TROOPERS17

20th-24th March 2017

Heidelberg, Germany

Intel ME: The Way of the Static Analysis

POSITIVE TECHNOLOGIES

ptsecurity.com

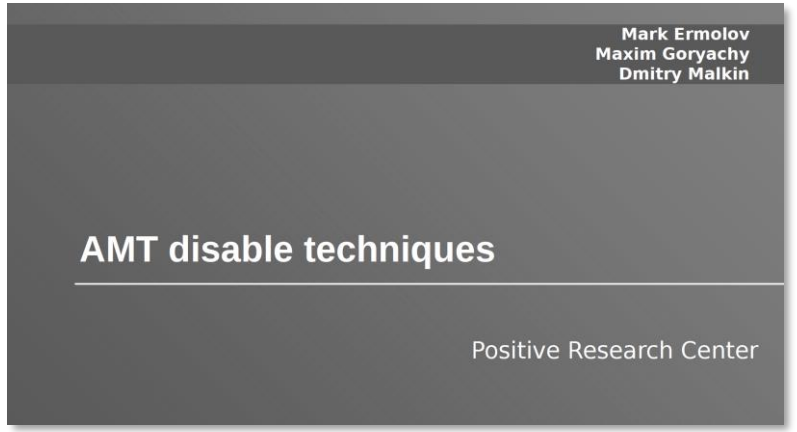
Research Team

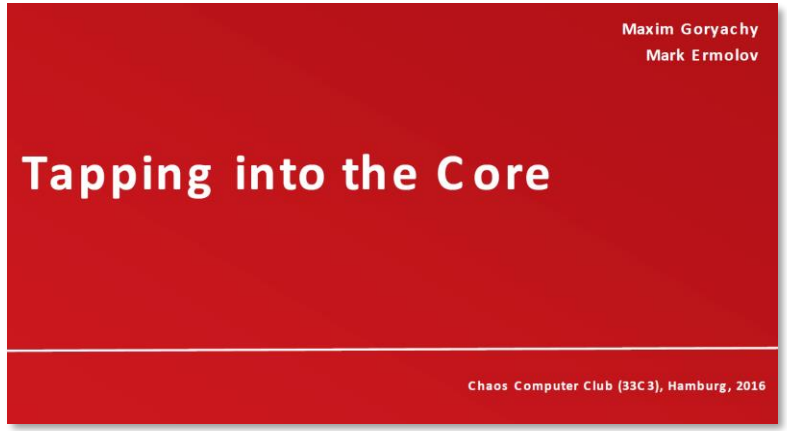
POSITIVE TECHNOLOGIES

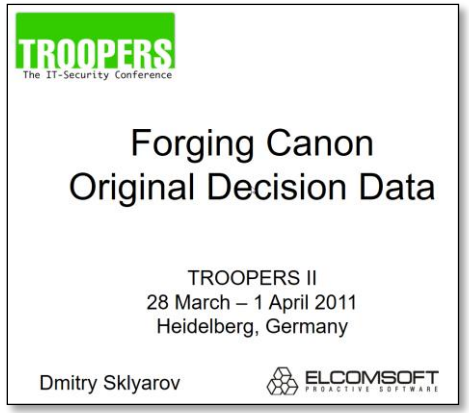
Maxim Goryachy

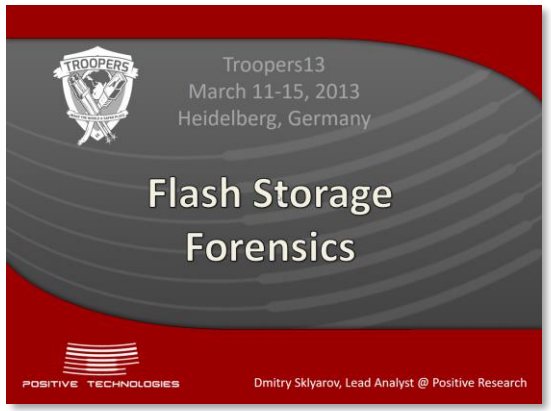
Mark Ermolov

Dmitry Sklyarov









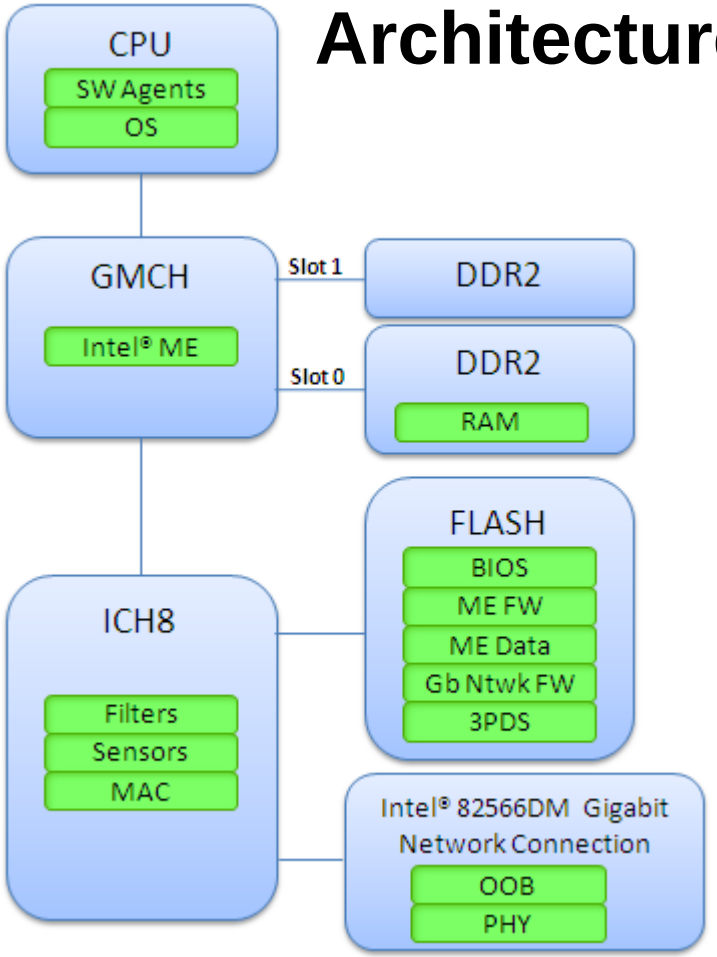
Intel ME

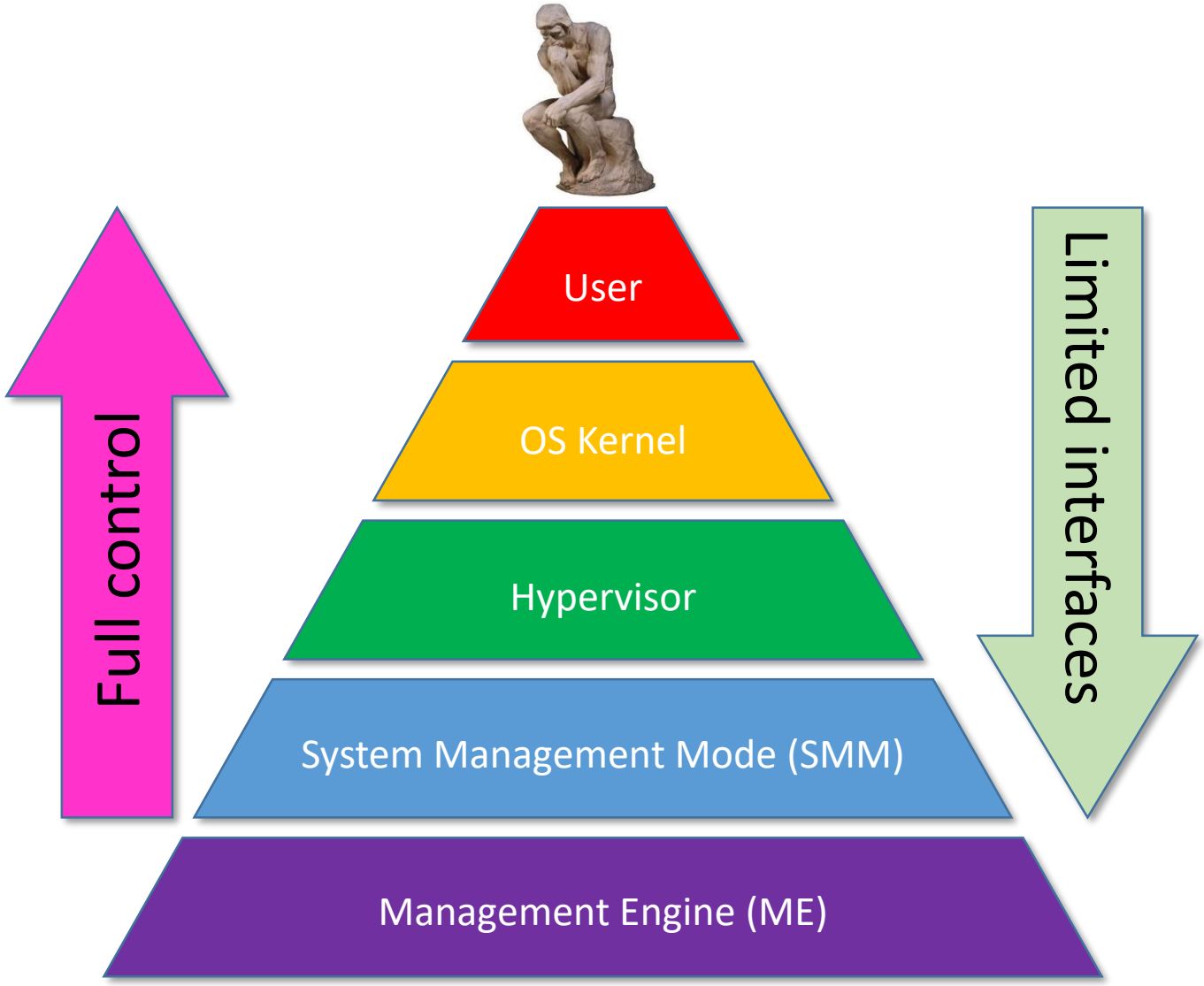
Known Facts

ME Position in Computer System

POSITIVE TECHNOLOGIES

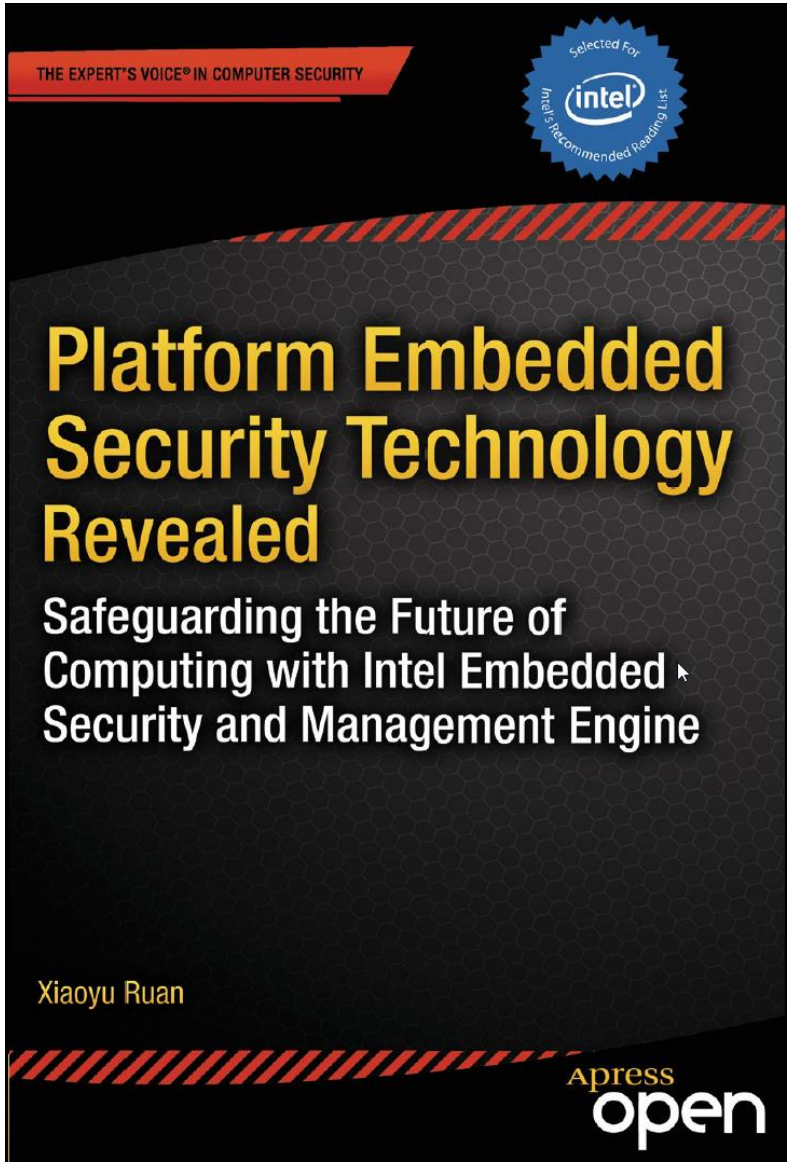
Intel AMT Release 2.0/2.1/2.2 Architecture



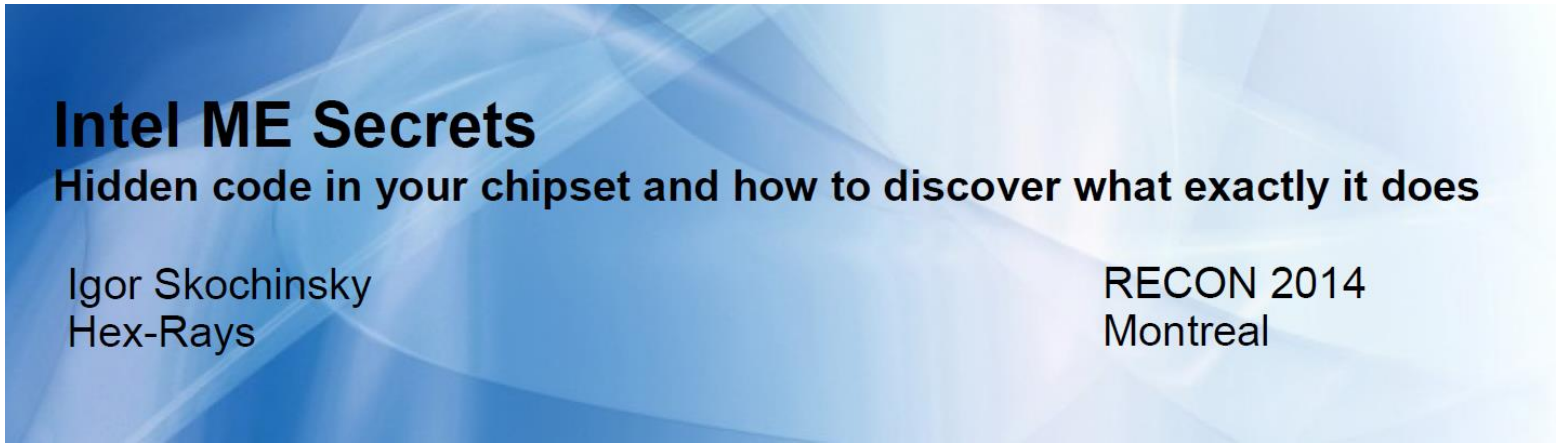


Sources of Information

POSITIVE TECHNOLOGIES



skochinsky / me-tools



intel

Patents / White Papers / Documentation

ME Platform Features

POSITIVE TECHNOLOGIES

Version	CPU Arch	LZMA	Huffman	OS Arch
ME 2.x-5.x SPS 1.x	ARCtangent	+	-	
ME 6.x-10.x SPS 2.x-3.x	ARCompact	+	+	ThreadX
TXE 1.x-2x	SPARC	+	-	ThreadX
TXE 3.x	x86	+	-	?
SPS 4.x CSE(ME) 11.x	x86	+	Tables unknown	?

} Area of interest

TXE: Trusted Execution Engine (for Intel Atom CPUs)

SPS: Server Platform Services

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

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? Never heard of it ;)

POSITIVE TECHNOLOGIES

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

POSITIVE TECHNOLOGIES

	name	offset:size	type	comments
1:	**[FTPR]**	**1000:C9000**	**Code**	# Main code partition
2:	[FTUP]	20C000:38B000	Code	# [NFTP]+[WCOD]+[LOCL]
3:	**[DLMP]**	**0:0**	**Code**	# IDLM partition
4:	**[PSVN]**	**E00:200**	**Data**	# Secure Version Number
5:	[IVBP]	208000:4000	Data	# IV + Bring Up cache
6:	[MFS]	CA000:13E000	Data	# ME Flash File System
7:	**[NFTP]**	**20C000:308000**	**Code**	# Additional code
8:	[ROMB]	0:0	Code	# ROM Bypass
9:	[WCOD]	514000:80000	Code	# WLAN uCode
10:	[LOCL]	594000:3000	Code	# AMT Localization
11:	[FLOG]	597000:1000	Data	# Flash Log
12:	**[UTOK]**	**598000:2000**	**Data**	# Debug Unlock Token
13:	[ISHC]	59A000:2B000	Code	# Integrated Sensors Hub

**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]
     name          compr  offs:size
 1:  FTPR.man              478:D9C
 2:  rbe           HUFF    3D80:4000
 3:  rbe.met               1214:96
 4:  kernel        HUFF    63C0:13000
 5:  kernel.met            12AA:8E
 6:  syslib        HUFF    5100:17000
 7:  syslib.met            1338:64
     ...
46:  touch_fw              CD140:2FFC
47:  touch_fw.met          3C36:110
```

*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

POSITIVE TECHNOLOGIES

H	cnt	name
+	46	amt
+	210	bup
	35	busdrv
	30	cls
	46	crypto
	61	dal_ivm
	24	dal_lnch
	23	dal_sdm
+	6	espi_drv
	22	evtdisp
	16	ews
	16	fpf
	57	fwupdate
+	16	gde

H	cnt	name
	30	gpio
	20	hci
	30	heci
	37	hotham
	100	icc
	19	ipc_drv
	11	ish_bup
	18	ish_main
	26	ish_srv
+	79	kernel
+	11	lh_app
	41	loadmgr
	41	maestro
	73	mca_boot

H	cnt	name
	49	mca_srv
	21	mctp
+	13	mctp_net
	21	nfc
	50	pavp
	24	pm
	38	pmdrv
	89	policy
	26	prtc
	96	ptt
+	31	rbe
+	13	rmtwake
	28	rosm
	19	sensor

H	cnt	name
	37	sigma
	29	smbus
	51	storage
	10	syncman
+	43	syslib
	22	tcb
	13	tls_iso
	48	touch_fw
	16	usbr
	26	vdm
	57	vfs
+	5	wapps
+	37	wlan_drv

**Important modules that always compressed with Huffman*

Playing with Intel(R) Manifest Extension Utility

POSITIVE TECHNOLOGIES

```
C:\MEU>meu.exe -gen CodePartition

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

C:\MEU\CodePartition.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

POSITIVE TECHNOLOGIES

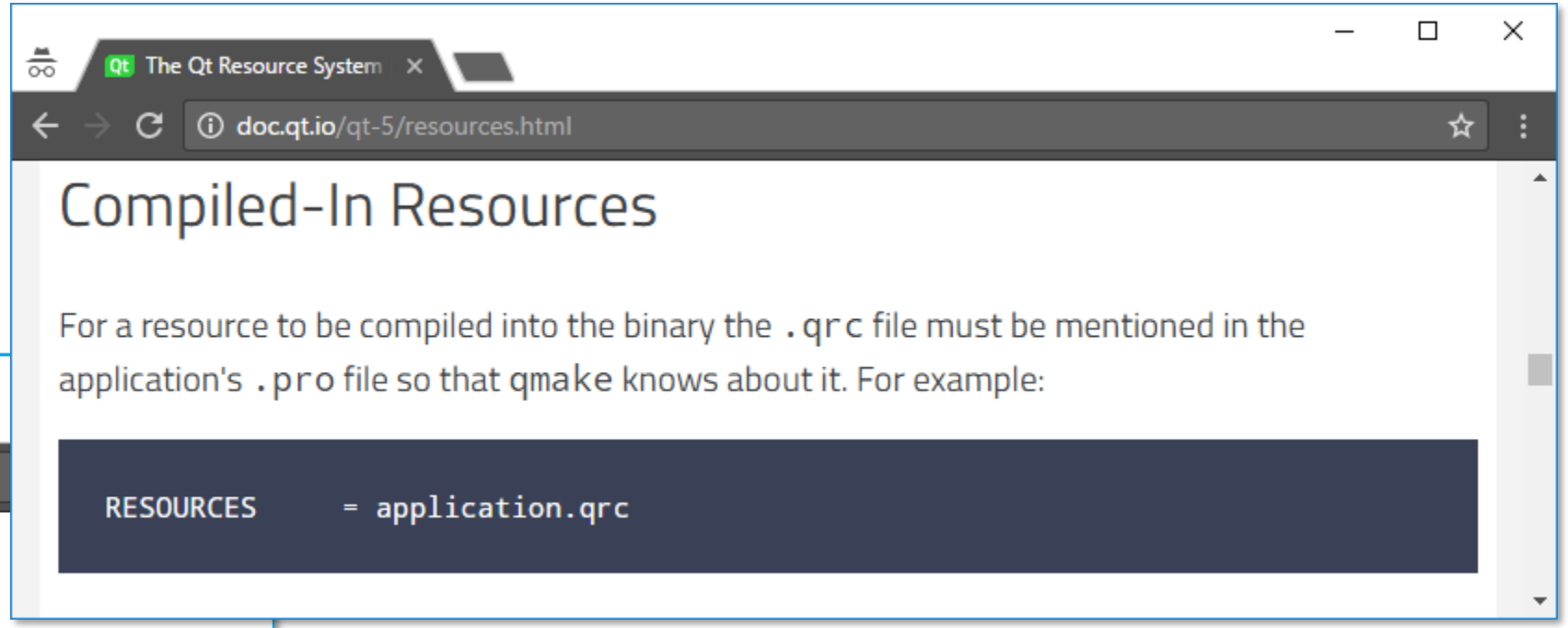
```
C:\MEU>python C:\Python27\Scripts\binwalk meu.exe

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Microsoft executable, portable (PE)
2810520       0x2AE298        XML document, version: "1.0"
2842816       0x2B60C0        Copyright string: "Copyright (c) "
2851456       0x2B8280        Zlib compressed data, default compression
2858473       0x2B9DE9        XML document, version: "1.0"
2860580       0x2BA624        Zlib compressed data, default compression
2867878       0x2BC2A6        Zlib compressed data, default compression
                              ...
```

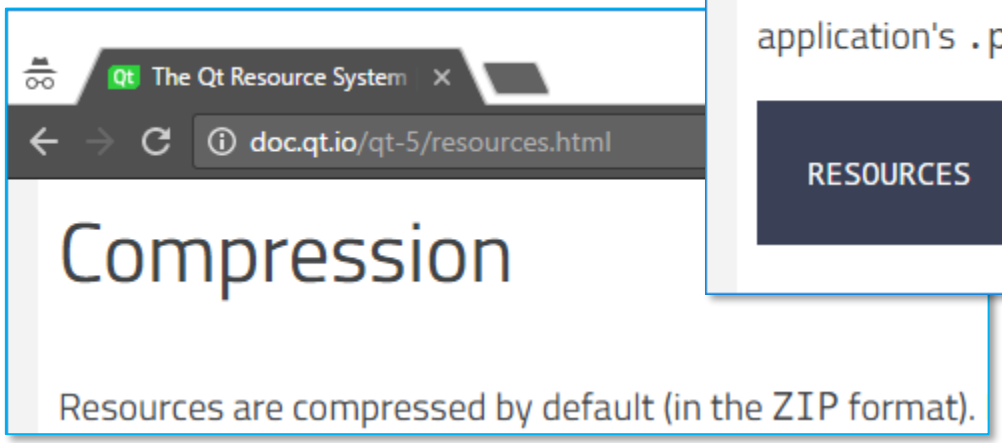
}?

Compiled-In Resources – it's so Qt ;)

POSITIVE TECHNOLOGIES

```
db 'QResourceInfo: Resource [%s] has both data and children!',0
                        ; DATA XREF: sub_3DB140+1FE↑o
align 10h
dd offset unk_6283FC
dd offset sub_3D93E0    ; DATA XREF: sub_3D8FD0+44↑o
db 'QResourceFileEngine::open: Missing file name',0
```





Extracted Resources

POSITIVE TECHNOLOGIES

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

POSITIVE TECHNOLOGIES

- 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*

Ext#1: Init Script

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

POSITIVE TECHNOLOGIES

- Number of features

For each feature:

- User ID that may change feature state

```
Ext#2 FeaturePermissions[3]:
 1: 0x0001
 2: 0x0002
 3: 0x0003
```

**No idea what is the "feature"*

Ext#3: Partition Info

POSITIVE TECHNOLOGIES

- 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: 2a8435bacac58db8a10fa7cb01e46149ad21ae064181912b073955a600cb3fc6
VCN: 10, Ver: 10000000, 10000, Instance ID: 1, Flags: 0
Modules[23]:
 1: Proc, Meta size: 96 h:cd98f2a383fef83676568daa6a76fa1dc149e06144129f3a06c6469fbfca9ecc rbe
 2: Proc, Meta size: 8E h:70d3081ae479d134b09f4d6ca2e7749cbd71f5b12f985357c452bd46f57f719c kernel
 3: Lib , Meta size: 64 h:9f9d22d747f07989aa1a22ea3cdf04153d7af8dcbe89702d2441b22132c0c2fe syslib
    ...
22: Proc, Meta size: F2 h:5b2bb9eacb6087eef7d913d5e729af700ddca8b792d7f8bc42921d102f1ab4a8 ptt
23: Proc, Meta size:110 h:12f6569429fde8ec0055a9c946c56aa487d1d54fe6b27c32fc0eae2a7ff09b1d touch_fw
```

Ext#15: Package Info (TXE)

POSITIVE TECHNOLOGIES

- 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

```
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:fc958d41ccb198515a2ba181b9375945567940dbf1ae5c44a32168dbad1299c4 bup
 4: Data, Meta size: 38 h:b72535afbaf26531f408d2435733d4c5cc9d64f1ed0ba3c24e28548e9fd30d59 intl.cfg
```

Ext#12: Client System Info

POSITIVE TECHNOLOGIES

- 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)

```
Ext#12 ClientSystemInfo:
  fw_sku_caps: fffffdf
  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]:
  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

POSITIVE TECHNOLOGIES

- 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

POSITIVE TECHNOLOGIES

- 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:b47aedef80d47c46fe02affb512532be4fe56ffd77c1bd597000c0b38bacd112
```

**Hash of most LZMA-packed modules calculated for compressed data*

Ext#11: Locked Ranges (`kernel` and `syslib`)

POSITIVE TECHNOLOGIES

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`)

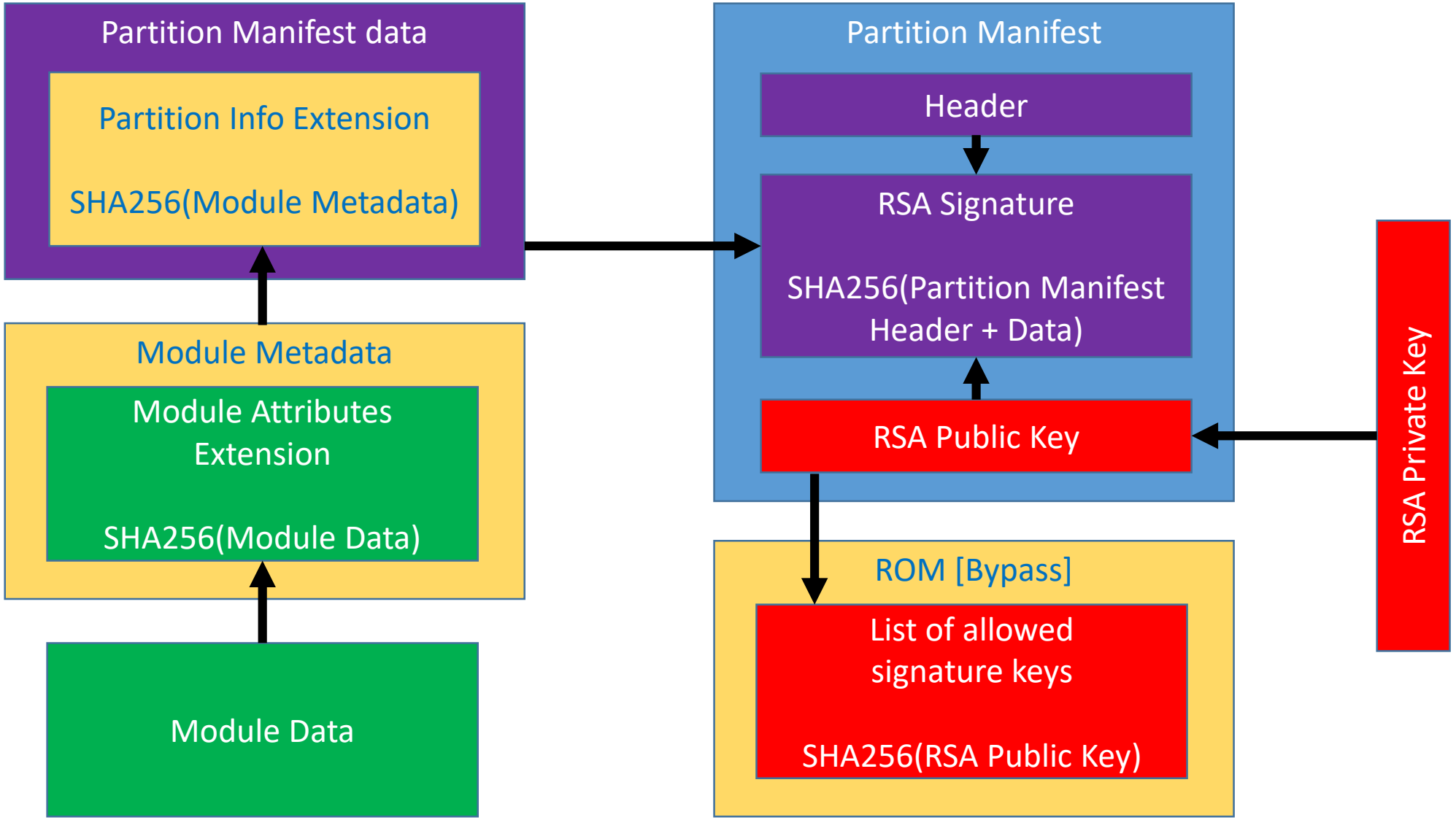
For each entry:

- 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

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

Integrity Dependencies

POSITIVE TECHNOLOGIES



ROM

Bypass

ROM Bypass Location

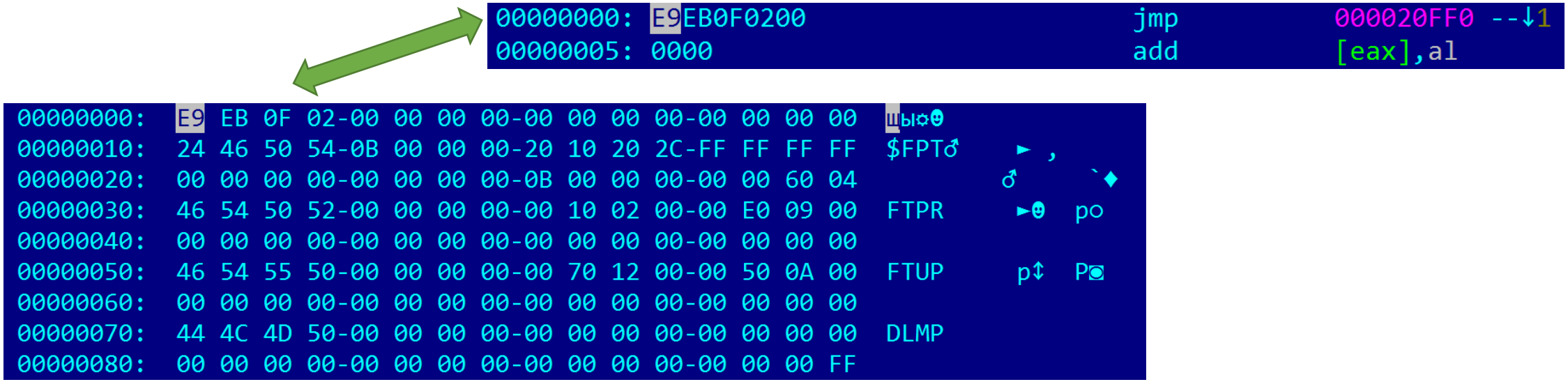
POSITIVE TECHNOLOGIES

Rare FW images (built between 2014-12-15 and 2015-07-01) has non-empty ROMB partition:

```
      name     offset:size       type     comments
 8: [ROMB]       1000:20000     Data # ROM Bypass
```

Partition data mapped in RAM at address 0x1000

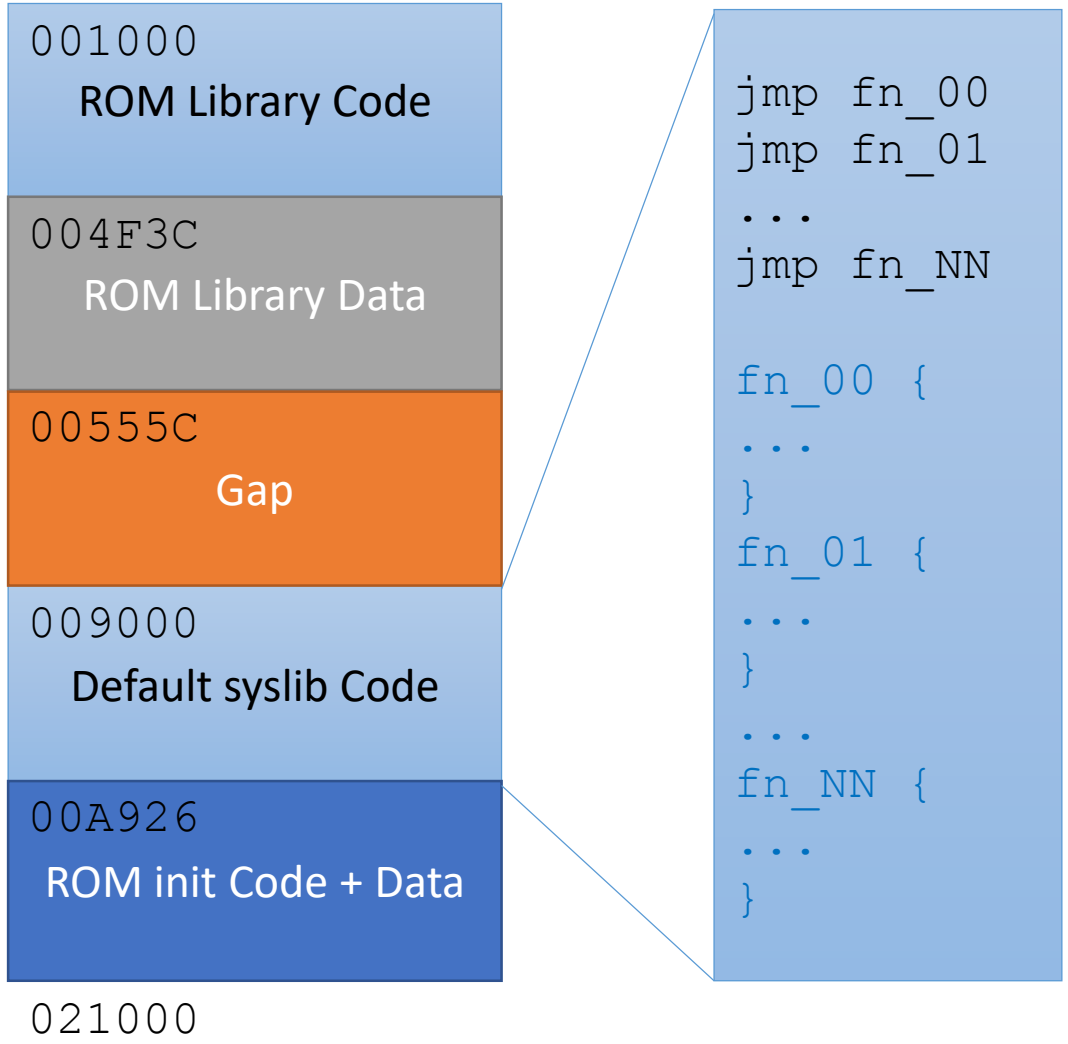
16 bytes before $FPT contains JMP instruction



ME ROM Bypass Layout

POSITIVE TECHNOLOGIES

- 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



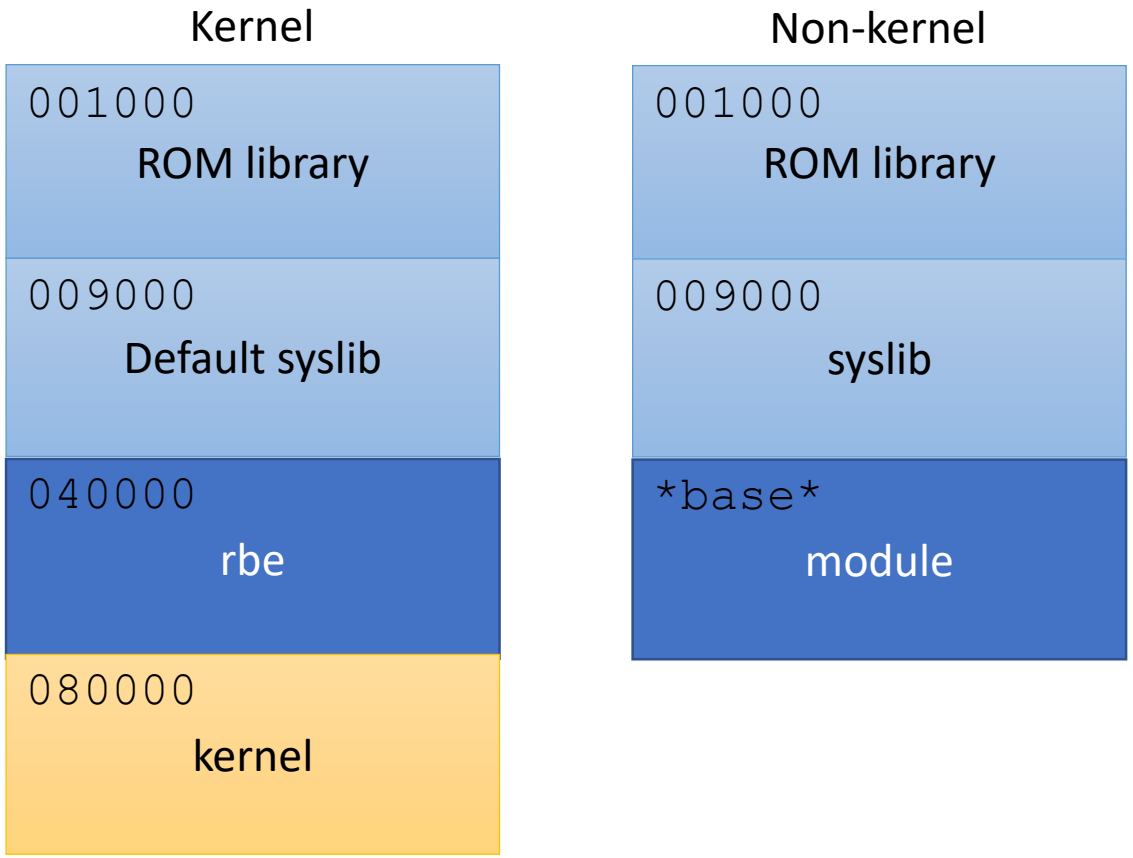
ROM Initialization Sequence

- 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

POSITIVE TECHNOLOGIES

- All non-kernel modules uses 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

POSITIVE TECHNOLOGIES

Version		FTPR code size (KB)	FTPR data size (KB)	NFTP code size (KB)	NFTP data size (KB)
11.0.0.1120		894	249	1029	422
11.0.1.1001		934	261	927	400
11.0.18.1002		946	265	928	399
11.5.1.1006		883	224	970	406
11.6.10.1196		897	234	998	417
11.0.0.1122	AMT	898	249	3087	976
11.0.1.1001	AMT	934	261	2984	971
11.0.18.1002	AMT	946	269	2986	969
11.5.1.1006	AMT	914	221	3026	917
11.6.10.1196	AMT	928	231	3031	908

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

SPS v4

- 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: `bup`, `kernel`, `syslib`
- FTPR also contains `intl.cfg` and `fitc.cfg` (in ME that files was embedded in MFS partition)
- `rbe` module in separate RBEP partition
- No ROM bypass seen

Intel Could Make ~~Us~~ You Rich!

POSITIVE TECHNOLOGIES

Secure | https://security-center.intel.com/BugBountyProgram.aspx

Home > Security Center >

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.

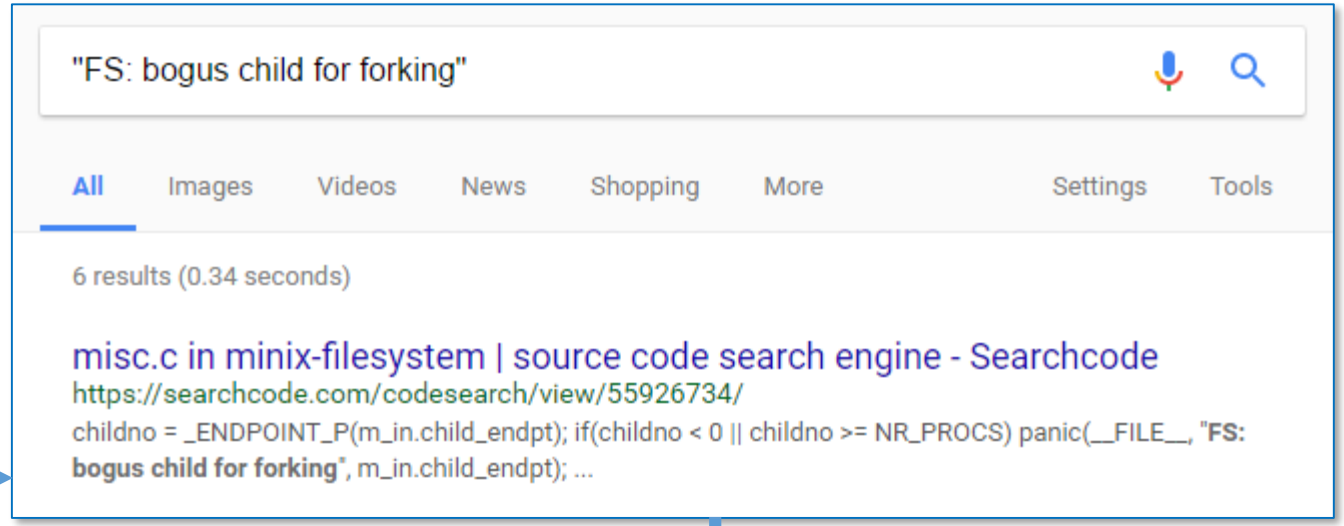
Scope and Severity Ratings

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”?

POSITIVE TECHNOLOGIES

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



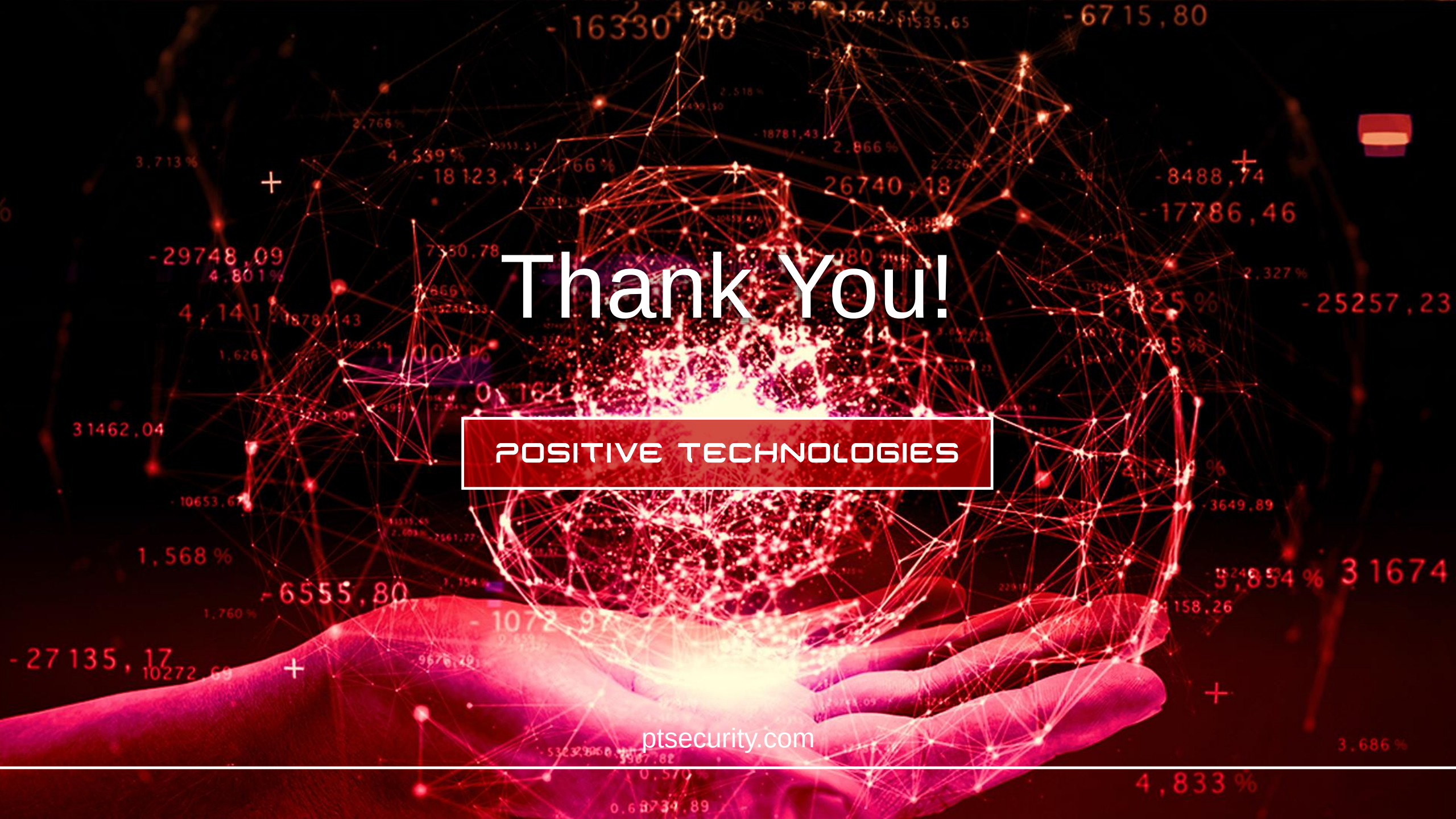
MINIX3
by Andrew Tanenbaum

```
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
                              ...
```

```
/* PM gives child endpoint, which implies process slot information.
 * Don't call isokendpt, because that will verify if the endpoint
 * number is correct in fproc, which it won't be.
 */
childno = _ENDPOINT_P(m_in.child_endpt);
if(childno < 0 || childno >= NR_PROCS)
      panic(__FILE__, "FS: bogus child for forking", m_in.child_endpt);
if(fproc[childno].fp_pid != PID_FREE)
      panic(__FILE__, "FS: forking on top of in-use child", childno);
```

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

Links

www.ptsecurity.com

blog.ptsecurity.com

@PTsecurity_UK

github.com/ptresearch