FaceDancer 2

Easy USB hacking, sniffing, and spoofing

Dominic Spill
@dominicgs

K. Temkin
@ktemkin

Thank you (in no particular order) to:

Travis Goodspeed (@travisgoodspeed)

Sergey Bratus (@sergeybratus)

Michael Ossmann (@michaelossmann)

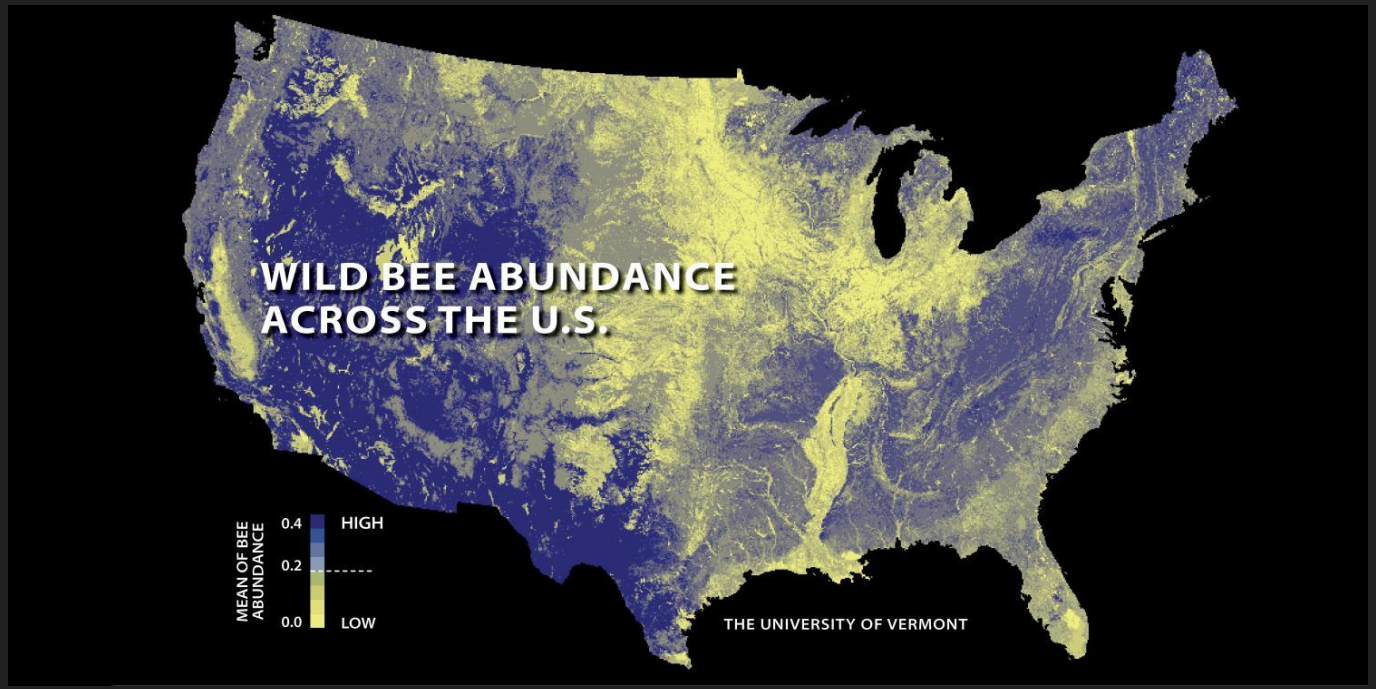
Micah Elizabeth Scott (@scanlime)

Assured Information Security (@ainfosec)

Great Scott Gadgets (@gsglabs)

Why hack on USB?

USB is **everywhere**.



Why hack on USB?

*Practical security does not improve until tools for **exploration of the attack surface** are made available.*

—Joshua Wright, Toorcon 11, 2009

Why hack on USB?

The capability to fuzz / monitor / emulate / MITM USB devices enables:

- **Finding vulnerabilities in** USB or driver stacks
- **Building tools** that work with existing hardware / software
- One to get a **foot in the door** for attacking black box systems.
- Building implants and tools for **playing NSA**.

USB, how does that work?

USB, how does that work?

USB Basics

TL;DR:

- USB is a **simple, two*-wire protocol developed** to address limitations of legacy communications ports (RS-232, PS/2, IBM EPP/LPT).
- Somewhat complex if you're a host, but **simple if you're a device**.
- From a very high-level developer perspective, it's basically just a fancy way of **squishing lots of bytes back and forth through a narrow pipe**.
    - USB does enforce some standardization of the data transferred.

Endpoints

An abstraction for each of multiple **communications channels** multiplexed over a single set of **USB data lines**.

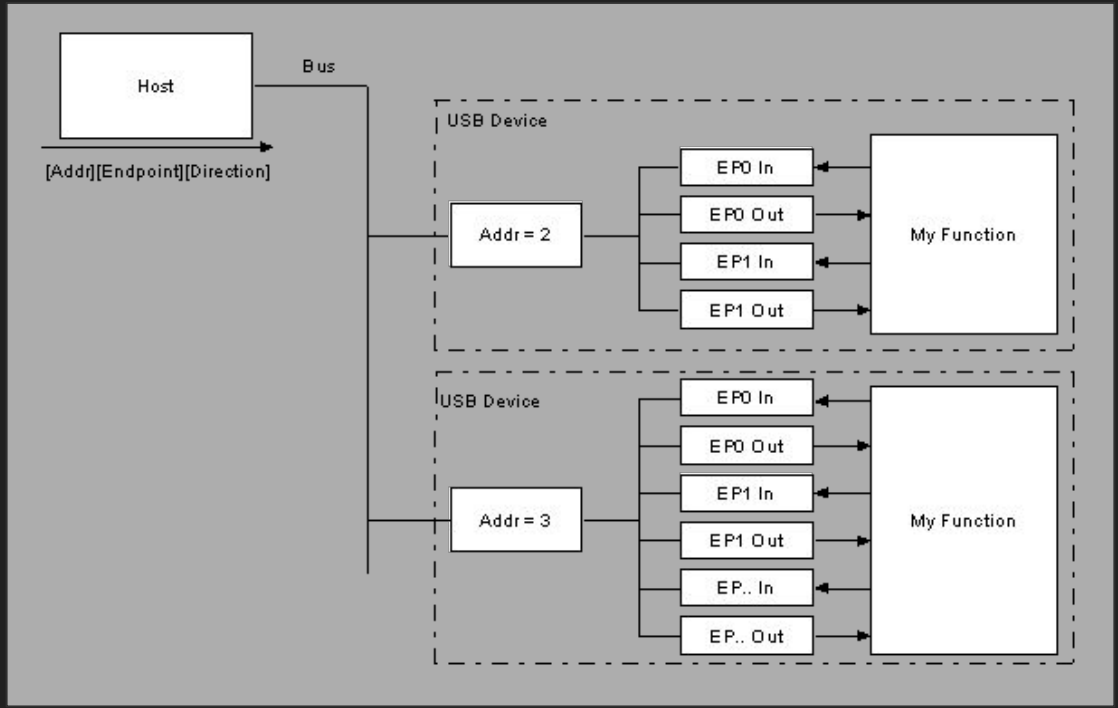


Image Credit: BeyondLogic “USB in a nutshell”

Endpoints

Control	Bidirectional communications channel used for standard communications and simple packetized back-and-forth. Used for initial device discovery and setup. Only transport that also specifies a packet format. EP0 is always a control endpoint.
Bulk	Unidirectional transport for shipping bytes ‘in bulk’-- bulk endpoints tend to get “all” of the leftover bandwidth on the bus.
Interrupt	Unidirectional transport for short bursts of latency-sensitive data. Used in cases that are similar to when you’d trigger an interrupt (e.g. keyboard keypress state).
Isochronous	Unidirectional transport for data that grows “stale” if not delivered quickly-- such as video frames from a camera.

Control Requests

Communications on EP0 are always packetized **control requests**, which are useful for sending simple **commands, data,** and **data requests**.

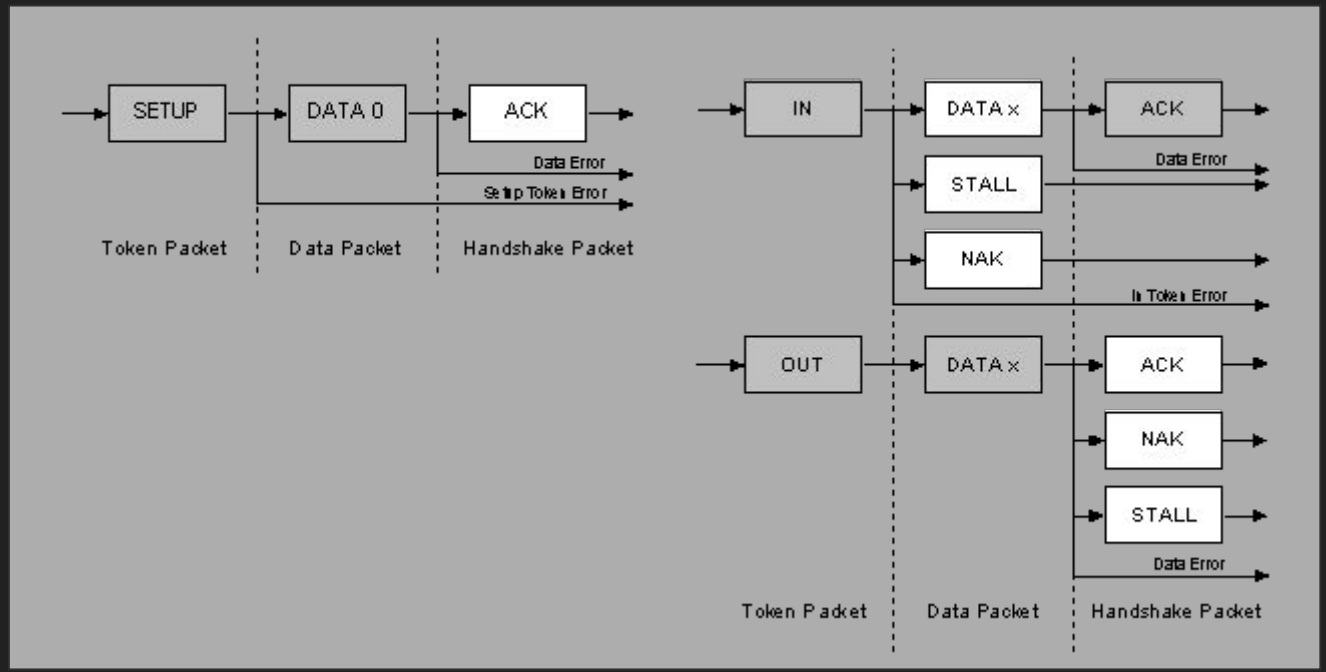


Image Credit: BeyondLogic
“USB in a nutshell”

Enumeration

One of the main advantages USB provides is the ability for devices to **self-describe**, a process known as enumeration:

- USB devices *describe themselves* and *their function(s)* by providing standard data blocks known as **descriptors** over the EP0 control channel**.**
  - These blocks provide a variety of information: the device's ID, string descriptions, how the endpoints can be configured, and etc.
- The device is initialized into a state where it can be **addressed on the bus**.

There are several valid ways to enumerate; many hosts do things slightly differently.

USB Classes

In addition to specifying the standard protocol used for enumeration/configuration, the specs also specify protocols for **standard device classes**, allowing e.g. operating systems to provide **standardized drivers**.

- Human Interface Device (keyboards, mice, datagloves; the usual)
- Serial (e.g. CDC-ACM)
- Mass storage (UMS bulk only / UAS)
- Audio / Video
- Midi
- Scanners
- Networking
- etc.

[show off real USB device here]

FaceDancer: a history

It’s not a bus, it’s a network - Sergey Bratus

FaceDancer: a history

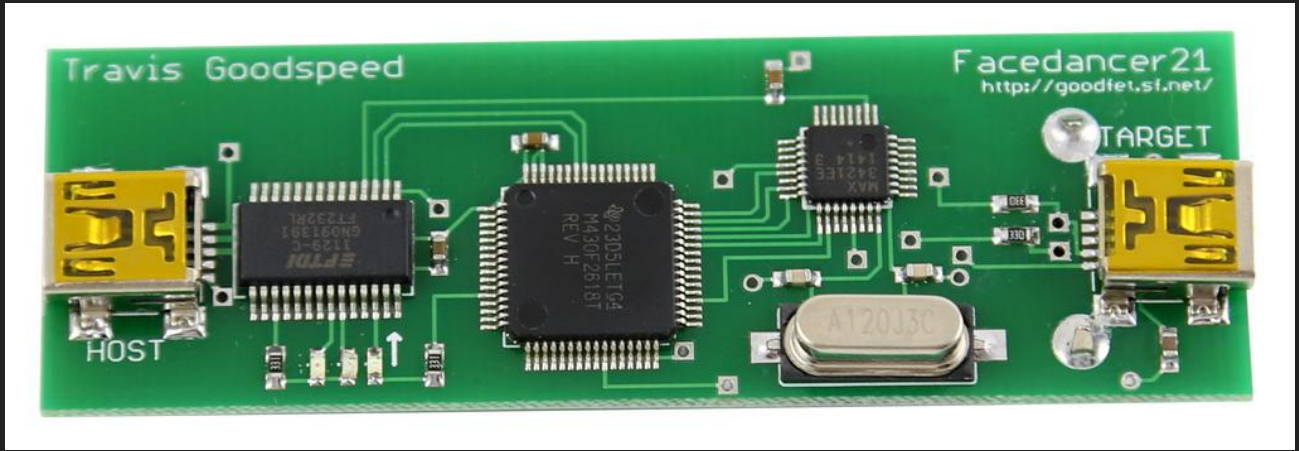
It’s not a bus, it’s a network - Sergey Bratus

I’ll build a thing! - Travis Goodspeed

FaceDancer: a history

It’s not a bus, it’s a network - Sergey Bratus

I’ll build a thing! - Travis Goodspeed



Limitations of Original Facedancer

Original Facedancer was a huge step forward, but suffered from limitations that prevented it from fully emulating modern devices:

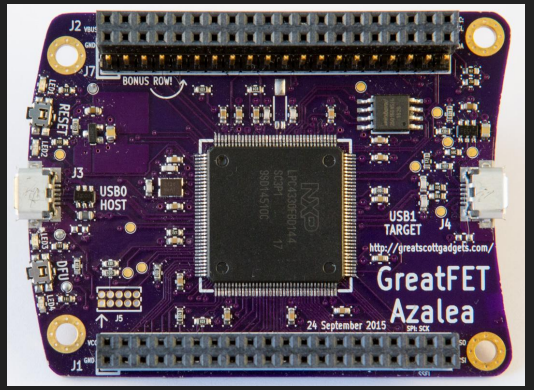
- The core USB chip, the MAX3421E, was capable only **providing four fixed-type endpoints**, preventing emulation of all but the simplest devices.
- The GoodFET-derived architecture passes all input through an FTDI USB-to-serial converter, **significantly slowing comms**.

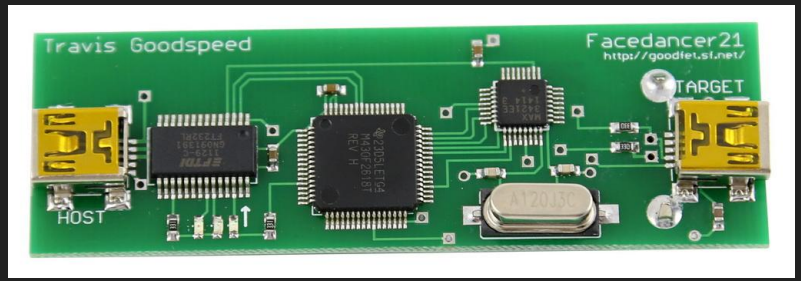
To overcome these, we worked to develop...

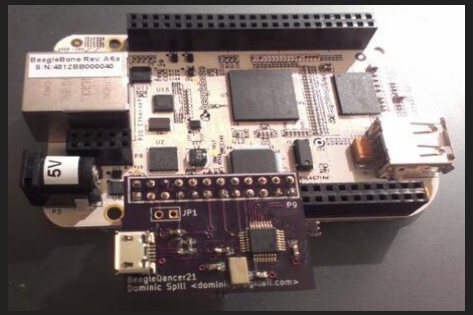
FaceDancer 2

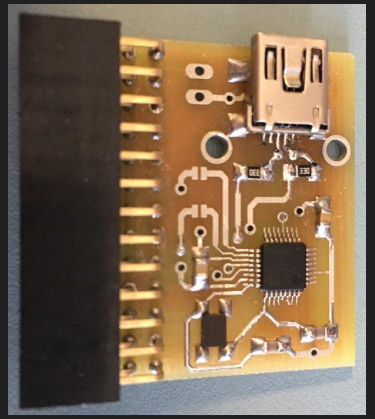
Including support for a variety of devices, including:

- FaceDancer
- RaspDancer / BeagleDancer
- GreatFET
- **Soon:** rad1o badge
- **Soon:** Linux with UDC - RPi Zero, BBB, etc.









New Features

New hardware platforms offer us the ability to add features well beyond the capability of the original FaceDancer hardware:

- Support for emulating USB 2.0 high speed devices*
- More flexible configurations; including:
  - Fully flexible endpoint configuration
  - More endpoints
- Support for MITM'ing USB connections for advanced attacks and monitoring

So, what can we do?

Quite a bit, it turns out:

- **Attack USB hosts** and driver stacks
  - Fuzzing, e.g. with umap
- **Fingerprint USB implementations** to e.g. identify host OSs
- Emulate **USB devices** for prototyping and emulation:
  - Keyboards
  - USB Mass Storage
  - FTDI
  - DFU (steal device firmware)

[it’s super easy; let’s look at an emulated USB-to-serial converter]

https://github.com/ktemkin/Facedancer/blob/master/USBSerial.py

device_emulation++: UMS double fetch

Of course, nothing says our emulated devices have to **behave nicely.**

Example: most systems assume that disk contents *don't change on their own*
Reality: in practice, *they totally can*

Example firmware update sequence:

- USB host reads firmware off flash drive, computing a checksum as it does
- USB host verifies the checksum, which passes
- USB host *rereads the firmware and flashes it to ROM*

Dominic Spill
@dominicgs

Currently discussing the logistics of carrying a full size photocopier to @SHA2017Camp as a target for a USB attack demo.

6:54 PM - 19 Jul 2017

3 Retweets **18** Likes

2 3 18

Tweet your reply

SHA2017 @SHA2017Camp · Jul 19

Replying to @dominicgs

Please discuss the logistics of taking it back as well. ;-)

1 13

dook @dooktwit · Jul 19

You've not seen how good his USB attack demo is. #Vaporized

1 4

SHA2017 @SHA2017Camp · Jul 19

If they can vaporise the copier on stage … that is fine as well. ;-)

3

[peanut butter demo time]

https://github.com/ktemkin/Facedancer/blob/master/facedancer-ums-doublefetch.py

device_emulation++: decompiling firmware

Neighbor @scanlime had an interesting use case:

- In her **Wacom-tablet-as-an-RFID work** (PoC||GTFO 0x13:4), she dumped firmware from an undocumented microcontroller using USB glitching magic.
- No **public ISA documentation** or **standalone disassembler** existed for this uC; but a vendor debug application would disassemble **any firmware read back by the vendor's unobtanium debug dongle**.

The natural solution?

- Emulate a debug dongle!

[thanks for the demo, Micah]

USBProxy

USBProxy is a tool that allows us to **proxy the connection** between a host and device. While proxying a connection we can:

- Log USB packets to pcap files
- Modify data being sent to or from a device
- Inject new packets into the connection

The original version was based on a BeagleBone Black. We are rewriting it to take advantage of FaceDancer 2.

[let’s monitor some USB]

https://github.com/ktemkin/Facedancer/blob/master/USBProxy.py

USBProxy using hardware

Some of the same capabilities are available through virtualisation tools, such as vUSBf and usbmon, so when would we want a hardware solution?

When we don’t control the host system, e.g. in:

- Games consoles
- In car entertainment
- Point of sale
- Televisions
- Any embedded device that allows USB devices to be connected to it

[annnnd MITM...]

https://github.com/ktemkin/Facedancer/blob/master/USBProxy.py

Future Work

- Support for anything with a UDC/OTC Linux driver, such as a RPi Zero
  - “Make your printer a FD2”
- Support for USB 3.0 (see also: Daisho)
- USB C and Power Delivery support
- USB Host support for e.g. controlled USB glitching

Questions?

https://github.com/ktemkin/FaceDancer

https://github.com/dominicgs/GreatFET-Experimental

@dominicgs

@ktemkin