RSAConference2019

San Francisco | March 4 – 8 | Moscone Center

BETTER.

SESSION ID: HTA-R03

Malicious, Misbehaving or Misunderstood Making Bad USBs Good Again

Eric D Knapp

Fellow & CPO, Connected Cyber
Honeywell Connected Enterprise
@EricDKnapp

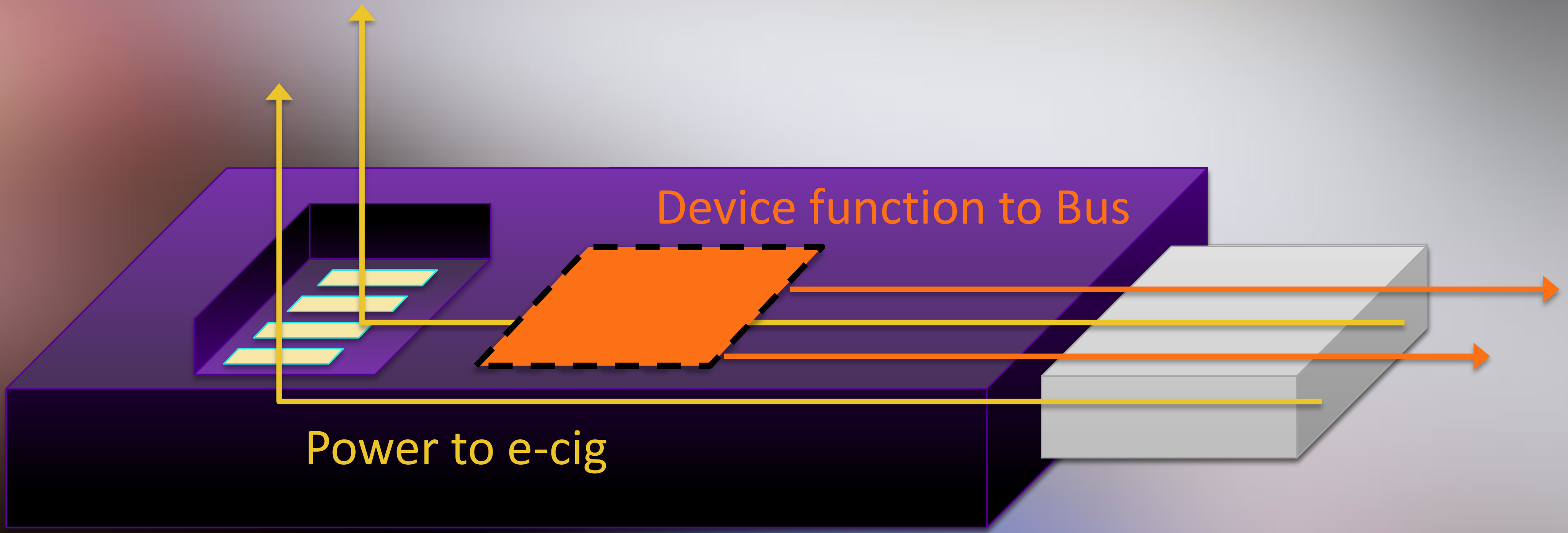
Peter Viscarola

Consulting Partner
OSR Open Systems Resources, Inc.
@osrdrivers

what just happened …?

“The Vape-inator”

what just happened …?



“The Vape-inator”

Why did that just happen …? (take-aways from this session)

- A broader understanding of the USB protocol as a threat vector
  - Think beyond malware and data theft
- (Some) knowledge of and respect for the USB protocol
  - 564 page standard
- A desire to re-evaluate USB security within your own organization
  - The most ubiquitous data flow in your infrastructure ***more prevalent than Ethernet***

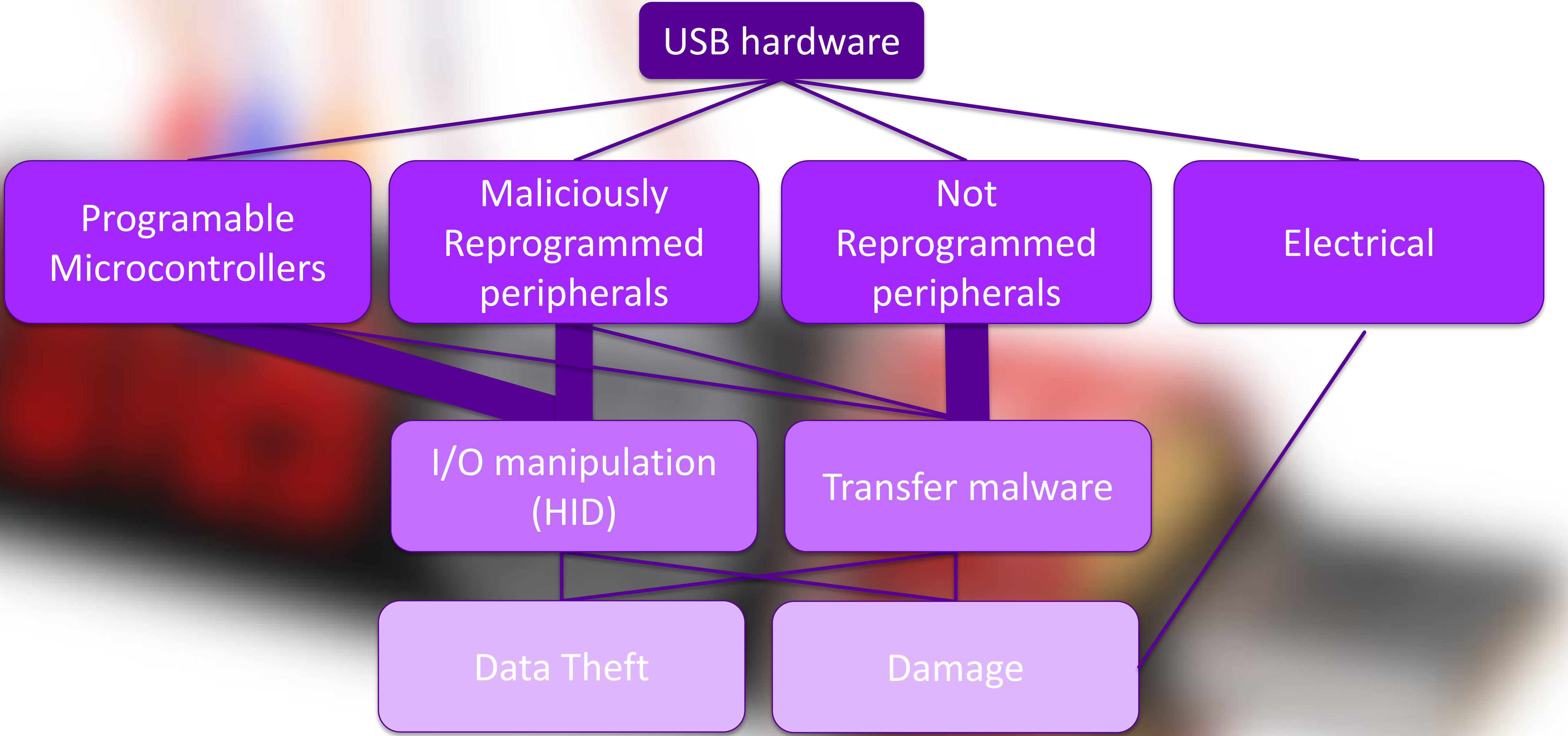
Universal Serial Bus (USB) is a powerful protocol that has revolutionized computing. Reap it's benefits while minimizing risk.

Viva la revolucion!

RSAConference2019

USB Threat Types

(more than malware)


USB hardware
Programable Microcontrollers
Maliciously Reprogrammed peripherals
Not Reprogrammed peripherals
Electrical
I/O manipulation (HID)
Transfer malware
Data Theft
Damage

Derived from “USB Based Attacks” by Nir Nissim, Ran Yahalom, and Yuval Elovici; Malware lab, Cyber Security Research Center, Ben-Gurion University of the Negev

Programable Microcontrollers

1 Rubber Ducky - 2010

2 PHUKD/URFUKED - 2010

3 USBdriveby - 2014

4 Evilduino - 2014

5 Unintended USB channels - 2011

6 TURNIPSCHOOL (COTTONMOUTH-1) - 2015

7 RIT attack via USB mass storage - 2012

8 Attacks on wireless USB dongles - 2015

9 Default gateway override - 2014

30 USB Harpoon

31 VAPE-inator

32 … What's next?

Maliciously Reprogrammed peripherals

10 Smartphone based HID attacks - 2010

11 DNS override by modified USB firmware - 2014

12 Keyboard emulation by modified USB firmware - 2014

13 Hidden partition patch - 2014 ☣

14 Password protection bypass patch - 2014 ☣

15 Virtual machine break-out - 2014

16 Boot sector virus - 2014 ☣

17 *iSeeYou*: Disabling the MacBook webcam indicator LED - 2014

Not Reprogrammed peripherals

18 .LNK Stuxnet/Fanny USB flash drive exploit (sh[illegible] extension exploits) - 2010 ☣

19 USB Backdoor into air-gapped hosts - 2014

20 Data hiding on USB mass storage - 2010 ☣

21 Autorun exploits - 2005 ☣

22 Cold boot - 2008

23 Buffer overflow - 2005 ☣

24 Driver update - 2011 ☣

25 Device firmware upgrade (DFU) - 2014

26 USB Thief - 2016 ☣

27 Attacks on smartphones via the USB port - 2010

28 USBee attack 2016

Electrical

29 USB Killer 2015

Source: "USB Based Attacks" by Nir Nissim, Ran Yahalom, and Yuval Elovici; Malware lab, Cyber Security Research Center, Ben-Gurion University of the Negev

What USB Threats Really Look Like

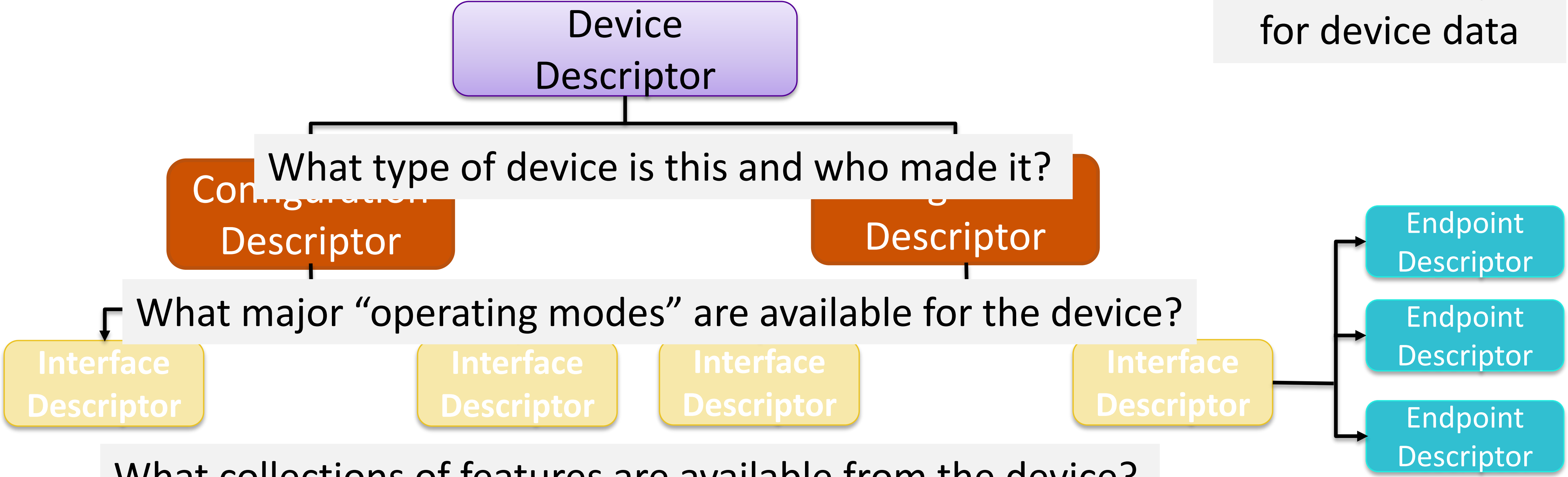
Universal Serial Bus

(the “S” doesn’t stand for “Simple”)

Identifying USB Devices

- During connection, each USB device identifies itself by sending a series of **descriptors** to the host

The Sources/Syncs for device data

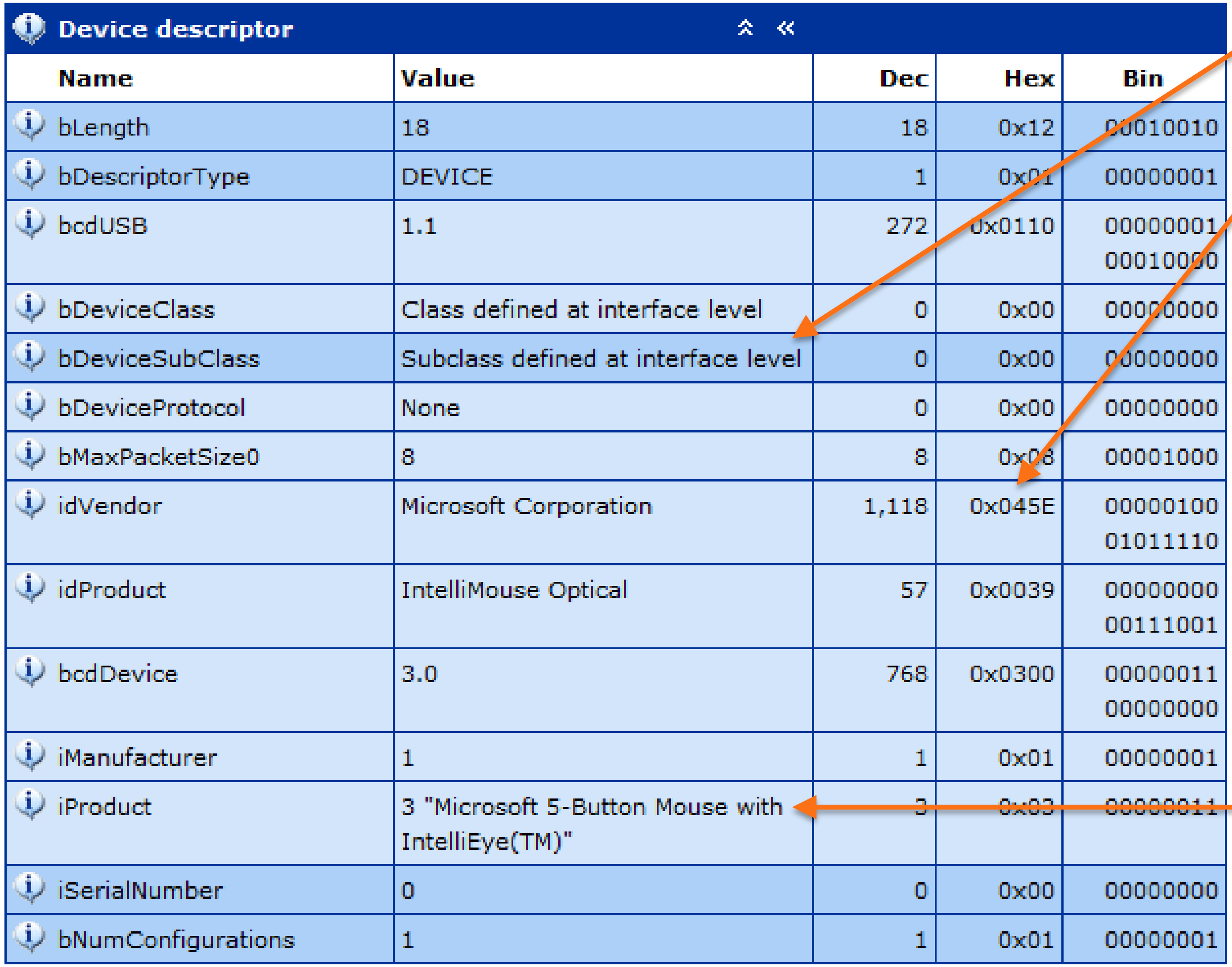


What type of device is this and who made it?

What major “operating modes” are available for the device?

What collections of features are available from the device?

This Was Designed to be “Simple”



Device descriptor

Name	Value	Dec	Hex	Bin
bLength	18	18	0x12	00010010
bDescriptorType	DEVICE	1	0x01	00000001
bcdUSB	1.1	272	0x0110	00000001 00010000
bDeviceClass	Class defined at interface level	0	0x00	00000000
bDeviceSubClass	Subclass defined at interface level	0	0x00	00000000
bDeviceProtocol	None	0	0x00	00000000
bMaxPacketSize0	8	8	0x08	00001000
idVendor	Microsoft Corporation	1,118	0x045E	00000100 01011110
idProduct	IntelliMouse Optical	57	0x0039	00000000 00111001
bcdDevice	3.0	768	0x0300	00000011 00000000
iManufacturer	1	1	0x01	00000001
iProduct	3 "Microsoft 5-Button Mouse with IntelliEye(TM)"	3	0x03	00000011
iSerialNumber	0	0	0x00	00000000
bNumConfigurations	1	1	0x01	00000001

No specific device type info here

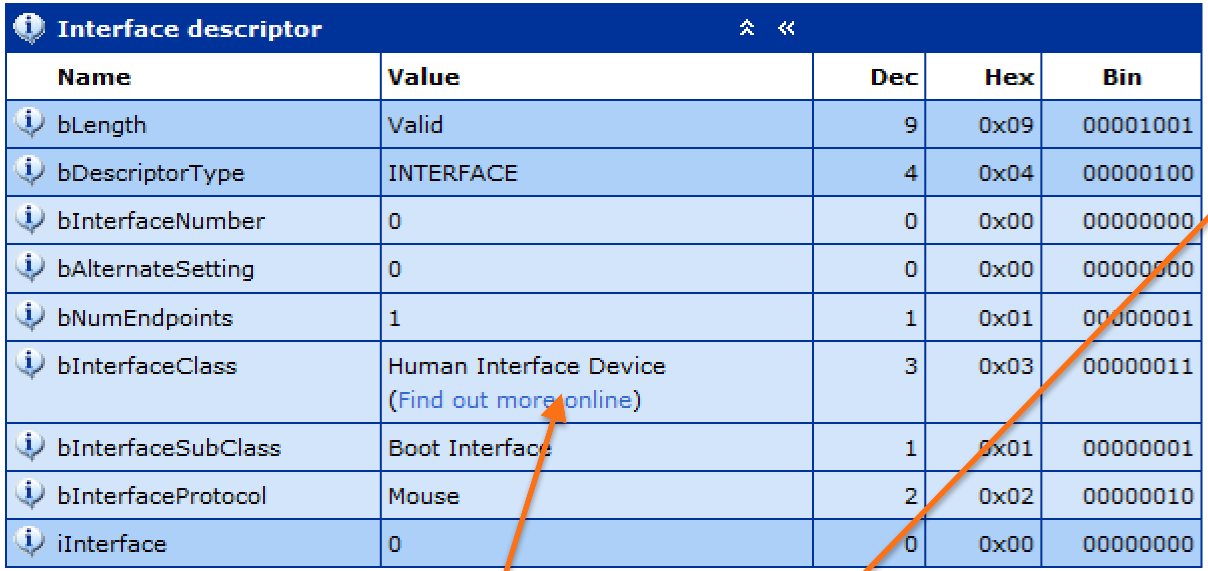
Vendor ID = 0x045E = Microsoft

What could *possibly* go wrong?

Remember… This scheme dates to USB V1.0
Which was released in January 1996

Product Name is
“Microsoft 5-Button Mouse …”

Even Drilling Down Into Lower Levels … Question Remain



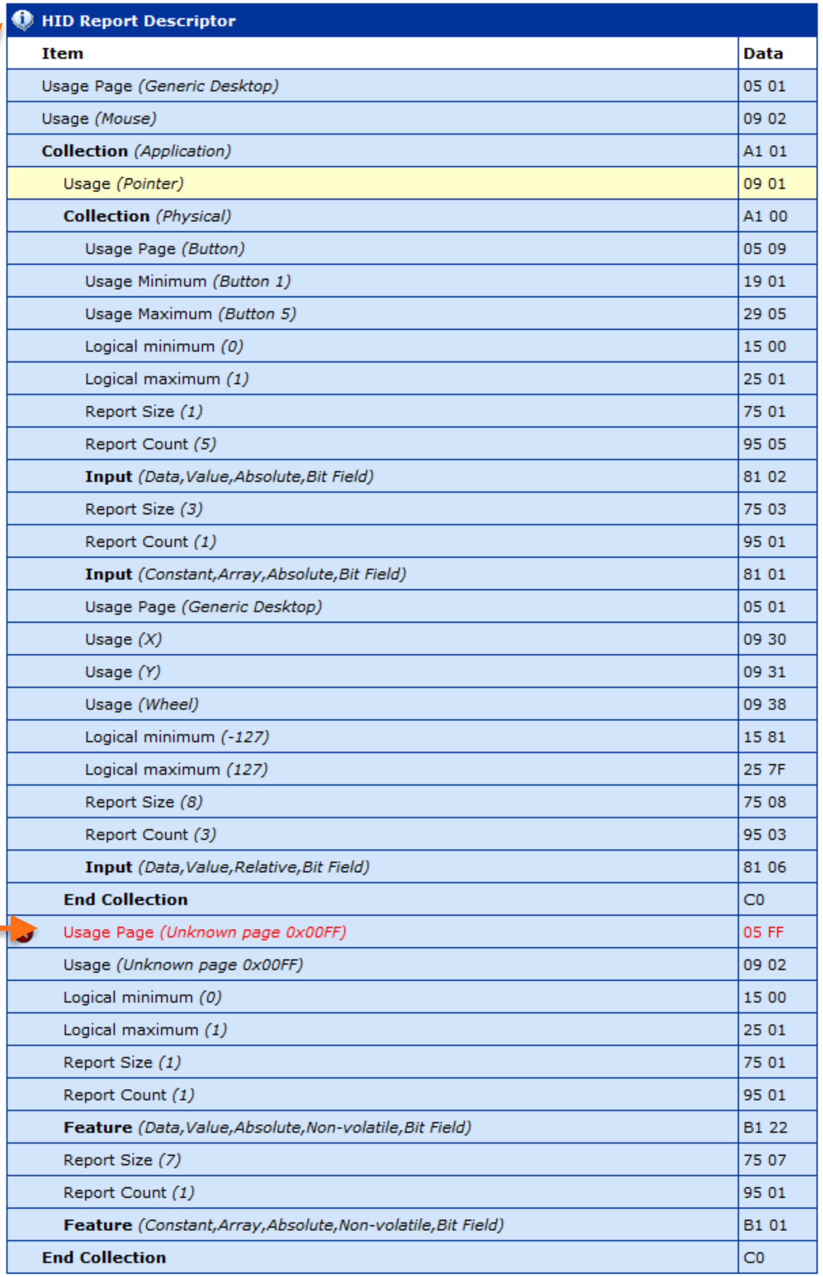
Interface descriptor

Name	Value	Dec	Hex	Bin
bLength	Valid	9	0x09	00001001
bDescriptorType	INTERFACE	4	0x04	00000100
bInterfaceNumber	0	0	0x00	00000000
bAlternateSetting	0	0	0x00	00000000
bNumEndpoints	1	1	0x01	00000001
bInterfaceClass	Human Interface Device (Find out more online)	3	0x03	00000011
bInterfaceSubClass	Boot Interface	1	0x01	00000001
bInterfaceProtocol	Mouse	2	0x02	00000010
iInterface	0	0	0x00	00000000

Class is “Human Interface Device”

Further described by HID Descriptor

Wonder what ***this*** *is?*

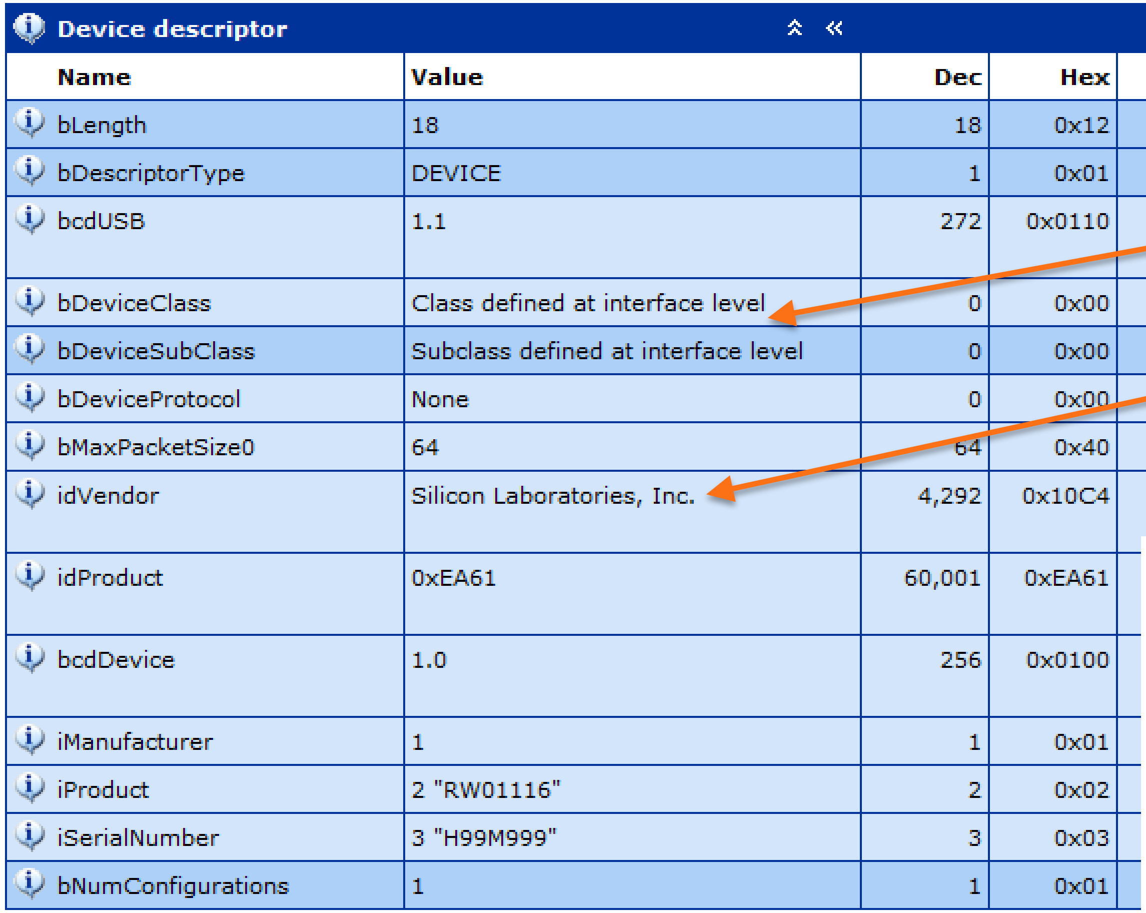


HID Report Descriptor

Item	Data
Usage Page *(Generic Desktop)*	05 01
Usage *(Mouse)*	09 02
Collection *(Application)*	A1 01
Usage *(Pointer)*	09 01
Collection *(Physical)*	A1 00
Usage Page *(Button)*	05 09
Usage Minimum *(Button 1)*	19 01
Usage Maximum *(Button 5)*	29 05
Logical minimum *(0)*	15 00
Logical maximum *(1)*	25 01
Report Size *(1)*	75 01
Report Count *(5)*	95 05
Input *(Data,Value,Absolute,Bit Field)*	81 02
Report Size *(3)*	75 03
Report Count *(1)*	95 01
Input *(Constant,Array,Absolute,Bit Field)*	81 01
Usage Page *(Generic Desktop)*	05 01
Usage *(X)*	09 30
Usage *(Y)*	09 31
Usage *(Wheel)*	09 38
Logical minimum *(-127)*	15 81
Logical maximum *(127)*	25 7F
Report Size *(8)*	75 08
Report Count *(3)*	95 03
Input *(Data,Value,Relative,Bit Field)*	81 06
End Collection	C0
Usage Page *(Unknown page 0x00FF)*	05 FF
Usage *(Unknown page 0x00FF)*	09 02
Logical minimum *(0)*	15 00
Logical maximum *(1)*	25 01
Report Size *(1)*	75 01
Report Count *(1)*	95 01
Feature *(Data,Value,Absolute,Non-volatile,Bit Field)*	B1 22
Report Size *(7)*	75 07
Report Count *(1)*	95 01
Feature *(Constant,Array,Absolute,Non-volatile,Bit Field)*	B1 01
End Collection	C0

There are plenty of devices like this one

Should we trust this device?

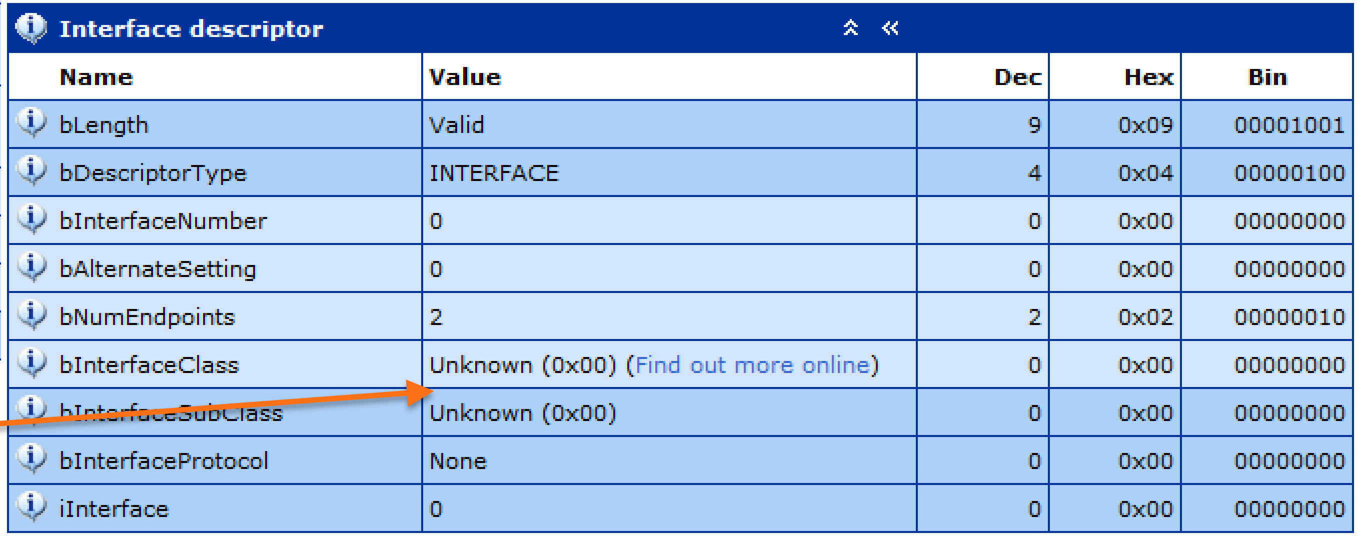


Device descriptor

Name	Value	Dec	Hex	Bin
bLength	18	18	0x12	00010010
bDescriptorType	DEVICE	1	0x01	00000001
bcdUSB	1.1	272	0x0110	00000001 00010000
bDeviceClass	Class defined at interface level	0	0x00	00000000
bDeviceSubClass	Subclass defined at interface level	0	0x00	00000000
bDeviceProtocol	None	0	0x00	00000000
bMaxPacketSize0	64	64	0x40	01000000
idVendor	Silicon Laboratories, Inc.	4,292	0x10C4	00010000 11000100
idProduct	0xEA61	60,001	0xEA61	
bcdDevice	1.0	256	0x0100	
iManufacturer	1	1	0x01	
iProduct	2 "RW01116"	2	0x02	
iSerialNumber	3 "H99M999"	3	0x03	
bNumConfigurations	1	1	0x01	

No specific device type info here

Vendor ID = Generic silicon mfg



Interface descriptor

Name	Value	Dec	Hex	Bin
bLength	Valid	9	0x09	00001001
bDescriptorType	INTERFACE	4	0x04	00000100
bInterfaceNumber	0	0	0x00	00000000
bAlternateSetting	0	0	0x00	00000000
bNumEndpoints	2	2	0x02	00000010
bInterfaceClass	Unknown (0x00) (Find out more online)	0	0x00	00000000
bInterfaceSubClass	Unknown (0x00)	0	0x00	00000000
bInterfaceProtocol	None	0	0x00	00000000
iInterface	0	0	0x00	00000000

No more specific info available

Honeywell OSR

So… What Does This Tell Us?

- The USB spec for device identification was initially created in a "simpler time"
- The device is entirely responsible for presenting its descriptor information to the OS at runtime, with no other validation/checks
- Many devices are insufficiently transparent in their descriptions
- The hierarchy of descriptors has grown and gotten more complex… The OS has a big job, full of heuristics developed over time, to determine what a device is (and what it will do once connected)

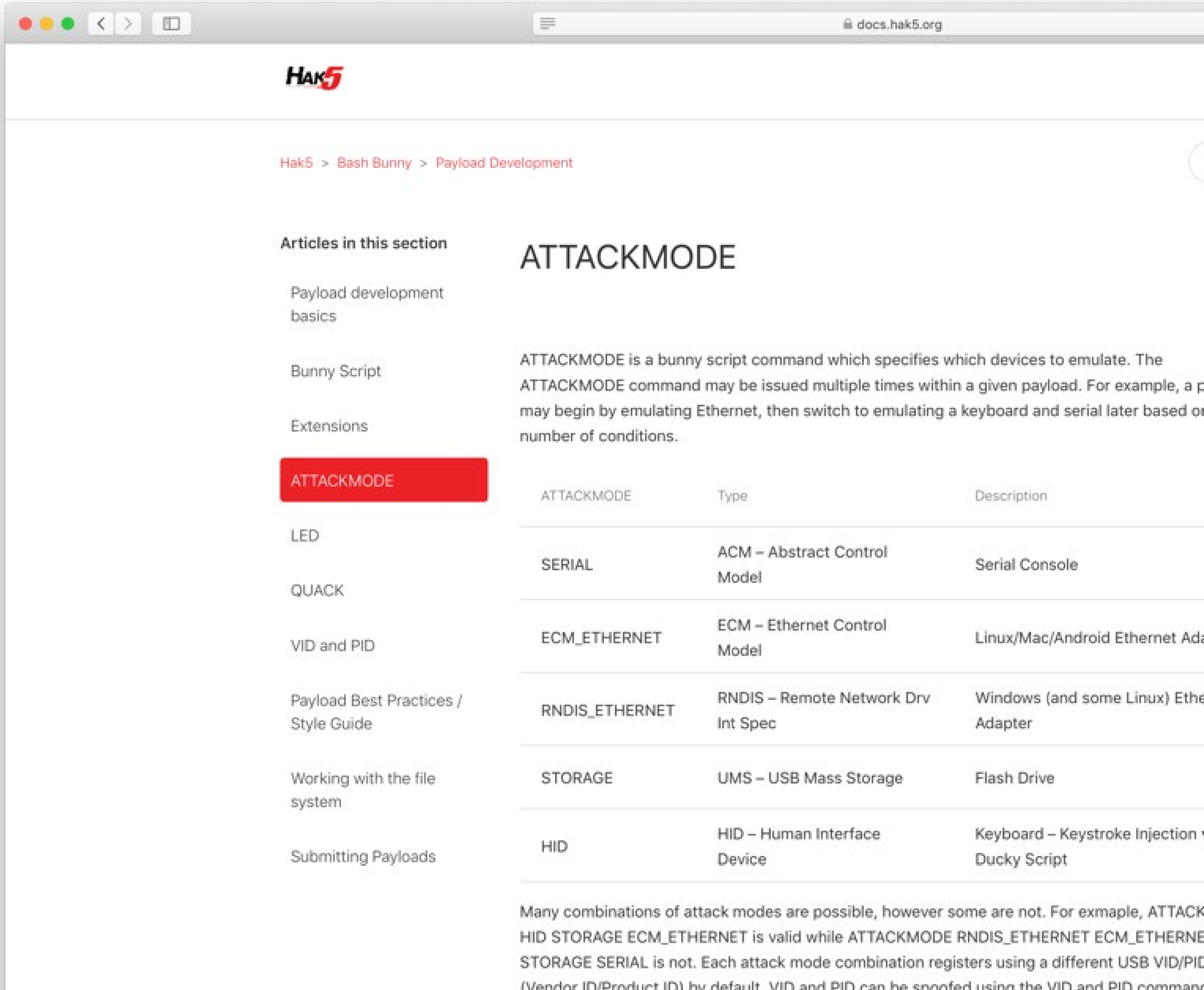
RSAConference2019

Hacking USB

(and why it works)

how to manipulate the standard

- Lie about what you are



docs.hak5.org

HAK5

Hak5 > Bash Bunny > Payload Development

Articles in this section

- Payload development basics
- Bunny Script
- Extensions
- ATTACKMODE
- LED
- QUACK
- VID and PID
- Payload Best Practices / Style Guide
- Working with the file system
- Submitting Payloads

ATTACKMODE

ATTACKMODE is a bunny script command which specifies which devices to emulate. The ATTACKMODE command may be issued multiple times within a given payload. For example, a p may begin by emulating Ethernet, then switch to emulating a keyboard and serial later based o number of conditions.

ATTACKMODE	Type	Description
SERIAL	ACM – Abstract Control Model	Serial Console
ECM_ETHERNET	ECM – Ethernet Control Model	Linux/Mac/Android Ethernet Ada
RNDIS_ETHERNET	RNDIS – Remote Network Drv Int Spec	Windows (and some Linux) Ethe Adapter
STORAGE	UMS – USB Mass Storage	Flash Drive
HID	HID – Human Interface Device	Keyboard – Keystroke Injection Ducky Script

Many combinations of attack modes are possible, however some are not. For exmaple, ATTACK HID STORAGE ECM_ETHERNET is valid while ATTACKMODE RNDIS_ETHERNET ECM_ETHERNE STORAGE SERIAL is not. Each attack mode combination registers using a different USB VID/PID (Vendor ID/Product ID) by default. VID and PID can be spoofed using the VID and PID comman

how to manipulate the standard

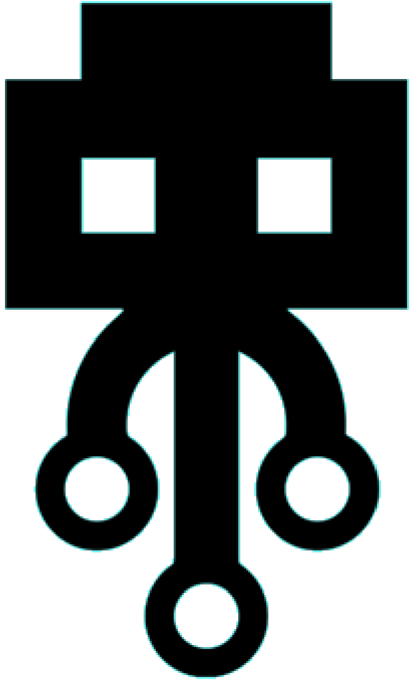
- Have multiple personalities

how to manipulate the standard

- Be something tempting and unknown

how to manipulate the standard

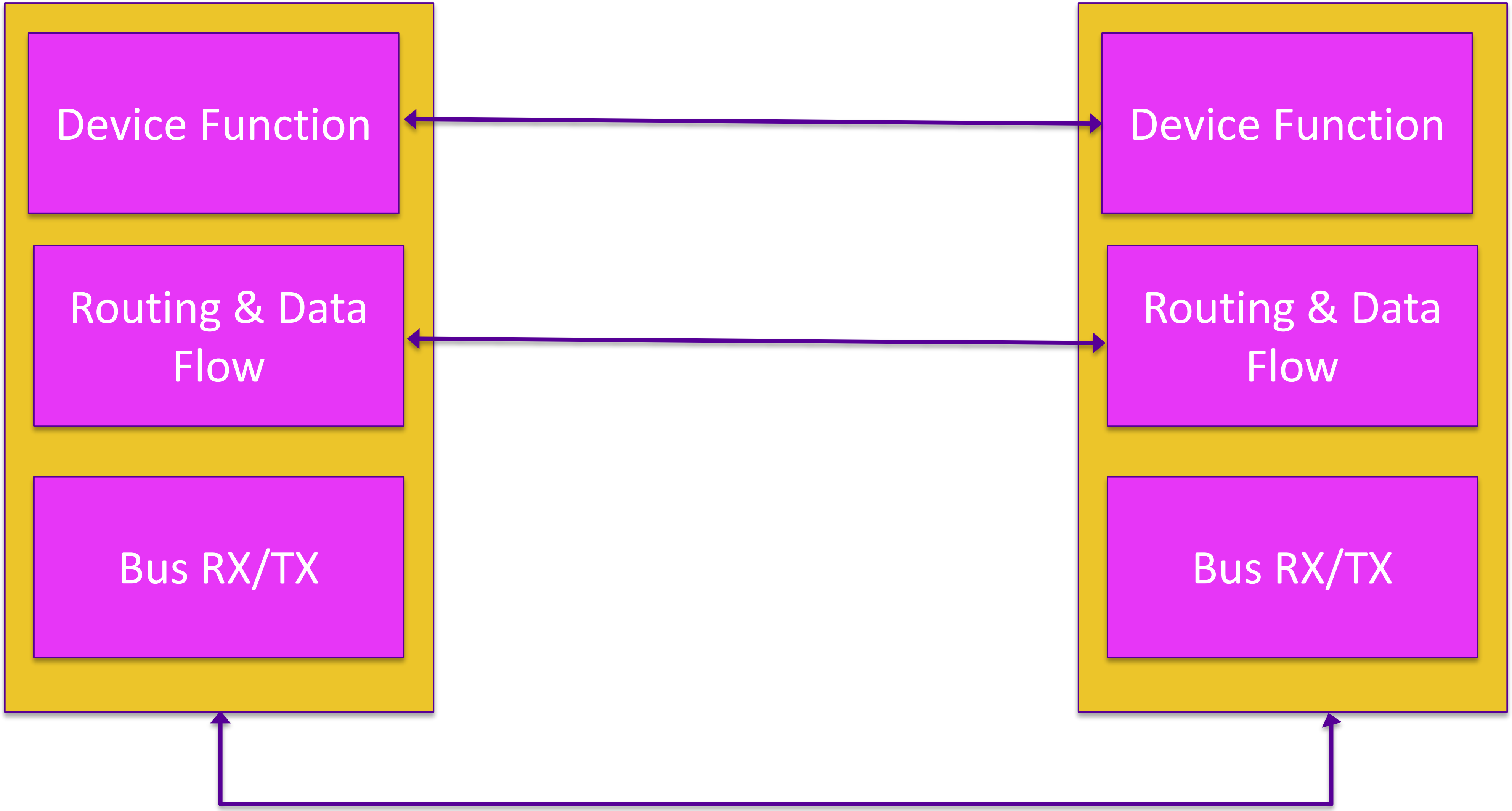
- Be electric



how to bypass the standard

- Be a vampire troll

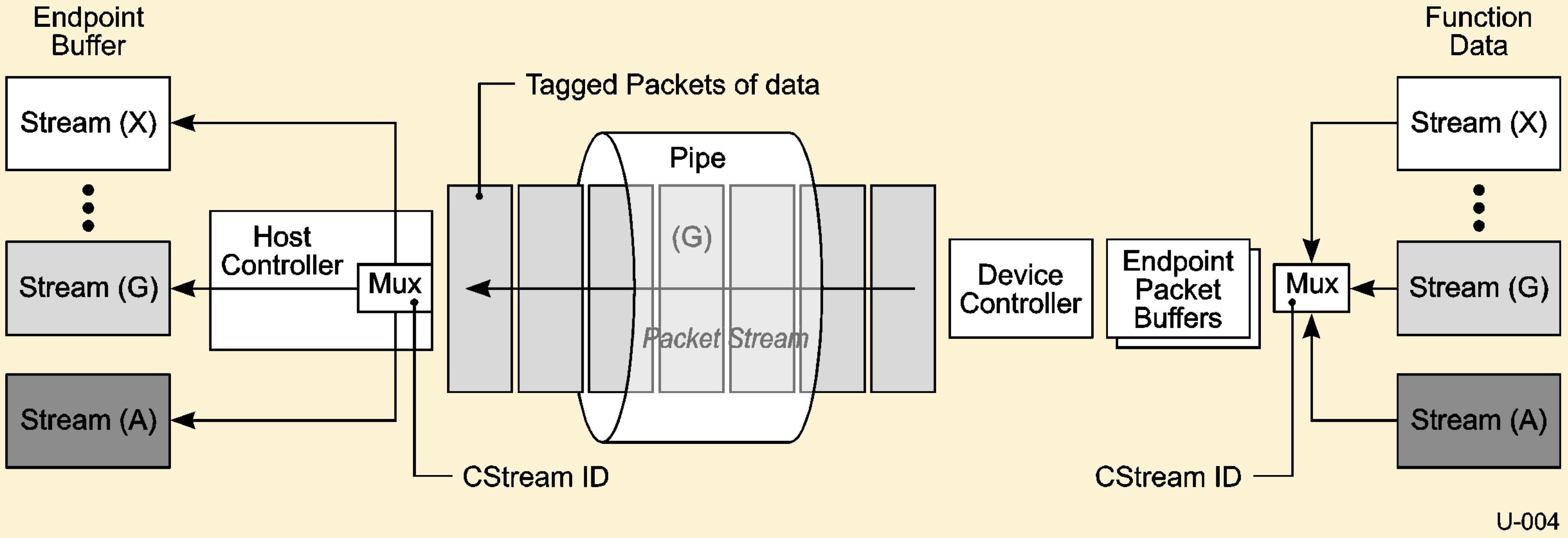
just the tip of the iceberg …



Source: USB.org, Universal Serial Bus 3.2 Specification

just the tip of the iceberg …

Figure 4-3. Enhanced SuperSpeed IN Stream Example



Source: USB.org, Universal Serial Bus 3.2 Specification

just the tip of the iceberg …

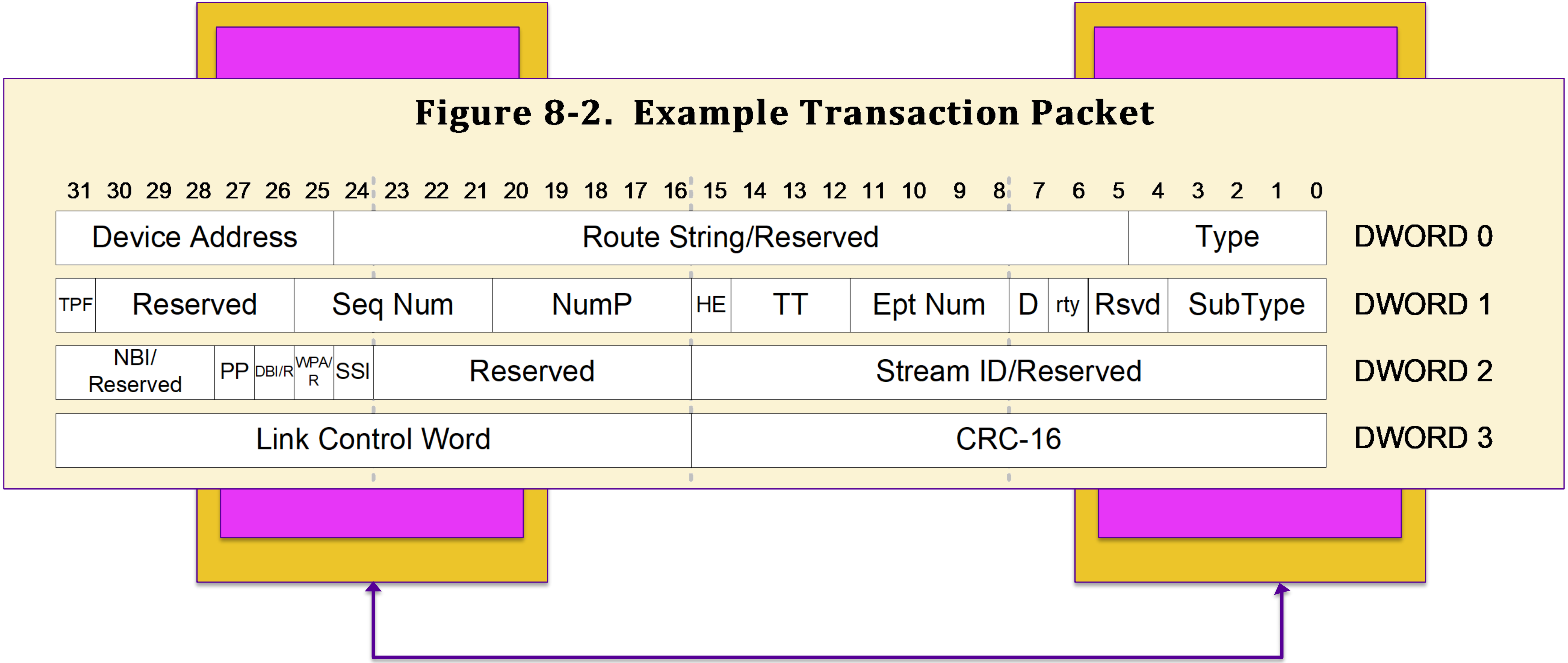


Figure 8-2. Example Transaction Packet

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Device Address							Route String/Reserved																				Type					DWORD 0
TPF	Reserved					Seq Num					NumP					HE	TT			Ept Num				D	rty	Rsvd		SubType				DWORD 1
NBI/Reserved				PP	DBI/R	WPA/R	SSI	Reserved								Stream ID/Reserved																DWORD 2
Link Control Word																CRC-16																DWORD 3

Source: USB.org, Universal Serial Bus 3.2 Specification

RSAConference2019

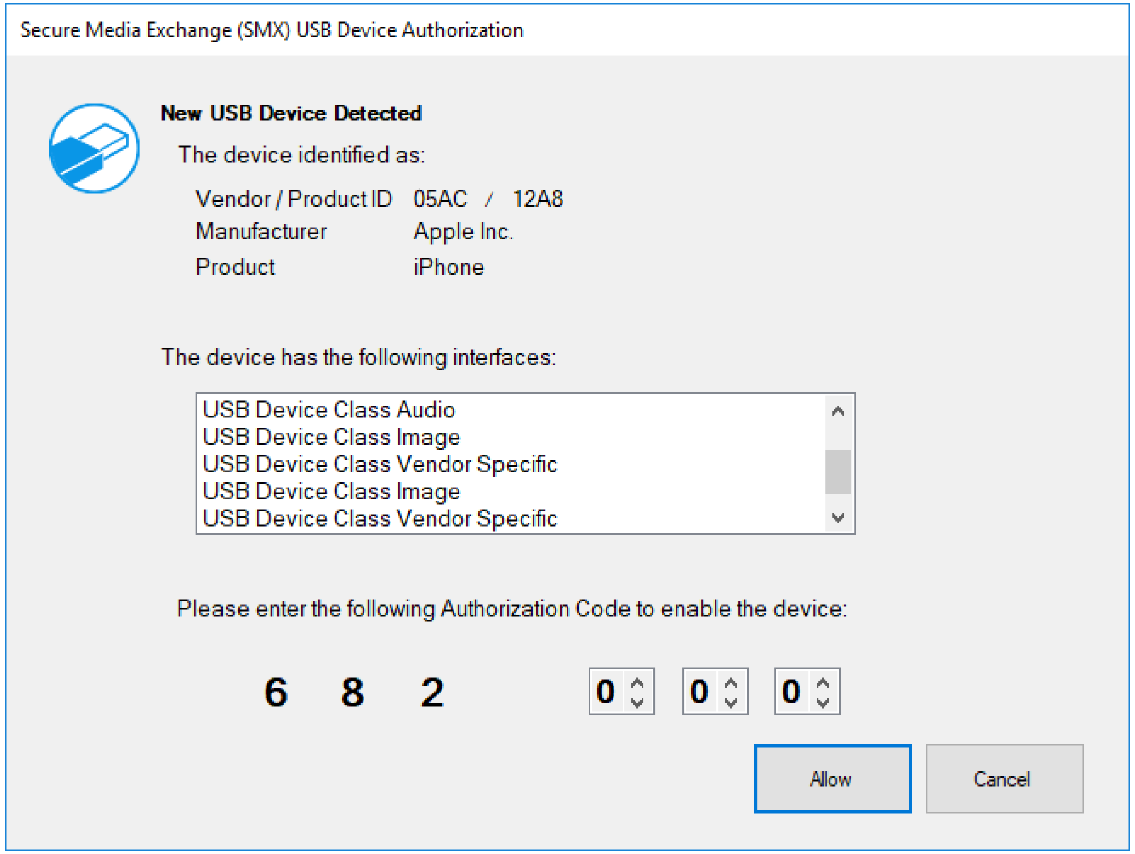
Don’t Worry USB Happy

(there’s some good news, too)

There’s Some Good News about Bad USB

- A way to TRUST your USB devices
- New Authorization Specifications

TRUST (Trusted User Substantiation Technology)



Secure Media Exchange (SMX) USB Device Authorization

New USB Device Detected

The device identified as:

Vendor / Product ID	05AC / 12A8
Manufacturer	Apple Inc.
Product	iPhone

The device has the following interfaces:

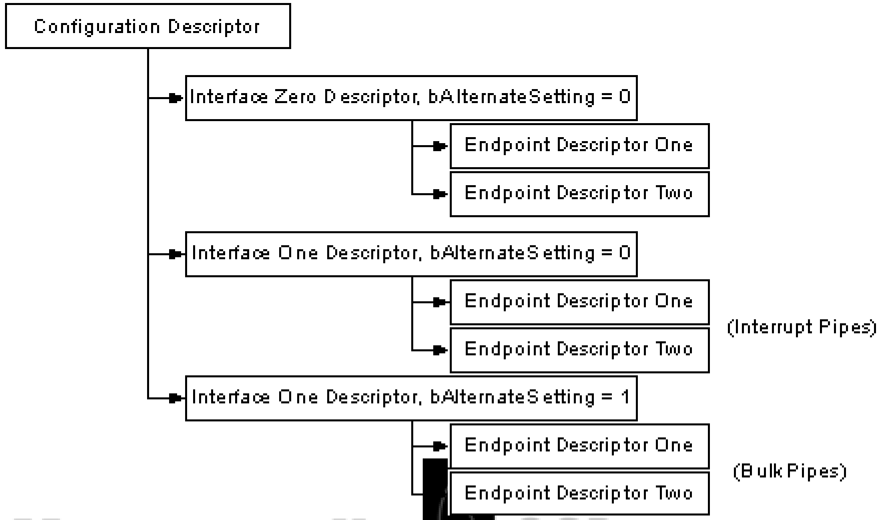
USB Device Class Audio
USB Device Class Image
USB Device Class Vendor Specific
USB Device Class Image
USB Device Class Vendor Specific

Please enter the following Authorization Code to enable the device:

6 8 2 0 0 0

TRUST Original Design Precepts

- Primary Design Principles:
  - Block device at the lowest level
  - Get information from the device, and make policy decision based solely on that info
  - Everything needs to be Consciously Authorized



Configuration Descriptor

Interface Zero Descriptor, bAlternateSetting = 0

Endpoint Descriptor One

Endpoint Descriptor Two

Interface One Descriptor, bAlternateSetting = 0

Endpoint Descriptor One

Endpoint Descriptor Two

(Interrupt Pipes)

Interface One Descriptor, bAlternateSetting = 1

Endpoint Descriptor One

Endpoint Descriptor Two

(Bulk Pipes)

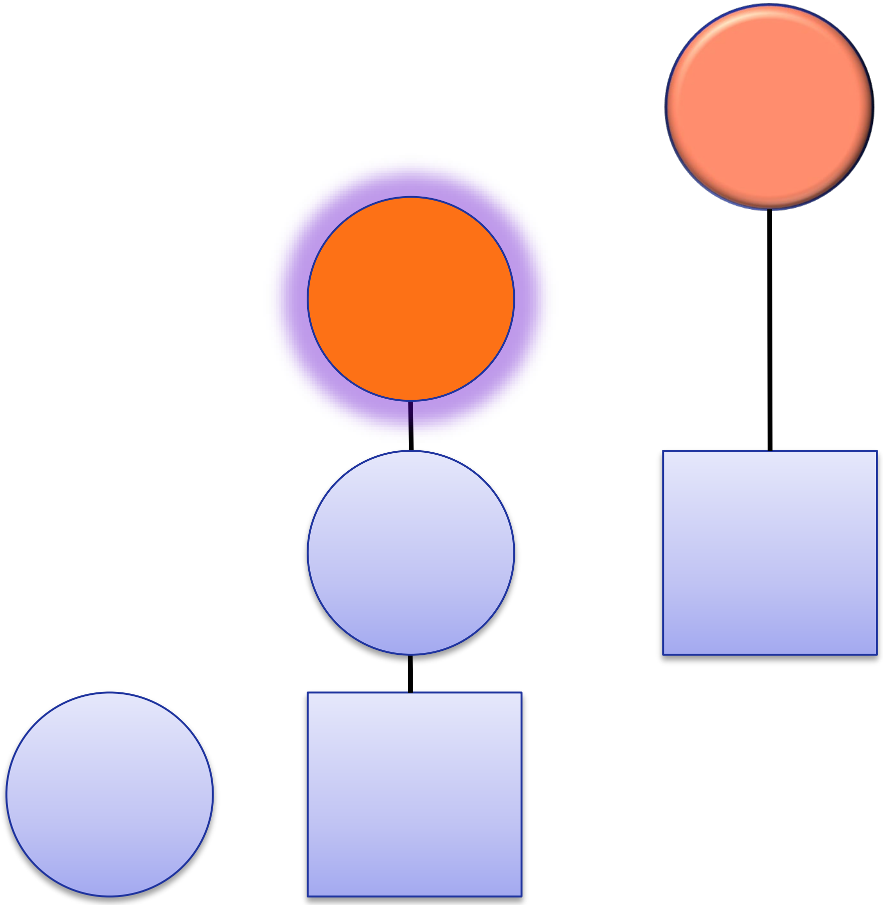
Blocking USB Devices at the Lowest Level

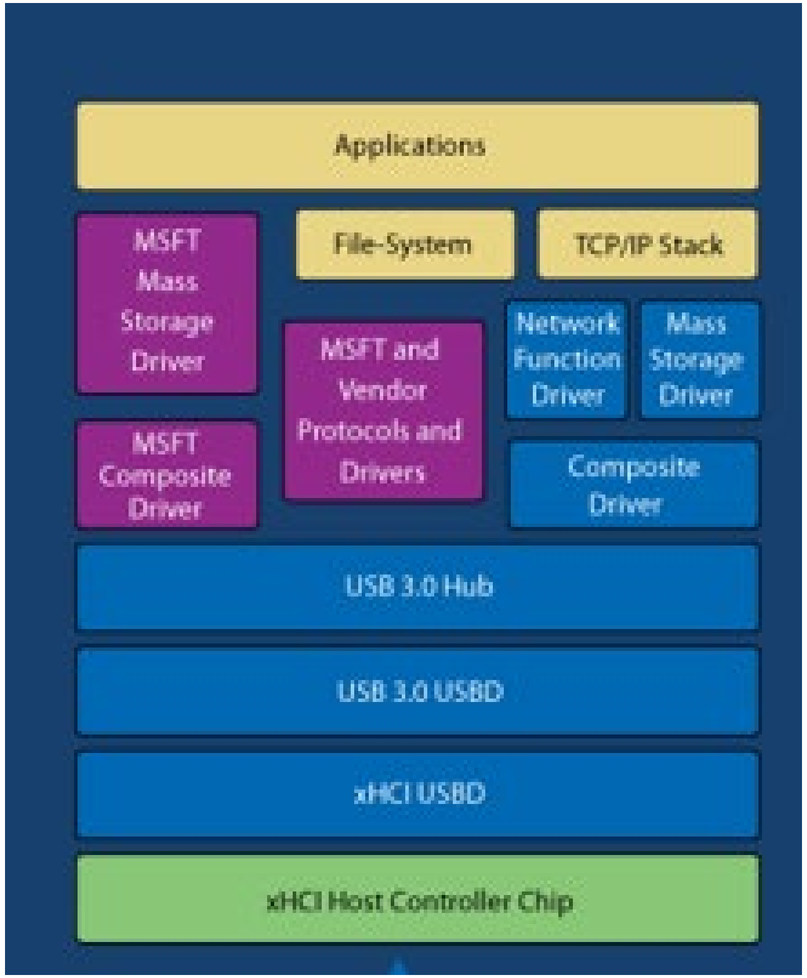
GoodUSB
Authorization

GoodUSB
Filter

USB Hub

USB
Controller





Applications

MSFT Mass Storage Driver

File-System

TCP/IP Stack

MSFT and Vendor Protocols and Drivers

Network Function Driver

Mass Storage Driver

MSFT Composite Driver

Composite Driver

USB 3.0 Hub

USB 3.0 USBD

xHCI USBD

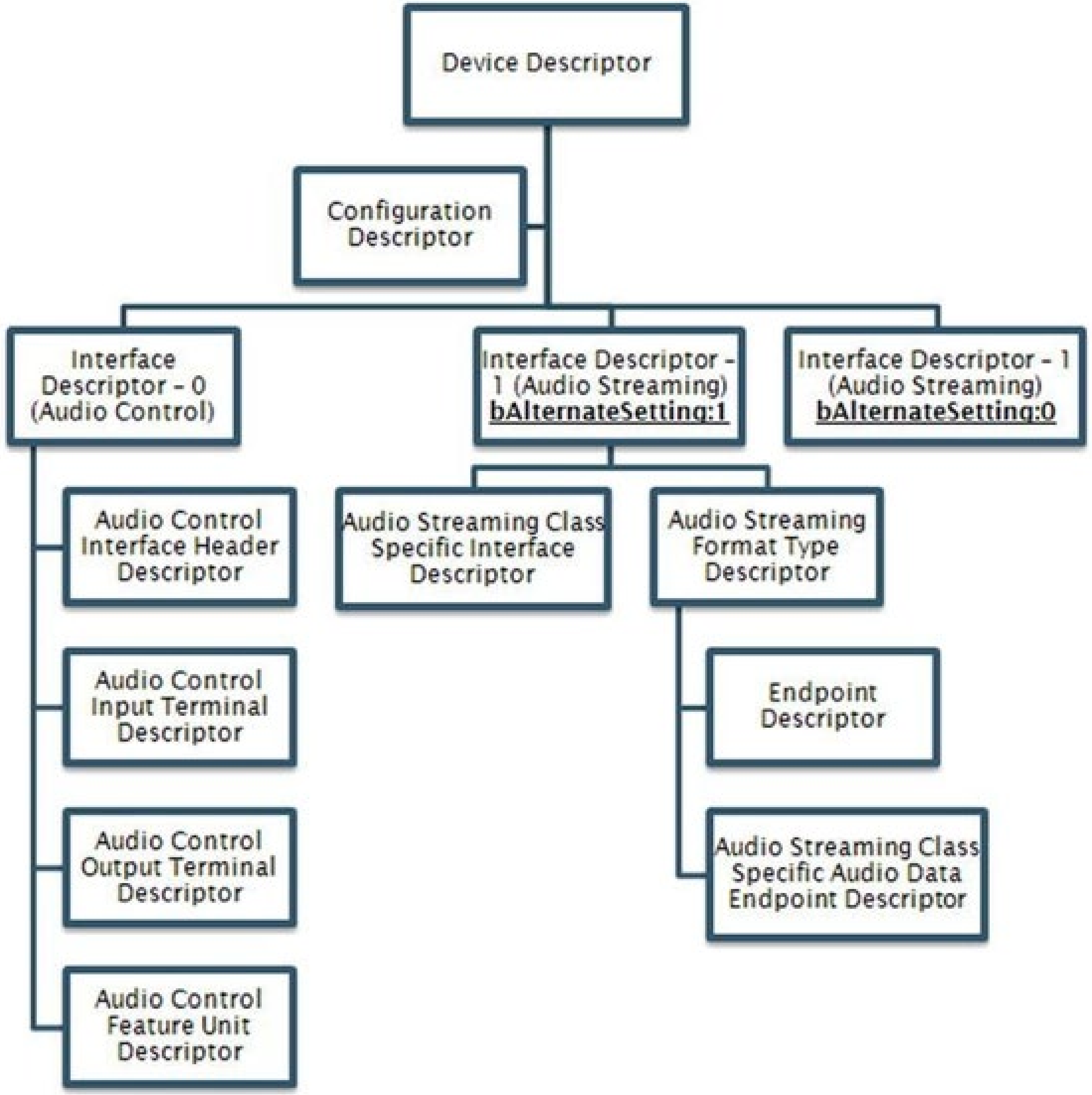
xHCI Host Controller Chip

What We Learned (Part 1)

- Relying solely on the USB Device information is not good enough
- What's definitive is not what the USB Device says it is…
  it's how the OS treats the device
  - OS decision process is complex, taking into account many factors
  - The driver the OS chooses may be "OS Standard" or "Vendor Specific"… makes all the difference

What We Learned (Part 2)

- Presenting the user info solely from the USB Device isn't very helpful

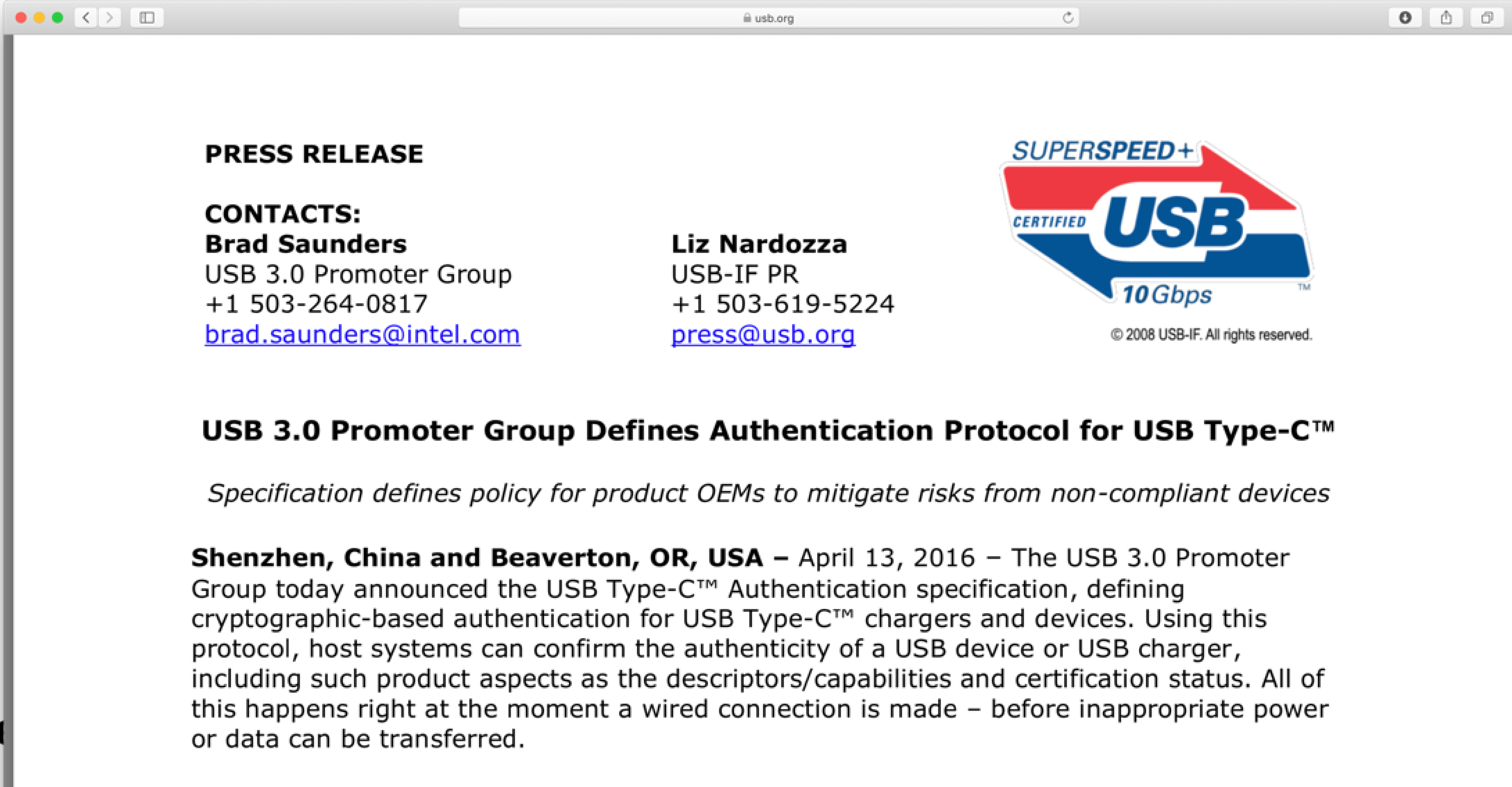

Device Descriptor
Configuration Descriptor
Interface Descriptor - 0 (Audio Control)
Interface Descriptor - 1 (Audio Streaming) bAlternateSetting:1
Interface Descriptor - 1 (Audio Streaming) bAlternateSetting:0
Audio Control Interface Header Descriptor
Audio Control Input Terminal Descriptor
Audio Control Output Terminal Descriptor
Audio Control Feature Unit Descriptor
Audio Streaming Class Specific Interface Descriptor
Audio Streaming Format Type Descriptor
Endpoint Descriptor
Audio Streaming Class Specific Audio Data Endpoint Descriptor

what we learned (3)

- **Ask Windows** how it will treat the device once it’s connected
  - Get information from the device
  - Use OS-provided SetupDiXxxx API to determine the INF/Driver that Windows would use
- For well-know classes where information from Windows isn’t sufficient (e.g. HID and Mass Storage), **mimic higher-level Windows processing** to determine data on device use
- When Conscious Authorization is needed, **present clear info** from the Device the OS, and our device device use data

“Hey Windows… What’ll you do with this device??”

USB-C authorization



PRESS RELEASE

CONTACTS:

Brad Saunders
USB 3.0 Promoter Group
+1 503-264-0817
brad.saunders@intel.com

Liz Nardozza
USB-IF PR
+1 503-619-5224
press@usb.org

USB 3.0 Promoter Group Defines Authentication Protocol for USB Type-C™

Specification defines policy for product OEMs to mitigate risks from non-compliant devices

Shenzhen, China and Beaverton, OR, USA – April 13, 2016 – The USB 3.0 Promoter Group today announced the USB Type-C™ Authentication specification, defining cryptographic-based authentication for USB Type-C™ chargers and devices. Using this protocol, host systems can confirm the authenticity of a USB device or USB charger, including such product aspects as the descriptors/capabilities and certification status. All of this happens right at the moment a wired connection is made – before inappropriate power or data can be transferred.

Apply What You Have Learned Today

- Next week you should:
  - Assess existing USB defensive measures, considering all 3 attack types
- In the first three months following this presentation you should:
  - Complete an inventory of USB devices currently in use: what role do these devices play in the daily operations of your business?
  - Assess your supply chain: what USB devices are you using? Are they trusted?
- Within six months you should:
  - Adjust USB and removable media policies to account for your findings.
  - Consider technical controls to enforce these policies

Special thanks to:

- Honeywell Connected Cyber research teams
- The Honeywell legal and media teams

  (for keeping an open mind about security presentations like this)
- The valuable research of:
  - Karsten Nohl and Jakob Lell (BadUSB)
  - @SamyKamkar (PosionTap)
  - @hak5darren and all at Hak5 (Rubber Duckies, Bash Bunnies & more)
  - Everyone who helped put the Vape-inator together
- Our partners at Open Systems Resources

Honeywell OSR

RSAConference2019

Thank You (SB)

@EricDKnapp @osrdrivers