

USB for All!!1

You should be looking at USB.

Yes, you.

Introduction

- Who We Are
 - Jesse Michael
 - Mickey Shkatov
- What We Do
 - Break things
 - Cry about the current level of security research focused on USB

DISCLAIMER: The views and opinions expressed in this presentation are those of the authors and not their employer.

Purpose of this talk



Purpose of this talk

- We want to demonstrate to attendees how easy it is to get started at performing their own USB security research and help them understand why they should undertake this challenge.

Why care about USB?

- We believe that even though USB is a pervasive technology in modern computing platforms, current security research has still only scratched its surface.
- USB has some interesting capabilities and is currently being used in a wide array of lesser-known usage models that can result in security problems.

Getting started

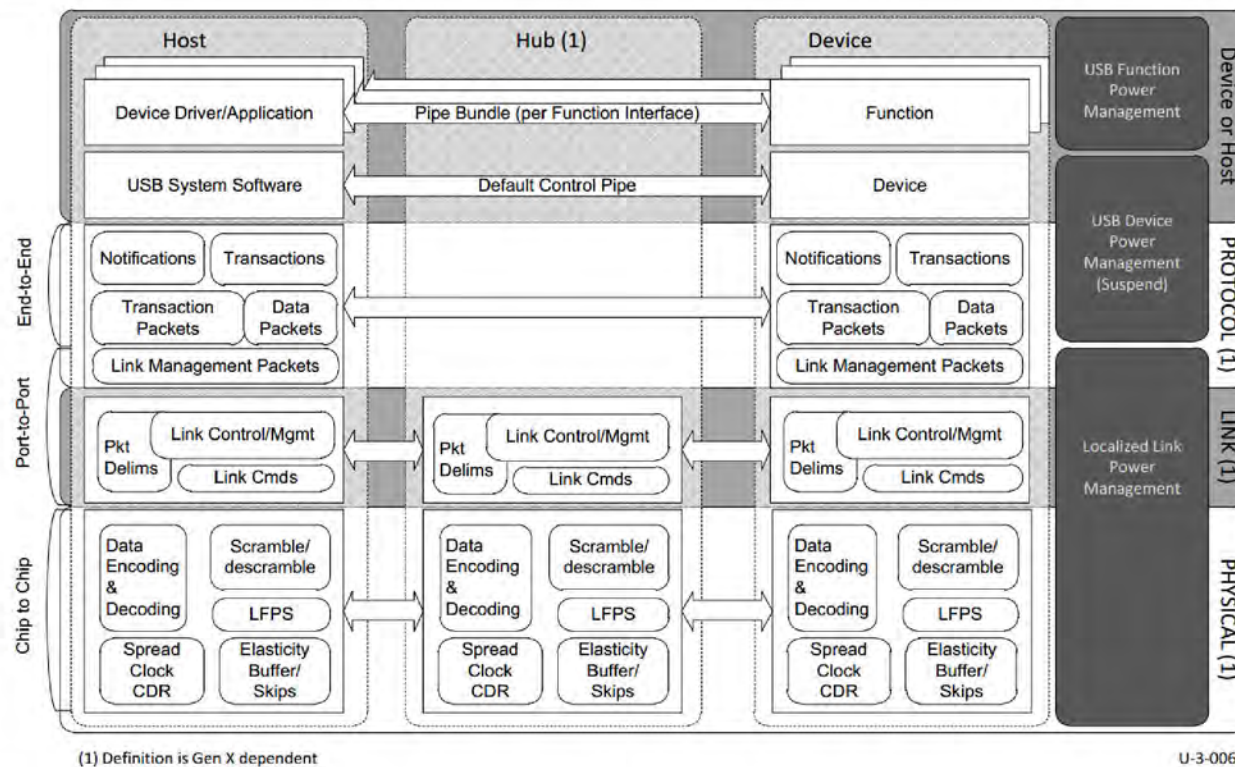


Figure 3-4. Enhanced SuperSpeed Bus Communications Layers and Power Management Elements

<http://www.usb.org/developers/docs/>

At over 600 pages, the USB specification can be a little intimidating...

Getting started

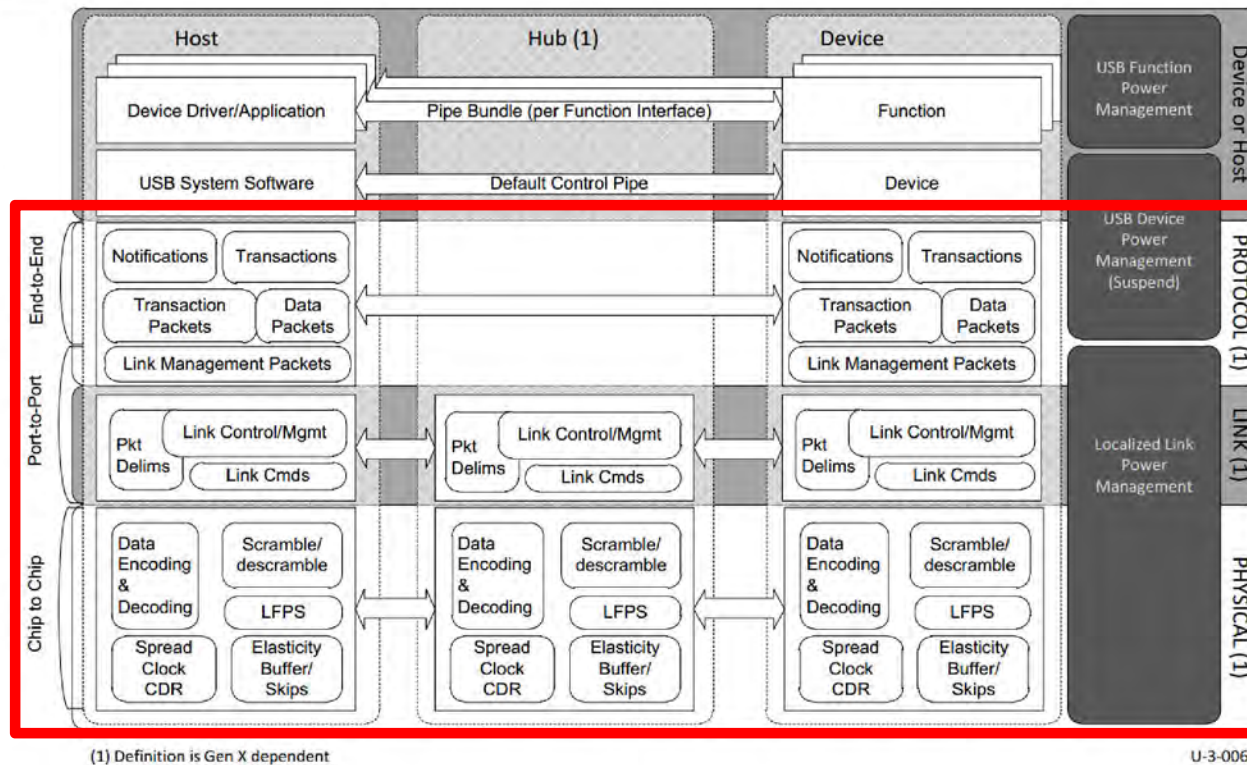


Figure 3-4. Enhanced SuperSpeed Bus Communications Layers and Power Management Elements

The physical, link, and protocol layers are mostly handled in hardware...

Getting started

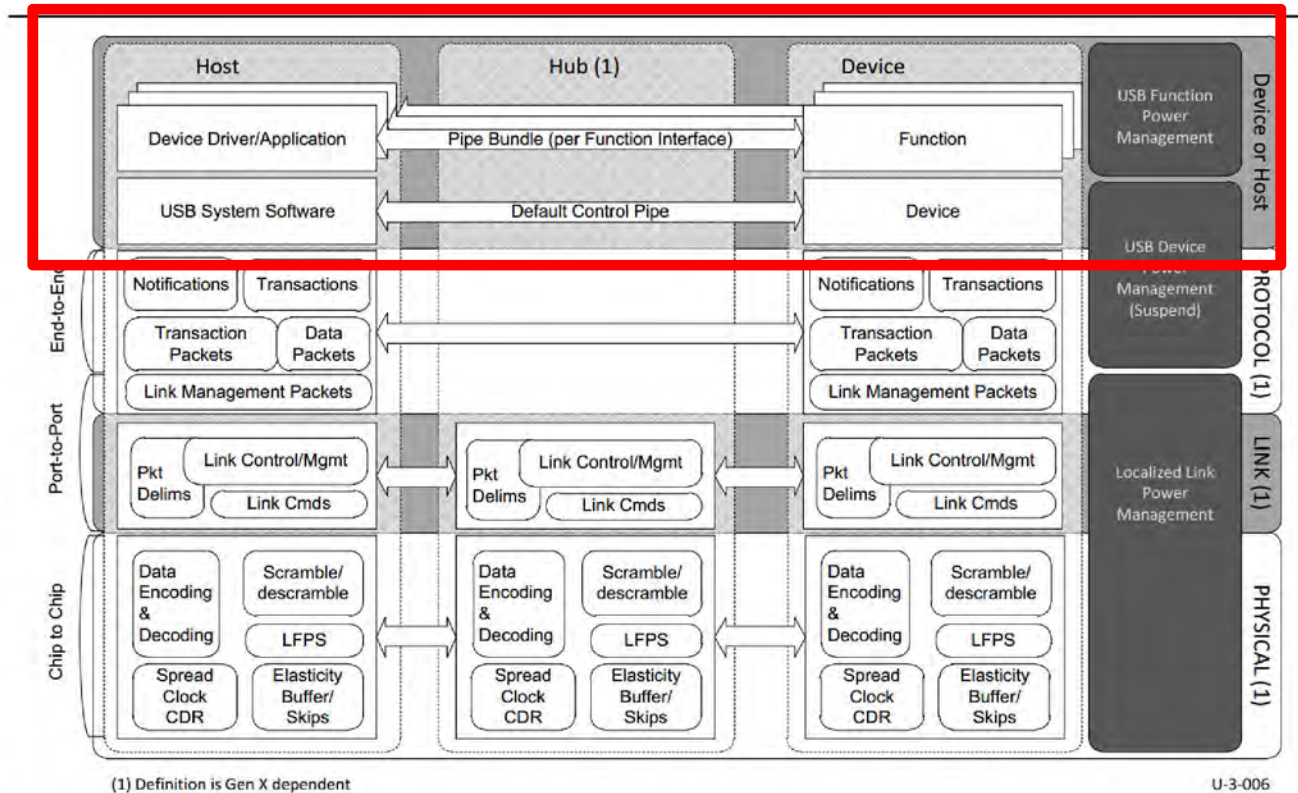
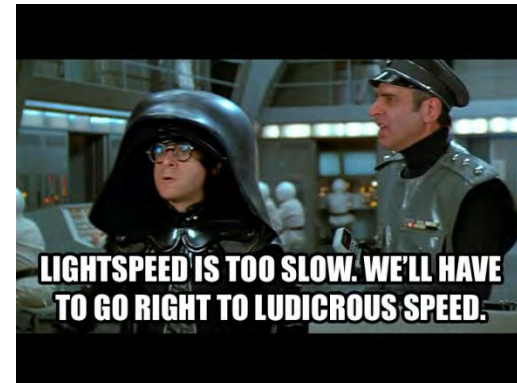


Figure 3-4. Enhanced SuperSpeed Bus Communications Layers and Power Management Elements

These areas are the easiest place to get started and find vulnerabilities so we'll focus here and on some bigger picture views of how USB devices are used in modern platforms.

USB Basics

- USB versions
 - 1.0, 1.1, 2.0, 3.0, 3.1
- Speeds
 - Low Speed, Full Speed, High Speed, Super Speed Gen1 and Gen2
- Device classes
 - HID, Mass Storage, Image, Video, Audio, Communications, Vendor Defined
- Physical connections
 - 1.x/2.0
 - D+, D-, Vbus, Gnd
 - 3.0+
 - Added SSTx+, SSTx-, SSRx+, SSRx-



Do I need to touch it?

Most people think about USB like this...



Do I need to touch it?

...but almost all modern laptops have internal USB devices.



Which often contain their own processors with firmware and are separate from the host CPU and operating system.

What could go wrong?

There's firmware in my USB?

Even “simple” USB devices can have interesting complexity

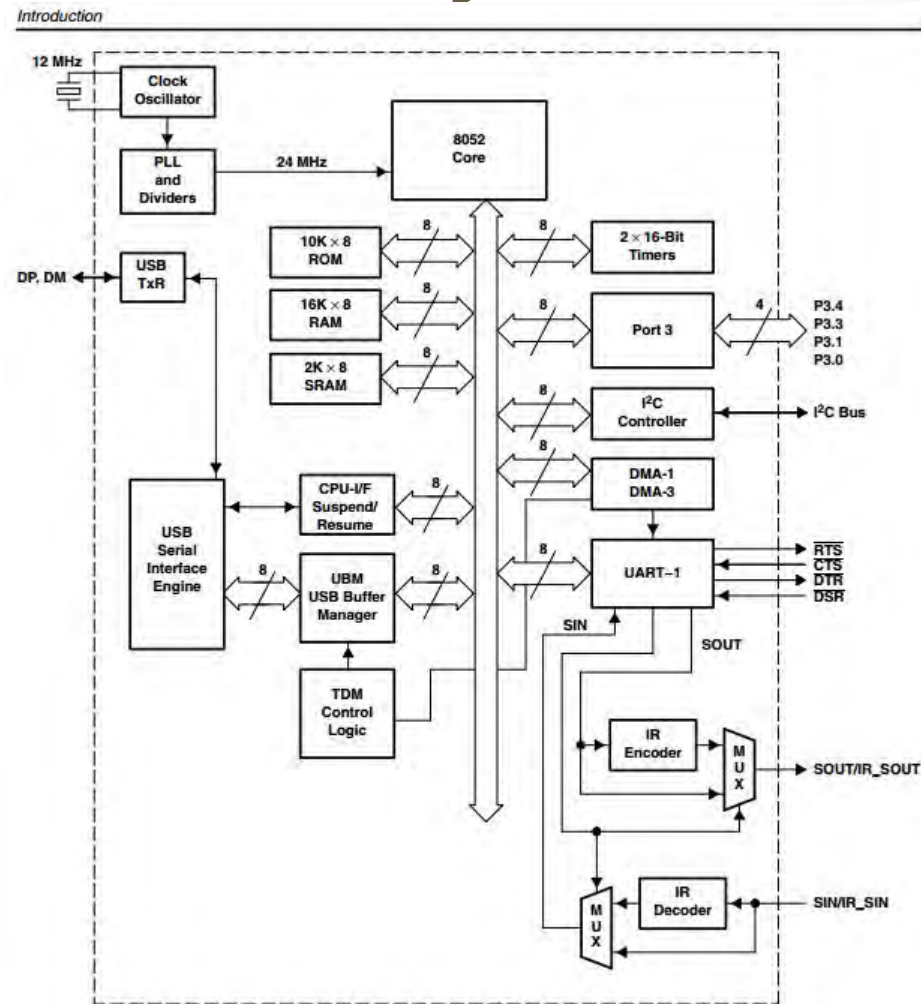
As an example, here's a sync cable for an older phone...



There's firmware in my USB?

That contains a USB to UART bridge chip that looks like this internally...

- 8052 processor
- 10K Boot ROM
- 16K RAM
- 2K SRAM
- Loads firmware from I²C



There's firmware in my USB?

And the datasheet describes how to run your own code in it...

11.8 Built-In Vendor Specific USB Requests

The bootcode supports several vendor specific USB requests. These requests are primarily for internal testing only. These functions should not be used in normal operation.

11.8.1 Reboot

The reboot command forces the bootcode to execute.

bmRequestType	USB_REQ_TYPE_DEVICE USB_REQ_TYPE_VENDOR USB_REQ_TYPE_OUT	01000000b
bRequest	BTC_REBOOT	0x85
wValue	None	0x0000
wIndex	None	0x0000
wLength	None	0x0000
Data	None	

11.8.2 Force Execute Firmware

The force execute firmware command requests the bootcode to execute the downloaded firmware unconditionally.

bmRequestType	USB_REQ_TYPE_DEVICE USB_REQ_TYPE_VENDOR USB_REQ_TYPE_OUT	01000000b
bRequest	BTC_FORCE_EXECUTE_FIRMWARE	0x8F
wValue	None	0x0000
wIndex	None	0x0000
wLength	None	0x0000
Data	None	

Arbitrary code execution inside your phone sync cable? Really?

Device Firmware Upgrade

http://www.usb.org/developers/devclass_docs/usbdfu10.pdf

There's actually a specification for how to create USB devices with upgradable firmware.

It doesn't mention security at all. And most devices that implement this capability don't bother to do any validation of the firmware image other than basic checksums which are easy to bypass.

DFU and similar custom device upgrade methods are a good way to easily get arbitrary code execution within a USB device.

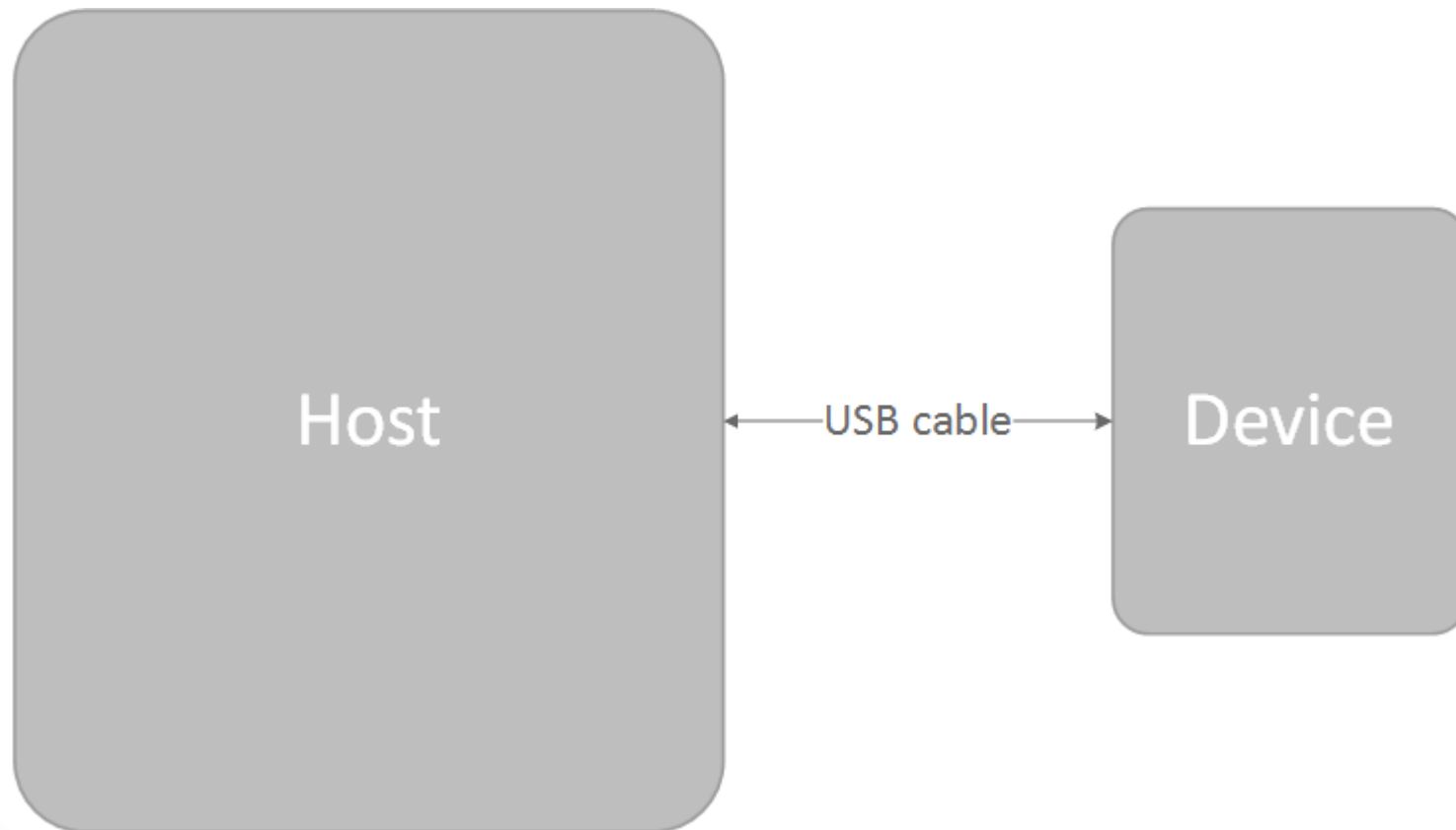
What can we do with that?

Debug Capability

- Allows low-level debug over USB
- Now required to gain Windows Logo certification
 - “If the XHCI controller in the SUT has any user-accessible ports, the controller must have debug capability.”

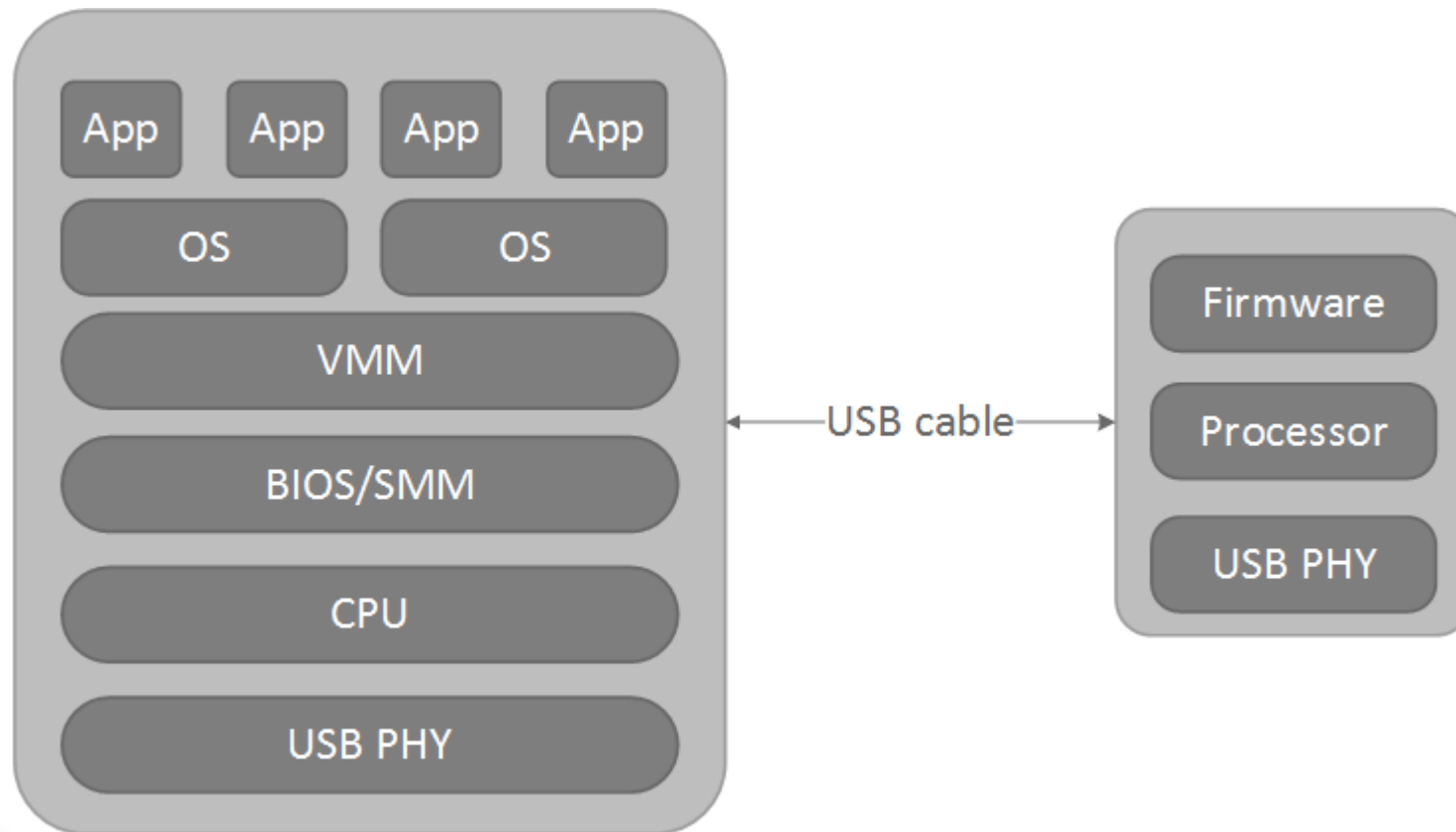
Attack Surfaces

So instead of looking at it like this...



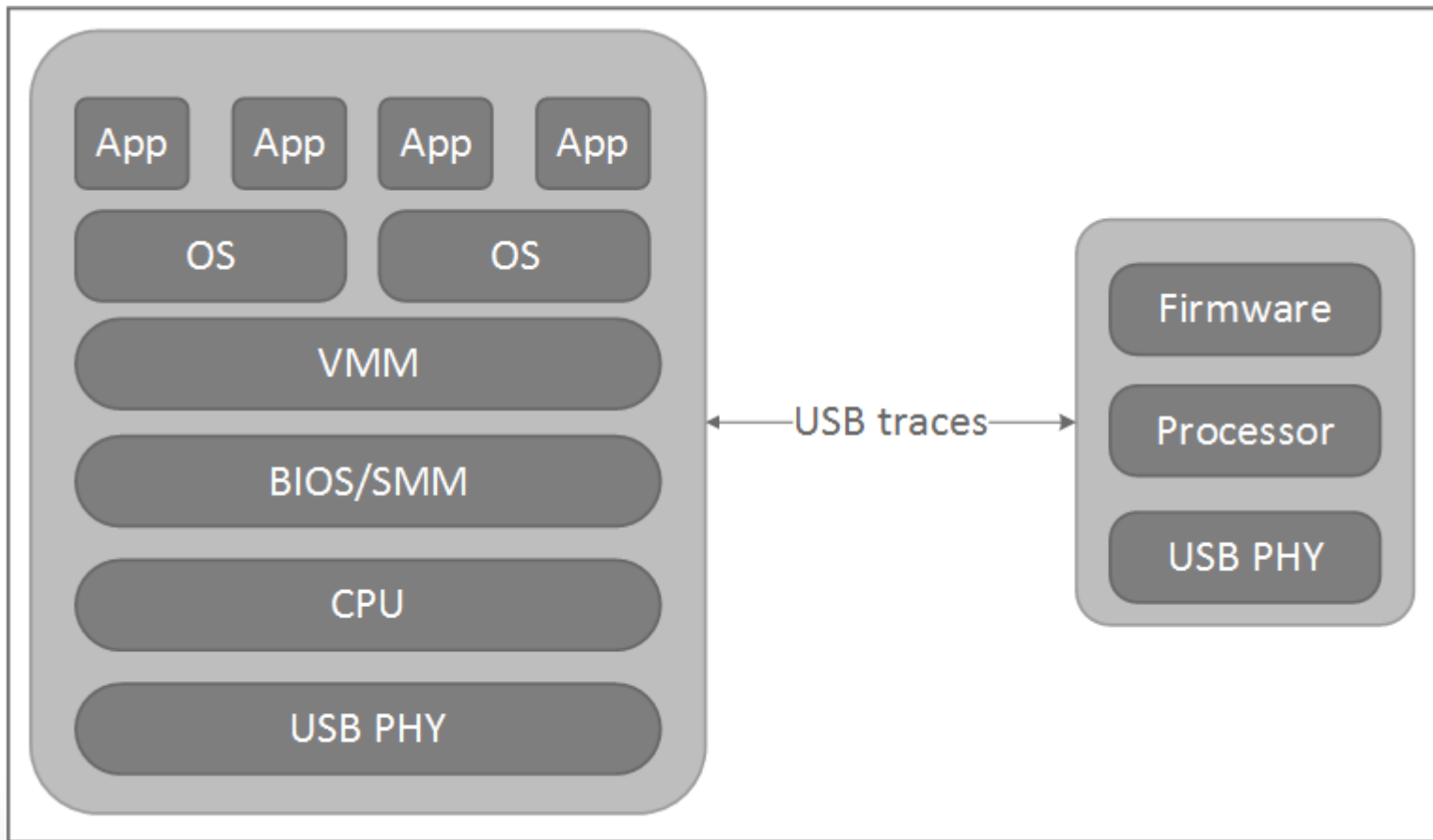
Attack Surfaces

There's actually a lot more going on...



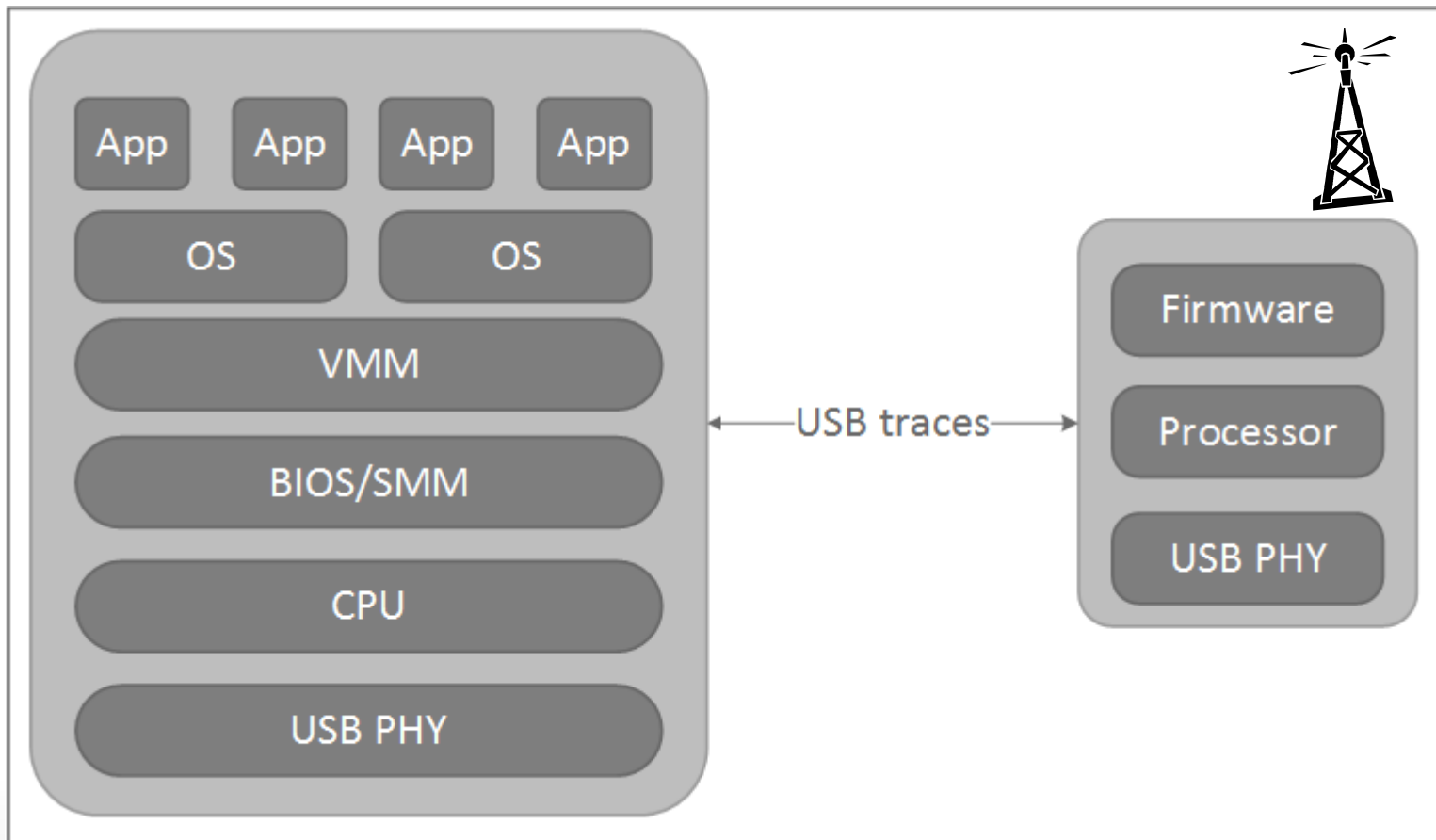
Attack Surfaces

All of this is probably happening inside your laptop right now.



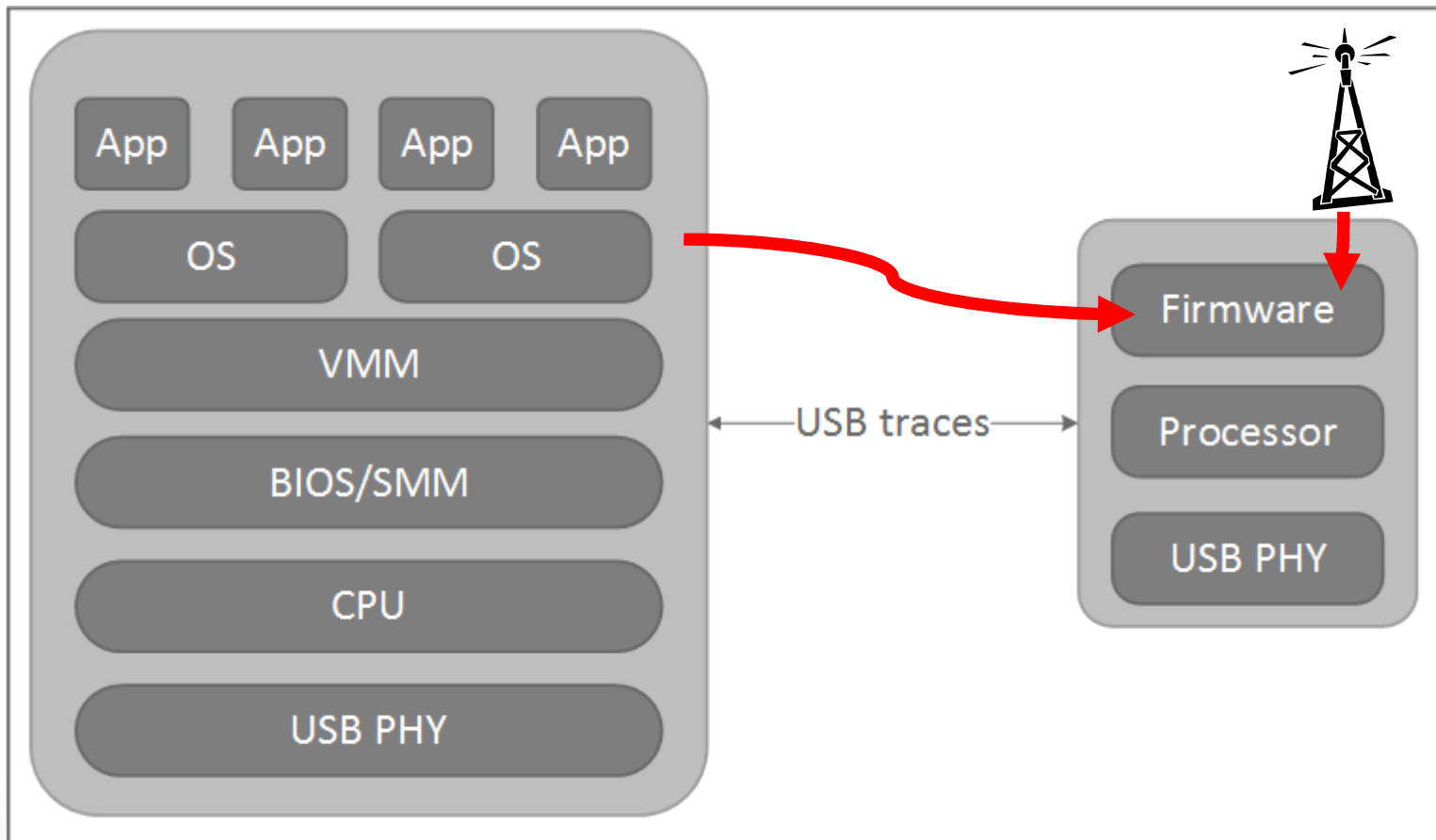
Attack Surfaces

Some USB devices even have radio interfaces...



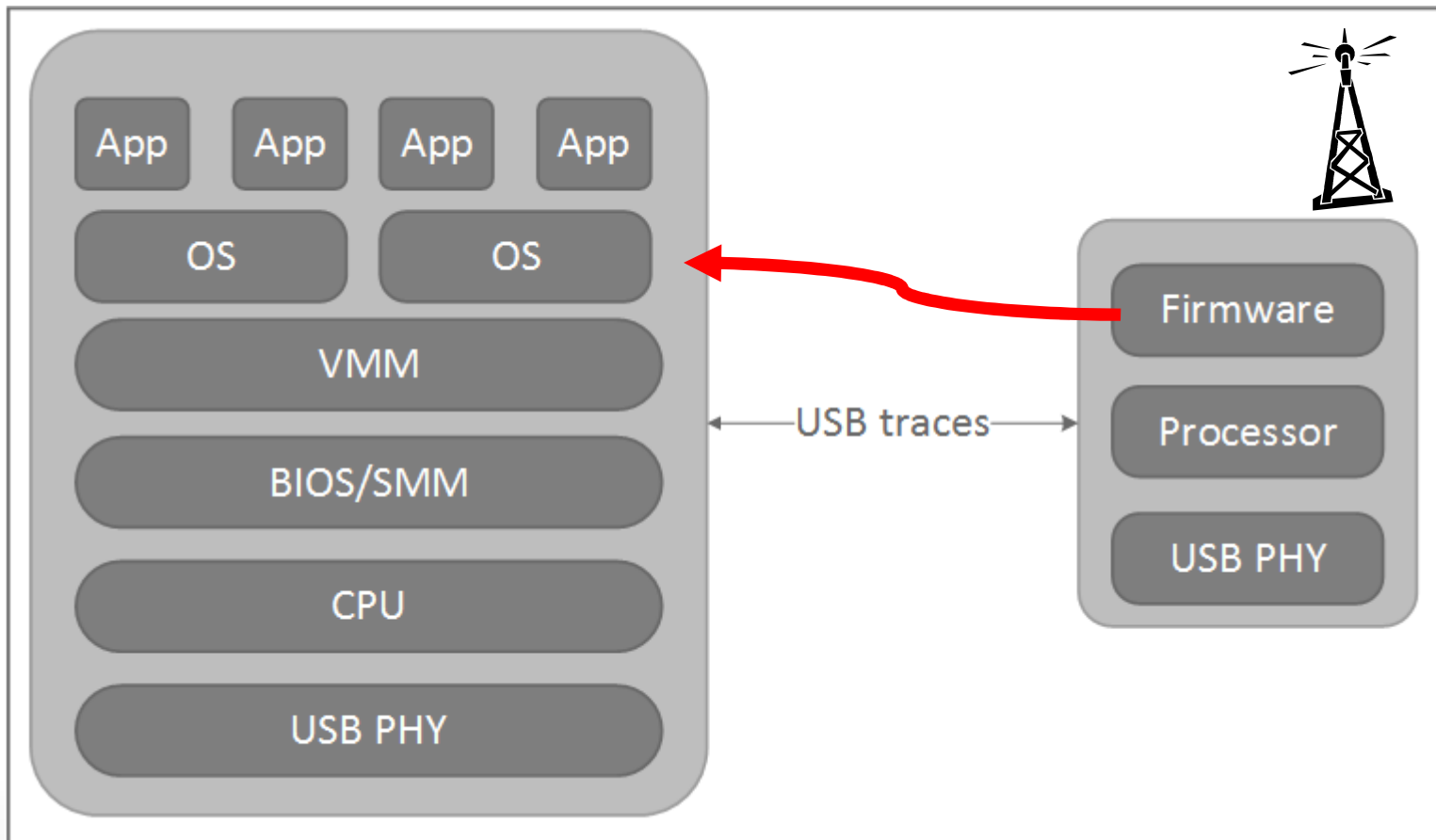
Attack Surfaces

If you can get arbitrary code execution within the USB device...



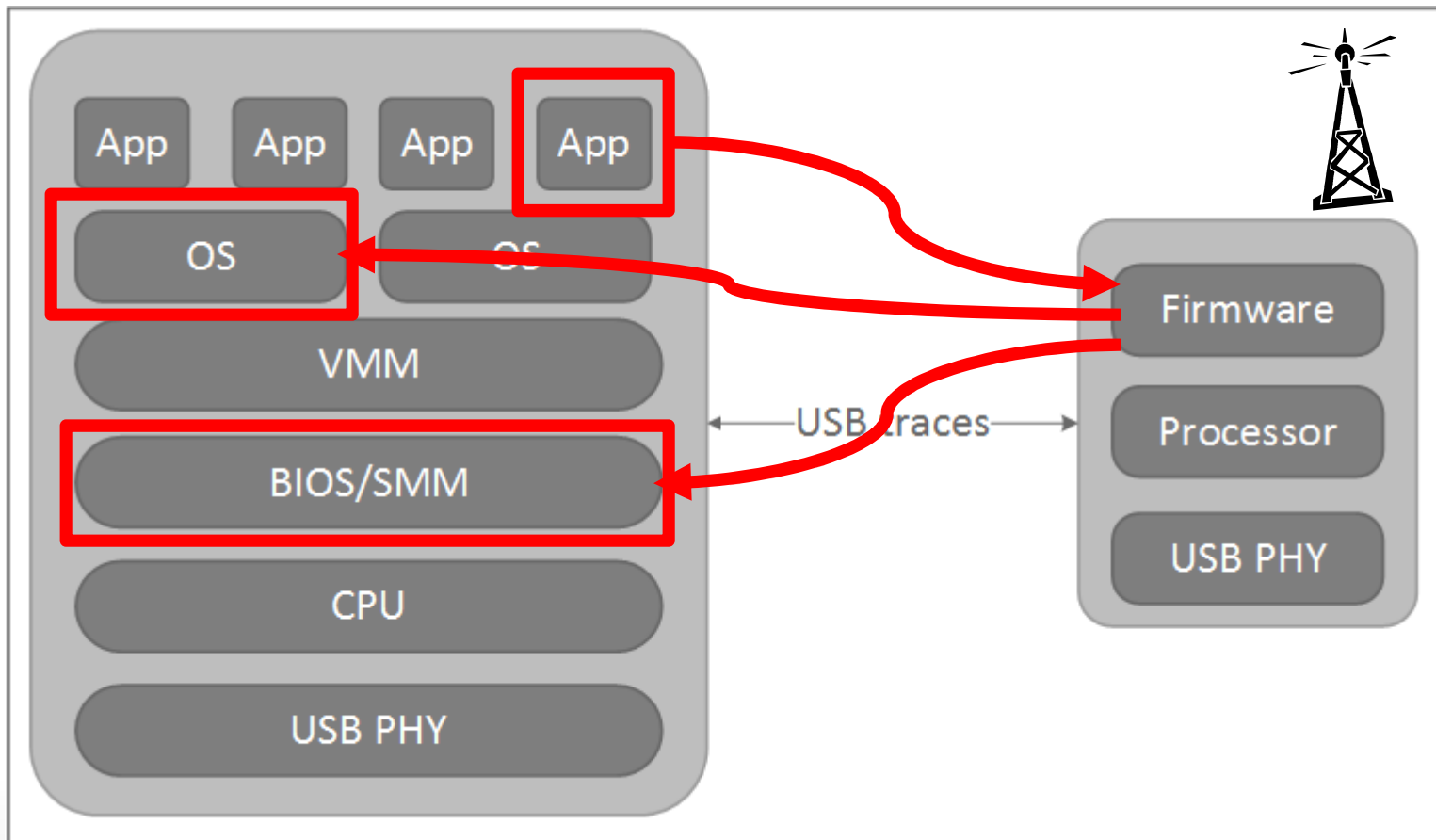
Attack Surfaces

It can be used to attack components within the host.



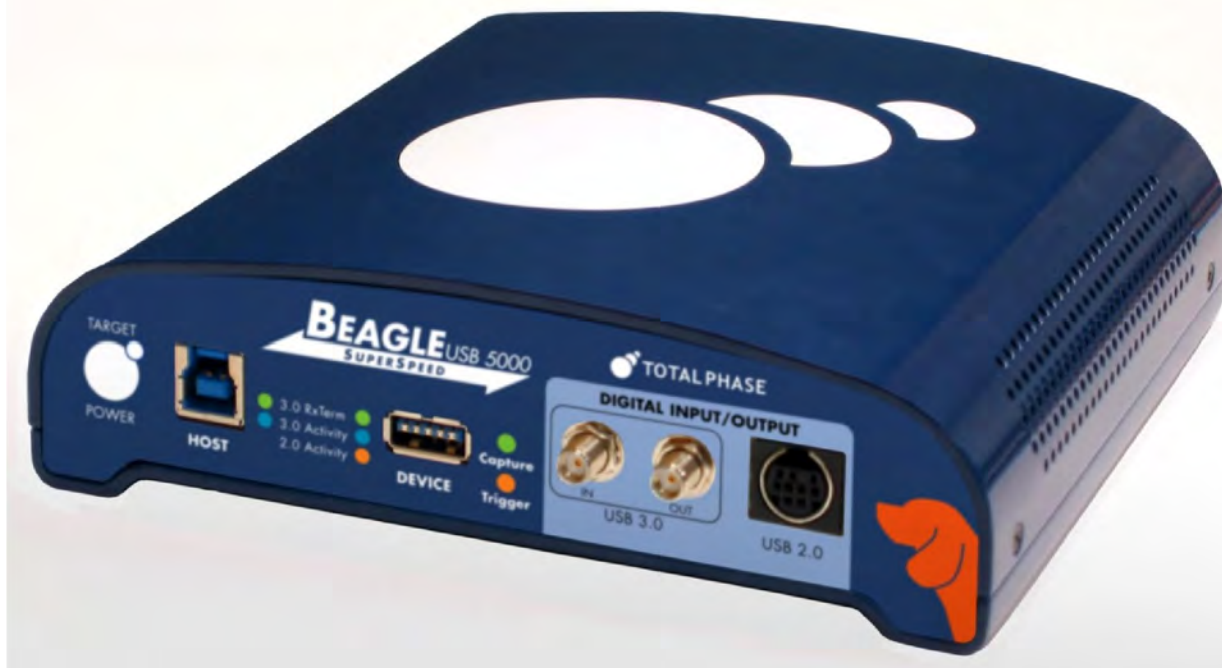
Attack Surfaces

Even with attacks originating from the host, these can cross privilege boundaries



Tools!

Total Phase Beagle 5000



<http://www.totalphase.com/protocols/usb/>

Supports USB 3.0 SuperSpeed, but very expensive. Can only be used for observation and not injection.

Total Phase Beagle 480



<http://www.totalphase.com/protocols/usb/>

Less expensive than Beagle 5000, but only supports USB 2.0. Can only be used for observation and not injection.

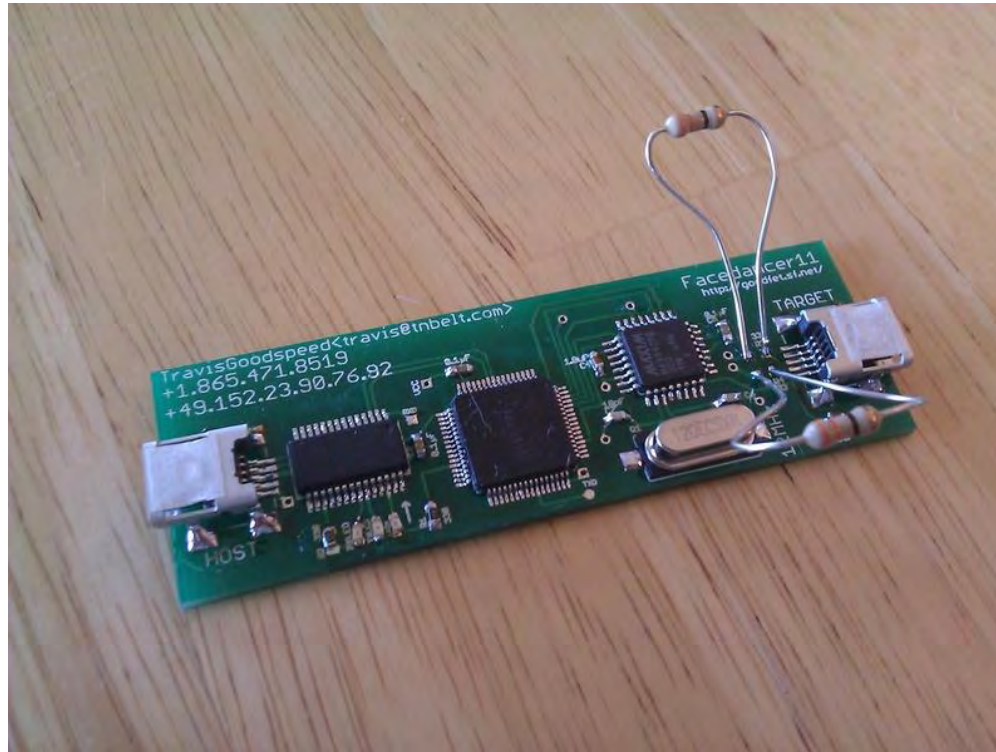
ITIC 1480A USB 2.0 Protocol Analyzer



<http://www.internationaltestinstruments.com/products/97-1480a-usb-20-protocol-analyzer.aspx>

HW less expensive than Beagle 480, but some SW modules sold separately.
Can only be used for observation and not injection.

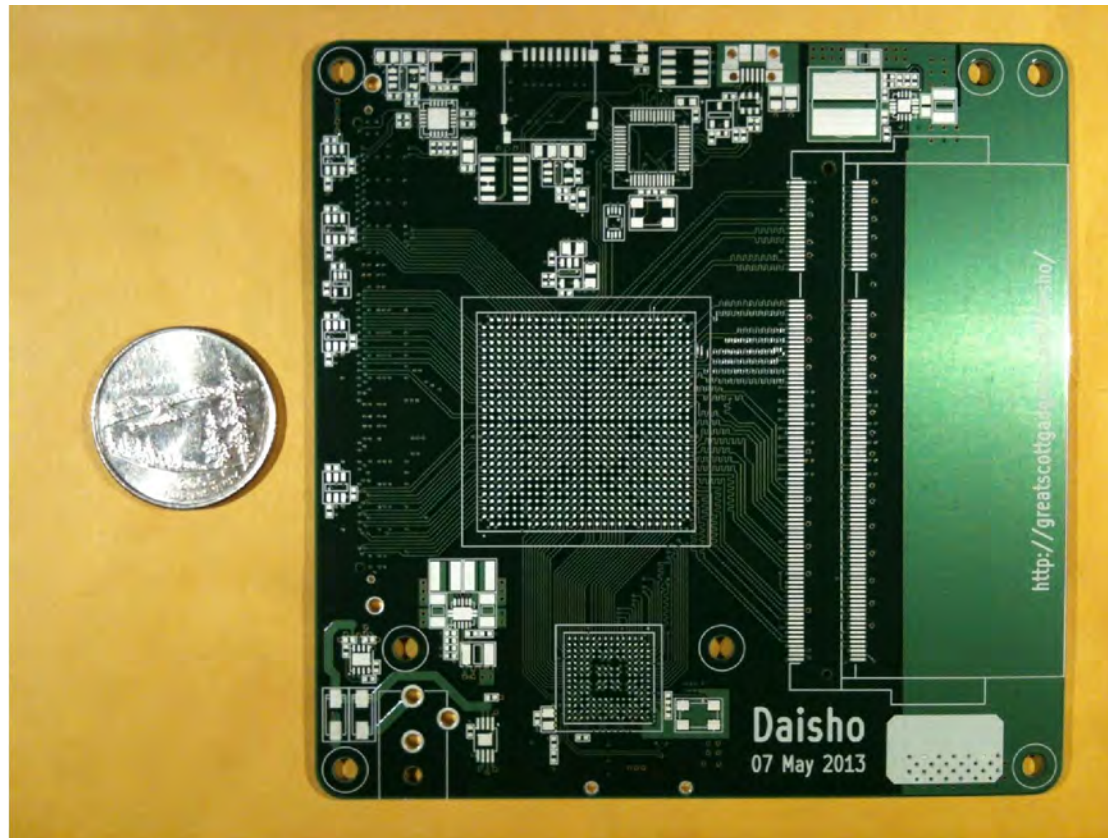
Facedancer



<http://goodfet.sourceforge.net/>

Open source, cheap and easy to build, allows arbitrary emulation of USB endpoints, but can be very slow

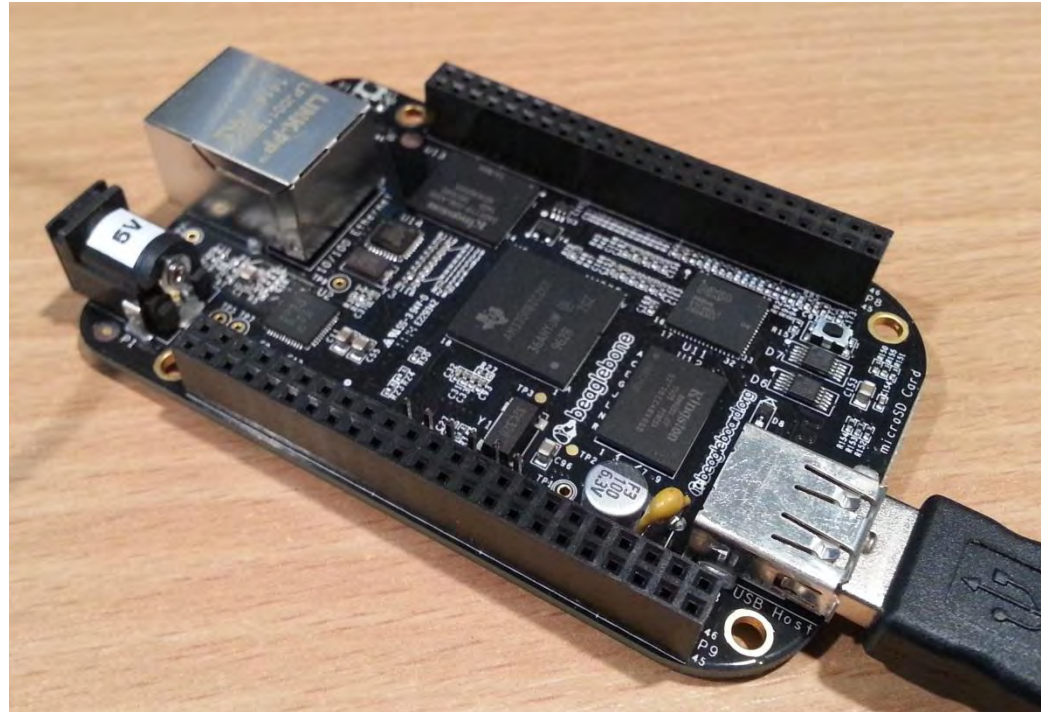
Daisho



<https://github.com/mossmann/daisho>

Open source, intended to support full USB 3.0 SuperSpeed monitoring and injection, but still in development

BeagleBone Black + USBProxy



<https://github.com/dominicgs/USBProxy>

Open source project to create a USB 2.0 MitM device using the BeagleBone Black, still in early stages, but already useful

Peach Publishers

- Data Publisher
- Configuration Publisher
- This uses libUsbDotNet which hasn't been maintained in a while, so it has its bugs, but has been useful for finding issues.
- Submitted to Peach maintainers as well as available on the DEFCON CD.

libusb

- <http://www.libusb.org/>
- Libusb is a good way to get started with writing tools to access USB devices

Demos!

Summary

- USB is a pervasive technology in modern computing devices.
- Not just external ports which require physical access to attack
- Devices connected over USB run upgradable firmware
- Debug capability required for Windows certification
- Interesting attack scenarios with internal devices

USB provides a rich set of capabilities and is being used in a variety of configurations that could result in security vulnerabilities and it's easier than people think to get started looking at this stuff.

Questions?