AMD PSP UEFI Firmware Structure

George Zaitseff

Junior Security Researcher // kks team

Moscow, August 25-26, 2022

Unified Extensible Firmware Interface

Unified Extensible Firmware Interface

Boot-services

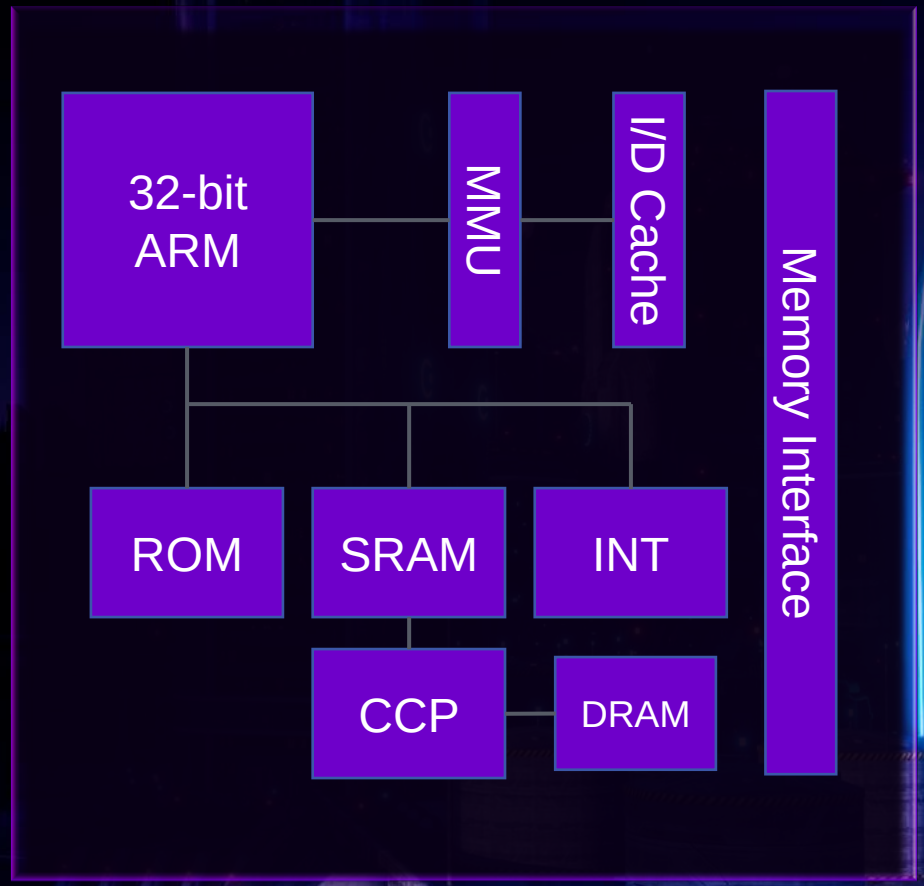
Runtime-services

About AMD PSP

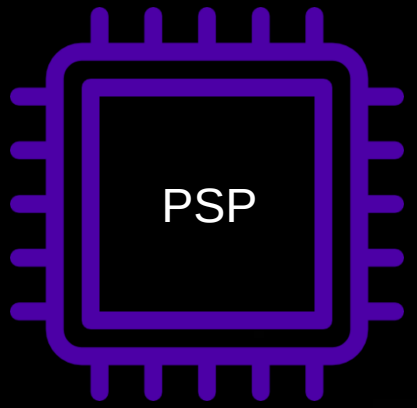
AMD PSP

Platform Security Processor

- Dedicated 32-bit ARM Cortex-A5 co-processor
- Part of SOC
- Has own OS
- Has TPM-like module (CCP)
- HW logic for secure control of boot process
- Has access to system memory / resources
- Has own
  - Local memory
  - Local registers
  - Etc… it's CPU in CPU
- Server and Desktop version



Boot stages



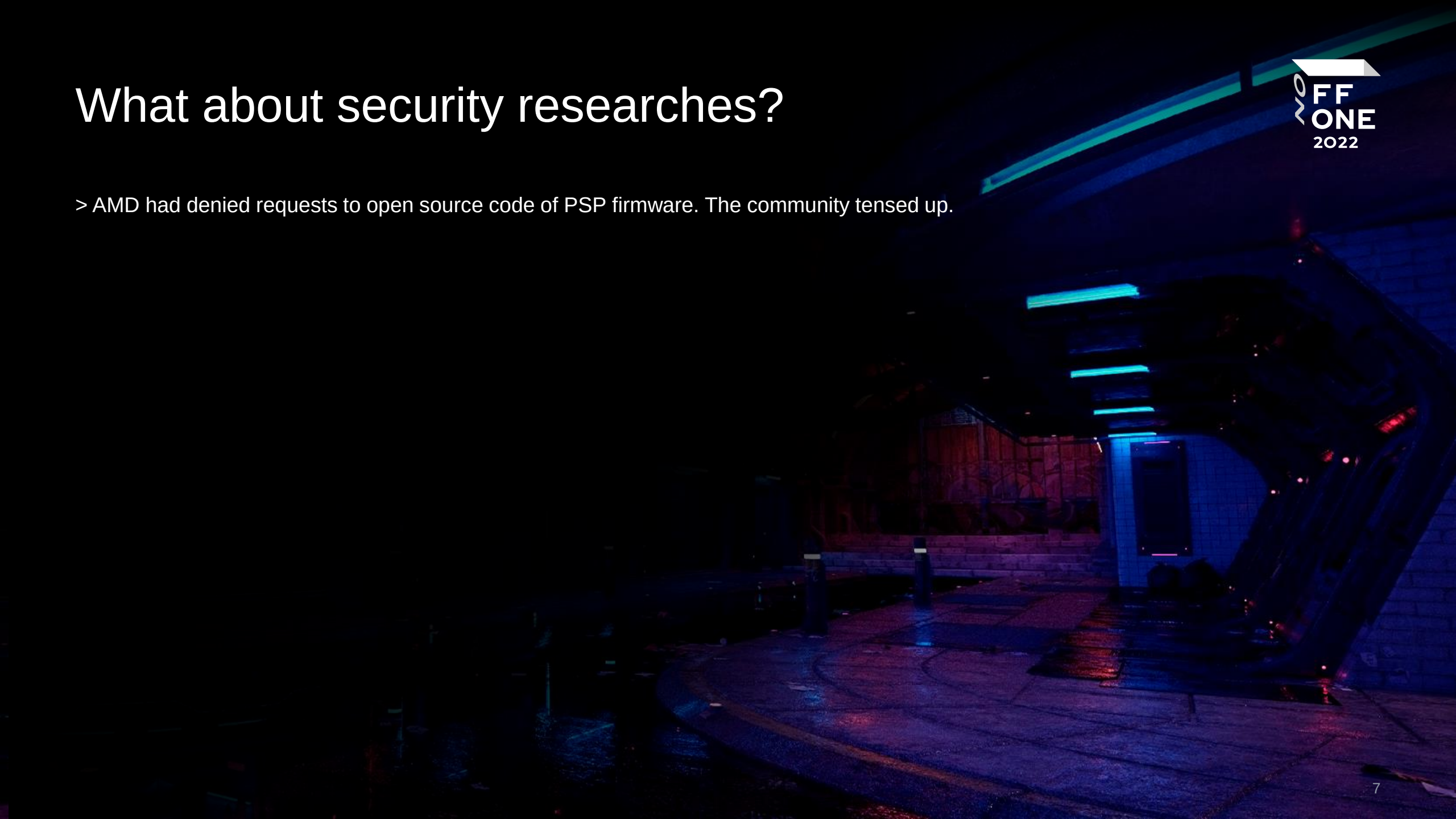
PSP Inner Bootloader

Security (SEC)	Pre EFI init (PEI)	Driver Execution Env (DXE)	Boot Select (BDS)

PSP+UEFI

What about security researches?

> AMD had denied requests to open source code of PSP firmware. The community tensed up.



What about security researches?

> AMD had denied requests to open source code of PSP firmware. The community tensed up.

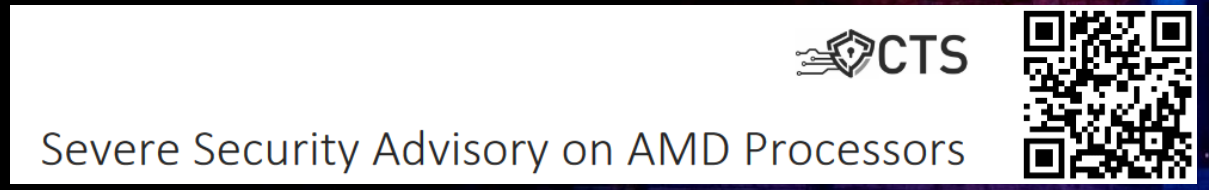
> 2017 – Google, Cfir Gohen, Google. Bug in CCP (TMP) module, stack overflow + code execution

What about security researches?

> AMD had denied requests to open source code of PSP firmware. The community tensed up.

> 2017 – Google, Cfir Gohen, Google. Bug in CCP (TMP) module, stack overflow + code execution

> 2018 - CTS-Labs, Israel. Backdoors, code execution and other fun staff

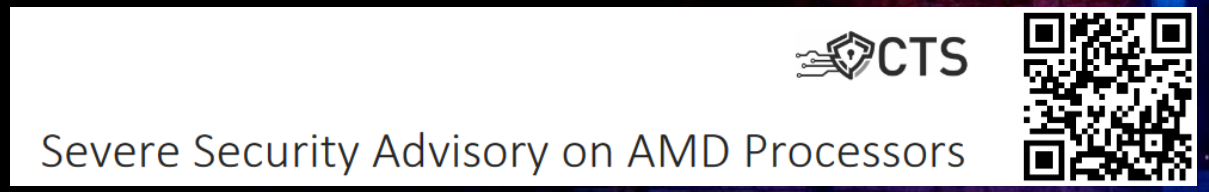


What about security researches?

> AMD had denied requests to open source code of PSP firmware. The community tensed up.

> 2017 – Google, Cfir Gohen, Google. Bug in CCP (TMP) module, stack overflow + code execution

> 2018 - CTS-Labs, Israel. Backdoors, code execution and other fun staff



> 2019 – 2021 – PSPTool team (Robert Buhren, …) – big amount if research papers/talk. Found bug in Secure Encrypted Virtualization

About PSPTool

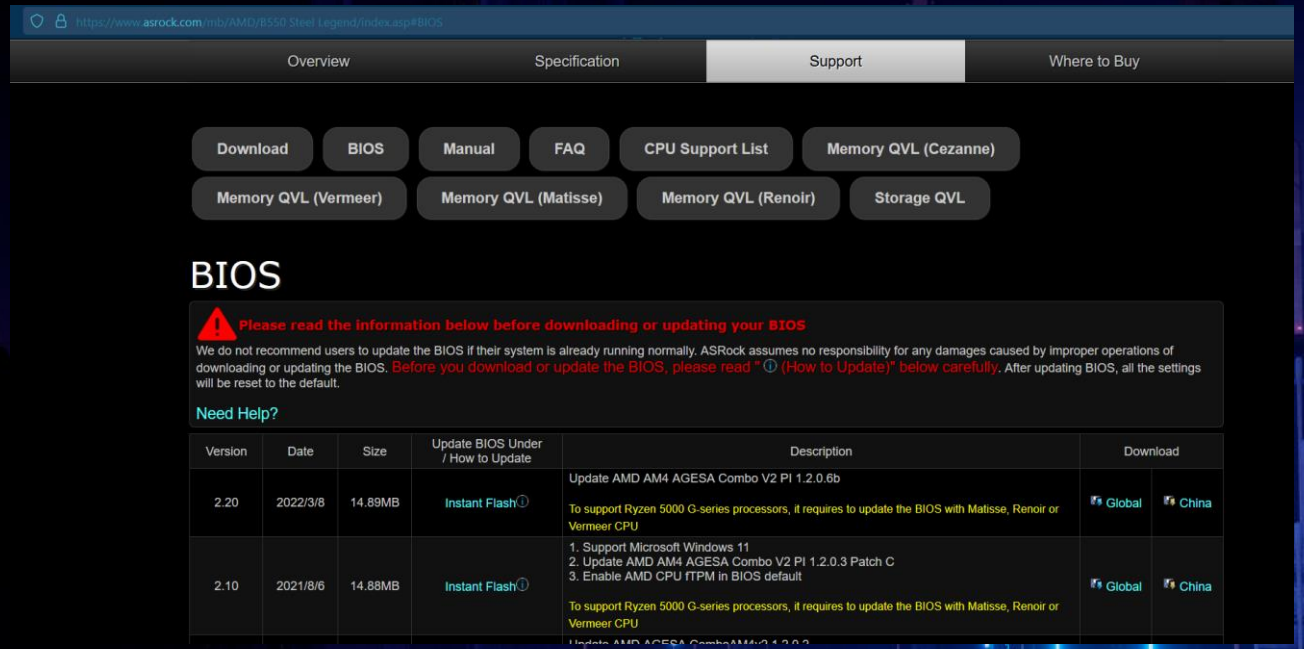
PSPTool was developed at TU Berlin in context of the following research:

- 2021, paper: One Glitch to Rule Them All: Fault Injection Attacks Against AMD's Secure Encrypted Virtualization
- 2020, talk: All You Ever Wanted to Know about the AMD Platform Security Processor and were Afraid to Emulate
- 2019, talk: Uncover, Understand, Own - Regaining Control Over Your AMD CPU
- 2019, paper: Insecure Until Proven Updated: Analyzing AMD SEV's Remote Attestation
- 2019, talk: Dissecting the AMD Platform Security Processor

```
+--+-----------+----------+------+-------+---------------------+
|  | Directory |   Addr   | Type | Magic | Secondary Directory |
+--+-----------+----------+------+-------+---------------------+
|  |     L0    | 0x5df000 | BIOS |  $BHD |          --         |
+--+-----------+----------+------+-------+---------------------+
+--+---+-------+----------+---------+-----------------+----------+----------+----------------------------+
|  |   | Entry |  Address |    Size |            Type | Magic/ID |  Version |                       Info |
+--+---+-------+----------+---------+-----------------+----------+----------+----------------------------+
|  |   |     0 | 0x5e0000 |   0x100 |       0x1000068 |        1 |  0.0.0.0 |      no_key, legacy_header |
|  |   |     1 | 0x5e4000 |   0x100 |       0x1000060 |          |  0.0.0.0 |      no_key, legacy_header |
|  |   |     2 | 0x5e5000 |   0x100 |       0x1800068 |          |  0.0.0.0 |      no_key, legacy_header |
|  |   |     3 |      0x0 |     0x0 |     FW_GEC~0x61 |          |          |                            |
|  |   |     4 |  0xe8000 | 0x28000 | FW_INVALID~0x63 |          |          |                            |
|  |   |     5 | 0x5e5200 |  0x4e20 |        0x100064 |     0x05 | 0.0.19.4 | compressed, verified(CCOD) |
|  |   |     6 | 0x5ea100 |   0x460 |        0x100065 |     0x05 | 0.0.19.4 | compressed, verified(CCOD) |
|  |   |     7 | 0x5ea600 |  0x4f20 |        0x400064 |     0x05 | 0.0.19.4 | compressed, verified(CCOD) |
|  |   |     8 | 0x5ef600 |   0x440 |        0x400065 |     0x05 | 0.0.19.4 | compressed, verified(CCOD) |
```

How to get firmware

1. Rw.exe
2. Get one on vendor site
   - On motherboards with support of Zen arch
   - Also named AMD 17h/18h/19h
   - Chipsets:
     - **X570**
     - **B550**
     - **A520**
     - **X470**
     - **B450**
     - …
   - Go to Support => BIOS => Flash
3. I created repo for firmwares with amd psp =>



Parsing 101

PSPTool →

```
3 | 0x1000000 |      0x0 | ?   FW_GEC~0x61
4 | 0x10e9000 | 0x28000 | FW_INVALID~0x63
5 | 0x1317200 |   0x4e20 |        0x100064
```

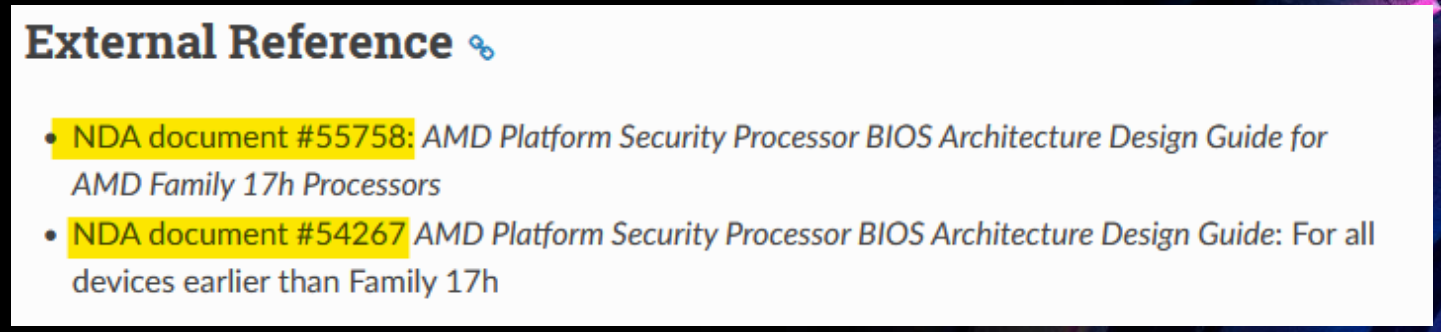
What I expected →

```
[>] BHD_Type=0xb0062
BHD_entry_len=4194336 (0x400020)
BHD_entry_address=0x1e02000
BHD_Enrty_Undf_blob=b'\x00\x00\x00\x00\xe0\xff\xbf\t\x00\x00\x00\x00'
[!] ZLib header found, trying do decompress
[+] Decompressed 4194336 bytes to ./flash_dump.bin.BHD_dump/BHD_0x131100

[>] BHD_Type=0x63
BHD_entry_len=163840 (0x28000)
BHD_entry_address=0x10e9000
BHD_Enrty_Undf_blob=b'\x00\x00\x00\x00\xff\xff\xff\xff\xff\xff\xff\xff'
[+] Dumped 163840 bytes to ./flash_dump.bin.BHD_dump/BHD_0x1311000/BHD_E
```

Houston we have a problem

Can I get some info?

1) CoreBoot (?)
2) PSPTool team repositories, talks and whitepapers.

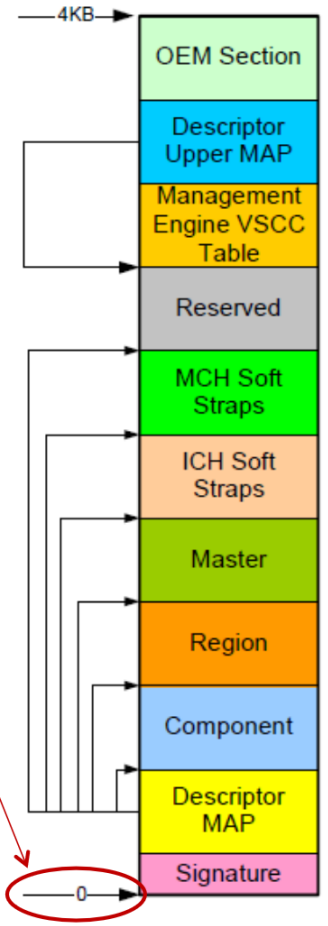


SPI Flash memory struct

Source: [Open Security Training // Intro BIOS // Day 2.]

- This "Flash Descriptor" structure is what's read by the ICH/PCH in order to populate and expose the information via RO registers (like FREG0) in SPIBAR

- **Signature 0FF0A55Ah** denotes the device has a valid descriptor and is therefore operating in Descriptor mode.
- Signature offset is located at 0 on ICH8, ICH9, and ICH10
- In PCH it has been moved to 0x10 and bytes 0 thru 0x0F are Reserved



Pictured: ICH 10 Flash Descriptor

FET

Firmware Entry Table

- Starts with 0xAA55AA55 bytes
- End on 0x0055FFFF bytes
- *Usually* located on offset 0x00020000
  - But not in every firmware

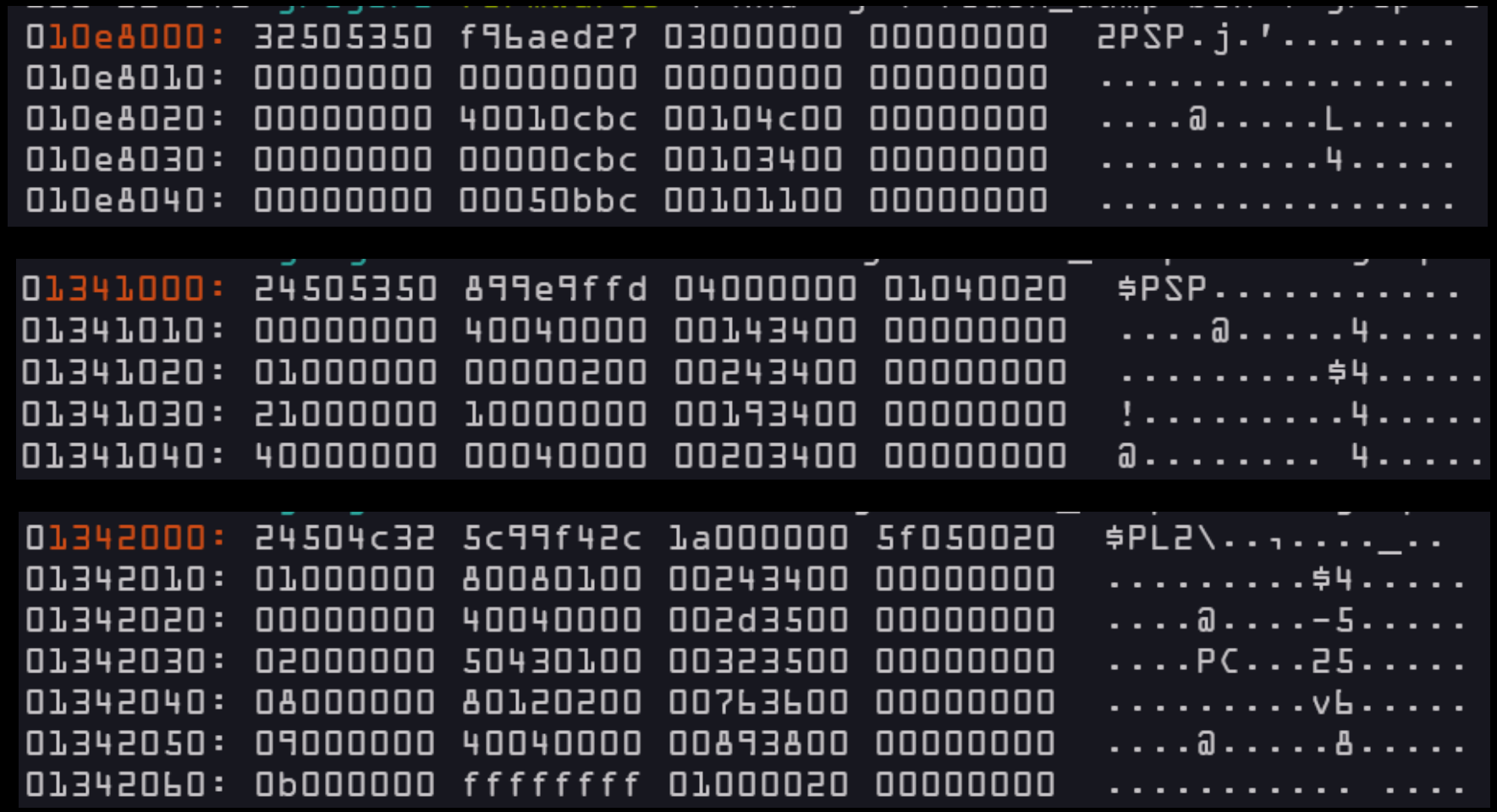
- Describes standard firmware addresses
- Describes sections offsets

```c
#include <stdint.h>

struct FET {
    uint32_t magic_start;            //AA55AA55h
    uint32_t IMC_addr;
    uint32_t Ethernet_addr;
    uint32_t xHCI_firmware_addr;
    uint32_t psp_bios_dir_tables[14]; // ???
    uint32_t magic_end;              //0055FFFFh

};
```

```
01020000: aa55aa55 00000000 00000000 00000000  .U.U............
01020010: 00000000 00800e00 00000000 00000000  ................
01020020: 00103100 feffffff 00880e00 ffffffff  ..1.............
01020030: 00000000 00000000 00106400 ffffffff  ..........d.....
01020040: ffffffff ffffffff 0055ffff ffffffff  .........U......
```

PSP sections

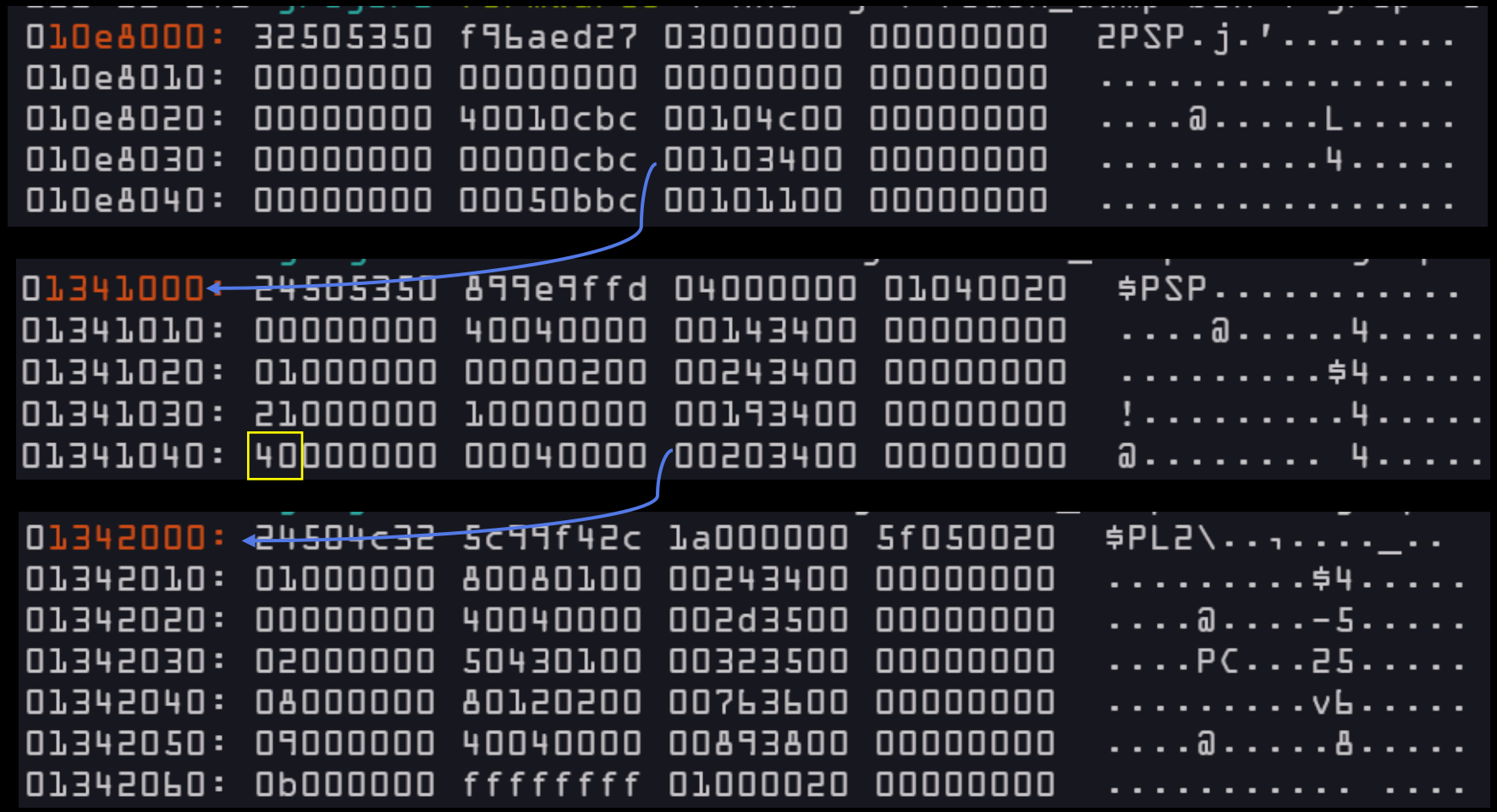
2PSP, $PSP, $PL2




```c
#include <stdint.h>

struct PSP_Header{
    uint32_t signature;
    uint32_t checksum;
    uint32_t entries_count;
    uint32_t unfd_blob;
}

struct PSP_Entry
{
   uint32_t type;
   uint32_t size;
   uint32_t offset;
   uint32_t undf_blob;
}
```

PSP sections

2PSP, $PSP, $PL2




```c
#include <stdint.h>

struct PSP_Header{
    uint32_t signature;
    uint32_t checksum;
    uint32_t entries_count;
    uint32_t unfd_blob;
}

struct PSP_Entry
{
   uint32_t type;
   uint32_t size;
   uint32_t offset;
   uint32_t undf_blob;
}
```

Bios sections

$BHD, 2BHD, $BL2

- $BHD = BIOS Header Directory
- Has addresses of some x86 boot modules. For example, FV Volume with PEICore.
- Has 2-level boot process
  - First $BHD sections
  - Second 2BHD=>$BHD=>$BL2

```c
#include <stdint.h>

struct BHD_Header{
    uint32_t signature;
    uint32_t checksum;
    uint32_t entries_count;
    uint32_t undf_blob;
}

struct BHD_Entry
{
   uint32_t type;
   uint32_t size;
   uint32_t offset;
   uint32_t undf_blob[3];
}
```

```
[16:22:21] greg0r0:firmwares $ xxd -g 4 flash_dump.bin | grep -i -A 20 "1311000:"
01311000: 24424844 a8b0ef06 16000000 30040020  $BHD........0..
01311010: 68000001 00400000 00203100 00000000  h....@... 1.....
01311020: ffffffff ffffffff 60000001 00100000  ........`.......
01311030: 00603100 00000000 ffffffff ffffffff  .`1.............
01311040: 68008001 e8010000 00703100 00000000  h........p1.....
01311050: ffffffff ffffffff 61000000 00000000  ........a.......
01311060: 00000000 00000000 0000200a 00000000  .......... .....
01311070: 62000b00 20004000 0020e000 00000000  b... .@.. ......
```

Bios sections

$BHD, 2BHD, $BL2

- $BHD = BIOS Header Directory
- Has addresses of some x86 boot modules. For example, FV Volume with PEICore.
- Has 2-level boot process
  - First $BHD sections
  - Second 2BHD=>$BHD=>$BL2

```c
#include <stdint.h>

struct BHD_Header{
    uint32_t signature;
    uint32_t checksum;
    uint32_t entries_count;
    uint32_t undf_blob;
}

struct BHD_Entry
{
   uint32_t type;
   uint32_t size;
   uint32_t offset;
   uint32_t undf_blob[3];
}
```

```
[16:22:21] greg0r0:firmwares $ xxd -g 4 flash_dump.bin | grep -i -A 20 "1311000:"
01311000: 24424844 a8b0ef06 16000000 30040020  $BHD........0..
01311010: 68000001 00400000 00203100 00000000  h....@... 1.....
01311020: ffffffff ffffffff 60000001 00100000  ........`.......
01311030: 00603100 00000000 ffffffff ffffffff  .`1.............
01311040: 68008001 e8010000 00703100 00000000  h........p1.....
01311050: ffffffff ffffffff 61000000 00000000  ........a.......
01311060: 00000000 00000000 0000200a 00000000  .......... .....
01311070: 62000b00 20004000 0020e000 00000000  b... .@.. ......
```

More BIOS sections

```
010e8800: 32424844 079ab24f 02000000 00000000  2BHD...O........
010e8810: 00000000 00000000 00000000 00000000  ................
010e8820: 00000000 40010cbc 00106200 00000000  ....@.....b.....
010e8830: 00000000 00000cbc 00104a00 00000000  ..........J.....
010e8840: ffffffff ffffffff ffffffff ffffffff  ................
010e8850: ffffffff ffffffff ffffffff ffffffff  ................
010e8860: ffffffff ffffffff ffffffff ffffffff  ................
```

```
01621000: 24424844 d4480f08 01000000 01040020  $BHD.H.........
01621010: 70000000 00040000 00206200 00000000  p........ b.....
```

```
01622000: 24424c32 c2ec896c 0b000000 1f040020  $BL2...l....... 
01622010: 68000000 00400000 00306200 00000000  h....@...0b.....
01622020: ffffffff ffffffff 60000000 00100000  ........`.......
01622030: 00706200 00000000 ffffffff ffffffff  .pb.............
01622040: 68008000 28020000 00806200 00000000  h...(.....b.....
01622050: ffffffff ffffffff 61000000 00000000  ........a.......
01622060: 00000000 00000000 0000200a 00000000  .......... .....
01622070: 62000b00 20004000 0020e000 00000000  b... .@.. ......
```

```c
#include <stdint.h>

struct s_2BHD_Header
{
   uint32_t signature;
   uint32_t checksum;
   uint32_t entries_count;
   uint32_t undf_blob[5];
}

struct s_BHD_Header
{
    uint32_t signature;
    uint32_t checksum;
    uint32_t entries_count;
    uint32_t undf_blob;
}

struct s_BHD_Entry
{
    uint32_t type;
    uint32_t cbc;
    uint32_t bhd_offset;
    uint32_t undf_blob;
}

struct s_BL2_Entry
{
   uint32_t type;
   uint32_t size;
   uint32_t offset;
   uint32_t undf_blob[3];
}
```

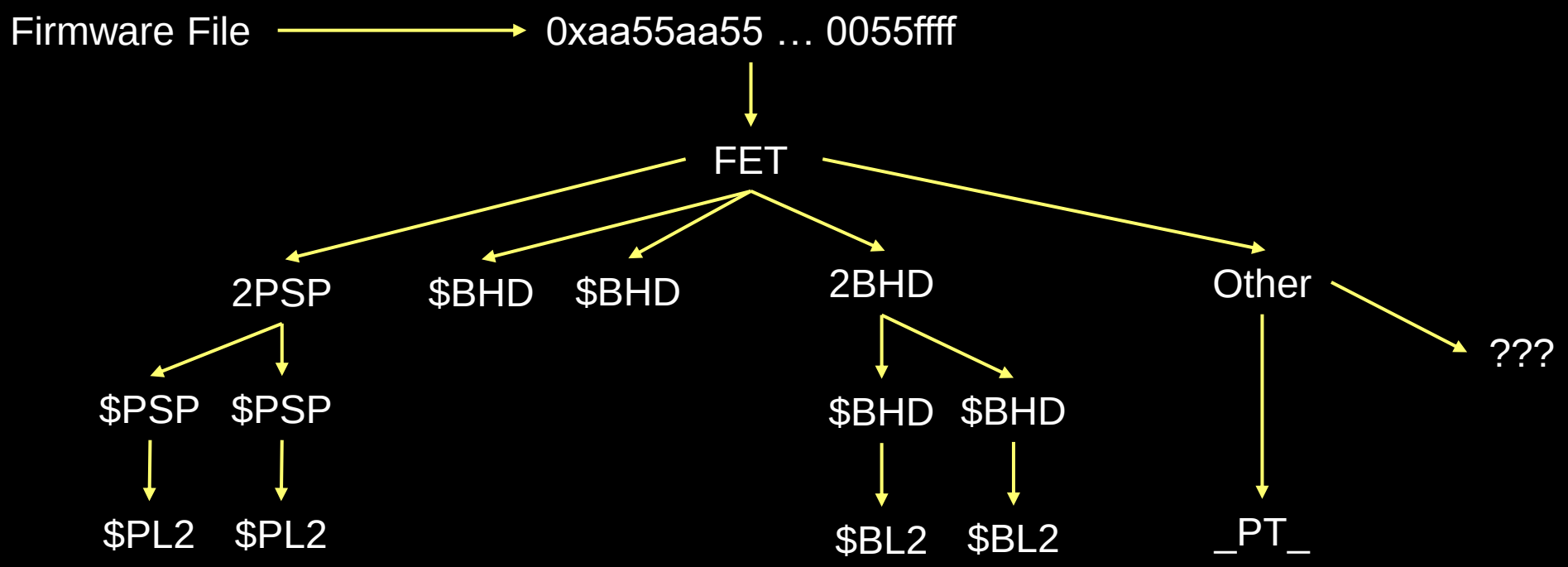
Some other modules

_PT_ for example

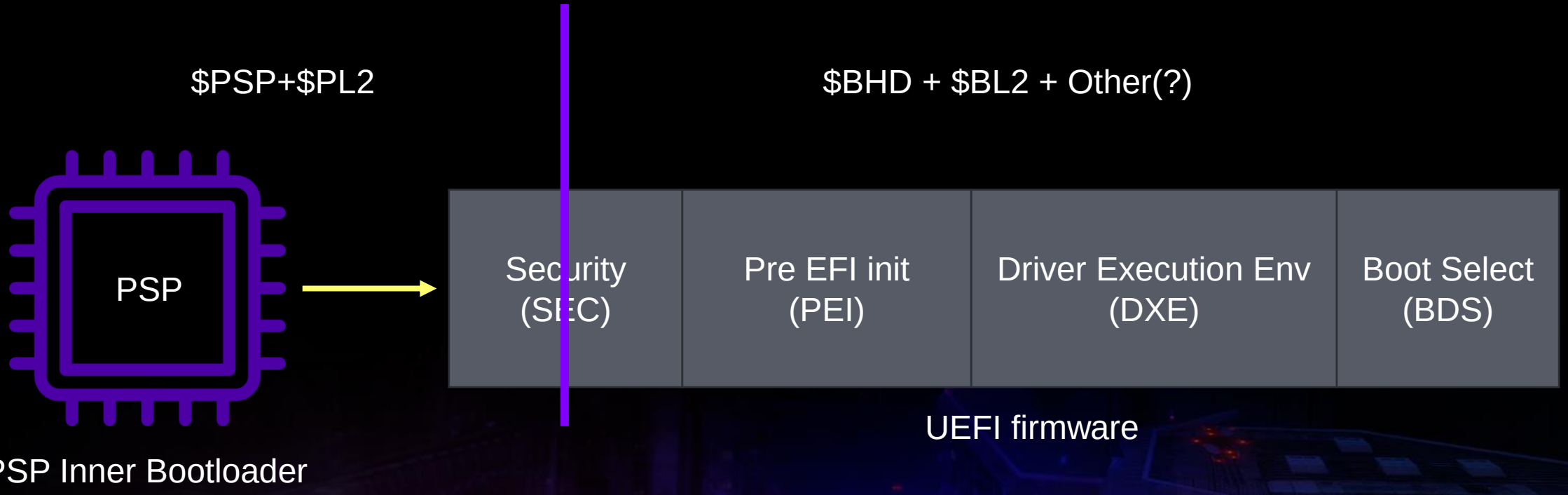
- Vendor-specific?

```
--
01641000: 5f505545f 0cc000100 c2e3a800 02012502  _PT_..........%.
01641010: b779c0000 000000000 000000000 000000002  g...............
01641020: a5bf00000 000000000 000000000 000000000  ................
01641030: 000000000 000000000 000000000 000000000  ................
01641040: 000000000 000000000 000000000 000000000  ................
01641050: 000000000 000000000 000000000 000000000  ................
01641060: 000000000 000000000 000000000 000000000  ................
01641070: 000000000 000000000 000000000 000000000  ................
01641080: 000000000 000000000 000000000 21060490  ............!...
01641090: 018100033 333038841 5f465700 0000a0fb  ...3308A_FW.....
016410a0: 120179b9 000302001 09e475a2 b075a35e  ..y.......u..u.^
016410b0: 75930078 047901a0 02f0a3d8 fcd9fae4  u..x.y..........
016410c0: 75a2b375 a35e7593 00780379 017a01a0  u..u.^u..x.y.z..
016410d0: 02f0a3d8 fcd9fada f87ab57b 5e7c0075  .........z.{^|.u
016410e0: a25d75a3 ad759300 78fa7901 75f001a0  .]u..u..x.y.u...
016410f0: 1de493a3 ad82ae83 af938a82 8b838c93  ................
01641100: f0a3aa82 ab83ac93 8d828e83 8f93d8e1  ................
01641110: d9dfd5f0 dc758200 75830d75 93007400  .....u..u..u..t.
```

Parsing scheme



Boot stages


$PSP+$PL2
$BHD + $BL2 + Other(?)
PSP
Security (SEC)
Pre EFI init (PEI)
Driver Execution Env (DXE)
Boot Select (BDS)
UEFI firmware
PSP Inner Bootloader

PoC 1/2

Github repo

```
                BHD_entry_len=3200 (0xc80)
                BHD_entry_address=0x1335200
                BHD_Enrty_Undf_blob=b'\x00\x00\x00\x00\xff\xff\xff\xff\xff\xff\xff\xff'
                [+] Dumped 3200 bytes to ./flash_dump.bin.BHD_dump/BHD_0x1311000/BHD_Entry_1335200_c80.bin

                [>] BHD_Type=0x200066
                BHD_entry_len=3200 (0xc80)
                BHD_entry_address=0x1335f00
                BHD_Enrty_Undf_blob=b'\x00\x00\x00\x00\xff\xff\xff\xff\xff\xff\xff\xff'
                [+] Dumped 3200 bytes to ./flash_dump.bin.BHD_dump/BHD_0x1311000/BHD_Entry_1335f00_c80.bin

[11:48:55] gregOrO:firmwares $ open .
[11:49:04] gregOrO:firmwares $ tree flash_dump.bin.BHD_dump
flash_dump.bin.BHD_dump
└── BHD_0x1311000
    ├── BHD_Entry_1000000_0.bin
    ├── BHD_Entry_10e9000_28000.bin
    ├── BHD_Entry_1312000_4000.bin
    ├── BHD_Entry_1316000_1000.bin
    ├── BHD_Entry_1317000_1e8.bin
    ├── BHD_Entry_1317200_4e20.decompress.bin
    ├── BHD_Entry_131c100_460.decompress.bin
    ├── BHD_Entry_131c600_4f20.decompress.bin
    ├── BHD_Entry_1321600_440.decompress.bin
    ├── BHD_Entry_1321b00_15c0.bin
    ├── BHD_Entry_1323100_15c0.bin
    ├── BHD_Entry_1324700_15c0.bin
    ├── BHD_Entry_1326000_4000.bin
    ├── BHD_Entry_132a000_1000.bin
    ├── BHD_Entry_132b000_1e8.bin
    ├── BHD_Entry_132b200_4990.decompress.bin
    ├── BHD_Entry_132fc00_460.decompress.bin
    ├── BHD_Entry_1330100_4be0.decompress.bin
    ├── BHD_Entry_1334d00_440.decompress.bin
    ├── BHD_Entry_1335200_c80.bin
    ├── BHD_Entry_1335f00_c80.bin
    └── BHD_Entry_1e02000_400020.decompress.bin
```

UEFITool NE alpha 60 (Jul 13 2022) - BHD_Entry_1e02000_400020.decompress.bin

File Action View Help

Structure

Name of Node	Action	Type	Subtype
UEFI image		Image	UEFI
Padding		Padding	Non-empty
44A2D731-D551-4594-93B4-BD2B60351E0E		Volume	FFSv2
ApprioriPei		File	Freeform
PKeyFileSha256Guid		File	Freeform
RomLayoutPei		File	PEI module
PeiCore		File	PEI core
CapsulePei		File	PEI module
DxeIplPei		File	PEI module
CpuIoPei		File	PEI module
NCT6683DPeiInit		File	PEI module
CpuPei		File	PEI module
4DF75915-6C88-4EF4-995D-834D866DBD…		File	PEI module
PciTableInit		File	PEI module
56FC768F-B603-4454-8083-BFE98D6084…		File	PEI module
168E8FFC-B8DE-48E4-9CFC-D8A667ABEF…		File	PEI module
E847BD26-1DC5-4DE0-BF2F-254F61A5B4…		File	PEI module
SBPEI		File	PEI module
56CBE66A-BF3C-48F9-B9DE-8DA42F9060…		File	PEI module

Parser FIT Security Search Builder

PoC 2/2

Github repo

```
[+] Possible tables addresses:
    0xe7000 -> b'2PSP'
    0x240000 -> b'$BHD'
    0x3af000 -> b'$BHD'
    0x5df000 -> b'$BHD'
    0xffffe -> b'\xff\xff'
    0xe7800 -> b'2BHD'
    0xfffff -> b'\xff'
    0x90f000 -> b'_PT_'
    0x92f000 -> b'_PT_'
    0xfffff -> b'\xff'
    0xfffff -> b'\xff'
    0xfffff -> b'\xff'
[+] bhd_address=0x240000
        bhd_header=b'$BHD'
        bhd_crc=3679094445
        bhd_number_of_entries=17
        bhd_undifined_blob=b' \x04\x00\x00'
        Location=./AB35PR47.20.BHD_dump/BHD_0x240000
```

```
[+] bhd_address=0x5df000
        bhd_header=b'$BHD'
        bhd_crc=416760829
        bhd_number_of_entries=22
        bhd_undifined_blob=b'0\x04\x00 '
        Location=./AB35PR47.20.BHD_dump/BHD_0x5df000
                [>] BHD_Type=0x1000068
                BHD_entry_len=16384 (0x4000)
                BHD_entry_address=0x5e0000
                BHD_Enrty_Undf_blob=b'\x00\x00\x00\x00\x
                [+] Dumped 16384 bytes to ./AB35PR47.20.
```

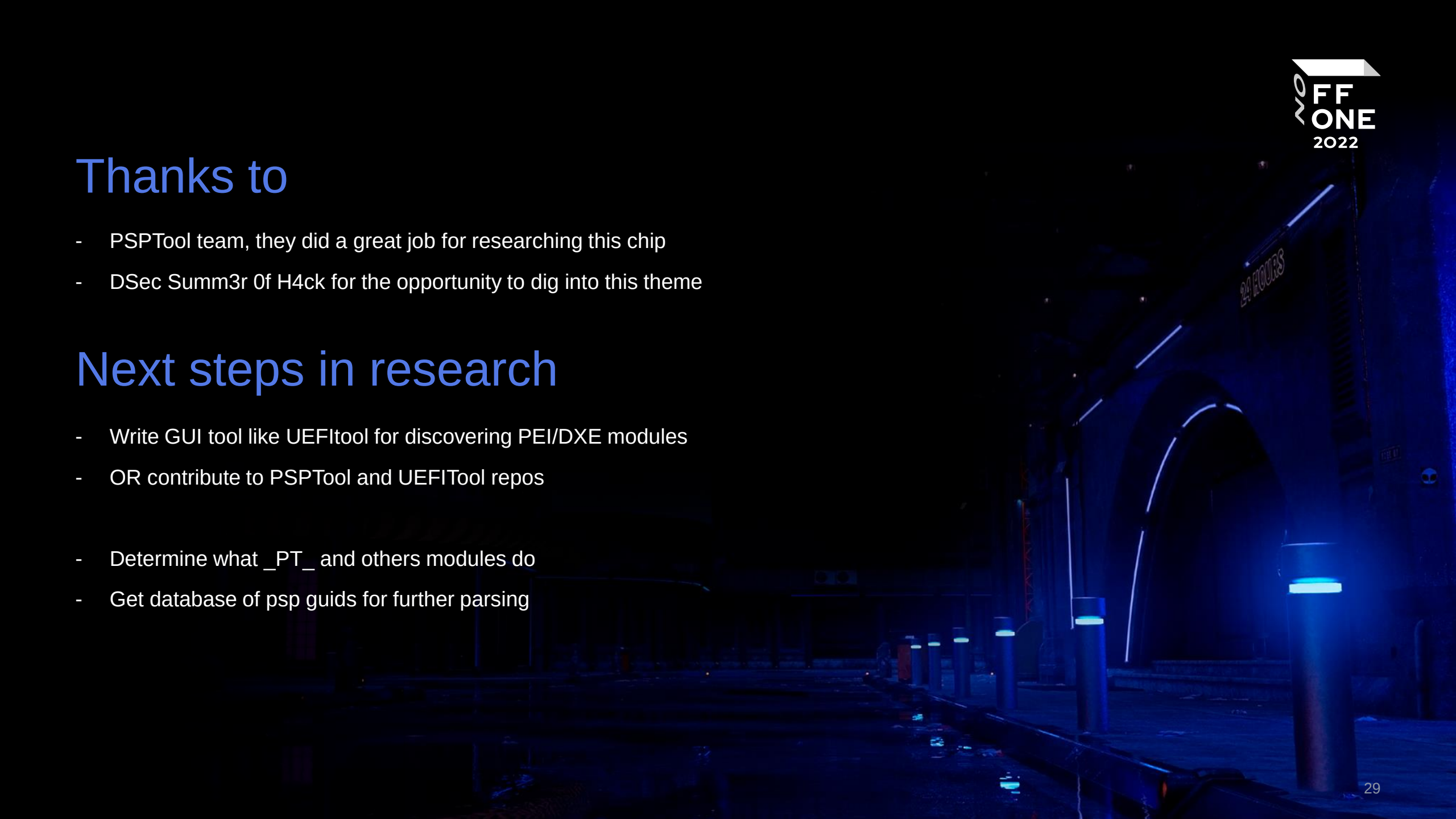
Thanks to

- PSPTool team, they did a great job for researching this chip
- DSec Summ3r 0f H4ck for the opportunity to dig into this theme

Next steps in research

- Write GUI tool like UEFItool for discovering PEI/DXE modules
- OR contribute to PSPTool and UEFITool repos

- Determine what _PT_ and others modules do
- Get database of psp guids for further parsing




OFF
ONE
2022