# Stoned Bootkit

Peter Kleissner

## Table of Contents

## Who am I?

- **Independent Operating System Developer**
- **Professional Software Engineer and Malware Analyst**
- **Living in Wiener Neudorf, a suburb of Vienna (Austria)**

## Introduction

## About

- **Bootkit = Rootkit + Boot Capability**
  **Introduced by Vipin and Nitin Kumar**
- **Stoned is a new bootkit targeting Windows operating systems**

Windows 2000
Windows XP
Windows Server 2003
Windows Vista
Windows Server 2008
Windows 7 RC
TrueCrypt

Main targets:

- Pwning all Windows versions from the boot
- Being able to bypass code integrity verifications & signed code checks

**www.stoned-vienna.com**

## Architecture

| Address | Size | Description |
|---|---|---|
| 0000 | 440 | Code Area |
| 01B8 | 6 | Microsoft Disk Signature |
| 01BE | 4*16 | IBM Partition Table |
| 01FE | 2 | Signature, 0AA55h |
| 0200 | - | Stoned Kernel Modules |
| - | - | Stoned Plugins |
| 7A00 | 512 | Backup of Original Bootloader |
| 7C00 | 512 | Configuration Area |

+

```
C:.
│   autoexec.bat
│   config.sys
│   ctapi_out_gr.txt
│   ntuser.dat
│
├───Stoned
│   │   Applications
│   │   Plugins
│   │   │

…
```

Master Boot Record

File System

**„A memory resident bootkit up to the Windows kernel“**

**+ Boot applications executed on startup**
**+ Drivers executed beside the Windows kernel**

# Stoned Virus

```
Your PC is now Stoned!                         (1987)
Your PC is now Stoned!  ..again                (2010)
```

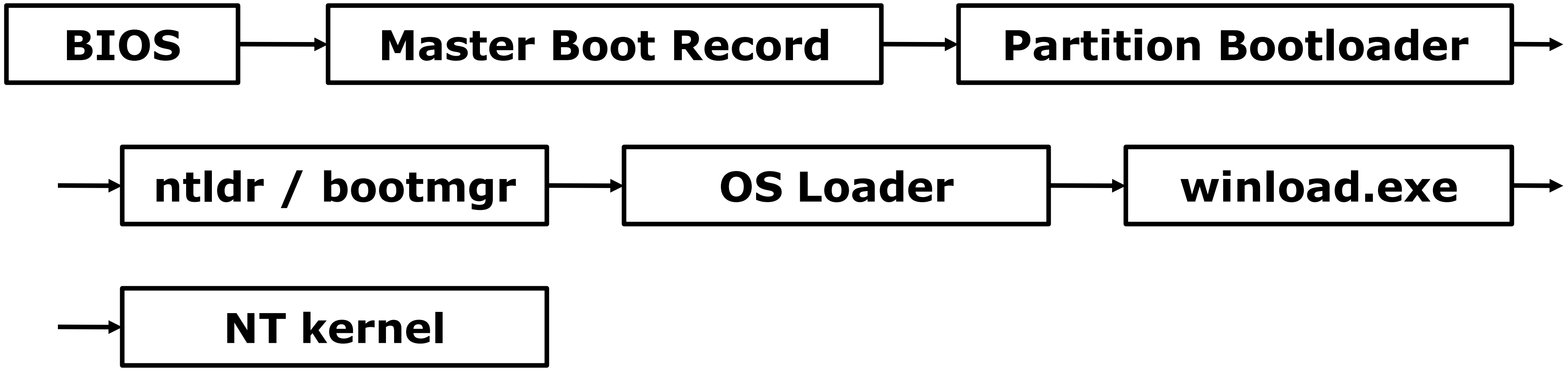
**Stoned is the name of a boot sector computer virus created in 1987, apparently in New Zealand. It was one of the very first viruses, and was, along with its many variants, very common and widespread in the early 1990s.**

**http://en.wikipedia.org/wiki/Stoned_(computer_virus)**

**Stoned was an OS independent boot sector infector.**

- **Probably the first bootkit?**
- **416 bytes of size, small & effective!**

# Windows Boot Process



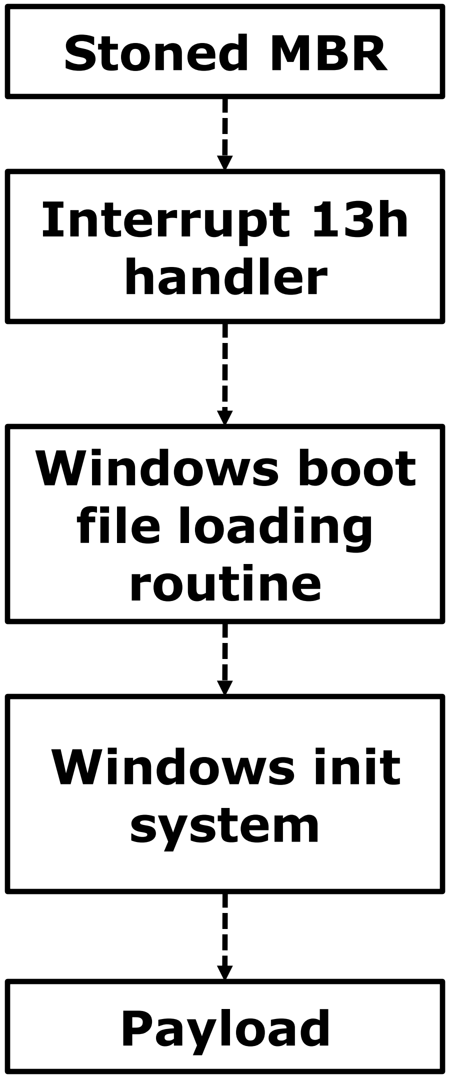
`Ntldr` = 16-bit stub + `OS Loader` (just binary appended)
Windows Vista splits up `ntldr` into `bootmgr`, `winload.exe` and `winresume.exe`

| **Windows XP** | **Windows Vista** | **Processor Environment** |
|---|---|---|
| ntldr | bootmgr | Real Mode |
| OS Loader | OS Loader | Protected Mode |
| - | winload.exe | Protected Mode |
| NT kernel | NT kernel | Protected Mode + Paging |

# Insecuring Windows

## Pwning Windows from the boot

| | |
|---|---|
| Bootkit Real Mode | Relocates the code to the end of memory (4 KB), hooks interrupt 13h and patches code integrity verification |
| Bootkit Protected Mode | Patches image verification and hooks NT kernel |
| Kernel Code | NT kernel base address and PsLoadedModuleList are used for resolving own imports |
| Driver Code | Loads, relocates, resolves, executes all drivers in the list |
| PE Loader | PE-image relocation & resolving |
| Subsystem | Core functions for the Stoned Subsystem installed in Windows |
| Payload | Kernel drivers<br>Applications using the subsystem |

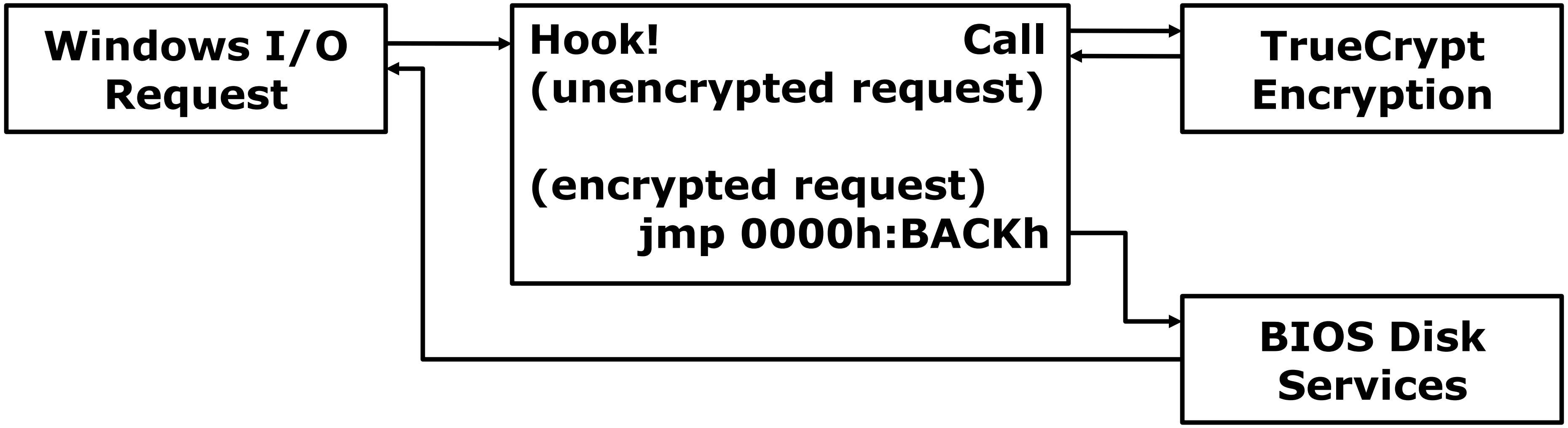


# TrueCrypt Attack

## There are two possible scenarios:

1. Only the system partition is encrypted
2. Full hard disk is encrypted

However, the master boor record always stays unencrypted.



A double forward for intercepting the encrypted and decrypted disk I/O.

# Previous Bootkits

| 1987 | … | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 |
|---|---|---|---|---|---|---|---|
| | | | | Mebroot | | | Stoned Bootkit |
| | | | BOOT KIT | | TPMkit | | |
| Stoned | | BootRoot | | Vbootkit | | Vbootkit 2.0 | |

**Previous research bootkits at conferences:**

| | | |
|---|---|---|
| BootRoot | Windows XP | Black Hat USA 2005 |
| Vbootkit | Windows Vista | Black Hat Europe 2007 |
| Vbootkit 2.0 | Windows 7 (x64) | Hack In The Box Dubai 2009 |

## Stoned Architecture

# Master Boot Record

**Master Boot Record = first 63 sectors of hard disks; contains Partition Table and Bootloader**

**Modularized Stoned MBR contains:**

| Address | Size | Description | |
|---|---|---|---|
| 0000 | 440 | Code Area (Bootloader) | Bootloader.sys |
| 01B8 | 6 | Microsoft Disk Signature | |
| 01BE | 4*16 | IBM Partition Table | |
| 01FE | 2 | Signature, 0AA55h | |
| 0200 | 2 KB | System Loader | System Loader.sys |
| 0A00 | 1 KB | Textmode User Interface | Textmode TUI.sys |
| 0E00 | 8 KB | Disk System | Disk System.sys |
| 2E00 | 2 KB | Load Application Programming Interface for Real Mode | API [RM].sys |
| 3600 | 512 | Rescue Module | Rescue Module.sys |
| 3800 | 8 KB | Free space (former User Interface and Hibernation File Attack) | [Embedded Boot Application] |
| 5800 | 1.5 KB | Crypto Module | Crypto Module.sys |
| 5E00 | 1 KB | Boot Module | Boot Module.sys |
| 6200 | 4 KB | Pwn Windows | Windows.sys |
| 7200 | 2 KB | Free Space | |
| Sector 61 | 512 | Original Bootloader Backup | |
| Sector 62 | 512 | Configuration Area / TrueCrypt volume-header information | |

## Stoned Modules

**Management Modules:**

| | | |
|---|---|---|
| Bootloader | System Loader | Plugin Manager |

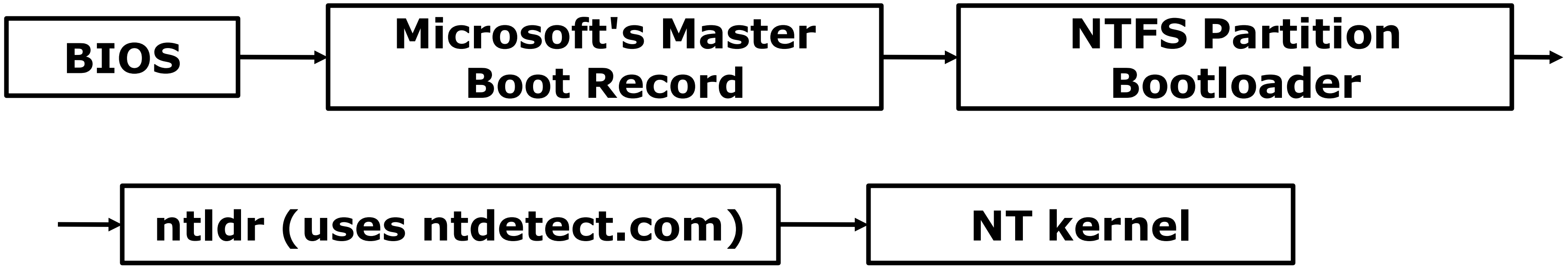
**API providing modules:**

| | | | |
|---|---|---|---|
| API [RM] | Boot Module | Crypto Module | Disk System |
| Locking Module | Rescue Module | Textmode UI | User Interface |

**Boot applications use the API provided by the modules.**

**They are independent from each other (this is also why the Windows pwning module can be injected into TrueCrypt’s MBR).**

# Windows XP Boot

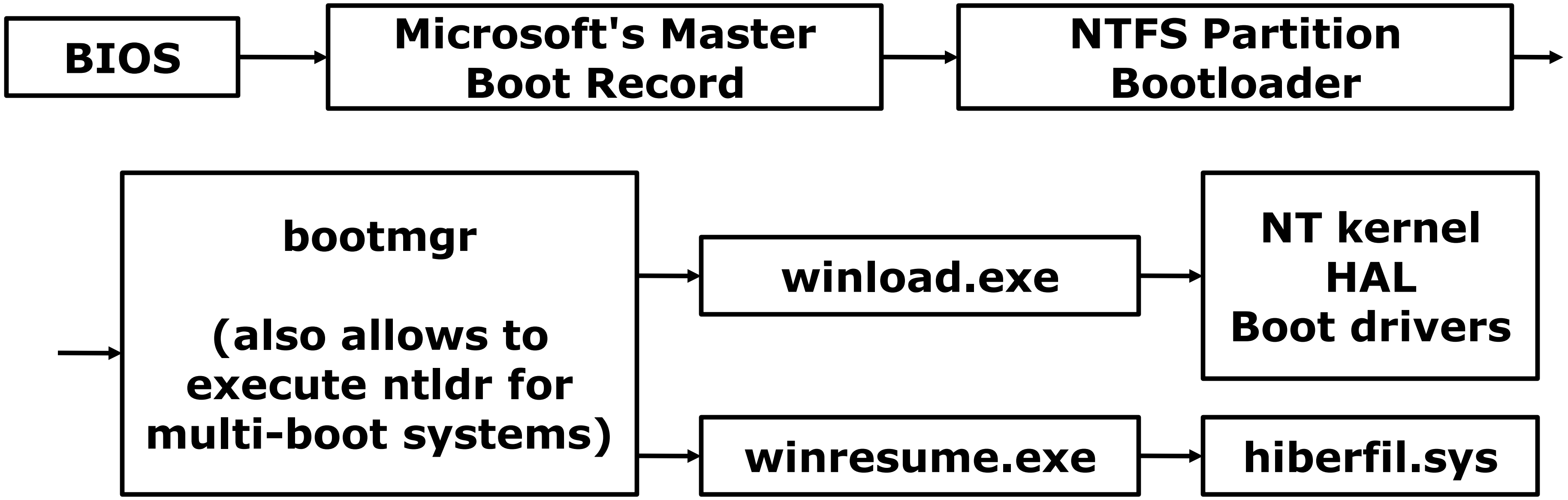

BIOS
Microsoft's Master Boot Record
NTFS Partition Bootloader
ntldr (uses ntdetect.com)
NT kernel

**Ntldr contains a 16-bit stub and a 32 bit PE Image (= OS Loader)**
**This concept has not been changed in Windows until Windows Vista**

## Hooking & Patching (simplified):

- **Interrupt 13h hooked**
- **Ntldr hooked for calling 32-bit code and patching the code integrity verification**
- **Patching the NT kernel**
- **Executing payloads (driver)**

## Windows Vista Boot


BIOS
Microsoft's Master Boot Record
NTFS Partition Bootloader
bootmgr
(also allows to execute ntldr for multi-boot systems)
winload.exe
NT kernel
HAL
Boot drivers
winresume.exe
hiberfil.sys

## Hooking & Patching (simplified):

- Interrupt 13h hooked
- Bootmgr hooked to call 32-bit code
- Patching winload.exe code integrity verifications
- Patching the NT kernel

## Boot Media

- Currently only IBM-conform legacy boot supported
- In future EFI (Extensible Firmware Interface) support

All common drives supported:

Floppy, CD, DVD, Blu-ray, USB flash drives, removable media, hard drives, network boot

➡ Media independent.

## Plugins

## About Plugins

Extending the core functionality by static bootkit attacks

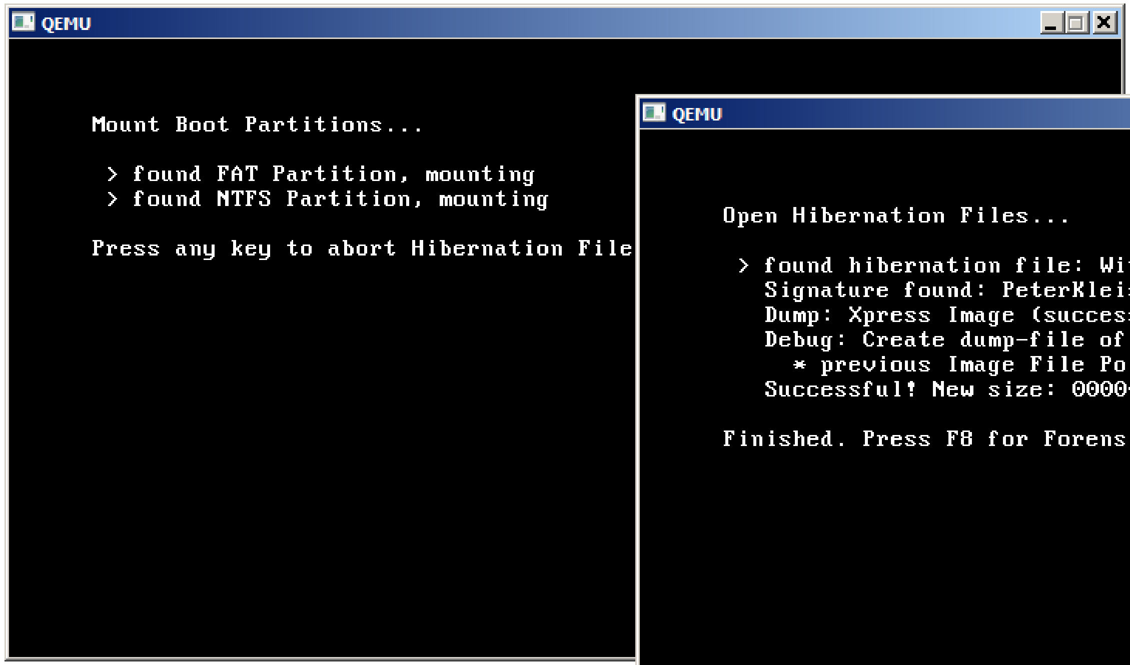
| | | | |
|---|---|---|---|
| User Interface | $CO_2$-Plugin | PE Infector | File Parsers |
| Hibernation File Attack | Pagefile Injector | Music Melody! | Boot Password Crack |
| AntiWPA | Persistent BIOS Infector | …and much more under development | |

May be out sourced to the file system.

User data stored in CMOS memory?

# Hibernation File Attack

- Predecessor of Stoned
- Static attack of bootkit
- Structures were revealed with BH USA 2008 „Windows Hibernation File for Fun and Profit"

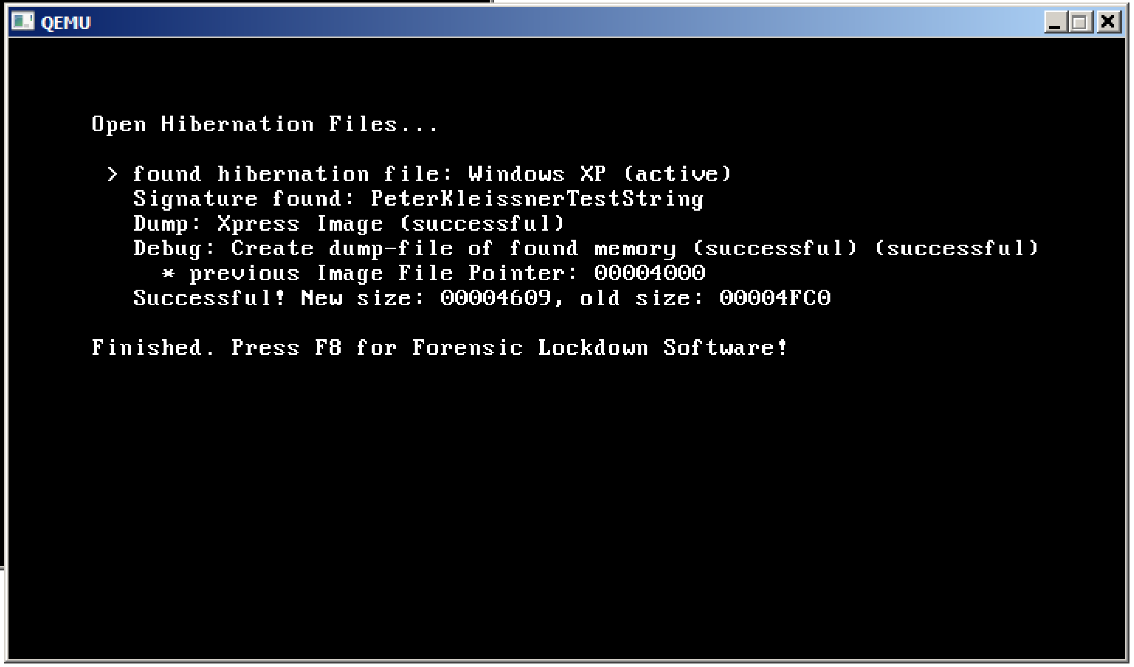


QEMU

```
Mount Boot Partitions...

 > found FAT Partition, mounting
 > found NTFS Partition, mounting

Press any key to abort Hibernation File
```



QEMU

```
Open Hibernation Files...

 > found hibernation file: Windows XP (active)
   Signature found: PeterKleissnerTestString
   Dump: Xpress Image (successful)
   Debug: Create dump-file of found memory (successful) (successful)
     * previous Image File Pointer: 00004000
   Successful! New size: 00004609, old size: 00004FC0

Finished. Press F8 for Forensic Lockdown Software!
```

## $CO_2$ Plugin

### Save The Environment!

- Example Plugin
- Throttling CPU speed down to 80%
- Normal user should not take any notice but our earth does :)
- Using the Advanced Configuration Programming Interface

## Boot Applications

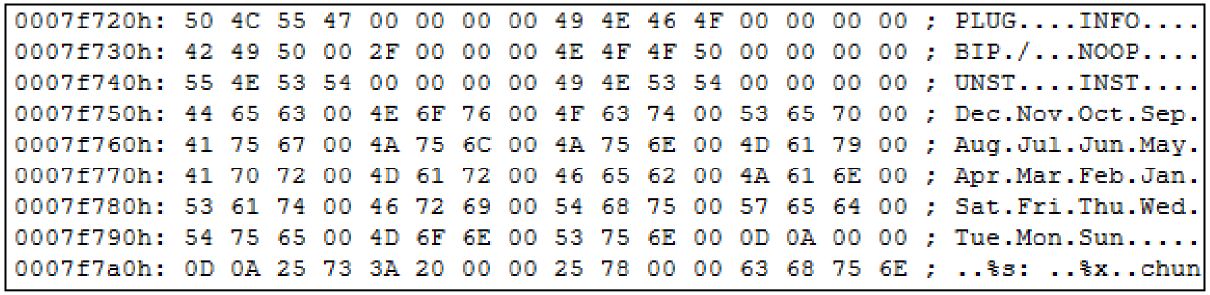
# Example: Sinowal Extractor

**Using Stoned Bootkit to execute Sinowal and then extract the unpacked kernel driver**

- **Tracing the memory by hooking the exports for `ExAllocatePool()` and `ExFreePool()` using the installed Stoned Subsystem**

  **→ Writing it out to disk for further analysis**

- **New Anti-Malware technology?**



```
0007f720h: 50 4C 55 47 00 00 00 00 49 4E 46 4F 00 00 00 00 ; PLUG....INFO....
0007f730h: 42 49 50 00 2F 00 00 00 4E 4F 4F 50 00 00 00 00 ; BIP./...NOOP....
0007f740h: 55 4E 53 54 00 00 00 00 49 4E 53 54 00 00 00 00 ; UNST....INST....
0007f750h: 44 65 63 00 4E 6F 76 00 4F 63 74 00 53 65 70 00 ; Dec.Nov.Oct.Sep.
0007f760h: 41 75 67 00 4A 75 6C 00 4A 75 6E 00 4D 61 79 00 ; Aug.Jul.Jun.May.
0007f770h: 41 70 72 00 4D 61 72 00 46 65 62 00 4A 61 6E 00 ; Apr.Mar.Feb.Jan.
0007f780h: 53 61 74 00 46 72 69 00 54 68 75 00 57 65 64 00 ; Sat.Fri.Thu.Wed.
0007f790h: 54 75 65 00 4D 6F 6E 00 53 75 6E 00 0D 0A 00 00 ; Tue.Mon.Sun.....
0007f7a0h: 0D 0A 25 73 3A 20 00 00 25 78 00 00 63 68 75 6E ; ..%s: ..%x..chun
```

**(Unpacked Sinowal kernel driver, here you see commands & domain name generation strings)**

## Bootkit Installation

## Installation Guide

1. **Backup original MBR**
2. **Overwrite Master Boot Record**
3. **Extract Files**

**Problem: Raw sector access is required**

| | |
|---|---|
| **Windows XP** | **Administrator rights** |
| **Windows Vista** | **Elevated Administrator rights** |

**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 the system for elevated rights at runtime using `ShellExecute()` or request it via a manifest

  If the user clicks “no” terrorize the user and ask again, e.g. start the elevated process until the user clicks “yes”

# Elevated Administrator Rights

## ▪ Method 1: Application Manifest

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

Application manifest (can be embedded into the application as resource)

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

Visual Studio linker option to generate and include such a manifest

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

## ▪ Method 2: ShellExecute() at runtime

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

## MBR is still writable

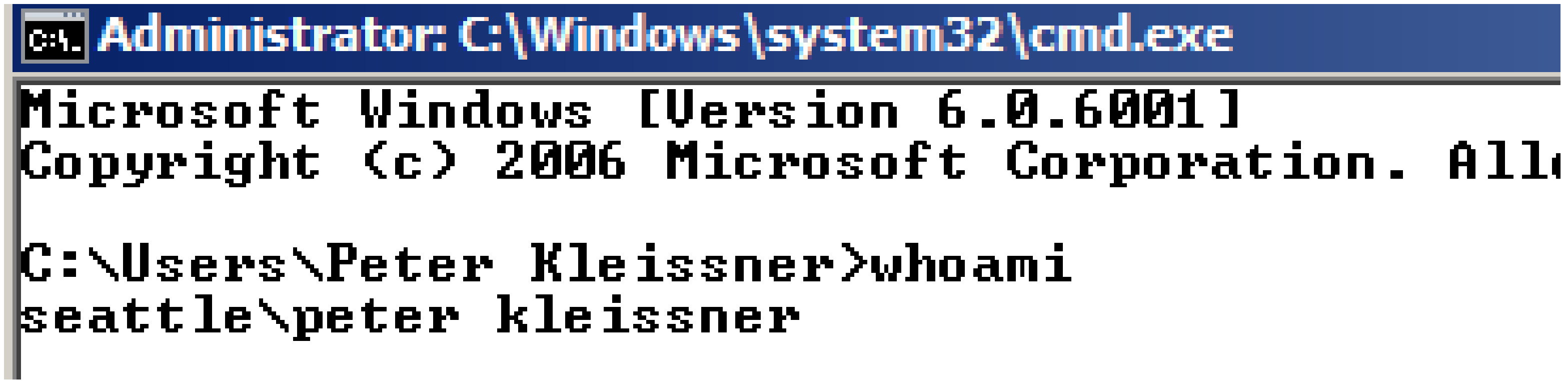
- CreateFile(“`\\.\PhysicalDrive0`”, …)
- Direct driver usage, IOCTLs
- Also works with Windows Vista and Server 2008:

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 the Microsoft Knowledgebase article #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)


Administrator: C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation. All

C:\Users\Peter Kleissner>whoami
seattle\peter kleissner
```

**Time for a live demonstration!**

## General Considerations

## 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 executables on disk, therefore no integrity check will fail

- **Data Execution Prevention**

## PoC Payload

- **Same as in Vbootkit (BHE 2007)**

  Thanks to my friends Vipin & Nitin Kumar!

- **Console Privilege Escalation**
  - Changing privilege of every `cmd.exe` process to the same as `services.exe`
  - Written as normal driver for Stoned

- **Displaying signature at startup**

```
Your PC is now Stoned!  ..again
```

# Developing with Stoned

1. **Download the framework**
2. **Write your own driver**
3. **Modify the infector, or just:**

**Use the installed Stoned Subsystem in Windows by your application**

```
syscall, int 2Eh with
function numbers = 3000h-3FFFh
```

**→ New open development platform**

## Secure the pre-boot Environment

**Advice to OS vendors and HW architects:**

**Take use of the Trusted Platform Module and full volume encryption.**

**Full volume encryption software should:**

1. **Secure its own software**
2. **Disable MBR overwrite in Windows**
3. **Make MBR genuine verifications**

**Do not try to fix software issues with security policies.**

## Future versions?

- Linux support (OS independency)
- 64-bit Windows support
- Defeating Trusted Platform Module
- Anti Windows Product Activation

## References

Black Hat Research Publications

**www.blackhat.com**

Sinowal / Mebroot

Vbootkit, Vipin & Nitin Kumar

**www.nvlabs.in**

## Presentation Materials

**Stoned project, papers & development framework**

**www.stoned-vienna.com**

Peter Kleissner at
Black Hat USA 2009

## Thanks...

Questions?
Comments?

...for your attention!

Peter Kleissner at
Black Hat USA 2009

**Thank You!**