SANDMAN PROJECT

Matthieu Suiche < matt@msuiche.net >

http://sandman.msuiche.net/

February, 2008

Abstract. For *Windows 2000*, Microsoft provides a feature called Hibernation also know as suspend to disk that aims to save the system state into an undocumented file called hiberfil.sys. This file contains all the physical memory saved by the Operating System and aims to be restored by the user the next time the computer is powered on. Live forensics analysis is used to use physical memory dump to recover information on the targeted machine. One of the main problems is to obtain a readable physical memory dump, hibernation is an efficient way to save and load physical memory. Hibernation analysis has notable advantages. System activity is totally frozen, therefore coherent data is acquired and no software tool is able to block the analysis. The system is left perfectly functional after analysis, with no side effects.

Introduction. The hibernation file opens two valuable doors:

The first one is forensics analysis for defensive computing. Hibernation is an efficient and easy way to get a physical memory dump. But the main issue about it was: How to read the hiberfil.sys? That's how the idea of *SandMan* born.

The second one is a new concept we will be introduced and called *"offensics"* which is a portmanteau from *"offensive"* and *"forensics".* If we can read hiberfil.sys, can we rewrite it? The answer is: Yes, with SandMan you can.

Headlines of hibernation process.

Windows kernel **(ntoskrnl.exe)** executable creates the hibernation file and write the physical memory dump inside it, when suspend to disk action is requested.

OSLoader **(osloader.exe)** executable read the hibernation file and load the uncompressed data to the physical memory.

Windows hibernation file structure.

The hibernation file globally looks like this:

- The first page contains, the *hibernation image header* another structure called *HIBER_PERF* is introduced in Windows XP and *BootLoaderLog* fields are introduced in Windows Vista.

  There is a field called *ImageType* inside the *hibernation image header* which is deleted from Windows Vista. And the 4 bytes signature at the very beginning of the hibernation file becomes “HIBR” instead of “hibr”.

- The second page contains a *ULONG* array which looks to be a table of free memory pages.

- The third page contains the *KPROCESSOR_STATE* which is saved by the Windows kernel function KiSaveProcessorControlState. This structure contains state registers saved by the **ntoskrnl.exe** like *CrX* registers, *Global descriptor table* (GDT), *Interrupt descriptor table* (IDT) and further registers.

- *From Windows Vista (and 2008),* the two previous pages position are switched.

- Next information are the compressed data, saved by the Windows Kernel while the hibernation process. It uses a kind of linked list schema which is redundant.

  MEMORY_RANGE_ARRAY contains two structures:

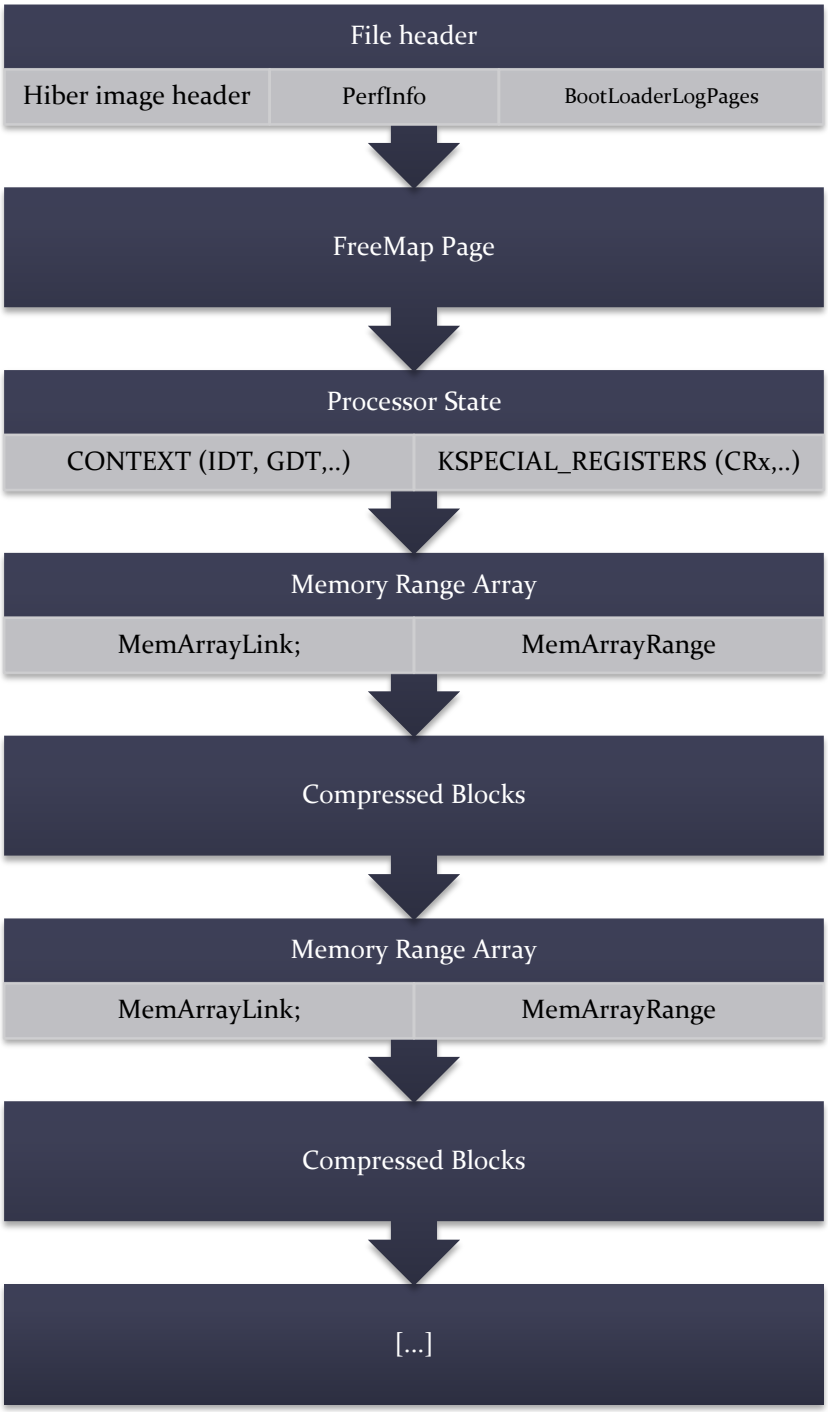
  - *MEMORY_RANGE_ARRAY_LINK*
  - *MEMORY_RANGE_ARRAY_RANGE*

  MEMORY_RANGE_ARRAY_RANGE structure can only contained 255 entries, that why *MEMORY_RANGE_ARRAY_LINK* contains a field that points to the next MEMORY_RANGE_ARRAY structure.

  From Windows XP, the compression block can be identified by the following signature *\x81\x81xpress.*
  Each compressed Xpress block contains 0x10 (16) uncompressed pages.

MEMORY_RANGE_ARRAY_RANGE contains two fields that delimit pages to copy. These two fields are used as index sequence to copy at the right position uncompressed pages from the Xpress blocks.

The following flowchart *(Figure - 1)* represents the hibernation file.



(Figure – 1)

SandMan.

SandMan is a framework that provides a *C Library* and a *Python portage.* This framework aims to make the hibernation file, readable and writable. It also provides internals functions which use hibernation file potential. They were developed especially for analysis like *HiberGetVersion()* which returns the exact target version and build.

For instance, here is a sample of Python scripting which returns the target version before generating a full memory dump from the hibernation file.

```
#!/usr/bin/python
#
#
#Module Name:
#
#    sample1.py
#
#Abstract:
#
#    - Display target version.
#    - Build a memory dump from a hibernation file.
#
#Environment:
#
#    - Python
#
#Revision History:
#
#    - Matthieu Suiche
#

import sandman

# Open hibernation file.
s = sandman.hiber_open("hiberfil.sys")

# Get the target version.
ver = sandman.hiber_get_version(s);

print "Windows version %d.%d.%d\n" % (ver & 0xFF, (ver &
0xFF00) >> 8, ver >> 16)


print "Generate physical memory dump...\n"

# Generate a memory dump from hiber file.
sandman.hiber_dump(s, "hibernate.dmp")

print "Done.\n"

# Close hibernation file.
sandman.hiber_close(s)
```

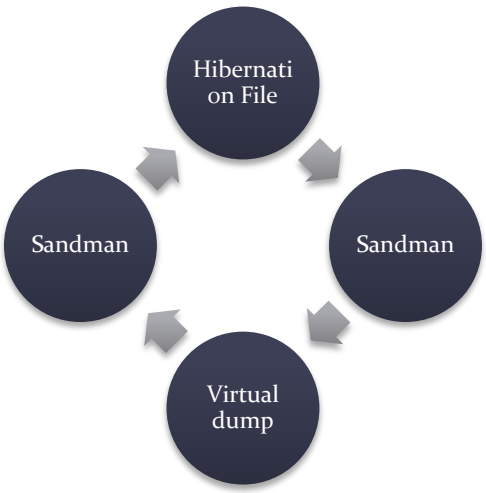
Here, is another sample doing the same thing but using the C Library.

```
/*++

Module Name:

    sample1.c

Abstract:

    - Display some header information.
    - Build a physical memory dump from a hibernation file.
    - Requires: SandMan library.

Environment:

    - User mode

Revision History:

    - Matthieu Suiche , Nicolas Ruff

--*/

#include <windows.h>
#include "sandman.h"

/*++

    Function Name: main

--*/

int main (int argc, TCHAR *argv[])
{
PSANDMAN_OBJECT s;

SYSTEMTIME st;
FILETIME ft;

ULONG ulStatus;

ULONG ulVersion = 0;
ULONG ulMajorVersion = 0;
ULONG ulMinorVersion = 0;
ULONG ulBuild = 0;

    printf("    Sandman demo! (c) %s %s.\n"
           "    %s\n"
           "    Demo: sample1.c\n\n",
            SANDMAN_DATE,
            SANDMAN_AUTHOR,
            SANDMAN_WEB);


    //
    // Open and parse hiberfil.sys
    //
    s = HiberOpen( L"C:\\hiberfiles\\Windows XP SP3 RTM
EN\\hiberfil.sys" );


    if (s == NULL) {
        printf("Error: Failed to open file.\n");
        return FALSE;
    }



    //
    // Display header info
    //
    printf("Signature:      %c%c%c%c\n",
        s->FileHdr->Signature[0],
```

```
        s->FileHdr->Signature[1],
        s->FileHdr->Signature[2],
        s->FileHdr->Signature[3] );



    //
    // Get and format SystemTime
    //
    ft.dwHighDateTime = s->FileHdr->SystemTime.HighPart;
    ft.dwLowDateTime  = s->FileHdr->SystemTime.LowPart;
    FileTimeToSystemTime( &ft, &st );




    printf("SystemTime:     %08x%08x [%d/%d/%d (DD/MM/YYYY)
%d:%d:%d (UTC)]\n",
        ft.dwHighDateTime,
        ft.dwLowDateTime,
        st.wDay,
        st.wMonth,
        st.wYear,
        st.wHour,
        st.wMinute,
        st.wSecond );

    //
    // Display some information from ProcessorState
    //

    printf("\nControl registers flags\n");

    printf("CR0: %08x\n",
s->ProcState->SpecialRegisters.u_cr0.Cr0);

    printf("CR0[PAGING]: %d\n",
s->ProcState->SpecialRegisters.u_cr0.Paging);

    printf("CR3: %08x\n",
s->ProcState->SpecialRegisters.Cr3);

    printf("CR4: %08x\n",
s->ProcState->SpecialRegisters.u_cr4.Cr4);

    printf("CR4[PSE]: %d\n",
s->ProcState->SpecialRegisters.u_cr4.PageSizeExtensions);

    printf("CR4[PAE]: %d\n",
s->ProcState-
>SpecialRegisters.u_cr4.PhysicalAddressExtension);


    ulVersion =  HiberGetVersion(s);

    //
    // Get the Windows version.
    //
    ulMajorVersion = (DWORD)(LOBYTE(LOWORD(ulVersion)));
    ulMinorVersion = (DWORD)(HIBYTE(LOWORD(ulVersion)));

    //
    // Get the build number.
    //
    if (ulVersion < 0x80000000)
        ulBuild = (DWORD)(HIWORD(ulVersion));


    printf("\nYou want more?\n");
    printf("Windows Version is %d.%d (%d)\n",
                ulMajorVersion,
                ulMinorVersion,
                ulBuild);
```

```
    printf("\nPhysical Memory dump.\n");


    //
    // Generate the memory dump.
    //
    ulStatus = HiberBuildPhysicalMemoryDump(s,
        L"C:\\hiberfiles\\Windows XP SP3 RTM
EN\\physmem.dmp");


    if (ulStatus == FALSE) return ulStatus;


    printf("Memory dump successfuly dumped.\n");




    //
    // Close the hibernation file.
    //
    HiberClose(s);

    return TRUE;
}
```

The following flowchart is a brief explanation of how file are used by *SandMan.*



(Figure - 2)

Future.

Actually, the public version only provides an access to hibernation file from Windows XP to Windows 2008 on x86 architecture. SandMan will continue to grow with his community and his actual developers to provide further practical uses and a generic hibernation file editor.

References.

SandMan Project, *Matthieu Suiche & Nicolas Ruff.*
http://sandman.msuiche.net

Enter SandMan, PacSec 2007, *Matthieu Suiche & Nicolas Ruff.*
http://www.msuiche.net/pres/PacSec07-slides-0.4.pdf *(English)*

Enter SandMan, PacSec 2007, *Matthieu Suiche & Nicolas Ruff.*
http://www.msuiche.net/pres/psj07ruffsuiche-jp.pdf (Japanese)