

Win32dd :
Challenges of Windows physical
memory acquisition and
exploitation



Matthieu Suiche (msuiche)

<http://www.msuiche.net>

<http://win32dd.msuiche.net>

<http://www.twitter.com/msuiche>

msuiche@jgmail.com

Shakacon - June 2009

Netherlands Forensics Institute - www.forensischinstituut.nl

Who am I?

- Forensics Researcher for the Netherlands Forensics Institute (NFI).
- Microsoft Enterprise Security MVP 2009
- Speaker at various security events, such as PacSec, BlackHat USA, Europol High Tech Crime Meeting etc.
- Some projects that may interest you:
 - HibrShell (<http://www.msuiche.net/hibrshell/>)
 - Shell for Windows hibernation files manipulation using Microsoft Debugging symbols and based on SandMan Framework.
 - Win32dd (<http://win32dd.msuiche.net>)
- Blog about reverse engineering : www.msuiche.net

Talk Outline

- WWW?
- Memory dump files
- Win32dd & Acquisition
- Exploitation
- Future
- Conclusions

Talk Outline

- **WWW?**
- Memory dump files
- Win32dd & Acquisition
- Exploitation
- Future
- Conclusions

- Why?
 - For investigation purposes, incident response, malware/rootkit analysis, etc.
- Who?
 - Incident response engineers, investigators etc..
- When?
 - As soon as possible
- What?
 - A copy of the physical memory

Coffee

Acquisition

Coffee

Exploitation

Talk Outline

- WWW?
- **Memory dump files**
- Win32dd & Acquisition
- Exploitation
- Future
- Conclusions

Win32dd & Acquisition

MEMORY IMAGING

Windows

Crash
dump file
(BSOD)

Hibernation
File
(Hibernate)

External Tools

Win32DD

...

Raw dump
file.

Crash dump
file (without
BSOD)

Raw dump
file.

Memory dump files (1/5)

Raw dump file

- Pros: exact, dd-style copy of physical memory
- Cons: dump file does not contain processor state

Memory dump files (2/5)

Crash dump file

- Small memory dump
- Kernel memory dump
- **Complete memory dump**
 - Pros: This is a full memory snapshot of pages used by Windows Memory Manager. This includes both user-land and kernel-land. It means both malware and rootkit analysis can be done. Can be analyzed with WinDbg.
 - Cons: This option is not available on computers that are running a 32-bit operating system and that have 2 gigabytes (GB) or more of RAM according to Microsoft KB 274598.

Memory dump files (3/5)

Microsoft Crash Dump Header

```
typedef struct _DUMP_HEADER32 {
    /* 0x000 */ ULONG Signature;
    /* 0x004 */ ULONG ValidDump;
    /* 0x008 */ ULONG MajorVersion;
    /* 0x00C */ ULONG MinorVersion;
    /* 0x010 */ ULONG DirectoryTableBase;
    /* 0x014 */ ULONG PfnDataBase;
    /* 0x018 */ PLIST_ENTRY PsLoadedModuleList;
    /* 0x01C */ PLIST_ENTRY PsActiveProcessHead;
    /* 0x020 */ ULONG MachineImageType;
    /* 0x024 */ ULONG NumberOfProcessors;
    /* 0x028 */ ULONG BugCheckCode;
    /* 0x02C */ ULONG BugCheckParameter1;
    /* 0x030 */ ULONG BugCheckParameter2;
    /* 0x034 */ ULONG BugCheckParameter3;
    /* 0x038 */ ULONG BugCheckParameter4;
    /* 0x03C */ UCHAR VersionUser[32];
    /* 0x05C */ UCHAR PaeEnabled;

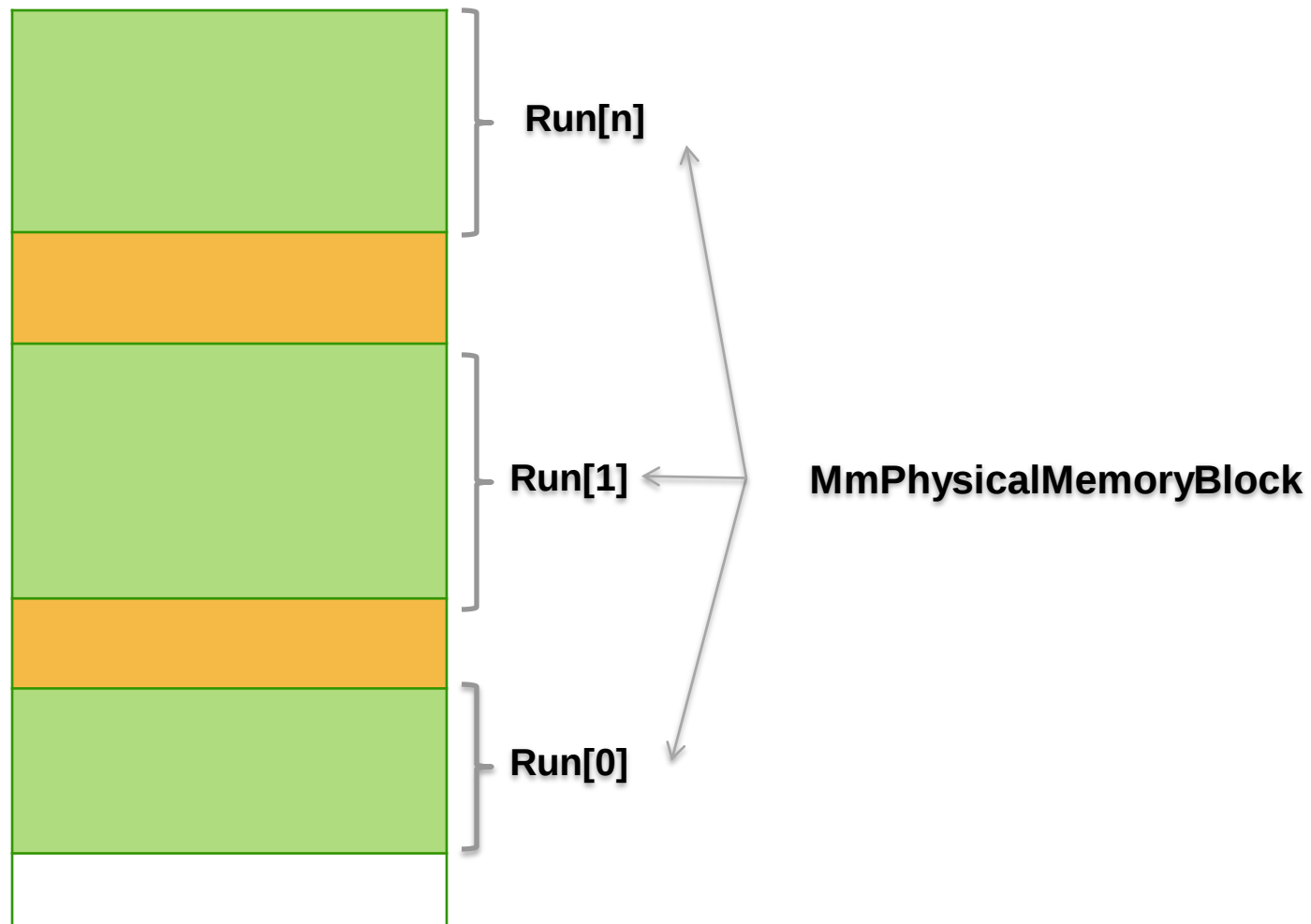
    /* 0x05d */ UCHAR KdSecondaryVersion;
    /* 0x05e */ UCHAR Spare3[2];
    /* 0x060 */ PKDDEBUGGER_DATA64 KdDebuggerDataBlock;

    union {
        /* 0x064 */ PHYSICAL_MEMORY_DESCRIPTOR32
        PhysicalMemoryBlock;
        /* 0x064 */ UCHAR PhysicalMemoryBlockBuffer[700];
    }

    union {
        /* 0x320 */ CONTEXT Context;
        /* 0x320 */ UCHAR ContextRecord[120];
    }

    /* 0x7d0 */ EXCEPTION_RECORD32 Exception;
    /* 0x820 */ UCHAR Comment[128];
    /* 0x8a0 */ UCHAR _reserved0[1768];
    /* 0xf88 */ ULONG DumpType;
    /* 0xf8c */ ULONG MiniDumpFields;
    /* 0xf90 */ ULONG SecondaryDataState;
    /* 0xf94 */ ULONG ProductType;
    /* 0xf98 */ ULONG SuiteMask;
    /* 0xf9c */ ULONG WriterStatus;
    /* 0xfa0 */ LARGE_INTEGER RequiredDumpSpace;
    /* 0xfa8 */ UCHAR _reserved2[16];
    /* 0xfb8 */ LARGE_INTEGER SystemUpTime;
    /* 0xfc0 */ LARGE_INTEGER SystemTime;
    /* 0xfc8 */ UCHAR _reserved3[56];
} DUMP_HEADER32, *PDUMP_HEADER32;
```

Memory dump files (4/5)



Physical memory ranges defined by Windows memory manager.

Memory dump files (5/5)

Microsoft Crash Dump Header

```
PHYSICAL_MEMORY_DESCRIPTOR32 PhysicalMemoryBlock;
```

```
typedef struct _PHYSICAL_MEMORY_DESCRIPTOR
{
    ULONG NumberOfRuns;
    ULONG NumberOfPages;
    PHYSICAL_MEMORY_RUN Run[1];
} PHYSICAL_MEMORY_DESCRIPTOR, *PPHYSICAL_MEMORY_DESCRIPTOR;
```

```
typedef struct _PHYSICAL_MEMORY_RUN
{
    ULONG BasePage;
    ULONG PageCount;
} PHYSICAL_MEMORY_RUN, *PPHYSICAL_MEMORY_RUN;
```

Talk Outline

- WWW?
- Memory dump files
- **Win32dd & Acquisition**
- Exploitation
- Future
- Conclusions

Win32dd & Acquisition

MEMORY IMAGING

Windows

Crash
dump file
(BSOD)

Hibernation
File
(Hibernate)

External Tools

Win32DD

...

Raw dump
file.

Crash dump
file (without
BSOD)

Raw dump
file.



DEMO #1: ACQUISITION

Win32dd & Acquisition

```

Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7100]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>win32dd.exe -d C:\snapshot.dmp

Win32dd - v1.2.2.20090608 - Kernel land physical memory acquisition
Copyright (c) 2007 - 2009, Matthieu Suiche <http://www.msuiche.net>

Name                               Value
-----
File type:                          Microsoft crash dump
Acquisition method:                 MmMapIoSpace()
Content:                             Original MmPhysicalMemoryBlock

Destination path:                    \??\C:\snapshot.dmp

O.S. Version:                        Microsoft Windows 7 (build 7100)
Computer name:                       HOMER-PC

Physical memory in use:               52%
Physical memory size:                 3140240 Kb < 3066 Mb>
Physical memory available:            1488476 Kb < 1453 Mb>

Paging file size:                     6278728 Kb < 6131 Mb>
Paging file available:                 3410792 Kb < 3330 Mb>

Virtual memory size:                  2097024 Kb < 2047 Mb>
Virtual memory available:              2071704 Kb < 2023 Mb>

Extended memory available:             0 Kb < 0 Mb>

Physical page size:                   4096 bytes
Minimum physical address:              0x1000
Maximum physical address:              0xBF065000

Address space size:                   3218497536 bytes <3143064 Kb>
Acquisition started at:                18/6/2009 <DD/MM/YYYY> 22:6:31 <UTC>

Processing....Done.

Acquisition finished at: 18/6/2009 <DD/MM/YYYY> 22:25:5 <UTC>
Time elapsed:                  18:33 minutes:seconds <1113 secs>

Created file size:                3215613952 bytes < 3066 Mb>

NtStatus <troubleshooting>:        0x00000103
Total of written pages:              785062
Total of inaccessible pages:          0
Total of accessible pages:            785062

SHA1: E77133432724B0324AE7284BE5926EE595118809

Physical memory in use:               53%
Physical memory size:                 3140240 Kb < 3066 Mb>
Physical memory available:            1465664 Kb < 1431 Mb>

Paging file size:                     6278728 Kb < 6131 Mb>
Paging file available:                 3380692 Kb < 3301 Mb>

Virtual memory size:                  2097024 Kb < 2047 Mb>
Virtual memory available:              2071704 Kb < 2023 Mb>

Extended memory available:             0 Kb < 0 Mb>

Minimum physical address:              0x1000
Maximum physical address:              0xBF065000

Address space size:                   3218497536 bytes <3143064 Kb>
C:\Windows\system32>

```

Win32dd and Acquisition

- Initial reason to create win32dd was because of user-land restriction access to `\\Device\\PhysicalMemory` since Windows 2003 SP1.
 - That's why win32dd is kernel-land based and requires Administrator privileges.
 - ... a tool like ShellRunas or Runas, to elevate privilege is also very useful to run win32dd.
- Goal of win32dd is to provide a small, useful swissknife for Windows memory analysis.
 - That's why multiple formats are supported.
 - That's why multiple technical options are provided.
 - SHA1 Hashing



Acquisition

RAW MEMORY DUMP

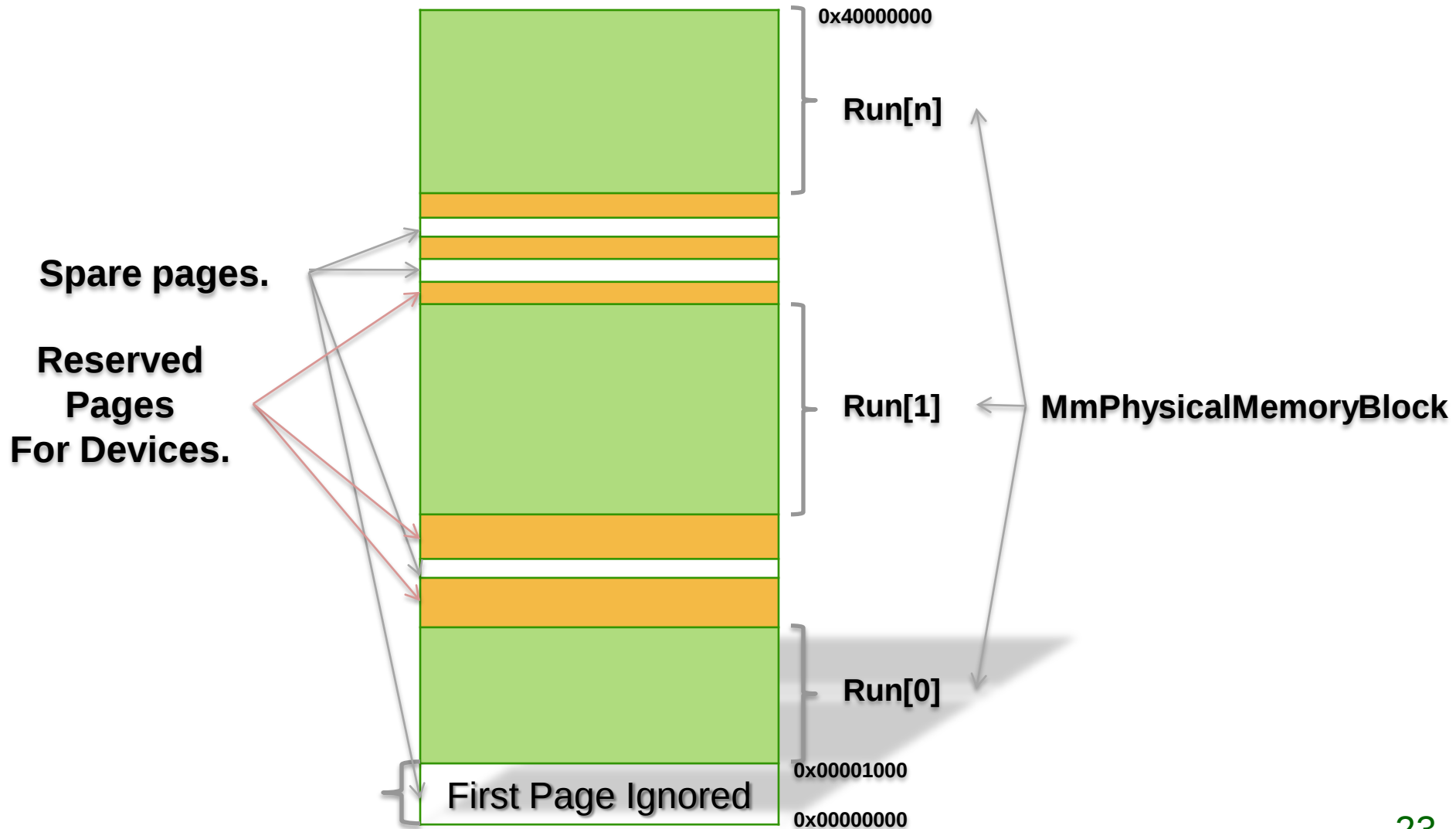
Win32dd & Acquisition

- Two ways provided to access to the physical memory through *-t argument*.
 - \\Device\\PhysicalMemory object
 - Level 0 - Used by default
 - MmMapIoSpace() mapping function.
 - Level 1 – *win32dd.exe -l 1 dest_file.bin*
- These levels only apply for the raw dumping.
- This dumps the whole processor view, not just the physical pages used by the Windows memory manager (Mm).

Win32dd & Acquisition

- It means output file size will be bigger than physical memory size.
- If physical memory size is 0x40000000 – it doesn't mean highest physical page would be 0x4000.
- Then, we have to determine the size of the physical address space using this formula :
 - $(\text{HighestPhysicalPage} + 1) * \text{PAGE_SIZE}$
- Instead of using physical memory size.

Physical Memory Layout





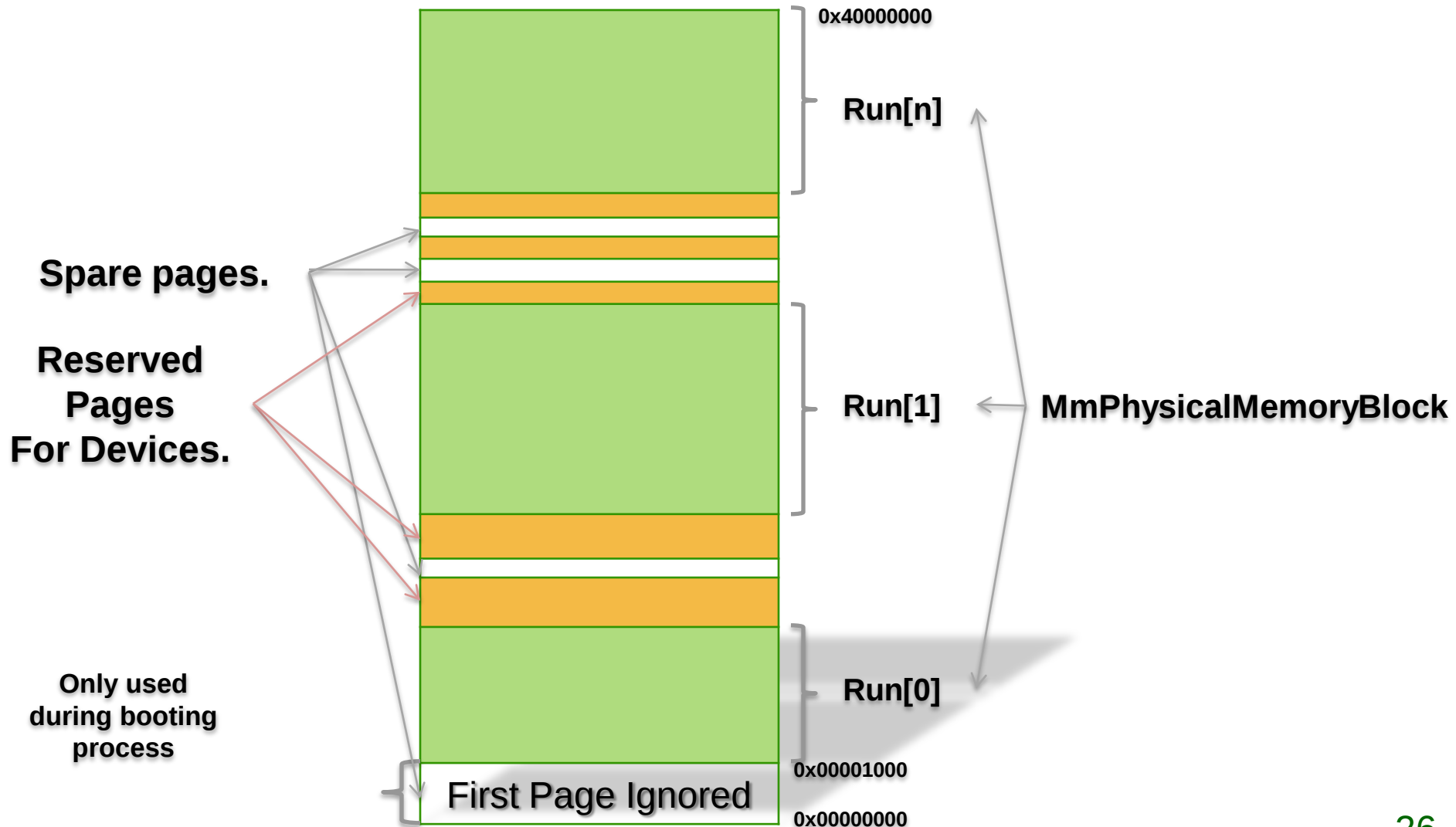
Acquisition

MICROSOFT CRASH MEMORY DUMP

Win32dd & Acquisition

- MmPhysicalMemoryBlock
 - Valid physical memory ranges detected
- Advantages
 - Reports only physical memory used by Windows Memory Manager, and ignores physical address space dedicated to hardware.
- Limitations
 - Skips interesting pages like the first physical page where pre-boot authentication password is present in plaintext. (< Windows Vista SP1)

Physical Memory Layout



Win32dd & Acquisition

Win32dd –d option advantages

- Win32dd provide a further option to force to save first physical page.
 - Type 1 – *win32dd -d -t 1 dest_file.dmp*
- Doesn't use Microsoft APIs to generate the Microsoft Crash dump – It means method used by rootkits, like Rustock.C, by registering a callback routine through KeRegisterBugCheckCallback to let the drive cleaning its memory when KeBugCheck happens doesn't affect win32dd.

Win32dd & Acquisition

- User should use the `-d` option to generate a Microsoft crash dump with Win32dd.
- Two methods for the `-d` option

Talk Outline

- WWW?
- Memory dump files
- Win32dd & Acquisition
- **Exploitation**
- Future
- Conclusions

Exploitation

- **#1** Extracting information is the most important part of an investigation.

Exploitation

- **#1** Extracting information is the most important part of an investigation.
 - Most efficient way would be to use existing tool maintained by a big vendor who well knows the Operating System.

- **#1** Extracting information is the most important part of an investigation.
 - Most efficient way would be to use existing tool maintained by a big vendor who well knows the Operating System.

Microsoft



- Microsoft Tools :
 - WinDbg
 - Debugging Symbols
 - PowerDbg (PowerShell library)
- Moreover, WinDbg also has a HUGE community then it's easier to find resources (scripts, add-ons, books, articles, ...)

Exploitation

- **WinDbg** is a multipurpose graphical debugger for Microsoft Windows, distributed on the web by Microsoft. It can be used to debug user mode applications, drivers, and the operating system itself in kernel mode. **WinDbg** also has the ability to automatically load debugging symbol files (e.g., PDB files) from Microsoft servers.
- **Debugging Symbols files** contains information about internal structures (e.g. EPROCESS), unexported symbols and so on.
- **PowerDbg** is a **PowerShell** library that enables you to easily create **PowerShell** scripts that automate the debugging session, interacting with **WinDbg**. You can use **PowerDbg** for Kernel Mode or User Mode, Post-Mortem debugging or Live Debugging and for native or managed code.

Exploitation

```

Command - Dump C:\Documents and Settings\Matthieu\Desktop\win32dd-v1.2.1.20090106\win32dd.1.2.1.20090106\
Microsoft (R) Windows Debugger Version 6.9.0003.113 X86
Copyright (c) Microsoft Corporation. All rights reserved.

Loading Dump File [C:\Documents and Settings\Matthieu\Desktop\win32dd-v1.2.1.20090106\win32dd.1.2.1.2009
Kernel Complete Dump File: Full address space is available

Symbol search path is: SRV*c:\symbols*http://msdl.microsoft.com/download/symbols
Executable search path is:
Windows XP Kernel Version 2600 (Service Pack 3) MP (2 procs) Free x86 compatible
Product: WinNt, suite: TerminalServer SingleUserTS
Built by: 2600.xpsp_sp3_gdr.090206-1234
Kernel base = 0x804d7000 PsLoadedModuleList = 0x8055d720
Debug session time: Wed Jun  3 09:45:52.727 2009 (GMT+2)
System Uptime: 28 days 0:24:54.571
WARNING: Process directory table base 0B0E07C0 doesn't match CR3 00EFC000
WARNING: Process directory table base 0B0E07C0 doesn't match CR3 00EFC000
Loading Kernel Symbols
.....
Loading User Symbols
.....
Loading unloaded module list
.....
0: kd> lm
start      end          module name
00400000 00413000  win32dd_400000 (deferred)
76390000 763ad000  IMM32         (deferred)
77dd0000 77e6b000  ADVAPI32      (deferred)
77e70000 77f02000  RPCRT4        (deferred)
77f10000 77f59000  GDI32         (deferred)
77fe0000 77ff1000  Secur32       (deferred)
7c800000 7c8f6000  kernel32      (deferred)
7c900000 7c9b2000  ntdll         (pdb symbols)  c:\symbols\ntdll.pdb\6992F4DAF4B144068D78669D6CB5D
7e410000 7e4a1000  USER32       (deferred)
804d7000 806e4000  nt            (pdb symbols)  c:\symbols\ntkrpamp.pdb\909FE6B806E4444B9230BAAF21
806e4000 80704d00  hal           (deferred)
8aa25000 8aa2e000  win32dd       (deferred)
8abe6000 8ac10180  kmixer        (deferred)
8f2f6000 8f319180  Fastfat       (deferred)
a07fb000 a07fe800  asyncmac      (deferred)
a0ce1000 a0cf5480  wdmaud        (deferred)
a0e0e000 a0e4ea80  HTTP          (deferred)
a0f1f000 a0f2dd80  svsaudio      (deferred)

0: kd>

```



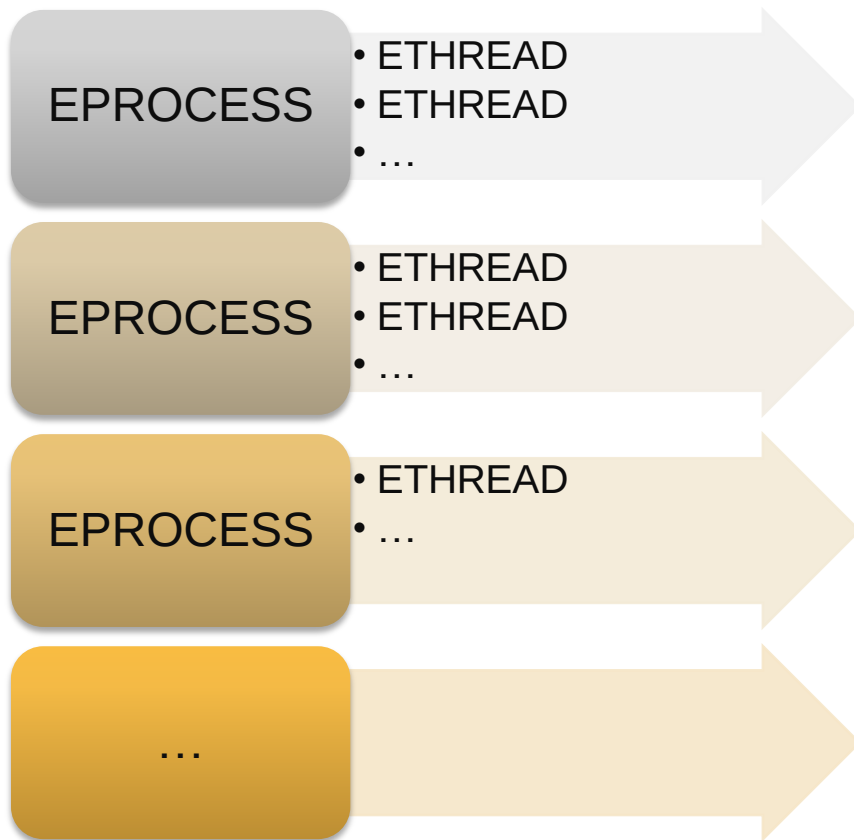
DEMO #1: FUto detection using all these technologies

Futo Mechanism

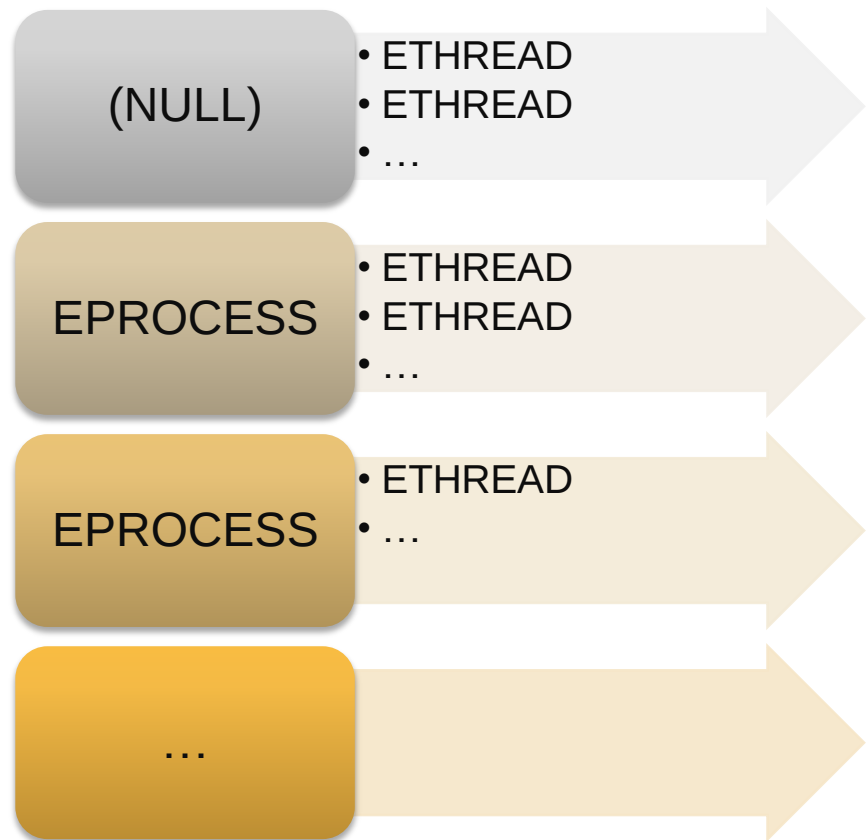
- Futo is Rootkit designed by Peter Silberman and CHAOS.
- Futo can hide processes using the following scheme
 1. Localize `PspCidTable`
 2. Erase entry pointing to `EPROCESS` which belong to the process
- ... but this Table also contains pointers to `ETHREAD`

Futo Mechanism

Original PspCidTable



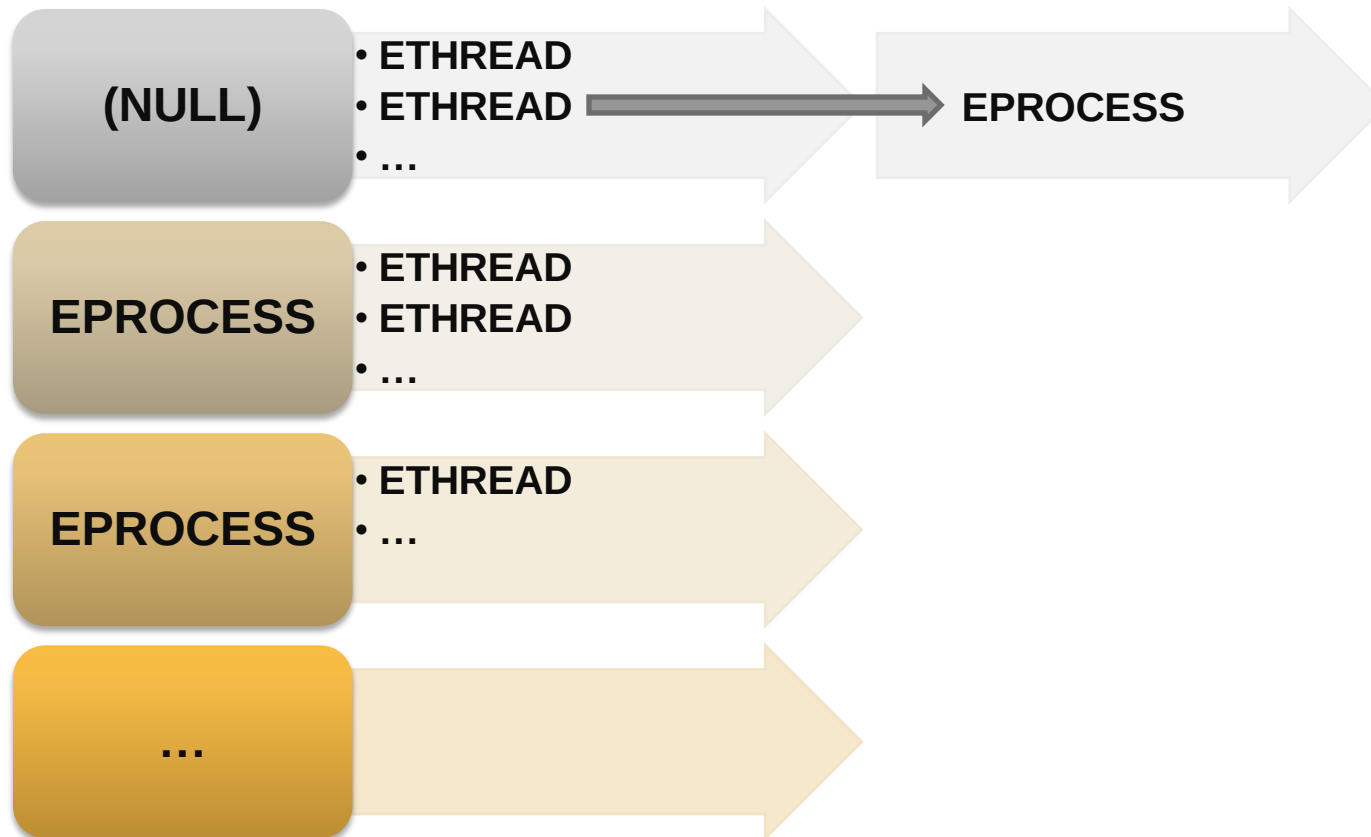
Modified PspCidTable



Futo Detection

- Guasconi Vincent designed a method to retrieve hidden processes.
 1. Scan every `PspCidTable` entries
 2. Check if the entry is a `ETHREAD` pointer using `ServiceTable` field
 3. Save `ThreadsProcess` field which is a pointer to its `EPROCESS` root.
 4. Then compare active `EPROCESS` pointers list with retrieved `EPROCESS` pointers list from `ETHREAD` entries.

Futo Detection



- PowerShell script
 - Because Python is so 2003, and PowerShell so 2010 ☺
 - Out-GridView command is really nice
- Uses WinDbg + PowerDbg + PowerShell
- Generic because it uses symbols and WinDbg commands
 - No need to reinvent the wheel

Futo.ps1

Filter

+ Add criteria

EPROCESS	Process Name	Status
0x81d76da0	"alg.exe"	Not Hidden
0x81d79020	"cmd.exe"	Hidden
0x81e5ada0	"cmd.exe"	Not Hidden
0x821ca268	"csrss.exe"	Not Hidden
0x8221dda0	"explorer.exe"	Not Hidden
0x81eb91a0	"lsass.exe"	Not Hidden
0x81e4e678	"services.exe"	Not Hidden
0x822d7020	"smss.exe"	Not Hidden
0x81e507f0	"spoolsv.exe"	Not Hidden
0x81ea2cf8	"svchost.exe"	Not Hidden
0x81f923c0	"svchost.exe"	Not Hidden
0x821cdda0	"svchost.exe"	Not Hidden
0x822b25a8	"svchost.exe"	Not Hidden
0x82339b28	"svchost.exe"	Not Hidden
0x823c8830	"System"	Not Hidden
0x821c52e8	"vmacthlp.exe"	Not Hidden
0x821bc8c8	"VMwareService.e"	Not Hidden
0x81f949b0	"VMwareTray.exe"	Not Hidden
0x81e73da0	"VMwareUser.exe"	Not Hidden
0x81e8c6e8	"win32dd.exe"	Not Hidden
0x81e45020	"winlogon.exe"	Not Hidden
0x82063020	"wscntfy.exe"	Not Hidden
0x821f9da0	"wuauclt.exe"	Not Hidden

Talk Outline

- WWW?
- Memory dump files
- Win32dd & Acquisition
- Exploitation
- **Future**
- Conclusions

Future

- 64-bits version
- Improved anti-anti-forensics features
 - Less dependency with native functions
 - Improvement of memory ranges manipulation
 - Etc.
- Speed optimization
- Improvement of the WinDbg <-> PowerShell interface

Talk Outline

- WWW?
- Memory dump files
- Win32dd & Acquisition
- Exploitation
- Future
- **Conclusions**

Conclusions

- **Win32dd** is a free tool for Windows physical memory snap shooting.
- **Win32dd** supports a wide range of Windows versions from Windows 2000 to Windows 7.
- **Win32dd** can produce handful snapshot like Microsoft crash dump file.
- **WinDbg** is great to analyze Microsoft crash dump files.

Ressources

- *Win32dd* (M. Suiche)
 - <http://win32dd.msuiche.net>
- *Microsoft Crash Dump Analysis weaknesses* (M. Suiche)
 - <http://www.msuiche.net/2008/10/16/microsoft-crash-dump-analysis-weaknesses/>
- *Pre-boot authentication password* (J. Brossard)
 - http://www.ivizsecurity.com/research/preboot/preboot_whitepaper.pdf
- *Rustock.C When a myth comes true* (F. Boldewin)
 - <http://www.reconstructor.org/papers/Rustock.C%20-%20When%20a%20myth%20comes%20true.pdf>
- *PowerDbg - Automated Debugging using WinDbg and PowerShell* (R. A. Farah)
 - <http://www.codeplex.com/powerdbg>
- *Win32dd – Memory Imaging, Debugged! MZ/PE: MagaZine for/from Practicing Engineers #1* (D. Vostokov, M. Suiche, R. A. Farah)
 - (ISBN-13: 978-1906717384)
- *Pushing Limits of Windows: Physical Memory* (M. Russinovich)
 - <http://blogs.technet.com/markrussinovich/archive/2008/07/21/3092070.aspx>
- *MemInfo: Peer Inside Memory Manager Behavior on Windows Vista and Server 2008*
 - <http://www.alex-ionescu.com/?p=51>
- *ShellRunas* (M. Russinovich & J. Schwartz)
 - <http://technet.microsoft.com/en-us/sysinternals/cc300361.aspx>
- *Détection de processus cachés* (G. Vincent)
 - <https://www.lasecuriteoffensive.fr/drup/d%C3%A9tection-de-processus-cach%C3%A9s>