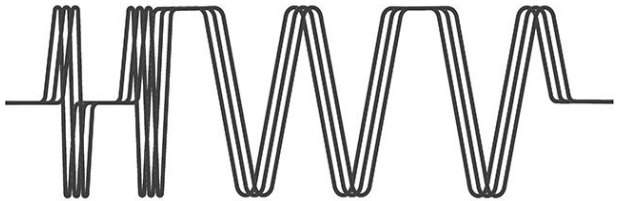


Introduction to USB Hacking

Andrey Konovalov <andreyknvl@gmail.com>

PHDays 2019, Hack Zone
May 21st 2019

Agenda

- Part 1: USB 101
- Part 2: USB Attack Surface
- Part 3: Consumer Ready BadUSB
- Part 4: Microcontroller Based BadUSB
- Part 5: Facedancer
- Part 6: Linux Gadget Subsystem
- Part 7: USB Fuzzing
- Part 8: USB Sniffing

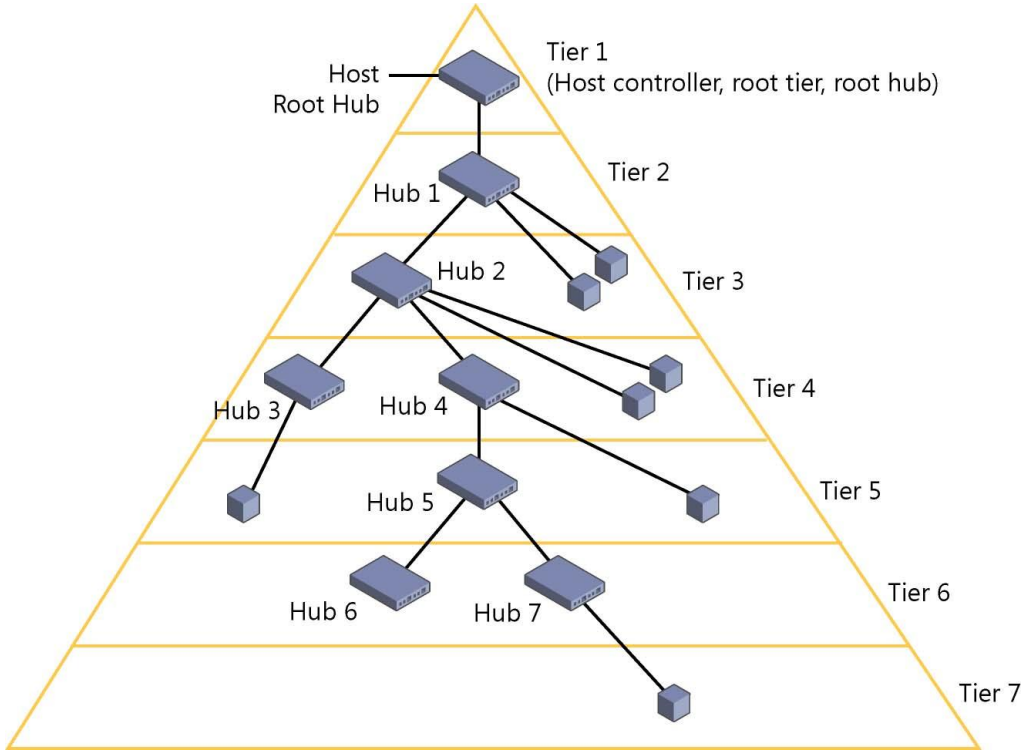
(Today mostly fuzzing related stuff)

Part 1: USB 101

USB Topology

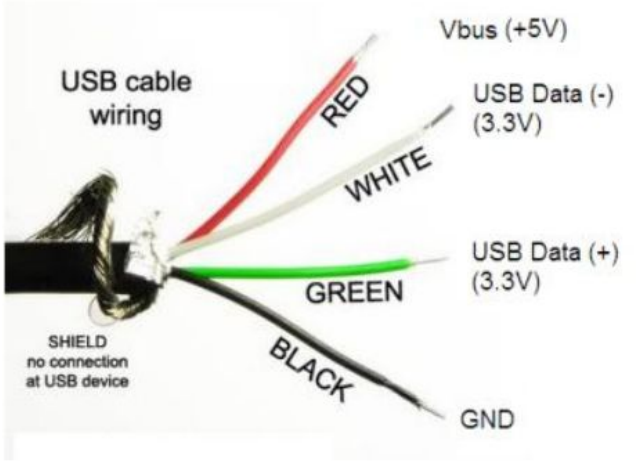
"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

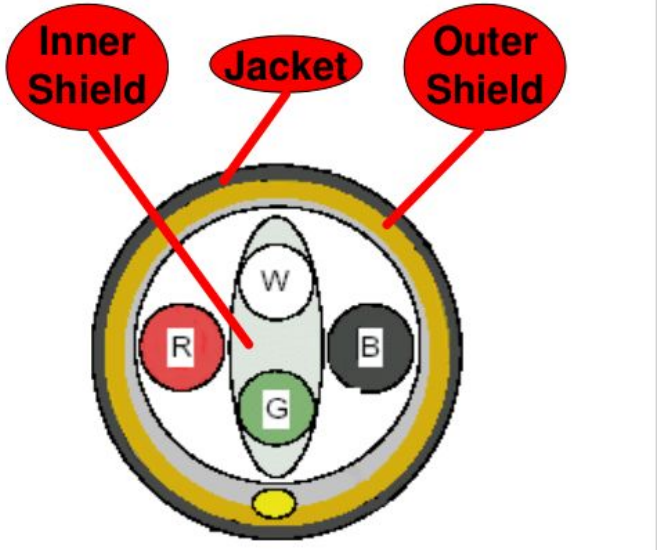
USB Hubs

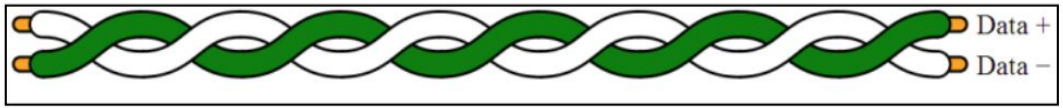


"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

USB Cable

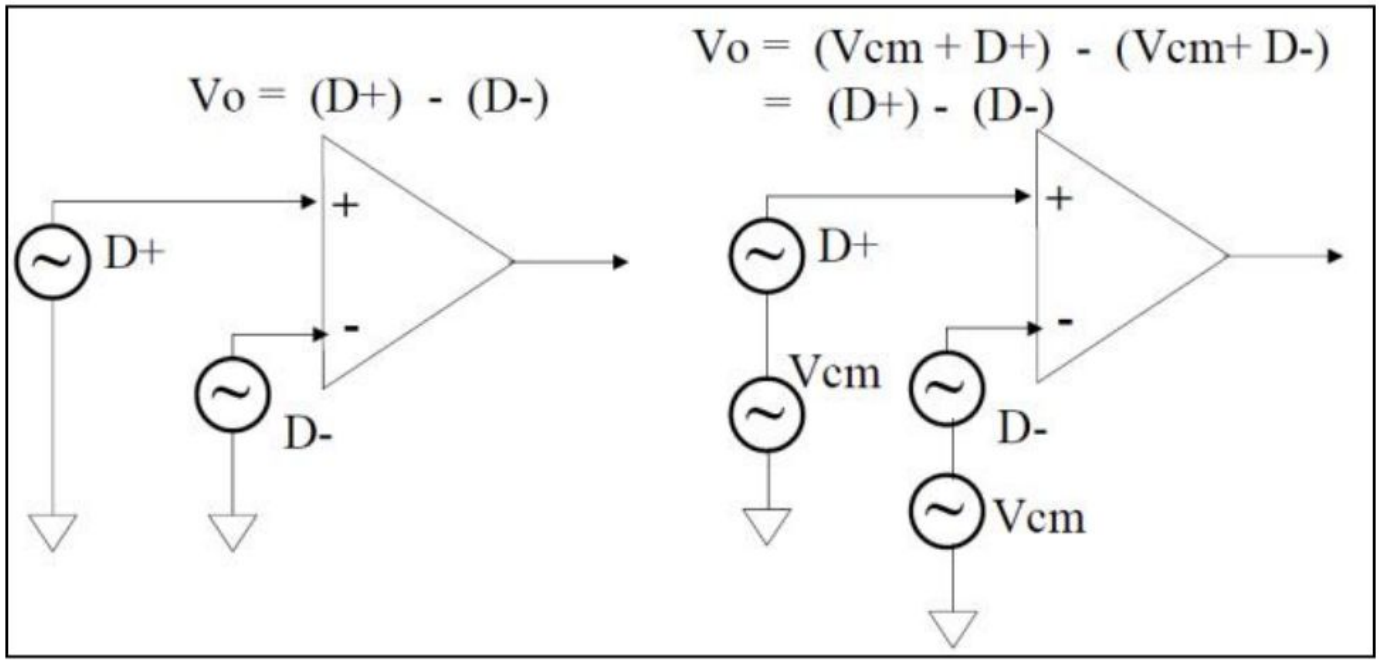






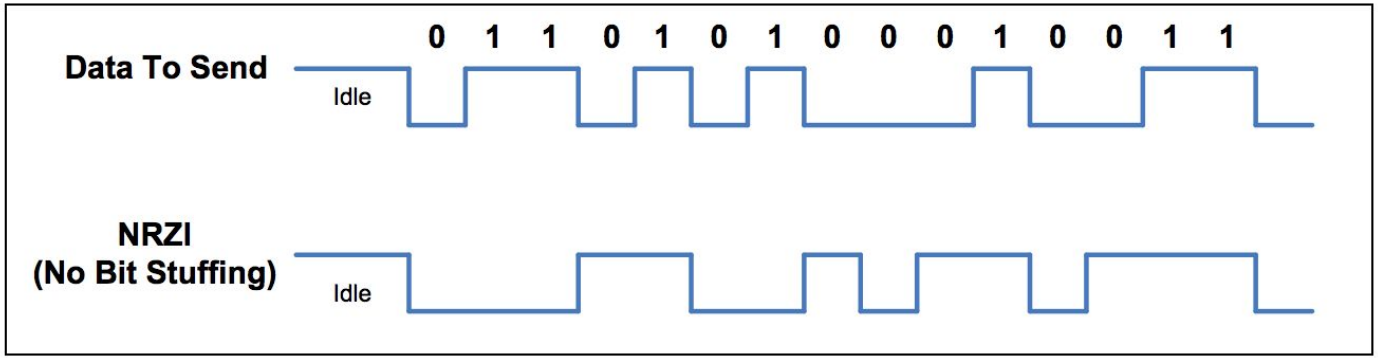
"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

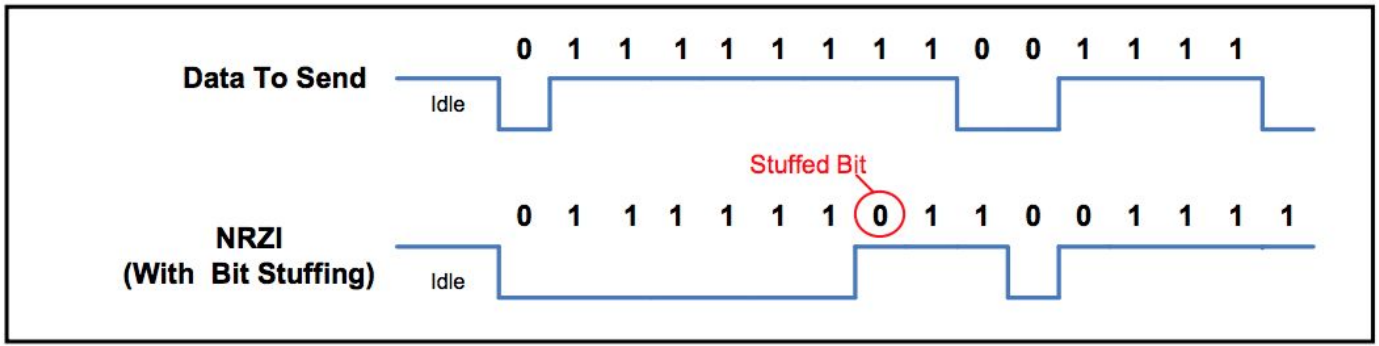
USB Differential Amplifier



"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

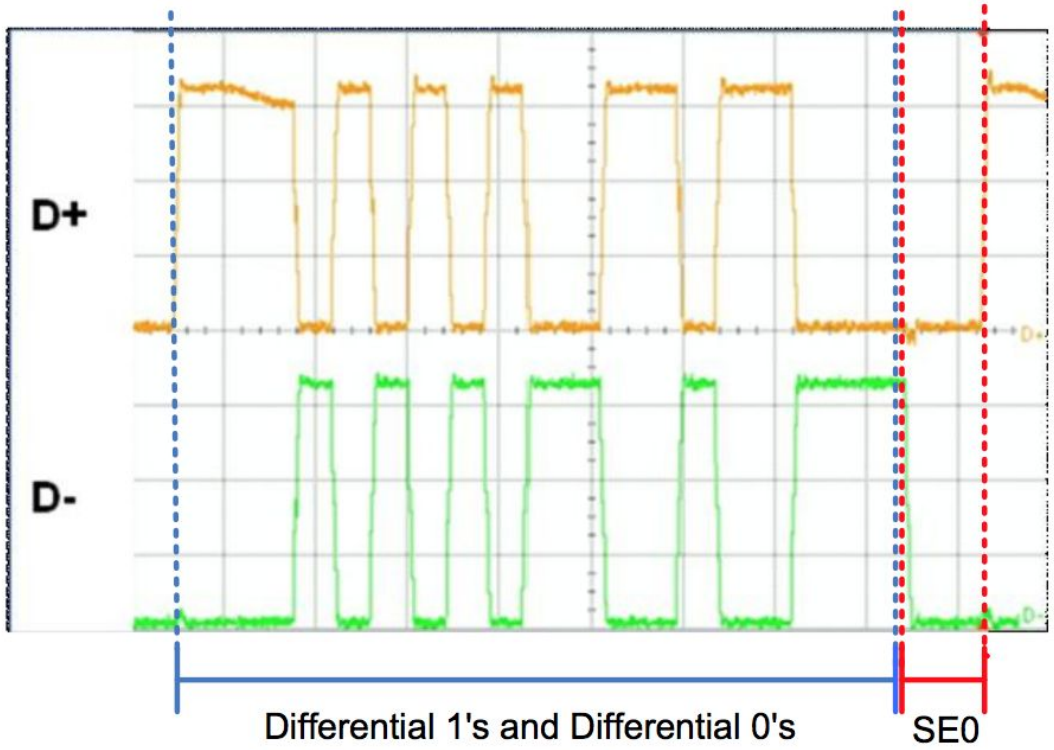
NRZI Encoding





"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

USB D+ and D- Communication



"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

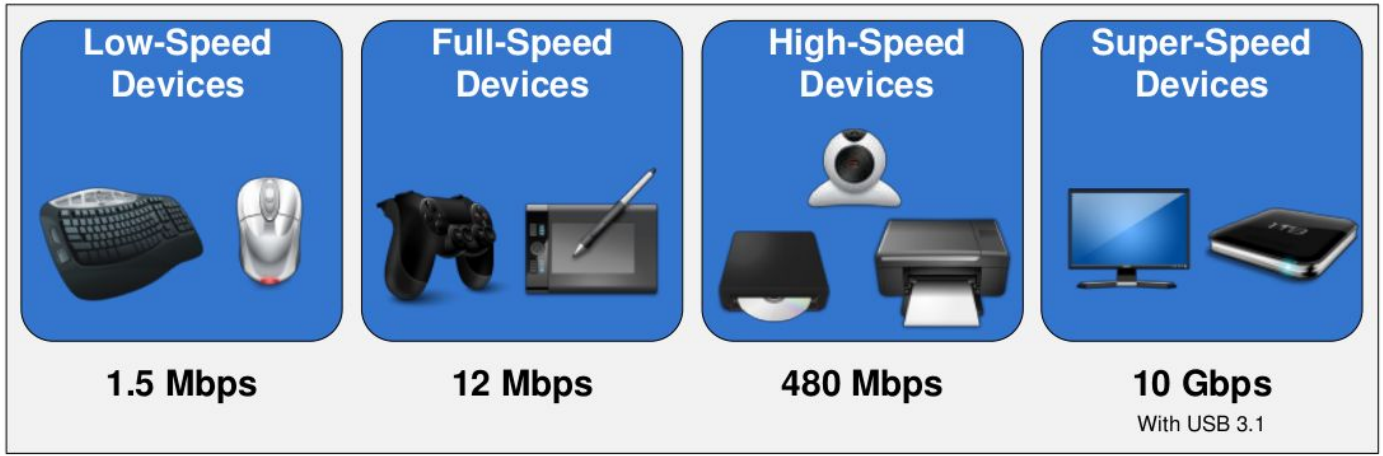
USB Communication States

Bus State	Indication
Differential 1	D+ High, D- Low
Differential 0	D+ Low, D- High
Single Ended 0 (SE0)	D+ and D- Low
Single Ended 1 (SE1)	D+ and D- High
J-State: Low-Speed Full-Speed High-Speed	 Differential 0 Differential 1 Differential 1
K-State: Low-Speed Full-Speed High-Speed	 Differential 1 Differential 0 Differential 0
Resume State:	K-State
Start of Packet (SOP)	Data lines switch from idle to K-State.
End of Packet (EOP)	SE0 for 2 bit time followed by J-State for 1 bit time.

"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

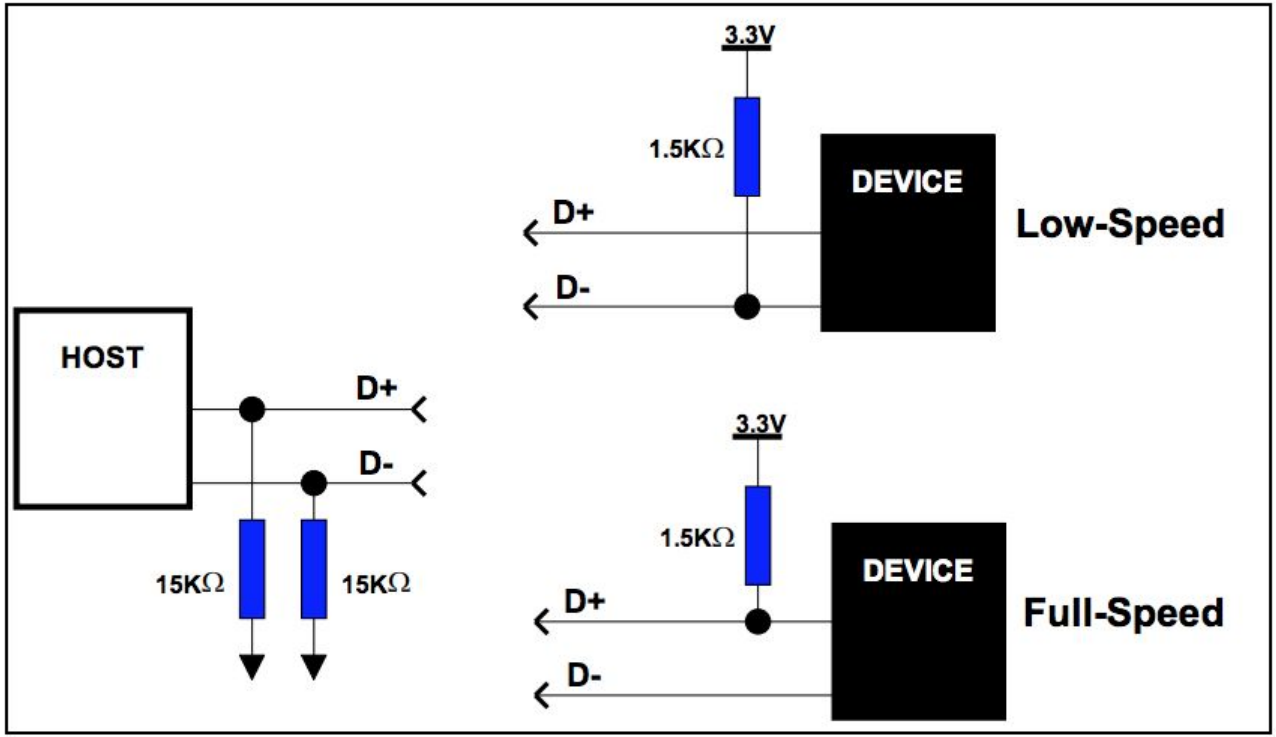
Demo: Sniffing USB with a Logic Analyzer

USB Transfer Speeds



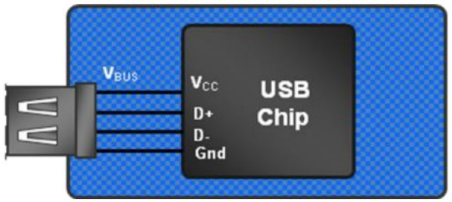
"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

USB Speed Detection

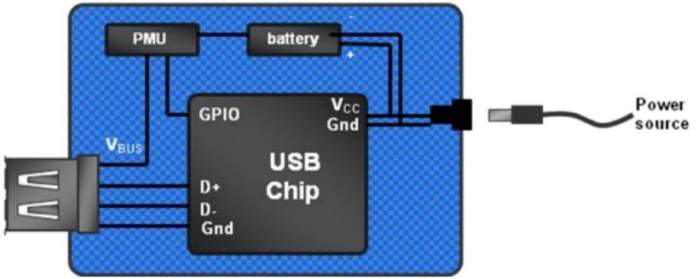


"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

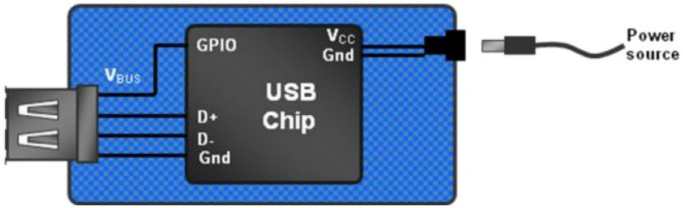
USB Power



bus-powered



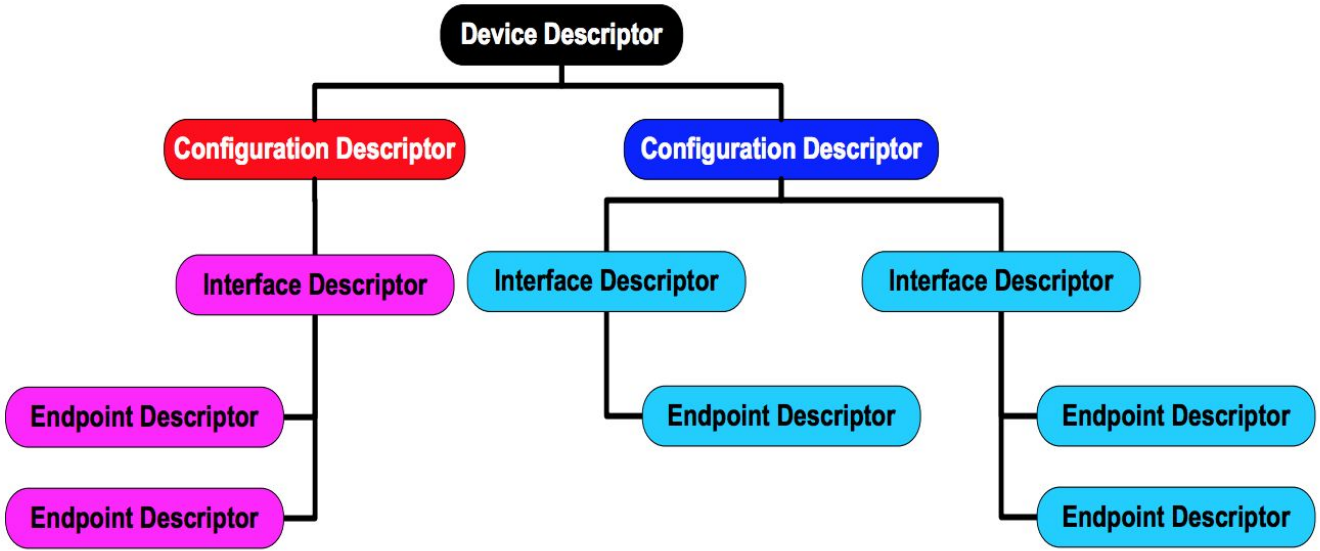
hybrid powered



self-powered

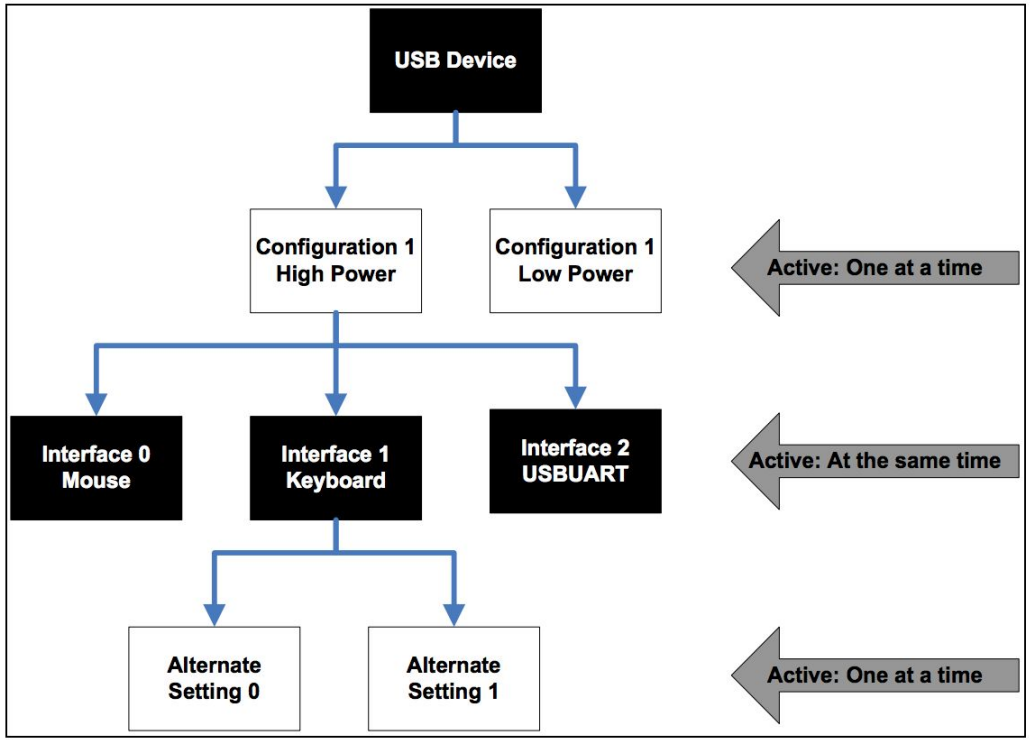
"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

USB Device Descriptor



"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

USB Device Descriptor: Example



"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

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 Class Codes

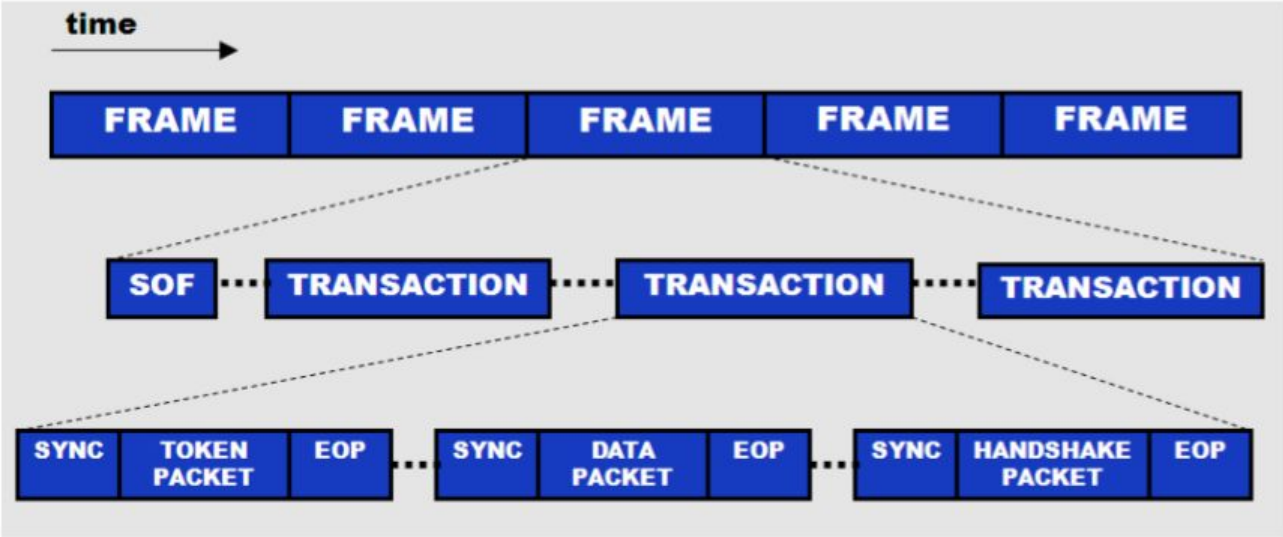
Class	Usage	Description	Examples
00h	Device	Unspecified	Device class is unspecified, interface descriptors are used to determine needed drivers
01h	Interface	Audio	Speaker, microphone, sound card, MIDI
02h	Both	Communications and CDC Control	Modem, ethernet adapter, Wi-Fi adapter
03h	Interface	Human Interface Device (HID)	Keyboard, mouse, joystick
05h	Interface	Physical Interface Device (PID)	Force feedback joystick
06h	Interface	Image	Camera, scanner
07h	Interface	Printer	Printers, CNC machine
08h	Interface	Mass Storage	External hard drives, flash drives, memory cards
09h	Device	USB Hub	USB hubs
0Ah	Interface	CDC-Data	Used in conjunction with class 02h.
0Bh	Interface	Smart Card	USB smart card reader
0Dh	Interface	Content Security	Fingerprint reader

and even more ...

"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

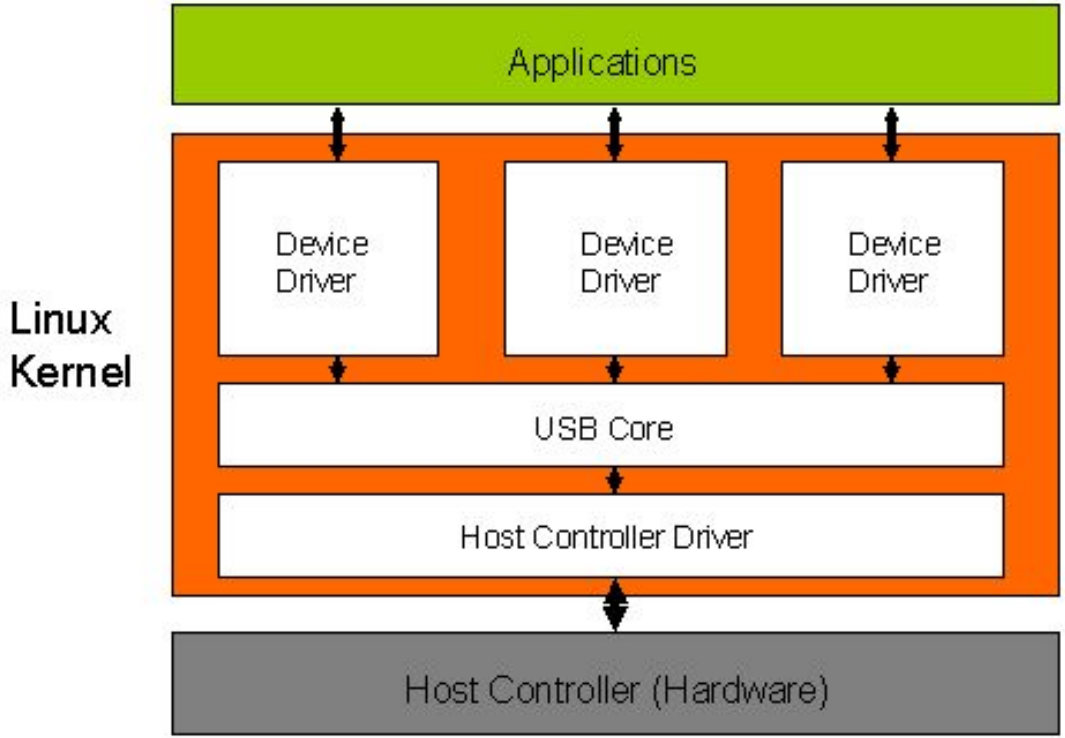
Demo: lsusb and syslog

USB Communication



"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

USB host



http://www.linux-usb.org/USBMon/dissertation/USB-dissertation.htm

USB enumeration (simplified)

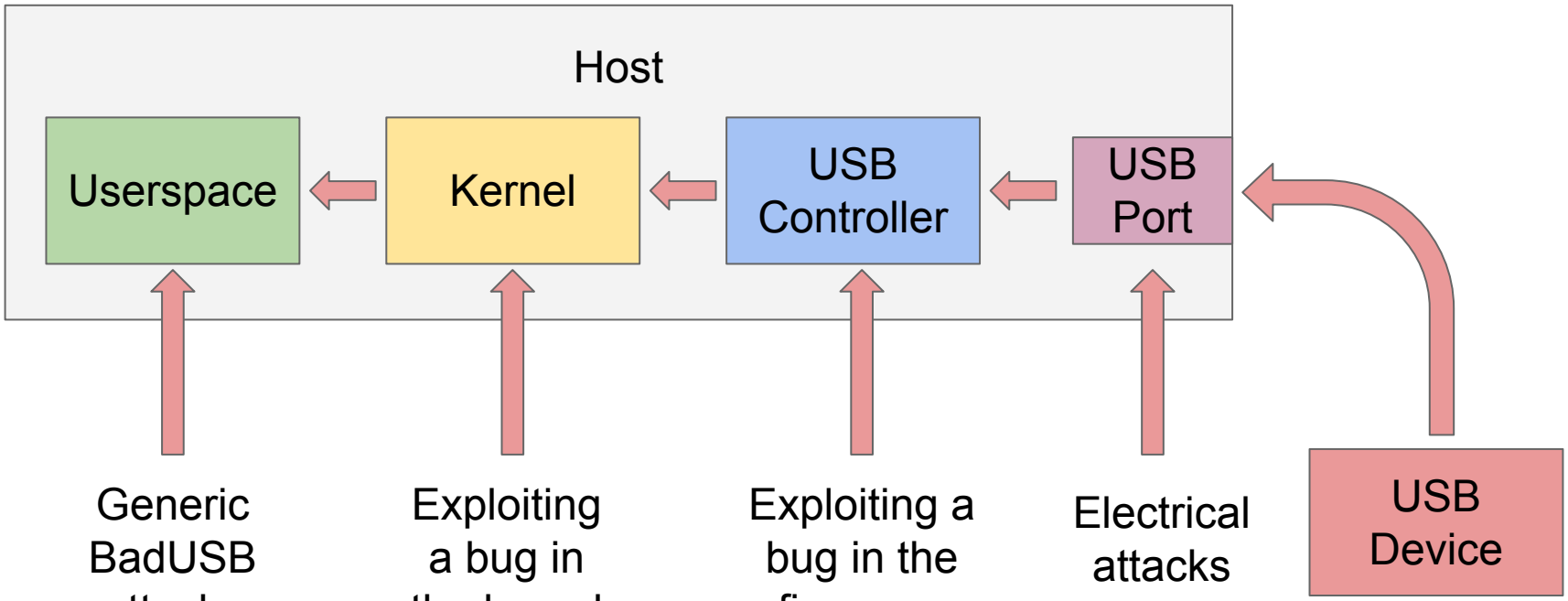
1. The device is plugged into a USB port
2. The host requests device descriptors
3. The host loads the appropriate driver
4. The host sets a specific device configuration
5. Done

Demo: Sniffing USB with usbmon and Wireshark

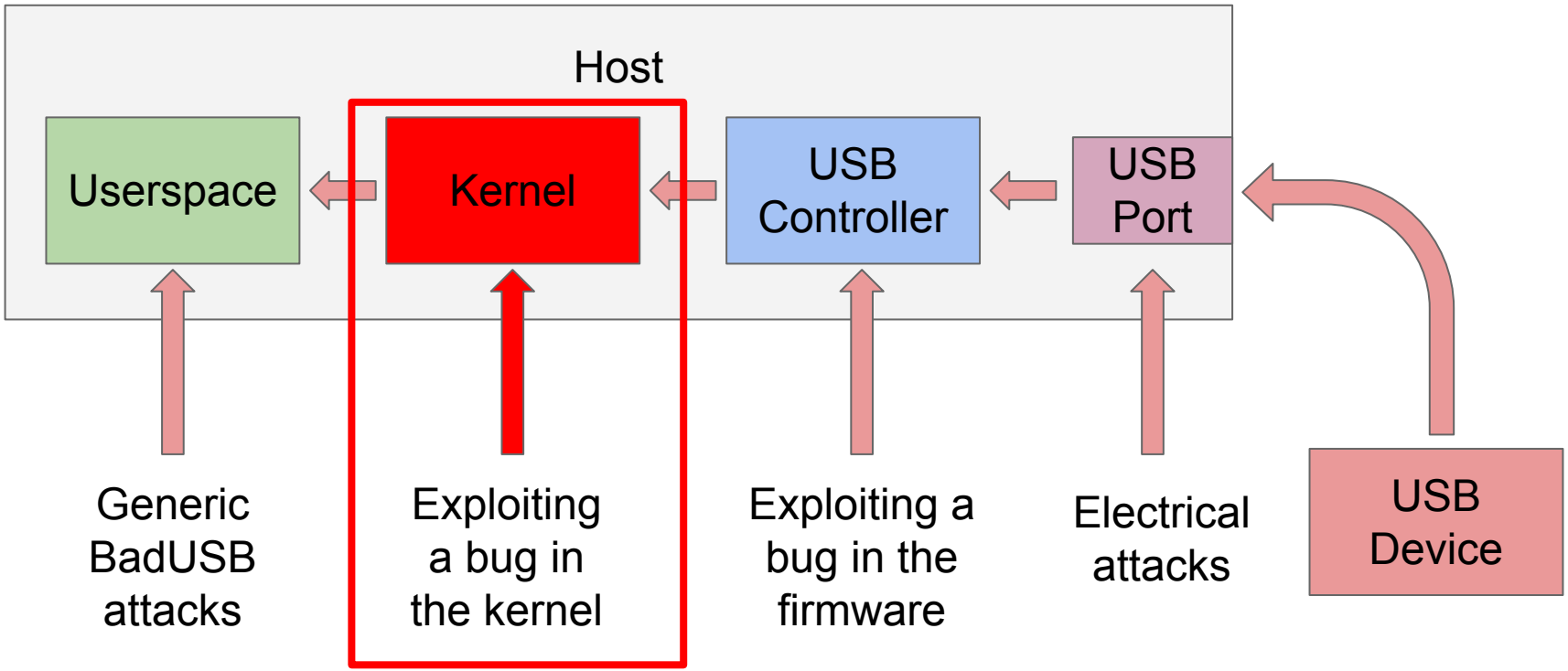
Demo: Turning off LED on a Logitech Web Camera

Part 2: USB Attack Surface

Attack Surface: Device => Host



Fuzzing Target



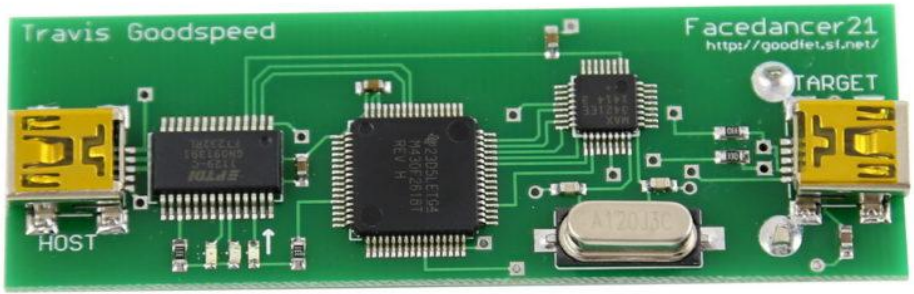
Part 3: Consumer Ready BadUSB
Part 4: Microcontroller Based BadUSB

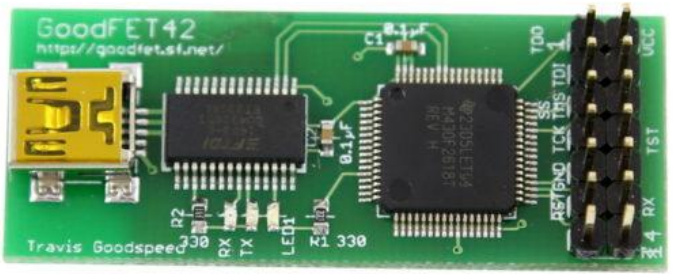
Part 5: Facedancer

Facedancer

- “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”
- Hardware:
  - Old: Facedancer21, GoodFET42
  - New: GreatFET One, NXP LPC4330-Xplorer Board
- Software:
  - Old: https://github.com/travisgoodspeed/goodfet
  - New: https://github.com/usb-tools/Facedancer

FaceDancer21 and GoodFET42

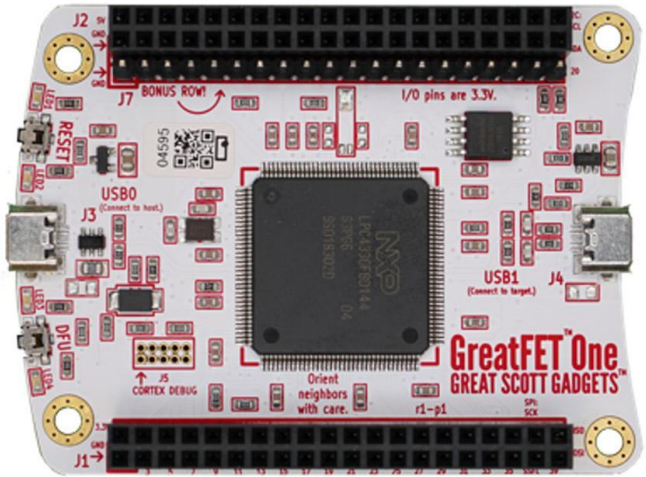




https://hackerwarehouse.com/product/facedancer21/

https://hackerwarehouse.com/product/goodfet42/

GreatFET One and NXP LPC4330-Xplorer



https://shop.hak5.org/products/greatfet

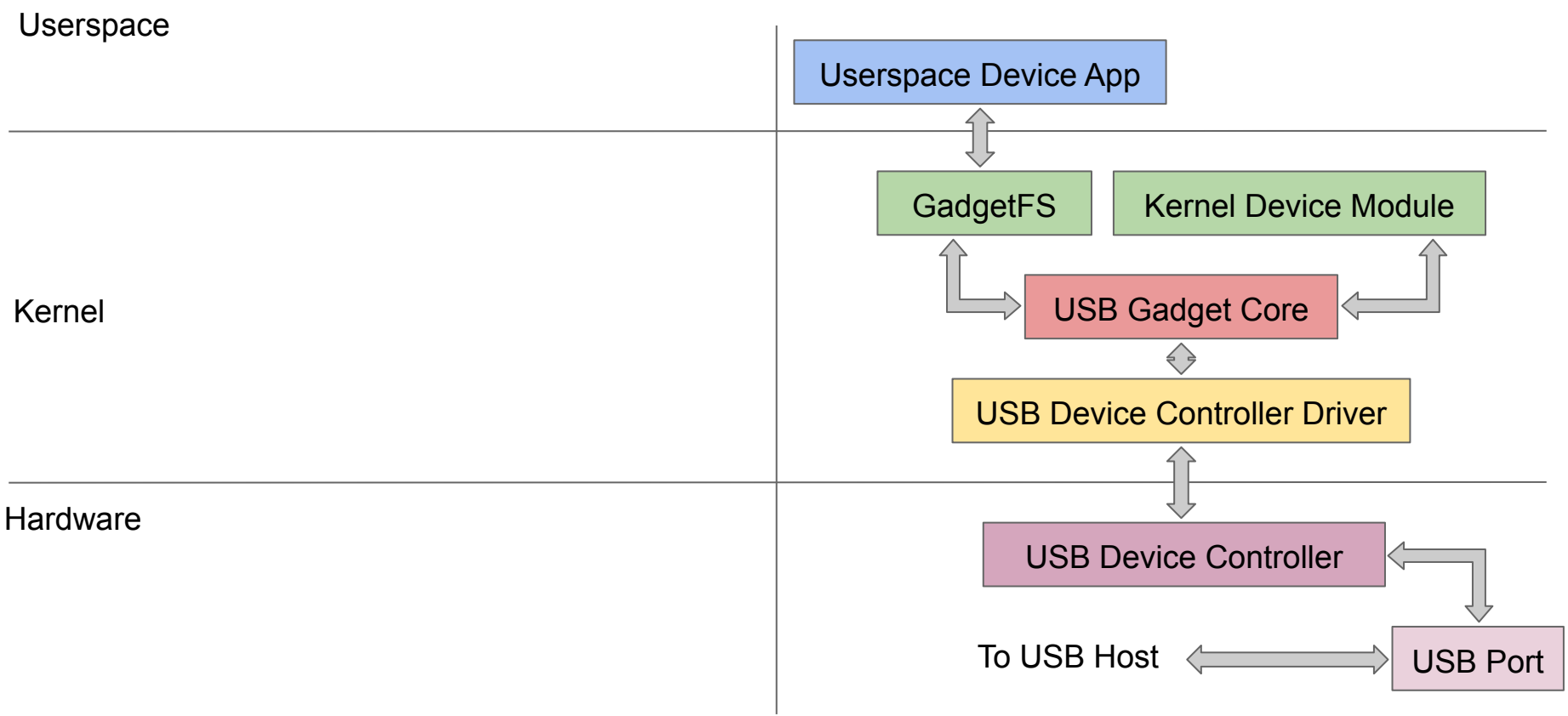
https://www.nxp.com/support/developer-resources/nxp-designs/lpc4330-xplorer-board:OM13027

Part 6: Linux Gadget Subsystem

Linux USB Subsystem: Gadget

- Allows to turn a Linux device into a USB device
- Requires a USB Device Controller and a driver for it
- Linux provides a few interfaces for the Gadget Subsystem
  - GadgetFS - a userspace interface, essentially allows to implement USB device logic as a userspace app
  - ConfigFS/FunctionFS and legacy modules (g_hid.ko, ...) - implement USB device logic as kernel modules, but are controlled from userspace
  - Gadget Subsystem API - allows to implement a USB device as a custom kernel module

Linux USB Subsystem: Gadget



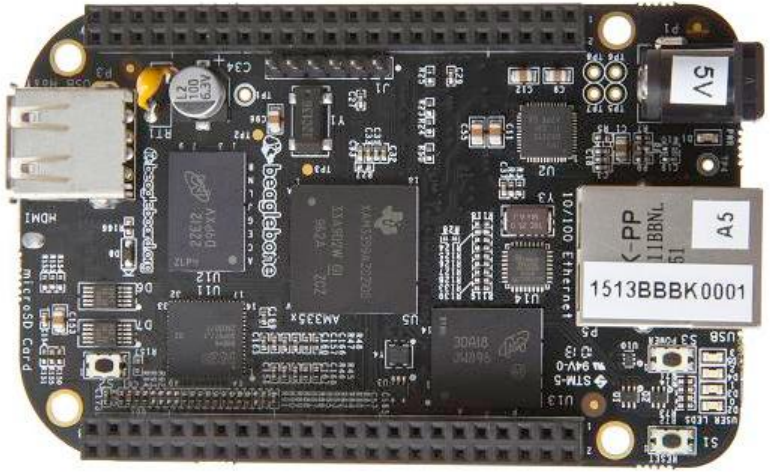
Linux Gadget Hardware

- Almost any Linux-based single-board-computer with a USB Device Controller, USB OTG port and proper driver support

- Common options:
  - Beagle boards (BeagleBone Black)
  - Odroid boards (ODROID-XU3, ODROID-C2)
  - Raspberry Pi Zero boards (v1.3, W)

BeagleBone Black

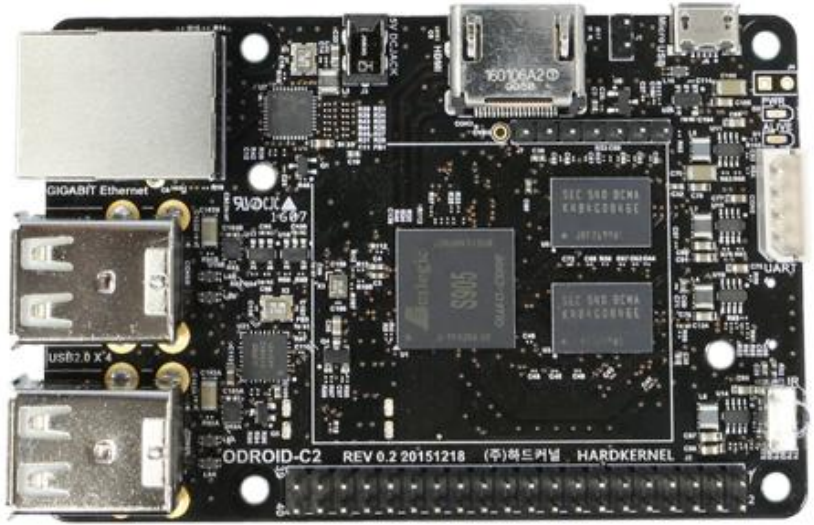
- 50$, Open Hardware, 1 GHz single-core CPU, 512 MB RAM
- Mini USB OTG port (supports Linux Gadget Subsystem)



https://beagleboard.org/static/ti/product_detail_black_lg.jpg

ODROID-C2

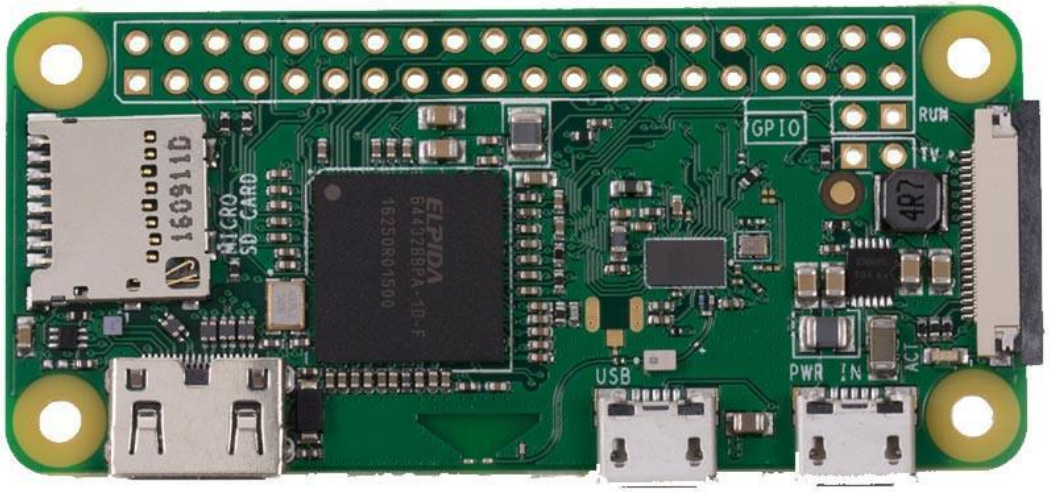
- ODROID-XU3 (discontinued, replaced by ODROID-XU4 without OTG)
- ODROID-C2 (50$)



https://www.hardkernel.com/main/_Files/prdt/2016/201602/C2BoardRevision2.jpg

Raspberry Pi Zero v1.3

- A 5$ ARM based single-board-computer



https://thepihut.com/products/raspberry-pi-zero-w

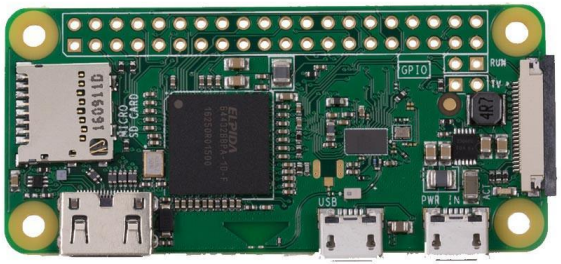
Raspberry Pi Zero

- 1 GHz single-core CPU, 512 MB RAM
- Micro USB OTG port (supports Linux Gadget Subsystem)
- P4wnP1: USB attack platform for RPi Zero

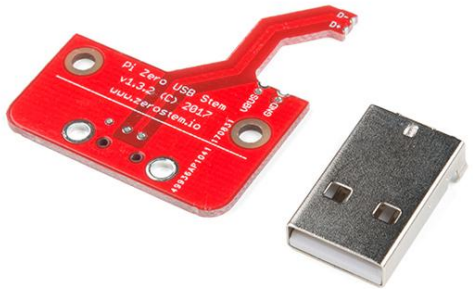
Raspberry Pi Zero W (10$):

- 802.11 b/g/n wireless LAN
- Bluetooth 4.1, Low Energy (BLE)

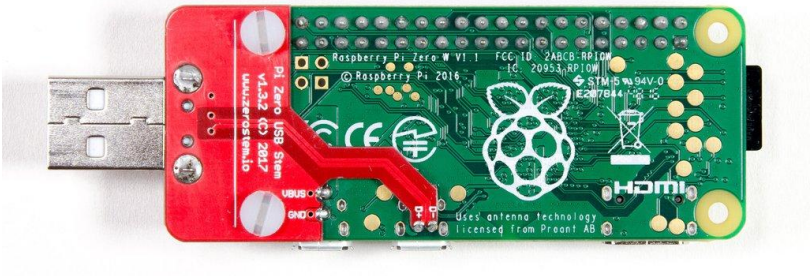
Raspberry Pi Zero W + Zero Stem



+



=



https://thepihut.com/products/raspberry-pi-zero-w

https://www.sparkfun.com/products/14526

https://shop.pimoroni.com/products/zero-stem-usb-otg-connector

Part 7: USB Fuzzing

USB Fuzzing Approaches

- Emulate USB devices via hardware
  - Plug in Facedancer into a USB host and use umap or umap2

- Emulate USB devices through a hypervisor
  - vUSBf - fuzzes the guest kernel running in QEMU by connecting USB devices via usbredir protocol

- Use syzkaller (fuzzing in VMs, reproducing with hardware)

Syzkaller

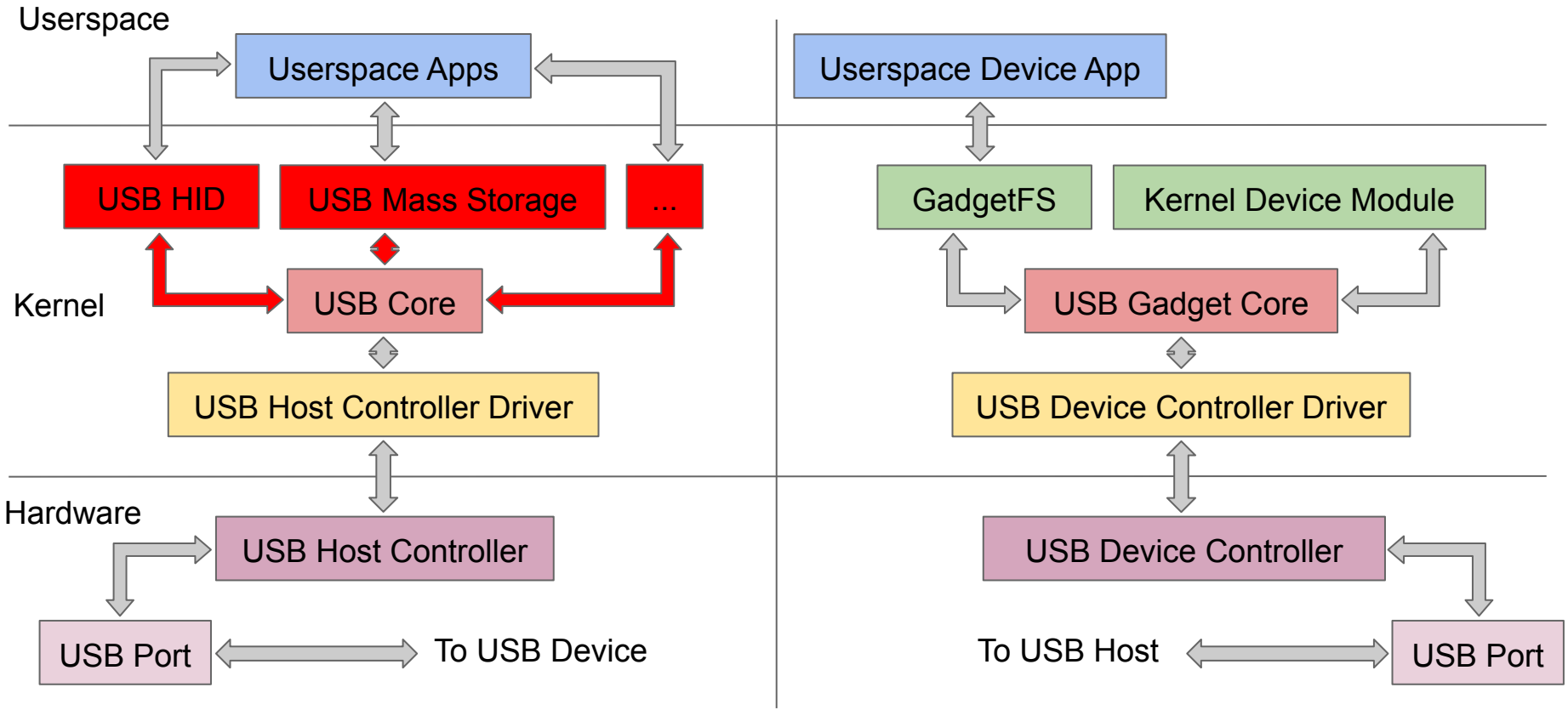
- Coverage-guided grammar-based kernel fuzzer
- Unsupervised
- Multi-
  - OS (Linux, *BSD, Fuchsia, …)
  - arch (x86-64, arm64, …)
  - machine (QEMU, GCE, Android phones, ...)
- Generates C reproducers for found bugs

Syzkaller

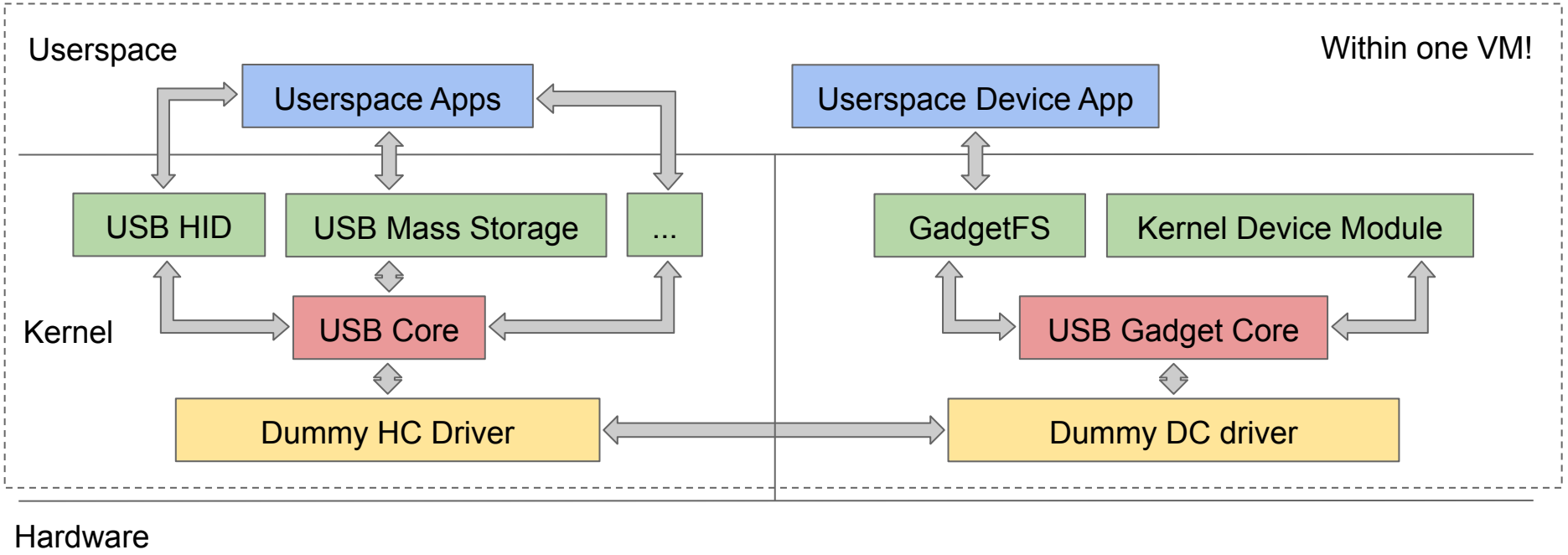
- As of now found over 2500 bugs (syzkaller/wiki/Found-Bugs + syzbot + internal)
- Numerous CVEs
- At least 4 public local privilege escalation exploits (CVE-2017-7308, CVE-2017-6074, CVE-2017-2636, CVE-2017-1000112)

syzkaller.appspot.com

Linux USB Subsystem

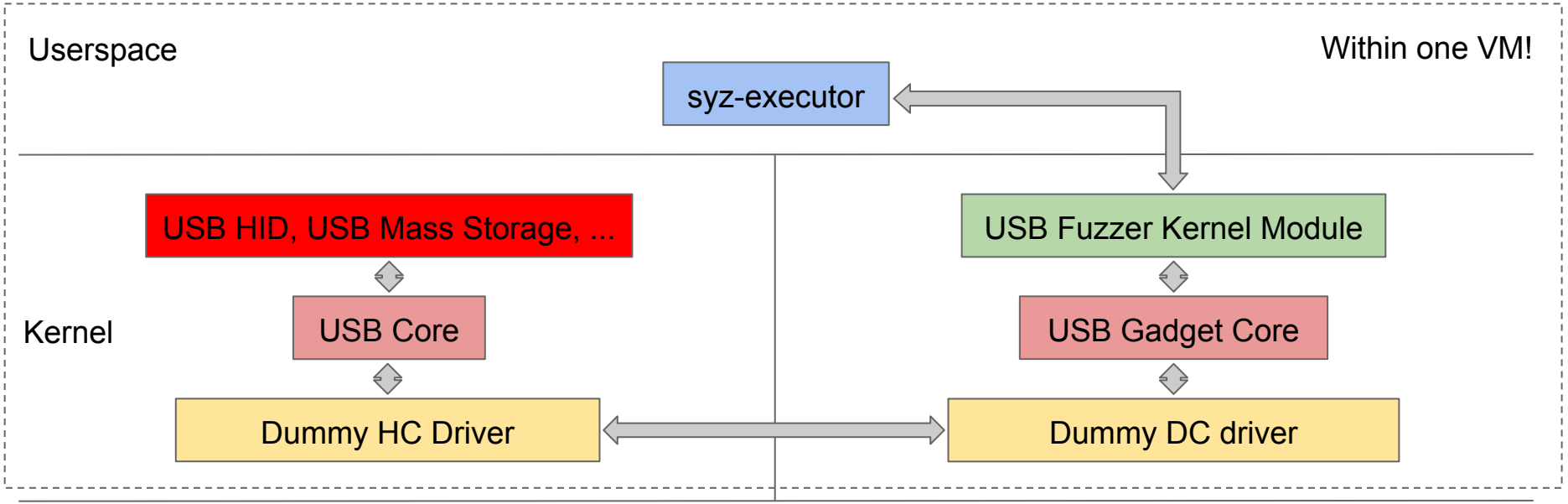


CONFIG_USB_DUMMY_HCD



No hardware (or hypervisors) required!

Syzkaller USB Fuzzing Approach

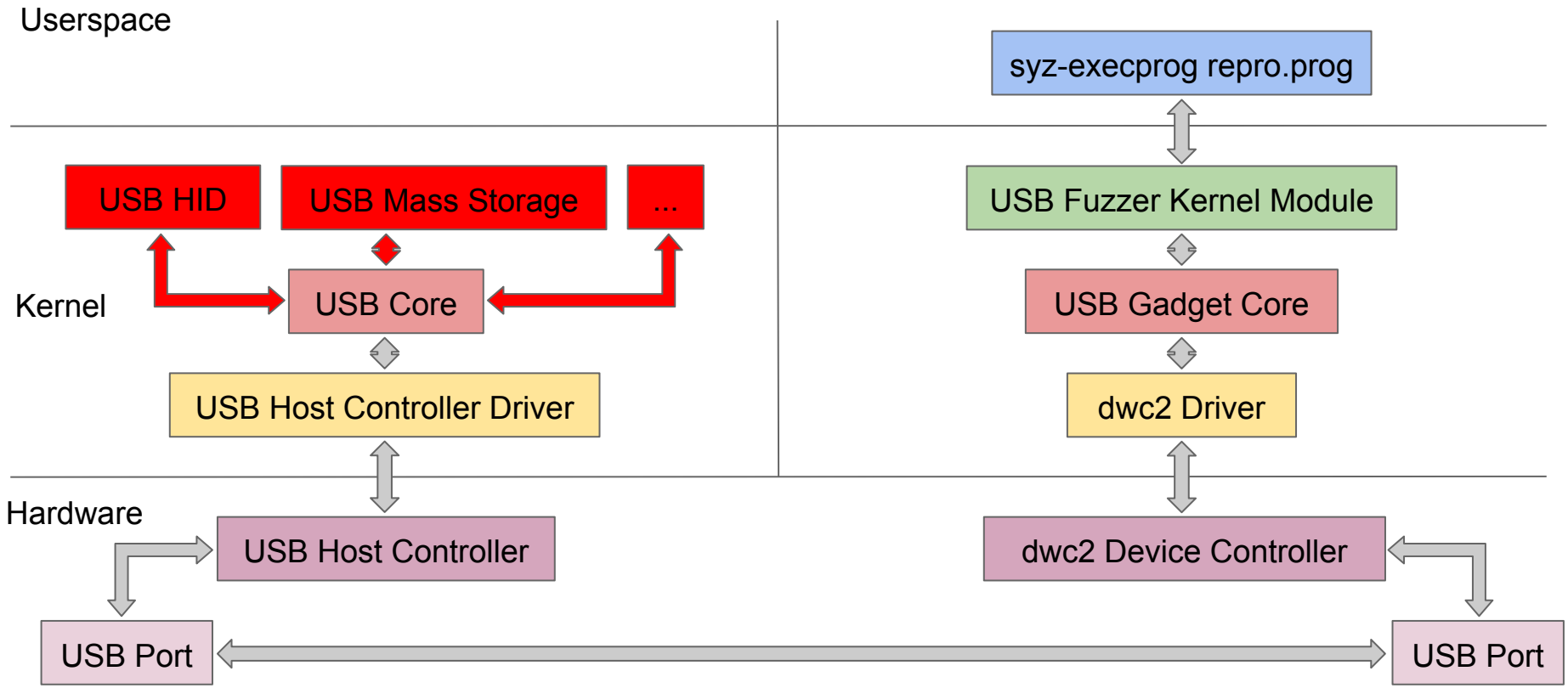


No hardware (or hypervisors) required!

Linux Kernel USB Fuzzing Results

- 80+ bugs in the USB subsystem manually reported
  - 5 bugs in USB core subsystem
  - 23 CVEs (for the bugs that got fixed)
  - 12 bugs in USB Gadget Subsystem (mostly GadgetFS)
- 60+ more reported automatically by syzbot since integration

Running Reproducers via Raspberry Pi Zero



Demo: Crashing Linux Over USB

Demo: Crashing Windows Over USB

Part 8: USB Sniffing

Thanks!
Questions?

https://github.com/xairy/hardware-village/tree/master/usb

Andrey Konovalov <andreyknvl@gmail.com>