The Most Important Advancement...
Memory Analysis has made its way into well-known digital forensic tools!
Takahiro Haruyama has ported Volatility to EnCase

The Most Important Advancement...
Takahiro Haruyama, Blog post "PsEntropyPEB Matching Mode", August 5, 2010
http://cci.cocolog-nifty.com/blog/files/MemoryForensicToolkit_Ver1.83.zip
Apple Mac OS X
Retrieval of symbols from binaries

Translation between virtual and physical addresses:

- assume KPA = KVA
- retrieve IdlePDPT, IdlePDPT64, IdlePML4, and IdlePTD at known address to reconstruct kernel address space
Apple Mac OS X
Refresher: VA to PA translation

This page mapping example is for 4-KByte pages and 40-bit physical address size.

*Physical Address
Apple Mac OS X
In-depth analysis

- Machine information
  - CPUs, memory, kernel version

- Kernel modules
  - unexpected/suspicious extensions?

- System calls
  - hooks, redirection

- BSD Processes
  - parent-child relation, ownership, timestamp

- Mounted file systems
  - type, mount point, media
Matthieu Suiche / NFI. Advanced Mac OS X Physical Memory Analysis. Black Hat Briefings DC, 2010


Linux kmem_cache
Cache facility to support system memory allocator

Provides quick access to data structures of the same size

Example:

```c
task_struct_cachep = kmem_cache_create("task_struct",
    sizeof(struct task_struct),
    ARCH_MIN_TASKALIGN, SLAB_PANIC, NULL);
```
Uniform access to in-memory data of terminated objects

- Processes
- File handles (meta-data)
- Memory mappings (shared memory, file contents)
- Filesystem inode cache
- Sockets (meta-data)
- Socket buffers (contents)
- Netfilter NAT table


http://www.dfrws.org/2010/proceedings/richard2.pdf (Presentation)
Robust Signatures
Robust Signatures
List walking

PsActive
ProcessHead

smrss
flink
blink

rk
flink
blink

explorer
flink
blink
Robust Signatures
Power and weakness of scanners

- Describe an object by a set of constraints
  - constants
  - range/set of values
  - complex conditions

- Scan whole memory image
  - finds active and "visible" objects
  - finds active and hidden objects
  - finds terminated objects

- Is the signature based on essential data?

kd> dt 812927c0 nt!_DISPATCHER_HEADER
    +0x000 Type : 0x3  // "process"
    +0x001 Absolute : 0
    +0x002 Size : 0x1b
    +0x003 Inserted : 0
Dolan-Gavitt (2009): Fuzzing of EPROCESS
- 221 member fields in structure nt\_EPROCESS
- 32 were never accessed (bad candidates)
- 72 were accessed for every examined application (good candidates)
- fuzzing: manipulating field, waiting for process to crash
- the more crashes there are, the harder is it to temper with the field

Image: Dolan-Gavitt et al. (2009)
The List of Lists
The List of Lists

Kernel Processor Control Region

nt!__KPCR

+0x000  NtTib : __NT_TIB
+0x01c  SelfPcr : Ptr32 __KPCR
+0x020  Prcb : Ptr32 __KPRCB
+0x024  Irql : UChar
+0x028  IRR : Uint4B
+0x02c  IrrActive : Uint4B
+0x030  IDR : Uint4B
+0x034  KdVersionBlock : Ptr32 Void
+0x038  IDT : Ptr32 __KIDTENTRY
+0x03c  GDT : Ptr32 __KGDTENTRY
+0x040  TSS : Ptr32 __KTSS
+0x044  MajorVersion : Uint2B
+0x046  MinorVersion : Uint2B
+0x048  SetMember : Uint4B
+0x04c  StallScaleFactor : Uint4B
+0x050  DebugActive : UChar
+0x051  Number : UChar
+0x052  Spare0 : UChar
+0x053  SecondLevelCacheAssociativity : UChar
+0x054  VdmAlert : Uint4B
+0x058  KernelReserved : [14] Uint4B
+0x090  SecondLevelCacheSize : Uint4B
+0x094  HalReserved : [16] Uint4B
+0x0d4  InterruptMode : Uint4B
+0x0d8  Spare1 : UChar
+0x0dc  KernelReserved2 : [17] Uint4B
+0x120  PrcbData : __KPRCB
The List of Lists

Kernel Processor Control Block

nt!_KPRCB
  +0x000 MinorVersion : Uint2B
  +0x002 MajorVersion : Uint2B
  +0x004 CurrentThread : Ptr32 _KTHREAD
  +0x008 NextThread : Ptr32 _KTHREAD
  +0x00c IdleThread : Ptr32 _KTHREAD
  +0x010 Number : Char
  +0x011 Reserved : Char
  +0x012 BuildType : Uint2B
  +0x014 SetMember : Uint4B
  +0x018 CpuType : Char
  +0x019 CpuID : Char
  +0x01a CpuStep : Uint2B
  +0x01c ProcessorState : _KPROCESSOR_STATE

... 
+0x8c0 CallDpc : _KDPC
+0x8e0 ChainedInterruptList : Ptr32 Void
+0x8e4 LookasideIrpFloat : Int4B
+0x8e8 SpareFields0 : [6] Uint4B
+0x900 VendorString : [13] UChar
+0x90d InitialApicId : UChar
+0x90e LogicalProcessorsPerPhysicalProcessor : UChar
+0x910 MHz : Uint4B
+0x914 FeatureBits : Uint4B
+0x918 UpdateSignature : _LARGE_INTEGER
+0x920 NpxSaveArea : _FX_SAVE_AREA
+0xb30 PowerState : _PROCESSOR_POWER_STATE
Windows XP (and earlier versions):
KPCR is at a fixed kernel virtual address (0xfff0f000)

Vista (and later versions):
Address space layout randomization (ASLR)

Damien Aumaitre (2009): KPCR is self-refencing

kd> dt fff00000 _KPCR
nt!_KPCR
    +0x000 NtTib : _NT_TIB
    +0x01c SelfPcr : 0xfff0f000 _KPCR
    +0x020 Prcb : 0xfff0f120 _KPRCB
...

The List of Lists
How to locate the KPCR?
Zhang et al. (2010) proposed a KPCR signature

- \[(x + 0x1c) == x\]
- \[(x + 0x20) == x + 0x120\]

proposed combination of scanning for KPCR, followed by walking of thread/process lists

Schatz (2010) wrote a plug-in for Volatility to scan for KPCR structures and adopted other plug-ins to use values from the KPCR.
References


- Bradley Schatz, Blog post "Finding Object Roots in Vista (KPCR)", July 7, 2010
Questions and Answers
Thank You for Your Attention!

Andreas Schuster

a.schuster@yendor.net

http://computer.forensikblog.de/