Post-Mortem RAM Forensics

(or Reversing Windows RAM after-the-fact)

CanSecWest 2007

Tim Vidas

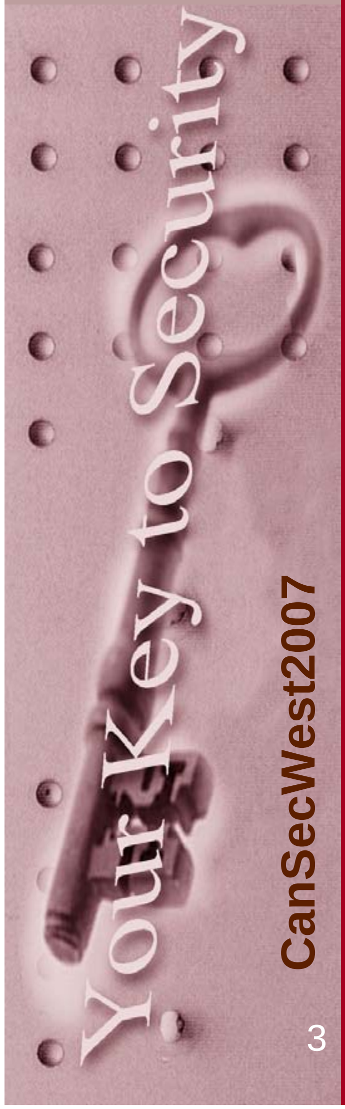
Who am I?

- Tim Vidas
  - Research was preformed under employment by the Nebraska University Consortium on Information Assurance (NUCIA) @ the University of Nebraska at Omaha (UNO)
  - Sr. Tech. Research Fellow
  - BS and MS in CS
    - -PhD in the works
  - Certs: CISSP, 40xx, Sec+, Guidance, etc.
  - Instructor: University, Guidance, LM RRCF

NUCIA

(obligatory sales pitch slide)

- Nebraska University Consortium on Information Assurance
- Sits in both CS and MIS programs
- IA full time
- NSA Center of Academic Excellence
- Traditional university coursework in IA, Crypto, Forensics, Secure Administration, Certification and Accreditation, etc
- STEAL Labs
- “Other work”



Talk Assumptions

- Only talking about x86 architecture
- Only talking about MS Windows (nt based)
- Only talking about ‘normal’ setups (no ‘weird’ boot switches or builds)

Evidence Volatility

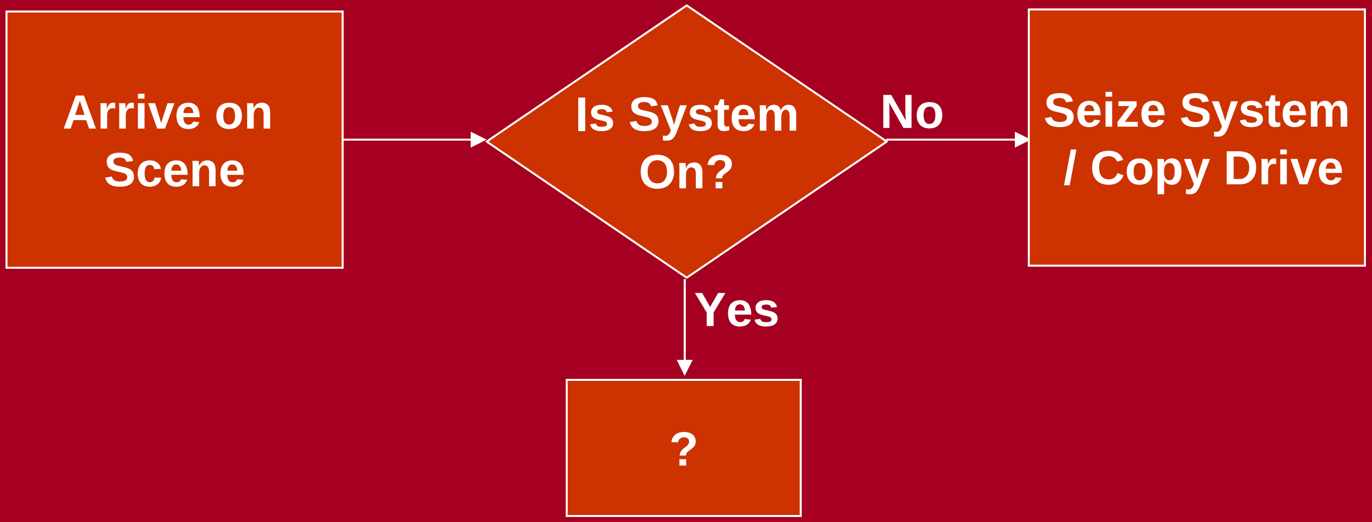
- Registers (more volatile)
- Caches
- Memory, process table, routing table, arp cache, etc
- Temp file systems
- File system / Disk Block
- Archival Media (less volatile)

Check out RFC 3227:

"Guidance for Evidence Collection and Archiving

IR: Current Process

- Currently there are two main states a system could be in at IR time.
- “Dead” System
  - Duplicate drives (non-volatile stores)
- “Live” System
  - ?



Current Process

- Live System
  - Pull the plug
    - Better than a 'shutdown'
  - Gather state information
    - More common in incident response
    - Interact with the machine
      - Observing the state changes it

Then, of course, the simple act of observing the outcome changes it, so the Heisenberg Uncertainty Principle comes into play here as well. You can't observe the result of an experiment because the act of observing it changes the result. Think of the Schrodinger's Cat.

- Professor Farnsworth

Why copy RAM

- Drive Encryption
  - OneHalf virus
- Completely memory resident malware
  - Nimda, SQLslammer
- Recovery of 'un-reallocated' space
  - Similar to recovery of deleted files.

    …but in memory
- Easier than unpacking manually
  - In some cases
- The Hacker Defense
- Strings luckiness (of course)
- Why not?

How to copy RAM

- Windows
  - \\.\\Physical\\Memory (bs=4096)
    - All except Server 2003 SP1 +
  - \\.\\DebugMemory anyone?*
  - Crash Dump
    - Forced?
    - Crash on CtrlScrl regkey
    - Notmyfault.exe
- DMA through something like Firewire**
- Special hardware (PCI card?)***

*Evidently accessed through [Nt|Zw]SystemDebugControl, also G. Garner Jr says neither object can access RAM fully…

**Proposed, in the firewire spec, but I don't know how successful (Becher)

***A Hardware Based Memory Acquisition Procedure for Digital Investigations (Carrier, Grand)

Problem

- Volatile stores like RAM change constantly
- Image cannot be validated as it can in the non-volatile world
  - We instead get a “time-sliding view”
  - Pre/post md5’s are meaningless as it is expected that RAM will be different by the time it is compared
  - Possibly use something like hash windows to show that two images made ‘quickley’ are ‘similar’ (or ssdeep…prob not needed)
- The act of creating the copy changes the state of the machine
- No write blocker installed

Time Sliding Window

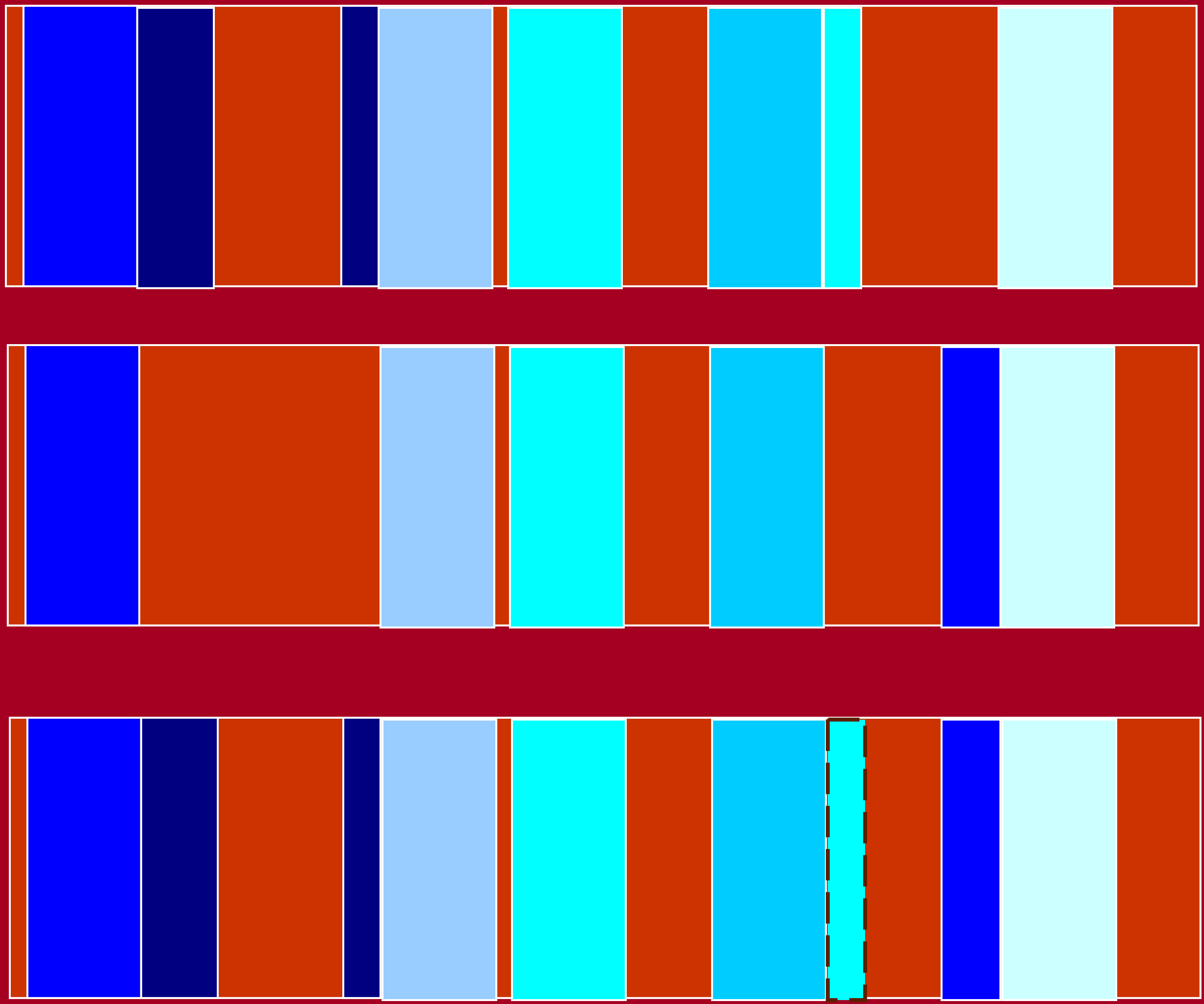
T = 0: "Pre" state

T = 1: copy is made

T = 2: "Post" state

Objects in the last half were both removed and created before being copied, and an object in the first half was removed after it was copied (but before the copy completed

T = 3: copy reflects neither state



The case for copying

- Even though it is known that creating the copy changes the state (ie. creates a new process)
- It creates less impact than interacting with the machine in order to gain insight as to the the current state

Impact

- If a first responder arrived on scene and a computer is powered off, is it considered ‘good practice’ to turn it on?
  - Why not?
  - Last booted times, file access times, pagefile, boot time run options
  - If it is preferred to not ‘touch’ the disk, why ‘touch’ RAM?
  - Lets start moving up the order of volatility chart

Minimize impact

- systeminfo.exe
- Psinfo
- netstat,
- date,
- Time
- psuptime,
- net statistics
- pulist,
- tlist,
- pslist,
- listdllsdir,
- afind,
- macmatch,
- autoruns,
- handle,
- pclipnet
- users,
- psloggedon,
- ntlast,
- Dumpusers
- ipconfig,
- fport,
- psservice,
- promiscdetect,
- netstat,
- nbstat,
- net,
- arp

vs

dd

(or similar)

…and the one on the right potentially has more information!!

Nolan, O'Sullivan, Branson, Waits. First Responders Guide to Computer Forensics. Carnegie Mellon University 2005.

The caveat

- Minimal impact is appealing, but the information is a requirement
- In order to be acceptable, at least the same amount of information that is attainable via interaction, must be attainable via analysis of the copy of the volatile store
  - Information gained: FromImageFile >= Interactive Response
  - Impact to system: FromImageCreation <= Interactive Response

Analysis

- As the area matures, the analysis of volatile stores will be able to recreate all the information regularly attained with all the previously mentioned commands
- It is essentially a combination of Reverse Engineering, Kernel Debugging… with a healthy dose of memory management and a dash of coding
- Information from non-volatile stores may be required / helpful to analysis
  - Pagefile comparison and/or “unification”
  - Another slide on this later on…

Current Analysis

- Even contemporary analysis (;Login 2005) is limited to comparing delta's in a hex editor and/or parsing for strings)
  - I'm not sure I can call the '05 login article contemporary any more. There have been an increasing number of references in the past 18 months. DFRWS, DODCyberCrime, BlackHat, IFIP…
- Samples created as part of my research showed that even a 'cleanly' booted machine would create 80-120 MB of strings output per 1 GB of RAM.
  - 100 MB of largely unusable text
  - Afhdksoi ← 1 string
- Better than nothing, but not great by any means

Proof of Concept

- Recreate the Task Manger from a RAM copy
  - Look for structures that might have been processes
    - Look for the ‘signature’ of a process
    - Discard structures that don’t meet a certain threshold (which can actually be quite stringent)
    - Rules vs recommendations
  - Brute force scan: Don’t trust linked list, active process marker, etc
  - Works on an image, dmp style dump, vmware mem file, etc
  - I unimaginatively call this “Process Locator” or procloc for the easier to type abbreviation

Proof of Concept

- Many hurdles
  - Binary level concepts
  - Different OS versions (2k vs XP, sp1 vs sp2) have <u>different offsets</u> into <u>similar</u> data structures
  - Many structures and even data types require quite a bit of decoding
  - Virtual Memory
  - Size of samples
  - Foundational concepts don’t hold
    - RAM is not as volatile as one might hope*

* Chow – uhm…a couple years ago?   - then repeated as part of this research, short story – minutes without power plug, depends on hardware

EPROCESS

- nt!_EPROCESS
- **+0x000 Pcb : _KPROCESS**
- +0x06c ProcessLock : _EX_PUSH_LOCK
- **+0x070 CreateTime : _LARGE_INTEGER**
- **+0x078 ExitTime : _LARGE_INTEGER**
- +0x080 RundownProtect : _EX_RUNDOWN_REF
- **+0x084 UniqueProcessId : Ptr32 Void**
- +0x088 ActiveProcessLinks : _LIST_ENTRY
- +0x090 QuotaUsage : [3] Uint4B
- +0x09c QuotaPeak : [3] Uint4B
- +0x0a8 CommitCharge : Uint4B
- +0x0ac PeakVirtualSize : Uint4B
- +0x0b0 VirtualSize : Uint4B
- +0x0b4 SessionProcessLinks : _LIST_ENTRY
- +0x0bc DebugPort : Ptr32 Void

etc…etc…

EPROCESS

- The EPROCESS structure is fundamental
- Among other information, PID, Creation / Deletion times, executing image name, priority, etc
- Used for scheduling
  - …well, sort of <insert discussion of threads if wanted>
- Pointers to previous and next process (double linked list)
  - Not particularly helpful in this case, as ‘rogue’ and ‘old’ processes are desirable to find as well

If it looks like an EProcess…

- Use a debugger like WinDbg (with LiveKD?) to obtain offsets to parts of an EProcess (version specific)

```
        +0x070 CreateTime       : _LARGE_INTEGER
        +0x078 ExitTime         : _LARGE_INTEGER
```

or in a more detail with the same tool as:

```
+0x070 CreateTime       : union _LARGE_INTEGER, 4 elements, 0x8 bytes
   +0x000 LowPart          : Uint4B
   +0x004 HighPart         : Int4B
   +0x000 u                : struct __unnamed, 2 elements, 0x8 bytes
      +0x000 LowPart          : Uint4B
      +0x004 HighPart         : Int4B
   +0x000 QuadPart         : Int8B
+0x078 ExitTime         : union _LARGE_INTEGER, 4 elements, 0x8 bytes
   +0x000 LowPart          : Uint4B
   +0x004 HighPart         : Int4B
```

…and smells like an EProcess…

- Use these offsets to perform various checks: (simplified for ppt)
  - Except for "IDLE" processes must have a priority > 0
  - Processes must have a page directory
  - All threads must be located in above the kernel memory bound
  - Quantum, workingset max, max # processes, sync events, etc

…it must be an EProcess!

- In practice it seems that even a few number of tests (like about 5) can produce extremely accurate results
- This methodology can also be applied to other structures… threads are an obvious next step

Cross Volatility Comparison

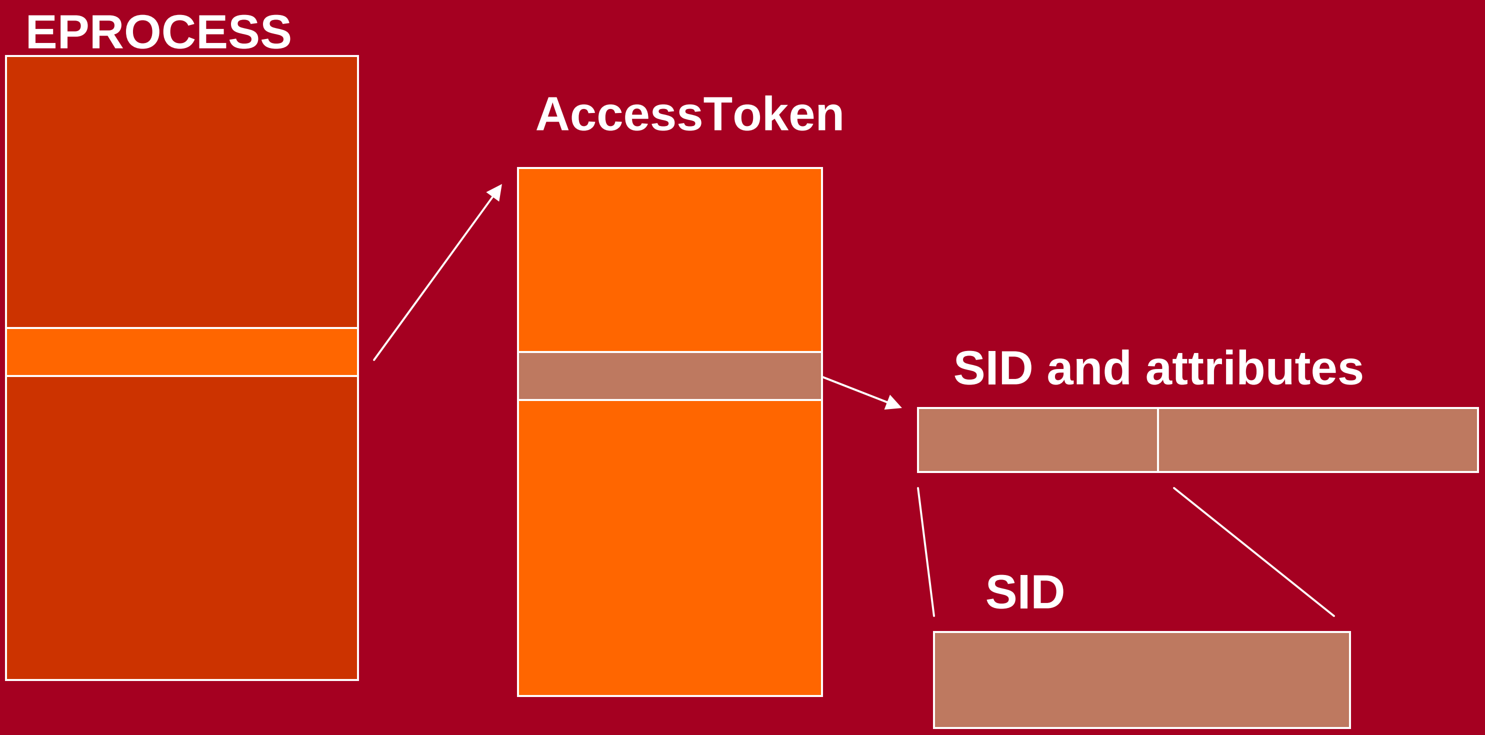
- Ideally, the analysis of volatile data stores can be aided (in practice) by information gleamed from non-volatile stores
  – Pagefile to RAM comparison (verification? Unification?)
    - A 'side effect' of crash dumps is that the page file is over written.
    - The formation of the DMP file is actually an interesting process…
  – Event log correlation
  – What if the disk shows Windows XP, but RAM shows Linux structures?
  – etc

NUCIA

Your Key to Security

CanSecWest2007

PoC: Process Owner



This can't actually be decoded further than SID, because the SID to "human readable" mapping is not held in RAM. This is a prime example of how information from a non-volatile store may be needed to aide the volatile analysis (registry, SAM, Domain)

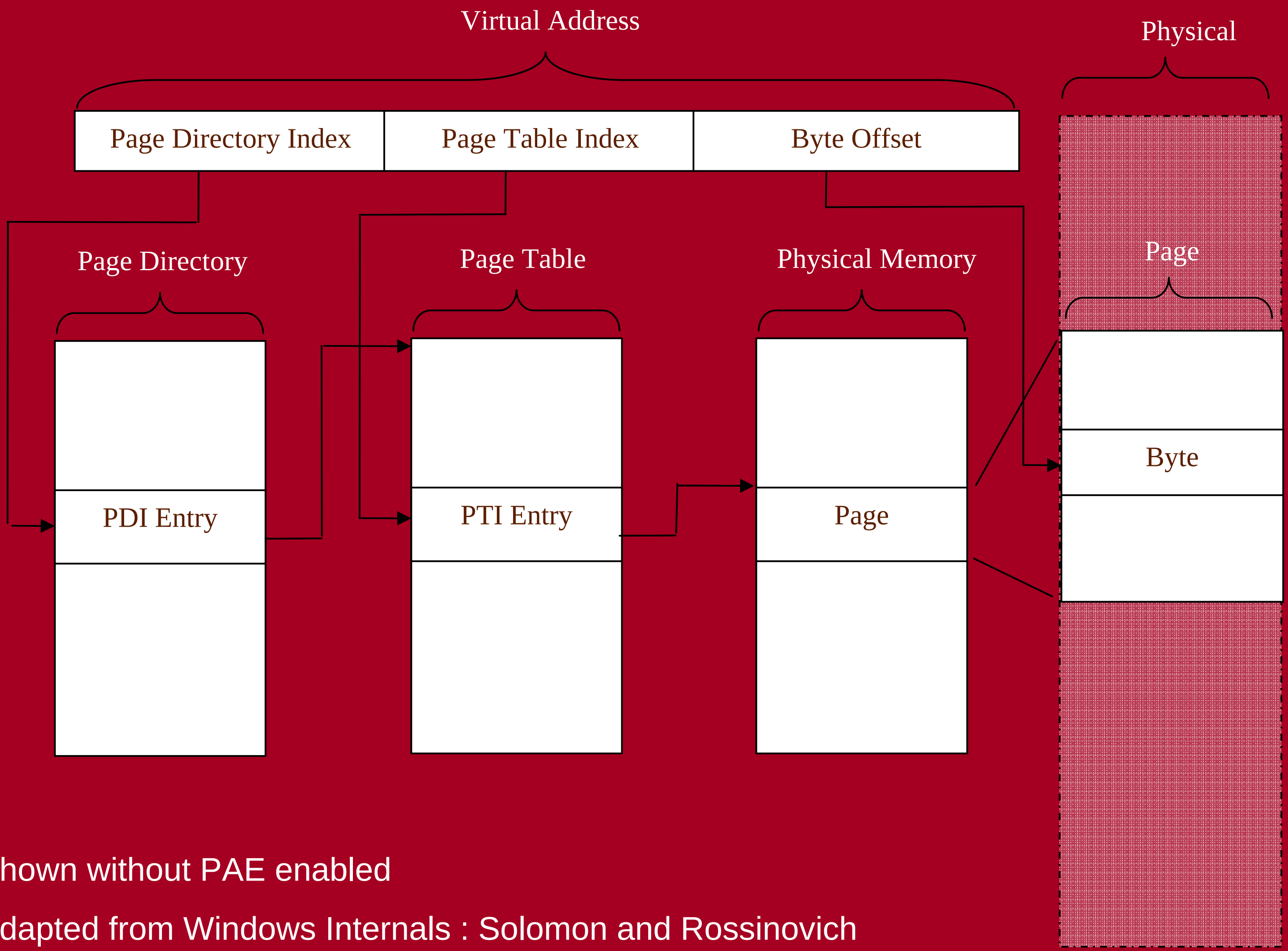
NUCIA

Your Key to Security

PoC: Virtual Addressing

- The Process Environment Block(PEB) is always in the same place!
  - Well, it’s a virtual address, so it’s ‘real’ location needs to be decoded from virtual to physical using other values from the structure

PoC: Virtual Memory



Shown without PAE enabled

Adapted from Windows Internals : Solomon and Rossinovich

See also Intel Software Developers Manuals

PoC: FileTime

- 100 nanosecond intervals since 1601
- vs UNIX 1 second intervals since 1970
- and it’s a 64 bit value, stored as two 32 bit values, each lil endian

- # Filetime conversions

# FFFFFF00 00000000	= under 1.5 seconds
# 00000001 00000000	= under 1.5 seconds
# 00000010 00000000	= about 26 seconds
# 00000000 01000000	= about 7:09
# 00000000 10000000	= about 1:51:31
# 00000000 00010000	= about 1 day 6:32:31
# 00000000 00100000	= about 21 days 8:40:18
# 00000000 00000100	= about 11 months 22 days 18:44:57
# 00000000 00001000	= about 14 years 3 months 10 months 11:59:22
# 00000000 00000001	= about 228 years 5 months 5 days 23:50:03
# 00000000 00000010	= about 6353 years 6 months 18 days 21:21:00

PoC: FileTime

```
•   sub Win2Unix4() {
•   my $Lval = shift;
•   my $Hval = shift;
•   my $Time = 0;
•   my $Shift = 11644473600;  #Shift of time
•   if(($Lval == 0) and ($Hval ==0)){
•           return $Time;
•   }else{
•           $Time = int(($Hval * 2**32 / 10000000) + ($Lval / 10000000));
•           $Time -= $Shift;
•   }
•           if ($Time < 0){
•                   $Time = 0;
•           }
•           return $Time;
•   }
```

Actually not that much code!

PoC: Demo

- Create Images
  - dd example
    - trusted binary' (live CD, statically linked)
    - external Mass storage container
    - 'raw' type
  - Forced Crash condition
    - registry keys
    - 3rd party testing tool
    - External Mass storage container
    - proprietary DMP format created on reboot

C

- Use ~~PERL~~ to parse through a ton of data
  - Practical Extraction and Reporting Language

PoC: Demo

- Images created from cleanly installed OSes
  - Only video/network drivers
- IBM MPro machine(s) with 512 MB RAM (turned off for 15 minutes)
- Helix 1.7 CD inserted and physical memory is imaged (if possible)
- Registry keys created to set crashdump to ‘on’ and ‘full’
- Nonmyfault.exe used to forced system crash and thus a crash-dump style image
- Considering posting test images publicly…

PoC: Demo

- On a removable hard drive
  - raw style captures via helix dd
  - Crash style captures via nonmyfault.exe & crashdump
- Just typical ~~PERL~~
  - ~~Activestate~~
  - Cross platform
- The idea is to replicate as much or more information that Windows Task Manager

PoC: Demo

- Ready….go!
  - MEMORY.DMP format
  - dd –style
  - Processes
  - Threads
  - Exe extraction
    - Virtual memory layer required
    - Finished coding this during Adam Laurie's talk yesterday – consider BETA

NUCIA

Your Key to Security

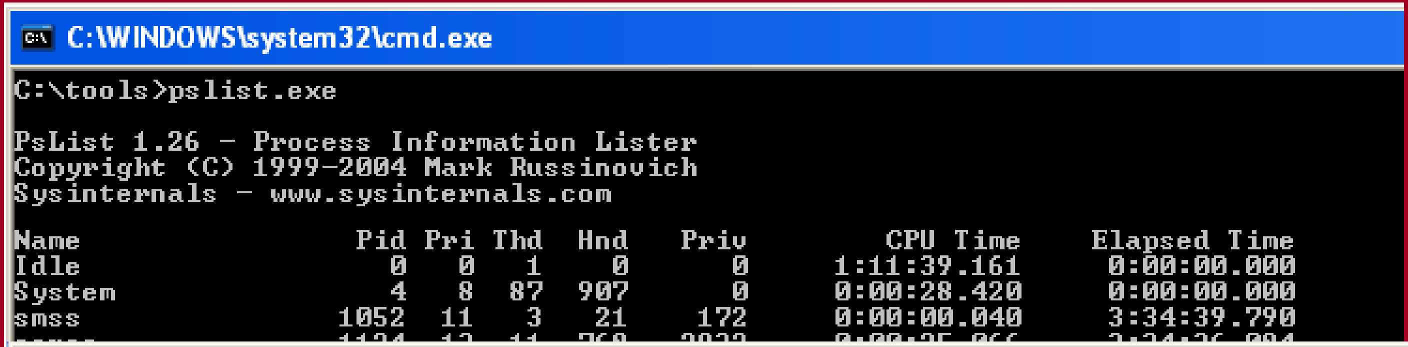
CanSecWest2007

PoC: Demo

- SO…given that the demo was successful
  - – We saw that it is possibly to get as much (or more) information post-incident while cause as much (or less) impact to the system
- OR if it wasn't successful
  - – We _should_ have seen the above ;-)

Goals met

- Information requirement
  - pslist > taskmanager
  - procloc ~= pslist

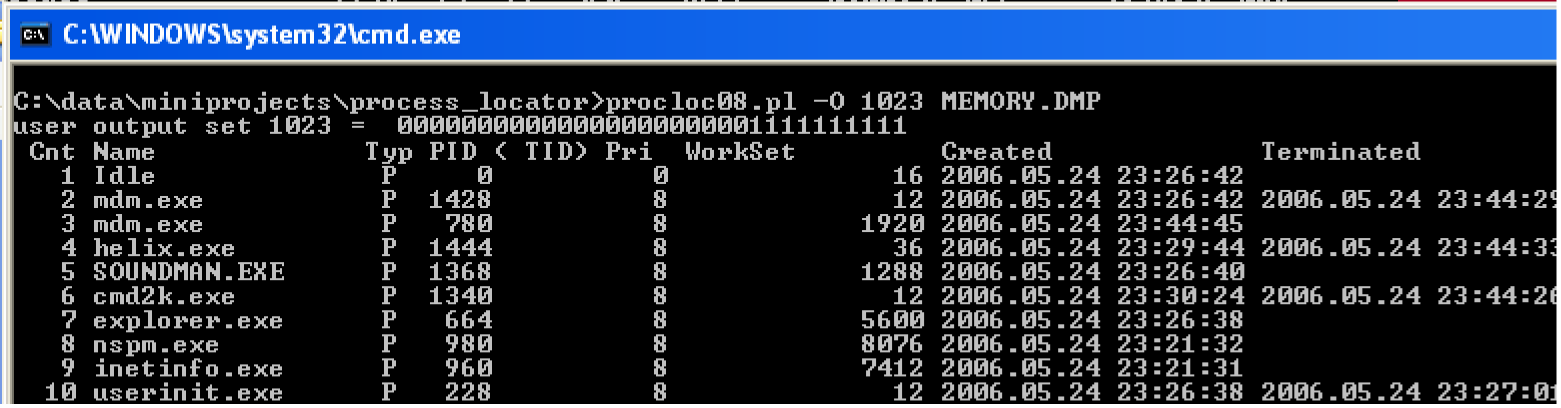


C:\WINDOWS\system32\cmd.exe

```
C:\tools>pslist.exe

PsList 1.26 - Process Information Lister
Copyright (C) 1999-2004 Mark Russinovich
Sysinternals - www.sysinternals.com

Name                Pid Pri Thd  Hnd   Priv        CPU Time    Elapsed Time
Idle                  0   0   1    0      0     1:11:39.161     0:00:00.000
System                4   8  87  907      0     0:00:28.420     0:00:00.000
smss               1052  11   3   21    172     0:00:00.040     3:34:39.790
```



C:\WINDOWS\system32\cmd.exe

```
C:\data\miniprojects\process_locator>procloc08.pl -O 1023 MEMORY.DMP
user output set 1023 =  00000000000000000000001111111111
 Cnt Name                Typ PID ( TID) Pri  WorkSet          Created             Terminated
   1 Idle                 P      0         0         16 2006.05.24 23:26:42
   2 mdm.exe              P   1428         8         12 2006.05.24 23:26:42 2006.05.24 23:44:29
   3 mdm.exe              P    780         8       1920 2006.05.24 23:44:45
   4 helix.exe            P   1444         8         36 2006.05.24 23:29:44 2006.05.24 23:44:33
   5 SOUNDMAN.EXE         P   1368         8       1288 2006.05.24 23:26:40
   6 cmd2k.exe            P   1340         8         12 2006.05.24 23:30:24 2006.05.24 23:44:26
   7 explorer.exe         P    664         8       5600 2006.05.24 23:26:38
   8 nspm.exe             P    980         8       8076 2006.05.24 23:21:32
   9 inetinfo.exe         P    960         8       7412 2006.05.24 23:21:31
  10 userinit.exe         P    228         8         12 2006.05.24 23:26:38 2006.05.24 23:27:0
```

Future work
(process specific)

- Compare the Brute force list to a list obtained by walking the list
  - ‘cross view diff’ off the untrusted list with the ‘more trusted’ list – red flags
- “unification of virtual memory”
  - Swap all pages ‘in’ kinda, then defrag it?!
- Good, OS version auto detection
- Automate correlation with other sources of information
  - If you supply a Registry hive, auto process owner from SID
- Automatically and/or selectively extract executables

NUCIA

Your Key to Security

CanSecWest2007

Future work
(process specific)

- Flag processes/threads that aren't "playing by the rules"
  - Window title, path, pointers, parent, etc
- Follow the entire tree
  - Attribute every thread to a process, every page to what allocated it, parent/child link…etc. Then what's left?
- Support the /PAE and /3G boot switches
- Vista support (right now, parsing looks to be easy, acquisition looks to be hard)
- Non i386 support
- Parsing from within EnCase?

Future work
(memory, not process specific)

- File cache
  - Delayed write to disk, usually for priority reasons
- Network connections
  - Tied to processes
- Video card?
  - Some malware is executing directly from video card memory

Future Setbacks
(perceived – opinion)

- Malware that manipulates acquisition
  - There are about 3 non-hardware ways to acquire, trivial to 'hook' these and hide during acquisition (of a live non-rebooted machine)
  - Not deny access, simply modify output – similar to techniques used in rootkits today to hide – processes from task manager, etc
- Microsoft will continue to make it more and more difficult to get to 'RAW' RAM
  - Restriction to objects
  - Other things like VISTA's randomization
- RAM becomes even more scattered that the current memory model
  - Like VISTA's RAM extender (USB) – ReadyBoost
- New architectures

Food for thought:

- But the cases I have don’t require all this stuff!
  - The hacker defense will bear it’s face eventually
  - RAM imaging is going to be ‘industry standard’ it’s just a matter of time
  - You may be more likely to have a Rootkit that you think*

*You’ve heard of Sony right?

Google starters

(in no real order)

- PhysicalMemory object
- MyFip.H
- Fanbot.A
- DKom
- Hacker Defender
- Shadow Walker
- EProcess
- The artist formerly called Sysinternals (process explorer for starters)
- “Blue Pill” + rootkit
- UPX
- Packer
- Sony Rootkit
- RAIDE
- TRUMAN
- Shimmer.a

- Tim Vidas ☺
- Mariusz Burdach
- Jesse Kornblum
- Andreas Schuster
- Aaron Walters
- Nick Petroni
- Harlan Carvey

- ProcLoc
- Volatools
- WMFT
- PTFinder
- LSPI
- Memparse

Question #1 from the Audience

- So how do you recommend that I implement RAM into my investigations?
  - Officially I'm probably not supposed to answer that
    - The whole I'm not a lawyer and don't play one on TV thing
    - The whole I'm an Academic not a practitioner thing
  - That said: If the situation allows, *maybe* the best way is to:
    - arrive on scene
    - get ready (BIOS cheat sheet, dd on bootable CD)
    - pull plug ***
    - plug back in immediately ***
    - boot to CD
    - copy RAM
    - image disk as normal
    - take both back with you

***Or maybe it's via dd on a USB mass storage – copy w/o unplugging, time / results will tell

Other Questions?

- Contact info
  - I've got a bunch of cool CanSecWest meishi (business cards)
- Source Code
  - Completely FREE
  - GPL
  - Perl is available on the net already
  - C is available right now (come and get it). It will be available on the net after I clean it up a bit.

Cited

- Windows Internals, Russinovich / Solomon
- Intel 64 and IA-32 Architectural Software Developers Manuals
  – PDFs are online
  – Dead Tree copies are FREE
- Rootkits, Hogland / Butler
- Reversing, Eilam
- And the papers/documents footnoted in the slides