Peter Kleissner

Hibernation File Attack

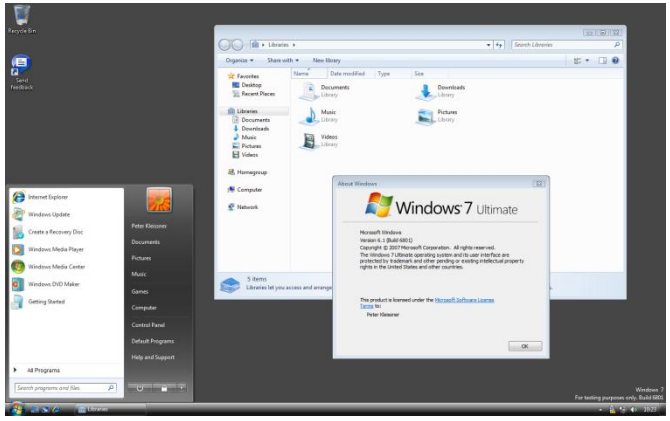
Black Hat Europe 2009

Who am I?

- Independent Operating System developer
  - Written an Operating System totally in Assembler (which is also the code base for this)
- Software Engineer / Malware Analyst for Ikarus Security Software (Vienna)
- Living in Wiener Neudorf – a suburb of Vienna (Austria)

About

- Compromising Windows 7 security
  - And Windows XP, Windows Server 2003, Windows Vista and Windows Server 2008 too ☺
- Loading malicious unsigned code into kernel
- Everything done from a running Windows
- No cheating :P



Microsoft

Windows 7 Developer Guide

Published October 2008

For more information, press only:
Rapid Response Team
Waggener Edstrom Worldwide
(503) 443-7070
rrt@waggeneredstrom.com

The Idea

- To patch the hibernation file („hiberfil.sys“) with our unsigned Kernel Code, which is loaded on startup by winresume.exe
- Modify hiberfil.sys via own Master Boot Record and inject code
- Write the MBR to disk via normal application using raw sector access

Hibernation

- To shut down PC and suspend-to-disk
- ACPI power state S4 “Non-Volatile Sleep”
- Windows:
  - powrprof.dll!SetSuspendState()
  - powercfg -h ON|OFF
  - shutdown -h
  - Hiberfil.sys contains physical memory and processor state

Hibernation API Functions

- kernel32.dll

```
BOOL WINAPI SetSystemPowerState(
  __in  BOOL fSuspend,
  __in  BOOL fForce
);
```

Marked as deprecated

- powrprof.dll

```
BOOLEAN WINAPI SetSuspendState(
  __in  BOOLEAN Hibernate,
  __in  BOOLEAN ForceCritical,
  __in  BOOLEAN DisableWakeEvent
);
```

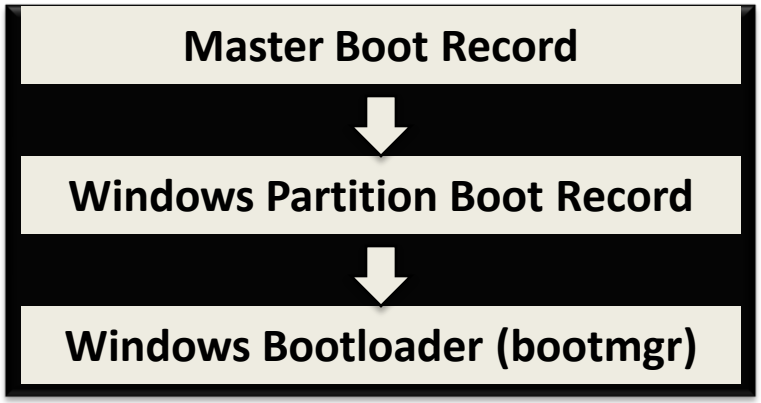
```
BOOLEAN WINAPI IsPwrHibernateAllowed(void);
```

deprecated

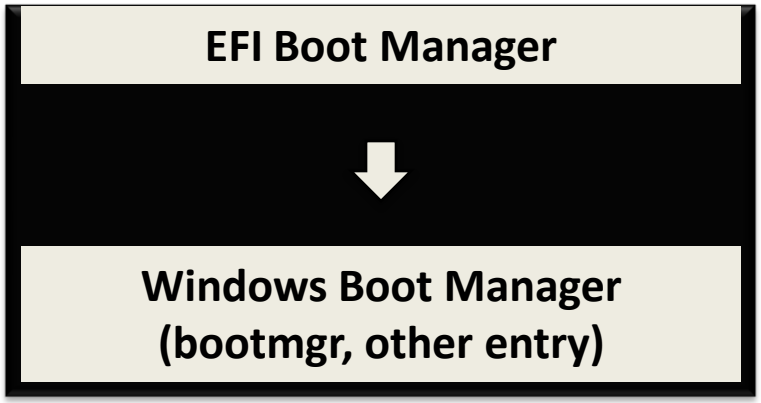
```
BOOLEAN WINAPI GetPwrCapabilities(
  __out  PSYSTEM_POWER_CAPABILITIES
lpSystemPowerCapabilities
);
```

Hibernation Resume

- Done by winresume.exe, executed as Boot Application given by Boot Configuration Data
- Winresume.exe executed by bootmgr



IBM-PC conform boot (legacy)



Extensible Firmware Interface

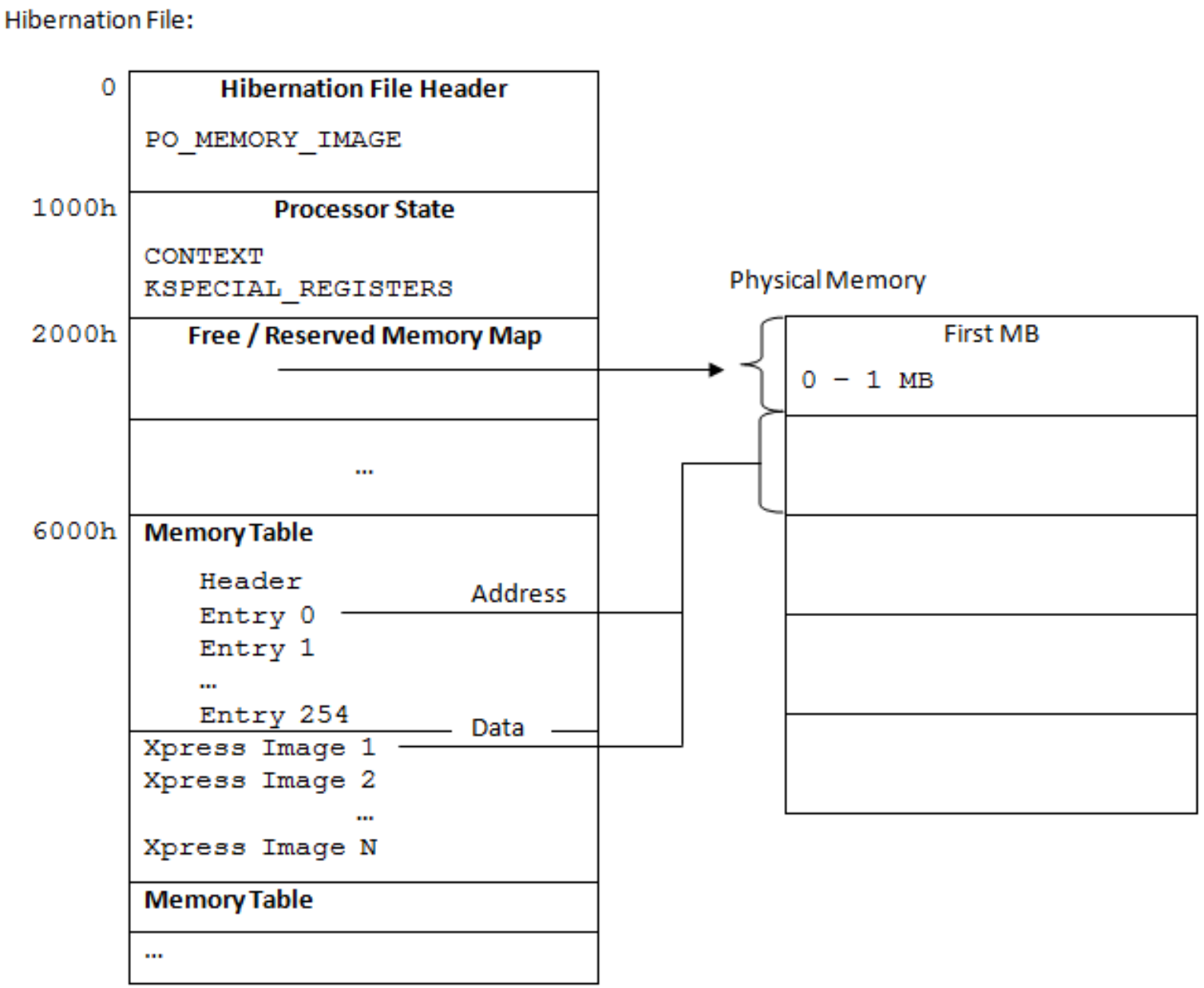
Windows Boot Manager

- Windows Loader (`winload.exe`)
- Windows Resume Loader (`winresume.exe`)

Hibernation File

- C:\hiberfil.sys
- As big as physical memory size
- Contains processor state and physical memory
- Physical memory stored compressed using LZ77 + DIRECT2 (other memory remains free)
- Loaded by winresume.exe during startup
- Everything is page aligned (4096 bytes)
- After resume first page is wiped out
- If never hibernated file is initialized with ‘u’s
- Documented first by Matthieu Suiche, BH US

Hibernation File Format



Hibernation File Format

Field	Content
Hibernation File Header	PO_MEMORY_IMAGE structure Contains information like Signature, Version, first memory map, log fields, etc.
Processor State	CONTEXT and KSPECIAL_REGISTERS structure Contains current processor state (all registers) and system registers (IDTR, GDTR)
Free/Reserved Memory Map	A map of all free and reserved pages
Memory Map 1	Has 255 entries describing memory ranges
Xpress Image 1, Xpress Image 2, …	The physical memory stored compressed Contains a short header and the compressed data
Memory Map 2	
…	

Hibernation File: Signatures

The Hibernation File Header contains in the first 4 bytes an identifier signature:

Signature	Meaning
‘hibr’	Valid Windows XP hibernation file, `ntldr` shall call `osloader.exe` to load hibernation file and process hibernation resume
‘wake’	Invalid Windows XP or lower hibernation file, system shall start normally
‘HIBR’	Valid Windows Vista hibernation file, winresume.exe can process hibernation resume
‘WAKE’	Invalid Windows Vista hibernation file
‘RSTR’	During restoration the state is RSTR (will only occur if hibernation file is erroneous), also used in NTFS; Windows Vista
‘PAGEDUMP’	Windows NT memory dump file
‘XDEC’, ‘SDEC’ or ‘CDEC’	Windows CE dump file

Hibernation File: Memory Maps

- Physical memory is stored in multiple parts in a list of memory maps
- Every memory map contains 255 entries and is followed by one or more Xpress Images
- Xpress Images contain the compressed data and holding 16 pages each image
- We can set checksums to zero to bypass checksum check
- Pages are not ordered
- FirstTablePage in hibernation file header points to first memory map

Hibernation File: Memory Tables

Every memory table is exactly 1 page (4096) bytes big:

```
struct MEMORY_TABLE
{
     DWORD PointerSystemTable;
     UINT32 NextTablePage;
     DWORD CheckSum;
     UINT32 EntryCount = 255;
     MEMORY_TABLE_ENTRY MemoryTableEntries[EntryCount];
};
```

```
struct MEMORY_TABLE_ENTRY
{
     UINT32 PageCompressedData;
     UINT32 PhysicalStartPage;
     UINT32 PhysicalEndPage;
     DWORD CheckSum;
};
```

StartPage to EndPage describes a memory range which data is stored in one or more Xpress Images

Checksums can be set to zero to bypass checksum check when loading

Hibernation File: Xpress Images

- LZ77 + DIRECT2 compression used
- Same as in Microsoft Exchange protocol
- Internal Name XPRESS, described in MS-OXCRPC: Wire Format Protocol Specification
- Xpress Images are following directly one the other

```
struct IMAGE_XPRESS_HEADER
{
     CHAR Signature[8] = 81h, 81h, "xpress";
     BYTE UncompressedPages = 15;
     UINT32 CompressedSize;
     BYTE Reserved[19] = 0;
};
```

Signature = 81h, 81h, “xpress”
Size of compressed data = CompressedSize / 4 + 1
Size of uncompressed data = (UncompressedPages + 1) * 4096

Practice

Things to get working

Step 1	Overwriting Master Boot Record
Step 2	Hibernate Windows
Step 3	Automatic Restart
Step 4	MBR code must modify hibernation file and pass control to boot loader
Step 5	Exploit Code

We bypass

- Kernel Patch Protection

  only for 64 bit and running systems

- Digital Signatures

  we can inject unsigned code, no signed code check will be performed

- Code integrity checks

  we do not patch system executable files, therefore no integrity check will fail

- Data Execution Prevention

Problems

- Raw sector access

  Windows XP: Administrator rights

  Windows Vista: elevated Administrator rights

- File system access

  NTFS, FAT and all its different types

  we need our own file system and disk driver in the MBR to modify the hibernation file

but every problem has its solution…

Raw Sector Access

- Solution 1:

  75% of the users have full admin privileges

> "However, outside the enterprise and the Parental Controls case, most machines (75%) have a single account with full admin privileges."

According to Ben Fathi, Windows 7 User Account Control Engineer

- Solution 2:

  ask to ask for elevated rights

  at runtime or via Manifest

  if user clicks "no" terrorize user and ask again

Elevated Administrator Rights

- Method 1: Application Manifest

```
<requestedPrivileges>
    <requestedExecutionLevel level="asInvoker" uiAccess="true"/>
</requestedPrivileges>
```

…as part of application manifest (can be embedded into application as resource)

```
/MANIFESTUAC:"level=asInvoker"
```

Visual Studio 2008 linker option to automatically generate and include manifest

```
Level to be "asInvoker", "highestAvailable" or "requireAdministrator"
```

- Method 2: ShellExecute() at runtime

```
HINSTANCE ShellExecute(
    HWND hwnd,
    LPCTSTR lpOperation = "runas",
    (…)
);
```

Writing the MBR

- CreateFile("\\.\PhysicalDrive0", …)
- Direct driver usage, IOCTLs
- Also works in Windows Vista and Server 2008:

A file system can write to a volume handle only if the following conditions are true:
Condition 1: The sectors that are being written to are boot sectors.
Condition 2: The sectors that are being written to reside outside the file system space.

According to Microsoft Knowledgebase, Article ID 92448 "Changes to the file system and to the storage stack to restrict direct disk access and direct volume access in Windows Vista and in Windows Server 2008"

- 63 Sectors (31.5 KB size, Sectors 0 – 62)

Conclusion: What we can and can't do

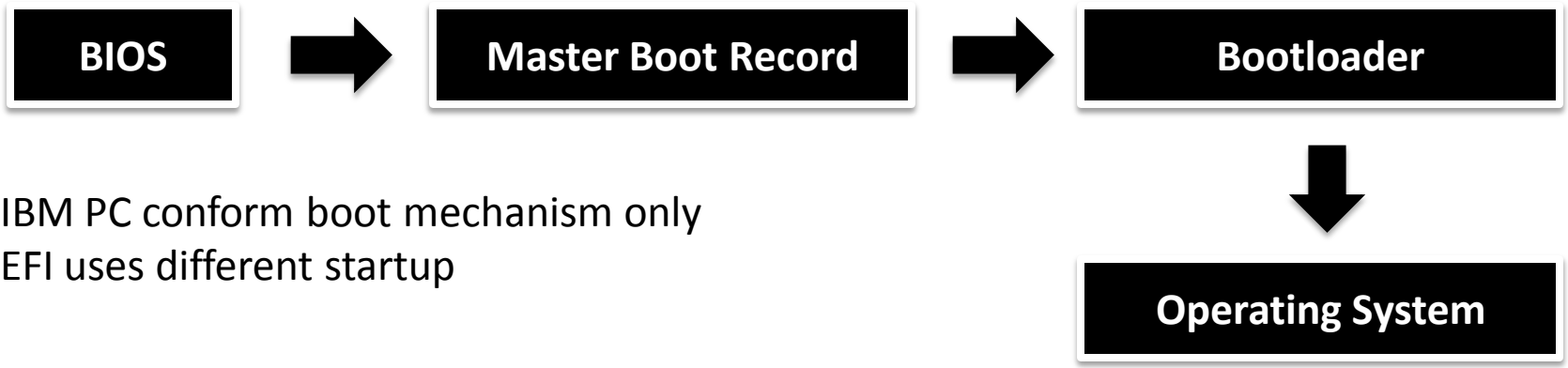
Windows 7 with User Account Control turned on and as Administrator			
	asInvoker	highestAvailable	requireAdministrator
Disk Open	✔	✔	✔
Read	✘	✔	✔
Write	✘	✔	✔
Prompt at	Never	Begin	Begin
Conclusion	Fails	Works if user clicks "Yes" on Consent UI	

Windows 7 without UAC, as Administrator			
	asInvoker	highestAvailable	requireAdministrator
Disk Open	✔	✔	✔
Read	✔	✔	✔
Write	✔	✔	✔
	Works (without a prompt)		

Master Boot Record

Master Boot Record

- First 63 sectors of hard disk
- Loaded by the BIOS, passes control to the Partition Bootloader (of the Operating System)
- Contains low level system code
- Partition Table
- GUID Partition Table



IBM PC conform boot mechanism only
EFI uses different startup

About the MBR Software

- Modularized design with Kernel Modules
- Based on the Forensic Lockdown Software
- It’s a small operating system
- Totally independent of other software
- Has its own file system drivers for FAT and NTFS (all types of them)
- And there is still space for future modules!

Kernel Modules

API

Boot Module

Bootloader

Crypto Module

Disk System

Locking Module

Rescue Module

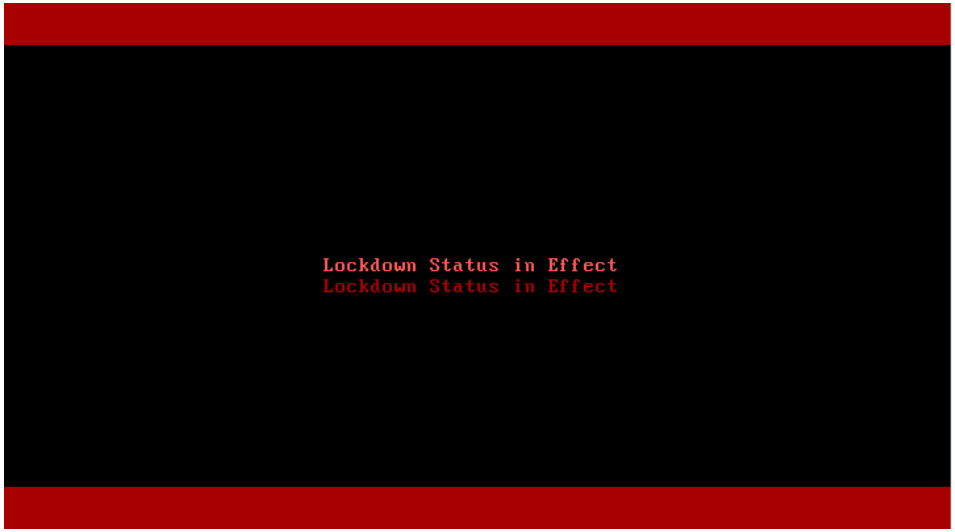
System Loader

Textmode TUI

User Interface

Special Hibernation File Attack Module
Application is added for Hibernation File Attack

Forensic Lockdown Software



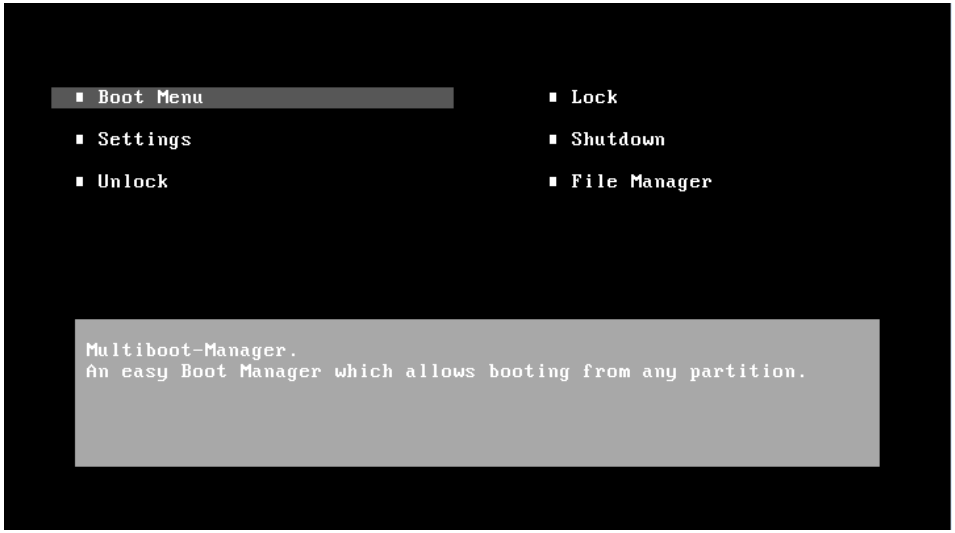
2007 developed, now code base for Hibernation File Attack's Master Boot Record

Developed by me, Peter Kleissner

Brings rich API functions and support

Was intended to totally lock down a machine by encrypting Partition Table and Boot Code, changing BIOS password and boot order and locking hard disk with a password among other things

Perfect for Hibernation File Attack!



Storage Device Driver

- Very central thing for the operation
- Supports FAT and NTFS, also write support
- Alpha compression of NTFS supported
- 7000 lines long, written in Assembler
- Occupies 8 KB of the MBR, biggest module
- Supports also multiple drives, file handles

Functions exported:

```
Mount Boot Partitions          Write File
Load System File               Seek File
Open File Callback
Read File
```

Executive Application

- Open Hibernation File ("`C:\hiberfil.sys`")
- Scanning for patterns
  - Prediction where memory physically resists
  - Decompress memory and search for pattern
  - If found => replace memory
- Original MBR will be loaded and executed
  - From `C:\Master Boot Record.bak`, created by Infector

Code Injection Technique

- Problem: If new compressed code is bigger than original one, the Xpress Image does not fit
- Solution: an additional Memory Map for the modified code is appended to the Hibernation File (physical memory appears twice in file)

- Problem: How to know what to patch?
- Solution: Using the VERV's virus-description language

Virus-Description Language (VDL)

- From “Virus Verification and Removal Tools and Techniques” publication
- David M. Chess for IBM Research Center, November 18, 1991

```
One or more VIRUS records
A NAME record
One or more LOAD records
Zero or more DEGARBLE and related
records
Zero or more ZERO records
One or more check records
Zero or more REPAIR blocks
```

```
VIRUS slow slow-1721
NAME the Slow-1721 virus
LOAD P-COM 0 6B4
LOAD S-EXE 0 6B4
DEXOR1 001E 06AD 0012 0000      ; Degarble the code
DEXOR1 00EB 0159 0061 0001      ;   and the data area
ZERO 0012 1                     ; Zero the code-garble key
ZERO 0061 1                     ;   and the data-garble key
CODE  0000 00EA 38d5dc08        ; Code up to first data area
CONST 0144 014E 0ff22ad9        ; COMMAND.COM
CODE  015A 063C 74e00962        ; Code between data areas
CODE  0657 06AD ad3b0b41        ; After the second data area
```

Unsigned Driver Loader

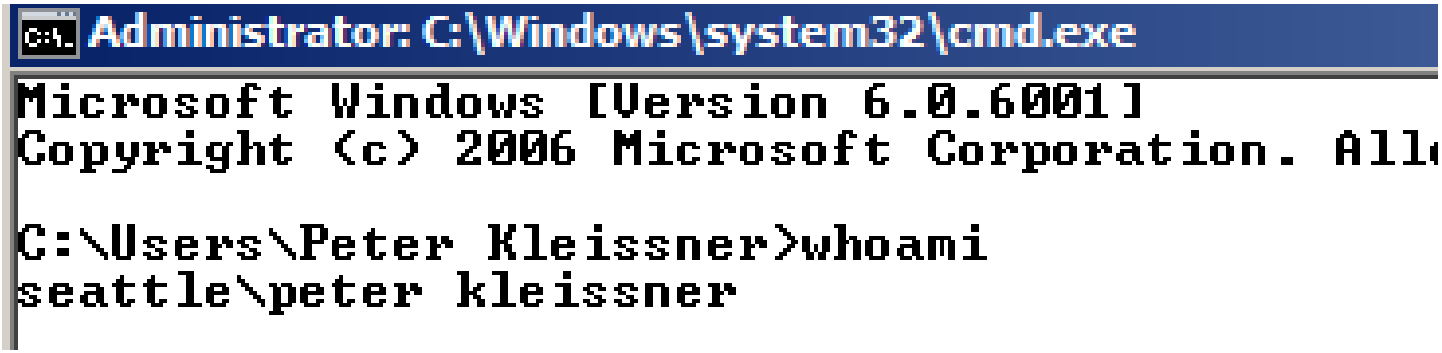
- Stored in MBR => could be out-sourced
- Is injected into Kernel (runs in System Context)
- Hooks ntoskrnl, to get called after resume
- Has its own PE loading and relocation module
- Directly injected into hibernation file:

  => we just hook some bytes in ntoskrnl and will out-source the Unsigned Driver Loader to the memory of the `NULL` device driver ☺
- loads and executes "`C:\Unsigned driver.sys`"

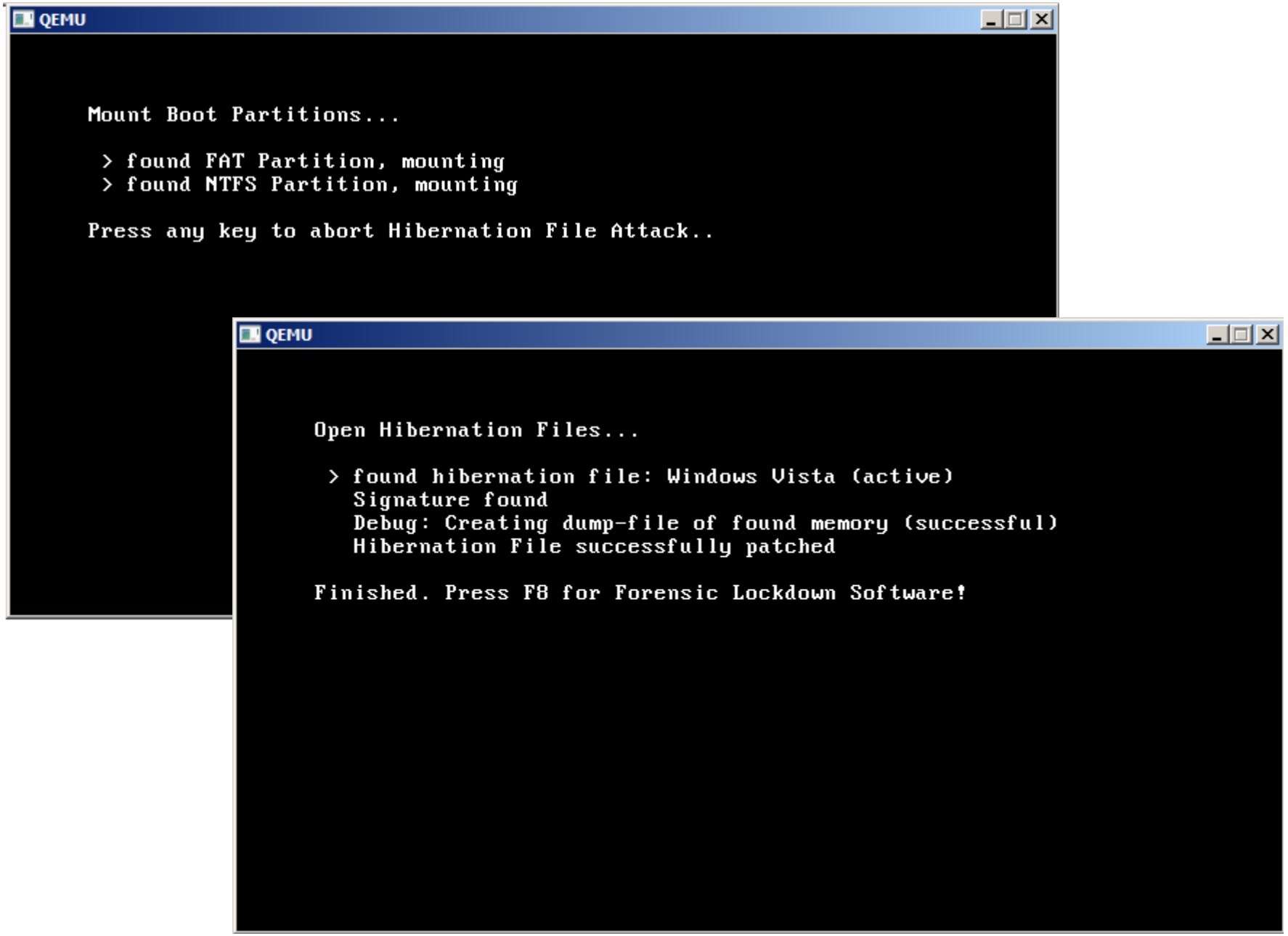
Exploit Code

Exploit Code

- Same as in Vboot Kit, Black Hat Europe 2007 (thanks to my friends Vipin & Nitin Kumar)
- Changes privileges of Console (`cmd.exe`) to services.exe
- Exploit Code runs as Real Time Clock Interrupt Handler – hooks Interrupt Descriptor Table
- Is not a thread or process registered in Windows
- Uses EPROCESS structure assigned to every process to modify permissions



Time for a live demonstration!


QEMU
Mount Boot Partitions...
> found FAT Partition, mounting
> found NTFS Partition, mounting
Press any key to abort Hibernation File Attack..
QEMU
Open Hibernation Files...
> found hibernation file: Windows Vista (active)
Signature found
Debug: Creating dump-file of found memory (successful)
Hibernation File successfully patched
Finished. Press F8 for Forensic Lockdown Software!

Conclusion

Putting everything together…

- Windows hibernation security flaw
- Not to nudge Microsoft, but hibernation has never been considered for any security

1. We can write Boot Sectors
2. Hibernation file is not encrypted, checksums are not used, there is no content verification
3. Windows doesn't check or verify its Kernel when resuming

What to change / how to prevent

- Solution 1: Encrypt Hibernation File

  Negative: no one can read (memory analysis…)

- Solution 2: Disallow Boot Sectors Write Access

  Negative: stoned-vienna no longer working, I don't have fun in future, no on-the-run partitioning

- Best Solution:
  - Load Windows Kernel normally from disk
  - Hibernate/resume only user mode and memory!
  - Verify hibernation file contents

Software Hibernation + Hardware Hibernation is more secure

References

- SandMan project – thanks to Matthieu Suiche
  http://sandman.msuiche.net/
- Hibernation File
  C:\hiberfil.sys
- VERV‘s virus-description language
  http://www.research.ibm.com/antivirus/SciPapers/Chess/CHESS3/chess3-node5.html
- Microsoft Wire Format Specification Protocol (MS-OXCRPC)
  http://msdn.microsoft.com/en-us/library/cc425493.aspx
- Boot Options in Windows Vista
  http://msdn.microsoft.com/en-us/library/aa468626.aspx
- User Account Control, Microsoft documentation
  http://msdn.microsoft.com/en-us/library/bb648649.aspx

Presentation Materials

Hibernation File Attack

www.viennacomputerproducts.com/hibernationfileattack

Peter Kleissner at

Black Hat Europe 2009

Thanks…

… for your attention!

Questions?
Comments?

Peter Kleissner at

Black Hat Europe 2009