Bootkits: Past, Present & Future

Eugene Rodionov
@vxradius

David Harley
@DavidHarleyBlog

Alexander Matrosov
@matrosov
Agenda

- Modern Bootkits History
  - Legacy BIOS vs. UEFI Boot Environment & Proof of Concept vs. In the Wild
  - Legacy BIOS Bootkit Classification

- UEFI Bootkits
  - Bootkit Implementation Strategies

- Attacks against Secure Boot

- Forensic Software
  - HiddenFsReader
  - CHIPSEC
Modern Bootkit History

In the Wild

Proof of Concept

Microsoft x64 platform gains popularity

Secure Boot implemented in Windows 8
Legacy BIOS vs. UEFI

- No more MBR and VBR/IPL code
- Different hard drive partitioning scheme: GPT (GUID Partition Table)
- Secure Boot technology is implemented in Windows 8
The Target of Modern Bootkits (MBR/VBR)
Classification of MBR/VBR Bootkits

Bootkits

MBR
- MBR Code modification
  - TDL4
- Partition Table modification
  - Olmasco

VBR/IPL
- IPL Code modification
  - Rovnix
- BIOS Parameter Block modification
  - Gapz
IPL Code Modification: Rovnix

- Win64/Rovnix overwrites bootstrap code of the active partition

“Hasta La Vista, Bootkit: Exploiting the VBR”
Gapz VBR Bootkit

Main features:

- Relies on Microsoft Windows VBR layout
- The infections result in modifying only 4 bytes of VBR
- The patched bytes might differ on various installations

“Mind the Gapz: The most complex bootkit ever analyzed?”
Gapz BPB Layout

![BPB Layout Diagram](Image)

**HiddenSectors field of BPB**

**VBR of the active partition**

---

```
struct BIOS_PARAMETER_BLOCK {
    WORD BytesPerSector;
    BYTE SecPerCluster;
    WORD ReservedSectors;
    BYTE Reserved[5];
    BYTE MediaDescriptorID;
    WORD Reserved2;
    WORD SectorsPerTrack;
    WORD NumberOfHeads;
    DWORD HiddenSectors;
    DWORD Reserved3[2];
    LONGLONG TotalSectors;
    LONGLONG StartingCluster;
    LONGLONG MFTMirrStartingCluster;
    DWORD ClustersPerMFTRecord;
    DWORD ClustersPerIndexBuffer;
    LONGLONG VolumeSerialNumber;
    DWORD Reserved4;
};
```
Gapz

Number of “Hidden Sectors”

before infection
after infection

Modified value of number of “Hidden Sectors”
<table>
<thead>
<tr>
<th>Functionality</th>
<th>Gapz</th>
<th>Olmarik (TDL4)</th>
<th>Rovnix (Cidox)</th>
<th>Goblin (XPAJ)</th>
<th>Olmasco (MaxSS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBR modification</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>VBR modification</td>
<td>✓</td>
<td></td>
<td>±</td>
<td>±</td>
<td>±</td>
</tr>
<tr>
<td>Hidden file system type</td>
<td>FAT32</td>
<td>custom</td>
<td>FAT16 modification</td>
<td>custom (TDL4 based)</td>
<td>custom</td>
</tr>
<tr>
<td>Crypto implementation</td>
<td>AES-256, RC4, MD5, SHA1, ECC</td>
<td>XOR/RC4</td>
<td>Custom (XOR+ROL)</td>
<td></td>
<td>RC6 modification</td>
</tr>
<tr>
<td>Compression algorithm</td>
<td>✓</td>
<td>±</td>
<td>aPlib</td>
<td>aPlib</td>
<td>±</td>
</tr>
<tr>
<td>Custom TCP/IP network stack</td>
<td>✓</td>
<td>±</td>
<td>✓</td>
<td>✓</td>
<td>±</td>
</tr>
</tbody>
</table>
HiddenFsReader as a Forensic Tool (MBR/VBR)

ESET Hidden File System Reader
1.0.3.1 (Apr 30 2013 16:31:34)
Copyright © 1992-2013 ESET, spol. s r.o. All rights reserved.

Processing... Please wait.
Parsing file systems...

"Gapx_VBR" file system found:
- vbr_original
- payload.bin
- cfg
- bie
- overlord32.dll
- overlord64.dll
- conf.z
- e59df022
- vbr_infected

File system(s) successfully exported!
HiddenFsReader as a Forensic Tool (MBR/VBR)

ESET Hidden File System Reader
1.0.3.1 (Apr 30 2013 16:31:34)

Copyright © 1992–2013 ESET, spol. s r.o. All rights reserved.

HfsReader.exe [params] [export_path]

Params:
/help or /?    - print help message
/no-output    - no output to command line
/no-export    - do not export files from file system(s)
/export-txt  - export file list from file system(s) to text file
/mbr          - make mbr dump
/vbr          - make active drive vbr dump
/dump=<o>,<s> - make hard drive dump
              - offset from beginning or "end"
              - size
Examples:
    /dump=512,1024
    /dump=end,4096

/zip           - pack all files into zip archive

Supported Hidden File Systems:

Win32/0lmariK (TDL3/TDL4/TDL4)
Win32/0lmasco (MaxUSB)
Win32/0siref (ZeroAccess)
Win32/RouinIX
Win32/0paj
Win32/0pez
Win32/0flamer
Win32/0urelas (GBPBoot)
Then World Moved to UEFI...
In The Beginning...

In 1998-99 **CIH (Chernobyl) virus** written by a student of Taipei Tatung Institute of Technology in Taiwan infected ~60 million PCs

CIH (Chernobyl) **erased BIOS ‘ROM’** boot block and boot sectors on a hard drive causing ~1B US dollars in damage
Signed BIOS Updates Are Rare

- **Mebromi** malware includes BIOS infector & MBR bootkit components
- Patches BIOS ROM binary injecting malicious ISA Option ROM with legitimate BIOS image mod utility
- Triggers SW SMI 0x29/0x2F to erase SPI flash then write patched BIOS binary

No Signature Checks of OS boot loaders (MBR/VBR)

- No concept of Secure or Verified Boot
- Wonder why **TDL4** and likes flourished?
UEFI BIOS Firmware

**CPU Reset**
- SEC
  - Pre-EFI Init (PEI)
    - S-CRTM; Init caches/MTRRs; Cache-as-RAM (NEM); Recovery; TPM Init
    - S-CRTM: Measure DXE/BDS
    - Early CPU/PCH Init
    - Memory (DIMMs, DRAM) Init, SMM Init
  - Driver Exec Env (DXE)
    - Continue initialization of platform & devices
    - Enum FV, dispatch drivers (network, I/O, service..)
    - Produce Boot and Runtime Services
- Boot Dev Select (BDS)
- Boot Manager (Select Boot Device)
  - EFI Shell/Apps; OS Boot Loader(s)

**ExitBootServices**
- ACPI, UEFI SystemTable, SMBIOS table
- Runtime / OS

- Minimal UEFI services (Variable)
UEFI Bootkits

- OS Kernel / Drivers
- UEFI OS Loaders
- UEFI DXE Core / Dispatcher
- System Firmware (SEC/PEI)
- Hardware
  - I/O
  - Memory
  - Network
  - Graphics
- UEFI Boot Loader
  - Bootx64.efi
  - Bootmgfw.efi

Malware

HDD

I/O
Memory
Network
Graphics
UEFI Bootkits
Replacing Windows Boot Manager
EFI System Partition (ESP) on Fixed Drive
ESP\EFI\Microsoft\Boot\bootmgfw.efi
UEFI technology: say hello to the Windows 8 bootkit! by ITSEC

Replacing Fallback Boot Loader
ESP\EFI\Boot\bootx64.efi
UEFI and Dreamboot by Sébastien Kaczmarek, QUARKSLAB

Adding New Boot Loader (bootkit.efi)
Modified BootOrder / Boot#### EFI variables
UEFI Bootkits
Adding/Replacing DXE Driver

- Stored on Fixed Drive
- Not embedded in Firmware Volume (FV) in ROM
- Modified DriverOrder + Driver#### EFI variables
UEFI Bootkits

- UEFI DXE Core / Dispatcher
- UEFI OS Loaders
- UEFI OROM
- Bootx64.efi
- Bootmgfw.efi
- OS Kernel / Drivers
- Hardware
  - I/O
  - Memory
  - Network
  - Graphics
- DXE Driver
- UEFI Boot Loader
- HDD
- Malware
- DXE Driver
- OROM
Patching UEFI “Option ROM”
UEFI DXE Driver in Add-On Card (Network, Storage..)
Non-Embedded in FV in ROM

Mac EFI Rootkits by @snare, Black Hat USA 2012
UEFI Bootkits

Replacing OS Loaders (winload.efi, winresume.efi)

Patching GUID Partition Table (GPT)
UEFI Bootkits
What about Secure Boot?
Secure Boot on MS Windows 8.1

Signed BIOS Update

Hardware
- I/O
- Memory
- Network
- Graphics

System Firmware (SEC/PEI)

UEFI DXE Core / Dispatcher

UEFI OROM

UEFI OS Loaders (winload.efi, winresume.efi)

UEFI

UEFI App

OS Kernel / Early Launch Anti-Malware (ELAM)

OS Driver

Windows 8.1 Secure Boot

UEFI Boot Loader

UEFI Boot Loader

UEFI Boot Loader

Signed BIOS Update

Windows 8.1 Secure Boot

Signed BIOS Update

Windows 8.1 Secure Boot

Signed BIOS Update
Secure Boot bypass possible?

Signed BIOS Update

OS Driver
OS Exploit
OS Kernel
UEFI OS Loaders
DXE Driver

UEFI Boot Loader
Bootx64.efi
Bootmgfw.efi

Modify Secure Boot FW or config in ROM

UEFI DXE Core / Dispatcher
System Firmware (SEC/PEI)

Hardware
I/O Memory Network Graphics
First Public Windows 8 Secure Boot Bypass (Aug 2013)

A Tale Of One Software Bypass Of Windows 8 Secure Boot
By booting this system up you agree
to have no expectation of privacy
in any communications or data,
transiting or stored on this system.
any communications or data may be
monitored, intercepted, recorded
and may be disclosed for any purpose.

press any key to continue...
Summary of Attacks Against BIOS and Secure Boot
Subzero Security Patching

“1-days from Hell... get it?”

Patch attempts to enable BIOS write protection (sets BIOS_CONTROL[BLE]). Picked up by Subzero

From Analytics, and Scalability, and UEFI Exploitation by Teddy Reed
CHIPSEC
Platform Security Assessment Framework

https://github.com/chipsec/chipsec
@CHIPSEC
CHIPSEC: Platform Security Assessment Framework
CHIPSEC: Platform Security Assessment Framework

chipsec_main.py runs modules (see modules dir below)
chipsec_util.py runs manual utilities (see utilcmd dir below)
/chipsec
  /cfg platform specific configuration
  /hal all the HW stuff you can interact with
  /helper support for OS/environments
  /modules modules (tests/tools/PoCs) go here
  /utilcmd utility commands for chipsec_util
## Known Threats and CHIPSEC modules

<table>
<thead>
<tr>
<th>Issue</th>
<th>CHIPSEC Module</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMRAM Locking</td>
<td>common.smm</td>
<td>CanSecWest 2006</td>
</tr>
<tr>
<td>BIOS Keyboard Buffer Sanitization</td>
<td>common.bios_kbrd_buffer</td>
<td>DEFCON 16 2008</td>
</tr>
<tr>
<td>SMRR Configuration</td>
<td>common.smrr</td>
<td>ITL 2009</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CanSecWest 2009</td>
</tr>
<tr>
<td>BIOS Protection</td>
<td>common.bios_wp</td>
<td>BlackHat USA 2009</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CanSecWest 2013</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Black Hat 2013</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NoSuchCon 2013</td>
</tr>
<tr>
<td>SPI Controller Locking</td>
<td>common.spi_lock</td>
<td>Flashrom</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Copernicus</td>
</tr>
<tr>
<td>BIOS Interface Locking</td>
<td>common.bios_ts</td>
<td>PoC 2007</td>
</tr>
<tr>
<td>Access Control for Secure Boot Keys</td>
<td>common.secureboot.keys</td>
<td>UEFI 2.4 Spec</td>
</tr>
<tr>
<td>Access Control for Secure Boot Variables</td>
<td>common.secureboot.variables</td>
<td>UEFI 2.4 Spec</td>
</tr>
</tbody>
</table>
BIOS/Firmware Forensics

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
How to dump BIOS firmware directly from chip?
How to dump BIOS firmware directly from chip?
DEMO TIME
Advanced Malware Analysis

Book is coming in 2015!
Stay Tuned ;}
Thank you for your attention!

Alexander Matrosov
@matrosov

Eugene Rodionov
@vxradius

David Harley
@DavidHarleyBlog