Virtualised USB Fuzzing

Breaking USB for Fun and Profit

Tobias Mueller

School of Computing
Dublin City University

4. September 2010

About me

Tobi(as) Mueller

Mail muellet2@computing.dcu.ie

974C F452 FDA0 99D8 CB7E

54F7 DC03 BAA3 D349 2A2A

- Talk $\sim$ 30 mins
- Ask immediately
- Demo afterwards

Motivation

What's the problem?

- USB drivers in kernel space
- USB supported by every major OS
- USB widely deployed
- Not easy to assess security
  - Development board?
  - Inject messages into kernel?

Digital Voting Pen

Yes, it uses USB. hehe

In-Flight entertainment

Based on Linux or VxWorks

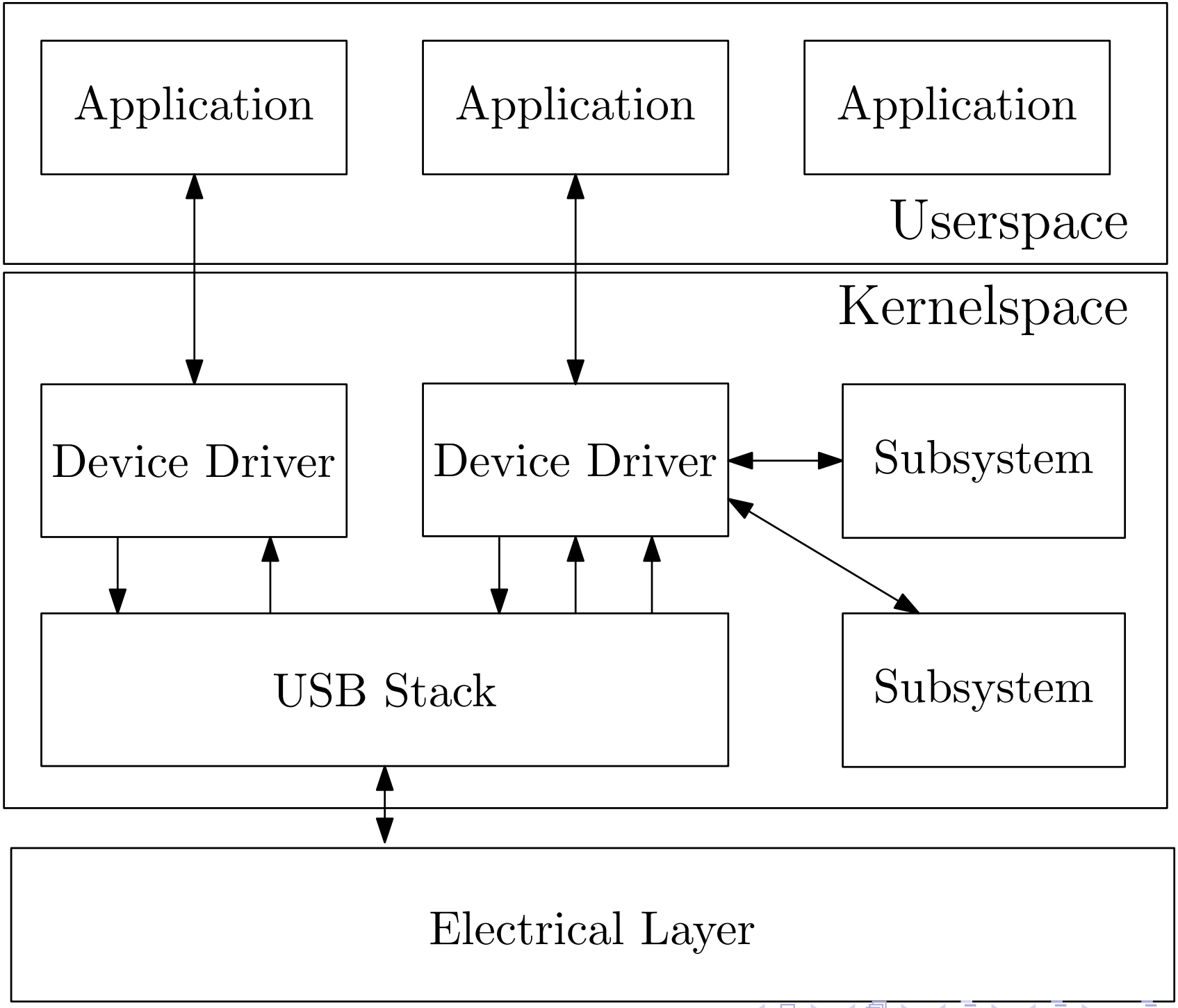
USB

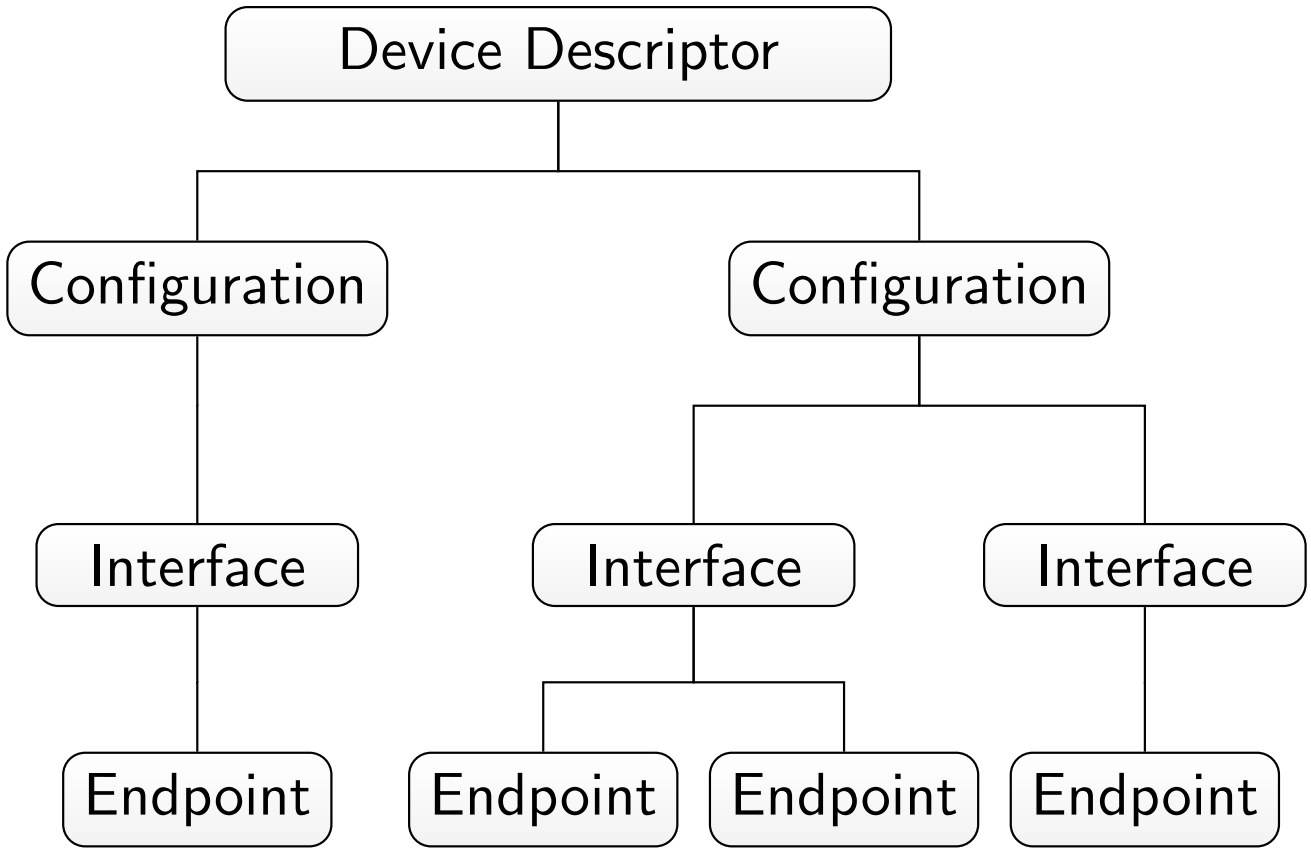
- <1990: IO and IRQs
- >1990: USB *yay*
- cheap to build
- hotplug
- auto config
- <5 metres
- device $\not\rightarrow$ device

Architecture

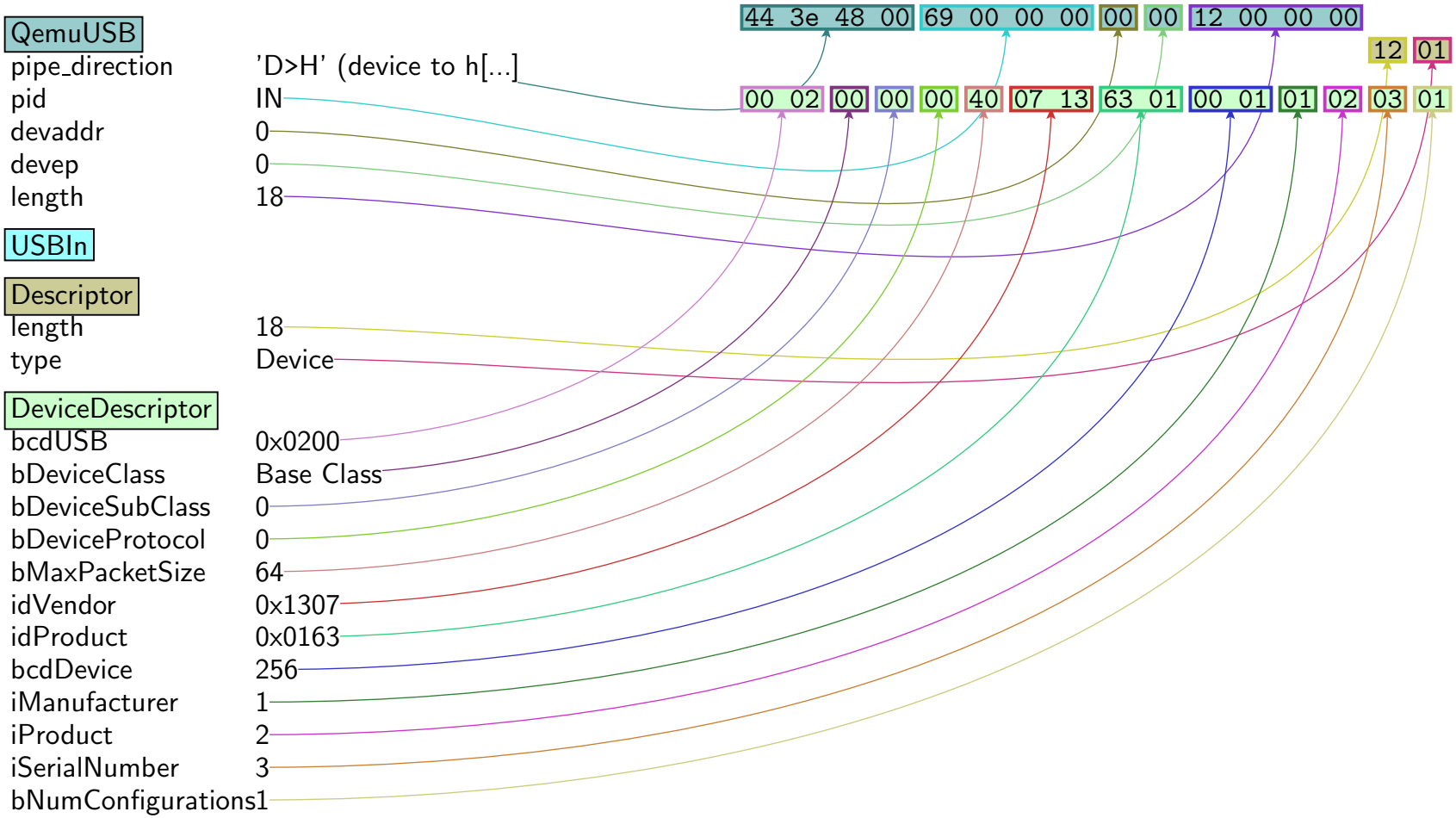
- USB/IP
- Wireless USB

- Host initiated communication
- packet-based
  - SETUP
  - IN
  - OUT
- interrupt, bulk, isochronous, control
- descriptors → Driver

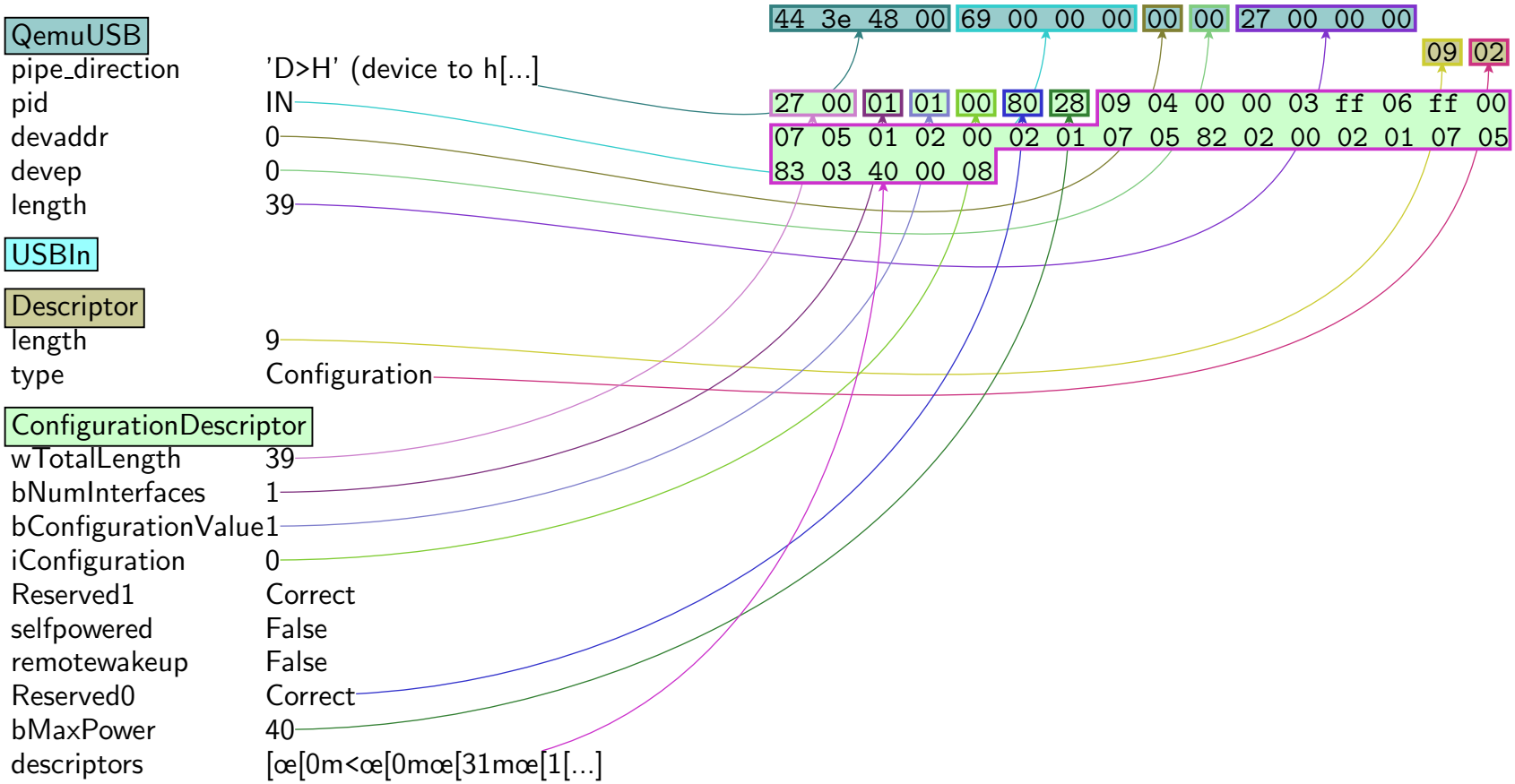

Application
Application
Application
Userspace
Kernelspace
Device Driver
Device Driver
Subsystem
USB Stack
Subsystem
Electrical Layer


Device Descriptor
Configuration
Configuration
Interface
Interface
Interface
Endpoint
Endpoint
Endpoint
Endpoint

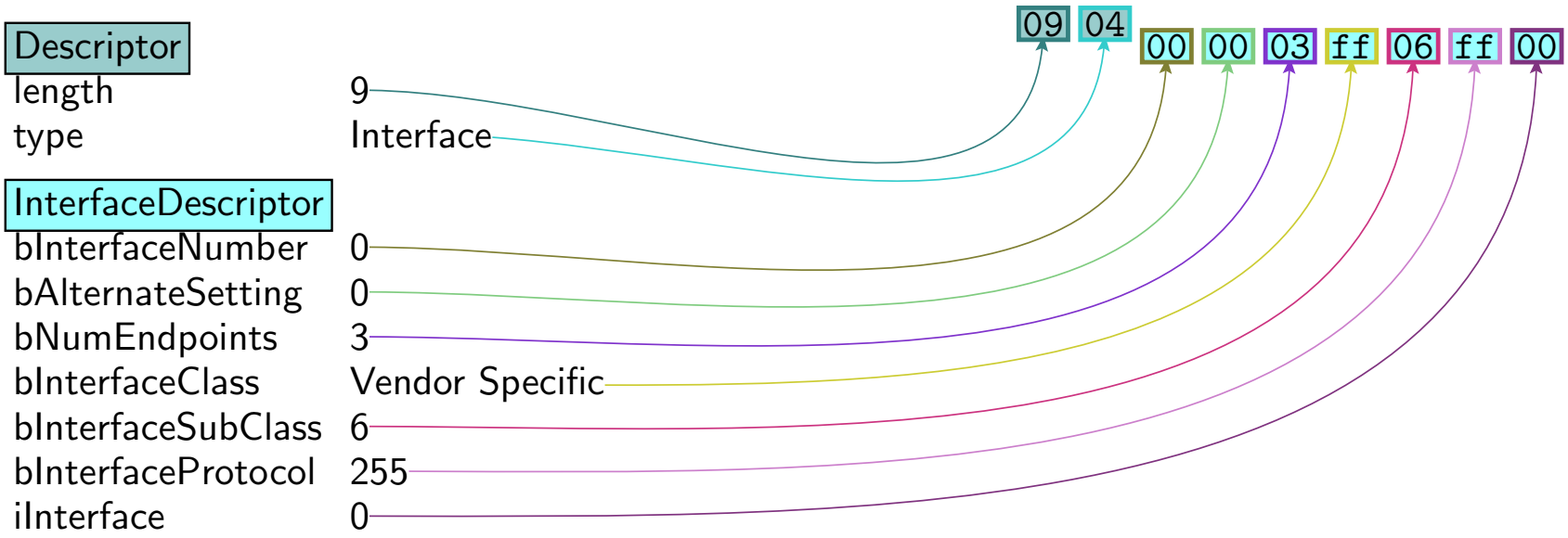
Device Descriptor



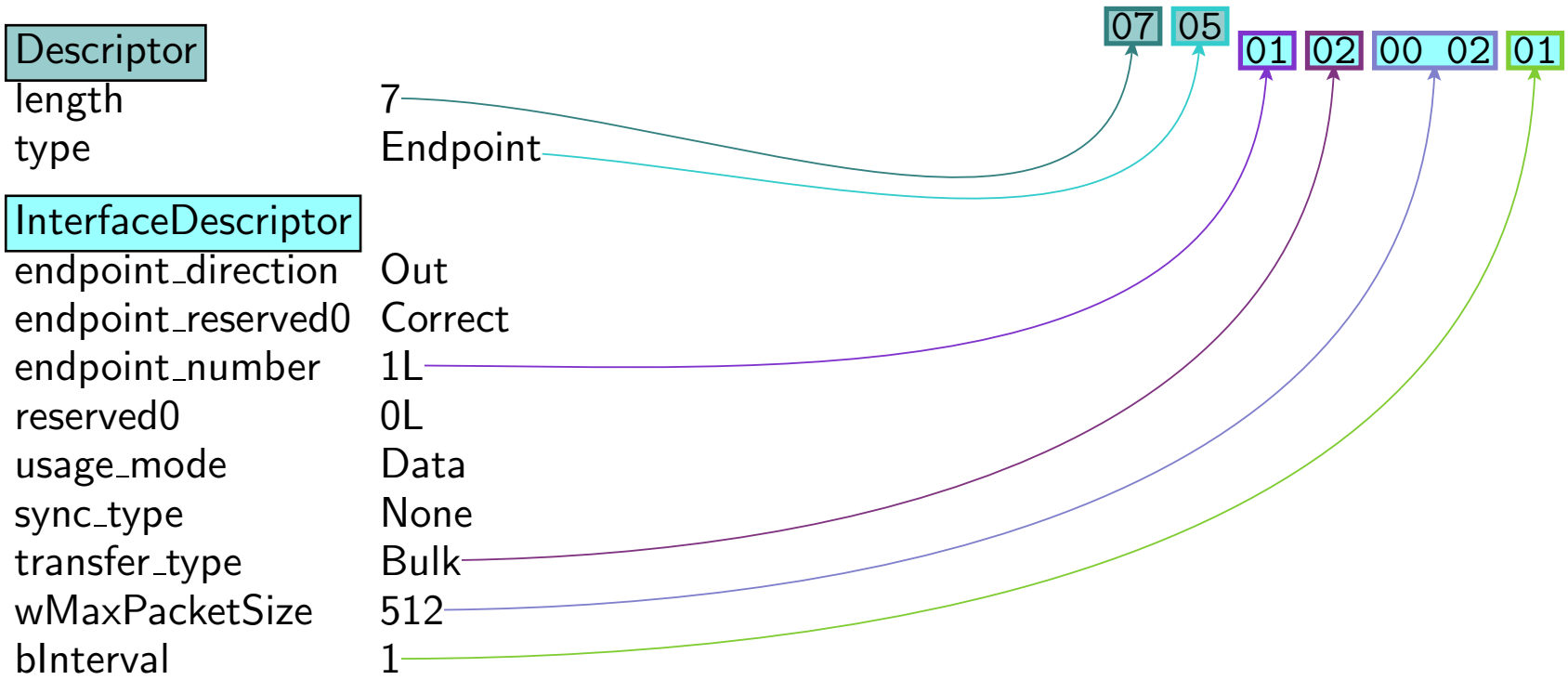
Configuration Descriptor



Interface Descriptor



Endpoint Descriptor



Fuzzing

- Dumb Fuzzing
  - coined in late 80's
  - feed program with random(?) data
  - got a lot attention in 2004
- Smart Fuzzing
  - Patent encumbered?
  - Modify existing valid structured data
  - Checksums
  - Cover more code
- Scapy
  - Awesome (!) framework
  - sniff, manipulate, craft, send (Ethernet) packets
  - models packets in Python

USB Stack stress testing

How many devices can you handle?

```
class USBInDeviceDescriptor(Packet):
    name = 'DeviceDescriptor'

    fields_desc = [
        LEShortField('bcdUSB', 0x0200),
        ByteEnumField('bDeviceClass', 0, CLASS_ENUM
        ByteEnumField('bDeviceSubClass', 0, SUBCLAS
        ByteEnumField('bDeviceProtocol', 0, PROTOCC
        ByteField('bMaxPacketSize', 64),
        LEXShortEnumField('idVendor', 0xffff, VENDO
        LEXShortField('idProduct', 0x1337),
        LEShortField('bcdDevice', 0x2342),
        ByteField('iManufacturer', 0),
        ByteField('iProduct', 0),
        ByteField('iSerialNumber', 0),
        ByteField('bNumConfigurations', 0),
    ]
```

What the Fuzz?

Goal

Framework to automatically fuzz-test OS's

- Virtualised Guest
- Attach virtual external USB devices
- Generate USB packets in software
- Monitor guest OS

QEMU

- Full virtualisation (not Xen, OpenVZ, UML, etc. . .)
- Free (as in speech) Virtualisation (not VMWare)
- Existing Virtual USB Drivers (not VirtualBox)

- Supports management via QMP
- had to patch in USB and other stuff

Virtual USB Device

- Take simple existing MSD or Serial driver
- Write out / Read in USB packets
- Implement desired behaviour externally
- `cat` and `echo`
- Or enhancing Scapy to read/write from pipes
- $\rightarrow$ Automaton class

Obtaining Valid USB communication

- Read specs :-(
- `mount none -t debugfs /sys/kernel/debug`
  `mount none -t usbmon`
  see Documentation/usb/usbmon.txt
  :-(
- Using QEMU: Implement filter to pipe out communication

USB Stack stress testing

How many devices can you handle?

```
def run_simple_test(qemu, timeout=4, delete=False):
    qemu.usb_add('mouse')
    time.sleep(timeout)
    cmd = list('dmesg') + ['space'] \
            + ['minus'] + ['c'] + ['enter']
    qemu.sendkeys(cmd)
    usb_devices = qemu.usb_info()
    if delete:
        for device in usb_devices['usbdevices']:
            qemu.usb_del('%d.%d' %
              (device['busnr'], device['devaddr']))


    print qemu.cpu_info()
```

USB Fingerprinting

Targeted attacks

OS	Packet Sequence	Retries	Remarks
Windows	SET, IN, OUT	3	IN length: 64
Linux 2.6.33	SET (9x), RESET	4+2	4 get descriptor then 2 s
OpenBSD 4.7	SET, IN, OUT	7	IN length: 8
FreeBSD 8.0	SET, IN, OUT	6	tries to set address right

Tabelle: USB Stack Fingerprints of various operating systems

Driver Flaws

- Demo: Who's got Linux?

Future Work

What's next?

- USB-3? (SuperSpeed, Device Initiated Communication)
- Making it work with GadgetFS
- Make that work on N900
- Get more fingerprints
- Exploit more drivers
- Run shellcode

Q&A

Questions?

Thanks!

Questions?!