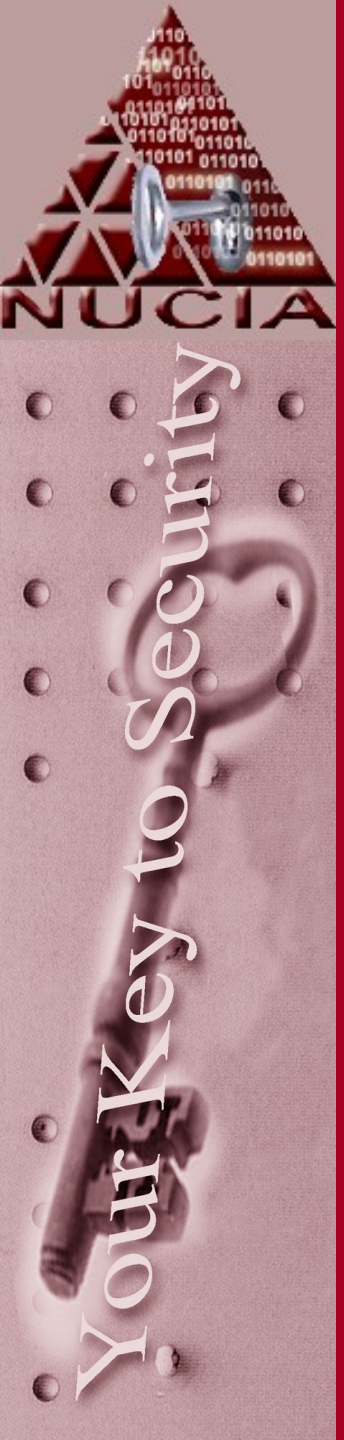


Forensic Analysis of Volatile Data Stores

CERTConf2006

Tim Vidas

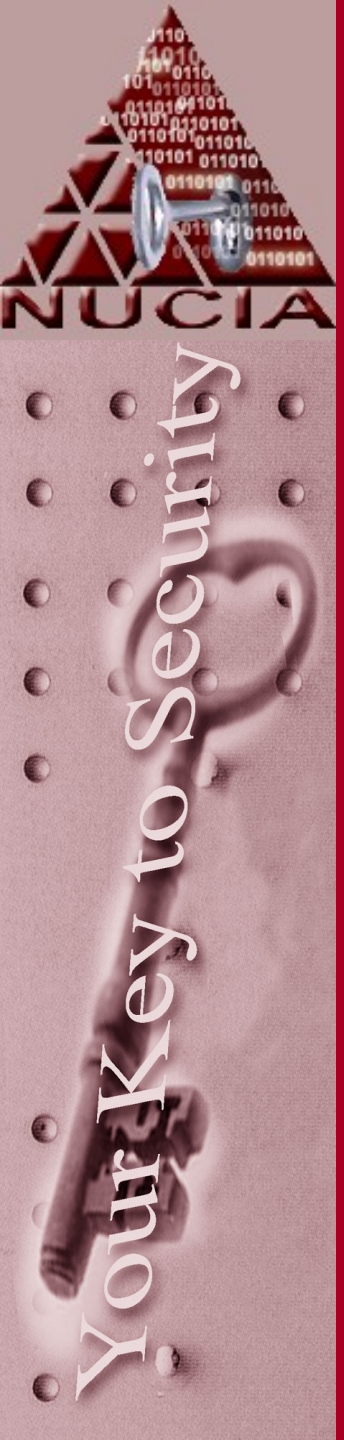
Because sometimes, your memory just fails you

Who am I?

- Tim Vidas
  - Sr. Tech. Research Fellow
  - UNO/PKI/NUCIA
  - Certs: CISSP, 40xx, Guidance, AccessData etc.
  - Instructor: UNO, Guidance, LM RRCF

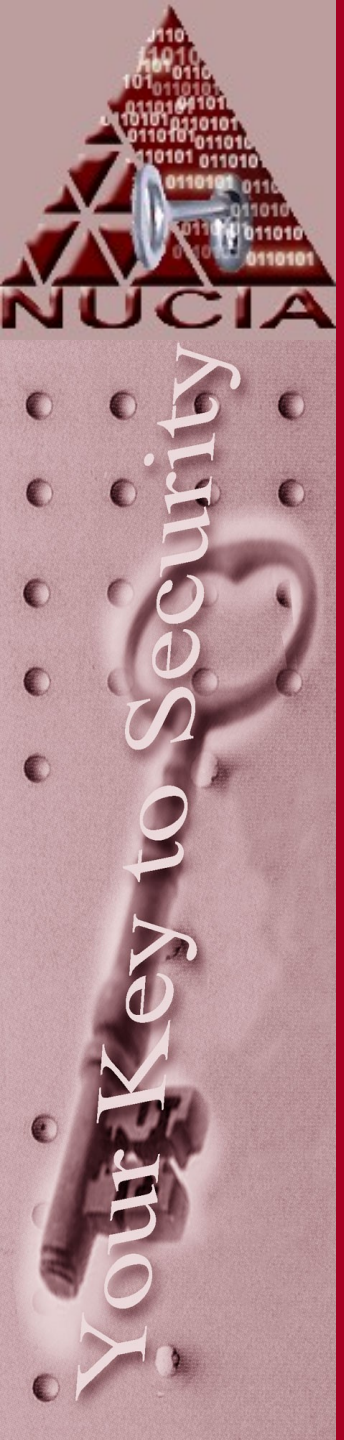
Assumptions

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



Evidence Volatility

- Memory (more volatile)
- Processes
- Network connections
- File system
- Disk Block (less volatile)

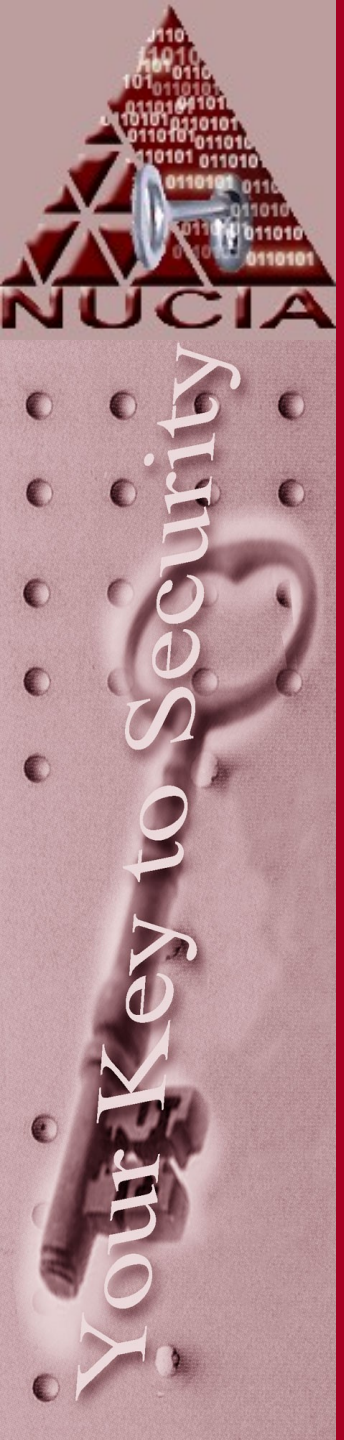


Live system consequences

- Running processes
- Active network connections, activities
- Logged in users
- Encryption
- Viruses (one-half)

When to pull the plug

- We've touched on it before,
  - Secure the Crime Scene?
    - Yellow tape
    - Entrance granted to only authorized people
  - Computer crime scene is where?
    - Yellow tape the whole internet?
    - Pulling the plug
      - "dirty shutdown" fairly regardless of OS
      - Bootable media analysis, r/o mount



When to pull the plug

- Why not?
  - Ongoing activity
  - Volatile information
    - Ram
    - Net connections
    - Etc
  - Clean shutdown can cause MANY changes to the system
  - Leaving on can cause changes…
- Investigate before shutdown
- Possible response strategy
  - If duplication is deemed necessary, shutdown
  - If ongoing activity needs attention, or the system cannot be downed, live-response.

In place analysis

- Who had access to the system?
  - Who was supposed to
- Is authentication required?
- What type of networks are present and how are they connected?

Your Key to Security

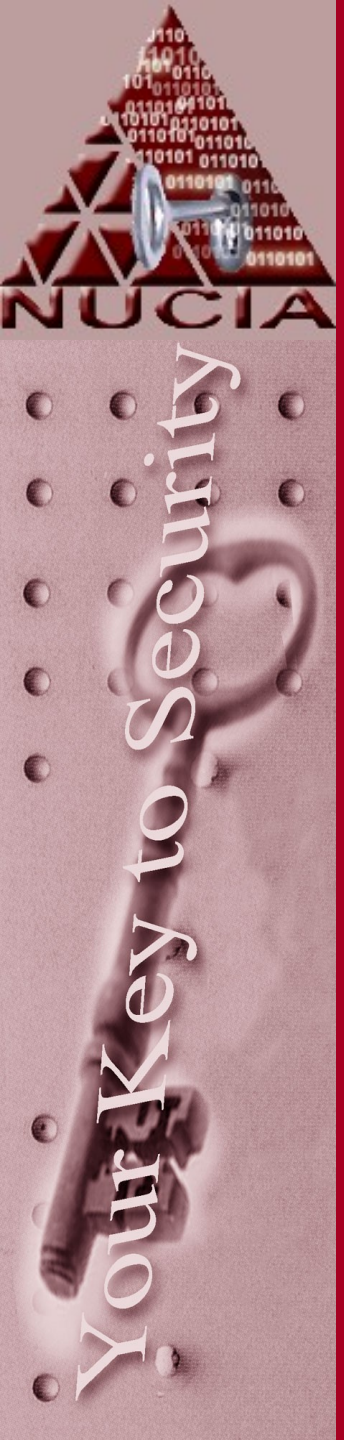
In place Analysis

- Netstat, search for new files, new services, high execution times, arp table, new users, open files, connected users,
- Fport, ps tools, process explorer, autoruns, rootkit revealer, regmon, filemon, listdlls,
  - Sys Internals – get the tools while you can!!!

Common Incident Response Steps (Nolan, O'Sullivan, Branson, Waits – Carnegie Mellon 2005)		
	Windows	Linux
System Profile	systeminfo.exe, psinfo	/proc (version, uptime, meminfo, filesystems, cpuinfo), uname
Date and Time	netstat, date, time	netstat, date
uptime	psuptime, net statistics	uptime, w
Runing Processes	netstat, pulist, tlist, pslist, listdlls	ps, w, top, fuser, modules.conf, ldd, ls
Open Files, startup, clipboard	dir, afind, macmatch, autoruns, handle, pclip	ls, find, lsof, file, /etc/rc* directories, chkconfig, inittab, cron, at
Users	net users, psloggedon, ntlast, dumpusers	who, last, lastlog, /etc/passwd, /etc/shadow
Network information	ipconfig, fport, psservice, promiscdetect, netstat, nbstat, net, arp	ifconfig, netstat, /var/log/messages, arp

Decision: in place Analysis

- Most of the commands we've seen to date are considered 'intrusive'
- As in, they change the state of the machine – in some cases considerably
- This is one of many reasons you must document all of your steps
- This also applies to any scripted response, both your custom ones and canned ones like IRCR or EnCase's SweepCase / SweepEnterprise.



THE PROBLEM

- When encountering a live system, the official stance for ‘powering down’ a system for duplication is pulling the plug
- Generally this is considered better than something like “start -> shutdown” because no shutdown tasks (cleanup) is performed
- Unfortunately this doesn’t preserve much about the system state

THE PROBLEM

- To record system state, responders may interact with the system using the aforementioned tools
- However this has an impact to the system
- Similar to how creating a new file has the possibility of overwriting un-reallocated clusters containing latent data of deleted files, new processes take up space in RAM

Acquisition

- Varies on situation
- dd
  - Just like dd on a hdd
  - Under windows there is a PhysicalMemory object that can be copied from
  - This may be ‘going away’, Server 2003 SP1 indicates this
  - Available on Helix CD (greater than version 1.5 … I believe)

Acquisition

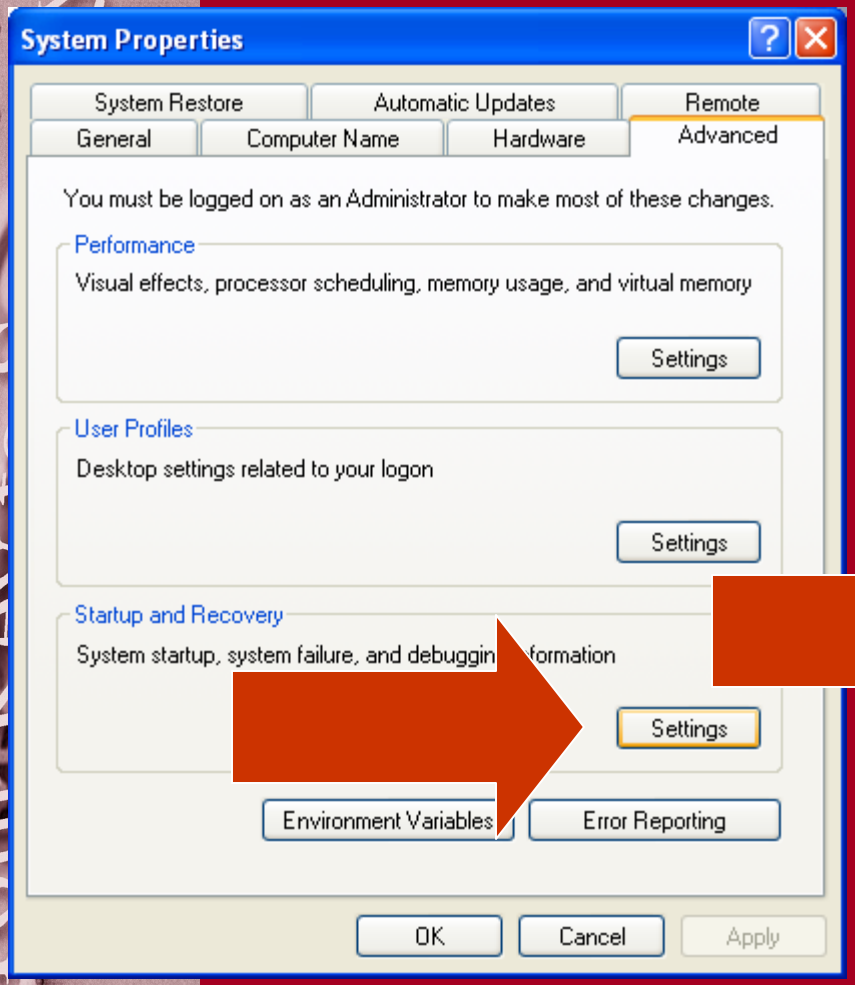
- dd if=\\.\Device\PhysicalMemory of=memory.bin bs=4096

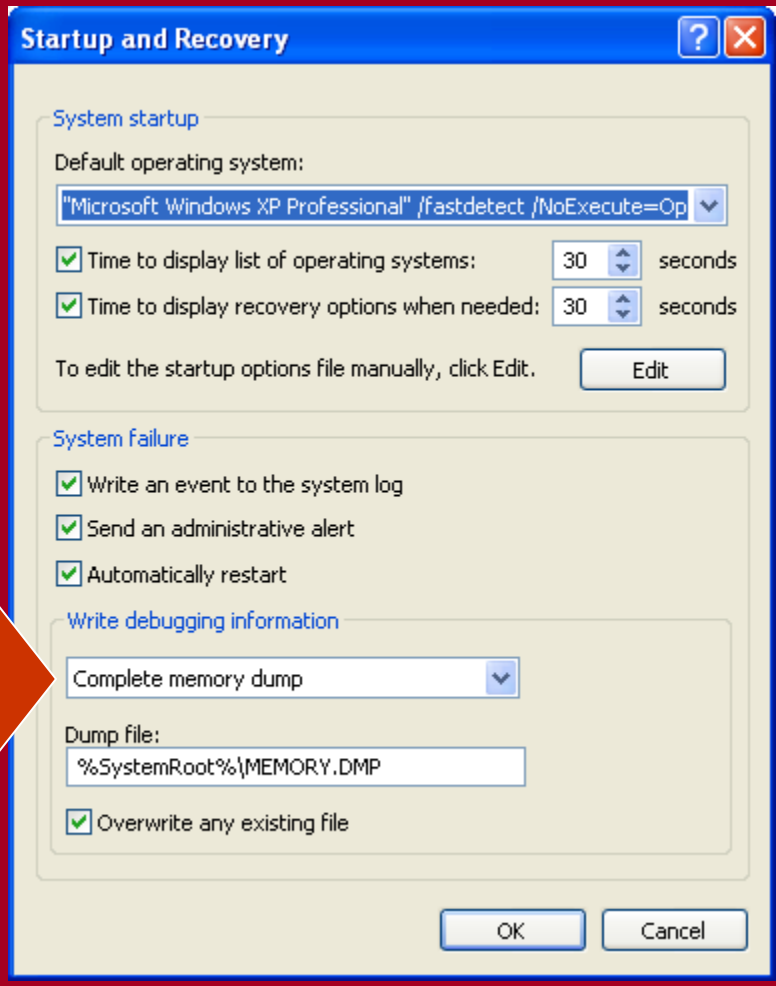
- 4096 – page size
- Memory.bin – normally you would not want this to go onto the suspect disk… otherwise you are potentially over-writing quite a bit of potential evidence

Acquisition

- Premeditation
  - This is usually not the case
  - If you can plan ahead
    - enabling full crash dumps
    - and crash on CTRL-SCRL Lock keys
  - Possibly even special hardware dumping device

My Computer


System Properties
System Restore
Automatic Updates
Remote
General
Computer Name
Hardware
Advanced
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings
User Profiles
Desktop settings related to your logon
Settings
Startup and Recovery
System startup, system failure, and debuggin formation
Settings
Environment Variables
Error Reporting
OK
Cancel
Apply


Startup and Recovery
System startup
Default operating system:
"Microsoft Windows XP Professional" /fastdetect /NoExecute=Op
Time to display list of operating systems:
30
seconds
Time to display recovery options when needed:
30
seconds
To edit the startup options file manually, click Edit.
Edit
System failure
Write an event to the system log
Send an administrative alert
Automatically restart
Write debugging information
Complete memory dump
Dump file:
%SystemRoot%\MEMORY.DMP
Overwrite any existing file
OK
Cancel

Crash keys

- HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\CrashControl
- CrashDumpEnabled REG_DWORD 0x0 = None
  CrashDumpEnabled REG_DWORD 0x1 = Complete memory dump
  CrashDumpEnabled REG_DWORD 0x2 = Kernel memory dump
  CrashDumpEnabled REG_DWORD 0x3 = Small memory dump (64KB)
- Related keys of interest are:
- AutoReboot REG_DWORD 0x1
  DumpFile REG_EXPAND_SZ %SystemRoot%\Memory.dmp
  MinidumpDir REG_EXPAND_SZ %SystemRoot%\Minidump

Forcing Crashes

- REG_DWORD named CrashOnCtrlScroll with a value of 1 added to HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\i8042prt\Parameters
- New functionality to windows!
  - Right control key plus two presses of scroll lock crashes the system
- Only works on PS2 keyboards (not usb) !!!! (argh)

Forcing Crashes

- NotMyFault
  - Mark Russinovich
  - involves loading a small driver (~7 kb) called MyFault.sys
  - user level application NotMyFault.exe (~50 kb).

The user executable issues calls to the kernel loaded driver which crashes the system on behalf of the user level executable in various ways. This pair of files can force a number of crash conditions in Windows

Acquisition: Time Sliding View

- Of course with a hdd type media there is an intrinsic benefit of the media being static
- When considering volatile stores, the data is continuously in use
  - No write blockers
  - No MD5 comparison to original
- The duplicate is not going to accurately represent a single point in time, but rather a “time sliding view” of the store

Acquisition: Privs

- Having “permission” to create an image
  - Both in the form of “written permission” if the situation warrents this
  - And system permissions
    - Administrator / root / poweruser?
- Upon initial response having the appropriate permissions may not be the case
  - this makes things interesting
  - There may be other talks about this… ☺

Memory Analysis

- Traditionally RAM information isn't even available
- In the rare cases that it is available, the analysis involves very simplistic actions such as running strings on the image
  - Research note: strings produces about 50-80 MB of largely unusable **text** for typical 512 MB RAM dumps
  - 50-80 MB is better than 512, but still not particularly useful

THE IDEA

- What if instead of stomping all over the current RAM state by running a bunch of tools we could create an image of RAM and go back and investigate the state of the system later?
- This only creates (at a minimum) one process to perform the copy
  - Realistically a few, cause the crash, possibly insert a few registry keys, but the bottom line is that the impact is potentially a LOT smaller than typical response today

THE IDEA

- To be acceptable, the information attained from the memory image must be comparable to information that is attainable by interacting with the system
- Methods comparison:
  - Information gained:

    FromImageFile >= Interactive Response
  - Impact to system:

    FromImageFile <= Interactive Response

THE PROOF (of concept)

- Scan through memory image and locate processes (task manager reconstruction)

Finding processes

- Simply:
  - Got through memory byte by byte and look for something that looks like a Dispatch Header
  - If found see if it is a header for an EProcess (DH can also be threads, semaphores, etc)
  - Perform various checks on potential process candidates

EPROCESS Structure (Russinovich, Solomon. Windows Internals 2005)	
EPROCESS Element	Purpose
Kernel process (KPROCESS or PCB) block	Common dispatcher object header, pointer to the process page directory, list of kernel thread (KTHREAD) blocks belonging to the process, default base priority, quantum, affinity mask, and total kernel and user time for the threads in the process.
Process identification	Unique process ID, creating process ID, name of image being run, window station process is running on.
Quota block	Limits on nonpaged pool, paged pool, and page file usage plus current and peak process nonpaged and paged pool usage. (*Note*: Several processes can share this structure: all the system processes point to the single systemwide default quota block; all the processes in the interactive session share a single quota block Winlogon sets up.)
Virtual address space descriptors (VADs)	Series of data structures that describe the status of the portions of the address space that exist in the process.
Working set information	Pointer to working set list (MMWSL structure); current, peak, minimum, and maximum working set size; last trim time; page fault count; memory priority; outswap flags; page fault history.
Virtual memory information	Current and peak virtual size, page file usage, hardware page table entry for process page directory.
Exception local procedure call (LPC) port	Interprocess communication channel to which the process manager sends a message when one of the process's threads causes an exception.
Debugging LPC port	Interprocess communication channel to which the process manager sends a message when one of the process's threads causes a debug event.
Access token (ACCESS_TOKEN)	Executive object describing the security profile of this process.
Handle table	Address of per-process handle table.
Device map	Address of object directory to resolve device name references in (supports multiple users).
Process environment block (PEB)	Image information (base address, version numbers, module list), process heap information, and thread-local storage utilization. (*Note*: The pointers to the process heaps start at the first byte after the PEB.)

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…

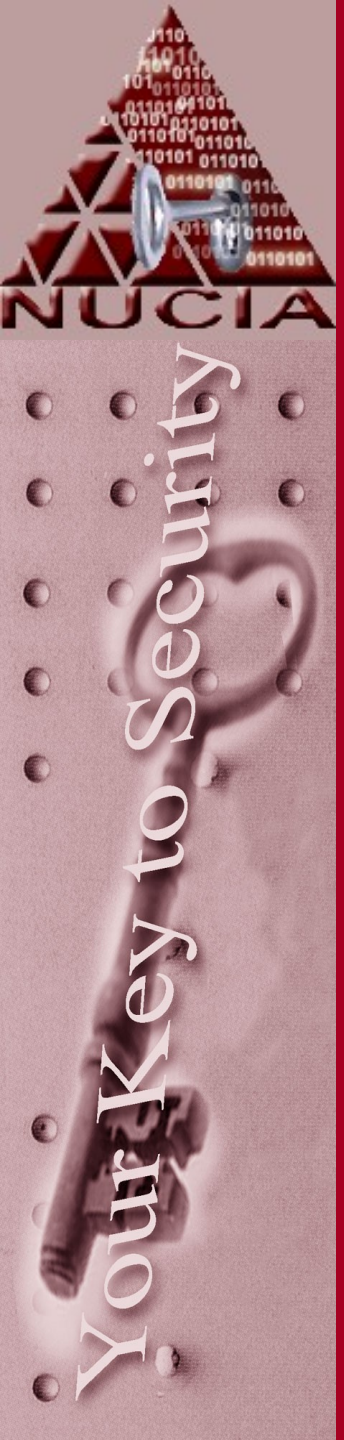
- Use these offsets to perform various checks:
  - Except for “IDLE” processes must have a priority > 0
  - Processes must have a page directory
  - All threads must be located in above the kernel memory bound
  - Quantum, workingset max, max # processes, sync events, etc

…it must be an EProcess!

- In practice it seems that even a few number of tests (like less than 5) produce extremely accurate results

Cross Volatility Comparison

- Ideally, the analysis of volatile data stores can be aided by information gleamed from non-volatile stores
- For example, a process runs under a given context. EProcesses contain an Access_Token which contains a SID, when reversing from memory this is as close as we can get to a username. If the SAM can be accessed this can be correlated to a human friendly username.



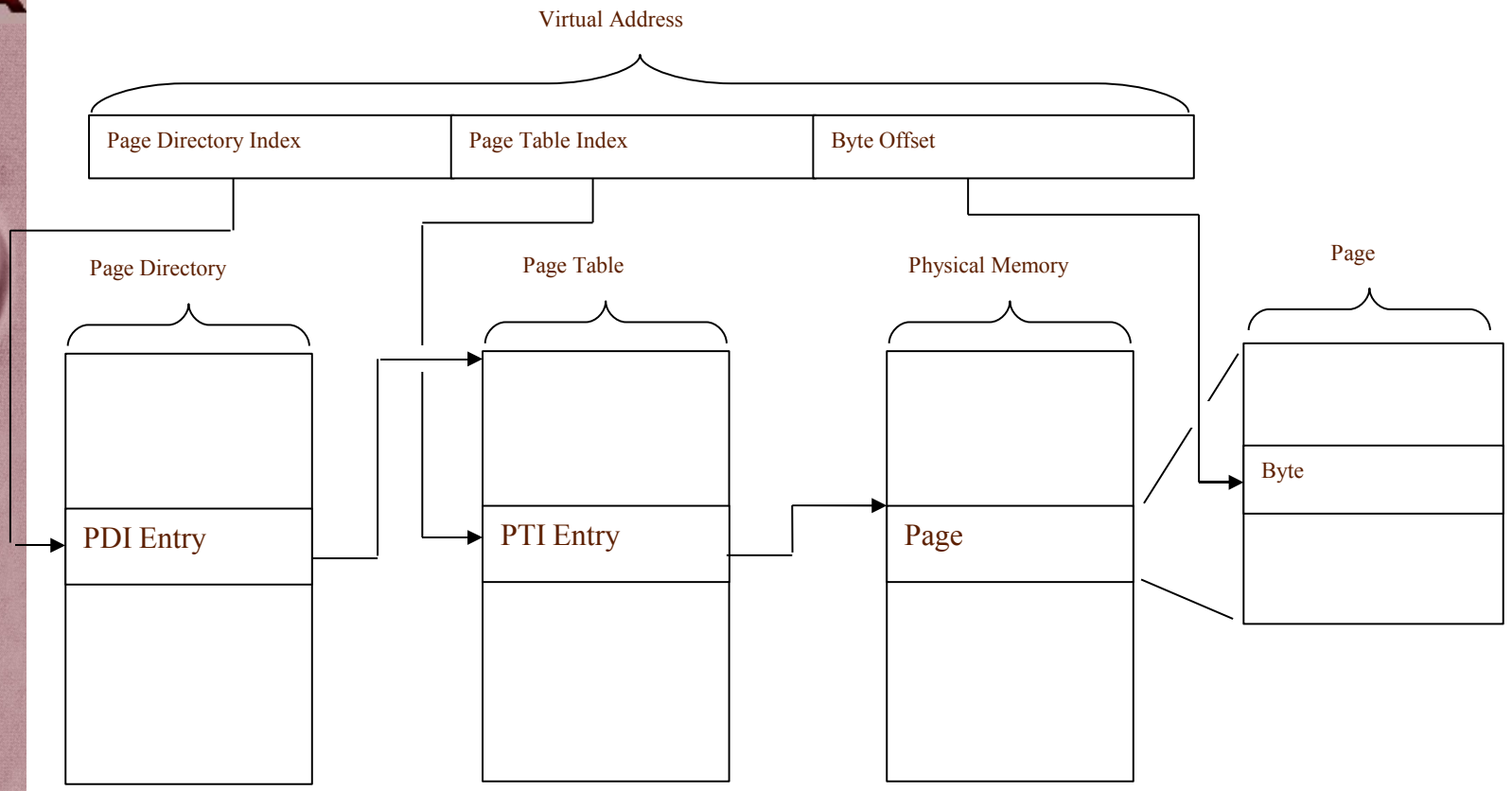
Cross Volatility Comparison

- Other interesting things can be done
  - Passwords in memory
  - Crypto keys
  - Pagefile to RAM comparison (verification?)
  - Latent files in RAM that have been wiped from disk

Cross Volatility Comparison

- A ‘side effect’ of crash dumps is that the page file is over written (is this is why the page file needs to be at least the size of physical RAM?).
- The formation of the DMP file is actually an interesting process…

Helpful slides for code deciphering



Helpful slides for code deciphering

- Only threads execute, not processes
- RAM is not at volatile as you might think. Power can be removed (as in pull the plug) in the order of 30 seconds without effecting the state of RAM
- Page files are typically capped at 4GB (larger paging is enabled by having multiple pagefiles)

Helpful slides for code deciphering

- Most Windows OSes (x86) support 4 or less GB of RAM (enterprise / datacenter allow for 32-64)

Demo

- Process Locator 0.6
  - View the source
    - There is a lot in there not mentioned in the slides, but it's commented pretty well
  - Works on any NT based OS
  - On this machine (1.5 MGhz p3m, 1GB RAM) it takes about 30 minutes to parse 1GB image.