Advanced x86:

BIOS and System Management Mode Internals UEFI Reverse Engineering

Xeno Kovah && Corey Kallenberg LegbaCore, LLC



All materials are licensed under a Creative Commons "Share Alike" license. http://creativecommons.org/licenses/by-sa/3.0/

You are free:



to Share — to copy, distribute and transmit the work

to Remix - to adapt the work

Under the following conditions:

Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).



Share Alike — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

Attribution condition: You must indicate that derivative work

"Is derived from John Butterworth & Xeno Kovah's 'Advanced Intel x86: BIOS and SMM' class posted at http://opensecuritytraining.info/IntroBIOS.html" 2

And the people yelled:



Simmer down y'all. I reckon what ya best do is...



- Find some subset of interesting code
 - You *could* search for B/D/F address of interest
- But better is to narrow down what you want to look at, by slicing and dicing the firmware filesystem with one of:
- EFIPWN
 - <u>https://github.com/G33KatWork/EFIPWN</u>
- UEFITool
 - <u>https://github.com/LongSoft/UEFITool</u>
- UEFI Firmware Parser
 - <u>https://github.com/theopolis/uefi-firmware-parser</u>
 - We're not going to cover this for now, since I haven't built it on Windows yet

Firmware Storage



- UEFI utilizes the physical flash device as a storage repository
- Comprised of 4 basic components:
 - Firmware Device
 - Firmware Volume
 - Firmware File System
 - Firmware Files

Firmware Volumes (FVs)



- A Firmware Device is a physical component such as a flash chip.
- We mostly care about Firmware Volumes (FVs)
- We often see separate volumes for PEI vs. DXE code
 - And occasional "duplicate" volumes for restore-from-backup
- FVs can contain multiple firmware volumes (nesting)
- FVs are organized into a Firmware File System (FFS)
- The base unit of a FFS is a file

Firmware File System (FFS)



- FVs are organized into a Firmware File System (FFS)
- A FFS describes the organization of files within the FV
- The base unit of a FFS is a file
- Files can be further subdivided into sections

Firmware Files





- We mostly care about file sections that are in PE (Portable Executable) file format
 - Alternatively can be a TE (Terse Executable) which is a "minimalist" PE

Oh, how interesting! My BIOS uses "Windows" executables? I know how to analyze those!



Yay Standardization!

A standard way of putting together the firmware filesystem, with nice human readable names, makes it easier for me to find my way around to the likely locations I want to attack A standard way of putting together the firmware filesystem, with nice human readable names, makes it easier for me to understand the context of what might have been attacked if I see a difference there



UEFITool/UEFIExtract

• The best and most up-to-date firmware filesystem parser

Go to File->Open and select the file dump (I selected the "e6430A03.bin")



This volume holds a bunch of PEIMs (and the one above it a bunch of DXE drivers.)

Structure						Information
Name	Action	Туре	Subtype	Text	*	FileSystem GUID:
⊿ Intel image		Image	Intel			7A9354D9-0468-444A-81C
Descriptor region		Region	Descriptor			E-08F617D890DF
GbE region		Region	GbE			Revision: 1
ME region		Region	ME		=	Attributes: ffff8eff
▲ BIOS region		Region	BIOS			Erase polarity: 1
▷ 7A9354D9-0468-444A-81C		Volume				Header size: 0048
▷ 7A9354D9-0468-444A-81C		Volume				
Padding		Padding				
▷ 7A9354D9-0468-444A-81C		Volume				
▲ 7A9354D9-0468-444A-81C		Volume	Boot			
> 3B42EF57-16D3-44CB-8		File	PEI module	MemoryInit		
CA9D8617-D652-403B-B		File	PEI module	TxtPei		
◊ ØD1ED2F7-E92B-4562-9		File	PEI module	CRBPEI		
A27E7C62-249F-4B7B-B		File	PEI module	DellFlashUpdatePei		
1D88C542-9DF7-424A-A		File	PEI module	WdtPei		
▷ 92685943-D810-47FF-A		File	PEI core	CORE_PEI		
▷ 01359D99-9446-456D-A		File	PEI module	CpuInitPei		
C866BD71-7C79-4BF1-A		File	PEI module	CpuS3Peim		
8B8214F9-4ADB-47DD-A		File	PEI module	SmmBasePeim		
▷ ØAC2D35D-1C77-1033-A		File	PEI module	CpuPolicyPei		
1555ACF3-BD07-4685-B		File	PEI module	CpuPeiBeforeMem		
> 2BB5AFA9-FF33-417B-8		File	PEI module	CpuPei		
C1FBD624-27EA-40D1-A		File	PEI module	SBPEI		
> 333BB2A3-4F20-4C8B-A		File	PEI module	AcpiPlatformPei		
▷ ØF69F6D7-ØE4B-43A6-B		File	PEI module	WdtAppPei	-	
•					P	

"AmgTcgPlatformPeiBeforeMem" is the PEIM we're going to be interested in shortly

To get a well-formed PE file, we extract it by right clicking and selecting "Extract body"

Structure								Inform	nation
Name	Action	Туре	Subtype	Text			*	Туре	: 10
 12345678-930A-4A95-A 6B844C5B-6B75-42CA-8 9B3F28D5-10A6-46C8-B E9312938-E56B-4614-A PEI dependency sect 		File File File File Section	PEI module PEI module PEI module PEI module PEI dependency	TogPe TogPe AmiTo AmiTo	eiAfterMem eiplatform gplatformPeiAf gplatformPeiBe	fterMem eforeMem		Size	: 000ba0
PE32+ image section		Section	PE32+ image	_				1	
User interface sect… > 0DCA793A-EA96-42D8-B… > 3FD1D3A2-99F7-420B-B…		Section File File	User interface Freeform Freeform		E <u>x</u> tract as is Extract <u>b</u> ody	Ctrl+E Ctrl+Shift+	E		
▷ ØA602C5B-05A0-40C4-9 ▷ 336CDDEA-AB28-4C4C-9		File File	PEI module PEI module	C C	<u>R</u> ebuild	Ctrl+Space			
 FB8415B7-EA7E-4E6D-9 DE8A5A2C-D788-47FB-A 4AD92749-732E-445A-B F665C81D-EFDE-4B5F-8 		File File File File	PEI module PEI module PEI module PEI module	C C C	Insert <u>i</u> nto Insert b <u>e</u> fore Insert <u>a</u> fter	Ctrl+I Ctrl+Alt+I Ctrl+Shift+	I		
 E9B60F94-7A0B-48CD-9 B178E5AA-0876-420A-B E9A60F94-7A8B-45BA-9 70E65212-F3AD-495F-B 		File File File File	PEI module PEI module PEI module PEI module	C V C	Rep <u>l</u> ace as is Replace b <u>o</u> dy	Ctrl+R Ctrl+Shift+	R		
> 5924BE03-9DD8-4BAB-8 > 81F0BCF2-F1AD-4DDE-9		File File	PEI module PEI module	C	Re <u>m</u> ove	Ctrl+Del			

UEFIExtract is a simple command line tool that just dumps everything out to the filesystem instead of making it navigable from a GUI

C:\Users\student\Desktop>UEFIExtract.exe UEFIExtract 0.2

Usage: uefiextract imagefile

C:\Users\student\Desktop>UEFIExtract.exe C:\Users\student\Desktop\e6430A03.bin parseRegion: ME region version is unknown, it can be damaged parseVolume: 17088572-377F-44EF-8F4E-B09FFF46A070, unaligned file

🐌 🕨 еб430А03	3.bin.dump 🕨						
✓ Include in	library 👻 Share with 👻						
rites	Name						
sktop	퉬 0 Descriptor region						
wnloads	퉬 1 GbE region						
ent Places	퉬 2 ME region						
	퉬 3 BIOS region						
ries	🖻 body.bin						
cuments	📋 info.txt						

The metadata will be stored off to the side in .txt files

This is good if you want to search all the files for a pattern. But it's less easy to navigate if you want to just get a single file (in that case just use

the GUI)

 Include in 	library 🔻 Share with 👻 Burn New folder			
ites	Name	Date modified	Туре	Size
ktop	퉬 0 IntelSaGopDriver	6/21/2014 7:11 PM	File folder	
nloads	퉬 1 IntellvbGopDriver	6/21/2014 7:11 PM	File folder	
ent Places	퉬 2 IntelSnbGopDriver	6/21/2014 7:11 PM	File folder	
	퉬 3 TcgDxe	6/21/2014 7:11 PM	File folder	
es	퉬 4 CpuDxe	6/21/2014 7:11 PM	File folder	
uments	퉬 5 FileSystem	6/21/2014 7:11 PM	File folder	
ic	퉬 6 DAC2B117-B5FB-4964-A312-0DCC77061B9B	6/21/2014 7:11 PM	File folder	
ures	퉬 7 9221315B-30BB-46B5-813E-1B1BF4712BD3	6/21/2014 7:11 PM	File folder	
os	🐌 8 CORE_DXE	6/21/2014 7:11 PM	File folder	
	🌗 9 BindingsDxe	6/21/2014 7:11 PM	File folder	
uter	퉬 10 DellFlashIoDxe	6/21/2014 7:11 PM	File folder	
l Disk (C:)	퉬 11 DellEcConfigDxe	6/21/2014 7:11 PM	File folder	
ovable Disk (E:)	퉬 12 DellTagsConfig	6/21/2014 7:11 PM	File folder	
	🌗 13 DxeEcIoDriver	6/21/2014 7:11 PM	File folder	
ork	퉬 14 SpiPartAtmelDxe-Edk1_06-Pi1_0-Uefi2_1	6/21/2014 7:11 PM	File folder	
	퉬 15 SpiPartEonDxe-Edk1_06-Pi1_0-Uefi2_1	6/21/2014 7:11 PM	File folder	
	퉬 16 SpiPartMicronDxe-Edk1_06-Pi1_0-Uefi2_1	6/21/2014 7:11 PM	File folder	
	퉬 17 SpiPartMxicDxe-Edk1_06-Pi1_0-Uefi2_1	6/21/2014 7:11 PM	File folder	
	퉬 18 SpiPartPromJetDxe-Edk1_06-Pi1_0-Uefi2_1	6/21/2014 7:11 PM	File folder	
	퉬 19 SpiPartSstDxe-Edk1_06-Pi1_0-Uefi2_1	6/21/2014 7:11 PM	File folder	
	퉬 20 SpiPartStMicroDxe-Edk1_06-Pi1_0-Uefi2_1	6/21/2014 7:11 PM	File folder	
	퉬 21 SpiPartWinbondDxe-Edk1_06-Pi1_0-Uefi2_1	6/21/2014 7:11 PM	File folder	
	퉬 22 DellTagsDxe-Edk1_06-Pi1_0-Uefi2_1	6/21/2014 7:11 PM	File folder	
	23 SpiControllerDxe	6/21/2014 7:11 PM	File folder	

- So as we know, Copernicus provides us the full dump of the BIOS flash
 - Repeated from previous: Copernicus maintains the FLA offsets for each region by reading even those which the CPU/BIOS master has no permissions to read (like the Management Engine, typically)
 - <u>Any</u> BIOS dump should work as long as it's a UEFI BIOS (structured for better parsing)
- Comparing BIOS dumps over a period of time can provide change detection
- How this differs from observing the TPM PCR registers is this:
- When a PCR tells you a change has been made, it cannot tell you where the change has been made
- Bios_diff.py uses the decomposition capability of EFIPWN to tell us the particular module(s) in which the change(s) is/are located

C:\Tools\CoP>python bios_diff.py -dpan -e ..\EFIPWN C:\uefi_bins\efi.bin C:\uefi_bins\efix.bin -o C:\

- This script uses EFIPWN to parse and diff the modules between two BIOS dumps
- EFIPWN decomposes the BIOS into its firmware volumes (FVs) and then decomposes each into the files/modules that comprise it
- In this example we're analyzing an earlier "known-good" BIOS with one which we notice has changed
 - We took a known good and purposefully made a small change in the "suspicious" one

C:\Tools\CoP>python bios_diff.py -dpan -e C:\EFIPWN-sam\EFIPWN "F:\UEFI Binaries\e6430A03.bin" "F:\U FI Binaries\e6430A03 haxed.bin"-o . Differing file found: .\e6430A03.bin\fv3\e9312938-e56b-4614-a252-cf7d2f377e26\PE32_73 (AmiTcgPlatformPeiBeforeMem) 7 unique bytes out of 2976 1036,1042 PE Information: Section .text RVA Øx40c VA 0xffe6d090 .\e6430A03_haxed.bin\fv3\e9312938-e56b-4614-a252-cf7d2f377e26\PE32_73 (AmiTcgPlatformPeiBeforeMem) 7 unique bytes out of 2976 1036.1042 PE Information: Section .text RUA 0x40c VA 0xffe6d090

- The script has found a difference located in firmware volume 3
- Some files/modules have user-friendly names and if this is the case the script outputs this name
- AmiTcgPlatformPeiBeforeMem
- Tcg could be Trusted Computing Group and this is likely a PEIM that executes before memory is established

C:\Tools\CoP>python bios_diff.py -dpan -e C:\EFIPWN-sam\EFIPWN "F:\UEFI Binaries\e6430A03.bin" "F:\U FI Binaries\e6430A03 haxed.bin"-o . Differing file found: .\e6430A03.bin\fv3\e9312938-e56b-4614-a252-cf7d2f377e26\PE32_73 (AmiTcgPlatformPeiBeforeMem) 7 unique bytes out of 2976 1036.1042PE Information: Section .text RVA Øx40c VA 0xffe6d090 .\e6430A03_haxed.bin\fv3\e9312938-e56b-4614-a252-cf7d2f377e26\PE32_73 (AmiTcgPlatformPeiBeforeMem) 7 unique bytes out of 2976 1036.1042 PE Information: Section .text RUA 0x40c VA Øxffe6d090

- If more than 1 diff is found they will all be listed here in this manner
- In this case it is just a single diff found
- Diff was found at offset 0x40C in the file "AmiTcgPlatformPeiBeforeMem"
- The length of the diff is 7 bytes

C:\Tools\CoP>python bios_diff.py -dpan -e C:\EFIPWN-sam\EFIPWN "F:\UEFI Binaries\e6430A03.bin" "F:\U FI Binaries\e6430A03_haxed.bin"-o . Differing file found: .\e6430A03.bin\fv3\e9312938-e56b-4614-a252-cf7d2f377e26\PE32_73 (AmiTcgPlatformPeiBeforeMem) 7 unique bytes out of 2976 1036,1042 PE Information: Section .text RVA Øx40c VA 0xffe6d090 .\e6430A03_haxed.bin\fv3\e9312938-e56b-4614-a252-cf7d2f377e26\PE32_73 (AmiTcgPlatformPeiBeforeMem) 7 unique bytes out of 2976 1036.1042 PE Information: Section .text RUA 0x40c VA 0xffe6d090

- Files in the UEFI Flash File System are in the PE format (or TE [Terse Executable], which is a minimalist PE file)
 - But still PE
- For this reason we can identify whether diffs are located in the .data or .text (code) sections of a given file
 - In this case the change occurs in the code section

C:\Tools\CoP>python bios_diff.py -dpan -e C:\EFIPWN-sam\EFIPWN "F:\UEFI Binaries\e6430A03.bin" "F:\U FI Binaries\e6430A03_haxed.bin"-o . Differing file found: .\e6430A03.bin\fv3\e9312938-e56b-4614-a252-cf7d2f377e26\PE32_73 (AmiTcgPlatformPeiBeforeMem) 7 unique bytes out of 2976 1036,1042 PE Information: Section .text RVA 0x40c VA Øxffe6d090-.\e6430A03_haxed.bin\fv3\e9312938-e56b-4614-a252-cf7d2f377e26\PE32_73 (AmiTcgPlatformPeiBeforeMem) 7 unique bytes out of 2976 1036.1042 PE Information: Section .text RUA 0x40c VA 0xffe6d090

- Also from the PE file we can get the Virtual Address of the change in the file
- From this we can derive both the Flash Linear Address of the change on the serial flash (provided the size of the BIOS region) and therefore its location in mapped high-memory
- The output also identifies the Relative-Virtual Address (RVA), which is the segment offset from the start of the PE file

C:\Tools\CoP>python FI Binaries\e6430A03 Differing file found .\e6430A03.bin\fv3\e 7 unique bytes out o	bios_diff.; _haxed.bin 9312938-e56 f 2976	ру – ' –о 56–4	dpa 614	n —a														
PE Information:	Offset(h)	00	01	02	03	04	05	06	07	80	09	AO	0B	0C	0D	0E	OF	
RUA 0x40c	00A6CC80	A4	0B	00	10	4D	5A	0	00	00	00	00	00	00	00	00	00	¤MZ
Neb430H03 naxed.hin	00A6CC90 00A6CCA0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
	00A6CCB0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
	00A6CCC0	C8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	É
	UUAGCCDU	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

- We can use the VA and RVA information to locate this PE file in the BIOS hex dump
- VA RVA = beginning of PE file
- But first let's convert that VA to a flash linear address:
- FFFF_FFFh FFE6_D090h = 19_2F6Fh
- <.bin size> 19_2F6Fh = BF_FFFh 19_2F6Fh = A6_D090h
- A6_D090h 40C = A6_CC84h

Analyzing UEFI Files with IDA (Search for "MITRE Copernicus Analyzing BIOS Differences with IDA Pro")

Analyzing UEFI Files

C4.	Administrator: Command Prompt		đ	×
CDFDOLPSRV	<pre>>>python bios_diff.py -e EFIPWN e6430A03.bin e6430A03_haxed.bin ifference in file firmwareVolume3\e9312938-e56b-4614-a252-cf7d2f377e26\PE32_ ile Name: AmiTcgPlatformPeiBeforeMem iff #0 fset: 0x40c ength: 0x1 Information ection: .text /A: 0x40c A: 0xffe6d090</pre>	_94		
c:				

- Following our example of finding a "diff" across multiple BIOS, let's find out how to analyze the change using IDA
- This should strike a sharp contrast to trying to analyze a legacy BIOS which does not follow public standards
 - Not to say they don't have internal standards, just that those standards are not public
- The free version of IDA will be adequate for these purposes



- The first step having identified a change between two BIOS dumps is to first locate the specific files in which the change(s) were detected
- In our example, the changes occur in Firmware Volume 3
- Find the directory where EFIPWN decomposed the UEFI binary and go to firmwareVolume3

Analyzing UEFI Files

Administrator: Command Prompt			
C:\cop>python bios_diff.py Difference in file firmware File Name: AmiTcgPlatformPe Diff #0 Offset: 0x40c	-e EFIPWN e6430A03.bin e6430A03 Volume3\e9312938-e56b-4614-a252 iBeforeMem	_haxed.bin -cf7d2f377e26\PE3	2_94 ₌
Length: 0x1	🔑 e9b60f94-7a0b-48cd-9c88-8484526c5719	3/1/2014 3:16 PM	File folder
Section: .text	↓ e4536585-7909-4a60-b ² -o-ecdea6ebfb54	3/1/2014 3:16 PM	File folder
	e9312938-e56b-4614-a252-cf7d2f377e26	3/1/2014 3:16 PM	File folder
VA: UXTTE60090	f665c81d-etde-4b5t-88e8-2160b/48d2b4	3/1/2014 3:16 PM	File folder
0	퉬 fac2efad-8511-4e34-9cae-16a257ba9488	3/1/2014 3:16 PM	File folder
	🐌 fb8415b7-ea7e-4e6d-9381-005c3bd1dad7	3/1/2014 3:16 PM	File folder
	Id236ae7-0791-48c4-b29e-29bdeee1a811	3/1/2014 3:16 PM	File folder

- Inside the firmwareVolume3 directory is a directory listing of GUIDS
- Find the GUID in which this diff was detected
- In this case it is GUID:
 - e9312938-e56b-4614-a252-cf7d2f377e26
- Inside this directory you will find the PE32_94 file which contains the file that has changed
- You can locate both of these files in this manner: the previous one which is assumed to be good, and the new one in which the change has been observed

PE32_94_haxed PE32_94 PE32_94_haxed PE32_94 Offset (h) OO	ile Ed	dit Se	earch	Vie	w A	nalys	sis E	:×tras	s Wi	ndow	?									_ 8 ×
PE32_94_haxed PE32_94 Offset (h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 00000000 4D 5A 0A 00<	e -		Same	9	•	• 16	5	~	AN	SI			h	ex		~				
Offset (h) 00 01 02 03 04 05 04 08 0C 0D 0E 0F 00000000 4D 5A 040 0	E32_9	94_hax	ked	50	PE32	_94														
000000000 \$\mathbf{h}D\$ 5A \$\mathbf{h}X\$ 00 00	fset	(h)	00	01	02	03	04	05	06	07	08	09	OA	ОВ	oc	OD	OE	OF		^
00000010 00	0000	000	4D	5A	00	00	00	00	00	00	00	00	00	00	00	00	00	00]]]Z	
00000020 00	0000	010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000030 00	0000	020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000040 00	0000	030	00	00	00	00	00	00	00	00	00	00	00	00	C8	00	00	00	È	
00000050 00	0000	040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000060 00	0000	050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000070 00	0000	060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000080 00	0000	070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000090 00	0000	080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000000 00	0000	90	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000B0 00	0000	OAO	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000000 00	0000	BO	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000000 54 D3 CA 4F 00 00 00 00 00 00 00 00 00 00 2 21 TÓÊOà.! 000000E0 0B 01 00 00 00 00 00 00 00 00 00 00 00 00	0000	000	00	00	00	00	00	00	00	00	50	45	00	00	4C	01	04	00	PEL	
000000E0 0B 01 00 00 08 00 00 40 01 00	0000	DDO	54	DЗ	CA	4F	00	00	00	00	00	00	00	00	ΕO	00	02	21	TÓÊOà!	
000000F0 0C 04 00 60 02 00 00 00 00 84 CC E6 FF `	0000	DEO	OB	01	00	00	00	08	00	00	40	01	00	00	00	00	00	00		
00000100 20 00	0000	FO	OC	04	00	00	60	02	00	00	60	OA	00	00	84	CC	Ε6	$\mathrm{F}\mathrm{F}$	``Ìæÿ	
00000110 00	0001	100	20	00	00	00	20	00	00	00	00	00	00	00	00	00	00	00		
00000120 00 00 00 00 0B 00 00 00 00 00 00 00 00	0001	10	00	00	00	00	00	00	00	00	AO	OВ	00	00	60	02	00	00	····	
00000130 00 00 00 00 00 00 00 00 00 00 00 00 10 00 0	0001	120	00	00	00	00	OВ	00	00	00	00	00	00	00	00	00	00	00		
00000140 00 00 00 00 00 00 00 00 00 00 00 00 0	0001	130	00	00	00	00	00	00	00	00	00	00	00	00	10	00	00	00		
	0001	L40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000150 00 00 00 00 00 00 00 00 00 00 00 00 0	0001	150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		~
																			Cueronalite .	

- One of the first things you can do upon acquiring both files is to observe them in a hex editor
- HxD allows you to easily perform binary comparisons between 2 files (Analysis > File-Compare > Compare, and then select the 2 files you want to compare)

File Edit Search View Analysis Extra	as Window ?	
📄 🚵 - 📊 🧼 😃 🖬 16	ANSI 🖌 hex 🖌	
8 PE32 94 haxed 1 PE32 94		
🔠 C:\Documents and Settings\Ad	ministrator\Desktop\PE32_94_haxed	
Offset(h) 00 01 02 03 04	1 05 06 07 08 09 0A OB OC OD OB	OF 🗖
000003D0 45 E4 8B 45 F8	3 8B 40 02 89 45 E8 8D 45 FC 50) 6A Ea(Eø(0.%Eè.EüPj
000003E0 OC 8D 45 BO 50) 56 E8 33 00 00 00 83 45 FC 18	6AE°PVè3fEü.j
000003F0 OC 8D 4D EO 51	L FF 75 FC 8B F8 8B 06 FF 50 50) 83MàQÿuü<ø<.ÿPPf
00000400 C4 1C 8B C7 EE	3 02 33 CO 5F 5E C9 C3 <mark>C3</mark> _44 24	108 Ä.<Çë.3À_^ÉÃÃD\$.
00000410 8B 08 68 44 D7	7 E6 FF 50 FF 51 18 59 59LC3 8B	54 < .hD׿ÿPÿQ.YYÃ <t< th=""></t<>
00000420 24 OC 8B 44 24	ł 04 8B 08 56 8B 74 24 14 56 83	C2 \$. <d\$.<.v<t\$.vfå< th=""></d\$.<.v<t\$.vfå<>
00000430 18 52 6A 04 50) FF 51 34 83 C4 10 85 CO 78 11	. 57 .Rj.PÿQ4fÄÀx.W
00000440 8B 3E 8B 74 24	ł 10 83 C7 08 A5 A5 A5 A5 33 C0) 5F <> <t\$.fç.¥¥¥¥3å th="" 🗡<="" 🛛=""></t\$.fç.¥¥¥¥3å>
🗟 C:\Documents and Settings\Ad	ministrator\Desktop\PE32_94	
Offset(h) 00 01 02 03 04	1 05 06 07 08 09 0A OB OC OD OB	OF 🔼
000003D0 45 E4 8B 45 F8	3 8B 40 02 89 45 E8 8D 45 FC 50) 6A EäkEøk@.‰Eè.EüPi
000003E0 OC 8D 45 B0 50	0 56 E8 33 00 00 00 83 45 FC 18	6AE°PVè3fEü.j
000003F0 OC 8D 4D E0 51	L FF 75 FC 8B F8 8B 06 FF 50 50) 83MàQÿuü<ø<.ÿPPf
00000400 C4 1C 8B C7 EE	3 02 33 CO 5F 5E C9 C3 📴 44 24	08 Ä.<Çë.3À ^ÉĂ <d\$.< th=""></d\$.<>
00000410 8B 08 68 44 D7	7 E6 FF 50 FF 51 18 59 59 C3 8B	54 < .hD׿ÿPÿQ.YYÃ <t< th=""></t<>
00000420 24 OC 8B 44 24	ł 04 8B 08 56 8B 74 24 14 56 83	C2 \$. <d\$.<.v<t\$.vfå< th=""></d\$.<.v<t\$.vfå<>
00000430 18 52 61 04 50) FF 51 34 83 C4 10 85 CO 78 11	. 57 .Rj.PÿQ4fÄÀx.W
00000440 8B 3E 8B 74 24	ł 10 83 C7 08 A5 A5 A5 A5 33 C0) 5F <> <t\$.fç.¥¥¥¥3å th="" 🛛="" 🞽<=""></t\$.fç.¥¥¥¥3å>
Offset: 40C Block: 40C-40C	Length: 1	Overwrite

- HxD's file comparison compares each file in parallel and highlights each byte that differs
- It's a quick way to "eyeball" changes which have been detected
- This is less helpful when the file-sizes differ and the area where you want to analyze the change occurs at an offset other than where it usually does

File Edit Search Vi	ew Analysis	s Extras Windo	W ?		
🗋 🚵 • 🔚 🧼	CH 🗗	16 🖌 AM	SI 🖌 🖌	nex 🔽	
BE32_94_haxed	1 PE32 9	4			
6		-			
C: Wocuments	and Settin	ıgs\Administra	itor Wesktop VP E	32_94_haxed	
Offset(h) 00	01 02 0	03 04 05 0	07 08 09 0	A OB OC OD OE OF	~
000003D0 45	E4 8B	45 F8 8B 40	02 89 45 E	8 8D 45 FC 50 6A	Eä <eø<0.‰eè.eüpj< th=""></eø<0.‰eè.eüpj<>
000003E0 0C	8D 45 1	BO 50 56 E8	33 00 00 0	O 83 45 FC 18 6A	E°PVè3fEü.j
000003F0 OC	8D 4D 1	EO 51 FF 75	FC 8B F8 8	B 06 FF 50 50 83	MàQÿuü<ø< .ÿPPf
00000400 C4	1C 8B (C7 EB O2 33	CO 5F 5E C	9 C3 <mark>C3</mark> ,44 24 08	Ă.<Çë.3À_^ÉÃÃD\$.
00000410 8B	08 68 4	44 D7 E6 F1	50 FF 51 1	8 59 59LC3 8B 54	< .hD׿ÿPÿQ.YYÃ< T
00000420 24	OC 8B	44 24 04 81	08 56 8B 7	4 24 14 56 83 C2	\$. <d\$.<.v<t\$.vfå< th=""></d\$.<.v<t\$.vfå<>
00000430 18	52 6A (04 50 FF 5:	34 83 C4 1	O 85 CO 78 11 57	.Rj.PÿQ4fÄÀx.W
00000440 8B	3 E 8 B	74 24 10 83	C7 08 A5 A	5 A5 A5 33 CO 5F	<> <t\$.fç.¥¥¥¥3à th="" 🞽<=""></t\$.fç.¥¥¥¥3à>
C:\Documents	and Settin	ngs∖Administra	tor\Desktop\P6	32_94	
_					
Offset(h) 00	01 02 0	03 04 05 0	07 08 09 0.	A OB OC OD OE OF	<u> </u>
000003D0 45	5 E4 8B	45 F8 8B 40	02 89 45 E	8 8D 45 FC 50 6A	Eä <eø<0.‰eè.eüpj< th=""></eø<0.‰eè.eüpj<>
000003E0 0C	8D 45 1	BO 50 56 E8	33 00 00 0	0 83 45 FC 18 6A	E°PVè3fEü.j
000003F0 0C	8D 4D 1	EO 51 FF 75	FC 8B F8 8	B 06 FF 50 50 83	MàQÿuü<ø<.ÿPPf
00000400 C4	1C 8B (C7 EB O2 33	CO 5F 5E C	9 C3 <mark>8B</mark> 44 24 08	Ă.«Çĕ.3À_^ÉĂKD\$.
00000410 8B	08 68 4	44 D7 E6 F1	50 FF 51 1	8 59 59 C3 8B 54	<.hD׿ÿPÿQ.YYÃ <t< th=""></t<>
00000420 24	OC 8B	44 24 04 81	08 56 8B 7	4 24 14 56 83 C2	\$. <d\$.<.v<t\$.vfå< th=""></d\$.<.v<t\$.vfå<>
00000430 18	52 6A (04 50 FF 5:	34 83 C4 1	0 85 CO 78 11 57	.Rj.PÿQ4fÅÅx.₩
00000440 8B	3 E 8 B 7	74 24 10 83	C7 08 A5 A	5 A5 A5 33 CO 5F	<>< t\$.fÇ.¥¥¥¥3Å
Offset: 40C	Block: 40	C-40C	Le	ength: 1	Overwrite

- In this simple example, the "haxed" version of the PE file has opcode 0xC3 at offset 0x40C while the original file has 0x8B
- Those who are familiar with the x86 instruction set may recognize the 0xC3 opcode as the RET (return) instruction
- Note that at the bottom of the HxD window it shows the file offset of the highlighted diff byte ("Block 40C-40C")
- This corresponds to the information outputted by our bios_diff.py

File Edit Search View Analysis Extra	as Window ?	
📄 🚵 🕶 📊 🧼 😃 🖬 16	🖌 ANSI 🖌 hex 🖌	
📓 PE32_94_haxed 🔝 PE32_94		
C: Wocuments and Settings Ad	ministratorWesktopWE3Z_94_haxed	
Offset(h) 00 01 02 03 04	05 06 07 08 09 0Å 0B OC OD OE OF	<u>^</u>
000003D0 45 E4 8B 45 F8	8 8B 40 02 89 45 E8 8D 45 FC 50 6A	Eä <eø<0.‰eè.eüpj< td=""></eø<0.‰eè.eüpj<>
000003E0 OC 8D 45 BO 50) 56 E8 33 00 00 00 83 45 FC 18 6A	E°PVè3fEü.j
000003F0 OC 8D 4D E0 5:	FF 75 FC 8B F8 8B 06 FF 50 50 83	MàQÿuü<ø<.ÿPPf
00000400 C4 1C 8B C7 EH	3 02 33 CO 5F 5E C9 C3 <mark>C3</mark> 44 24 08	Ă.<Çë.3À ^ÉÃĂD\$.
00000410 8B 08 68 44 D	7 E6 FF 50 FF 51 18 59 59 C3 8B 54	< .hD׿ÿPÿQ.YYÃ< T
00000420 24 OC 82 14 0	01 00 00 55 00 54 94 14 56 83 C2	\$. <d\$.<.v<t\$.vfå< th=""></d\$.<.v<t\$.vfå<>
00000430 18 52 6 Informa	tion 🛛 🔀 5 CO 78 11 57	.Rj.PÿQ4fÄàx.W
00000440 8B 3E 8	5 A5 33 CO 5F	<> <t\$.fç.¥¥¥¥3à< td=""></t\$.fç.¥¥¥¥3à<>
🗄 C:Vocuments and Se 🕠	No further differences found.	×
Offset(h) 00 01 0	B OC OD OE OF	~
000003D0 45 E4 8	OK D 45 FC 50 6A	Eác Eac 0.% Fé, EüPi
000003E0 OC 8D 45 80 50	56 68 33 00 00 00 83 45 FC 18 6A	E°PVè3fEü.j
000003F0 OC 8D 4D E0 51	FF 75 FC 88 F8 88 06 FF 50 50 83	MàOÿuücøc.ÿPPf
00000400 C4 1C 8B C7 EH	3 02 33 CO 5F 5E C9 C3 8B 44 24 08	Ä. (Cë.3À ^ÉĂ D\$.
00000410 8B 08 68 44 D	7 E6 FF 50 FF 51 18 59 59 C3 8B 54	<.hD׿VPVO.YYÃ <t< td=""></t<>
00000420 24 OC 8B 44 24	04 8B 08 56 8B 74 24 14 56 83 C2	\$.< D\$.<. V< t\$.VfÅ
00000430 18 52 6A 04 50) FF 51 34 83 C4 10 85 C0 78 11 57	.Rj.P∜O4fÄàx.W
00000440 8B 3E 8B 74 24	1 10 83 C7 08 A5 A5 A5 A5 33 C0 5F	<> <t\$.fç.¥¥¥¥3à< td=""></t\$.fç.¥¥¥¥3à<>
Offset: 40C Block: 40C-40C	Length: 1	Overwrite

- You can cycle through each byte that is different by pressing 'F6' (Next Difference)
- In this simple example, there is only this single byte that is different

Analyzing UEFI Files with IDA

🖹 IDA - C:\Document	ts and Setting	s\Administrator\Desktop\PE32_94
File Edit Jump Sea	rch View De	ebugger Options Windows Help
: 📂 🔒 : 🗢 - 🔿 -		a 🚳 🖡 🌜 🐼 i 🕰 🥥 i 📾 📾 🕼 📌 * 🛠 🖆 🗙 i 🕨 🔟 🗖 No debugger 🛛 😒 😢 i 🗊 🚏 » i 👉 » i 🚓 » i 🕳
Library function	Data 📕 Regula	r function 📕 Unexplored 📕 Instruction 🗾 External symbol
F Functions window		[IDA View-A 🛛 🚺 Hex View-A 🗶 🖪 Structures 🗶 📜 Enums 💌 🛐 Imports 🗵 📝 Exports 🗵
Function name	^	
f sub_FFE6CEE4 f start f sub_FFE6D0A2	=	
f sub_FFE6D0D6		
f sub_FFE6D170		public start
f sub_FFE6D281		start proc near
f sub_FFE6D30C	~	arg 4= dword ptr 8
	>	mouth oper forstand ki
Crach quantian		mov ecx, [esp+ary_4]
All diapitoverview		push offset unk_FFE6D744 push eax call dword ptr [ecx+18h] pop ecx pop ecx
		retn
		100.00% (0,0) (242,243) 0000040C FFE6D090: start
Output window		
IDAPython v1.5.5	final (se	rial 0) (c) The IDAPython Team <idapython@googlegroups.com></idapython@googlegroups.com>

- Now we'll actually take a look at these files in IDA
 - Free version is mostly adequate, minus the Hex-Rays pseudo-code view
- Notice IDA recognizes the PE file format and opens the file accordingly
 - IDA 6.7 will recognize UEFI files! (but can't distinguish between PEI and DXE drivers, and so just applies a DXE entry point definition in both cases)
- Shown here is the non-hacked version of the TPM driver showing real instructions at the entry point

Analyzing UEFI Files with IDA

ি IDA - C:\Documents	s and Setting	s\Administrator\Desktop\PE32_94_haxed	_ & ×
File Edit Jump Sear	ch View De	bugger Options Windows Help	
📑 🔚 🗄 🗢 🗝 🗝		1 🚳 📭 🐼 i 🗖 🥥 i 📾 🏙 🗗 🖈 * 📽 🗙 i 🕨 🔲 🗖 No debugger 🛛 💌 💽 💽 🚼 🏞 » i 🔊 🗤	» i — »
			~
Library function)ata 📃 Regula	function 🗾 Unexplored 📕 Instruction 🙁 External symbol	
Functions window		[IDA View-A 🔀 🚺 Hex View-A 🔀 🖪 Structures 🗵 🔃 Enums 🗵 🛐 Imports 🗵 📝 Exports 🗵	
Function name	^		
📝 sub_FFE6CEE4			
f start			
f sub_FFE6D0A2			
f sub_FFE6D0D6			
f sub_FFE6D170			
f sub FFE6D281			
f sub_FFE6D30C	~	public start	
<	>	start proc near	
		retn	
Graph overview		start endp	
		100.00% (0,0) (179,288) 0000040C FFE6D090: start	
Output window			
IDAPython v1.5.5	final (se	rial 0) (c) The IDAPython Team <idapython@googlegroups.com></idapython@googlegroups.com>	^
	`		
llcing ELIDT cign	sturo. CEU	Ear 107/11	

- Shown above is the hacked file with just the RET at the entry point
- This simple example assumes the attacker has placed this instruction here so that the TPM driver never performs any of its activities

Analyzing UEFI Files with IDA



- To see the pseudo-code you will need the full version of IDA Pro with Hex-Rays
- The non-hacked file is dereferencing a DWORD at offset 24 of arg 2
 - IDA displays offsets in base 10 by default; 24 is 0x18
- The dereference is followed by a call: (a2, &unk_FFE6D744)
- So this appears to be calling a function pointer from out of a table

Applying UEFI Structure Definitions

8	DA - C	Docur	nents ar	nd Sett	tings\Admi	nistrator	\Desktop\	PE32_	_94					
File	Edit	Jump	Search	View	Debugger	Options	Windows	Help						
	New ins	tance										No debugger	•	-
2	Open													
	Load file Produce	; file								Ì	š	Reload the input file		
R	Script fil	e								Alt+F7		IDS/IDT file		h
S	Script co	ommand.								Shift+F2		PDB file		- JH
-	Save									Ctrl+W		DBG file		
	Save as											TDS file		
•	Take da	tabase si	napshot							Ctrl+Shift+W	1	FLIRT signature file		
	Close											Parse C header file	Ctrl+F9	
	Quick st	art									Р			_

- UEFI uses publically-defined data structures
- We're going to import 'behemoth.h' which was created by Snare (using scripts)
 - <u>https://github.com/snarez/ida-efiutils/blob/master/behemoth.h</u>
 - Snare has done a talk on attacking Apple's EFI implementation
 - Black Hat USA 2012: <u>http://ho.ax/downloads/De_Mysteriis_Dom_Jobsivs_Black_Hat_Slides.pdf</u>
 - White Paper: <u>http://ho.ax/De_Mysteriis_Dom_Jobsivs_Black_Hat_Paper.pdf</u>

Applying UEFI Structure Definitions

🗽 IDA - C:\Doc	uments and Set	tings \A dministrator	Wesktop\PE32_	94		
Select C heade	er file to parse				? 🛛	
Look in:	🕑 Desktop		v G d	b 📂 🎟-		No debugger
My Recent Documents Desktop My Documents	My Documents My Computer My Network Pl cleanup	s laces		<i>V</i>		A Structures E En art c near ord ptr 8 ax, [esp+arg_4] cx, [eax] Ffset unk_FFE6D744 ax word ptr [ecx+18h]
My Computer	File name:	behemoth h		~	Open	*
My Network	Files of type:	*.h		~	Cancel	

- Our behemoth.h file is located in the C:\Tools\ directory
- It contains a lot of structure definitions from the EFI Specification
 - Plus enumerated values and types

Applying UEFI Structure Definitions

🗽 IDA - C:\Documents and Settir	ngs\Administrator\Desktop\PE32_94
File Edit Jump Search View I	Debugger Options Windows Help
: 📂 🔚 : 🗢 - 🔿 - : 🛱 🛱 I	🏝 🗳 🗼 🏡 🗄 🕰 😪 🗄 📾 📾 🗗 🖈 🥓 📾 🗙 🗄 🕨 🖬 🖸 No debugger 🔤
· · · ·	
Library function Data Regu	ular function 🗾 Unexplored 📕 Instruction 🚺 External symbol
Functions window	🕻 [IDA View-A 🔀 🛛 [Pseudocode-A 🖂 🛛 🖸 Hex View-A 🐹 🚺 Structures 🔀 🛛 🧮 Enums 🗵
Function name Image: sub_FFE6CEE4 Image: start Image: sub_FFE6D0A2 Image: sub_FFE6D0A2 Image: sub_FFE6D1A7 Image: sub_FFE6D1A7 Image: sub_FFE6D1A7 Image: sub_FFE6D1A7 Image: sub_FFE6D1A7 Image: sub_FFE6D30C Image: sub_FFE6D30C <	<pre>public start start proc near arg_4= dword ptr 8 mov eax, [esp+arg_4] mov ecx, [eax] Warning There were 7 error(s). Please look at the message window to see the details CK 100.00% (-270,37) (127,9) 00000418 FFE6D09C: start+C</pre>
= Output window	

- Ignore any errors you see when importing this file
 - Importing the structures we use will still work
| IDA - C:\Documents and Setting | s\Administrator\Desktop\PE32_9 | 94 | |
|--------------------------------|--|---------------------------------|--------------------|
| File Edit Jump Search View De | bugger Options Windows Help | | |
| 📂 🔒 🗄 🗢 🖙 🕘 👬 🏝 🎂 | n 🏝 🦊 🏑 ! 🔼 🔵 ! 📾 🖬 | 🗄 🗗 📌 🛸 🗙 🗄 🕨 | No debugger |
| : • | | | |
| Library function Data Regula | r function 📕 Unexplored 📕 Instructio | n 📃 External symbol | |
| Functions window | [🔄 IDA View-A 🖂 🛛 [🔄 Pseudoco | de-A 🔀 📔 🖸 Hex View-A 🔣 🔇 | 🖪 Structures 🔀 🛛 🚺 |
| Function name | 00000000 ; Ins/Del : crea | ate/delete structure | |
| f sub EFE6CEE4 | 00000000 ; D/A/* : crea | ate structure member (d | lata/ascii/array) |
| f start | 00000000; N : ren | ame structure or struc | ure member |
| f sub FFE6D0A2 | 00000000; U : 0010 | ete structure member | |
| f sub FFE6D0D6 | 66666666 | | |
| f sub FFE6D170 | 00000000 ; (Class Inform | er) | |
| f sub FFE6D1A7 | 00000000 type_info | struc ; (sizeof=0x8, u | variable size) |
| f sub FFE6D281 | 00000000 vftable | dd ? | ; offset (0000 |
| f sub_FFE6D30C | 00000004 _m_data | dd ? | |
| f sub_FFE6D354 | UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU | db U dup(?) | ; string(C) |
| f sub_FFE6D419 | aaaaaaas | enus | |
| f sub_FFE6D464 | 00000000 : | | |
| f sub_FFE6D494 | 0000000 | | |
| f sub_FFE6D619 | 00000000 ; (Class Inform | er) | |
| | 00000000 PMD | <pre>struc ; (sizeof=0xC)</pre> | ; XREF: RTTIBa: |
| | 00000000 mdisp | dd ? | |
| | 000000004 pdisp | 00 ?
dd 2 | |
| < | aggaggg Antzh | uu : | |
| Line 1 of 13 | 1. type info:0000 | | |
| Output window | | | |

- Now go to the Structures tab
- Hit 'Insert'

Y IDA - C: Wocuments and Setting	gsVAdministratorVDesktopVPE32_94	
File Edit Jump Search View D	ebugger Options Windows Help	
: 📂 🔒 : 🗢 - 🖝 - : 🛍 🛍 🦉	🗄 🚳 🔍 🐜 🗄 🗛 🖓 🚵 📾 📾 💣 🖈 🐨 🖨 📈 🗄 🕨 🔲 🗖 No debugger	
Library function 📃 Data 📃 Regula	ar function 🔤 Unexplored 📕 Instruction 🔛 External symbol	
Functions window	[IDA View-A 🛛 [Pseudocode-A 🛛 🚺 Hex View-A 🛛 🖪 Structures 🗙 🔛 B	Enur
Function name	00000000 ; Ins/Del : create/delete structure	
f sub_FFE6CEE4	100000000 ; D/A/* : create structure member (data/ascii/array)	
f start	AAAAAAAA : II	
f sub_FFE6D0A2	666666666 ; 🦹 Create structure/union 🛛 🔀	
f sub_FFE6D0D6	0000000	
f sub_FFE6D170	00000000 ; (C1 Structure name struc_1	
f sub_FFE6D1A7	00000000 type_	19.9
f sub_FFE6D281		เยย
f sub_FFE6D30C	00000008 m d	
f sub_FFE6D354	00000008 type Don't include in the list	
f sub_FFE6D419	0000008 Create union	
f sub_FFE6D464	0000000 ;	
f sub_FFE6D494	0000000	
f sub_FFE6D619	Add standard strutture	
		CI
	GGGGGGGBB Ddisp OK Cancel Help	
	00000008 vdisp	
Line 1 of 12	1 type info:0000	
Life For 13	r. cype into.coco	
 Output window 		

• Select 'Add a Standard Structure'

Please choose a structure

Type name 🔺	Peclaration
To EFI_MTFTP6_TOKEN	struct _EFI_MTFTP6_TOKEN
to EFI_NARROW_GLYPH	struct \$4C25BD3AB5FE1A20
to EFI_NETWORK_INTERFACE_IDENTIFIER_INTERFACE	EFI_NETWORK_INTERFACE_
to EFI_NETWORK_INTERFACEENTIFIER_PROTOCOL	struct _EFI_NETWORK_INTE
EFI_NETWORK_STATISTICS	struct \$775CA34CBECCC7D2
5 EFI_OPEN_PROTOCOL_INFORMATION_ENTRY	struct \$69792873A83EE8C5
to EFI_PARTITION_ENTRY	struct \$528E0275AD911FE53
to EFI_PARTITION_TABLE_HEADER	struct \$922F44E67A0121CC
to EFI_PCD_PROTOCOL	struct _EFI_PCD_PROTOCOL
€ FI_PCI_HOST_BRIDGE_RESOURCE_ALLOCATION	struct _EFI_PCI_HOST_BRID
€ FI_PCI_HOTPLUG_REQUEST_PROTOCOL	struct _EFI_PCI_HOTPLUG_F
€ FI_PCI_HOT_PLUG_INIT_PROTOCOL	struct _EFI_PCI_HOT_PLUG
€ FI_PCI_IO_PROTOCOL	struct _EFI_PCI_IO_PROTO
€ FI_PCI_IO_PROTOCOL_ACCESS	struct \$E346F8498CA5CDD2
€ FI_PCI_IO_PROTOCOL_CONFIG_ACCESS	struct \$A52CA55E61E81B9D
to EFI_PCI_OVERRIDE_PROTOCOL	EFI_PCI_PLATFORM_PROTO
5 EFI_PCI_PLATFORM_PROTOCOL	struct _EFI_PCI_PLATFORM
€ FI_PCI_ROOT_BRIDGE_IO_PROTOCOL	struct _EFI_PCI_ROOT_BRID
EFI_PCI_ROOT_BRIDGE_IO_PROTOCOL_ACCESS	struct \$1A1FB1F5818659240
€ FI_PCI_ROOT_BRIDGE_IO_PROTOCOL_PCI_ADDR	struct \$D7773540B8449B72E
to EFI_PE32_SECTION	EFI_COMMON_SECTION_HE
DEFI_PE32_SECTION2	EFI_COMMON_SECTION_HE
DEFI_PEI_DEPEX_SECTION	EFI_COMMON_SECTION_HE
to EFI_PEI_DEPEX_SECTION2	EFI_COMMON_SECTION_HE
to EFI_PEI_FIRMWARE_VOLUME_PPI	struct _EFI_PEI_FIRMWARE
EFI_PEI_HOB_POINTERS	union \$B30AF9753A4819180
EFI_PEI_NOTIFY_DESCRIPTOR	struct \$98608686980957845
EFI_PEI_PPI_DESCRIPTOR	struct \$B511F1D301007EC4
TO EFI PEI SECURITY2 PRI	etruct EEL PEL SECURITY2
EFI_PEI_SERVICES	struct _EFI_PEI_SERVICES
to EFI_PIC_SECTION	EFI_COMMON_SECTION_HE
	ОК

- We can sort the structures by name to make search easier
- We're looking for EFI_PEI_SERVICES
- These are services used by PEIMs during the PEI phase
- An (incomplete) ٠ sampling is below:

/ICES

EFI_PEI_SERVICE	S struc ; (sizeof=0x78)
Hdr	EFI_TABLE_HEADER ?
InstallPpi	dd ?
ReInstallPp >	dd ?
LocatePpi	dd ?
NotifyPpi	dd ?
GetBootMode	dd ?
SetBootMode	dd ?
GetHobList	dd ?
CreateHob	dd ?
FfsFindNextVolu	medd?
FfsFindNextFile	dd ?
FfsFindSectionD	ata dd ?
InstallPeiMemor	ydd?

🗽 IDA - C: Wocuments and Setting	gs\Administrator\Desktop\PE32_94	
File Edit Jump Search View D	ebugger Options Windows Help	
i 🥌 🔒 i 🧇 🔹 🛶 🗉 🟥 👫 🦓	🗄 🗳 🍹 🛵 i 🗖 😪 i 📾 📾 🗗 🖈 🧭 🖬 🗙 i 🕨 🔲 🗖 No debugger	
1		
Library function Data Regula	ar function 🔛 Unexplored 📕 Instruction 🔛 External symbol	
Functions window	[IDA View-A 🛛 [Pseudocode-A 🔀 🚺 Hex View-A 🛛 🖪 Structures 🛛 🔃 E	nur
Function name	00000000 ; Ins/Del : create/delete structure	_
f sub FFE6CEE4	00000000 ; D/A/* : create structure member (data/ascii/array)	
f start	000000000; N : rename structure or structure member	
f sub_FFE6D0A2	00000000 ; 🦹 Create structure/union 🛛 🔀	
f sub_FFE6D0D6	0000000	
f sub_FFE6D170	00000000 ; (C1 Structure name struc_1	
f sub_FFE6D1A7		
f sub_FFE6D281	00000004 m da Create before current structure	ยย
f sub_FFE6D30C		
J SUD_FFE6D354	00000008 type_ Don't include in the list	
f sub_EEEED464	00000008 Create union	
f sub_FFE6D494	66666666	
f sub FFE6D619		
	88888888 PMD Add standard stru, gire	C1
	0000000 mdisp OK Cancel Help	
< >	aaaaaaa aa i	
Line 1 of 13	1. type info:0000	
Output window		

• Now we're going to add an EFI_GUID structure

Please choose a structure

Type name	Declaration
EFI_FIRMWARE_VOLUME_EXT_ENTRY_OEM_TYPE	struct \$D117E6D6ADF5AE6E9
to EFI_FIRMWARE_VOLUME_EXT_HEADER	struct \$9A94C1850C5771676
to EFI_FIRMWARE_VOLUME_HEADER	struct \$331CF4F13D46CA83F
EFI_FIRMWARE_VOLUME_IMAGE_SECTION	EFI_COMMON_SECTION_HEA
EFI_FIRMWARE_VOLUME_IMAGE_SECTION2	EFI_COMMON_SECTION_HEA
EFI_FONT_DISPLAY_INFO	struct _EFI_FONT_DISPLAY_1
DEFI_FONT_INFO	struct \$CA721A3BB17EE8BE9
EFI_FORM_BROWSER2_PROTOCOL	struct _EFI_FORM_BROWSEF
EFI_FREEFORM_SUBTYPE_GUID_SECTION	struct \$A092FDE5FBE135C43
EFI_FREEFORM_SUBTYPE_GUID_SECTION2	struct \$7B33F66C6091FD0FD
EFI_FTP4_COMMAND_TOKEN	struct _EFI_FTP4_COMMAND
DEFI_FTP4_CONFIG_DATA	struct \$48E1A16206B652E9F
EFI_FTP4_CONNECTION_TOKEN	struct \$F95AF53F7A4A08C24
EFI_FTP4_PROTOCOL	struct _EFI_FTP4_PROTOCOL
EFI_FV_BLOCK_MAP_ENTRY	struct \$1AFA0B1BEAF8357FC
EFI_FV_FILE_INFO	struct \$88A6A407AB7AF507E
to EFI_FV_INFO	struct \$531CE289FE509C6B7
to EFI_FV_WRITE_FILE_DATA	struct \$9CAD97D4ED8FDCD8
DEFI_FX_SAVE_STATE_IA32	struct \$0F200C5991ECE5F6E
EFI_FX_SAVE_STATE_X64	struct \$383942B824C6F4B80
EFI_GCD_IO_SPACE_DESCRIPTOR	struct \$D846051A28653AE82
EFI_GCD_MEMORY_SPACE_DESCRIPTOR	struct \$08D8893954E5EBB7A
EFI_GLYPH_GIBT_END_BLOCK	struct _EFI_GLYPH_GIBT_END
EFI_GPT_DATA	struct tdEFI_GPT_DATA
EFI_GRAPHICS_OUTPUT_BLT_PIXEL	struct \$A63D8E7FAAB623CF8
EFI_GRAPHICS_OUTPUT_BLT_PIXEL_UNION	union \$87345348CB55A1BB2;
EFI_GRAPHICS_OUTPUT_MODE_INFORMATION	struct \$A4A71508AC7A43D1
EFI_GRAPHICS_OUTPUT_PROTOCOL	struct _EFI_GRAPHICS_OUTF
TO EFI_GRAPHICS OUTPUT_PROTOCOL_MODE	struct \$333E6E9D3946E19E3
EFI_GUID	GUID
The EFI_GUIDED_SECTION_EXTRACTION_PROTOCOL	STRUCT_EFI_GUIDED_SECTIO

- A GUID is a 16-byte data structure used as a name for many of the EFI objects:
 - Dword
 - Word
 - Word
 - Char array[8]

EFI_GUID	<pre>struc ; (sizeof=0x10)</pre>
Data1	dd ?
Data2	dw ?
Data3	dw ?
Data4	db 8 dup(?)
EFI_GUID	ends

EFI_GUID

IDA - C:\Documents and	Setting	gs\Administrator\Desktop\PE32_94	×
File Edit Jump Search Vi	iew De	ebugger Options Windows Help	
i 📂 🔒 i 🗢 - 🖮 i 🏝		ha 🐴 🗼 🐜 🗄 🕰 🥥 🗄 📾 💣 📌 * 🧩 🛋 🗙 🗄 🕨 🔲 🗖 No debugger 🔛 💌 🛃 🚼 🚏 » 🗄 🛣 » 🗄 📥 » 🗄	>>
1 I I I I I I I I I I I I I I I I I I I			~
Library function 📃 Data	Regula	ar function 🔝 Unexplored 🔜 Instruction 🔝 External symbol	
f Functions window	₽ ×	[IDA View-A 🛛 🚺 Hex View-A 🖾 🖪 Structures 🖾 🔃 Enums 🖾 🛐 Imports 🖾 📝 Exports 🖾	
Function name	^		Π
<pre>\$ sub_FFE6CEE4 \$ start \$ sub_FFE6D0A2 \$ sub_FFE6D0D6 \$ sub_FFE6D170 \$ sub_FFE6D1A7 \$ sub_FFE6D281 \$ sub_FFE6D30C \$ sub_FFE6D30C</pre>	~	Image: wide wide wide wide wide wide wide wide	
Graph overview	đ×	100.00% (0,0) (242,243) 0000040C FFE6D90: start	
Output window			×

- Likely this file will be using the PEI Services table:
- The name of the file is 'AmiTcgPlatformPeiBeforeMem'
- It's a common structure used during the PEI phase so PEIMs can use common services

🚺 IDA - C:Wocuments and Settin	s\Administrator\Desktop\PE32_94	
File Edit Jump Search View D	ebugger Options Windows Help	
	a 🐴 🔍 🖾 🖬 💁 🤮 🖬 📾 🗶 👯	No debugger
1		
Library function 📗 Data 📕 Regul	r function 📕 Unexplored 📕 Instruction 📃 External symbol	
Functions window	📳 IDA View-A 🔀 🛛 📳 Pseudocode-A 🔀 🚺 Hex View-A 🛛	🛛 🛛 A Structures 🔀 🛛 🧮 Enums 🛛 🖉
Function name		
f sub_FFE6CEE4		
f start	A Choose a structure for offset	
f sub_FFE6D0A2	Operand representation	Structure cize
f sub_FFE6D0D6		Structure size
f sub_FFE6D170		0010
f sub_FFE6D1A7	EFI_PEI_SERVXLES.InstallPpi	00/8
f sub_FFE6D281	A size R ITIBaseClassDescriptor	0018
f sub_FFE6D30C	A RTTClassHierarchyDescriptor.numBaseClasses+10h	0010
< >	A RTTICompleteObjectLocator.offset+14h	0014
Line 1 of 13	A RTCI.m_pClassInit	001⊂
	A size EFI_TABLE_HEADER	0018
Graph overview	A type_infom_d_name+10h	0008
	A PMD.mdisp+18h	000C
	OK Cancel S	earch Help
	Line 2 of 9	
	100.00% (-270,37) (440,173) 00000418 FFE6D09C: sta	art+C

- Hit 't' to have IDA interpret that value as a structure
- Select EFI_PEI_SERVICES based on our hypothesis

\$* .s [‡] •	* 🛋 🗙 ! 🕨 🔲 🗖 No debugger 💌 🍖 🛃 ! 🗊 🚏 » ! :
External	symbol
	D Hex View-A 🖾 🖪 Structures 🖾 🖽 Enums 🖾 🕅 Imports 🖾 💷 E
	<pre>public start start proc near arg_4= dword ptr 8 mov eax, [esp+arg_4] mov ecx, [eax] push offset stru_FFE6D744 ; EFI_PEI_PPI_DESCRIPTOR * push eax</pre>

- Hit Ok or 'y' to accept this definition
- IDA does not have an undo, so it's always good to save first
 - But we have a hunch that this is the right object

5 × IDA View-A ▼ 1 intcdecl s 2 { 3 return (*(i 4 }	🖹 Pseudocode-A 🔀 tart(int a1, i .nt (cdecl **	O Hex View-A ⊠ nt <mark>a2</mark>))(int, _UNKNOWN	A Structures X	<pre>Enums ▼</pre> *)a2 + 24))(Timports X	Expor
Please enter a string			G.		×	
Please enter type declaration	EFI_PEI_SERVICES ** EFI_PEI_SERVICES **		1		~	

- In the pseudo-code view you can do the same thing
- Select the a2 argument and hit 't'
- Select the EFI_PEI_SERVICES structure
- When we enter the above, we see the code simplifies:



- We see that this function immediately calls the InstallPpi() PEI Service
- InstallPpi() takes 2 arguments:
 - The EFI_PEI_SERVICES structure
 - Some Unknown argument
- Per the EFI Specification, InstallPpi installs an interface in the PEI PEIM-to-PEIM Interface (PPI) database by GUID
- We could look up the prototype in the spec:

```
typedef
EFI_STATUS
(EFIAPI *EFI_PEI_INSTALL_PPI) (
   IN CONST EFI_PEI_SERVICES **PeiServices,
   IN CONST EFI_PEI_PPI_DESCRIPTOR *PpiList
  );
```



Always let the GUIDs be your GUIDe

- UEFI uses a lot of "GUIDs" Globally Unique IDentifiers.
- Used to identify files on the filesystem
 - Filesystem GUIDs often reused between EDK & production systems. Or between the same IBV code on different OEMs' systems
- Used to identify structures (PPIs in PEI phase, Protocols in DXE phase) that contain data and/ or function pointers





- But in this case IDA also recognizes this structure
- We can double-click on it to see that IDA has identified it as an EFI_PEI_PPI_DESCRIPTOR :
 - First is the Flags 80000010h
 - Second is the pointer to the GUID
 - Third is the pointer to the PPI that will be installed



- Select the GUID structure
- One thing we can do is try and determine if this is a known-GUID or an unknown GUID
 - The UDK defines a lot of GUIDS, these would likely be the same across all vendors
 - Vendors also implement their own proprietary GUIDS

1		
2	efiguids.	. рү
3		
4	This is a	a giant list of protocol GUIDs I grepped out of the TianoCore source code.
5	but won't	t contain any of Apple's proprietary GUIDs. I'll add those as I come acros
6	r	
7	See the	Find
8	https://	Find Replace Find in Files Mark
9		
10	227 miss	Find what : c1e6791d Find Next
11		
12		Coun <u>r</u>
13	GUIDs =	Find All in All Opened
14	'ACPI_TA	Documents E,
15	'APPLE_R	Find All in Current
16	'ARM_GLO	Match whole word only Document
17	'ARM_HOB	Match case
18	'ARM_MP_	Wrap around Xd8
19	'ARM_MP_	, Oracle Made
20	'BDS_LIB	Direction ✓ Iransparency 5,
21	BLOCKIO	<u>N</u> ormal <u>Up</u> On losing focus , 0:
22	BLOCK M	© Extended (\n, \r, \t, \0, \x) © Down © Always 0x
23	BOOT_MA	Regular expression
24	'BOOT_MA	
25	CONNECT	Find: Can't find the text "c1e6791d"
26	'DEVICE_L	
27	'DP_HII_0	GUID':[0xeb832fd9, 0x9089, 0x4898, 0x83, 0xc9, 0x41, 0x61, 0x8f, 0x5c, 0x4

- Snare also provides the efiguids.py file which contains GUIDs he pulled out of the UDK
- Our efiguids.py is located in C:\Tools\ and contains previously identified GUIDs
- In this case it is not in this file. We can name it 'UnknownGuid1'



- Now if we follow the pointer it will take us to the PPI that is going to be installed
- This function is what will get called when someone uses this PPI

• IDX * C. Documents and Settingsordiministrator Desktop #152_24			
File Edit Jump Search View D	ebugger Options Windows Help		
📂 🖬 🗄 🗢 🔹 🔶 🗄 🏪 🦉	s 🐴 🔍 🖾 🔍 🗄 📾 📾 🗗 🖈 📽 🖬 🗶 🗼 🗖 🗖		
1			
Library function 📃 Data 📃 Regula	ar function 📕 Unexplored 📕 Instruction 🔛 External symbol		
Functions window	[IDA View-A 🛛 [Pseudocode-A 🛛 🚺 Hex View-A 🛛 🖪 Str		
Function name	1 intcdecl sub_FFE6CEE4(int a1)		
f sub_FFE6CEE4	2 { 3 int u1: // esi@1		
f start	4 int v2: // eax@1		
f sub_FFE6D0A2	5 int result; // eax@2		
f sub_FFE6D0D6	6 int v4; // eax@13		
f sub_FFE6D170	7 int v5; // edi@13		
f sub_FFE6D1A7	8 int v6; // [sp+8h] [bp-60h]@10		
f sub_FFE6D281	9 INT V/; // [Sp+18N] [Dp-50N]@1		
f sub_FFE6D30C	11 int16 u9: // [sp+16h] [bp-46h]@1		
f sub_FFE6D354	12 char v10: // [sp+20h] [bp-48h]@1		
f sub_FFE6D419	13 char v11; // [sp+21h] [bp-47h]@1		
f sub_FFE6D464	14 char v12; // [sp+22h] [bp-46h]@1		
f sub_FFE6D494	15 char v13; // [sp+23h] [bp-45h]@1		
f sub_FFE6D619	16 char v14; // [sp+24h] [bp-44h]@1		
	17 Char V15; // [sp+25h] [bp-43h]@1		
	19 char u17: // [sp+27h] [bp-42h]@1		
	20 int v18; // [sp+28h] [bp-40h]@1		
	21		
Output window			

- We can analyze this is pseudo-code or the main view
- Since it accepts one argument we can hypothesize again that it takes in an instance of the EFI_PEI_SERVICES structure

File Edit Jump Search View Debugger Options Windows Help
🗄 🔚 🗄 🗢 - 👄 - 🏝 🛍 🚳 🚳 🍬 🗼 i 🖾 🕥 📾 📾 🗗 👉 - 🖈 📾 🗙 Þ 💷 🗖 No debugger 🛛 💌 🐿 💽 💽
Library function 📃 Data 🧧 Regular function 🔛 Unexplored 🔛 Instruction 🚺 External symbol
📝 Functions window 🛛 🗗 🗙 📑 IDA View-A 🖾 📑 Pseudocode-A 🐹 💽 Hex View-A 🖾 🖪 Structures 🖾 🔃 Enums 🖾 🛅 Imports 🗵
Function name 37 char v35; // [sp+43h] [bp-25h]@1 f sub_FFE6CEE4 38 char v36; // [sp+44h] [bp-24h]@1 f start 39 char v37; // [sp+45h] [bp-23h]@1 f sub_FFE6D0A2 40 char v38; // [sp+46h] [bp-22h]@1 f sub_FFE6D0D6 41 char v39; // [sp+47h] [bp-21h]@1 f sub_FFE6D170 43 int 16 v41; // [sp+48h] [bp-26h]@13 f sub_FFE6D1A7 int v40; // [sp+50h] [bp-16h]@13 f sub_FFE6D1A7 44 int v42; // [sp+50h] [bp-18h]@13 f sub_FFE6D361 Please enter a string X f sub_FFE6D354 Please enter type declaration EFI_PEI_SERVICES **all EFI_PEI_SERVICES **all Y f sub_FFE6D464 EFI_PEI_SERVICES **all Image: Services **all Image: Services **all
f sub_FFE6D619 52 u43 = 0; u19 = 10265; u19 = 10265; u20 = 102529; u20 = 102529; u8 = 5694; 0 u1 = a1; u8 = 5694;

• As before, we can define this as EFI_PEI_SERVICES**a1

TDA - C:\Documents and Settings\Administrator\Desktop\PE32_94					
File Edit Jump Search View	Debugger Options Windows Help				
: 📂 🔚 : 🗢 - 🔿 - 🤅 🛍 🛍 (🛅 🚔 🗼 🐜 🗄 🗛 🕴 🗛 🕹 🗰 📾 💣 📌 🐲 📾 🗙 🗄 🕨 🖬 🖬 No debugger 💽 💌 😭 🛃				
1					
Library function 📃 Data 📃 Regu	Library function 🗾 Data 🗖 Regular function 🔛 Unexplored 📰 Instruction 🔛 External symbol				
F Functions window	🕻 📑 IDA View-A 🐹 🔚 Pseudocode-A 🔀 🚺 Hex View-A 🔟 🖪 Structures 🗵 🔢 Enums 🗵 🛐 Imports 🗵				
Function name	37 char v35; // [sp+43h] [bp-25h]@1				
f sub_FFE6CEE4	38 char v36; // [sp+44h] [bp-24h]@1				
f start	39 Char V37; // [Sp+45h] [Dp-23h]@1				
f sub_FFE6D0A2	41 char $v39$; // [sp+47h] [bp-22h]@1				
f sub_FFE6D0D6	42 int v40; // [sp+48h] [bp-20h]@13				
f sub_FFE6D170	43int16 v41; // [sp+4Ch] [bp-1Ch]@13				
f sub_FFE6D1A7	44 int v42; // [sp+50h] [bp-18h]@13				
📝 sub_FFE6D281	Please enter a string				
f sub_FFE6D30C					
f sub_FFE6D354	Please enter type declaration EFI_PEI_SERVICES **v1				
f sub_FFE6D419					
f sub_FFE6D464	OK Cancel				
f sub_FFE6D494					
f sub_FFE6D619	• 52 $v43 = 0;$				
	53 0.19 = 10265;				
	54 = (101)a1;				
	$56 \ u8 = 5694$				

- Also we can define v1 in the same way since its equal to a1
- EFI_PEI_SERVICES**v1



- Now we can scroll down and see that we were right in assuming this was an instance of a EFI_PEI_SERVICES
- We see a call to LocatePpi(), and then GetBootMode(), followed by InstallPpi()
- This series of EFI services "makes sense"



- We can look up the definitions for the new services LocatePpi(), GetBootMode()
- Can we identify the GUID located in the EFI_PEI_PPI_DESCRIPTOR passed into InstallPpi?

'PEI_TPM_INITIALIZED_PPI_GUID':[0xe9db0d58, 0xd48d, 0x47f6, 0x9c, 0x6e, 0x6f, 0x40, 0xe8, 0x6c, 0x7b, 0x41],

Analyzing UEFI Files with IDA

- So from here the strategy would be to use the same methodology to identify and "fill out" LocatePpi(), GetBootMode(), etc.
- For you, cross-correlating where the PPIs are defined that you see getting called later will take a bit of grunt work (grepping for guids, finding their usage, etc)...
- For us, it's already scripted ;)



Further GUID-based analysis strategies

- If you binary grep for a GUID (or search by GUID in UEFITool), you may find that it is specifically referenced/loaded by some other file.
- Pick a GUID in the spec that you're interested in.
 E.g. EFI_DHCP4_PROTOCOL_GUID
- If you grep for it, you'll find everywhere that particular protocol/PPI is used (to include installation, lookup, and things that have registered to be notified when it's available)

Then you just have to sift through the results

TODO:

- Add discussion of diffing things against EDK & against other known stuff
- Here comes a new challenger!
- <u>http://joxeankoret.com/blog/2015/03/13/</u> <u>diaphora-a-program-diffing-plugin-for-ida-</u> <u>pro/</u>

UEFI/Secure Boot Summary

- Secure boot can help you protect your firmware
 - If your BIOS is UEFI but Secure Boot isn't used, you can self-sign keys and turn it on
- But if the SPI flash isn't locked down, secure boot doesn't provide any protection
 - And neither does System Management Mode, or signed firmware updates, or TPM Measured Boot...
- UEFI does add complexity to locking down the SPI flash SPI Protected Range (PR) registers can be used to lock down the UEFI executable firmware
- But the NVRAM variables must remain writeable

A Locked Down UEFI/BIOS Does the Following:

- Has a properly-configured flash descriptor
 - Read-only, provides proper Flash Master permissions
- Protects the UEFI executable code using the PR registers
- Locks down the SPI flash configuration registers (FLOCKDN)
- Uses BIOS_CNTL to protect the flash
- Implements signed firmware updates
- Implements Secure Boot
- Ensures SMM_BWP is asserted so that the flash is writeable only when the processor is in SMM
- Ensures SMRAM is locked down (D_LCK is set and SMRR are used)
- Ensures SMI's are enabled and cannot be suppressed
- If possible uses Measured Boot and observes PCRs
- Sounds simple enough...

csc\fvsc\fv3\43172851-cf7e-4345-9fe0-d7012bb17b88\csc	iFfsSmm
csc\fvsc\fv3\5552575a-7e00-4d61-a3a4-f7547351b49e\csc	SmmBaseRuntime
csc\fvsc\fv3\59287178-59b2-49ca-bc63-532b12ea2c53\csc	PchSmbusSmm
csc\fvsc\fv3\6869c5b3-ac8d-4973-8b37-e354dbf34add\csc	CmosManagerSmm
csc\fvsc\fv3\753630c9-fae5-47a9-bbbf-88d621cd7282\csc	SmmChildDispatcher
csc\fvsc\fv3\77a6009e-116e-464d-8ef8-b35201a022dd\csc	DigitalThermalSensorSmm
csc\fvsc\fv3\7fed72ee-0170-4814-9878-a8fb1864dfaf\csc	SmmRelocDxe
csc\fvsc\fv3\8d3be215-d6f6-4264-bea6-28073fb13aea\csc	SmmThunk
csc\fvsc\fv3\921cd783-3e22-4579-a71f-00d74197fcc8\csc	HeciSmm
csc\fvsc\fv3\9cc55d7d-fbff-431c-bc14-334eaea6052b\csc	SmmDisp
csc\fvsc\fv3\a0bad9f7-ab78-491b-b583-c52b7f84b9e0\csc	SmmControl
csc\fvsc\fv3\abb74f50-fd2d-4072-a321-cafc72977efa\csc	SmmRelocPeim
csc\fvsc\fv3\acaeaa7a-c039-4424-88da-f42212ea0e55\csc	PchPcieSmm
csc\fvsc\fv3\bc3245bd-b982-4f55-9f79-056ad7e987c5\csc	AhciSmm
csc\fvsc\fv4\025b3ec4-28dc-44ae-8c94-d07563be743f\csc	DellFnUsbEmulationSmm
csc\fvsc\fv4\0369cd67-fa74-45a3-bdcb-d25675d5ffde\csc	DellOA30CtrlSmm-Edk1_06-Pi1_0-Uefi2_1
csc\fvsc\fv4\08abe065-c359-4b95-8d59-c1b58eb657b5\csc	IntelLomSmm
$\texttt{csc}fvscfv4\\099fd87f-4b39-43f6-ab47-f801f99209f7\\csc$	DellDcpRegisterSmm-Edk1_06-Pi1_0-Uefi2_1
$\texttt{csc}fvscfv4\\09d2cb46-c303-42c2-9726-5704a1fdfbbd\\csc$	DellVariableSmmWrapper
csc\fvsc\fv4\0d28c529-87d4-4298-8a54-40f22a9fe24a\csc	DellDaHddProtectionSmm-Edk1_06-Pi1_0-Uefi2_1
<pre>csc\fvsc\fv4\0d81fdc5-cb98-4b9f-b93b-70a9c0663abe\csc</pre>	DellDccsSmmDriver
<pre>csc\fvsc\fv4\0dde9636-8321-4edf-9f14-0bfca3b473f5\csc</pre>	DellIntrusionDetectSmm
csc\fvsc\fv4\1137c217-b5bc-4e9a-b328-1e7bcd530520\csc	DellThermalDebugSmmDriver
$\texttt{csc}fvscfv4\1181e16d-af11-4c52-847e-516dd09bd376\csc}$	DellCenturyRolloverSmm
csc\fvsc\fv4\119f3764-a7c2-4329-b25c-e6305e743049\csc	DellSecurityVaultSmm-Edk1_06-Pi1_0-Uefi2_1
csc\fvsc\fv4\12963e55-5826-469e-a934-a3cbb3076ec5\csc	SmmSbAcpi
$\verb csc fvsc fv4 1478454a-4584-4cca-b0d2-120ace129dbb csc $	DellMfgModeSmmDriver
csc\fvsc\fv4\166fd043-ea13-4848-bb3c-6fa295b94627\csc	DellVariableSmm-Edk1_06-Pi0_9-Uefi2_1
csc\fvsc\fv4\16c368fe-f174-4881-92ce-388699d34d95\csc	SmmGpioPolicy
csc\fvsc\fv4\1afe6bd0-c9c5-44d4-b7bd-8f5e7d0f2560\csc	DellDiagsSbControlSmm
csc\fvsc\fv4\26c04cf3-f5fb-4968-8d57-c7fa0a932783\csc	SbServicesSmm
csc\fvsc\fv4\2a502514-1e81-4cda-9b50-8970fa4ac311\csc	R5U242Smm
<pre>csc\fvsc\fv4\2aeda0eb-1392-4232-a4f9-c57a3c2fa2d9\csc</pre>	BindingsSmm

- Oh but vendors also need to ensure that none of the code they implement in SMRAM is buggy
- On the Dell Latitude E6430, ~144 out of 495 EFI modules appear to contribute code to SMM ...

Backup

 Used EFIPWN to backup because we don't recommend its use as a primary tool anymore (but it is still used behind the scenes for Copernicus' bios_diff.py)

EFIPWN

https://github.com/G33KatWork/EFIPWN

Setting up EFIPWN

- This describes using a version of EFIPWN modified by Sam Cornwell who added some improvements:
- EFIPWN requires the following:
- Python (I use 2.7.x-something)
- Mako: <u>http://www.makotemplates.org/</u>
- ArgParse: <u>https://pypi.python.org/pypi/argparse</u>
- Pylzma: <u>http://www.joachim-bauch.de/projects/pylzma/</u>
- I have an easier time downloading the source and installing using "python setup.py install"
- You will also need the 'xz' utility
 - Mac and Linux: you get it either automatically or by easy download
 - Windows: <u>http://tukaani.org/xz/</u>
 - The pre-built binaries work fine. I tested it by putting the bin_x86-x64 version into the local EFIPWN directory and it worked fine

Testing EFIPWN Functionality

C:\Tools\EFIPWN>python dump.py -h usage: dump.py [-h] [-d] file {print,dump,genfdf}			
EFI Firmware exploration tool			
positional arguments: file	The firmware file		
optional arguments: —h, ——help —d, ——debug	show this help message and exit Display debug information (DEBUG)		
Operations: {print,dump,genfdf} print dump genfdf	Print a tree of the structure of the EFI firmware image Dump all files in an EFI firmware image into a directory structure Try to create a EDK2 FDF file for generating a firmware image out of a dump		
C:\Tools\EFIPWN>			

- Once you have all the dependencies installed, typing the following 'python dump.py –h' should yield the above output
- The arguments are a little confusing for EFIPWN, as a general rule they go like this:
- Python dump.y <file> <print, dump> <output>
- * The genfdf function does not work yet

EFIPWN 'print'



 Before we decompose a UEFI binary, we'll use the 'print' functionality to print a text file containing the UEFI firmware volume information and the PE files/modules contained therein

🔚 ReadMe.txt 🗵 🔚 efipwn.txt 🗵		
1	EFI FIRMWARE VOLUME:	
2	Base Offset: 0x00600000	
3	Header Length: 0x48	
4	Data Length: 0x0001ffb8	
5	Total Length: 0x00020000	
6	Signature: _FVH	
7	Attributes: 0xffff8eff	
8		
9		
10	EFI_FIRMWARE_FILE:	
11	Base Offset: 0x0000000	
12	Length: 0x0001ffa0	
13	GUID: 0xcef5b9a3-476d-497f-9fdc-e98143e0422c	
14	Type: RAW (0x01)	
15	Attributes: 0x00	
16	State: 0xf8	
17		
18		
19	EFI_FIRMWARE_VOLUME:	
20	Base Offset: 0x00620000	
21	Header Length: 0x48	
22	Data Length: 0x0001ffb8	
23	Total Length: 0x00020000	
24	Signature: _FVH	
25	Attributes: 0xffff8eff	
26		
27		
28	EFI_FIRMWARE_FILE:	
29	Base Offset: 0x0000000	
30	Length: 0x0001ffa0	
31	GUID: 0xcef5b9a3-476d-497f-9fdc-e98143e0422c	
32	Type: RAW (0x01)	
33	Attributes: 0x00	
34	State: 0xf8	
35		

- The base offset is the Flash Linear Address (FLA) in the file where the volume begins
- This page shows one FV beginning at 60_0000h and another immediately following it at 62_0000h

🔚 Rea	dMe.bt 🗷 🔚 efipwn.bt 🗵		
1	EFI_FIRMWARE_VOLUME:		
2	Base Offset: 0x00600000		
3	Header Length: 0x48		
4	Data Length: 0x0001ffb8	typedef struct (
5	Total Length: 0x00020000	cypeder Scruce (Zanalia at an [16] .
6	Signature: _FVH	UINT8	Zerovector[10];
7	Attributes: 0xffff8eff	EFI_GUID	FileSystemGuid;
8		UINT64	FvLength;
9	PET PIDNUDE FITE.	UINT32	Signature;
11	Eri_rikHWARE_riLE:	EFI FVB ATTRIBUTES 2	Attributes:
12	Length: 0x0001ffa0	UINT16	HeaderLength:
13	GUID: 0xcef5b9a3-476d-497f-9fdc-e98143e04	UINT16	Checksum:
14	Type: RAW (0x01)	UINII0	
15	Attributes: 0x00	UINT16	ExtHeaderOffset;
16	State: 0xf8	UINT8	Reserved[1];
17		UINT8	Revision;
		EFI FV BLOCK MAP	<pre>BlockMap[];</pre>
		} EFI_FIRMWARE_VOLUME_HE	ADER;

- The Header length refers to the length in bytes of the FV header
- The Data length refers to the length in bytes of the FV minus the header
- The Total length refers to the total length of the FV including the header

🔚 Read	dMe.txt 🗵 🔚 efipwn.txt 🗵		
1	EFI_FIRMWARE_VOLUME:		
2	Base Offset: 0x00600000		
3	Header Length: 0x48		
4	Data Length: 0x0001ffb8	typedef struct {	
5	Total Length: 0x00020000	cypeder Struct (Zanalia at an [16] .
6	Signature: FVH	UINT8	Zerovector[16];
7	Attributes: 0xffff8eff	EFI_GUID	FileSystemGuid;
8		UINT64	FvLength;
10	FFT FTDMMADF FTTF.	UINT32	Signature;
11	Base Offset: 0x00000000	EFI FVB ATTRIBUTES 2	Attributes;
12	Length: 0x0001ffa0	UINT16	HeaderLength;
13	GUID: 0xcef5b9a3-476d-497f-9fdc-e98143e04	UINT16	Checksum:
14	Type: RAW (0x01)	IIINII 6	Extuandor Officit:
15	Attributes: 0x00	UINTIS	ExtheaderOllSet,
16	State: 0xf8	UINT8	Reserved[1];
17		UINT8	Revision;
		EFI_FV_BLOCK_MAP	BlockMap[];
		} EFI_FIRMWARE_VOLUME_HE	ADER;

- The Signature of a firmware volume is {'_', 'F', 'V', 'H'}
- The signature field only applies to Firmware Volumes

🔚 Read	Me.txt 🗵 🔚 efipwn.txt 🗵		
1	EFI_FIRMWARE_VOLUME:		
2	Base Offset: 0x00600000		
3	Header Length: 0x48		
4	Data Length: 0x0001ffb8	typedef struct (
5	Total Length: 0x00020000	cypeder Scruct (Zenette et en [16] :
6	Signature: _FVH	UINT8	zerovector[10];
7	Attributes: 0xffff8eff	EFI_GUID	FileSystemGuid;
8		UINT64	FvLength;
10	EFI FIRMWARE FILE:	UINT32	Signature;
11	Base Offset: 0x0000000	EFI_FVB_ATTRIBUTES_2	Attributes;
12	Length: 0x0001ffa0	UINT16	HeaderLength;
13	GUID: 0xcef5b9a3-476d-497f-9fdc-e98143e04	UINT16	Checksum;
14	Type: RAW (0x01)	UINT16	ExtHeaderOffset.
15	Attributes: 0x00		Decentradia 1
16	State: 0xf8	UINT8	Reservea[1];
17		UINT8	Revision;
		EFI_FV_BLOCK_MAP	BlockMap[];
	} EFI_FIRMWARE_VOLUME_HEADER;		ADER;

• The attributes field declares capabilities and power-on defaults for the firmware volume
EFIPWN 'print': Firmware Volume

// Attributes bit definitions #define EFI FVB2 READ DISABLED CAP 0x00000001 #define EFI FVB2 READ ENABLED CAP 0x0000002 #define EFI FVB2 READ STATUS 0x00000004 #define EFI FVB2 WRITE DISABLED CAP 0x00000008 #define EFI FVB2 WRITE ENABLED CAP 0x0000010 #define EFI FVB2 WRITE STATUS 0x0000020 #define EFI FVB2 LOCK CAP 0x0000040 #define EFI FVB2 LOCK STATUS 0x0000080 #define EFI FVB2 STICKY WRITE 0x00000200 #define EFI FVB2 MEMORY MAPPED 0x00000400 #define EFI_FVB2_ERASE_POLARITY 0x00000800 #define EFI FVB2 READ LOCK CAP 0x00001000 #define EFI FVB2 READ LOCK STATUS 0x00002000 #define EFI FVB2 WRITE LOCK CAP 0x00004000 #define EFI_FVB2_WRITE_LOCK_STATUS 0x00008000 #define EFI FVB2 ALIGNMENT 0x001F0000 #define EFI FVB2 WEAK ALIGNMENT 0x8000000 #define EFI FVB2 ALIGNMENT 1 0x0000000 #define EFI FVB2 ALIGNMENT 2 0x00010000 #define EFI_FVB2_ALIGNMENT_4 0x00020000 #define EFI FVB2 ALIGNMENT 8 0x00030000 #define EFI FVB2 ALIGNMENT 16 0x00040000 #define EFI FVB2 ALIGNMENT 32 0x00050000 #define EFI FVB2 ALIGNMENT 64 0x00060000 #define EFI_FVB2_ALIGNMENT_128 0x00070000 #define EFI FVB2 ALIGNMENT 256 0x00080000 #define EFI FVB2 ALIGNMENT 512 0x00090000

#define EFI FVB2 ALIGNMENT 1K 0x000A0000 #define EFI FVB2 ALIGNMENT 2K 0x000B0000 #define EFI FVB2 ALIGNMENT 4K 0x000C0000 #define EFI_FVB2_ALIGNMENT 8K 0x000D0000 #define EFI FVB2 ALIGNMENT 16K 0x000E0000 #define EFI FVB2 ALIGNMENT 32K 0x000F0000 #define EFI FVB2 ALIGNMENT 64K 0x00100000 #define EFI FVB2 ALIGNMENT 128K 0x00110000 #define EFI FVB2 ALIGNMENT 256K 0x00120000 #define EFI FVB2 ALIGNMENT 512K 0x00130000 #define EFI FVB2 ALIGNMENT 1M 0x00140000 #define EFI FVB2 ALIGNMENT 2M 0x00150000 #define EFI FVB2 ALIGNMENT 4M 0x00160000 #define EFI FVB2 ALIGNMENT 8M 0x00170000 #define EFI FVB2 ALIGNMENT 16M 0x00180000 #define EFI_FVB2_ALIGNMENT_32M 0x00190000 #define EFI FVB2 ALIGNMENT 64M 0x001A0000 #define EFI FVB2 ALIGNMENT 128M 0x001B0000 #define EFI FVB2 ALIGNMENT 256M 0x001C0000 #define EFI FVB2 ALIGNMENT 512M 0x001D0000 #define EFI_FVB2_ALIGNMENT_1G 0x001E0000 #define EFI FVB2 ALIGNMENT 2G 0x001F0000

 Defined in Vol. 3 Shared Architectural Elements

EFIPWN 'print': Firmware Files

🔚 Read Me.t	bt 🗵 📄 efipwn.txt 🗵		
1 EF1	I_FIRMWARE_VOLUME:		
2	Base Offset: 0x00600000		
3	Header Length: 0x48		
4	Data Length: 0x0001ffb8	<pre>typedef struct {</pre>	
5	Total Length: 0x00020000	EFI GUID	Name:
6	Signature: _FVH	PET EES INMECRIMY CUECK	IntegrityCheck:
7	Attributes: 0xffff8eff	EFI_FFS_INTEGRITI_CHECK	-
8		EFI_FV_FILETYPE	Type;
9		EFI_FFS_FILE_ATTRIBUTES	Attributes;
10	EFI FIRMWARE FILE:	UINT8	Size[31:
11	Base Offset: 0x0000000		State:
12	Length: 0x0001ffa0	LEI_FES_FIDE_STATE	State,
13	GUID: 0xcef5b9a3-476d-497f-9fdc-e98143e0422c	<pre>} EFI_FFS_FILE_HEADER;</pre>	
14	Type: RAW (0x01)		
15	Attributes: 0x00		
16	State: 0xf8		
3 4 5 6 7 8 9 10 11 12 13 14 15 16	<pre>Header Length: 0x48 Data Length: 0x0001ffb8 Total Length: 0x00020000 Signature: _FVH Attributes: 0xffff8eff EFI FIRMWARE FILE: Base Offset: 0x00000000 Length: 0x0001ffa0 GUID: 0xcef5b9a3-476d-497f-9fdc-e98143e0422c Type: RAW (0x01) Attributes: 0x00 State: 0xf8</pre>	<pre>typedef struct { EFI_GUID EFI_FFS_INTEGRITY_CHECK EFI_FV_FILETYPE EFI_FFS_FILE_ATTRIBUTES UINT8 EFI_FFS_FILE_STATE } EFI_FFS_FILE_HEADER;</pre>	Name; IntegrityCl Type; Attributes; Size [3] ; State;

- Firmware files are code and/or data stored within firmware volumes
- Combined, Firmware Files are described/contained within a Firmware File System
- Base offset refers to its relative location within the volume
- Length refers to the length of the file
- GUID is its ID

EFIPWN 'print': Firmware File

E ReadMe.txt 🗵 🖶 efipwn.txt 🗵						
	1	EFI_FIRMWARE_VOLUME:	Name		Value	Description
	2	Base Offset: 0x00600000	EFI FV FIL	ETYPE RAW	0x01	Binary data
	3	Data Length: 0x0001ffb8	EFI FV FIL	ETYPE FREEFORM	0x02	Sectioned data
	5	Total Length: 0x00020000	EFI FV FIL	ETYPE SECURITY CORE	0x03	Platform core code used during
	6	Signature: _FVH				the SEC phase
	7	Attributes: 0xffff8eff	EFI_FV_FIL	ETYPE_PEI_CORE	0x04	PEI Foundation
	9		EFI_FV_FIL	ETYPE_DXE_CORE	0x05	DXE Foundation
1	10	EFI_FIRMWARE_FILE:	EFI_FV_FIL	ETYPE_PEIM	0x06	PEI module (PEIM)
1	11	Base Offset: 0x00000000	EFI_FV_FIL	ETYPE_DRIVER	0x07	DXE driver
1	12 Length: 0x0001ffa0					
1	13 GUID: 0xcef5b9a3-476d-497f-9fdc-e98143e0422c • •					
1	14	Type: RAW (0x01)	EFI FV FIL	ETYPE FFS PAD	0xF0	Pad File For FFS
1	15	Attributes: 0x00				
1	16	State: 0xf8				
1	17					

- There are different enumerated types of Firmware Files
- Defined in Vol3 Shared Architectural Elements Section 2.1.4.1

EFIPWN 'print': Firmware Files

🔚 Rea	dMe.txt 🗵 🔚 efipwn.txt 🗵		
1	EFI_FIRMWARE_VOLUME:		
2	Base Offset: 0x00600000		
3	Header Length: 0x48		
4	Data Length: 0x0001ffb8		
5	Total Length: 0x00020000	// FFS File Attributes	
6	Signature: _FVH	#define FFS ATTRIB LARGE FILE	0x01
7	Attributes: 0xffff8eff		004
8		#define FFS_ATTRIB_FIXED	0x04
9		#define FFS ATTRIB DATA ALIGNMENT	0x38
10	EFI_FIRMWARE_FILE:	#define FFS ATTRIE CHECKSUM	0x40
11	Base Offset: 0x0000000		
12	Length: 0x0001ffa0		
13	GUID: 0xcef5b9a3-476d-497f-9fdc-e98143e0	122c	
14	Type: RAW (0x01)		
15	Attributes: 0x00		
16	State: 0xf8		
17			

- Firmware Files have attributes like Firmware Volumes do (and are the same)
- The State of the file is intended to preserve integrity

EFIPWN 'print': Firmware Files

🔚 Rea	dMe.txt 🗵 🔚 efipwn.txt 🗵	
1	EFI_FIRMWARE_VOLUME:	
2	Base Offset: 0x00600000	// FFS File State Bits
3	Header Length: 0x48	#define EFT FILE HEADER CONSTRUCTION 0x01
4	Data Length: 0x0001ffb8	
5	Total Length: 0x00020000	#define EFI_FILE_HEADER_VALID 0x02
6	Signature: _FVH	#define EFI_FILE_DATA_VALID 0x04
7	Attributes: 0xffff8eff	#define EFI FILE MARKED FOR UPDATE 0x08
8		#define EFI FILE DELETED 0x10
10	EFI_FIRMWARE_FILE:	#define EFI_FILE_HEADER_INVALID 0x20
11	Base Offset: 0x0000000	
12	Length: 0x0001ffa0	
13	GUID: 0xcef5b9a3-476d-497f-9fdc-e98143e0	Dita C. 7 and recommed bits
14	Type: RAW (0x01)	Bits 6, 7 are reserved bits.
15	Attributes: 0x00	
16	State: 0xf8	
17		

 You can see that it includes the provision for marking files as deleted, which is kind of interesting. But unlike filesystem forensics, all these tools should basically show you all files, whether they're deleted or not