Linux USB fuzzing

Andrey Konovalov <andreyknvl@gmail.com>

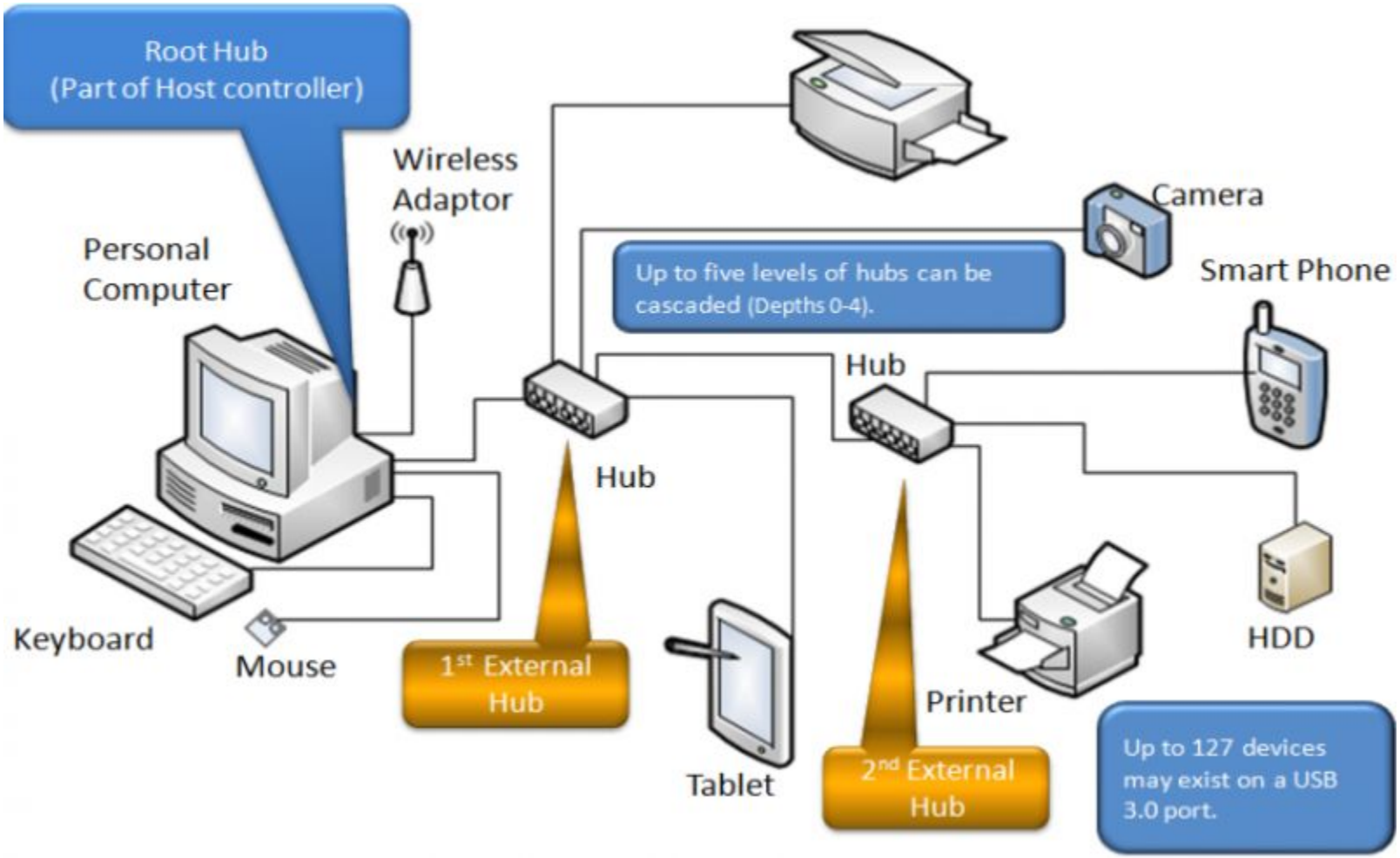
Chaos Constructions 2017, Hardware Village
August 26th 2017

Agenda

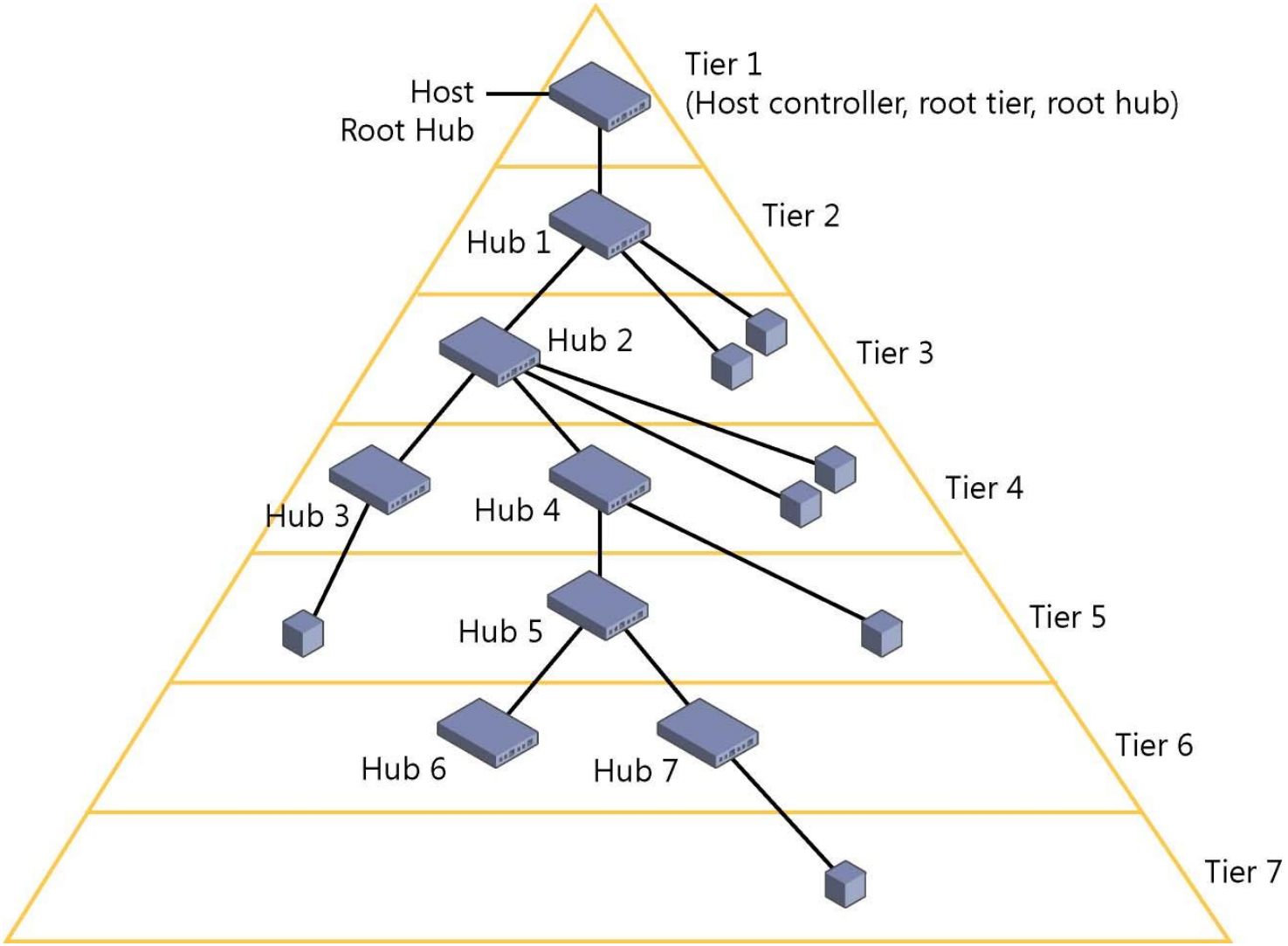
- Introduction to USB
- Linux kernel fuzzing
- Hardware USB fuzzing
- Virtual USB fuzzing

Introduction to USB

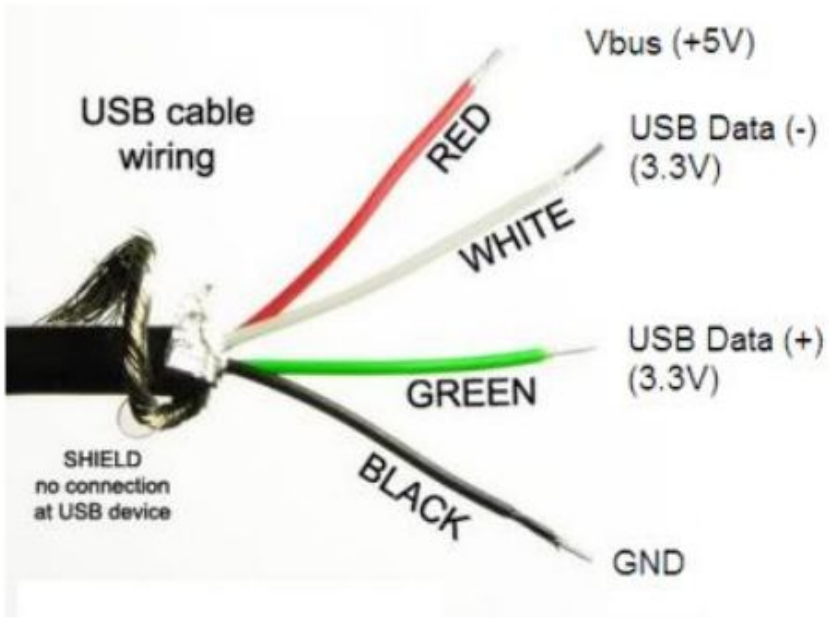
USB topology

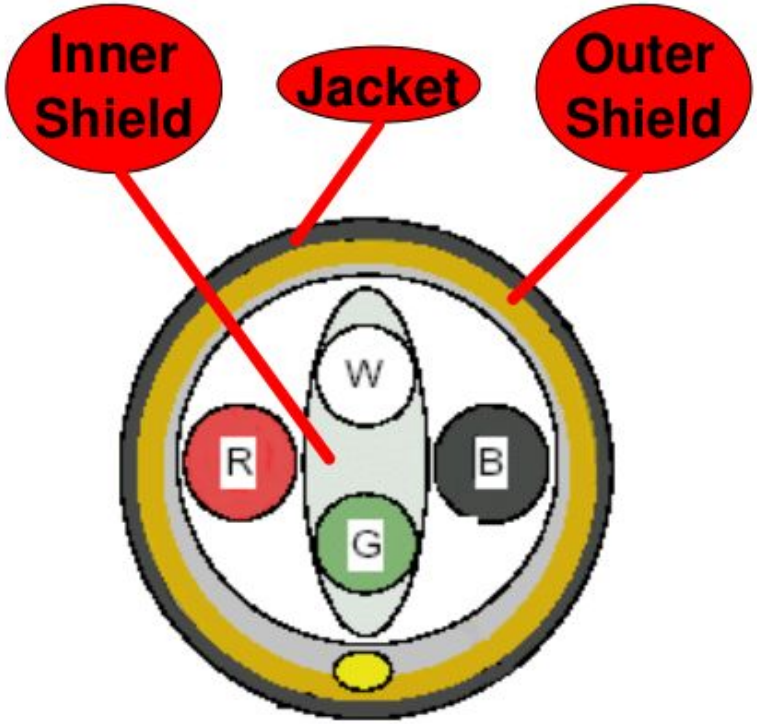


USB hubs


Tier 1
(Host controller, root tier, root hub)
Host
Root Hub
Tier 2
Hub 1
Hub 2
Tier 3
Tier 4
Hub 3
Hub 4
Tier 5
Hub 5
Tier 6
Hub 6
Hub 7
Tier 7

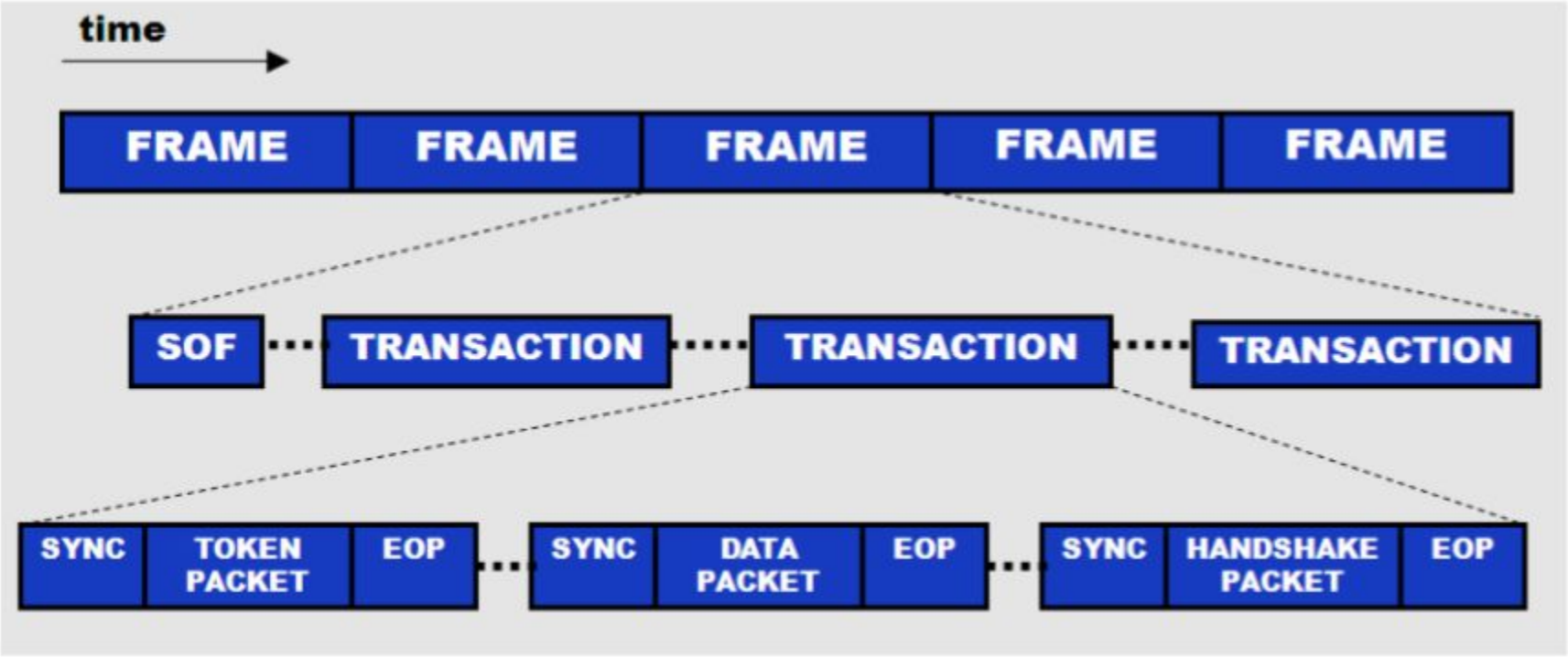
USB cable


USB cable
wiring
Vbus (+5V)
RED
USB Data (-)
(3.3V)
WHITE
USB Data (+)
(3.3V)
GREEN
SHIELD
no connection
at USB device
BLACK
GND

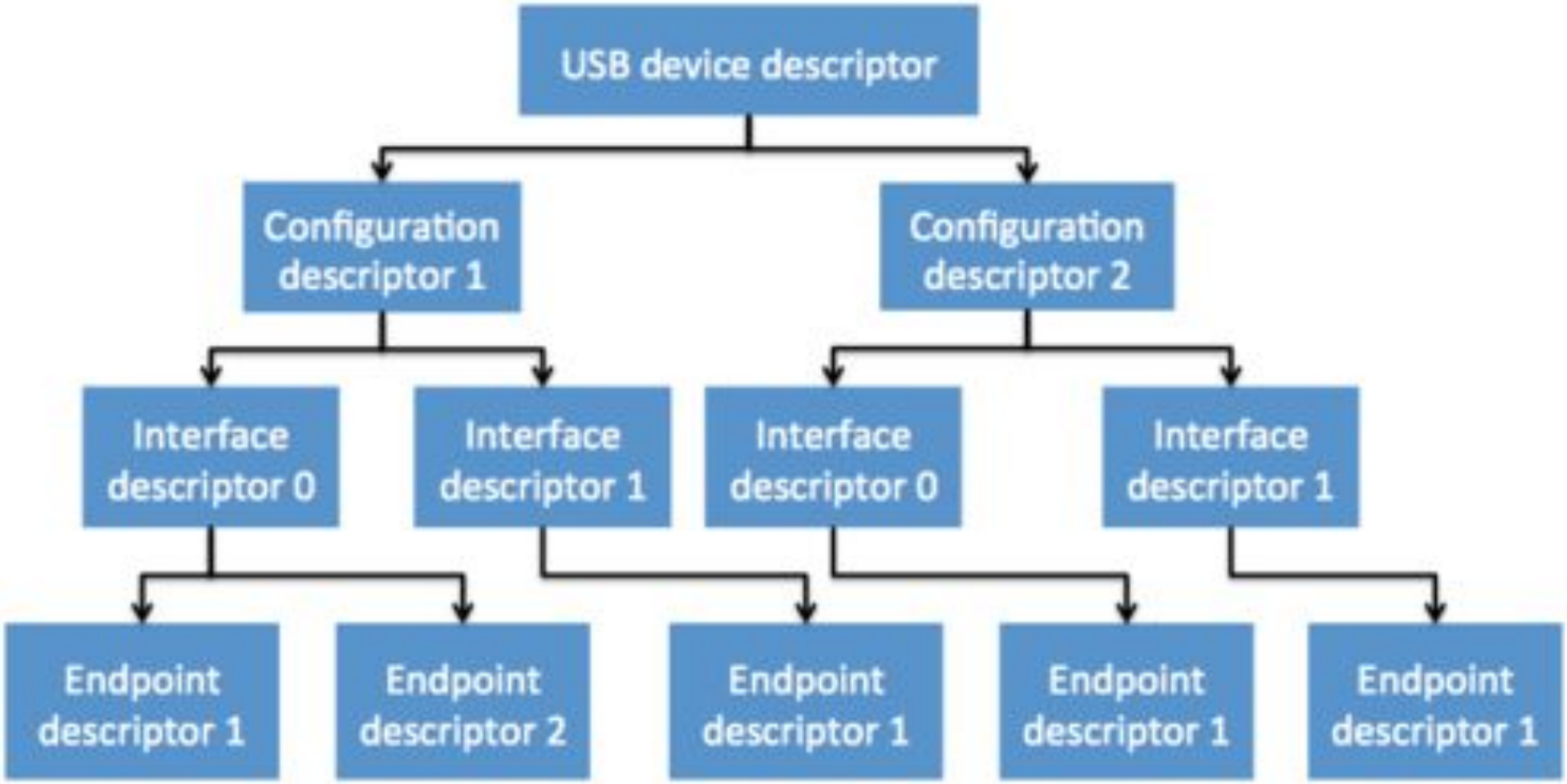

Inner
Shield
Jacket
Outer
Shield
W
R
B
G

USB connectors

USB communication



USB device descriptor

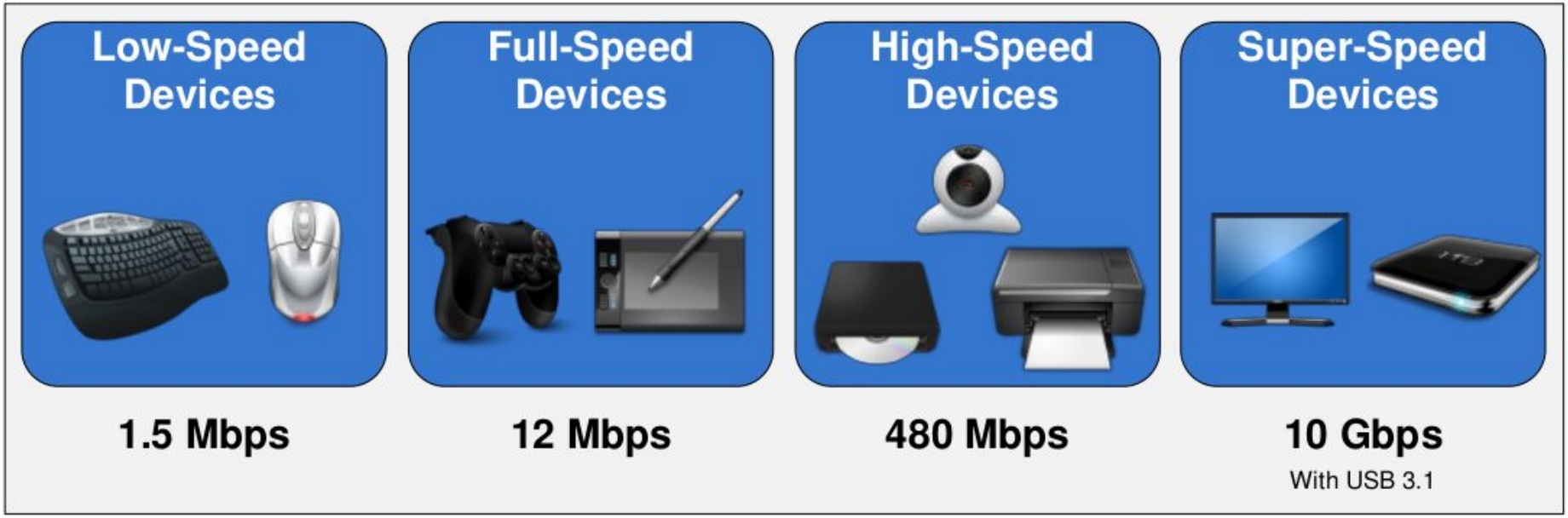


USB endpoint types

Transfer Type	Control	Interrupt	Bulk	Isochronous
Typical Use	Device Initialization and Management	Mouse and Keyboard	Printer and Mass Storage	Streaming Audio and Video
Low-Speed Support	Yes	Yes	No	No
Error Correction	Yes	Yes	Yes	No
Guaranteed Delivery Rate	No	No	No	Yes
Guaranteed Bandwidth	Yes (10%)	Yes (90%)[1]	No	Yes (90%)[1]
Guaranteed Latency	No	Yes	No	Yes
Maximum Transfer Size	64 bytes	64 bytes	64 bytes	1023 bytes (FS) 1024 bytes (HS)
Maximum Transfer Speed	832 KB/s	1.216 MB/s	1.216 MB/s	1.023 MB/s

[1]Shared bandwidth between isochronous and interrupt.

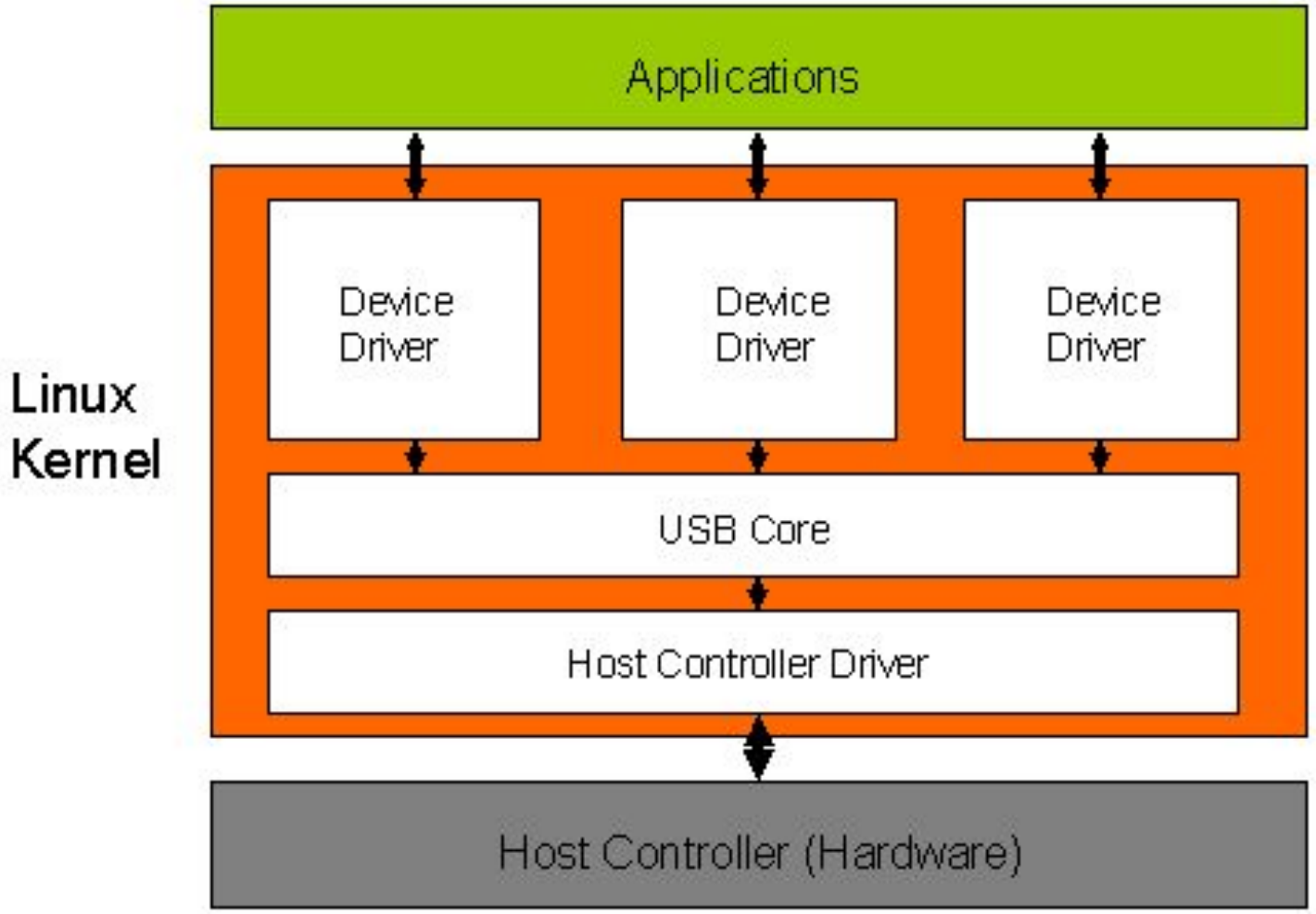
USB transfer speeds



USB descriptors

- lsusb -t
- lsusb -v

USB host


Applications
Device Driver
Device Driver
Device Driver
Linux Kernel
USB Core
Host Controller Driver
Host Controller (Hardware)

USB enumeration (simplified)

1. Device is connected to a USB port and detected
2. Host asks device for the descriptor and device replies
3. Host loads appropriate driver
4. Host sets specific device configuration
5. Done

Linux kernel fuzzing

Finding bugs in the Linux kernel

Dynamic bug finding tools:

- Bug detectors

- Fuzzers

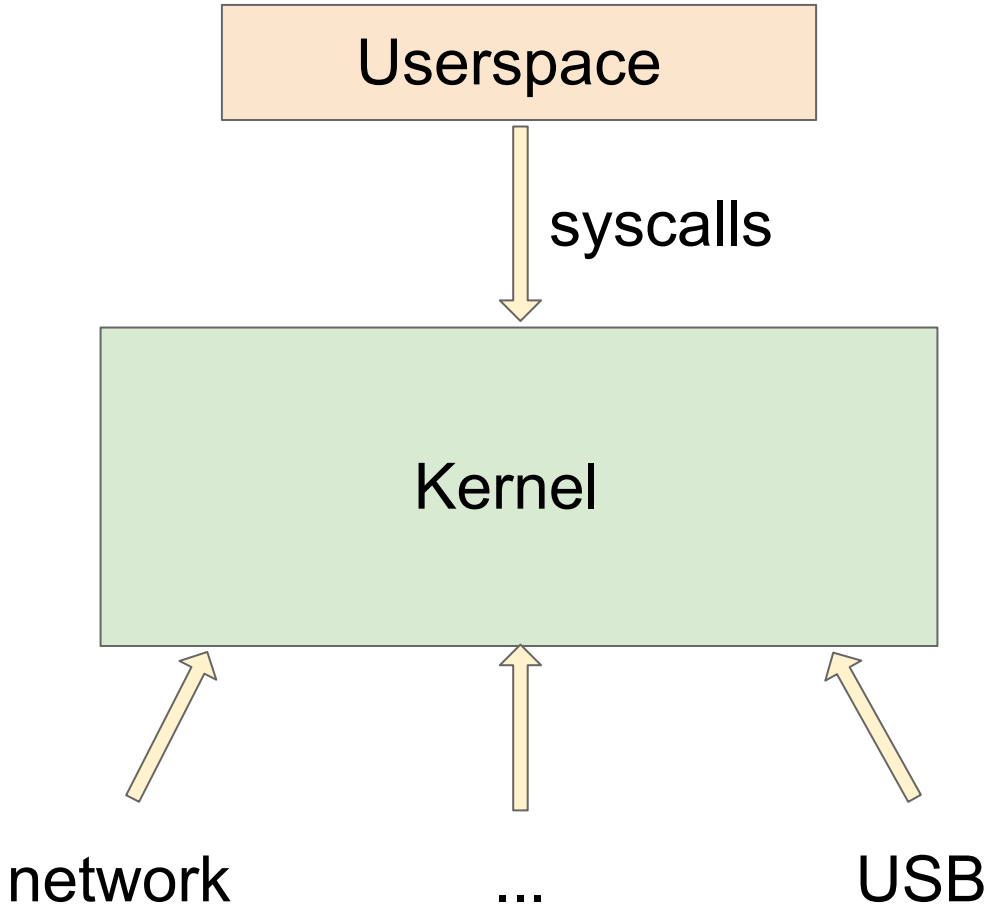
Kernel Sanitizers

- KASAN (use-after-frees and out-of-bounds)
  - CONFIG_KASAN available upstream since 4.0

- KTSAN (data-races and deadlocks)
  - prototype available at https://github.com/google/ktsan

- KMSAN (uninitialized-memory-use)
  - prototype available at https://github.com/google/kmsan

Kernel inputs



Kernel system call fuzzers

- Trinity (https://github.com/kernelslacker/trinity)

- syzkaller (https://github.com/google/syzkaller)

syzkaller

- Coverage-guided syscall fuzzer for the Linux kernel

- As of now found over 500 bugs (https://github.com/google/syzkaller/wiki/Found-Bugs)
- Dozens of CVEs
- 4 public local privilege escalation bugs over the last few months (CVE-2017-7308, CVE-2017-6074, CVE-2017-2636, CVE-2017-1000112)
- Can generate C reproducers for found bugs

Other Linux kernel fuzzers

- https://github.com/oracle/kernel-fuzzing
- https://github.com/nccgroup/TriforceLinuxSyscallFuzzer
- http://web.eece.maine.edu/~vweaver/projects/perf_events/fuzzer/
- https://github.com/schumilo/vUSBf

USB fuzzing

USB attack surface

- USB hardware (USB Kill)
- USB kernel drivers
- BadUSB

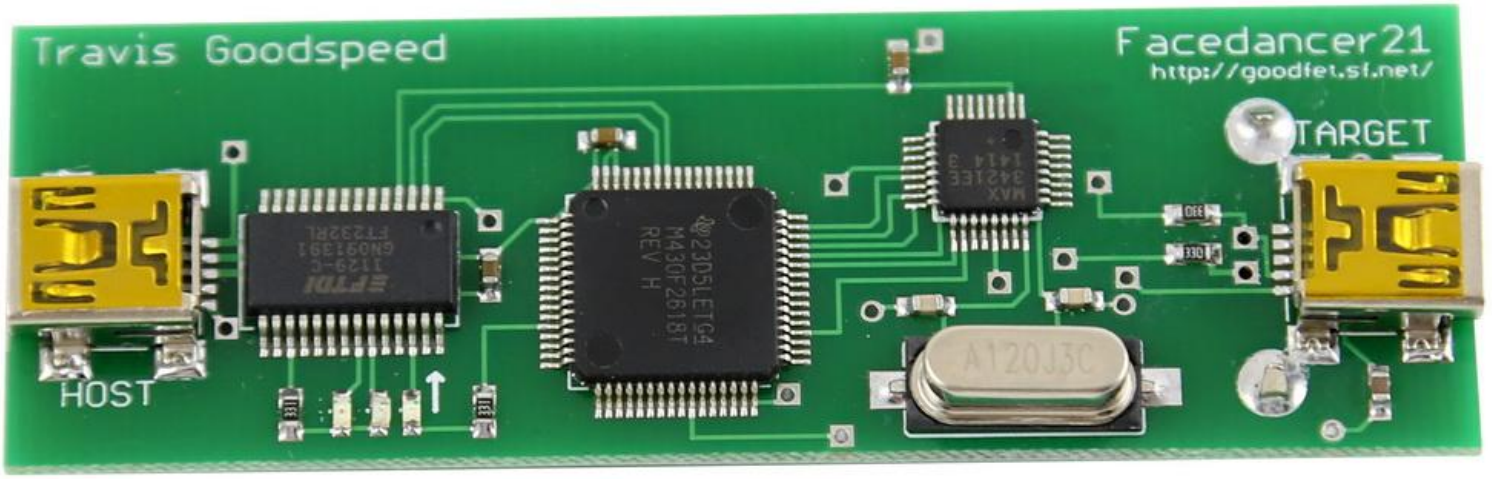
USB fuzzing

- Hardware: FaceDancer21

- In VM: vUSBf

FaceDancer21

- “The purpose of this board is to allow USB devices to be written in host-side Python, so that one workstation can fuzz-test the USB device drivers of another host”
- http://goodfet.sourceforge.net/hardware/facedancer21/



FaceDancer21

- https://github.com/travisgoodspeed/goodfet
- https://github.com/ktemkin/Facedancer
- https://github.com/nccgroup/umap
- https://github.com/nccgroup/umap2

vUSBf

- Virtual USB fuzzer
- QEMU + usbredir
- https://github.com/schumilo/vUSBf

CVE-2016-2384

- Double-free in USB-MIDI Linux kernel driver
- Found with vUSBf
- Confirmed and exploited with FaceDancer21
- https://xairy.github.io/blog/2016/cve-2016-2384

Questions?

Andrey Konovalov, andreyknvl@gmail.com