SBA Research

Observing the Clouds

Container Defenses for Embedded Systems using eBPF

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

Observing the Clouds

Container Defenses for Embedded Systems using eBPF

Alexandr P, https://www.pexels.com/video/a-beautiful-view-of-sunset-1893623/

Oleksandr P, https://www.pexels.com/video/a-beautiful-view-of-sunset-1893623/

Cloud Technologies in Cyber Physical Systems

Automotive

ECU PLATFORM

Co-engineering next generation ECU firmware with automotive suppliers & Implementation of security testing schemes

CAN ETHERNET

Research prototype for automatic combinatorial testing of CAN-based ECUs

CLOUD

Security assessments of Kubernetes environments (AWS, Azure, and Google Cloud)

LINUX CONTAINER

Security trainings on container attacks & Research on embedded containers (HPC)

OVER THE AIR UPDATES

System test and assessment for secure updates on embedded platforms

SERVER

Best practices for configuration and deployment

Production Systems

FIELDBUS

Research of Modbus fuzzing security testing for civil infrastructure projects

HMI AND PLC

Secure Integration project for a HMI platform

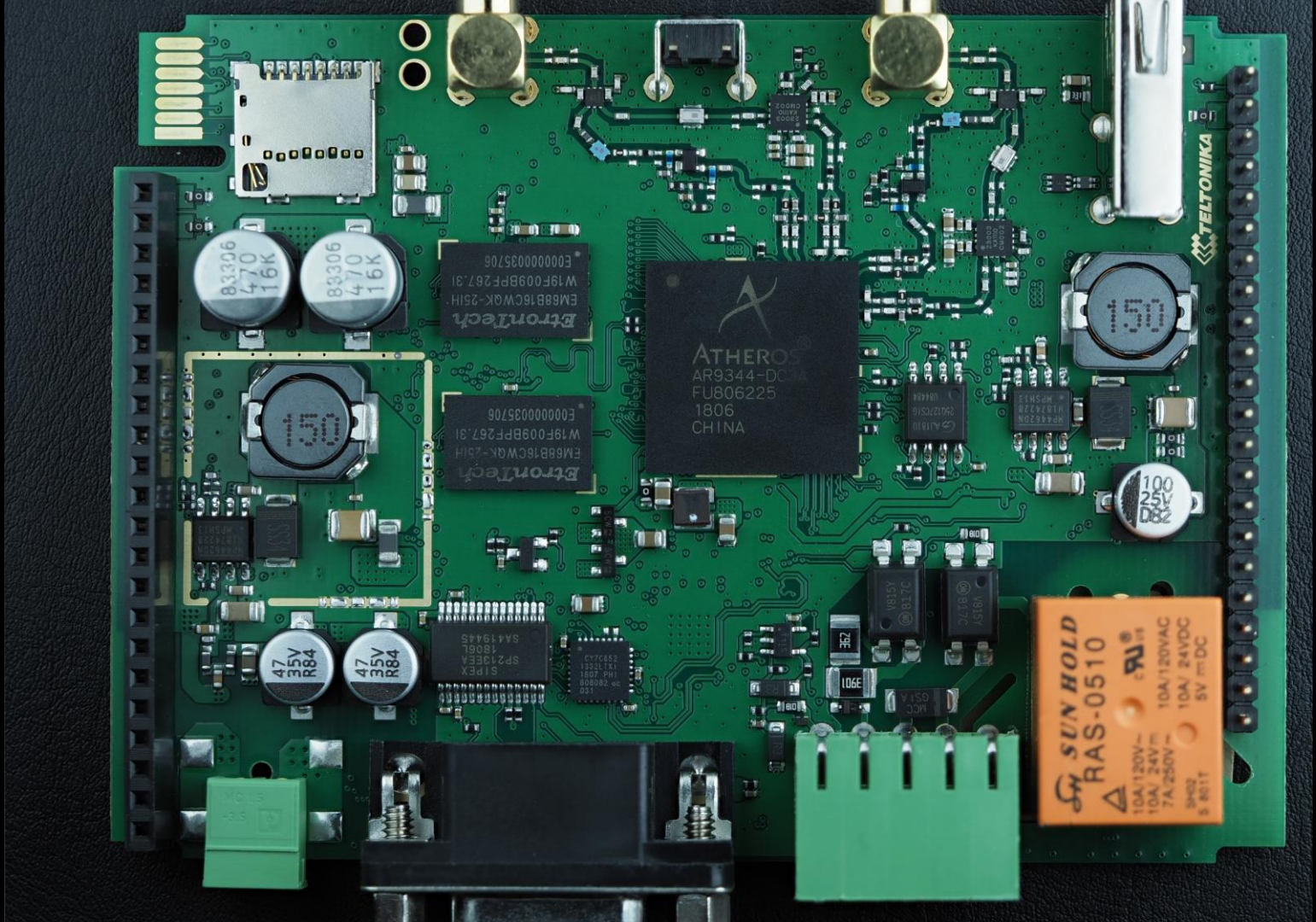
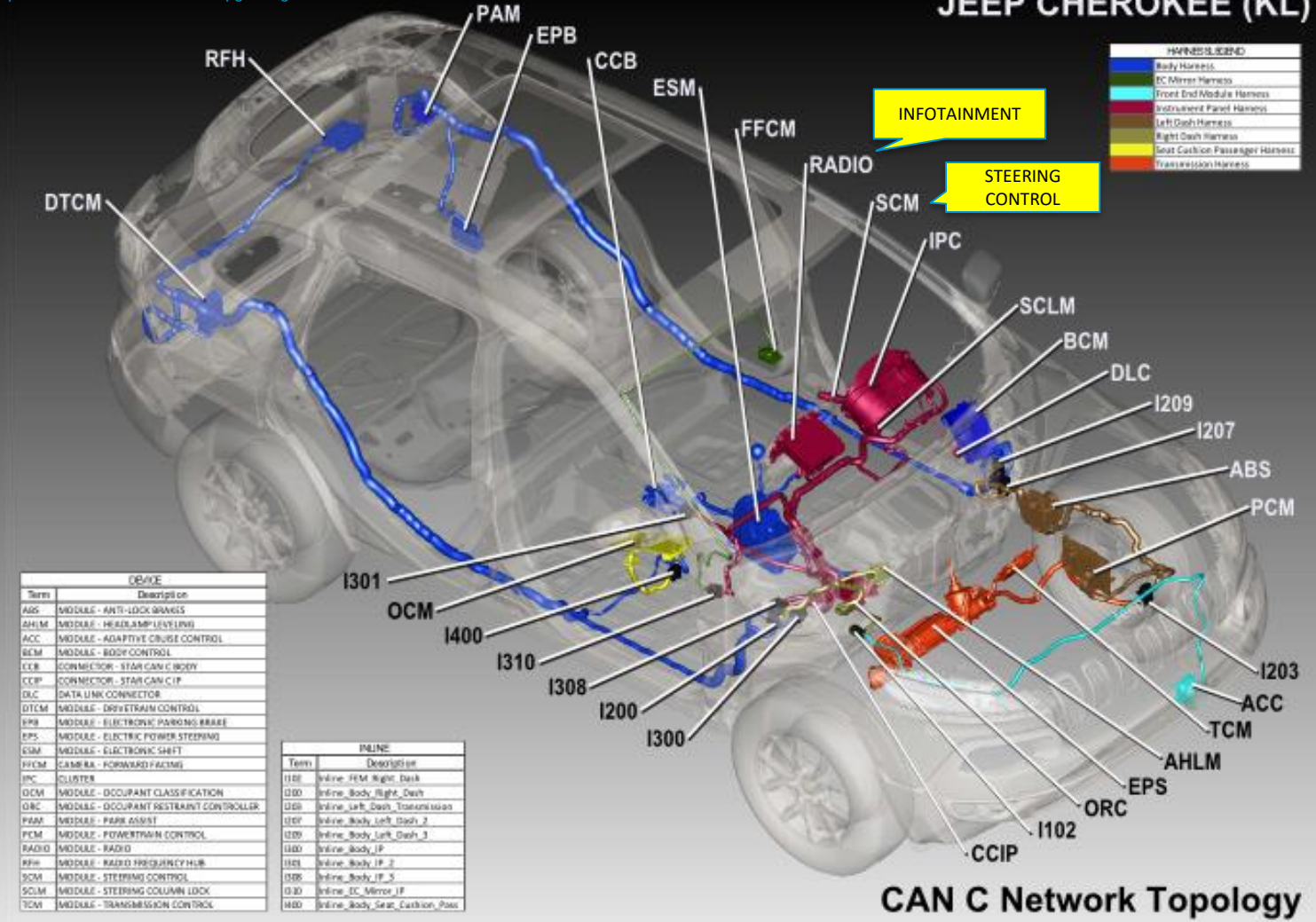

TELTONIKA
ATHEROS
AR9344-DC
FU806225
1806
CHINA
SUN HOLD
RAS-0510
10A/120V~
10A/24V=
7A/250V~
10A/120VAC
10A/24VDC
5V=DC
150
150
83306
470
16K
83306
470
16K
47
35V
R84
47
35V
R84
100
25V
D82

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

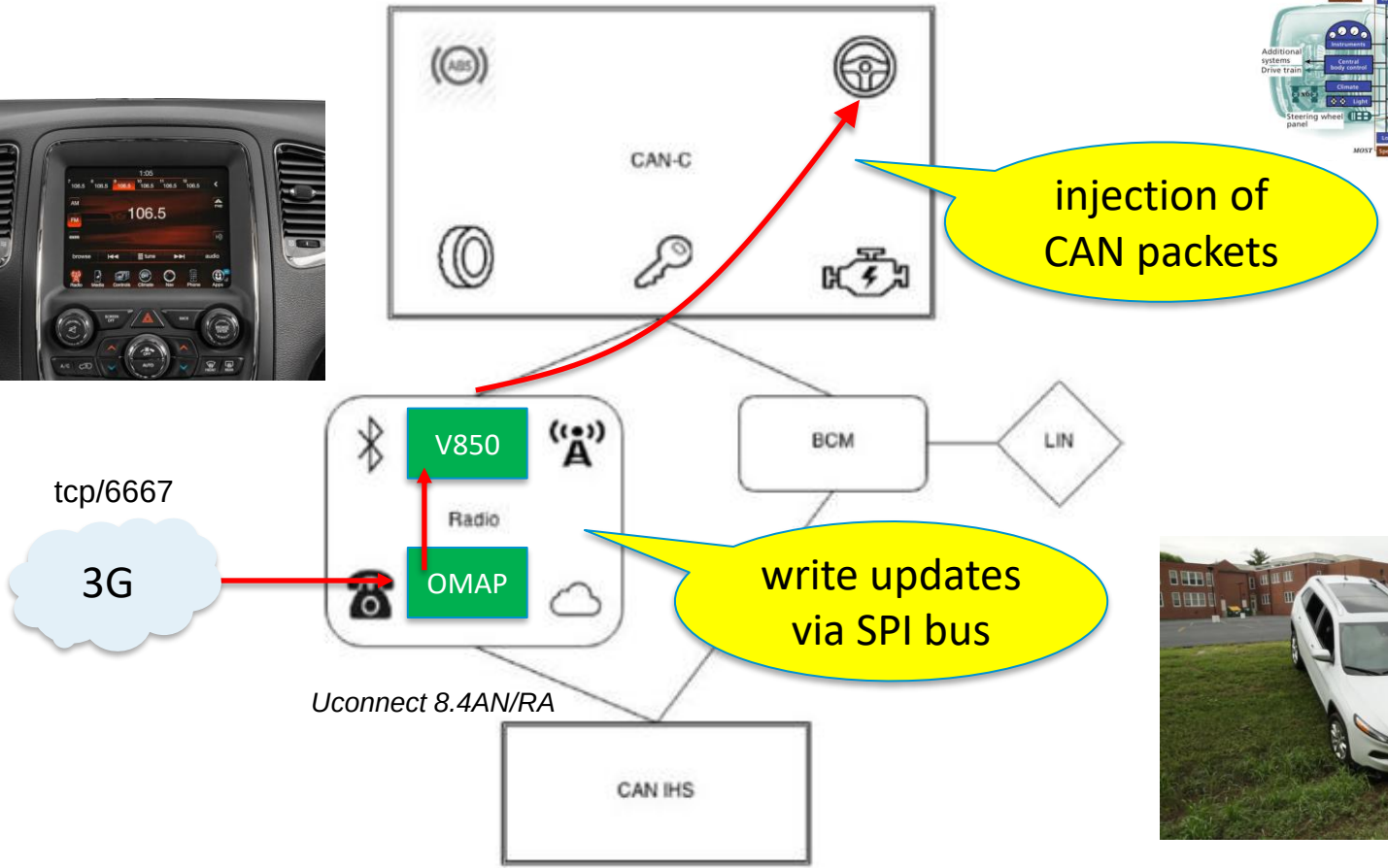
JEEP CHEROKEE (KL)



DEVICE	
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
I102	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



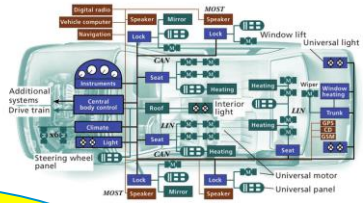
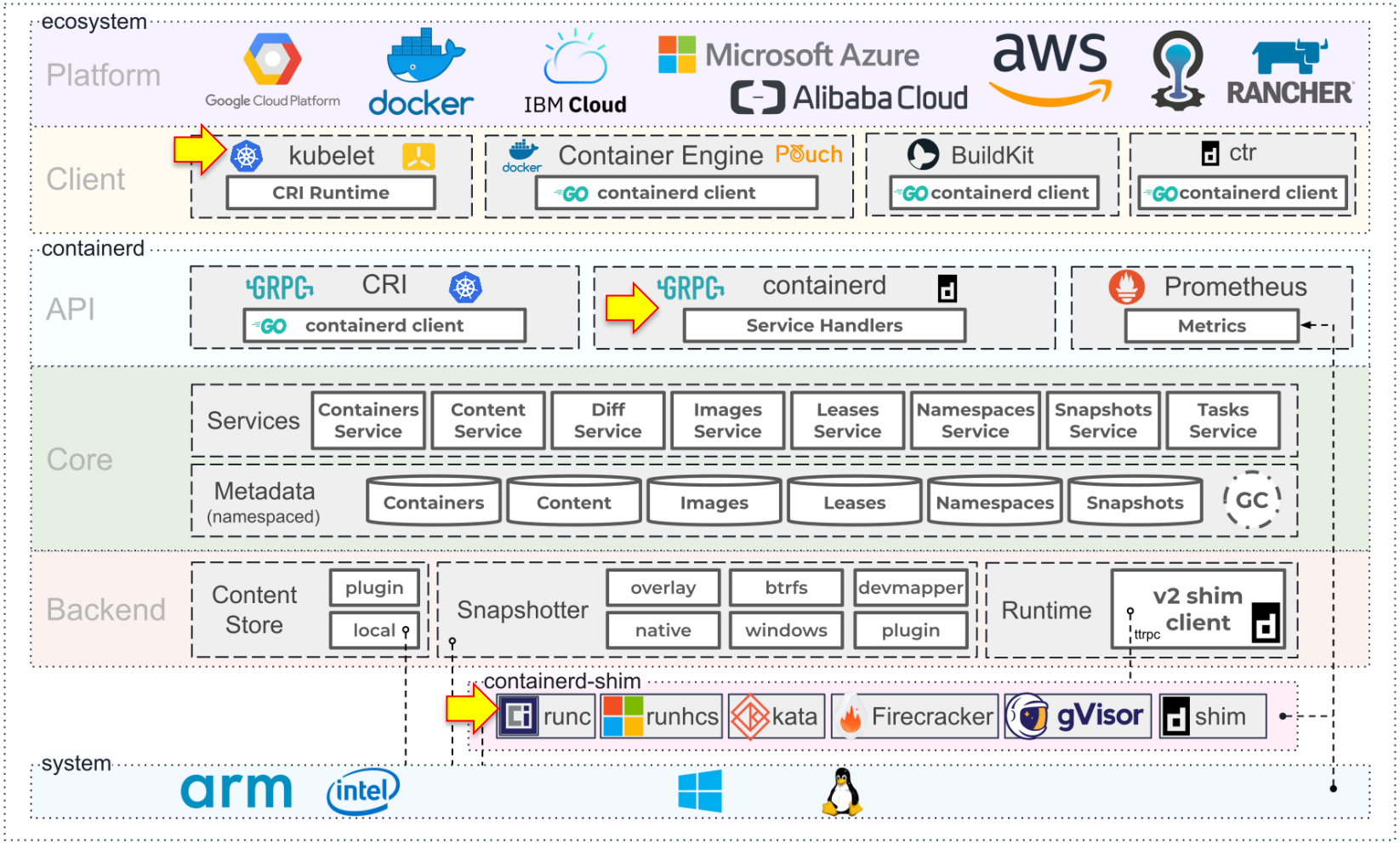


Image: Remote Car Hacking, Miller, Valasek (2015), p.8

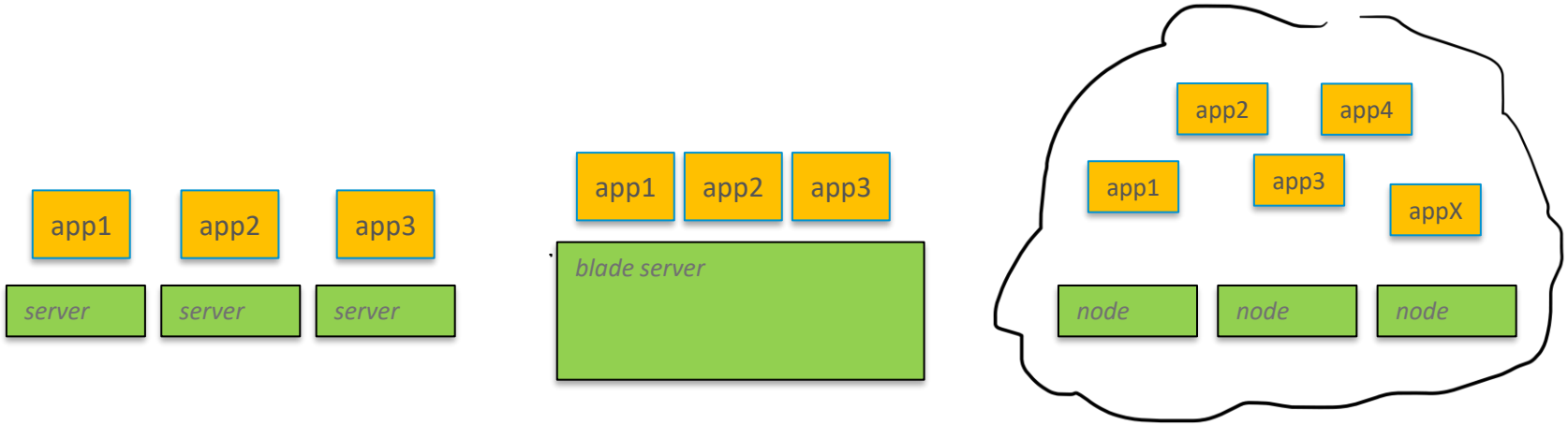

ecosystem
Platform
Google Cloud Platform
docker
IBM Cloud
Microsoft Azure
Alibaba Cloud
aws
RANCHER
Client
kubelet
CRI Runtime
Container Engine
containerd client
BuildKit
containerd client
ctr
containerd client
containerd
API
GRPC
CRI
containerd client
containerd
Service Handlers
Prometheus
Metrics
Core
Services
Containers Service
Content Service
Diff Service
Images Service
Leases Service
Namespaces Service
Snapshots Service
Tasks Service
Metadata
(namespaced)
Containers
Content
Images
Leases
Namespaces
Snapshots
GC
Backend
Content Store
plugin
local
Snapshotter
overlay
btrfs
devmapper
native
windows
plugin
Runtime
v2 shim client
ttrpc
containerd-shim
runc
runhcs
kata
Firecracker
gVisor
shim
system
arm
intel

Containers

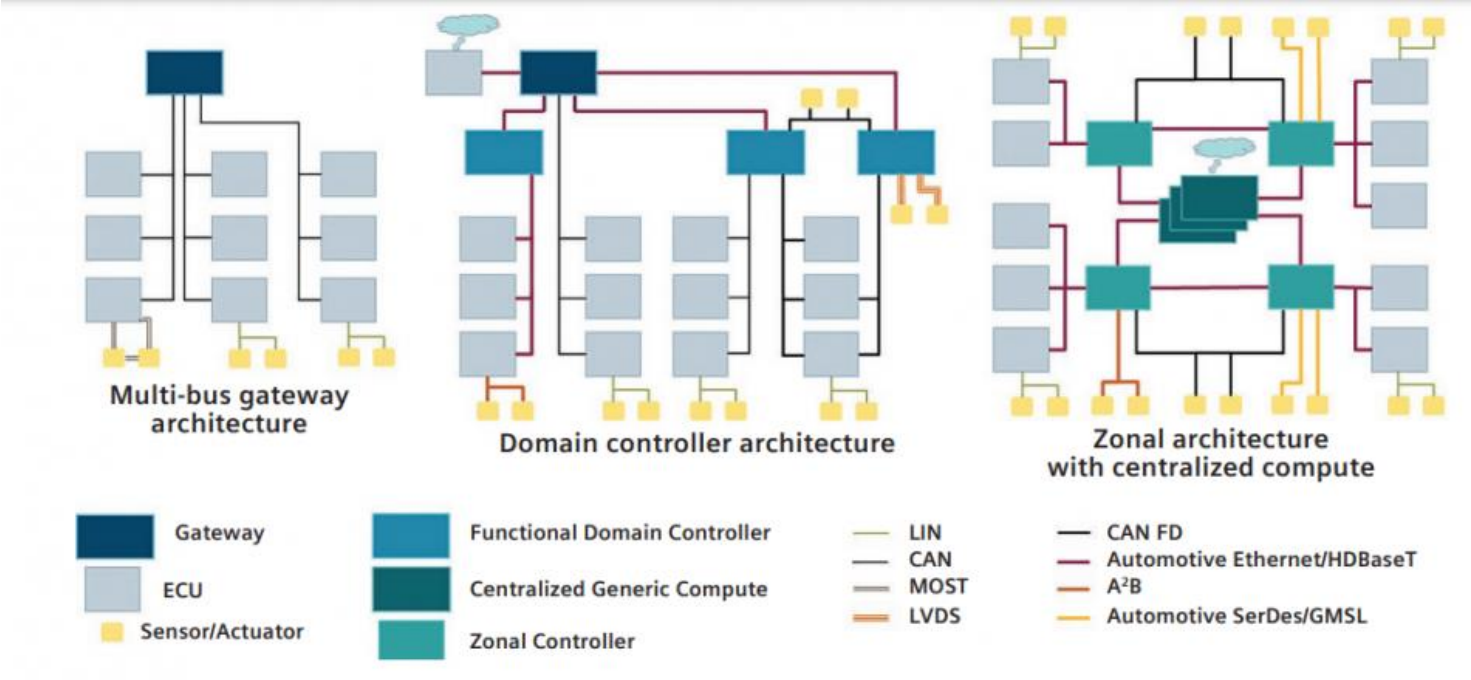
isolate all the things

containerd

Shared hosting-dedicated hosting-cloud hosting

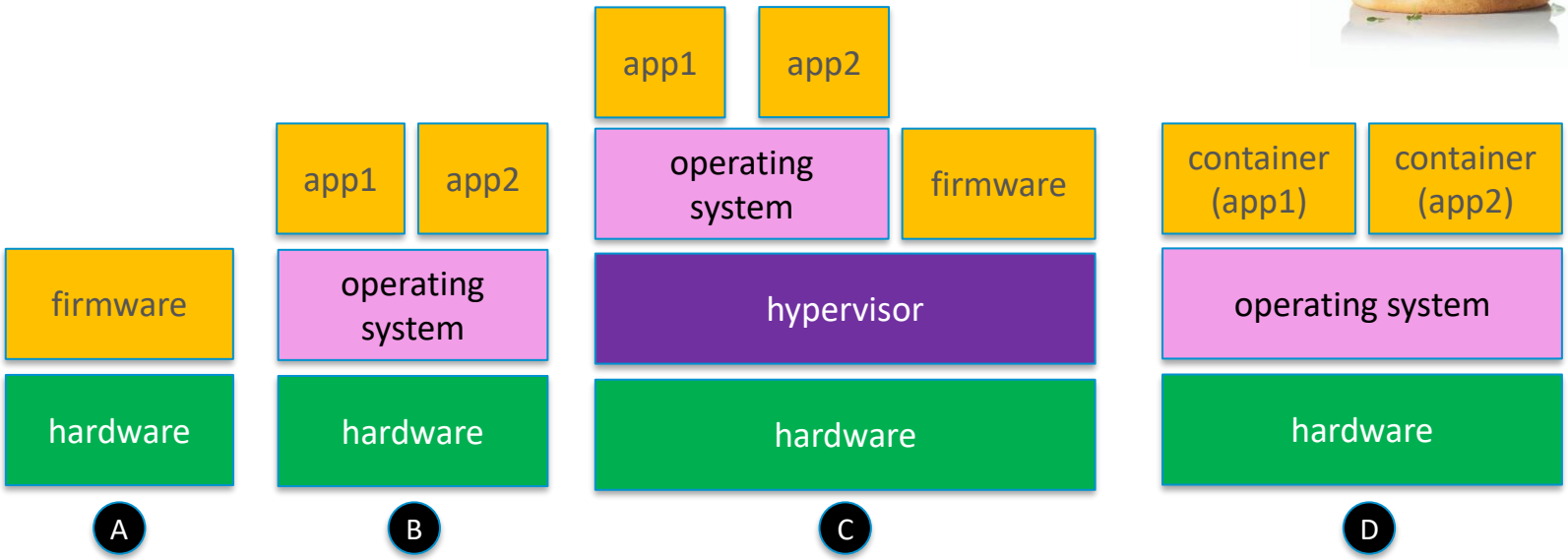


E/E Architectures in Transition



https://www.techdesignforums.com/blog/2021/11/17/balancing-the-requirements-of-e-e-architectures-for-automotive-design/

Container, VM, Hypervisor, Process, …



Container Internals

separate **network and routing**
(network namespace)

processes view
(pid namespace)

limit resources
(control groups)

separate **file system**
(mount namespace)

Capabilities

"Starting with kernel 2.2, Linux divides the privileges traditionally associated with superuser into distinct units, known as capabilities, which can be independently enabled and disabled."

http://man7.org/linux/man-pages/man7/capabilities.7.html

- Break down root into (32) smaller pieces.
- Ability to acquire or drop capabilities at runtime

CAP_CHOWN
CAP_DAC_OVERRIDE
CAP_MKNOD
CAP_NET_ADMIN
CAP_NET_BIND_SERVICE
CAP_NET_RAW
CAP_SYS_ADMIN
CAP_SYS_BOOT
CAP_SYS_CHROOT
(...)

```
root@d0ac6a80c675:/# capsh --print
Current: cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,
cap_net_raw,cap_sys_chroot,cap_mknod,cap_audit_write,cap_setfcap=ep
Bounding set =cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_ser
vice,cap_net_raw,cap_sys_chroot,cap_mknod,cap_audit_write,cap_setfcap
Ambient set =
Current IAB: !cap_dac_read_search,!cap_linux_immutable,!cap_net_broadcast,!cap_net_admin,!cap_ipc_lock,!cap_ipc_owner,!cap
_sys_module,!cap_sys_rawio,!cap_sys_ptrace,!cap_sys_pacct,!cap_sys_admin,!cap_sys_boot,!cap_sys_nice,!cap_sys_resource,!ca
p_sys_time,!cap_sys_tty_config,!cap_lease,!cap_audit_control,!cap_mac_override,!cap_mac_admin,!cap_syslog,!cap_wake_alarm,
!cap_block_suspend,!cap_audit_read,!cap_perfmon,!cap_bpf,!cap_checkpoint_restore
Securebits: 00/0×0/1'b0 (no-new-privs=0)
 secure-noroot: no (unlocked)
 secure-no-suid-fixup: no (unlocked)
 secure-keep-caps: no (unlocked)
 secure-no-ambient-raise: no (unlocked)
uid=0(root) euid=0(root)
```

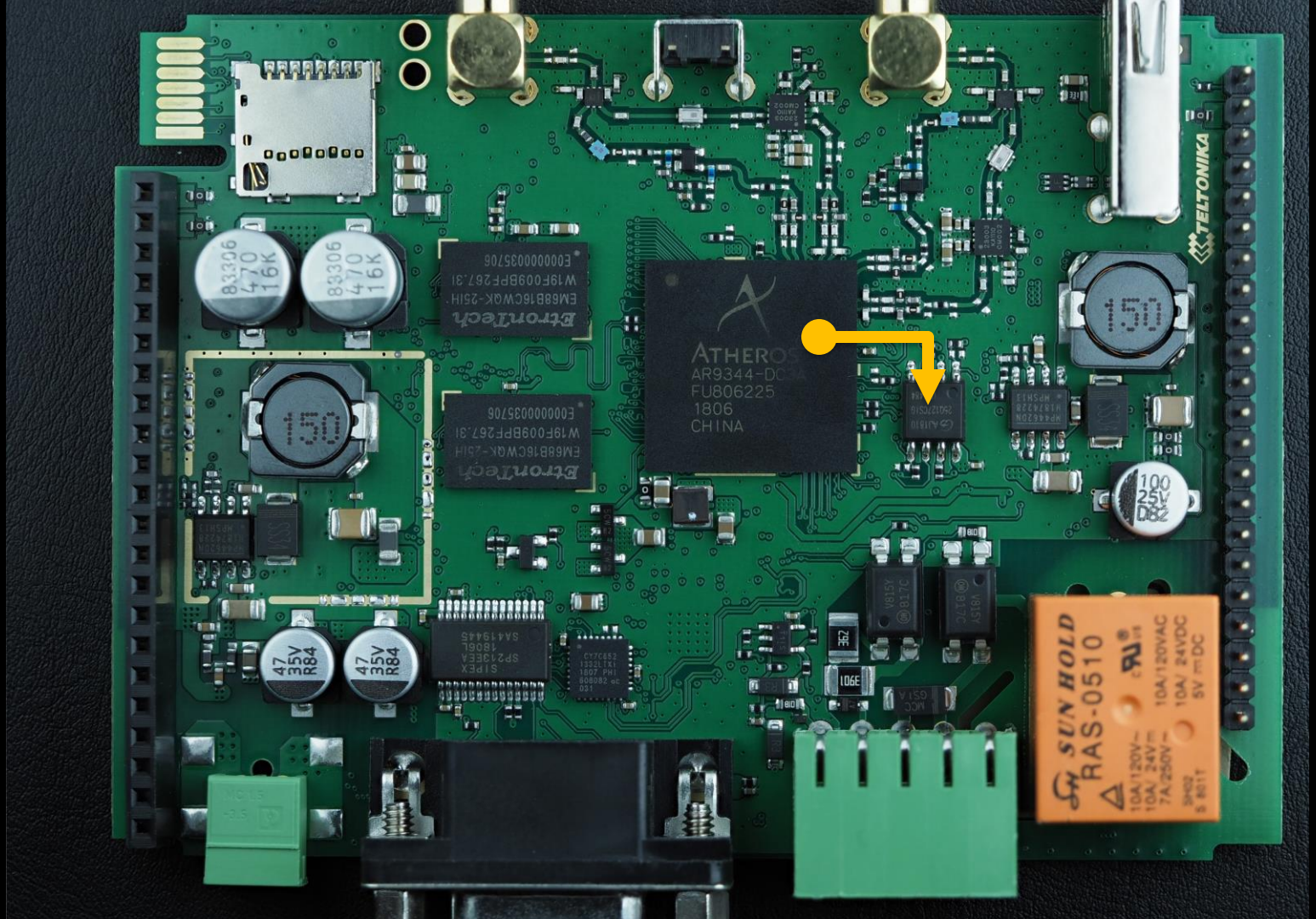
container × host × kali@kali: ~/ebpf/spyspi × kali@kali: ~/dbus ×

```
┌──(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]
└─$
```

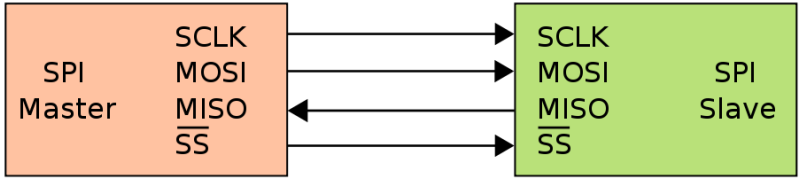
Demo: create a container with runc

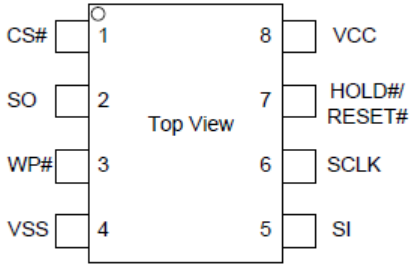
Securing the Bus

everybody listens


TELTONIKA
ATHEROS
AR9344-DC
FU806225
1806
CHINA
SUN HOLD
RAS-0510
150
150
83306
470
16K
83306
470
16K
47
35V
R84
47
35V
R84
100
25V
D82

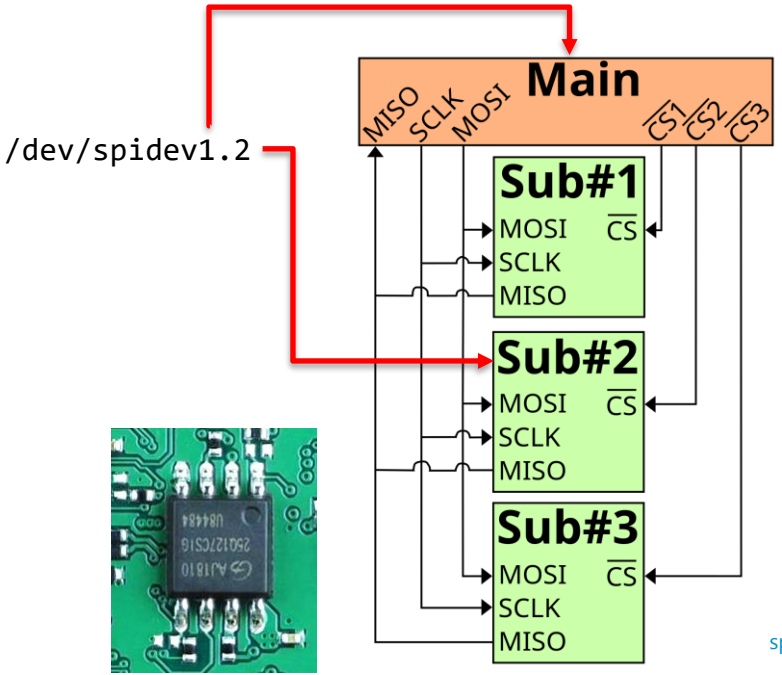
Example: Flash chip with SPI

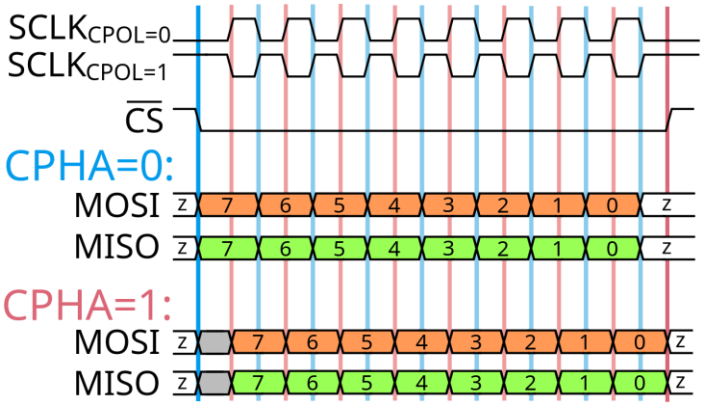




8 – LEAD SOP/VSOP/DIP

SPI topology and signals





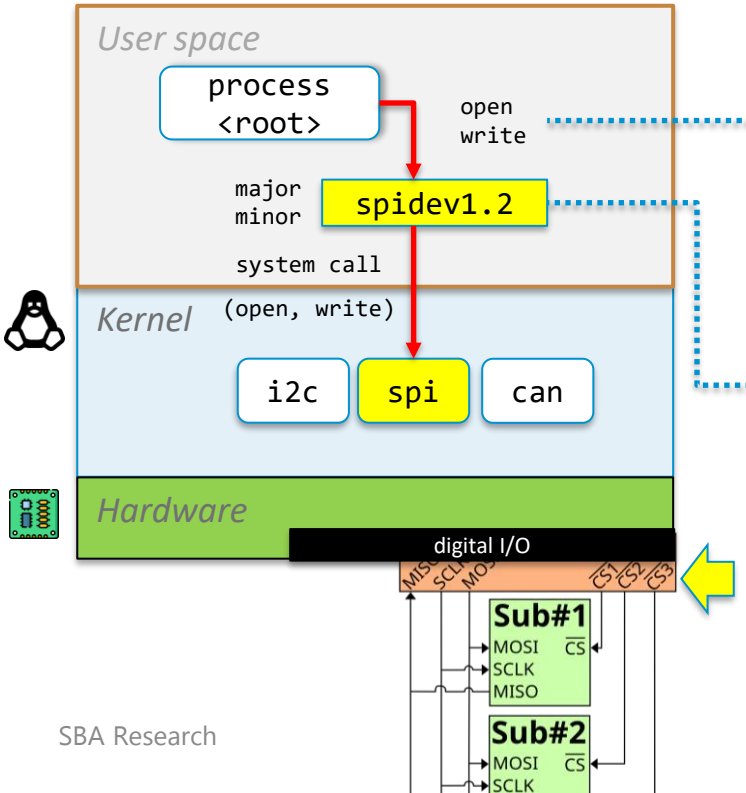
Images: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface

spi (Linux Kernel): https://www.kernel.org/doc/Documentation/spi/spi-summary

Spidev: https://www.kernel.org/doc/Documentation/spi/spidev

Python Library (py-spidev): https://pypi.org/project/spidev/

Container Security Mechanisms



Capabilities

CAP_MKNOD, CAP_DAC_OVERRIDE, ...

Filesystem permissions (DAC) for /dev/spidevX.Y (read/write for owner, group, others)

Control Group, eBPF program (device control)

read (r), write (w), mknod (m)

device type, major number, minor number

Mandatory Access Control (MAC) and eBPF

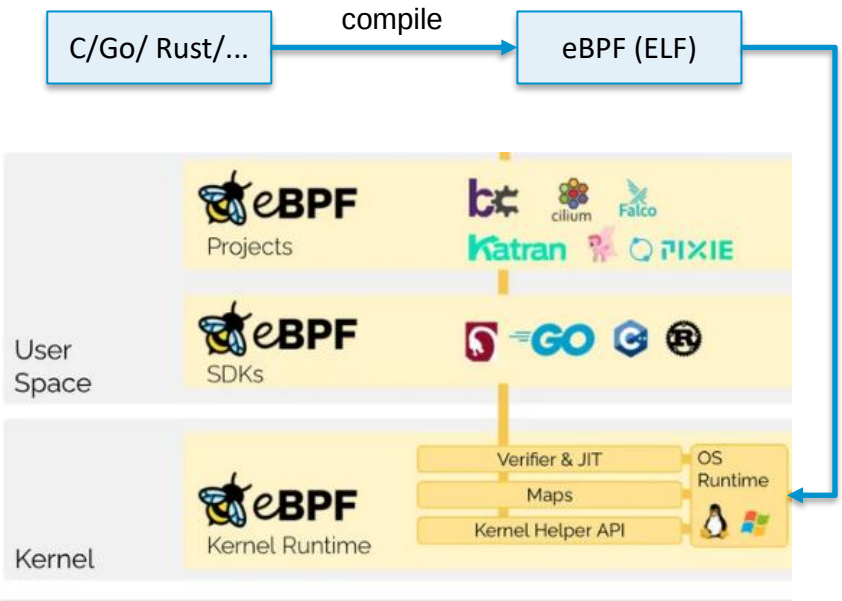
additional Kernel protections (LSM/SeLinux/AppArmor/eBPF)

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

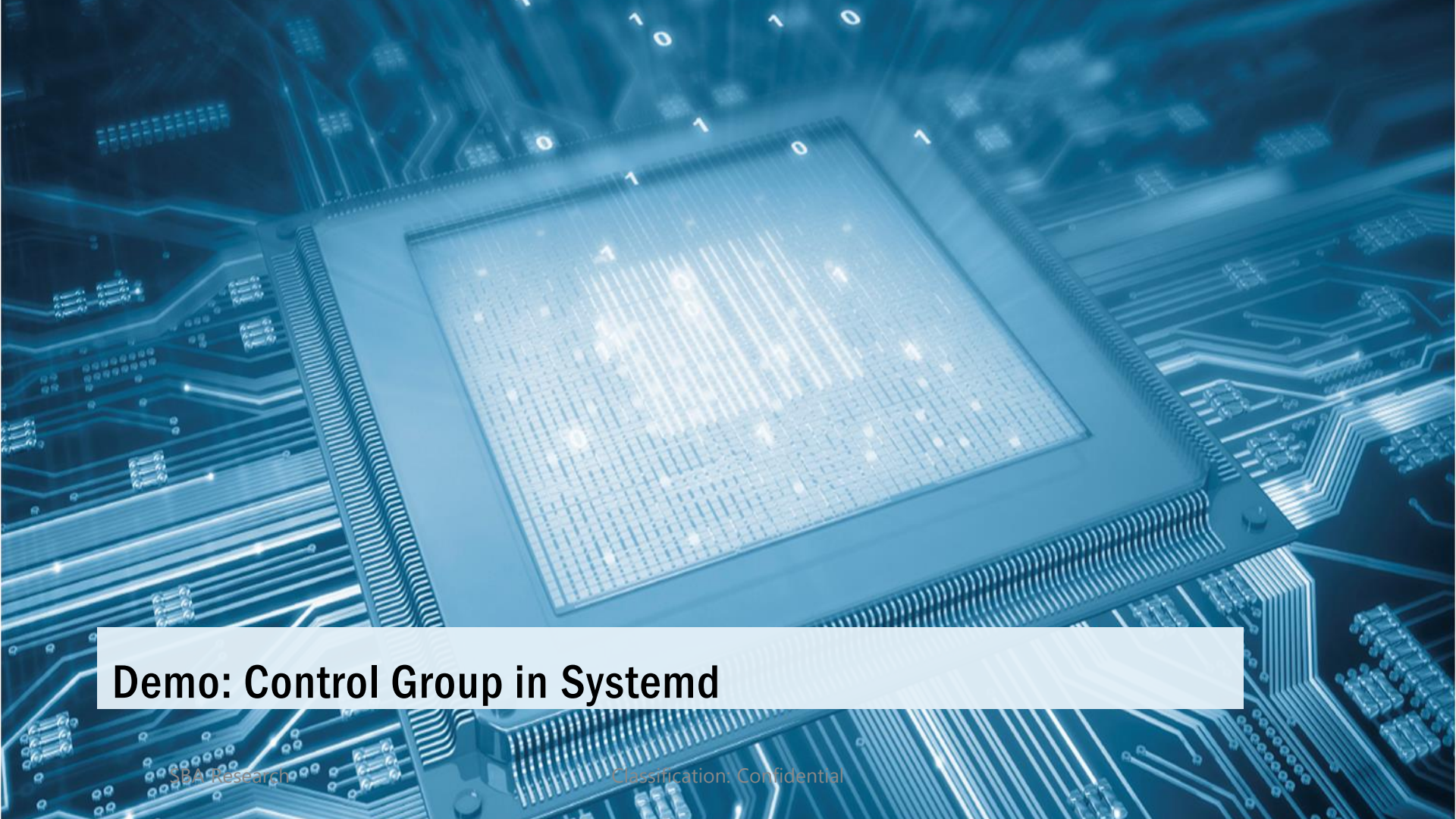


Trace points and Probes

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



Demo: Control Group in Systemd

container × host ×

```
root@crun:/#
```

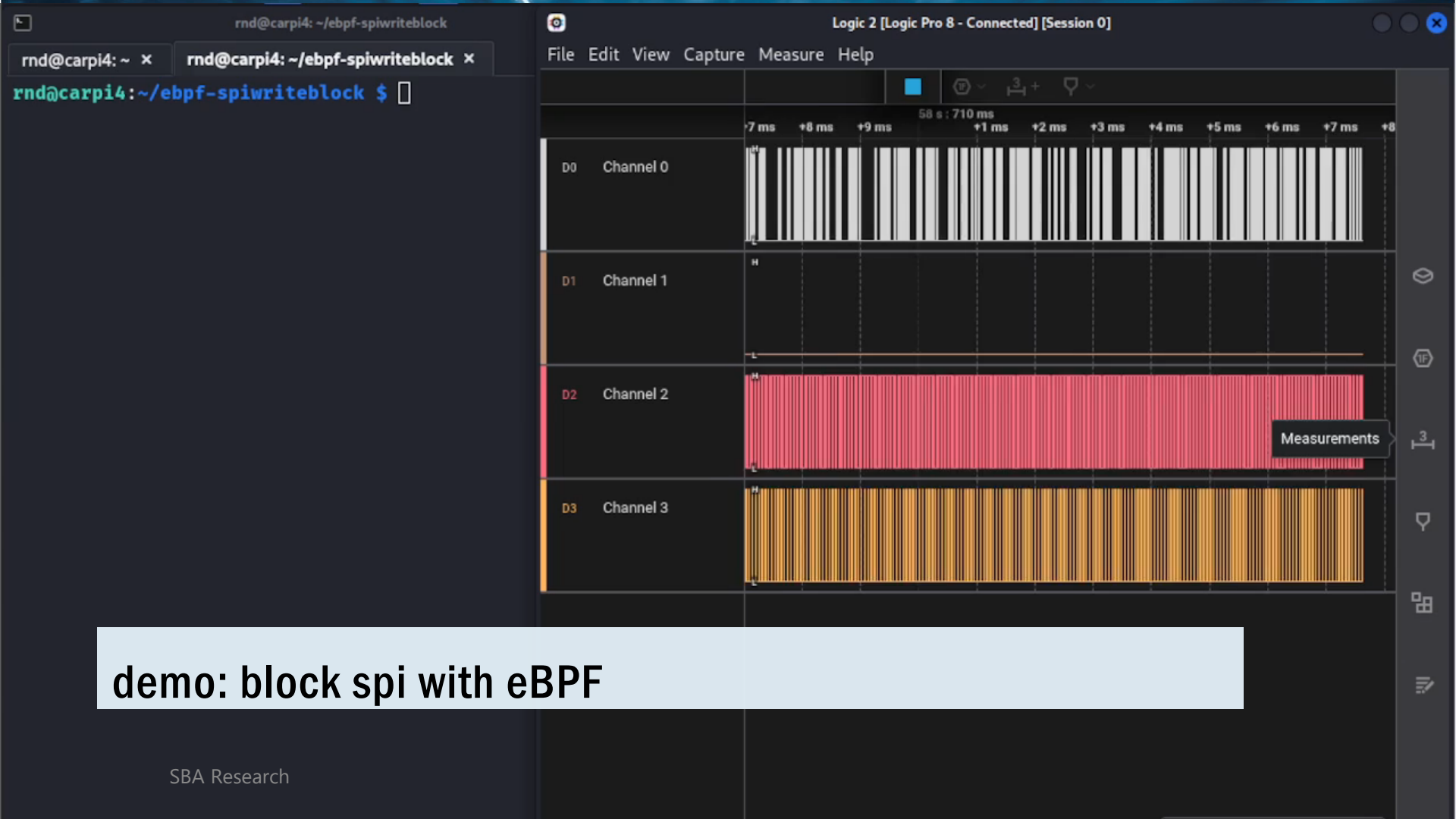
container × host ×

```
┌──(kali㉿kali)-[~]
└─$ bpftool cgroup detach /sys/fs/cgroup/user.slice/user-1000.slice/telematics_container id 604
Error: too few parameters for cgroup detach

┌──(kali㉿kali)-[~]
└─$ bpftool cgroup detach /sys/fs/cgroup/user.slice/user-1000.slice/telematics_container id 604

┌──(kali㉿kali)-[~]
└─$ bpftool cgroup
Error: 'cgroup' needs at least 1 arguments, 0 found

┌──(kali㉿kali)-[~]
└─$ bpftool
Usage: bpftool [OPTIONS] OBJECT { COMMAND | help }
       bpftool batch file FILE
       bpftool version

       OBJECT := { prog | map | link | cgroup | perf | net | feature | btf | gen | struct_ops | iter }
       OPTIONS := { {-j├──json} [{-p├──pretty}] | {-d├──debug} |
                    {-V├──version} }

┌──(kali㉿kali)-[~]
└─$ bpftool cgroup detach /sys/fs/cgroup/user.slice/user-1000.slice/telematics_container device id 604
```

container

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

```
root@crun:~#
```



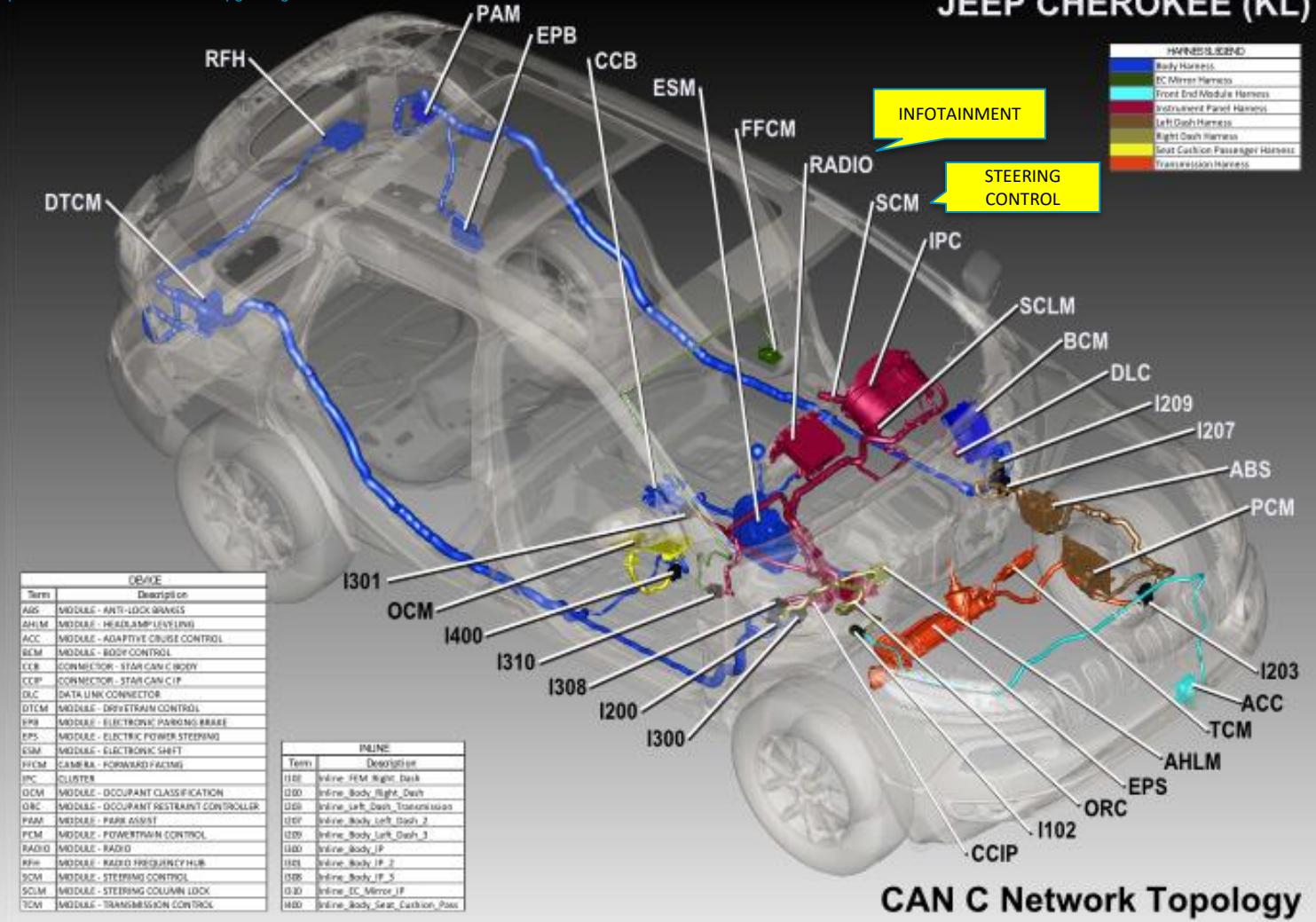
demo: block spi with eBPF

eBPF for in-vehicle networks

you CAN

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

JEEP CHEROKEE (KL)

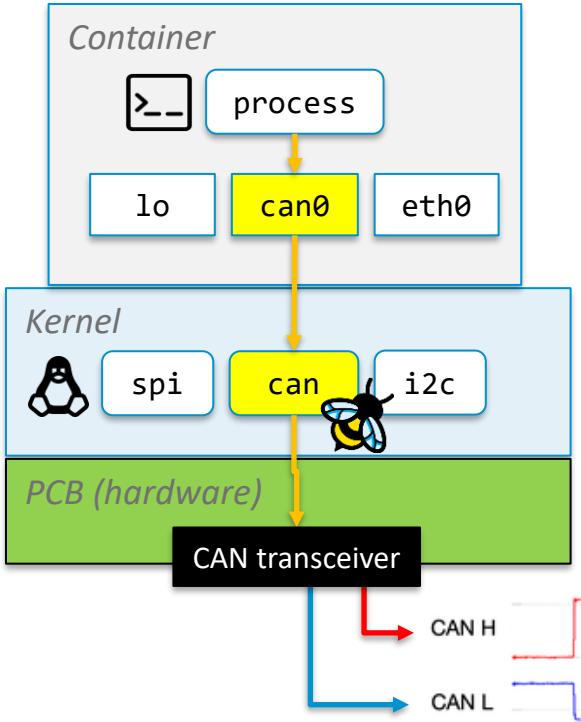


DEVICE	
Term	Description
ABS	MODULE - ANTI-LOCK BRAKES
AHLM	MODULE - HEADLAMP LEVELING
ACC	MODULE - ADAPTIVE CRUISE CONTROL
BCM	MODULE - BODY CONTROL
CCB	CONNECTOR - STAR CAN C BODY
CCIP	CONNECTOR - STAR CAN C IP
DLC	DATA LINK CONNECTOR
DTCM	MODULE - DRIVETRAIN CONTROL
EPB	MODULE - ELECTRONIC PARKING 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

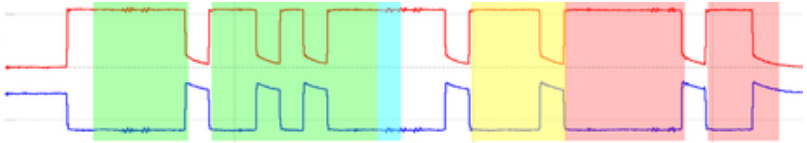
INLINE	
Term	Description
I102	Inline_FEM_Right_Dash
I200	Inline_Body_Right_Dash
I203	Inline_Left_Dash_Transmission
I207	Inline_Body_Left_Dash_2
I209	Inline_Body_Left_Dash_3
I300	Inline_Body_IP
I301	Inline_Body_IP_2
I308	Inline_Body_IP_5
I310	Inline_EC_Mirror_IP
I400	Inline_Body_Seat_Cushion_Pass

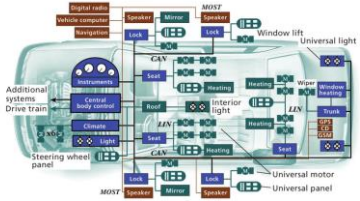
CAN C Network Topology

eBPF for CAN networks (Linux)




```
└─$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueu
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
    link/ether 00:0c:29:a1:fe:ac brd ff:ff:ff:ff:ff:
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu
    link/ether 02:42:33:80:62:b1 brd ff:ff:ff:ff:ff:
4: can0: <NOARP,UP,LOWER_UP> mtu 72 xdpgeneric qdisc
    link/can
    prog/xdp id 392
```





kali@kali: ~/ebpf-can/can-xdp-id-filter

File Actions Edit View Help

```
can0  123   [8]  5A 0B 78 77 11 C4 C5 16
can0  123   [6]  1D 63 A1 26 53 EA
can0  123   [6]  24 36 F5 27 94 F8
can0  123   [8]  2A 81 AD 46 E5 C0 63 19
can0  123   [8]  2D 5F A7 34 BD 85 0A 07
can0  123   [7]  F3 C9 7F 31 39 FA F1
can0  123   [8]  3F 9A D8 52 E4 C8 55 42
can0  123   [8]  B3 D5 36 78 EA 95 55 7C
can0  123   [8]  DD 19 54 20 E6 32 E9 24
can0  123   [8]  19 CA DC 42 F7 9E 31 38
can0  123   [8]  32 86 EE 55 51 AA A9 2F
can0  123   [8]  98 7D 9F 7F 6F 0D 4B 56
can0  123   [1]  6E
can0  123   [5]  26 F7 47 49 BE
can0  123   [8]  42 61 5F 24 EB 23 95 79
can0  123   [8]  D9 28 79 3D DE ED 14 2B
can0  123   [1]  D5
can0  123   [5]  2F 90 63 66 D0
can0  123   [8]  D7 1D 54 65 AD 77 78 16
can0  123   [6]  F6 74 31 17 C7 41
can0  123   [8]  D3 74 71 54 F9 C7 43 2F
can0  123   [8]  C1 38 77 28 91 45 E3 2E
can0  123   [8]  BB CA E5 6E 17 EB BD 79
can0  123   [6]  51 7B 3E 6B 46 7B
```

kali@kali: ~/ebpf-can/can-xdp-id-filter

File Actions Edit View Help

kali@ka...-filter × kali@ka...-filter × kali@ka...-filter × kali@ka...-filter × kali@ka...-filter ×

```
    bpf_printk("ID %x %d = %d", can_id, can_id, ret);
    if(ret < 0)
        return XDP_DROP;

    bpf_probe_read_kernel(&can_data, 8, frame->data);
    bpf_probe_read_kernel(&value, sizeof(__u32), ret);
    switch(value) {
        case 0: // XDP_ABORTED
        case 1: // XDP_DROP
        case 2: // XDP_DROP
            return XDP_DROP;
        case 3: // XDP_PASS
        case 4: // XDP_TX
        case 5: // XDP_REDIRECT
            return XDP_PASS;
        case 6: // LEN
            if(check_dlc(frame->can_dlc))
            {
                return XDP_PASS;
            } else {
                return XDP_DROP;
            }
            break;
        case 7: // CRC
            if(check_crc(can_data))
            {
                return XDP_PASS;
            } else {
                return XDP_DROP;
            }

    // default block
    return XDP_DROP;
}
```

89,17-24 97%

Demo: eBPF on XDP for CAN – trace and filter

Offensive eBPF

the Kernel turns against you!

eBPF, I thought we were friends !

Guillaume Fournier
Sylvain Afchain

August 2021

DEFCON

DATADOG

Warping Reality

Creating and countering the next generation of Linux rootkits using eBPF

Pat Hogan
@PathToFile

DEFCON 29

generation of Linux rootkits using eBPF | PatH | Creating

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Cross Container Attacks: The Bewildered eBPF on Clouds

Yi He and Roland Guo, *Tsinghua University and BNRist;* Yunlong Xing, *George Mason University;* Xijia Che, *Tsinghua University and BNRist;* Kun Sun, *George Mason University;* Zhuotao Liu, Ke Xu, and Qi Li, *Tsinghua University*

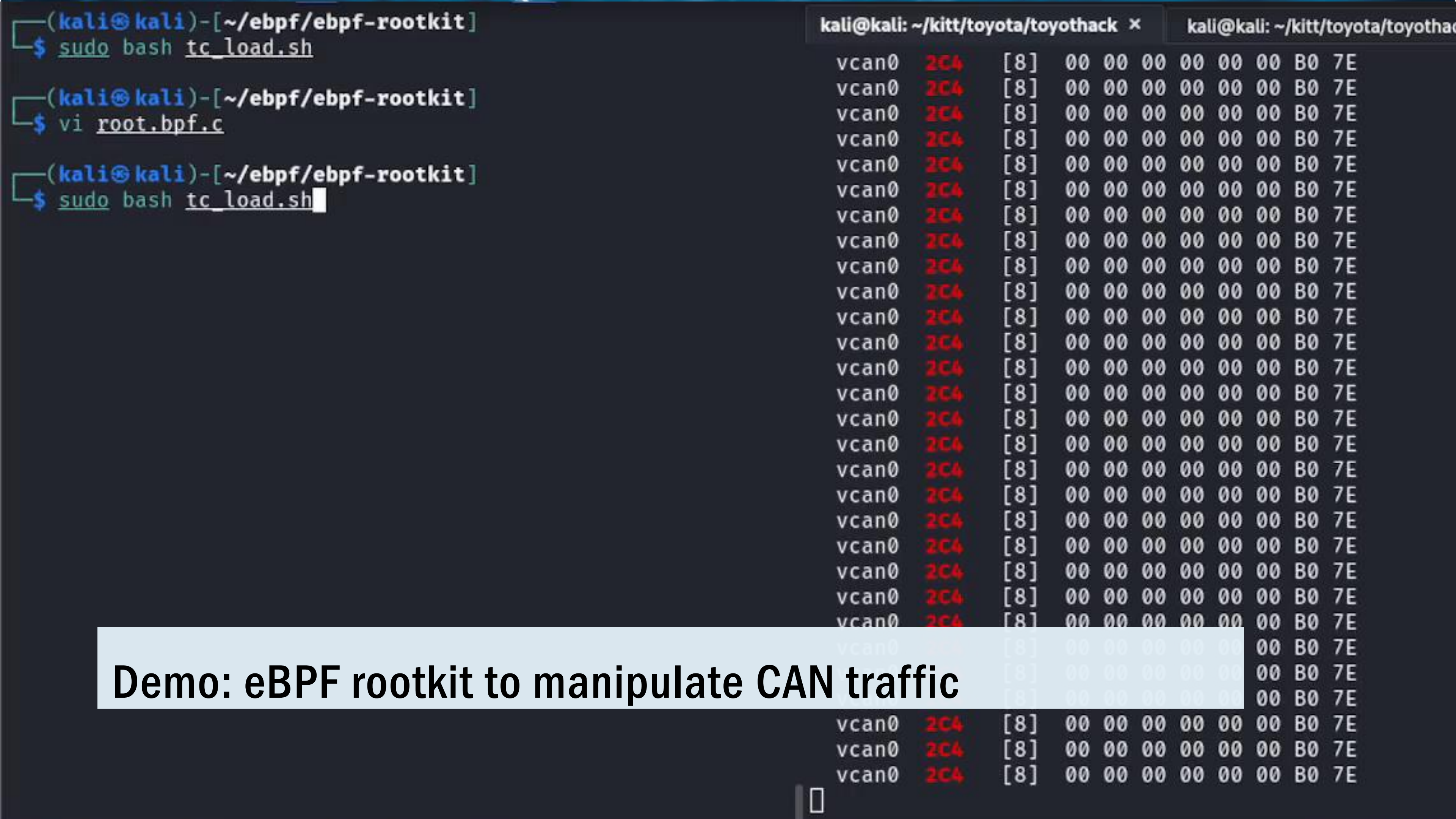
https://www.usenix.org/conference/usenixsecurity23/presentation/he

This paper is included in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Open access to the Proceedings of the 32nd USENIX Security Symposium is sponsored by USENIX.

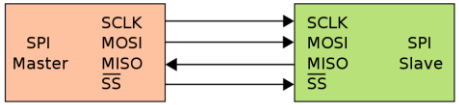


```
┌──(kali㉿kali)-[~/ebpf/ebpf-rootkit]
└─$ sudo bash tc_load.sh

┌──(kali㉿kali)-[~/ebpf/ebpf-rootkit]
└─$ vi root.bpf.c

┌──(kali㉿kali)-[~/ebpf/ebpf-rootkit]
└─$ sudo bash tc_load.sh
```

```
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
  vcan0  2C4   [8]  00 00 00 00 00 00 B0 7E
```

Demo: eBPF rootkit to manipulate CAN traffic

Conclusion



Lessons learned from the Clouds

- **software-defined protections**: code instead of dedicated devices.
- **flexible architecture**: development at runtime without starting from scratch.
- **mutation and fast evolution**: rolling updates are key to react to attack patterns.

You don’t know how the next attack will be!

Limitations and further research

- **induced latency**: does the eBPF code induce unwanted side effects for real-time applications?
- **E/E architecture**: does the benefit of software-defined belance the cost to create this infrastructure?
- **maintainability**: who defines the rules?
- **observability and action**: who reacts how?

Applied Research and Consulting

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

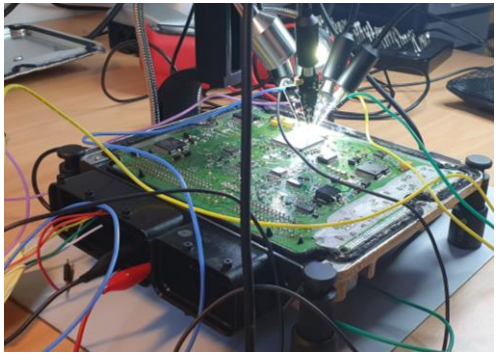
Threat Analysis	Coding and Exploitation	Container Security
Crypto Engineering	IoT Hacking	Combinatorial Security Testing

Embedded Systems · Containers · Servers and Data Centers · Industrial IoT

MATRIS Research Group

SBA Research

SBA Meetup // ASRG Vienna // eBPF Vienna // embedded Austria



Reinhard Kugler

MATRIS Research Lab

SBA Research

Floragasse 7, 1040 Vienna

rkugler@sba-research.org

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

MATRIS Research Group

SBA Research