Intel ME: The Way of the Static Analysis
Intel ME
Known Facts
ME Position in Computer System

Intel AMT Release 2.0/2.1/2.2

Architecture

- CPU
  - SW Agents
  - OS

- GMCH
  - Intel® ME
  - Slot 1
  - DDR2
  - Slot 0
  - DDR2
  - RAM

- ICH8
  - Filters
  - Sensors
  - MAC
  - Intel® 82566DM Gigabit Network Connection
  - OOB
  - PHY

- Full control

- Limited interfaces

- User
- OS Kernel
- Hypervisor
- System Management Mode (SMM)
- Management Engine (ME)
Sources of Information

- Platform Embedded Security Technology Revealed
  - Safeguarding the Future of Computing with Intel Embedded Security and Management Engine
  - By Xiaoyu Ruan

- Intel ME Secrets
  - Hidden code in your chipset and how to discover what exactly it does
  - By Igor Skochinsky
  - Hex-Rays
  - RECON 2014
  - Montreal

- Patents / White Papers / Documentation
### ME Platform Features

**TXE: Trusted Execution Engine (for Intel Atom CPUs)**

**SPS: Server Platform Services**

**CSE: Converged Security Engine (new name for Security+Management Engine?)**

<table>
<thead>
<tr>
<th>Version</th>
<th>CPU Arch</th>
<th>LZMA</th>
<th>Huffman</th>
<th>OS Arch</th>
</tr>
</thead>
<tbody>
<tr>
<td>ME 2.x-5.x</td>
<td>ARCtangent</td>
<td>+</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SPS 1.x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ME 6.x-10.x</td>
<td>ARCompact</td>
<td>+</td>
<td>+</td>
<td>ThreadX</td>
</tr>
<tr>
<td>SPS 2.x-3.x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXE 1.x-2x</td>
<td>SPARC</td>
<td>+</td>
<td>-</td>
<td>ThreadX</td>
</tr>
<tr>
<td>TXE 3.x</td>
<td>x86</td>
<td>+</td>
<td>-</td>
<td>?</td>
</tr>
<tr>
<td>SPS 4.x</td>
<td>x86</td>
<td>+</td>
<td>Tables unknown</td>
<td>?</td>
</tr>
<tr>
<td>CSE(ME) 11.x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Area of interest
Curious Questions

• What is actually included in the ME 11.x

• How parts of ME interact with each other

• How ME interact with outer world

• Is there any backdoors or vulnerabilities in ME
Mission

• Help others to start researches on ME

• Avoid Security-Through-Obscurity

• Find and eliminate ME vulnerabilities (if any;)

• Become the sole owner of the owned equipment

• Make the world a safer place
DMCA security research exemption for consumer devices

By: Aaron Alva | Oct 28, 2016 2:12PM

**TAGS:** Data security | Office of Technology Research and Investigation (OTRI) | Research

With the stroke of a pen, the Librarian of Congress has authorized security researchers who are acting in good faith to conduct controlled research on consumer devices so long as the research does not violate other laws such as the Computer Fraud and Abuse Act (CFAA). This temporary exemption to the Digital Millennium Copyright Act (DMCA) begins today. The new temporary exemption is a big win for security researchers and for consumers who will benefit from increased security testing of the products they use.
Brief look at ME v11.x Firmware Binaries
Tip: Start building your own collection today! ;}
## FPT: Flash Partition Table

<table>
<thead>
<tr>
<th>name</th>
<th>offset:size</th>
<th>type</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTPR</td>
<td>$1000:C9000$</td>
<td>Code</td>
<td>Main code partition</td>
</tr>
<tr>
<td>FTUP</td>
<td>$20C000:38B000$</td>
<td>Code</td>
<td>[NFTP]+[WCOD]+[LOCL]</td>
</tr>
<tr>
<td>DLMP</td>
<td>0:0</td>
<td>Code</td>
<td>IDLM partition</td>
</tr>
<tr>
<td>PSVN</td>
<td>$E00:200$</td>
<td>Data</td>
<td>Secure Version Number</td>
</tr>
<tr>
<td>IVBP</td>
<td>$208000:4000$</td>
<td>Data</td>
<td>IV + Bring Up cache</td>
</tr>
<tr>
<td>MFS</td>
<td>$CA000:13E000$</td>
<td>Data</td>
<td>ME Flash File System</td>
</tr>
<tr>
<td>NFTP</td>
<td>$20C000:308000$</td>
<td>Code</td>
<td>Additional code</td>
</tr>
<tr>
<td>ROMB</td>
<td>0:0</td>
<td>Code</td>
<td>ROM Bypass</td>
</tr>
<tr>
<td>WCOD</td>
<td>$514000:80000$</td>
<td>Code</td>
<td>WLAN uCode</td>
</tr>
<tr>
<td>LOCL</td>
<td>$594000:3000$</td>
<td>Code</td>
<td>AMT Localization</td>
</tr>
<tr>
<td>FLOG</td>
<td>$597000:1000$</td>
<td>Data</td>
<td>Flash Log</td>
</tr>
<tr>
<td>UTOK</td>
<td>$598000:2000$</td>
<td>Data</td>
<td>Debug Unlock Token</td>
</tr>
<tr>
<td>ISHC</td>
<td>$59A000:2B000$</td>
<td>Code</td>
<td>Integrated Sensors Hub</td>
</tr>
</tbody>
</table>

*Never seen**

**Holds most of the modules**
$CPD$: Code Partition Directory

- **Header (16 bytes)**
  - Marker “$CPD”
  - Number of LUT entries
  - Some versions/checksum info
  - Partition name [4 chars]

- **Look-Up Table (24 bytes per entry)**
  - File name [12 chars max]
  - Huffman compression flag
  - File data offset (relative to CPD start)
  - File data length (unpacked size for Huffman)

### FTPR[47]

<table>
<thead>
<tr>
<th></th>
<th>name</th>
<th>compr</th>
<th>offs:size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FTPR.man</td>
<td></td>
<td>478:D9C</td>
</tr>
<tr>
<td>2</td>
<td>rbe</td>
<td>HUFF</td>
<td>3D80:4000</td>
</tr>
<tr>
<td>3</td>
<td>rbe.met</td>
<td></td>
<td>1214:96</td>
</tr>
<tr>
<td>4</td>
<td>kernel</td>
<td>HUFF</td>
<td>63C0:13000</td>
</tr>
<tr>
<td>5</td>
<td>kernel.met</td>
<td></td>
<td>12AA:8E</td>
</tr>
<tr>
<td>6</td>
<td>syslib</td>
<td>HUFF</td>
<td>5100:17000</td>
</tr>
<tr>
<td>7</td>
<td>syslib.met</td>
<td></td>
<td>1338:64</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>touch_fw</td>
<td></td>
<td>CD140:2FFC</td>
</tr>
<tr>
<td>47</td>
<td>touch_fw.met</td>
<td></td>
<td>3C36:110</td>
</tr>
</tbody>
</table>

*Manifest** **Metadata
Manifest and Metadata

Manifest

- Header
  - Versions for Header (v4) and Data (v11.x.y.z)
  - Size of Header + Crypto Block (161 DWORDs) and whole Manifest
  - Vendor ID (0x8086 for Intel), Flags, Date, SVN
  - Modulus & Exponent sizes (64 and 1 DWORDs)

- Crypto Block
  - RSA Modulus (256 bytes == 2048 bits) + Exponent (usually 0x11 or 0x10001)
  - SHA-256 of Manifest (without Crypto Block) signed with RSA

- Extensions as Tag-Length-Value list

Metadata

- Extensions as Tag-Length-Value list

Funny TXE cases:

- CPD without Manifest
- Data File without Metadata
- Metadata without Data File
You cannot achieve the impossible without attempting the absurd
## Modules Statistics for ME Firmware Collection

<table>
<thead>
<tr>
<th>H</th>
<th>cnt</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>46</td>
<td>amt</td>
</tr>
<tr>
<td>+</td>
<td>210</td>
<td>bup</td>
</tr>
<tr>
<td>35</td>
<td></td>
<td>busdrv</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>cls</td>
</tr>
<tr>
<td>46</td>
<td></td>
<td>crypto</td>
</tr>
<tr>
<td>61</td>
<td></td>
<td>dal_ivm</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>dal_inch</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>dal_sdm</td>
</tr>
<tr>
<td>+</td>
<td>6</td>
<td>espi_drv</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>evtdisp</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>ews</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>fpf</td>
</tr>
<tr>
<td>57</td>
<td></td>
<td>fwupdate</td>
</tr>
<tr>
<td>+</td>
<td>16</td>
<td>gde</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>gpio</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>hci</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>heci</td>
</tr>
<tr>
<td>37</td>
<td></td>
<td>hotham</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>icc</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>ipc_drv</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>ish_bup</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>ish_main</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>ish_srv</td>
</tr>
<tr>
<td>+</td>
<td>79</td>
<td>kernel</td>
</tr>
<tr>
<td>+</td>
<td>11</td>
<td>lh_app</td>
</tr>
<tr>
<td>41</td>
<td></td>
<td>loadmgr</td>
</tr>
<tr>
<td>41</td>
<td></td>
<td>maestro</td>
</tr>
<tr>
<td>73</td>
<td></td>
<td>mca_boot</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>mca_srv</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>mctp</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>mctp_net</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>nfc</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>mctp_net</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>pRTC</td>
</tr>
<tr>
<td>96</td>
<td></td>
<td>ptt</td>
</tr>
<tr>
<td>38</td>
<td></td>
<td>pm</td>
</tr>
<tr>
<td>89</td>
<td></td>
<td>policy</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>pm</td>
</tr>
<tr>
<td>89</td>
<td></td>
<td>policy</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>tcb</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>tls_iso</td>
</tr>
<tr>
<td>48</td>
<td></td>
<td>touch_fw</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>usbr</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>vdm</td>
</tr>
<tr>
<td>57</td>
<td></td>
<td>vfs</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>vdm</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>vdm</td>
</tr>
<tr>
<td>57</td>
<td></td>
<td>vfs</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>wapps</td>
</tr>
<tr>
<td>+</td>
<td>37</td>
<td>wlan_drv</td>
</tr>
</tbody>
</table>

*Important modules that always compressed with Huffman*
Playing with Intel(R) Manifest Extension Utility

C:\MEU>meu.exe -gen CodePartition

Saving XML ...
XML file written to CodePartition.xml

C:\MEU\CodePartition.xml

```xml
<CPModules>
  <CPDataModule name="ish_main" enabled="true">
    <InputFile value="ish_main.bin" />
    <CompressionType value="LZMA" value_list="NOT_COMPRESSED,,LZMA" />
    <ProcessId value="0xf6" />
  </CPDataModule>
</CPModules>
```

String without reference!

```
dd offset aNot_compressed ; "NOT_COMPRESSED"
dd offset aHuffman ; "HUFFMAN"
dd offset aLzma ; DATA XREF: sub_246AF0+506↑r
  ; "LZMA"
dd offset aInvalid_compre ; "INVALID_COMPRESSION_SETTING"
```
BinWalk [Firmware Analysis Tool] misuse

C:\MEU>python C:\Python27\Scripts\binwalk meu.exe

<table>
<thead>
<tr>
<th>DECIMAL</th>
<th>HEXADECIMAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x0</td>
<td>Microsoft executable, portable (PE)</td>
</tr>
<tr>
<td>2810520</td>
<td>0x2AE298</td>
<td>XML document, version: &quot;1.0&quot;</td>
</tr>
<tr>
<td>2842816</td>
<td>0x2B60C0</td>
<td>Copyright string: &quot;Copyright (c) &quot;</td>
</tr>
<tr>
<td>2851456</td>
<td>0x2B8280</td>
<td>Zlib compressed data, default compression</td>
</tr>
<tr>
<td>2858473</td>
<td>0x2B9DE9</td>
<td>XML document, version: &quot;1.0&quot;</td>
</tr>
<tr>
<td>2860580</td>
<td>0x2BA624</td>
<td>Zlib compressed data, default compression</td>
</tr>
<tr>
<td>2867878</td>
<td>0x2BC2A6</td>
<td>Zlib compressed data, default compression</td>
</tr>
</tbody>
</table>

...
Compiled-In Resources – it’s so Qt ;)

For a resource to be compiled into the binary the .qrc file must be mentioned in the application’s .pro file so that qmake knows about it. For example:

```
RESOURCES = application.qrc
```
**Extracted Resources**

**Flash Image Tool:**
- `/res/bin/AFS_region_1272K.bin`
- `/res/bin/AFS_region_256K.bin`
- `/res/bin/AFS_region_400K.bin`
- `/res/bin/descriptor_template.bin`
- `/res/ftool/spt_layout.xsd`
- `/res/xml/bxt_fit_cfg_dflt.xml`
- `/res/xml/bxt_fit_layout.xml`
- `/res/xml/spt_fit_cfg_dflt.xml`
- `/res/xml/spt_fit_cfg_dflt_H.xml`
- `/res/xml/spt_fit_layout.xml`
- `/res/xml/spt_fit_layout_common.xml`
- `/res/xml/spt_fit_layout_H.xml`
- `/res/xml/spt_ftool_cfg_dflt.xml`
- `/res/xml/spt_ftool_layout.xml`
- `/res/xml/spt_layout.xsd`

**Manifest Extension Utility:**
- `/res/xml/bxt_meu_cfg_dflt.xml`
- `/res/xml/bxt_meu_layout.xml`
- `/res/xml/spt_meu_cfg_dflt.xml`
- `/res/xml/spt_meu_layout.xml`

*Resource files that referred from code*
Code Partition Directory
Manifest Extensions
Ext#0: System Info

- Minimum UMA* size required for this SKU** in bytes
- Chipset version
- SHA-256 hash of a 'defaults' file added to the image
- Size of pageable space within UMA in bytes
- List of Independent Partition Entry
  - Name
  - Version
  - User ID

Ext#0 SystemInfo[3]: uma_size:0x2000000, chipset_version:0x10000, pageable_uma_size:0x1390000
hash:09c7d8f41adb3d4daac980c7781027e25bb0ba80496535854eefac84b6de249c
  1: [LOCL] user_id:0x0021 ver:0x10000000
  2: [ISHC] user_id:0x0001 ver:0x10000000
  3: [WCOD] user_id:0x007B ver:0x00010000

*UMA: Unified Memory Architecture
**SKU: Stock Keeping Unit
For each entry:

- Manifest Partition Name + Module Name
- Flags used govern initialization flow
  - Ibl, IsRemovable, InitImmediately, RestartPolicy, Cm0_u, Cm0_nu, Cm3
- Boot path flag bits to indicate applicable boot path(s)
  - Normal, HAP, HMRFPO, TmpDisable, Recovery, SafeMode, FWUpdate

```
Ext#1 InitScript[57]:
1: FTPR:kernel    Init: Ibl, InitImmediately; Boot: Normal, Recovery
2: FTPR:syslib    Init: Ibl, InitImmediately; Boot: Normal, Recovery
3: FTPR:rbe      Init: Ibl, InitImmediately; Boot: Normal, Recovery
...              ...
12: NFTP:mca_boot Init: InitImmediately, Cm0_u; Boot: Normal
...              ...
55: NFTP:mctp_net Init: InitImmediately, Cm0_u, Cm0_nu, Cm3; Boot: Normal
56: NFTP:lh_app   Init: InitImmediately, Cm0_u, Cm0_nu, Cm3; Boot: Normal
57: NFTP:rmtwake  Init: InitImmediately, Cm0_u, Cm0_nu, Cm3,
                    Restart On Next Boot; Boot: Normal
```
Ext#2: Feature Permissions

• Number of features

For each feature:
  • User ID that may change feature state

<table>
<thead>
<tr>
<th>Ext#2 FeaturePermissions[3]:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: 0x0001</td>
</tr>
<tr>
<td>2: 0x0002</td>
</tr>
<tr>
<td>3: 0x0003</td>
</tr>
</tbody>
</table>

*No idea what is the “feature”*
Ext#3: Partition Info

- Name, Length and Hash of the partition
- Version Control Number (VCN), Partition and Data Format versions
- Instance ID and Flags

For each Entry:
- Name and Type (Process, Shared Library, Data)
- Metadata Size and Hash

Ext#3 PartitionInfo:
Name: [FTPR], Length: 0012F000
Hash: 2a8435bacac58a10fa7cb01e46149ad21ae064181912b073955a600cb3fd6
VCN: 10, Ver: 10000000, 10000, Instance ID: 1, Flags: 0

Modules[23]:
1: Proc, Meta size: 96 h:cd98f2a383fee83676568daa6a76fa1dc149e06144129f3a06c6469fbdca9ecc rbe
2: Proc, Meta size: 8E h:70d3081ae479d134b09f4d6ca2e7749cbd71f5b12f985357c452bd46f5f719c kernel
3: Lib , Meta size: 64 h:9f9d22d747f07989aa1a22ea3cdf04153d7af8dcbe89702d2441b22132c0c2fe syslib
   ...
22: Proc, Meta size: F2 h:5b2bb9eacb6087eef7d913d5e729af700ddca8b792d7f8bc42921d102f1ab4a8 ptt
23: Proc, Meta size:110 h:12f6569429fdde8ec0055a9c946c56a487d1d54fe6b27c32fc0eae2a7ff09b1d touch_fw
Ext#15: Package Info (TXE)

- Name of the partition (package)
- Version Control Number (VCN) and Secure Version Number (SVN)
- Bitmap of usages (which key is used to sign the manifest)

For each Entry:
- Name and Type (Process, Shared Library, Data)
- Hash Type (SHA1 or SHA256) and Length (actually, always 32)
- Metadata Size and Hash

```plaintext
Ext#15 PackageInfo:
Name: [FTPR], VCN: 10, Usage Bitmap: 01000000000000000000000000000000, svn: 1
Modules[4]:
1: Proc, Meta size: 8E h:52adfcec5b9596fe133129652d4a72db6cd84485a932eb80c7e1136b54c6f5d8 kernel
2: Lib , Meta size: 64 h:b991d991601d0555c2203804e187a6ef350f5b9fc1d824cc5a69c8af9f5cd864 syslib
3: Proc, Meta size:482 h:fc958d41cbb198515a2ba181b9375945567940dbf1ae5c44a32168dbad1299c4 bup
4: Data, Meta size: 38 h:b72535afbaef26531f408d2435733d4c5cc9d64f1ed0ba3c24e28548e9fd30d59 int1.cfg
```
**Ext#12: Client System Info**

- **FW SKU Capabilities**
- **FW SKU Attributes:**
  - CSE region size (in multiples of 0.5 MB)
  - Firmware SKU (0 for 5.0MB, 1 for 1.5MB, 2 for slim SKU)
  - Patsburg support
  - M3 support
  - M0 support
  - Si class (all H M L)

```python
Ext#12 ClientSystemInfo:
  fw_sku_caps: ffffffff
  fw_sku_attributes: CSE region size: 0.00, firmware sku: 5.0MB, Si class: 4, M3, M0
```
Code Partition Directory
Metadata Extensions
Ext#4: Shared Lib (syslib)

- Size in bytes of the shared library context
- Total alloc virtual space (including padding pages for library growth)
- Base address for the library private code
- Size of Thread-Local-Storage used by the shared library

Ext#4 SharedLib:
context_size:0x218, total_alloc_virtual_space:0x24000, code_base_address:0x0, tls_size:0x0

*Actual Code Base Address is 0x9000 for ME and 0x6000 for TXE*
Ext#5: Process Manifest

- Process Flags
  - Fault Tolerant, Permanent, Single Instance, Trusted/Public SendReceive Sender, Trusted/Public Notify Sender
- Thread ID for main thread. Optional for IBL processes only
- Base address for code
- Size of uncompressed process code
- Size of TLS & BSS
- Default Heap size
- Main thread entry
- Allowed syscalls
- User ID and list of Group IDs

Ext#5 Process:
```
flags: permanent_process, single_instance
main_thread_id: 0xC
priv_code_base_address: 0x00040000
uncompressed_priv_code_size: 0x29C6
cm0_heap_size: 0x0
bss_size: 0x7004
default_heap_size: 0x1000
main_thread_entry: 0x0004020A
allowed_sys_calls: e000c783f804000000000000
user_id: 0x005C
group_ids[1]: [0x0121]
```
Ext#6: Threads

For each entry:

• Size of main thread stack in bytes (not including guard page including space reserved for TLS)

• Flags

• Scheduling
  • Policy
  • Attributes

Ext#6 Threads[3]:
1: stack_size:0x00002000, flags:0, scheduling_policy:00000000
2: stack_size:0x00001000, flags:0, scheduling_policy:00001E00
3: stack_size:0x00001000, flags:0, scheduling_policy:00000000
Ext#7: Device IDs

List of Device IDs

Ext#7 DeviceIds[3]:
1:00020000
2:00020008
3:00020058
Ext#8: MMIO Ranges

For each entry:

- Base address of the MMIO range
- Limit in bytes of the MMIO range
- Access permissions (Write, Read)

Ext#8 MmioRanges[41]:

```plaintext
  sel=  7, base:F5022000, size:00000C00, flags:00000003 :: SUSRAM_S
  sel=  F, base:F5029000, size:00001000, flags:00000003 :: PRTC_S
  sel= 17, base:F461A000, size:00002000, flags:00000003 :: PMC_PCIP
  sel= 1F, base:F4628000, size:00004000, flags:00000003 :: PMC_PCIP
...
  sel=137, base:F1000000, size:00001000, flags:00000003 :: HECI1_PCIP
  sel=13F, base:F1007000, size:00001000, flags:00000003 :: GPIO_PROXY_PCIP
  sel=147, base:E00D0000, size:00001000, flags:00000003 :: PMC_PCIP
```

*LDT allows MMIO access from non-kernel code*

*NB: Range names are extracted from busdrv module*
Ext#9: Special File Producer

• Major number

For each entry:
  • Name (12 chars max)
  • User ID
  • Group ID
  • Minor number

```
Ext#9 SpecialFileProducer[3]: major_number=0x0005
1: crypto   access_mode:0660, user_id:0x000D group_id:0x0002 minor_number:00
2: dma_cse  access_mode:0660, user_id:0x000D group_id:0x0003 minor_number:01
3: crypto_gkey access_mode:0660, user_id:0x000D group_id:0x0004 minor_number:02
```

*Each record becomes entry in /dev/*
Ext#10: Module Attributes

- Compression type (Uncompressed, Huffman, LZMA)
- Encrypted (seen only for pavp module in ME 11.6)
- Uncompressed image size
- Compressed image size
- Module number unique in the scope of the vendor
- Vendor ID (PCI style). For Intel modules must be 0x8086
- SHA2 Hash of uncompressed image

Ext#10 ModAttr: Huff enc=0 00022760->00028000 id:000E.8086
h:b47aedef80d47c46fe02affb512532be4fe56fffd77c1bd597000c0b38bacd112

*Hash of most LZMA-packed modules calculated for compressed data*
Ext#11: Locked Ranges (kernel and syslib)

For each entry:

• Base address of range to be locked
• Size of range to be locked

```
Ext#11 LockedRanges[1]:
  1: base:0x80000, size:60C4
```

kernel
  base=0x80000

syslib for ME
  base=0x9000, size:0

syslib for TXE
  base=0x6000, size:0
### Ext#13: User Info (vfs)

For each entry:
- **User ID**
- **Maximum size of non-volatile storage area**
- **Maximum size of RAM storage area**
- **Quota to use in wear-out prevention algorithm**
- **Starting directory for the user**

<table>
<thead>
<tr>
<th>Ext#13 UserInfo[51]:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: user id:0x0001, NV quota: 75CCE, RAM quota: 0, WOP quota: 75CCE, working dir: [amt]</td>
</tr>
<tr>
<td>2: user id:0x0003, NV quota: 850, RAM quota: 1300, WOP quota: 850, working dir: [bup]</td>
</tr>
<tr>
<td>3: user id:0x0005, NV quota: 0, RAM quota: 0, WOP quota: 0, working dir: [busdrv]</td>
</tr>
<tr>
<td>4: user id:0x000A, NV quota: 2937, RAM quota: 0, WOP quota: 2937, working dir: [cls]</td>
</tr>
<tr>
<td>5: user id:0x000D, NV quota: 0, RAM quota: 0, WOP quota: 0, working dir: [crypto]</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>49: user id:0x0074, NV quota: 0, RAM quota: 0, WOP quota: 0, working dir: [vdm]</td>
</tr>
<tr>
<td>50: user id:0x0076, NV quota: D00, RAM quota: 0, WOP quota: D00, working dir: [vfs]</td>
</tr>
<tr>
<td>51: user id:0x007B, NV quota: A3A2, RAM quota: 0, WOP quota: A3A2, working dir: [wlan_drv]</td>
</tr>
</tbody>
</table>
Integrity Dependencies

Partition Manifest data
- Partition Info Extension
  - SHA256(Module Metadata)
- Module Metadata
  - Module Attributes Extension
  - SHA256(Module Data)
- Module Data

Partition Manifest
- Header
- RSA Signature
  - SHA256(Partition Manifest Header + Data)

RSA Public Key

ROM [Bypass]
- List of allowed signature keys
  - SHA256(RSA Public Key)
ROM
Bypass
Rare FW images (built between 2014-12-15 and 2015-07-01) have non-empty ROMB partition:

- **name**: [ROMB]  
- **offset:size**: 1000:20000  
- **type**: Data  
- **comments**: # ROM Bypass

Partition data mapped in RAM at address 0x1000

16 bytes before $FPT$ contains JMP instruction
ME ROM Bypass Layout

- ROM library persists in memory for all processes
- Default syslib is used by `rbe` & `kernel`
- Non-kernel modules uses FW `syslib` code loaded from Flash at the same address
- ROM initialization code required during startup only

```
jmp fn_00
jmp fn_01
...
jmp fn_NN
fn_00 {
  ...
}
fn_01 {
  ...
}
...
fn_NN {
  ...
}
```
• Setup ME hardware and CPU state (IDT, GDT, …)

• Setup internal data structures

• Derive some keys (CSE Wrapping Key, CSE Suspend Key, CSE UMA 128 Key)

• Try to load and execute startup module:
  • DLMP:idlm
  • FTPR:rbe or FTUP:rbe
ME Modules Base Address

- All non-kernel modules use the same *base* address
- Known values for *base*:
  - 0x21000 v11.0.0.1100
  - 0x24000 v11.0.0.1115–1122
  - 0x2C000 v11.0.0.1131–1133
  - 0x2D000 v11.0.0.1140–1205
  - v11.0.1–11.0.18
  - 0x2E000 SPS v4
  - 0x31000 v11.5–11.6
  - 0x26000 TXE v3

- TXE v3 load syslib at 0x6000 instead of 0x9000
Some Finds
And Future Plans
**ME v11**

*NB: v11.6 modules contains less debug messages in comparison with v11.0 modules*

<table>
<thead>
<tr>
<th>Version</th>
<th>FTPR code size (KB)</th>
<th>FTPR data size (KB)</th>
<th>NFTP code size (KB)</th>
<th>NFTP data size (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.0.0.1120</td>
<td>894</td>
<td>249</td>
<td>1029</td>
<td>422</td>
</tr>
<tr>
<td>11.0.1.1001</td>
<td>934</td>
<td>261</td>
<td>927</td>
<td>400</td>
</tr>
<tr>
<td>11.0.18.1002</td>
<td>946</td>
<td>265</td>
<td>928</td>
<td>399</td>
</tr>
<tr>
<td>11.5.1.1006</td>
<td>883</td>
<td>224</td>
<td>970</td>
<td>406</td>
</tr>
<tr>
<td>11.6.10.1196</td>
<td>897</td>
<td>234</td>
<td>998</td>
<td>417</td>
</tr>
<tr>
<td>11.0.0.1122</td>
<td>AMT</td>
<td>898</td>
<td>249</td>
<td>3087</td>
</tr>
<tr>
<td>11.0.1.1001</td>
<td>AMT</td>
<td>934</td>
<td>261</td>
<td>2984</td>
</tr>
<tr>
<td>11.0.18.1002</td>
<td>AMT</td>
<td>946</td>
<td>269</td>
<td>2986</td>
</tr>
<tr>
<td>11.5.1.1006</td>
<td>AMT</td>
<td>914</td>
<td>221</td>
<td>3026</td>
</tr>
<tr>
<td>11.6.10.1196</td>
<td>AMT</td>
<td>928</td>
<td>231</td>
<td>3031</td>
</tr>
</tbody>
</table>
• All seen FW images have no FPT
• Two types of CPD: REC and OPR
• REC (Recovery):
  • Just 4 modules: rbe, kernel, syslib, bup_rcv
  • All modules compressed with Huffman
• OPR (Operation):
  • 36 modules
  • ish_bup compressed with LZMA, all others – with Huffman
• No AMT
• No ROM Bypass seen (could be compatible with ME ROM Bypass)
TXE v3 (first FW seen in March 2017)

- No Huffman – all modules could be decompressed with LZMA

- FTPR contains just 3 modules: \texttt{bup, kernel, syslib}

- FTPR also contains \texttt{intl.cfg} and \texttt{fitc.cfg} (in ME that files was embedded in MFS partition)

- \texttt{rbe} module in separate RBEP partition

- No ROM bypass seen
Intel® launches its first bug bounty program

**Intel® Bug Bounty Program**

At the CanSecWest Security conference on March 14, 2017, Intel launched its first Bug Bounty program targeted at Intel Products. We want to encourage researchers to identify issues and bring them to us directly so that we can take prompt steps to evaluate and correct them, and we want to recognize researchers for the work that they put in when researching a vulnerability. By partnering constructively with the security research community, we believe we will be better able to protect our customers.

**Scope and Severity Ratings**

Intel Software, Firmware, and Hardware are in scope. The harder a vulnerability is to mitigate, the more we pay.
Do you ever read “Modern Operating Systems”?

MINIX3 by Andrew Tanenbaum

Directory of minix3-master\servers\vfs

14.03.2010 23:52 14'978 main.c
14.03.2010 23:52    741 Makefile
14.03.2010 23:52 17'653 misc.c
14.03.2010 23:52    677 mmap.c
14.03.2010 23:52 15'650 mount.c

> strings vfs
  ...
  ..\..\src\os\servers\vfs\misc.c
FS: bogus child for forking
FS: forking on top of in-use child
...
Future Plans

- Make slides about 7 types of File Systems supported by ME (including MFS FTL details)

- Win a battle with the Huffman compression

- Figure out how ME works

- Try to get something useful from all above ;)}
Thank You!

POSITIVE TECHNOLOGIES

ptsecurity.com
www.ptsecurity.com

blog.ptsecurity.com

@PTsecurity_UK

github.com/ptresearch