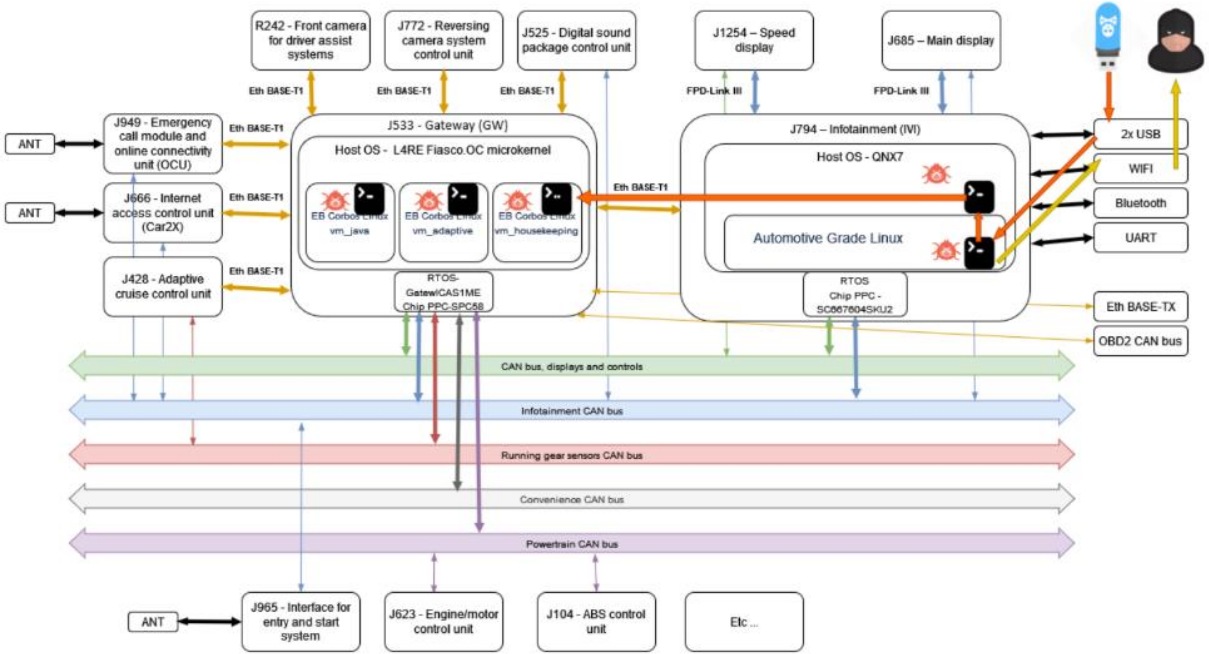
netidee
FÖRDERUNGEN

Infotainment module ICAS3 (IVI)

External view of the IVI (ICAS3) module

Back-connect to the connected car. Search for vulnerabilities in the VW electric car, Navinfo, Blackhat EU 2022
https://i.blackhat.com/EU-22/Wednesday-Briefings/EU-22-Serdyuk-Back-connect-to-the-connected-car.pdf

Attack vector via USB interfaces



Back-connect to the connected car. Search for vulnerabilities in the VW electric car, Navinfo, Blackhat EU 2022
https://i.blackhat.com/EU-22/Wednesday-Briefings/EU-22-Serdyuk-Back-connect-to-the-connected-car.pdf

- ht
hi

Image: https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

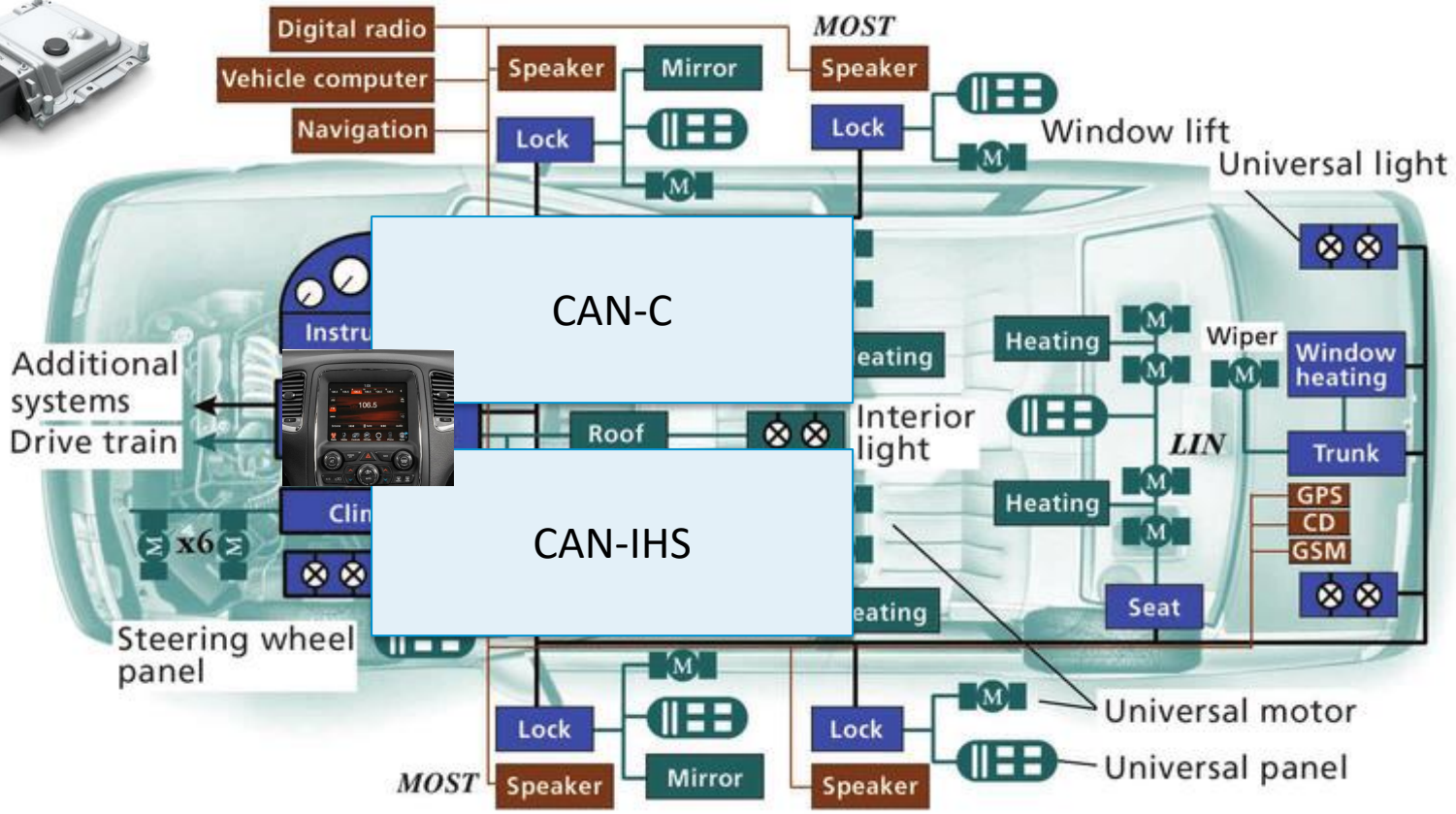
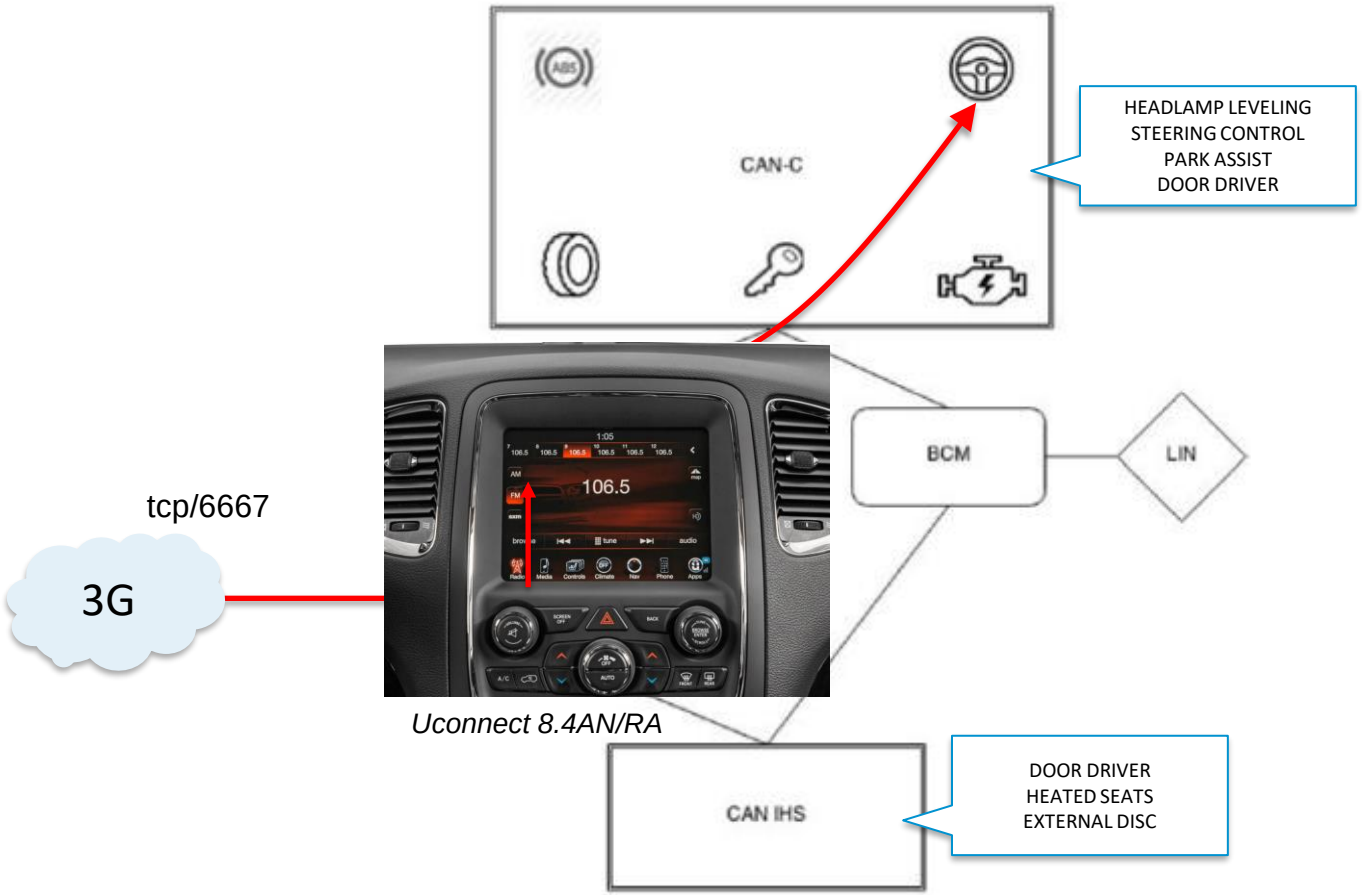


Image: https://www.researchgate.net/figure/One-subset-of-a-modern-vehicles-network-architecture-showing-the-trend-toward_fig1_2955571



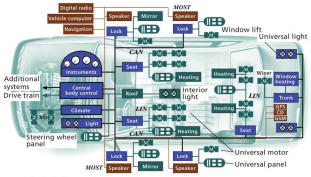


Image: Remote Car Hacking, Miller, Valasek (2015), p.8

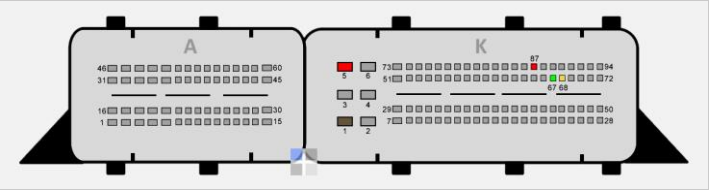
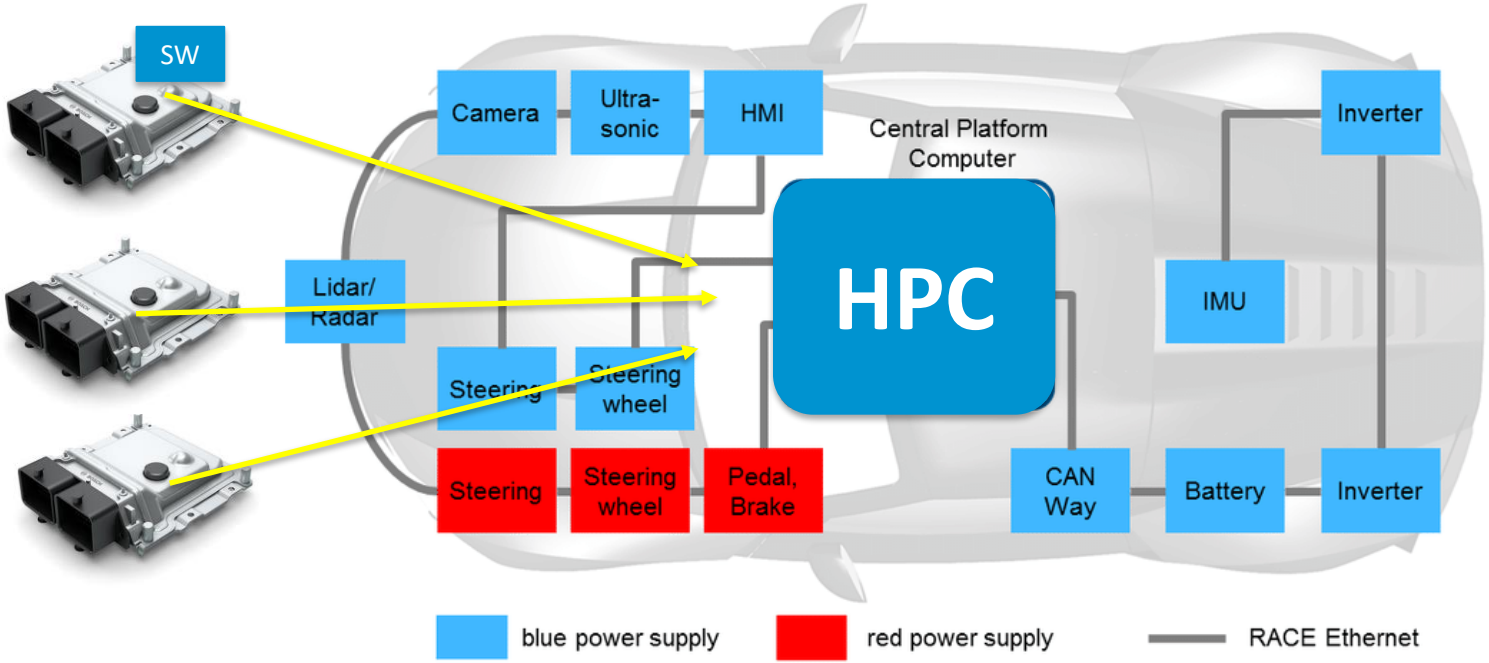


Image: https://auto24parts.com/

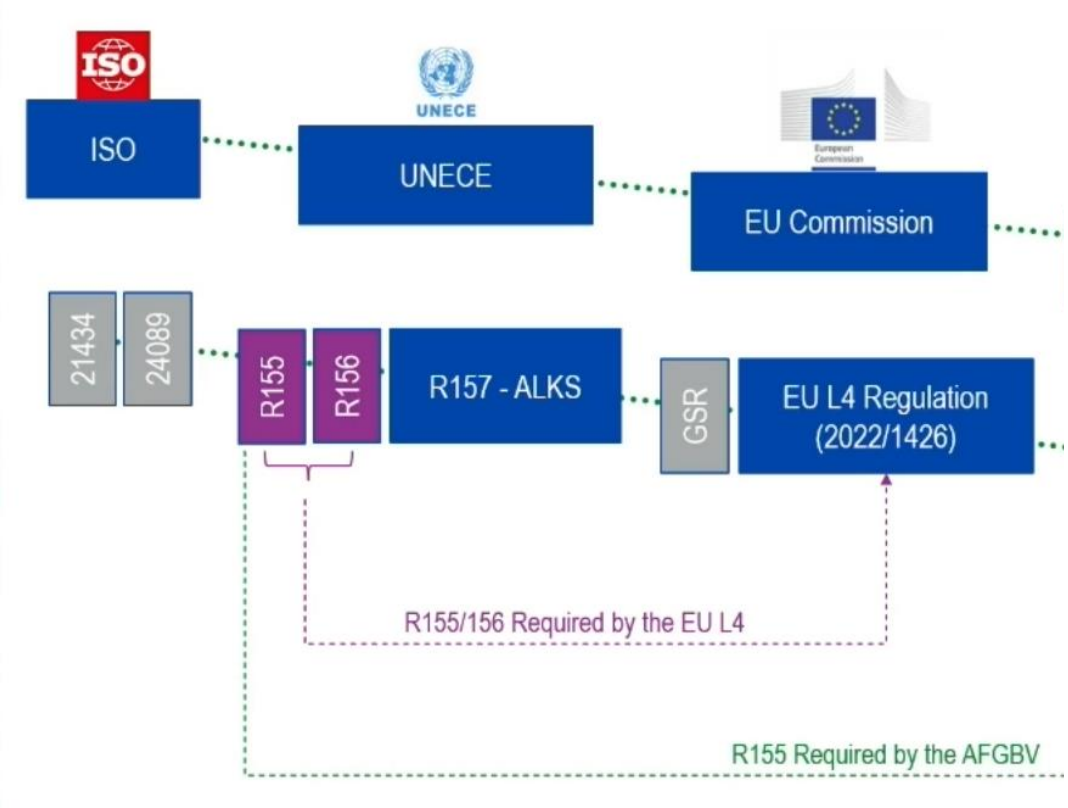
https://www.youtube.com/watch?v=3Wd4H91qfso

The Advent of the Software Defined Vehicle



ISO 24089:2023

Road vehicles — Software update engineering



Abstract

Preview

This document specifies requirements and recommendations for software update engineering for road vehicles on both the organizational and the project level.

This document is applicable to road vehicles whose software can be updated.

The requirements and recommendations in this document apply to vehicles, vehicle systems, ECUs, infrastructure, and the assembly and deployment of software update packages after the initial development.

This document is applicable to organizations involved in software update engineering for road vehicles. Such organizations can include vehicle manufacturers, suppliers, and their subsidiaries or partners.

This document establishes a common understanding for communicating and managing activities and responsibilities among organizations and related parties.

The development of software for vehicle functions, except for software update engineering, is outside the scope of this document.

Finally, this document does not prescribe specific technologies or solutions for software update engineering.

General information

Status : Published	**Publication date** : 2023-02
Edition : 1	**Number of pages** : 24

Red Hat Products Solutions Training & services Resources Partners About

Running containers in cars

October 19, 2022 | Pierre-Yves Chibon, Daniel J Walsh, Alexander Larsson

< Back to all posts Tags: Container

A little over a year ago, Red Hat announced its intention to collaborate with the automotive industry to help drive the transition to software-defined vehicles (SDVs) In May 2022, this intention became concrete with Red Hat and General Motors announcing their collaboration to help trailblaze SDVs at the edge. Our goal is to produce a base operating system to run all sorts of in-vehicle software for safety-critical use cases as well as non-safety ones.

Containers have become the de facto standard in the wider IT industry, and they are front and center in the vision of the **software-defined vehicle**, allowing for applications to be isolated, providing more flexibility for developers and deployment, and generally allowing for faster innovation. Red Hat leads the work on containers in the cloud, and now it's time to take that expertise to the automotive industry.

Continental Focus Topics Industry Solutions Components Partnerships Services

Smart Cockpit HPC

Continental's Smart Cockpit High-Performance Computer (HPC) targets the sweet spot between user experience, costs, and system performance, while enabling a short development time within only 18 months. It combines the powerful processors of the Telechips Dolphin family with pre-integrated functions of cluster, infotainment and ADAS visualization. In collaboration with Google Cloud, Continental integrates generative AI technology directly into the vehicle server. This allows drivers to interact with their car in a natural dialogue.

Learn more

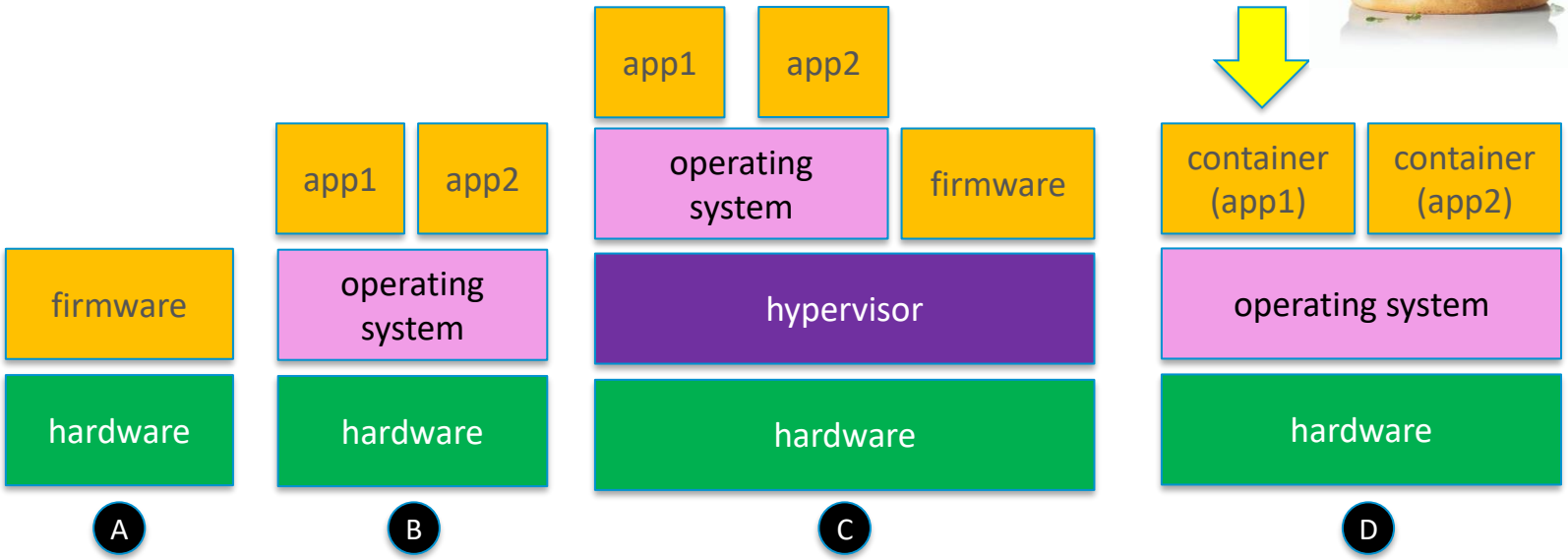
Vehicle Control HPC

The Vehicle Control High-Performance Computer (VC HPC) is the central high performance computing platform for the integration of applications from multiple domains and is therefore a key enabler for the Software-Defined Vehicle (SDV).

https://www.redhat.com/en/blog/running-containers-cars
https://www.continental-automotive.com/en/components/hpc-product-portfolio.html

Containers to the rescue?

Container, VM, Hypervisor, Process, …



Container

- create a separate view on the **file system** (mount namespace and pivot_root)
- show only **processes** inside the container (pid namespace)
- create a separate **network** and routing rules for the container processes (network namespace) and
  - **add CAN interfaces inside the container**
- **limit resources** inside the container (control groups)
- load eBPF programs for device control

create a container using crun (runc) with setup Ethernet/IP and CAN network (via custom CNI script)

container × host × kali@kali: ~/ebpf/spyspi × kali@kali: ~/dbus ×

```
┌──(kali㉿kali)-[~/dbus]
└─$ cat Dockerfile

FROM debian:bookworm-slim

RUN apt-get update -y && apt-get install -y python3-gi libgirepository1.0-dev dbus python3-dbus
RUN apt-get install -y can-utils
ADD service.py /app/service.py
COPY dbus-system.conf /etc/dbus-1/system.conf
RUN apt-get install -y procps iproute2 libcap2 iputils-ping curl
COPY start.sh /usr/local/bin
RUN chmod +x /usr/local/bin/start.sh
#CMD python3 /app/service.py


ENTRYPOINT ["/bin/bash", "/usr/local/bin/start.sh"]

┌──(kali㉿kali)-[~/dbus]
└─$
```

Demo: create a container with runc

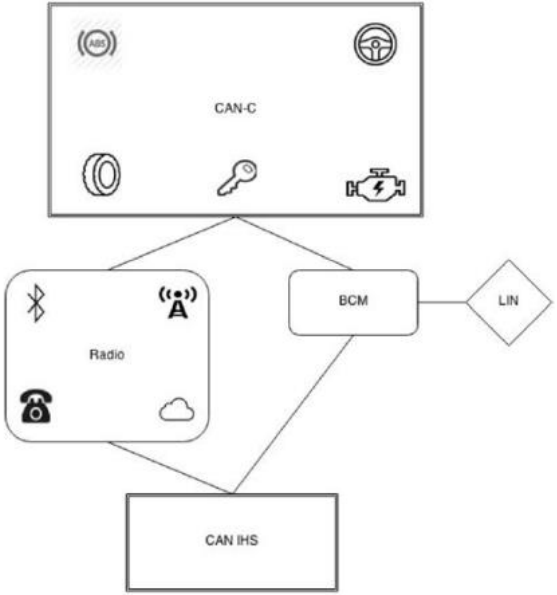
container × host × kali@kali: ~/ebpf/spyspi × kali@kali: ~/dbus ×

```
┌──(kali㉿kali)-[~/dbus]
└─$
```

runc run

Virtual CAN networks

routing and filtering



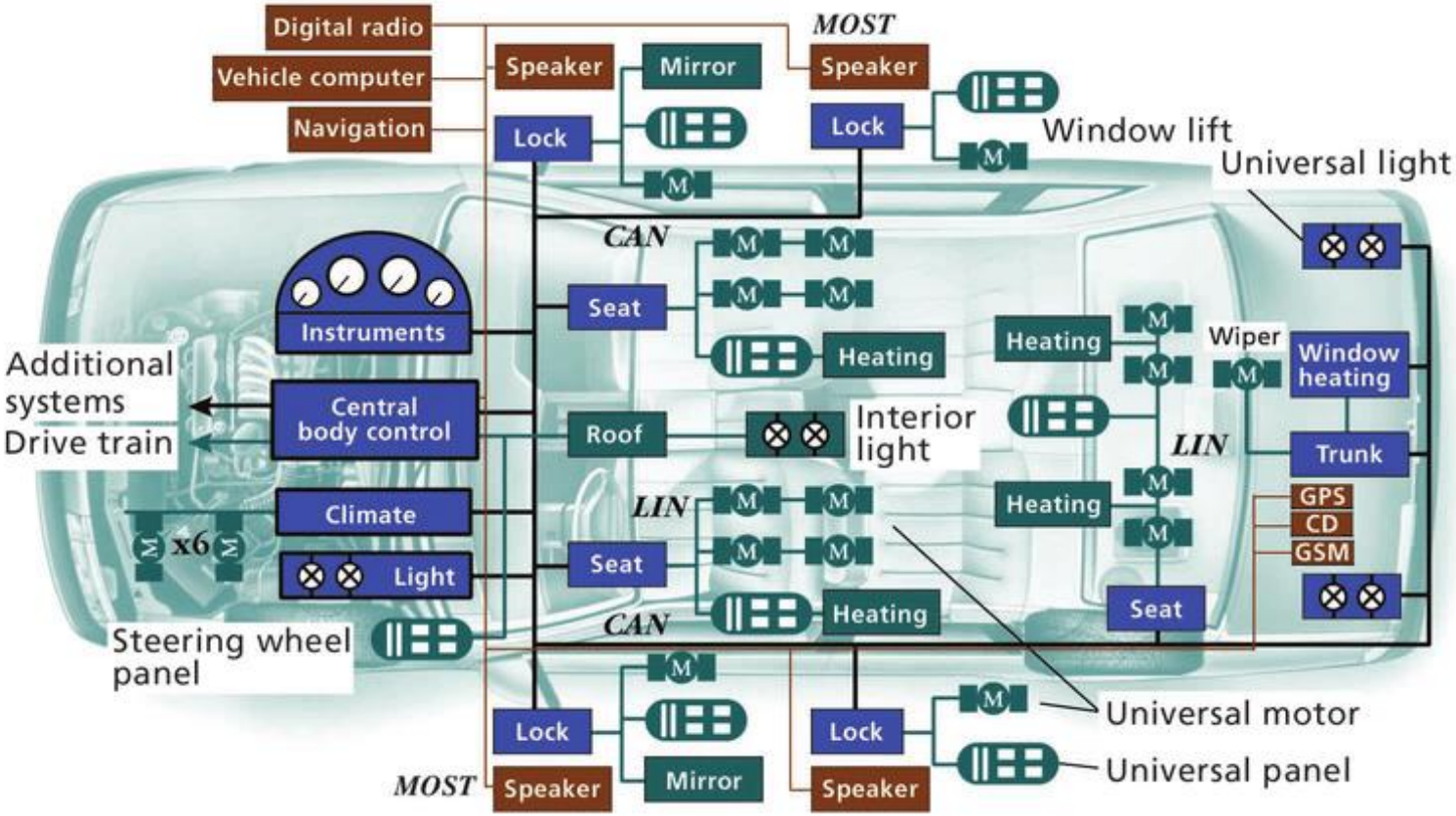
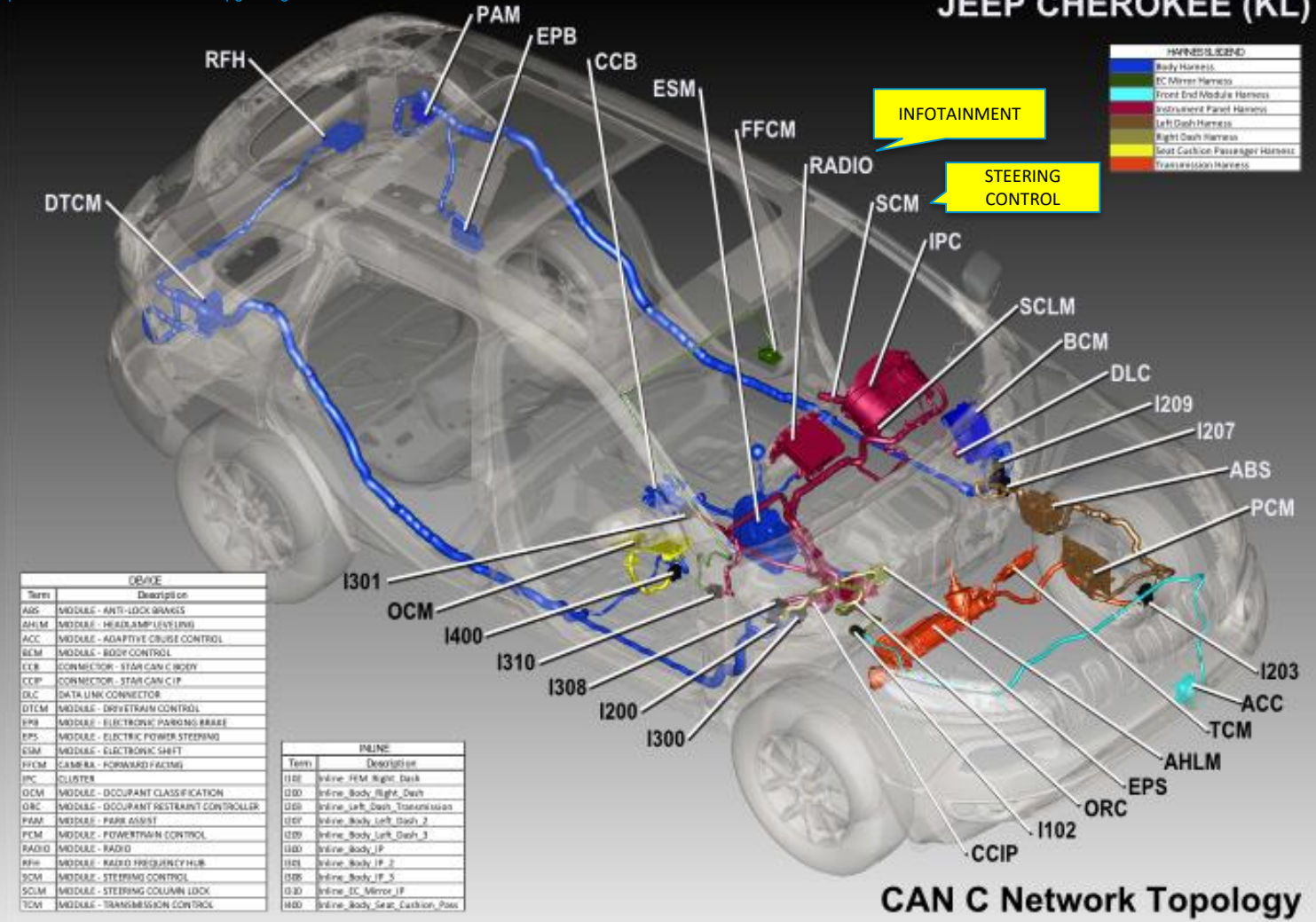


Image: https://www.researchgate.net/figure/One-subset-of-a-modern-vehicles-network-architecture-showing-the-trend-toward_fig1_2955571

Image: https://www.jeepcherokeeclub.com/threads/upgrading-radio.19169/

JEEP CHEROKEE (KL)

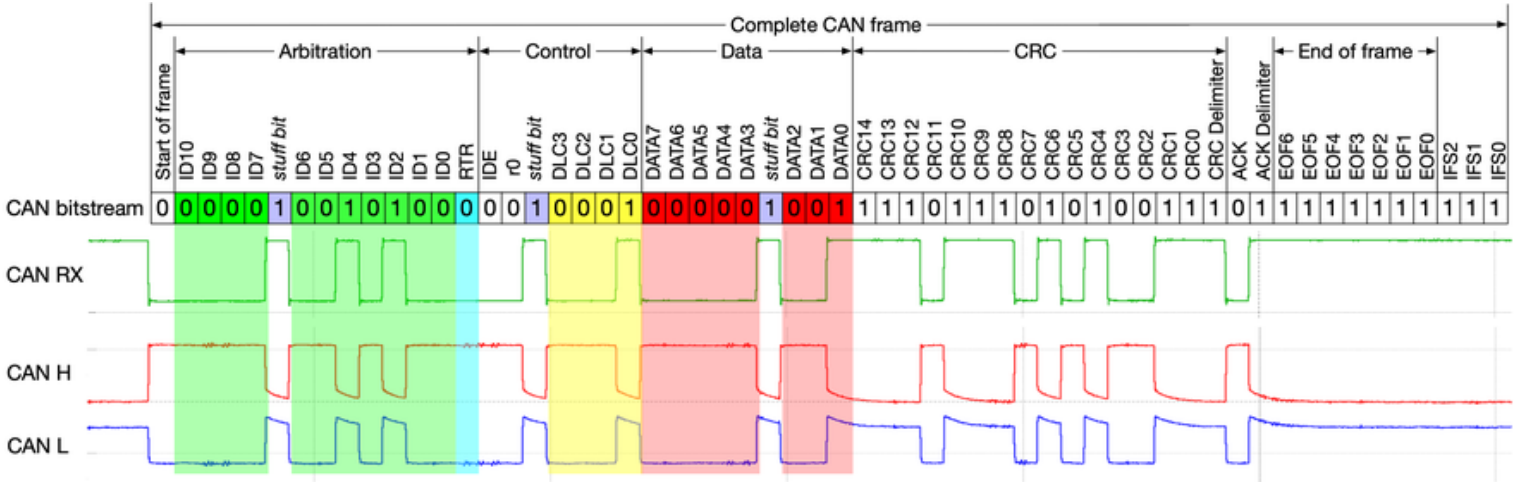


OB/OE	
Term	Description
ABS	MODULE - ANTI-LOCK BRAKES
AHLM	MODULE - HEADLAMP LEVELING
ACC	MODULE - ADAPTIVE CRUISE CONTROL
BCM	MODULE - BODY CONTROL
CCB	CONNECTOR - STAR CAN C BODY
CCIP	CONNECTOR - STAR CAN C IP
DLC	DATA LINK CONNECTOR
DTCM	MODULE - DRIVETRAIN CONTROL
EPB	MODULE - ELECTRONIC PARKING BRAKE
EPS	MODULE - ELECTRIC POWER STEERING
ESM	MODULE - ELECTRONIC SHIFT
FFCM	CAMERA - FORWARD FACING
IPC	CLUSTER
OCM	MODULE - OCCUPANT CLASSIFICATION
ORC	MODULE - OCCUPANT RESTRAINT CONTROLLER
PAM	MODULE - PARK ASSIST
PCM	MODULE - POWERTRAIN CONTROL
RADIO	MODULE - RADIO
RFH	MODULE - RADIO FREQUENCY HUB
SCM	MODULE - STEERING CONTROL
SCLM	MODULE - STEERING COLUMN LOCK
TCM	MODULE - TRANSMISSION CONTROL

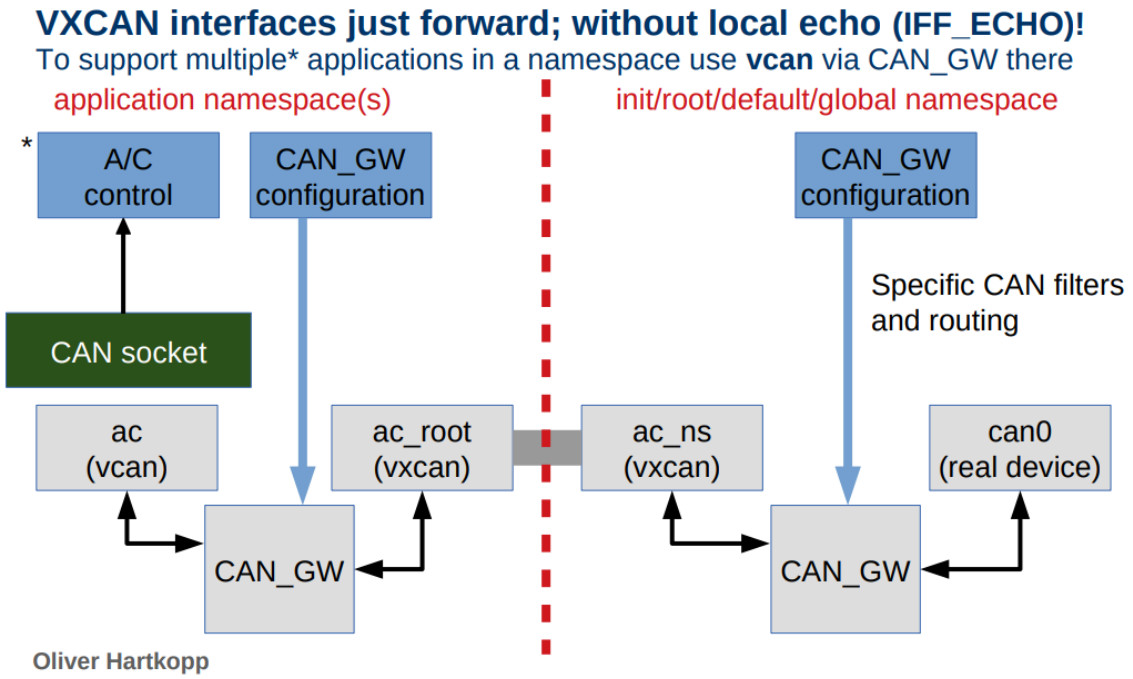
INLINE	
Term	Description
I101	Inline_FEM_Right_Dash
I200	Inline_Body_Right_Dash
I203	Inline_Left_Dash_Transmission
I207	Inline_Body_Left_Dash_2
I209	Inline_Body_Left_Dash_3
I300	Inline_Body_IP
I301	Inline_Body_IP_2
I308	Inline_Body_IP_5
I310	Inline_EC_Mirror_IP
I400	Inline_Body_Seat_Cushion_Pass

CAN C Network Topology

CAN Frame


Complete CAN frame
Arbitration
Control
Data
CRC
End of frame
Start of frame
ID10 ID9 ID8 ID7 stuff bit ID6 ID5 ID4 ID3 ID2 ID1 ID0 RTR
IDE r0 stuff bit DLC3 DLC2 DLC1 DLC0
DATA7 DATA6 DATA5 DATA4 DATA3 stuff bit DATA2 DATA1 DATA0
CRC14 CRC13 CRC12 CRC11 CRC10 CRC9 CRC8 CRC7 CRC6 CRC5 CRC4 CRC3 CRC2 CRC1 CRC0 CRC Delimiter
ACK ACK Delimiter
EOF6 EOF5 EOF4 EOF3 EOF2 EOF1 EOF0
IFS2 IFS1 IFS0
CAN bitstream
0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 1 1 1 1 0 1 1 1 0 1 0 1 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1
CAN RX
CAN H
CAN L

VCAN, VXCAN Kernel drivers and Automotive Grade Linux



https://wiki.automotivelinux.org/_media/agl-distro/agl2018-socketcan.pdf

CNI

CNI - the Container Network Interface

What is CNI?

CNI (*Container Network Interface*), a Cloud Native Computing Foundation project, consists of a specification and libraries for writing plugins to configure network interfaces in Linux containers, along with a number of supported plugins. CNI concerns itself only with network connectivity of containers and removing allocated resources when the container is deleted. Because of this focus, CNI has a wide range of support and the specification is simple to implement.

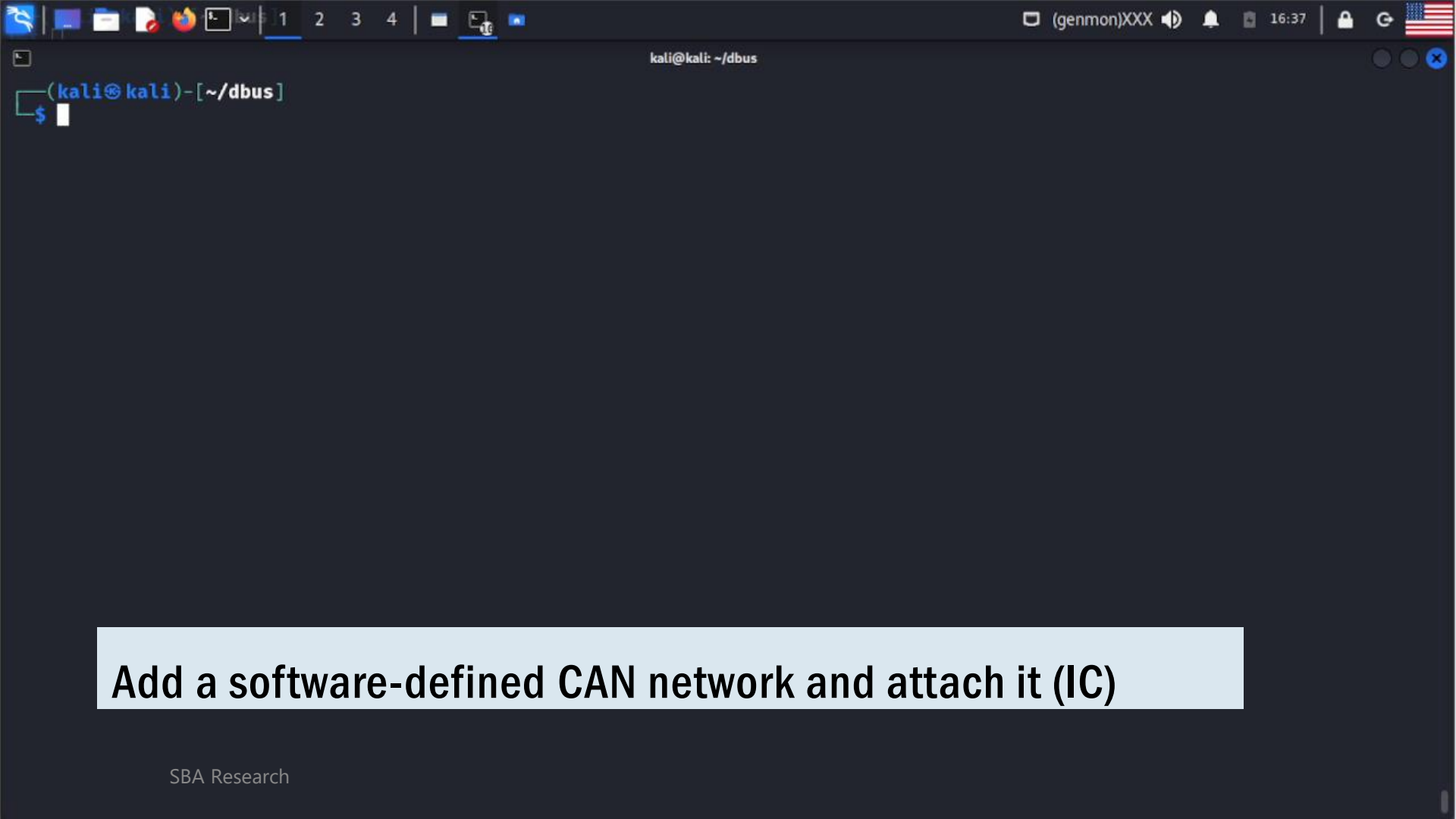
As well as the specification, this repository contains the Go source code of a library for integrating CNI into applications and an example command-line tool for executing CNI plugins. A separate repository contains reference plugins and a template for making new plugins.

The template code makes it straight-forward to create a CNI plugin for an existing container networking project. CNI also makes a good framework for creating a new container networking project from scratch.

Here are the recordings of two sessions that the CNI maintainers hosted at KubeCon/CloudNativeCon 2019:

- Introduction to CNI
- CNI deep dive

```
# check if process is running (debug)
ps $containerpid || exit 1

# set up a vcan bridge, all container
ip link show $canbridgename
if [ $? -ne 0 ] ; then
    ip link add name $canbridgename t
    ip link set $canbridgename up
fi

ip link add $vxcanname_container type
ip link set $vxcanname_host up
ip link set $vxcanname_container netn

# set up and rename to vxcan0 (contai
nsenter -n -t $containerpid ip link s
nsenter -n -t $containerpid ip link s

# set up a bridge (gw) between can br
cangw -A -s $canbridgename -d $vxcann
cangw -A -s $vxcanname_host -d $canbr

# flow in (physical to container)
#cangw -A -s "$physical_can" -d "$vxc
# flow out (container to physical)
#cangw -A -d "$physical_can" -s "$vxc
```



Add a software-defined CAN network and attach it (IC)

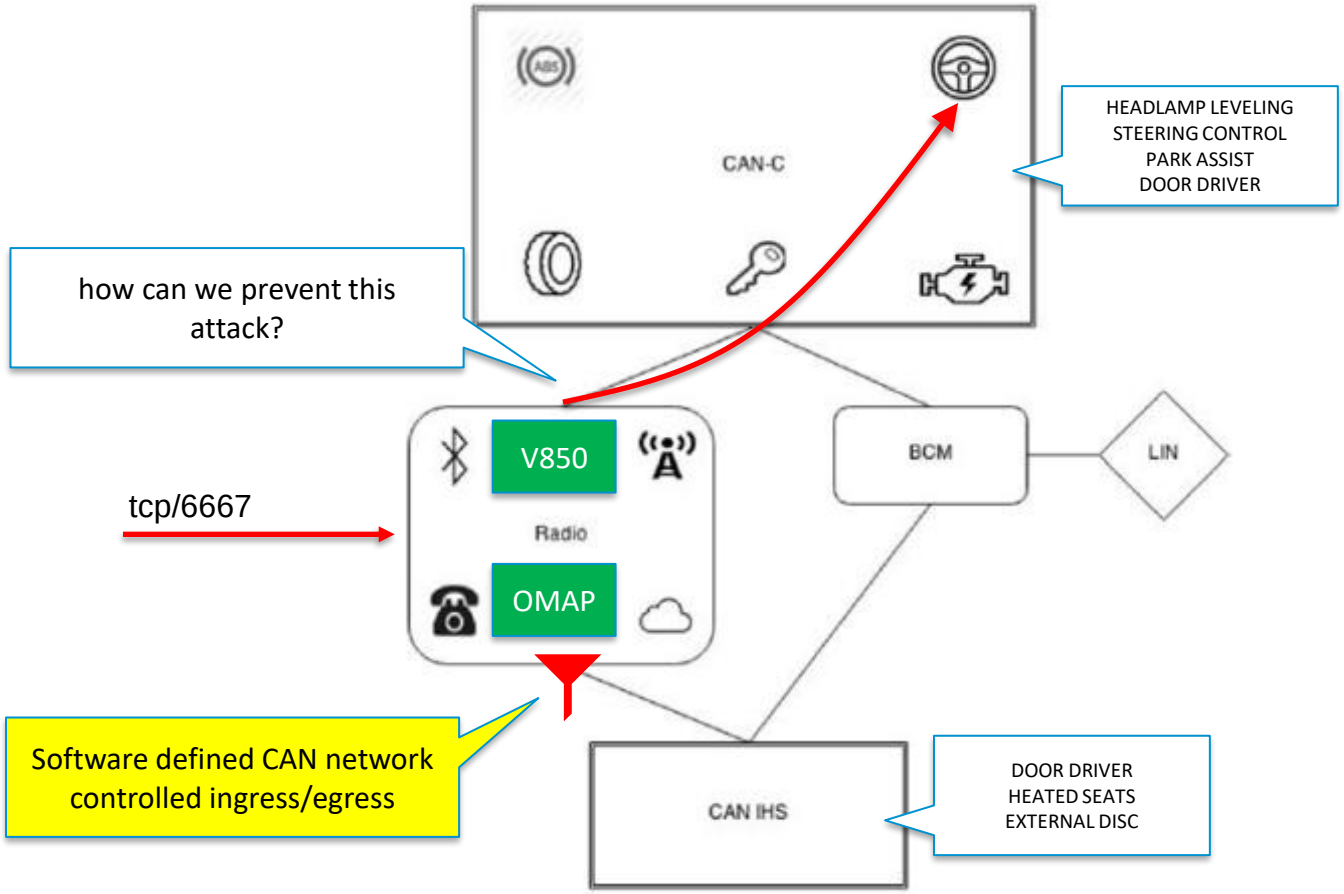
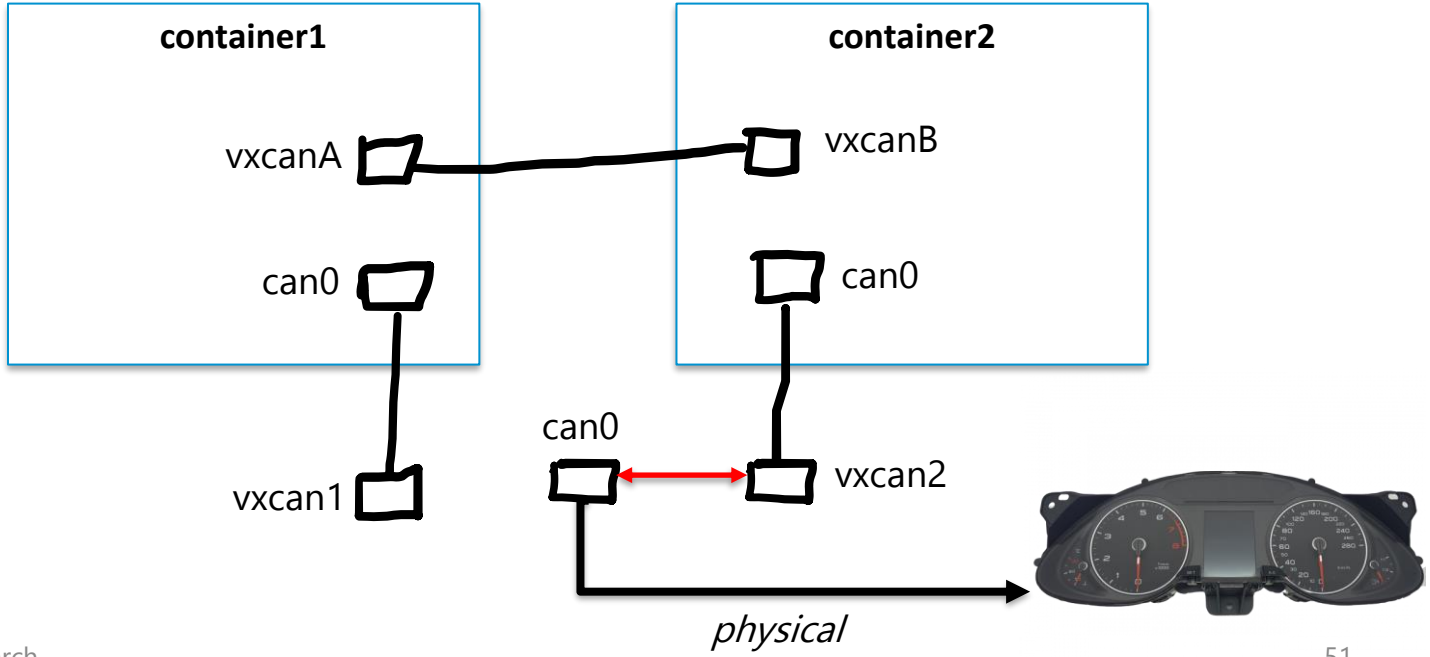


Image: Remote Car Hacking, Miller, Valasek (2015), p.8

CANGW and VXCAN

```
# cangw -A -s vcan2 -d can0 -e
# cangw -A -s can0 -d vcan2 -e
```

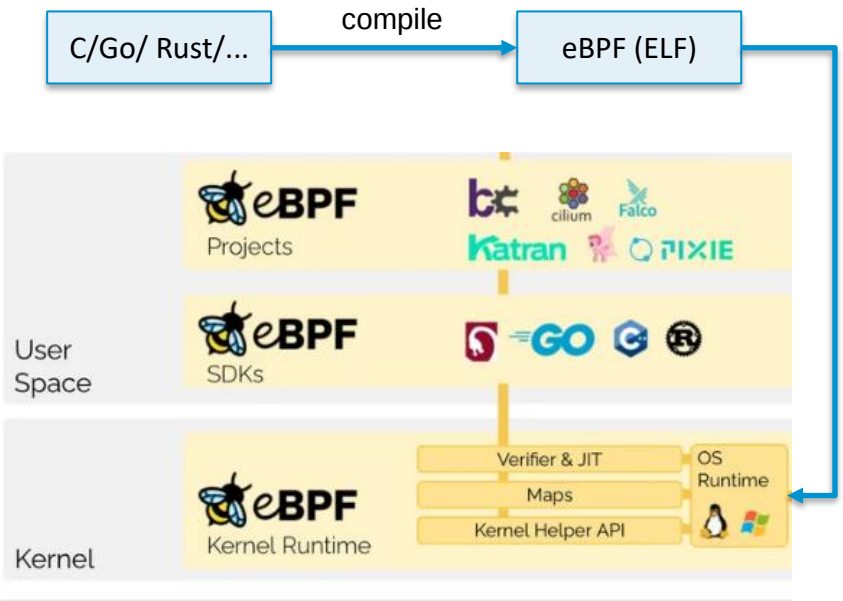


eBPF recap on one slide

eBPF code runs in a virutal machine in the Kernel.

The code contains functions that can be attached to a trigger (e.g. syscall, trace event, network).

1. **load** BPF code into the Kernel
2. **attach** function
3. Kernel runs the program if a **trigger** is hit
4. The eBPF program can produce **data** and store it in maps/queues
5. **consume** data from userspace



Trace points and Probes

BPF_PROG_TYPE_SK_SKB		sk_skb
	BPF_SK_SKB_STREAM_PARSER	sk_skb/stream_parser
	BPF_SK_SKB_STREAM_VERDICT	sk_skb/stream_verdict
BPF_PROG_TYPE_SOCKET_FILTER		socket
BPF_PROG_TYPE_SOCK_OPS	BPF_CGROUP_SOCK_OPS	sockops
BPF_PROG_TYPE_STRUCT_OPS		struct_ops+
BPF_PROG_TYPE_SYSCALL		syscall
BPF_PROG_TYPE_TRACEPOINT		tp+ [9]
		tracepoint+ [9]
BPF_PROG_TYPE_TRACING	BPF_MODIFY_RETURN	fmod_ret+ [1]
		fmod_ret.s+ [1]
	BPF_TRACE_FENTRY	fentry+ [1]
		fentry.s+ [1]
	BPF_TRACE_FEXIT	fexit+ [1]
		fexit.s+ [1]
	BPF_TRACE_ITER	iter+ [10]
		iter.s+ [10]
	BPF_TRACE_RAW_TP	tp_btf+ [1]
BPF_PROG_TYPE_XDP	BPF_XDP_CPUMAP	xdp.frags/cpumap
		xdp/cpumap
	BPF_XDP_DEVMAP	xdp.frags/devmap
		xdp/devmap
	BPF_XDP	xdp.frags
		xdp

- kprobes (scoped to a cgroup)
- cgroup/dev
- perf events
- **syscall**
- **socket buffer (SK SKB)**
- **XDP**
- tracepoints
- raw_tracepoints
- (...)

https://docs.kernel.org/bpf/libbpf/program_types.html

Linux Event Tracing

```
└─$ sudo ls /sys/kernel/debug/tracing/events/
alarmtimer        enable        iocost        msr             regmap      thermal_power_allocator
amd_cpu           error_report  iomap         napi            regulator   thp
avc               exceptions    iommu         neigh           resctrl     timer
block             ext4          io_uring      net             rpm         tlb
bpf_test_run      fib           ipi           netlink         rseq        udp
bpf_trace         fib6          irq           nmi             rtc         vmalloc
bridge            filelock      irq_matrix    notifier        sched       vmscan
cgroup            filemap       irq_vectors   oom             scsi        vsock
clk               fs_dax        jbd2          page_isolation  sd          vsyscall
compaction        ftrace        kmem          pagemap         signal      wbt
context_tracking  gpio          ksm           page_pool       skb         workqueue
cpuhp             handshake     libata        percpu          smbus       writeback
cros_ec           header_event  lock          power           sock        x86_fpu
csd               header_page   maple_tree    printk          spi         xdp
cxl               huge_memory   mce           pwm             sunrpc      xen
dev               hwmon         migrate       qdisc           swiotlb     xhci-hcd
devfreq           hyperv        mmap          qrtr            syscalls
devlink           i2c           mmap_lock     ras             task
dma_fence         initcall      module        raw_syscalls    tcp
drm               intel_iommu   mptcp         rcu             thermal
```

https://docs.kernel.org/trace/events.html

/sys/kernel/debug/tracing/events/net/

- net_dev_xmit
- net_dev_start_xmit
- netif_receive_skb
- netif_rx

```
     kworker/1:3-3041    [001] b..1.  7923.796270: net_dev_xmit: dev=eth0 skbaddr=00000000d64f667f len=90 rc=0
  NetworkManager-782     [000] b....  7924.025676: net_dev_queue: dev=eth0 skbaddr=00000000ce04a983 len=62
  NetworkManager-782     [000] b..1.  7924.025766: net_dev_start_xmit: dev=eth0 queue_mapping=0 skbaddr=00000000ce04a983 vlan
_tagged=0 vlan_proto=0×0000 vlan_tci=0×0000 protocol=0×86dd ip_summed=0 len=62 data_len=0 network_offset=14 transport_offset_
valid=1 transport_offset=54 tx_flags=0 gso_size=0 gso_segs=0 gso_type=0×0
  NetworkManager-782     [000] b..1.  7924.025826: net_dev_xmit: dev=eth0 skbaddr=00000000ce04a983 len=62 rc=0
     kworker/1:3-3041    [001] b....  7924.403806: net_dev_queue: dev=eth0 skbaddr=000000009442b710 len=90
     kworker/1:3-3041    [001] b..1.  7924.403948: net_dev_start_xmit: dev=eth0 queue_mapping=0 skbaddr=000000009442b710 vlan
_tagged=0 vlan_proto=0×0000 vlan_tci=0×0000 protocol=0×86dd ip_summed=0 len=90 data_len=0 network_offset=14 transport_offset_
valid=1 transport_offset=62 tx_flags=0 gso_size=0 gso_segs=0 gso_type=0×0
     kworker/1:3-3041    [001] b..1.  7924.404019: net_dev_xmit: dev=eth0 skbaddr=000000009442b710 len=90 rc=0
  NetworkManager-782     [000] b....  7924.830590: net_dev_queue: dev=eth0 skbaddr=00000000e8f37504 len=42
  NetworkManager-782     [000] b..1.  7924.830717: net_dev_start_xmit: dev=eth0 queue_mapping=0 skbaddr=00000000e8f37504 vlan
_tagged=0 vlan_proto=0×0000 vlan_tci=0×0000 protocol=0×0806 ip_summed=0 len=42 data_len=0 network_offset=14 transport_offset_
valid=1 transport_offset=14 tx_flags=0 gso_size=0 gso_segs=0 gso_type=0×0
  NetworkManager-782     [000] b..1.  7924.830773: net_dev_xmit: dev=eth0 skbaddr=00000000e8f37504 len=42 rc=0
```

eXpress Data Path (XDP)

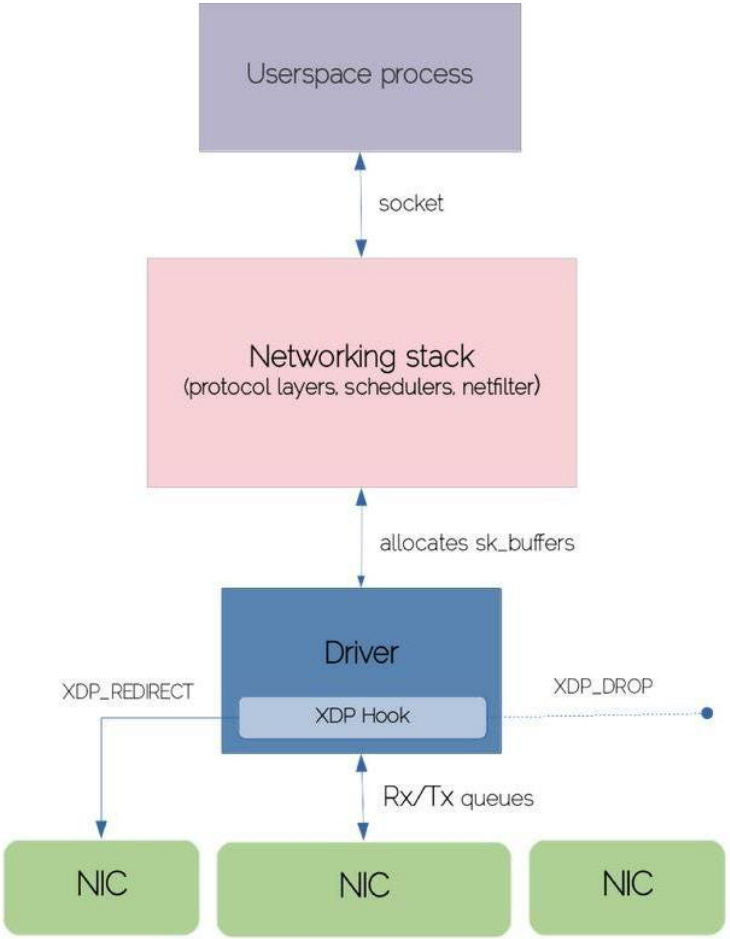


Image: https://medium.com/@kcl17/unlocking-network-performance-with-xdp-and-ebpf-67c712128025

home > kali > ebpf > ebpf-can > C xdp_pass.bpf.c

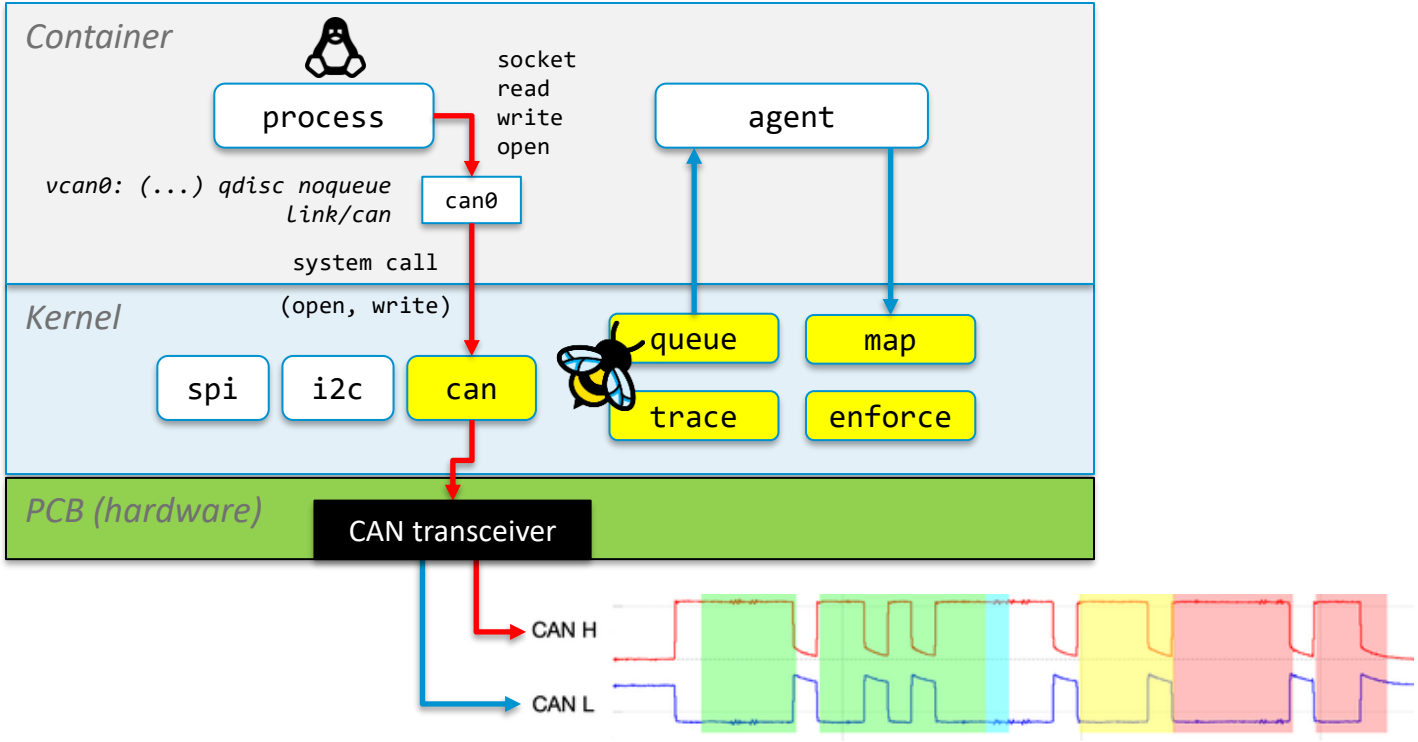
```c
SEC("xdp")
int  xdp_filter_can(struct xdp_md *ctx)
{
    void *data_end = (void *)(long)ctx->data_end;
    void *data = (void *)(long)ctx->data;
    struct can_frame *frame = data;

    if(data + sizeof(struct can_frame) > data_end) // fo
    {
        bpf_printk("drop");
        return XDP_DROP;
    }

    bpf_printk("can/xdp id=0x%x dlc=%d data=0x%02x.%02x.
        frame->can_id,
        frame->can_dlc,
        frame->data[0],
        frame->data[1],
        frame->data[2],
        frame->data[3],
        frame->data[4],
        frame->data[5],
        frame->data[6],
        frame->data[7]); // 8 data bytes

    if(frame->can_id == 0x111) // firewall 0x111 packets

    return XDP_PASS;
}

char LICENSE[] SEC("license") = "GPL";
```

kali@kali: ~/ebpf/ebpf-can

```
┌──(kali㉿kali)-[~/ebpf/ebpf-can]
└─$
```

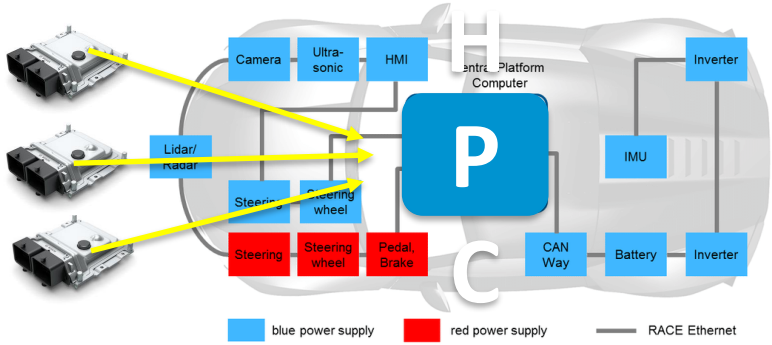
Demo: eBPF for CAN bus

Process of eBPF for CAN/network interfaces (Linux)

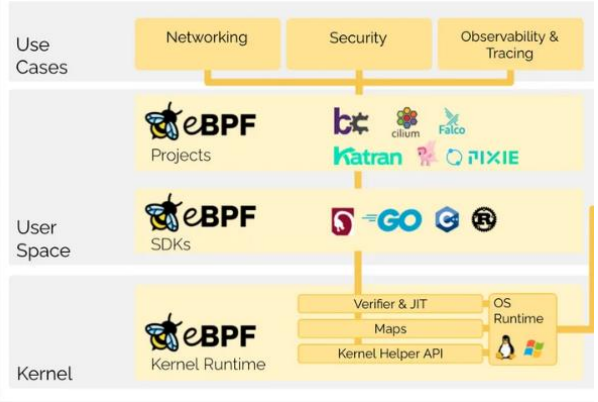


Tetragon

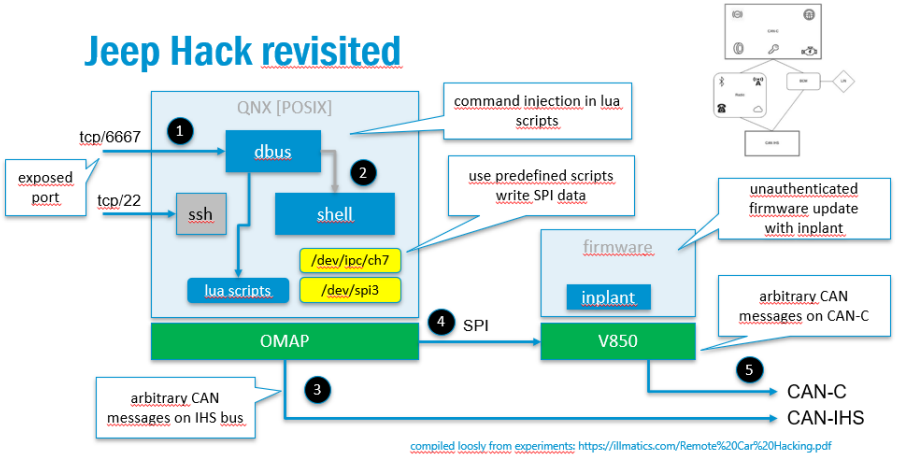
falco



Conclusion



Jeep Hack revisited



Takeaways (containers and eBPF)

- containers contribute to incapsulation
- security parameters are detailed but efficient
- eBPF provides observabilty and reaction capabilities
- the application is untouched (all runs in Kernel)

SBA Research // ASRG-VIE // eBPF-Vienna

- SBA Research Meetups
  - https://www.meetup.com/security-meetup-by-sba-research/
- Automotive Security Research Group Vienna (ASRG-VIE)
  - https://www.meetup.com/automotive-security-research-group-vienna-asrg-vie/
- eBPF Vienna
  - https://www.meetup.com/de-DE/ebpf-vienna/events/

Applied Research Consulting in Customer Projects

ASSESS

Threats and Security Architecture

- Security Requirements
- Crypto Designs
- Threat Modelling

EVALUATE

Identify Attacks and Defenses

- Static Code Analysis
- Security Test Case Generation
- Defense in Depth (DiD)

VALIDATE

Software and System Testing

- Fuzz Testing
- Black- and Grey-box Testing
- Proof of Concept Validation

Tailored Security Trainings

Threat Analysis	Coding and Exploitation	Container Security
Crypto Engineering	IoT Hacking	Combinatorial Security Testing

Embedded Systems · Containers · Servers and Data Centers · Industrial IoT · Operational Technologies · Cloud Systems

Reinhard Kugler

MATRIS Research Lab

SBA Research

Floragasse 7, 1040 Wien

rkugler@sba-research.org

https://matris.sba-research.org/marc/