Emulating USB Device Firmware Update

for Quickly Reversing and Exploiting Embedded Systems

Travis Goodspeed

Breakpoint 2012, Melbourne, Australia

Facedancing with Sergey Bratus

Related Work

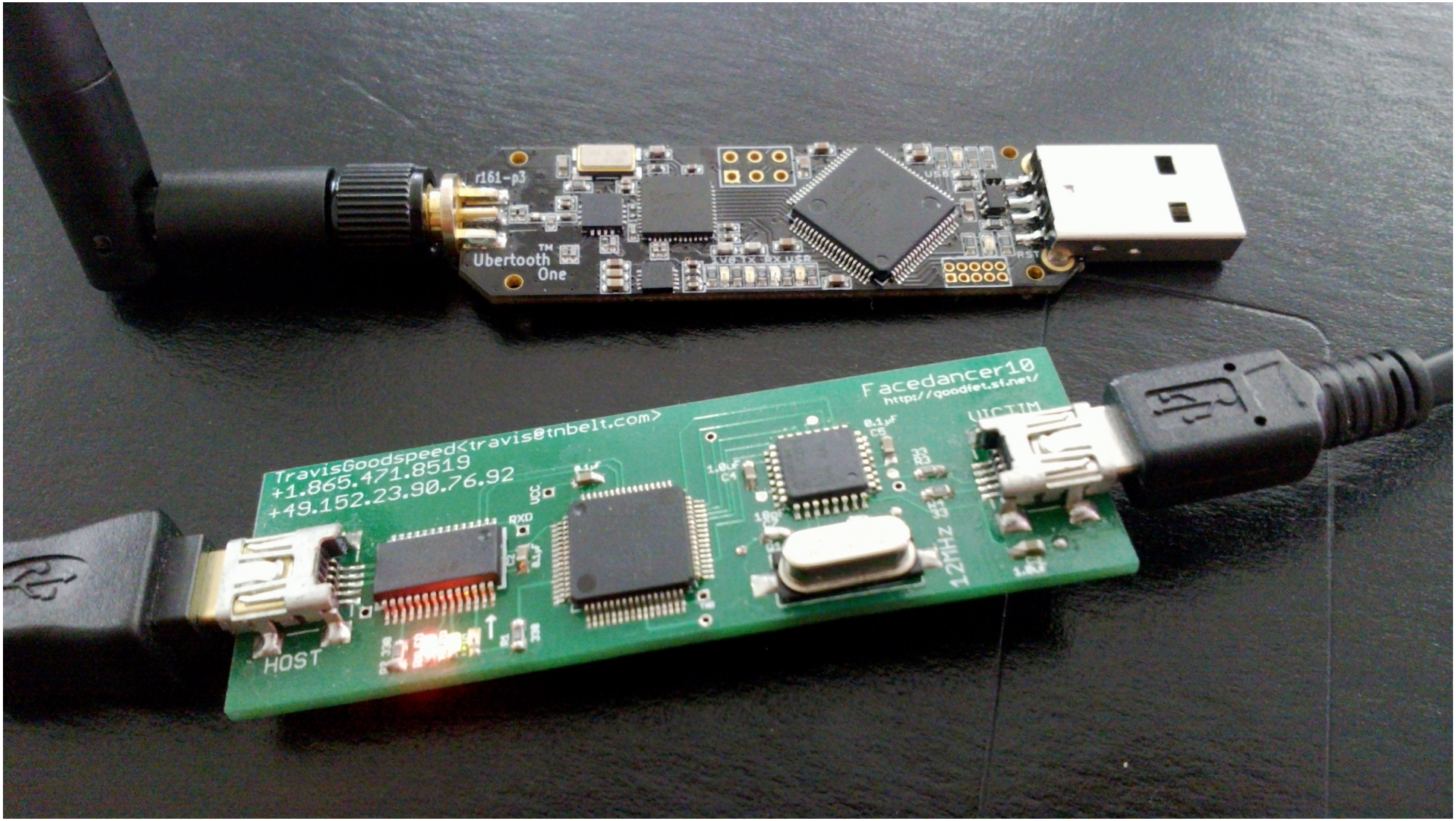
Virtualized hardware (VMWare, Qemu, . . .)

- R. D. Vega, *Linux USB device driver - buffer overflow. MWRI Security Advisory*, CVE-2009-4067, 2009.
- M. Jodeit and M. Johns, *USB device drivers: A stepping stone into your kernel*, European Conference on Computer Network Defense, 2010.

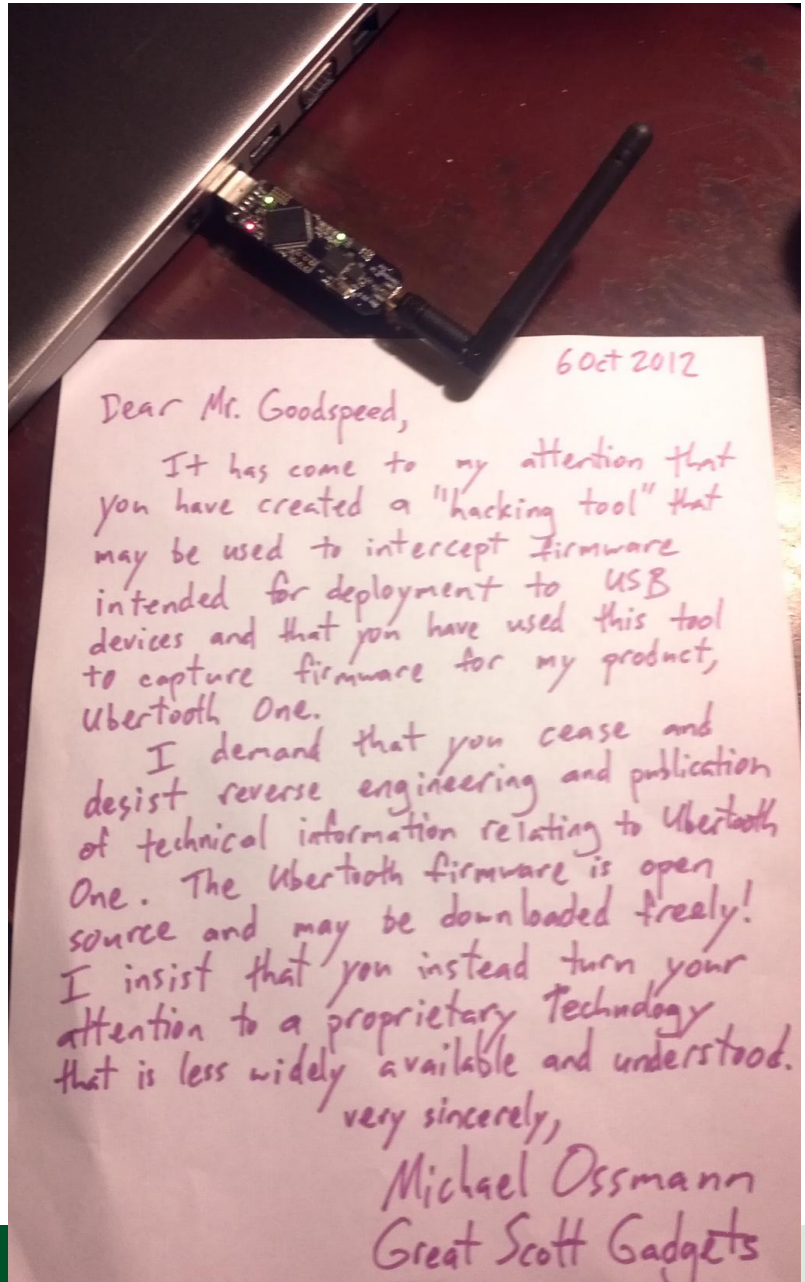
Stand-alone boards & special hardware

- Teensy, `http://www.pjrc.com/teensy`
- A. Davis, *USB - undermining security barriers*, Black Hat Briefings, 2011.
- PSGroove, `https://github.com/psgroove/psgroove`

Facedancing to catch Device Firmware Updates



Legal Threats



Facedancing to catch Device Firmware Updates

Stealing Firmware in 30 Seconds

- USB has a semi-standard way to replace device firmware.
- We can emulate this, pretending to be a device.

Buses are like networks:

- Scannable for vulnerable endpoints.
- Path to vuln set up by packet data.
- Need tools to explore.

Stealing Firmware in 30 Seconds

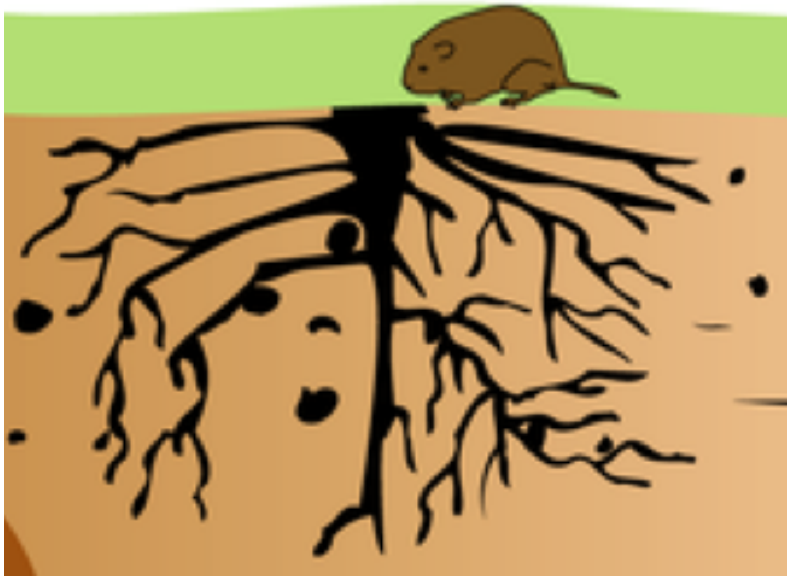
- USB has a semi-standard way to replace device firmware.
- We can emulate this, pretending to be a device.

Buses are like networks:

- Scannable for vulnerable endpoints.
- Path to vuln set up by packet data.
- Need tools to explore.

How many networks?

Through the port, down the rabbit hole

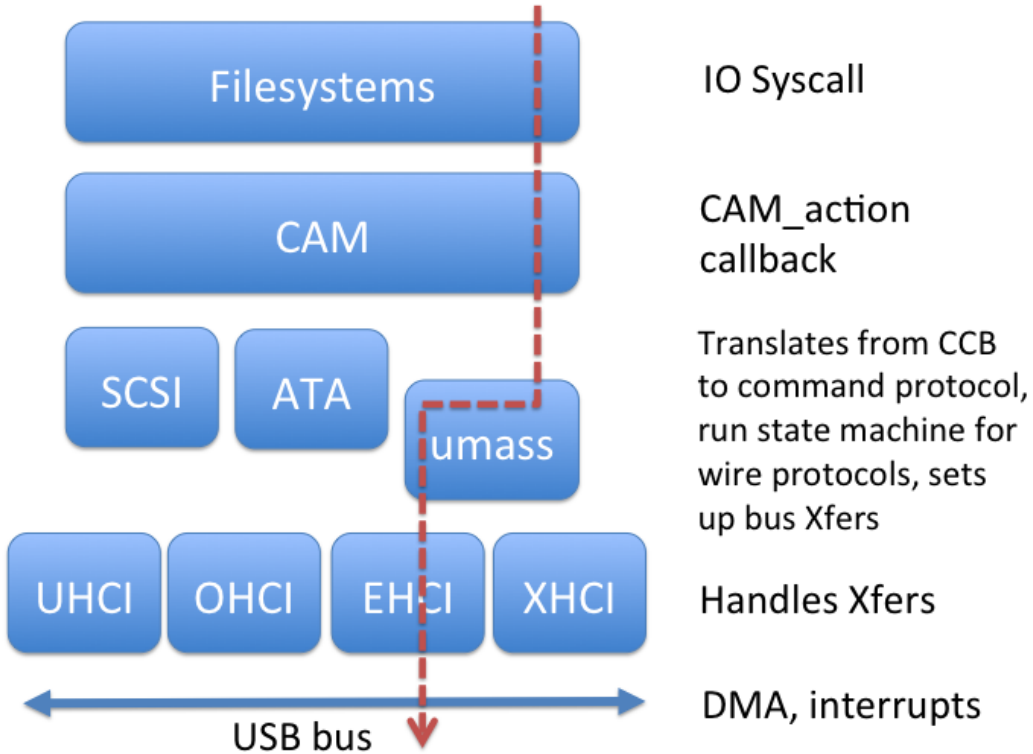


- View from the outside ↑
- View from the inside →

An Attacker's Mapping of Abstractions

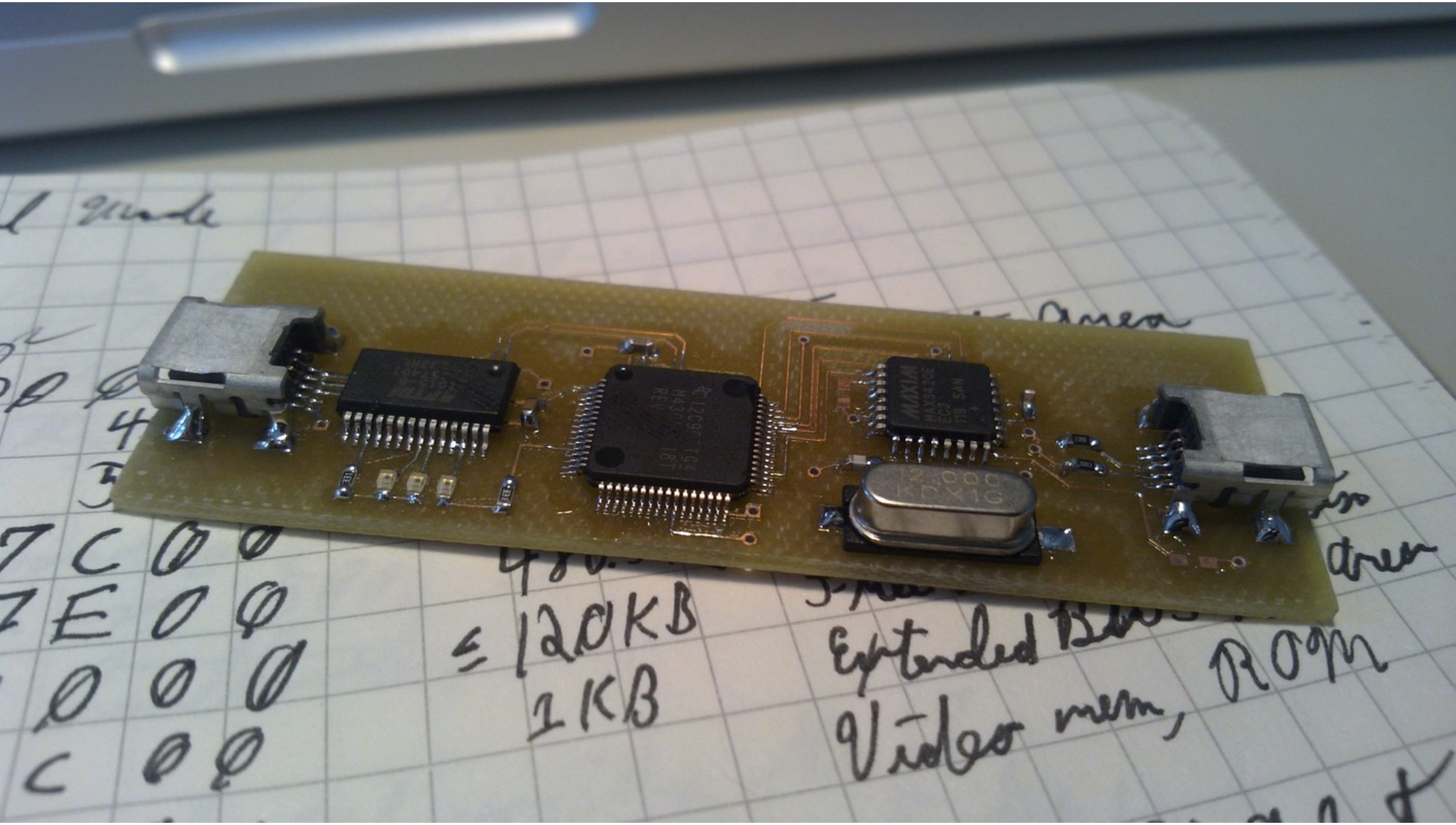
USB	Ethernet	Assumption	Violation	Attack Use
Transfer	One round-trip, maybe NAK-ed	**Intended** device will reply to the transfer	Non-compliant controller	Hijack session, change state under the feet of the host
Transaction	One set of transfers, all but the last NAK-ed	Host controller complies with the USB spec on transactions	Hijack session on disconnect	Use of trusted session context
Packet	Packet Fragment	**Implicit** length of concatenated frames will match **explicit** length of transaction	Non-compliant device	Memory corruption, info leak
Controller	Ethernet Card	—	—	—
Bus	D+/D- Pair	Electrically legal signals, but in reality those **widely outside** of spec are accepted	Non-compliant controller	Damage frames for session hijack, jamming

A Lot Hangs On These Wires

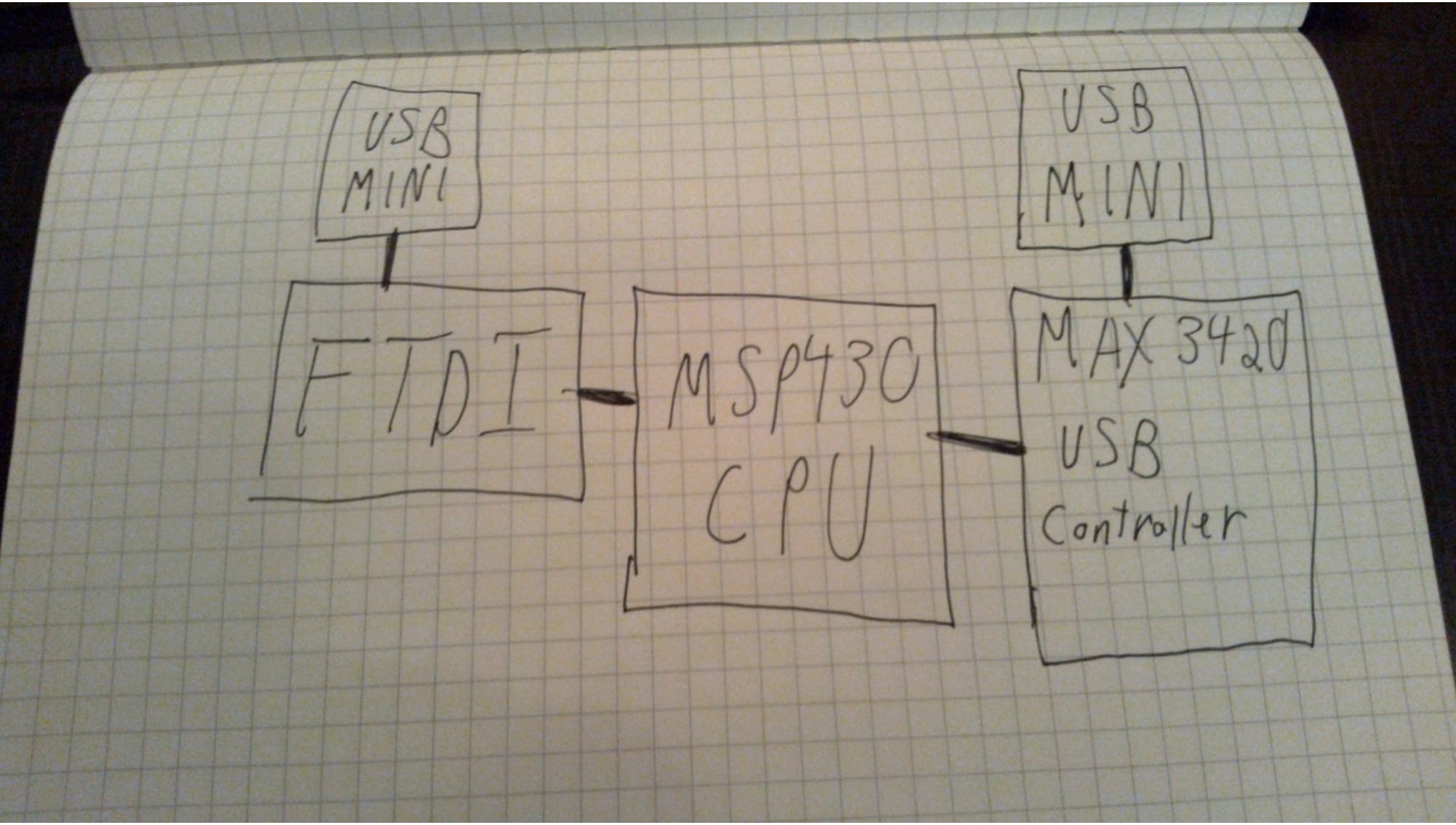


The Dark Side of ~~Socks~~ OS Code

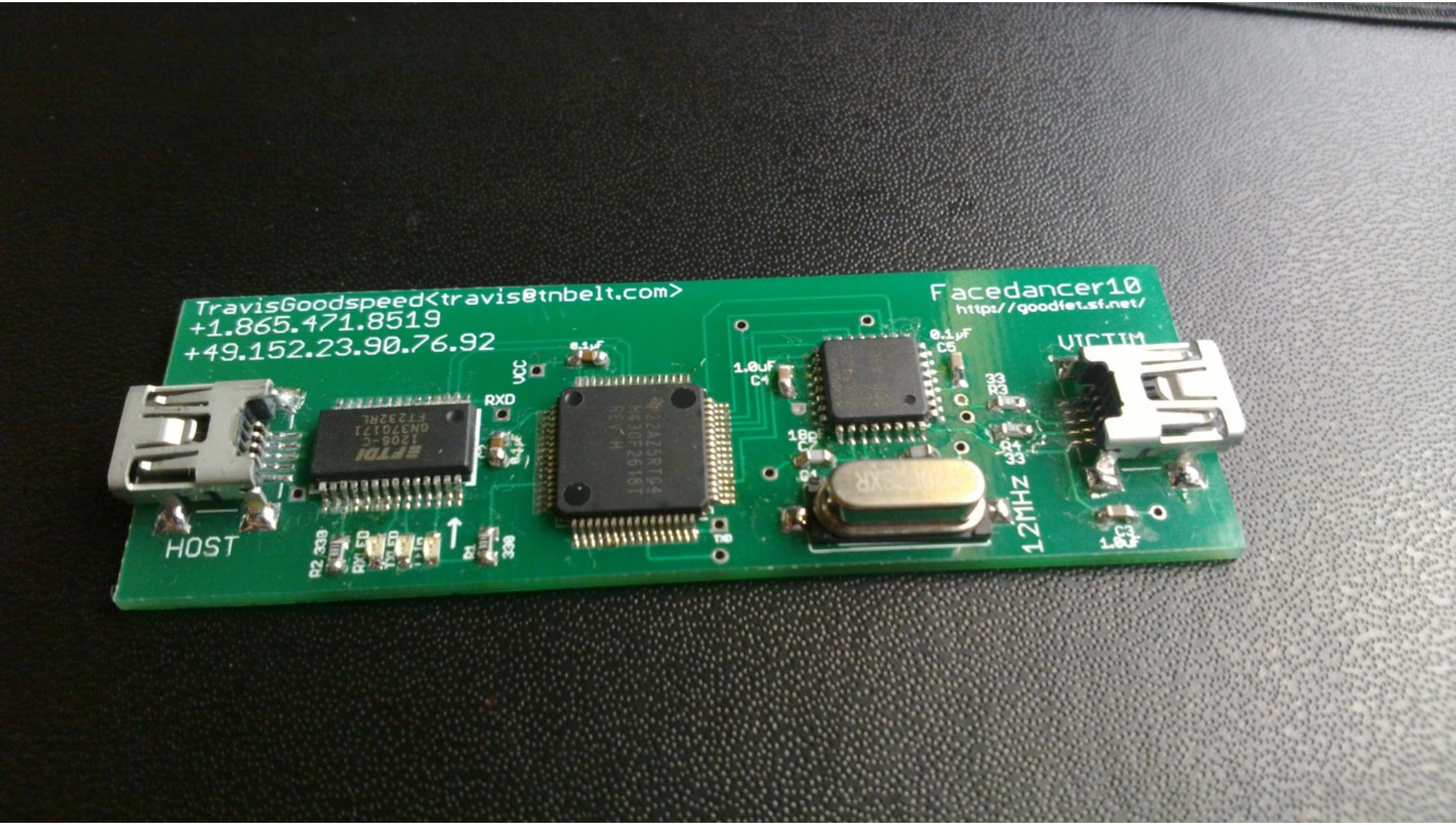
Facedancer Prototype



Facedancer Architecture



Facedancer Board



Building USB Packets, Session

b7	b6	b5	b4	b3	b2	b1	b0
Reg4	Reg3	Reg2	Reg1	Reg0	0	DIR 1=wr 0=rd	**ACKSTAT**

Figure 8. The MAX3420E SPI command byte.

Transfer	F
0	S

Control	ADDR	ENDP	bRequest	wValue	wIndex	Descriptors
GET	0x00	0x0	GET_DESCRIPTOR	DEVICE type	0x0000	DEVICE descriptor

Transaction	F
0	S

SETUP	ADDR	ENDP
0xB4	0x00	0x0

T	D	Tp	R	bRequest	wValue	wIndex	wLength	ACK
0	D->H	S	D	GET_DESCRIPTOR	DEVICE type	0x0000	64	0x4B

Packet	Dir	F
61	-->	S

Sync	SETUP	ADDR	ENDP	CRC5	EOP	Idle
00000001	0xB4	0x00	0x0	0x08	3.00	4

Packet	Dir	F
62	-->	S

Sync	DATA0	Data	CRC16	EOP	Idle
00000001	0xC3	80 06 00 01 00 00 40 00	0xBB29	3.00	4

Packet	Dir	F
63	<--	S

Sync	ACK	EOP	Time
00000001	0x4B	2.80	108.083 µs

Transaction	F
7	S

IN	ADDR	ENDP
0x96	0x00	0x0

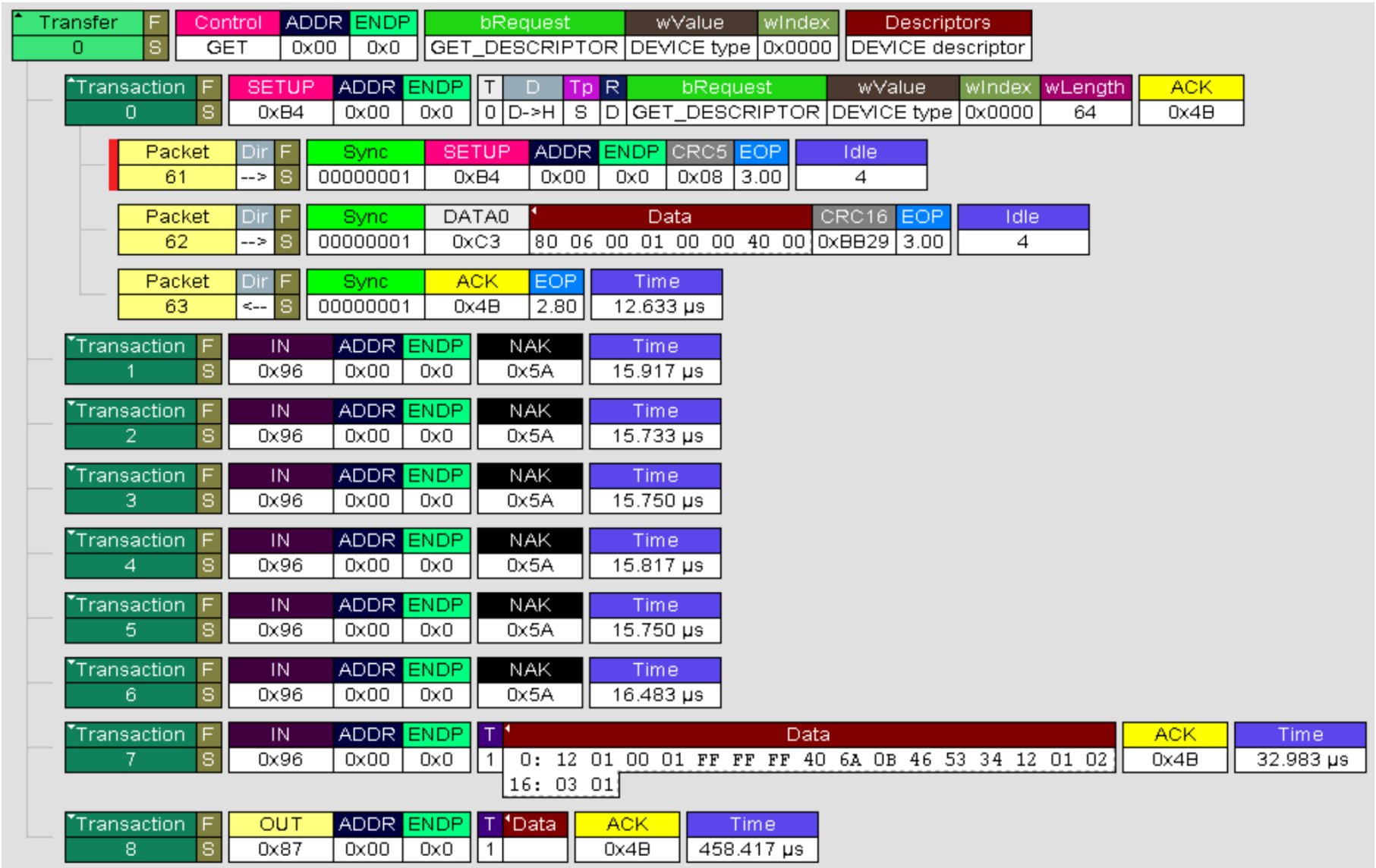
T	Data	ACK	Time
1	0: 12 01 00 01 FF FF FF 40 6A 0B 46 53 34 12 01 02 16: 03 01	0x4B	32.983 µs

Transaction	F
8	S

OUT	ADDR	ENDP
0x87	0x00	0x0

T	Data	ACK	Time
1		0x4B	458.417 µs

Building USB Packets, Session (with NAKs)



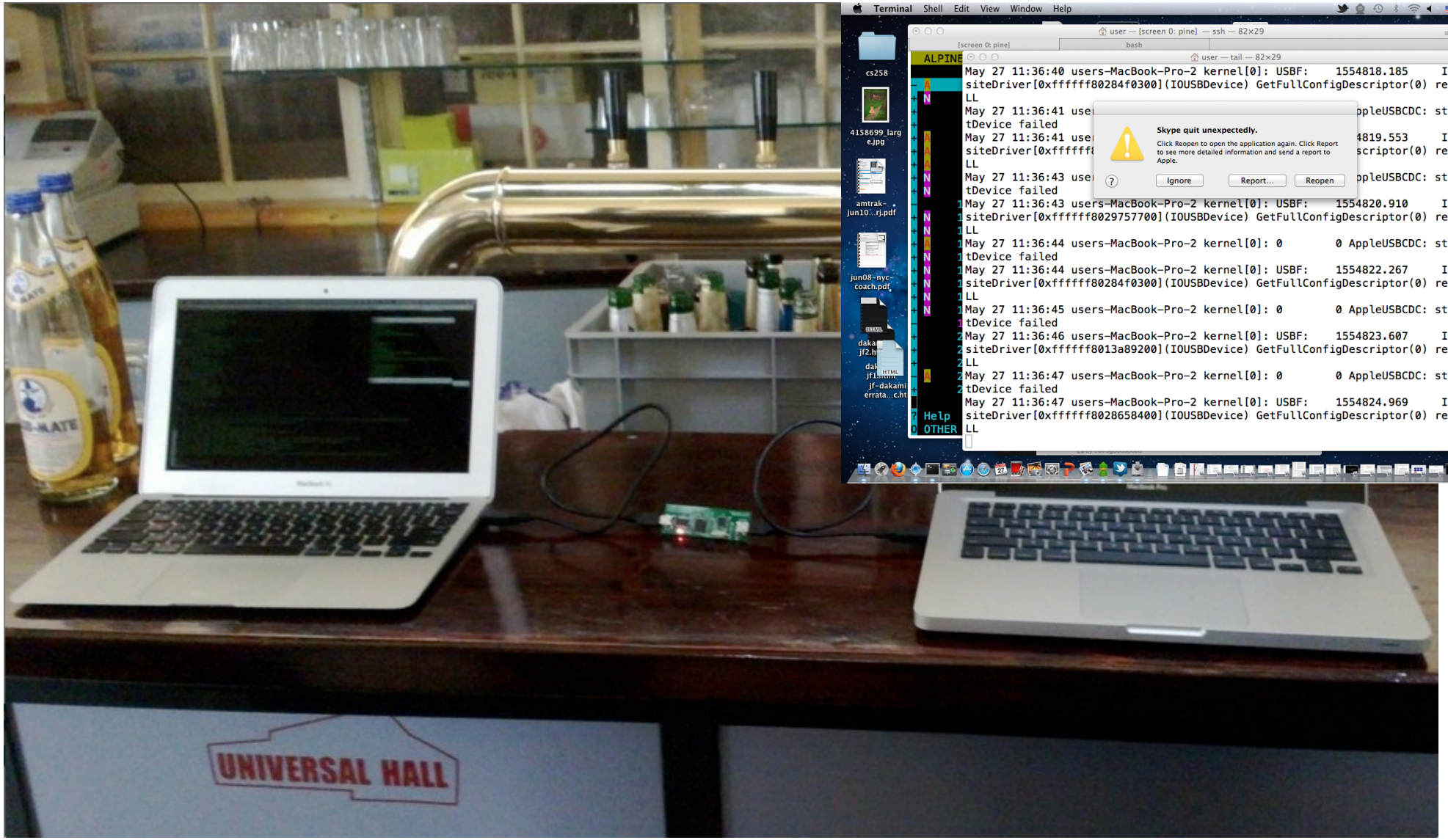
USB Recap

- Ports are called Endpoints.
- EP0 or the SETUP endpoint is for autoconfiguration.
- The setup exchange is called Enumeration.
- Devices are described by Descriptors.
  - Structs unique to each device class.
  - Nested lengths, offsets spell trouble
- Class types are standardized. (HID, Mass Storage)
- Vendor types are not. (FTDI, Wi-Fi)

Facedancer Rapid Exploit Development

- Easy to build raw USB packets
- Emulators written in Host-side Python
- Easy to rig up a quick fuzzer – for any kernel component routed to by USB stack

Facedancer in Action



HID Format String

- Ubuntu 12.04, Xorg
- Manufacturer String: “%n%s%n%s%n%s”
- Device String: “%n%s%n%s%n%s”
- Thanks to the ChromeOS team!

Exploiting Enumeration

- Host requests the first few bytes of the descriptor.
- Host mallocs that many bytes.
- Host reads the entire descriptor into a temporary buffer.
- Host memcpy() the descriptor into the malloced buffer.
- PSGroove exploits this on the Playstation 3!

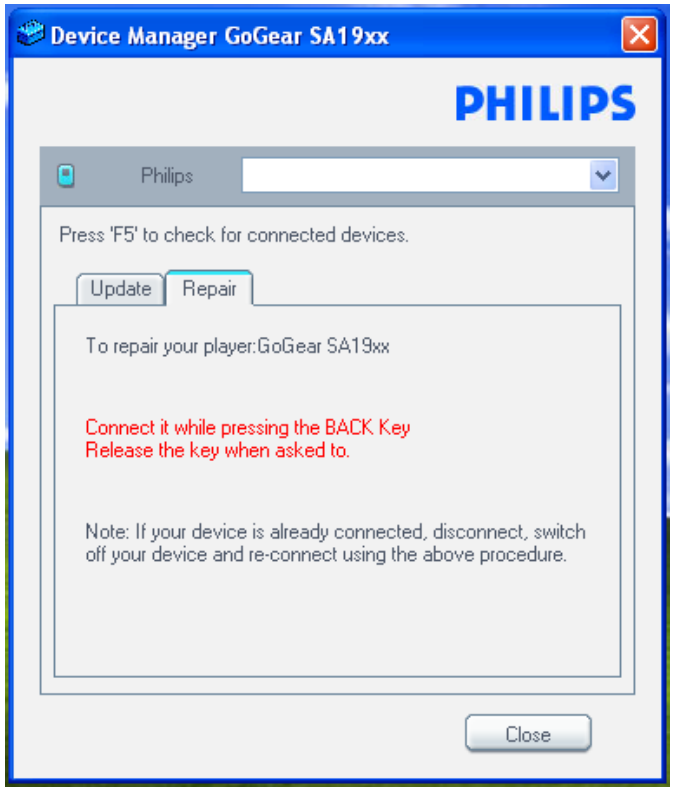
Exploiting Enumeration

- Host requests the first few bytes of the descriptor.
- Host mallocs that many bytes.
- Host reads the entire descriptor into a temporary buffer.
- Host memcpy() the descriptor into the malloced buffer.
- PSGroove exploits this on the Playstation 3!

Public Facedancer Emulators

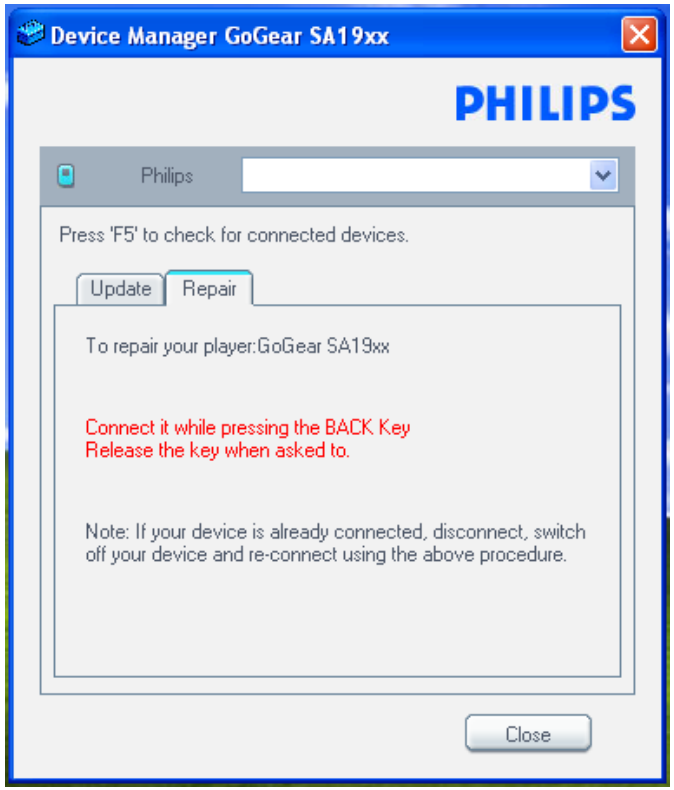
- Human Interface Device
- FTDI USB->Serial
- *Device Firmware Update*
- Mass Storage

USB Device Firmware Update



- DFU is a standard class for accepting new firmware.
- Every implementation uses a different dialect.
- Often requires a key combo or recovery mode.

DFU Emulation



- Catch firmware updates.
- Emulator logs allow quick learning of new dialects.
- Updates can be replayed to patch devices.

DFU Verbs

0x00 DETACH
0x01 DNLOAD
0x02 UPLOAD
0x03 GETSTATUS
0x04 CLRSTATUS
0x05 GETSTATE
0x06 ABORT

0x05 GETSTATE

- GETSTATE (0x05) often comes first.
- dfuIDLE (0x02) is often a safe answer.

0x00 appIDLE
0x01 appDETACH
0x02 dfuIDLE
0x03 dfuDNLOAD_SYNC
0x04 dfuDNBUSY
0x05 dfuDNLOAD_IDLE
0x06 dfuMANIFEST_SYNC
0x07 dfuMANIFEST
0x08 dfuMANIFEST_WAIT_RESET
0x09 dfuUPLOAD_IDLE
0x0a dfuERROR

0x05 GETSTATE

- GETSTATE (0x05) often comes first.
- dfuIDLE (0x02) is often a safe answer.

0x00 appIDLE
0x01 appDETACH
0x02 dfuIDLE
0x03 dfuDNLOAD_SYNC
0x04 dfuDNBUSY
0x05 dfuDNLOAD_IDLE
0x06 dfuMANIFEST_SYNC
0x07 dfuMANIFEST
0x08 dfuMANIFEST_WAIT_RESET
0x09 dfuUPLOAD_IDLE
0x0a dfuERROR

0x03 GETSTATUS

- GETSTATUS (0x03) describes the success of UPLOAD or DNLOAD.
- Returning six bytes of zeroes usually works.
- See documentation for the exact meaning.

0x01 DNLOAD

- DNLOAD (0x01) copies data into device memory.
- Uses EP0, like all other DFU commands.
- 16-bit length.
- 16-bit block index.

0x02 UPLOAD

- DNLOAD (0x01) copies data out of device memory.
- Uses EP0, like all other DFU commands.
- 16-bit length.
- 16-bit block index.

Block Addressing

- DNLOAD/UPLOAD use awkward addressing.
- Address is a 16-bit block number.
- Length often implies block size, sometimes not.
- Sometimes block zero is special, often not.
- Sometimes start address is variable.
- Usually start address is beginning of Flash region.

DFU Emulator

What's missing?

- Many common functions are undefined by DFU.
- Each DFU programmer makes these up himself.
-
- Erase Segment, Erase Chip
- Protect, Unprotect
- Move Base Address
- Read Model Number
- Enter DFU Mode!

Entering DFU Mode

- Selected by software, as in Ubertooth.
- Selectable USB mode, as in Bluetooth adapters.
- Selected by a key combo, as in iPhone, iPod.
- Selected by IO pins, as in STM32, MSP430.

Failure to Enter DFU

```
u410% sudo ./ubertooth-dfu --write ~/Desktop/bluetooth_rxtx.dfu
Checking firmware signature
No DFU devices found - attempting to find Ubertooth devices

1) Found 'Ubertooth Zero' with address 0x1d50 0x6000

Select a device to flash (default:1, exit:0):1
Could not initialise Ubertooth - is the device connected and in DFU mode?
u410%
```

Unhandled Vendor

```
u410% ./goodfet.maxusbdfu 1d50 6000
Connected to MAX342x Rev. 4

The DFU emulator is now running.  Any firmware
the virtual device will be locked to this consc
block device.
Starting a DFU device as 1D50:6000

Blindly accepting unhandled vendor request 19
```

Always DFU Mode

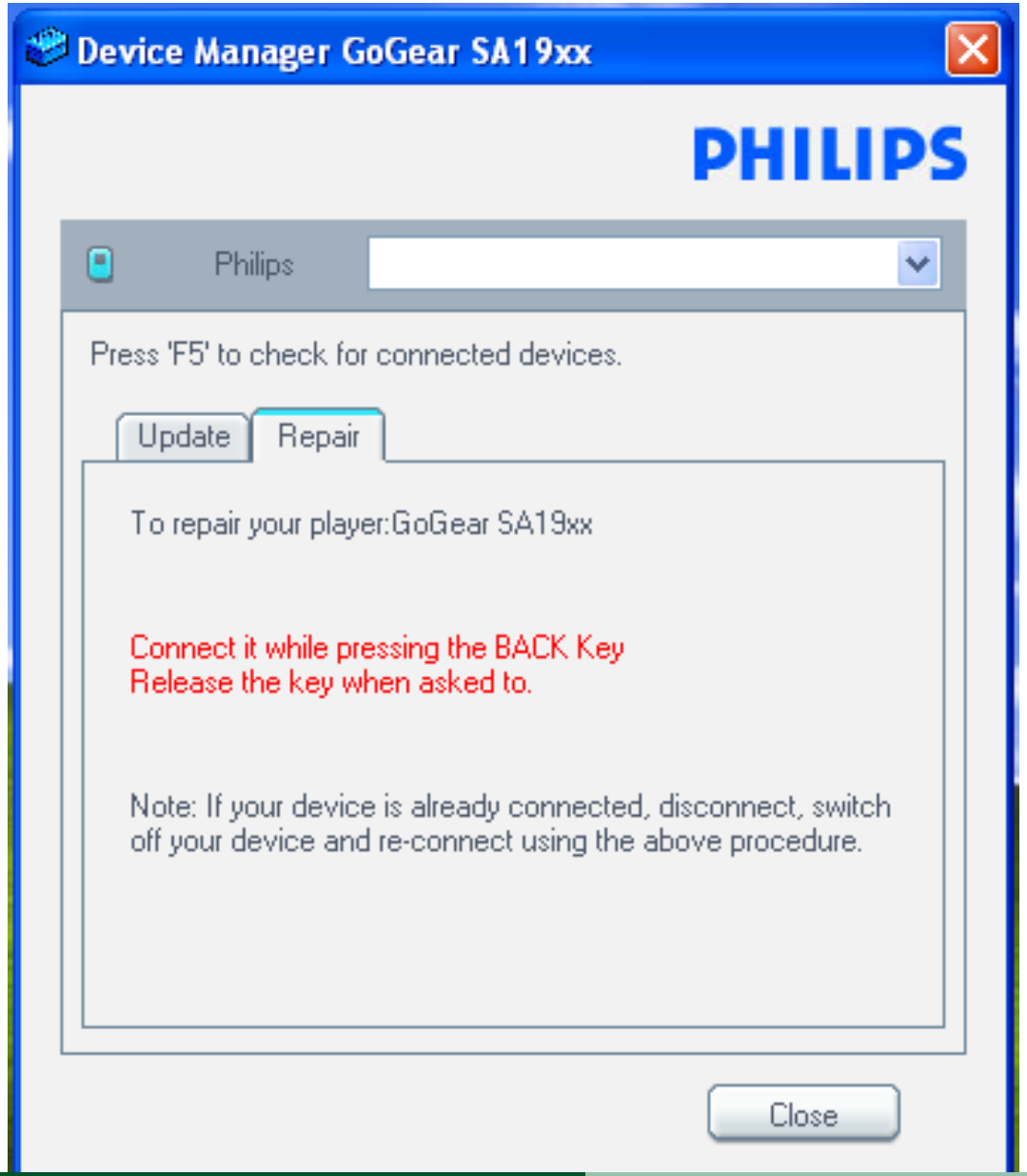
Always DFU Mode

- Sometimes DFU exists but is not advertised.
- Usage of EP0 allows DFU to coexist with other protocols.
- Scanning (probably) won't hurt.

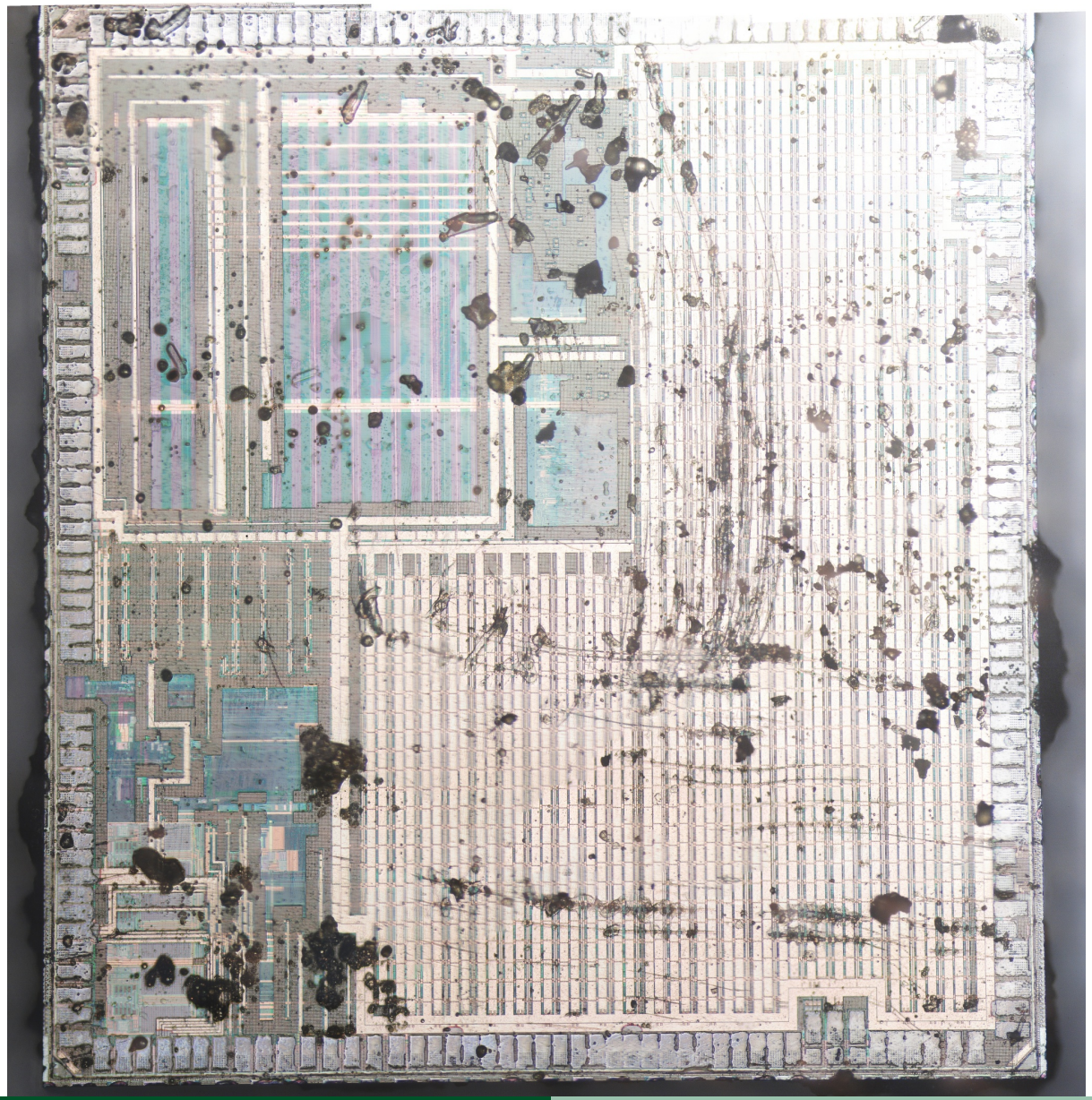
Selectable State

- Bluetooth in a 2011 MacBook Pro
- DFU Mode is a selectable USB Configuration.
- Run 'sudo lsusb -v | less' to see details details.

Key Combination



Selectable by IO Pins



Layers of Abstraction Are Boundaries of Competence

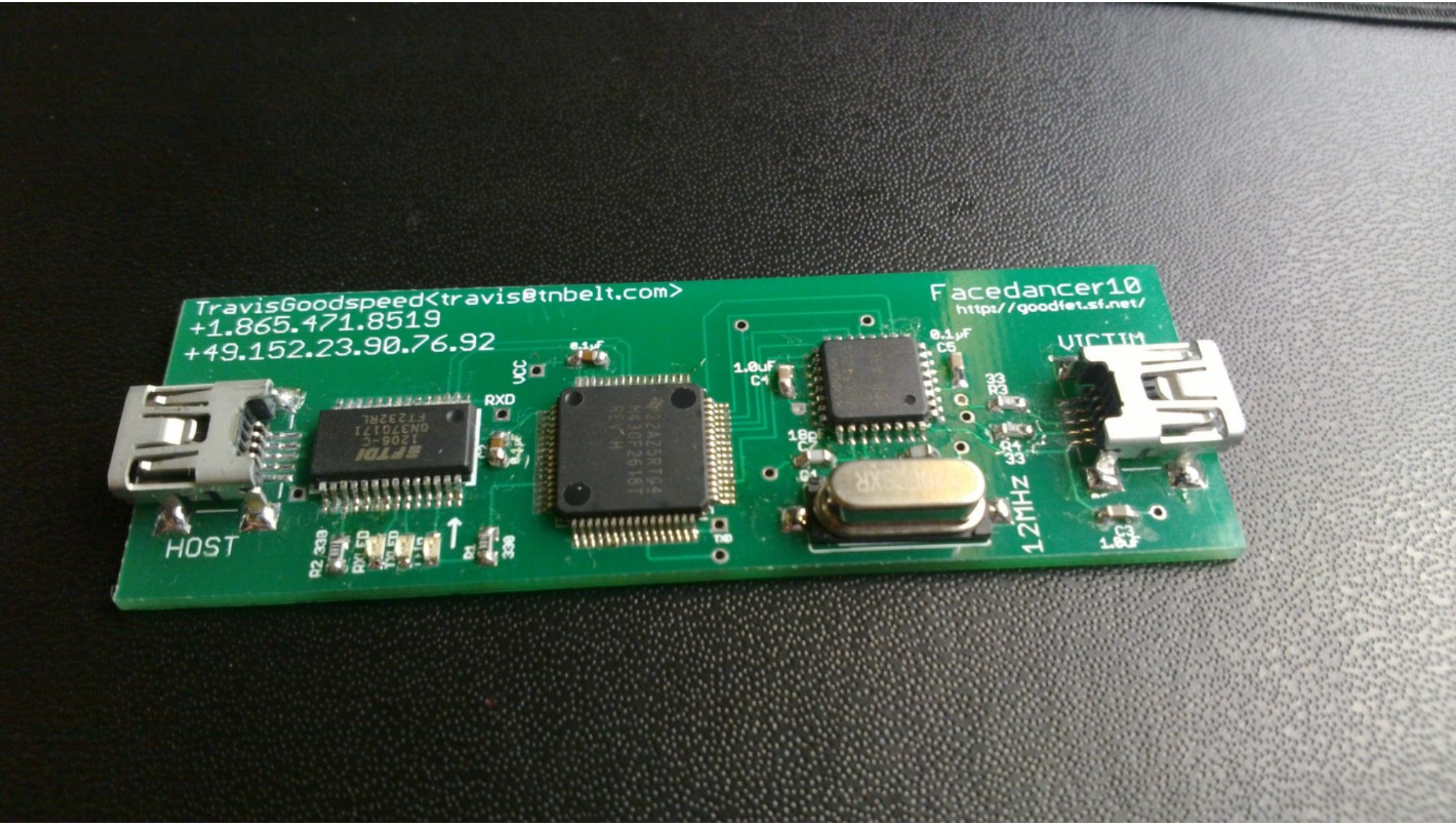
← “Fast path”, cross-layer design

WTF 1.0, reference implementation →

Conclusions

- USB opens a massive attack surface to inputs.
- Tools are finally available.
- Device emulators/fuzzers are easy to write.
- This is a fountain of 0day.

Facedancer Board



Read the Fucking Papers!

- http://travisgoodspeed.com/
- http://goodfet.sf.net/
- Academic search: “goodspeed AND bratus”