Defeating Windows memory forensics

29c3

December 28, 2012.

Luka Milković
Luka.Milkovic@infigo.hr

INFIGO IS http://www.infigo.hr

Agenda

- Memory forensics
  - Why?
  - How?
- Previous memory anti-forensic techniques
  - Windows related
- Memory acquisition process – flawed by design?
- Defeating Windows memory forensics
  - What about user mode?
- Possible solutions

INFIGO Information security

whoami

- As Carlos would say – *nobody* (but working on a privilege escalation)
- In six (and a half) words and two pics
  - Infosec consultant

  - Avid cyclist

  - Love coding/hacking

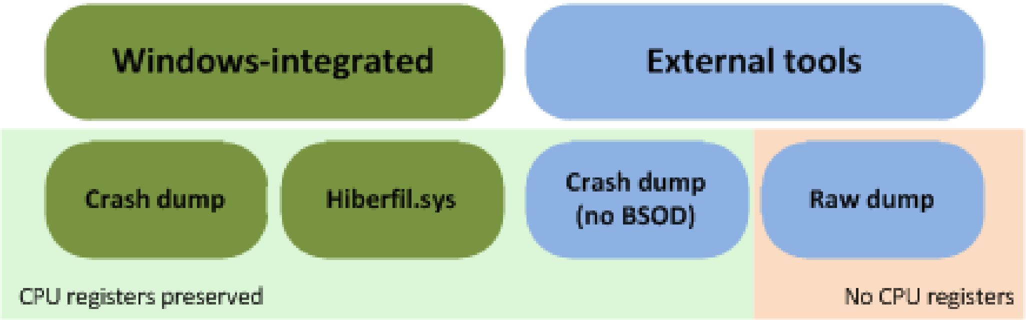
Memory forensics – why?

- Disk forensics prevalent, but memory forensics increasingly popular

- Used by incident handlers…
  - Malware detection
    - objects hidden by rootkits (processes, threads, etc.)
    - memory-resident malware
    - unpacked/unencrypted images
  - Recently used files
  - Valuable objects (both live and „dead”)
    - processes, threads, connections…
- … and the bad guys
  - Password recovery

Memory forensics – how?

- Two consecutive processes
  - *Memory acquisition*
  - *Memory analysis*
- Acquisition (software based)



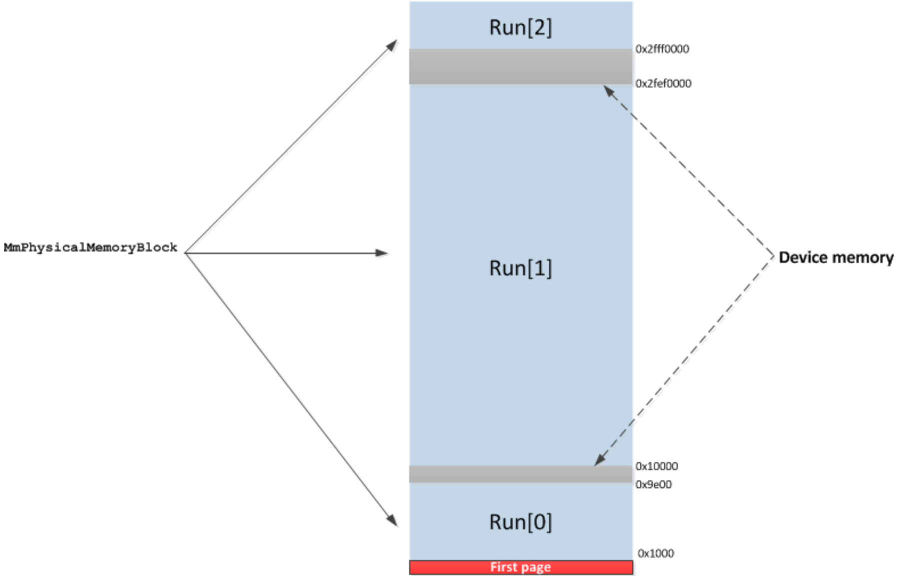
  - Many tools, focus on popular (and free)
    - Moonsols Windows Memory Toolkit (Win32dd)
    - Mandiant Memoryze
    - FTK Imager
    - MDD
    - other (will be mentioned later)

Memory forensics – how? (2)

- Acquisition internals
  - User mode and kernel mode (driver) component
  - Why driver?
    - physical memory cannot be read from the user mode (after Windows 2k3 SP1)
    - usually just a proxy for `\\Device\PhysicalMemory`
    - documented kernel APIs – `MmMapIoSpace()`
    - undocumented kernel functions – `MmMapMemoryDumpMdl()` – win32dd „PFN mapping"

Memory forensics – how? (3)

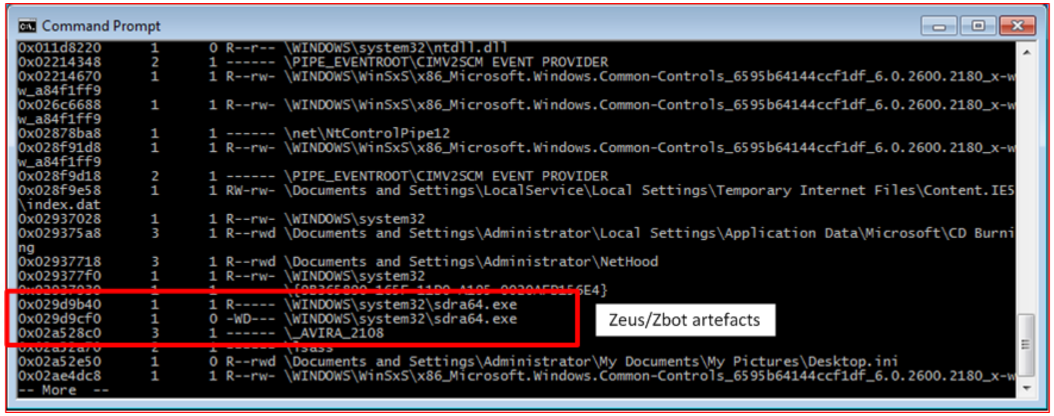
- Format differences
  - Crash dump contains registers, but no first page and device memory mappings



  - Raw dump – no registers
    - some tools omit device memory and first page
    - if important, check the tool documentation

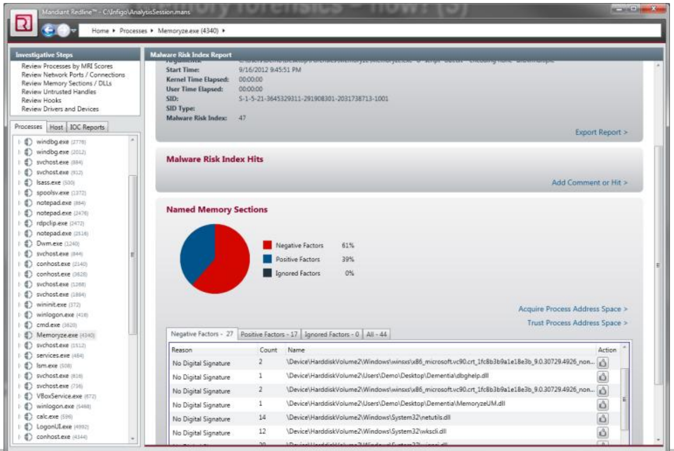
Memory forensics – how? (4)

- Analysis
  - Finding OS and „user” artifacts in the image
  - Free and commercial tools
    - Volatility Framework
    - Mandiant Redline/Memoryze
    - HBGary Responder, partially EnCase and many other
  - All support raw dump, weak support for hib/crash file



```
0x011d8220       1        0 R--r-- \WINDOWS\system32\ntdll.dll
0x02214348       2        1 ------ \PIPE_EVENTROOT\CIMV2SCM EVENT PROVIDER
0x02214670       1        1 R--rw- \WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2180_x-w
w_a84f1ff9
0x026c6688       1        1 R--rw- \WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2180_x-w
w_a84f1ff9
0x02878ba8       1        1 ------ \net\NtControlPipe12
0x028f91d8       1        1 R--rw- \WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2180_x-w
w_a84f1ff9
0x028f9d18       2        1 ------ \PIPE_EVENTROOT\CIMV2SCM EVENT PROVIDER
0x028f9e58       1        1 RW-rw- \Documents and Settings\LocalService\Local Settings\Temporary Internet Files\Content.IE5
\index.dat
0x02937028       1        1 R--rw- \WINDOWS\system32
0x029375a8       3        1 R--rwd \Documents and Settings\Administrator\Local Settings\Application Data\Microsoft\CD Burni
ng
0x02937718       3        1 R--rwd \Documents and Settings\Administrator\NetHood
0x029377f0       1        1 R--rw- \WINDOWS\system32
0x02937030       1        1 ------ \{0B365800-165F-11D0-A195-0020AFD156E4}
0x029d9b40       1        1 R----- \WINDOWS\system32\sdra64.exe
0x029d9cf0       1        0 -WD--- \WINDOWS\system32\sdra64.exe
0x02a528c0       3        1 ------ \_AVIRA_2108
0x02a52a70       2        1 ------ \lsass
0x02a52e50       1        0 R--rwd \Documents and Settings\Administrator\My Documents\My Pictures\Desktop.ini
0x02ae4dc8       1        1 R--rw- \WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2180_x-w
-- More  --
```

Zeus/Zbot artefacts



Mandiant Redline™ - C:\Infigo\AnalysisSession.mans
Home ▸ Processes ▸ Memoryze.exe (4340) ▸
Investigative Steps
Review Processes by MRI Scores
Review Network Ports / Connections
Review Memory Sections / DLLs
Review Untrusted Handles
Review Hooks
Review Drivers and Devices
Processes
Host
IOC Reports
Malware Risk Index Report
Export Report >
Add Comment or Hit >
Acquire Process Address Space >
Trust Process Address Space >
Negative Factors - 27
Positive Factors - 17
Ignored Factors - 0
All - 44

windbg.exe (2776)
windbg.exe (2012)
svchost.exe (884)
svchost.exe (912)
lsass.exe (500)
spoolsv.exe (1372)
notepad.exe (884)
notepad.exe (2476)
rdpclip.exe (2472)
notepad.exe (2516)
Dwm.exe (1240)
svchost.exe (844)
conhost.exe (2140)
conhost.exe (3628)
svchost.exe (1268)
svchost.exe (1884)
wininit.exe (372)
winlogon.exe (416)
cmd.exe (3620)
Memoryze.exe (4340)
svchost.exe (1512)
services.exe (484)
lsm.exe (508)
svchost.exe (816)
svchost.exe (736)
VBoxService.exe (872)
winlogon.exe (5468)
calc.exe (596)
LogonUI.exe (4992)
conhost.exe (4344)

Start Time: 9/16/2012 9:45:51 PM
Kernel Time Elapsed: 00:00:00
User Time Elapsed: 00:00:00
SID: S-1-5-21-3645329311-291908301-2031738713-1001
SID Type:
Malware Risk Index: 47

Malware Risk Index Hits

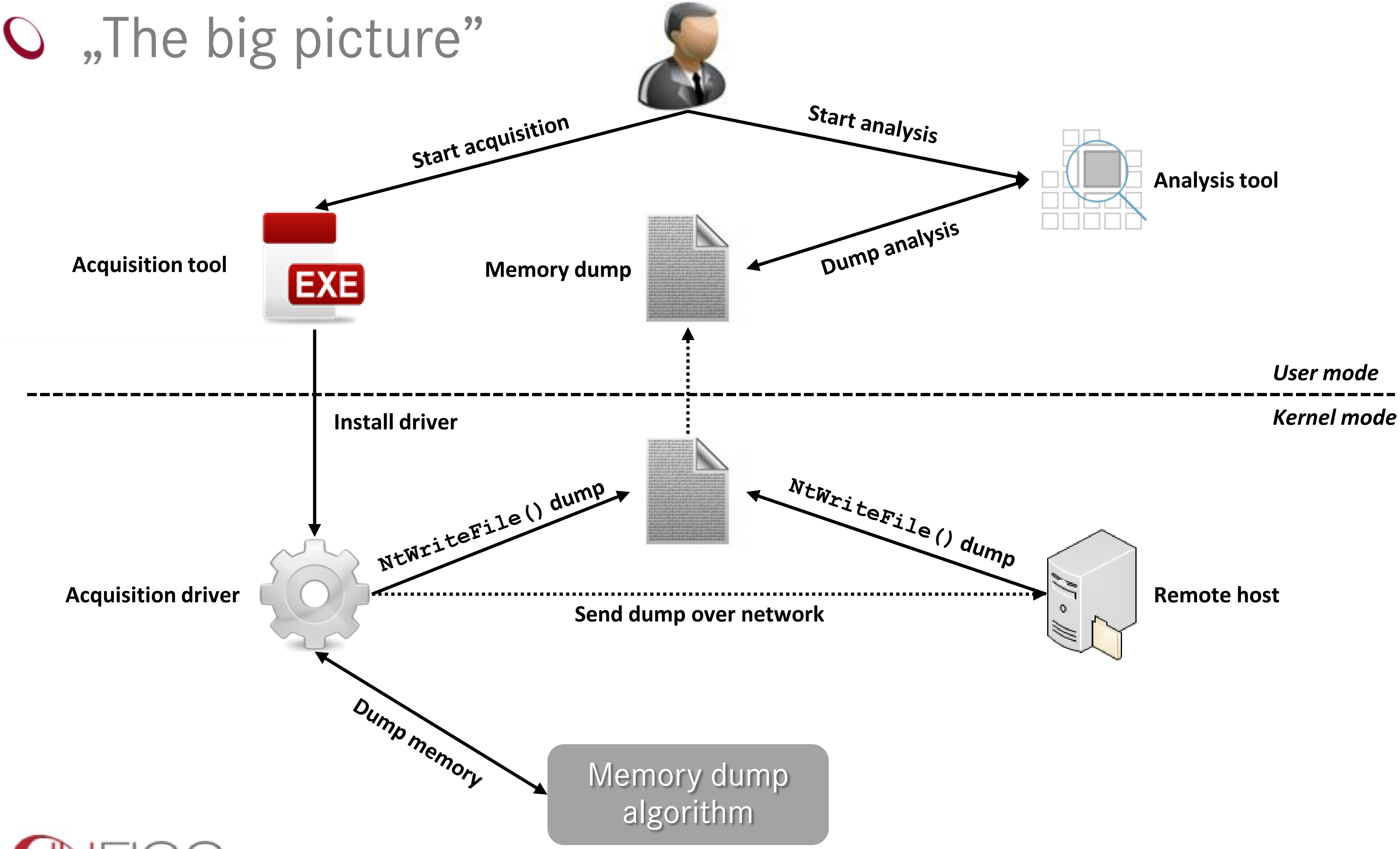
Named Memory Sections

🟥	Negative Factors	61%
🟦	Positive Factors	39%
■	Ignored Factors	0%

Reason	Count	Name	Action
No Digital Signature	2	\Device\HarddiskVolume2\Windows\winsxs\x86_microsoft.vc90.crt_1fc8b3b9a1e18e3b_9.0.30729.4926_non...	
No Digital Signature	1	\Device\HarddiskVolume2\Users\Demo\Desktop\Dementia\dbghelp.dll	
No Digital Signature	2	\Device\HarddiskVolume2\Windows\winsxs\x86_microsoft.vc90.crt_1fc8b3b9a1e18e3b_9.0.30729.4926_non...	
No Digital Signature	1	\Device\HarddiskVolume2\Users\Demo\Desktop\Dementia\MemoryzeUM.dll	
No Digital Signature	14	\Device\HarddiskVolume2\Windows\System32\netutils.dll	
No Digital Signature	12	\Device\HarddiskVolume2\Windows\System32\wkscli.dll	

Memory forensics – how? (5)

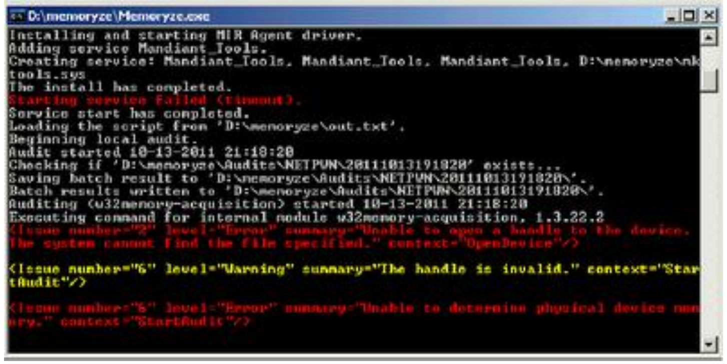
- „The big picture”



1. Reading `\\Device\\PhysicalMemory`
2. Physical space mapping (`MmMapIoSpace()`)
3. Other (for example `MmMapMemoryDumpMdl()`)

Previous works - simple

- Blocking acquisition
  - Killing memory acquisition tool process
    - tools always have the same names
  - Blocking driver installation
    - names (usually) not random
  - Metasploit script
    - not available anymore
- Evasion very simple
  - Rename process
  - Rename driver
    - not that easy if you don’t have the source



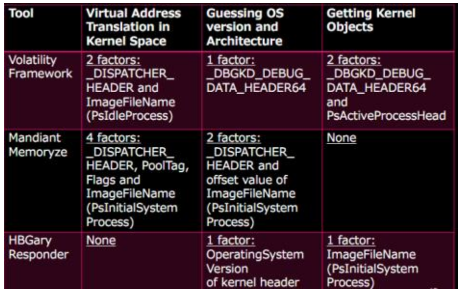
```
Installing and starting MIR Agent driver.
Adding service Mandiant_Tools.
Creating service: Mandiant_Tools, Mandiant_Tools, Mandiant_Tools, D:\memoryze\mk
tools.sys
The install has completed.
Starting service Failed (timeout).
Service start has completed.
Loading the script from 'D:\memoryze\out.txt'.
Beginning local audit.
Audit started 10-13-2011 21:18:20
Checking if 'D:\memoryze\Audits\NETPWN\20111013191820' exists...
Saving batch result to 'D:\memoryze\Audits\NETPWN\20111013191820\'.
Batch results written to 'D:\memoryze\Audits\NETPWN\20111013191820\'.
Auditing (w32memory-acquisition) started 10-13-2011 21:18:20
Executing command for internal module w32memory-acquisition, 1.3.22.2
<Issue number="2" level="Error" summary="Unable to open a handle to the device.
The system cannot find the file specified." context="OpenDevice"/>

<Issue number="6" level="Warning" summary="The handle is invalid." context="Star
tAudit"/>

<Issue number="5" level="Error" summary="Unable to determine physical device mem
ory." context="StartAudit"/>
```

Previous works – advanced

- Blocking analysis
  - Haruyama/Suzuki BH-EU-12: *One-byte Modification for Breaking Memory Forensic Analysis*
    - minimal modifications to OS artifacts in memory
    - targets key steps of analysis to make it impossible/difficult
      - so-called *abort factors*
      - tool specific

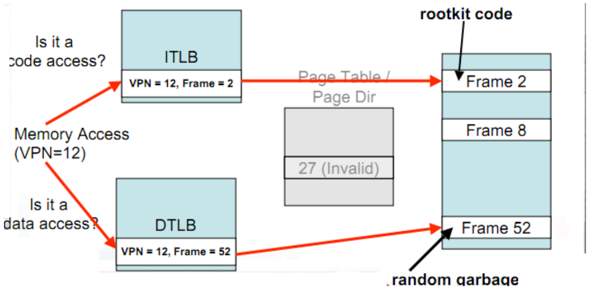


Tool	Virtual Address Translation in Kernel Space	Guessing OS version and Architecture	Getting Kernel Objects
Volatility Framework	2 factors: _DISPATCHER_HEADER and ImageFileName (PsIdleProcess)	1 factor: _DBGKD_DEBUG_DATA_HEADER64	2 factors: _DBGKD_DEBUG_DATA_HEADER64 and PsActiveProcessHead
Mandiant Memoryze	4 factors: _DISPATCHER_HEADER, PoolTag, Flags and ImageFileName (PsInitialSystem Process)	2 factors: _DISPATCHER_HEADER and offset value of ImageFileName (PsInitialSystem Process)	None
HBGary Responder	None	1 factor: OperatingSystem Version of kernel header	1 factor: ImageFileName (PsInitialSystem Process)

  - Pros:
    - subtle modifications (harder detection)
  - Cons:
    - cannot hide arbitrary object (could theoretically)
    - breaks entire (or big part of) analysis – can raise suspicion

Previous works – advanced (2)

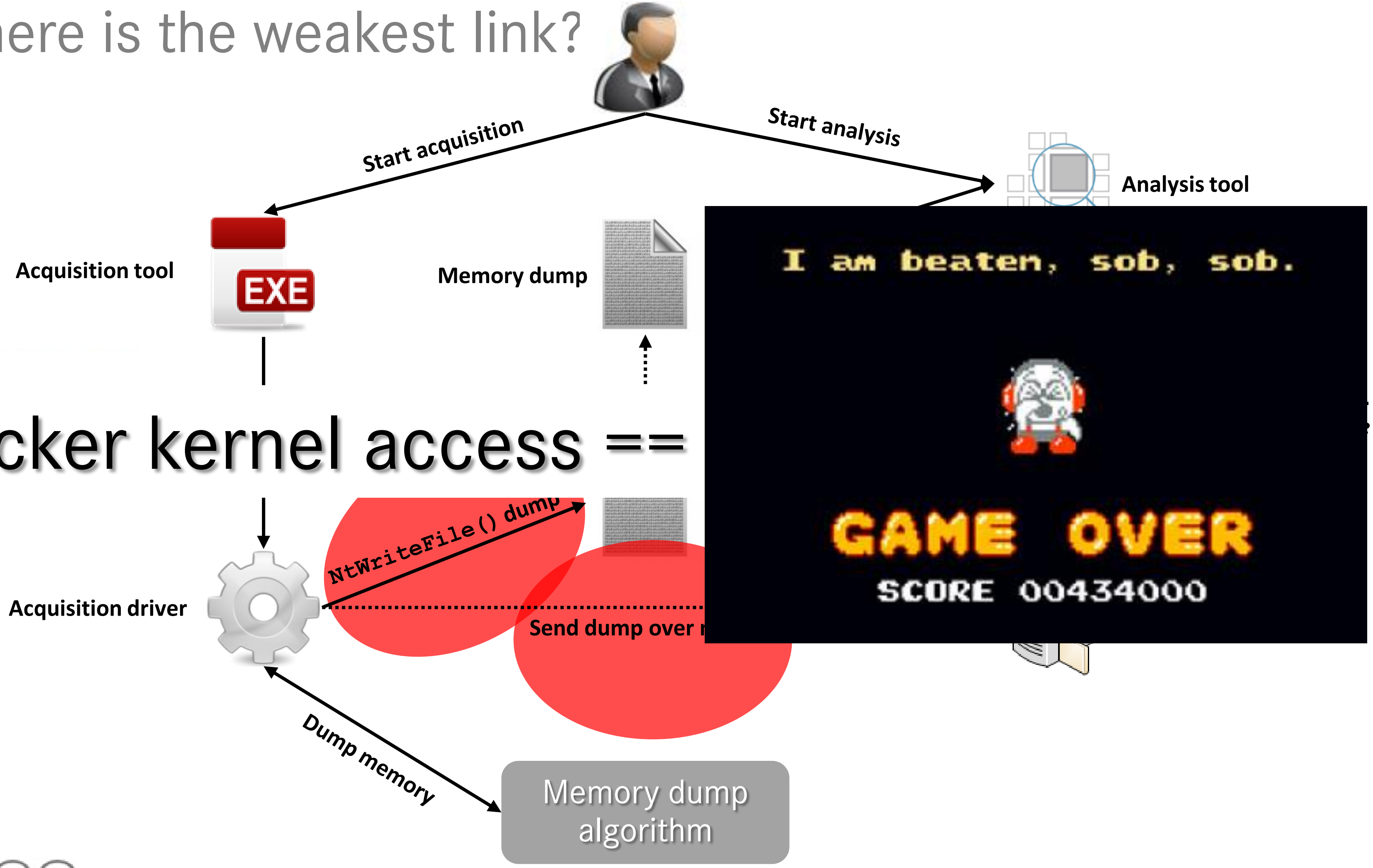
- Attacking acquisition & analysis
  - Sparks/Butler BH-JP-05: *Shadow Walker – Raising the bar for Rootkit Detection*
    - custom page fault handler
    - intentional desynchronization of ITLB/DTLB
    - faking reads of and writes to „arbitrary” memory location
      - execute access not faked



  - Pros:
    - awesome idea:)
    - hides (almost) arbitrary objects
  - Cons:
    - not very stable (and no MP/HT support)
    - page fault handler visible (code and IDT hook)
    - performance

Memory acquisition – flawed by design?

- Where is the weakest link?



Attacker kernel access ==

1. Reading \\Device\\PhysicalMemory
2. Physical space mapping (MmMapIoSpace())
3. Other (for example MmMapMemoryDumpMdl())

Sounds familiar?

- Of course it does, it’s an old technique!
  - Darren Bilby – DDefy rootkit (BH-JP-06)
    - disk filter driver – faking disk reads
    - faking physical memory device reads/mappings
- This is a „mapping” of disk anti-forensics to memory anti-forensics
  - evolution, not revolution

Defeating Windows memory forensics

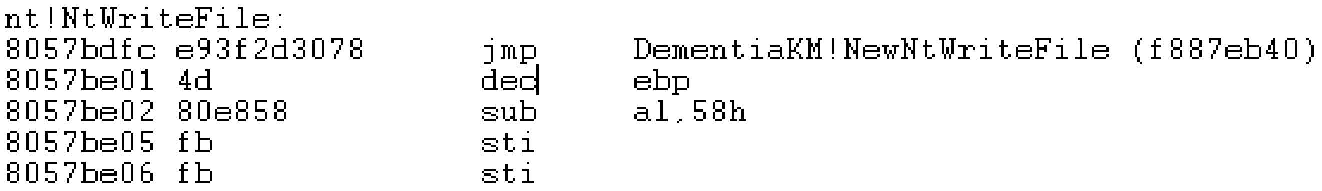
"Senile dementia's not 'so bad,' Mrs. Dupont.
It's kind of like having brand new friends every day."

Val Jones, founder of http://www.getbetterhealth.com

- Introducing **Dementia**
- PoC tool for hiding objects in memory dumps
- User mode components and kernel mode components
- Tested on Windows XP, Vista and Windows 7
- Three hiding methods
  - user mode injection
  - 2 different (but very similar) kernel methods
- All methods work on 32-bit systems
  - user mode works on 64-bit systems
  - Experimental driver support on 64-bit
    - read: **it will BSOD for sure!**

Dementia – How?

- Intercepting `NtWriteFile()` calls
  - Two methods
    - inline hook
      - stable even on multi(core)processor systems, but ask Don Burn and Raymond Chen about it☺
    - filesystem minifilter
      - preferred method of write-interception
      - from a blackhat perspective – maybe too noisy, IRP hooks would suit better☺
      - hooking is a no-no in x64 kernels so this is the way to go



```
nt!NtWriteFile:
8057bdfc e93f2d3078       jmp     DementiaKM!NewNtWriteFile (f887eb40)
8057be01 4d               dec     ebp
8057be02 80e858           sub     al,58h
8057be05 fb               sti
8057be06 fb               sti
```

Dementia – Detecting forensic app?

- OK, we have the „hook” in place, but what now?
  - Is the file being written a memory dump?
- Memory acquisition tools have „patterns”
  - Specific `NtWriteFile()` arguments
  - Context (i.e. process, driver, ...)
  - Specific `FILE_OBJECT` values and flags

Tool	Handle	Event	ApcRoutine	ApcContext	IO	Buffer	Length	Offset	Key	Add. flags	Process	Ext.	Driver	FILE_OBJECT flags
FTK Imager	UM	NULL	NULL	NULL	UM	UM	0x8000	0	NULL	W,SR,SW	FTK Imager.exe	mem	ad_driver.sys	0x40042
MDD	UM	NULL	NULL	NULL	UM	UM	0x1000	0	NULL	W	mdd_1.3.exe	*	mdd.sys	0x40042
Memoryze	UM	NULL	NULL	NULL	UM	UM	mostly 0x1000	0	NULL	W,SR,SW	Memoryze.exe	img	mktools.sys	0x40042
OSForensics	KM	NULL	NULL	NULL	KM	UM	0x1000	KM	NULL	W	osf32.exe	bin	DirectIo32	0x40062
Win32DD	KM	NULL	NULL	NULL	KM	KM	variable (0x1000 - 0x100000)	KM	NULL	R,W,SR,SW	win32dd.exe	*	win32dd.sys	0x4000a
Winen (EnCase)	UM	NULL	NULL	NULL	UM	UM	totally variable	0	NULL	R,W,SR,SW	winen.exe	E01	winen_.sys	0x40062
Winpmem	UM	NULL	NULL	NULL	UM	UM	0x1000	0	NULL	W,SR	winpmem_...*	*	*(temporary file - random)	0x40042

These will be important later

Dementia – Hiding?

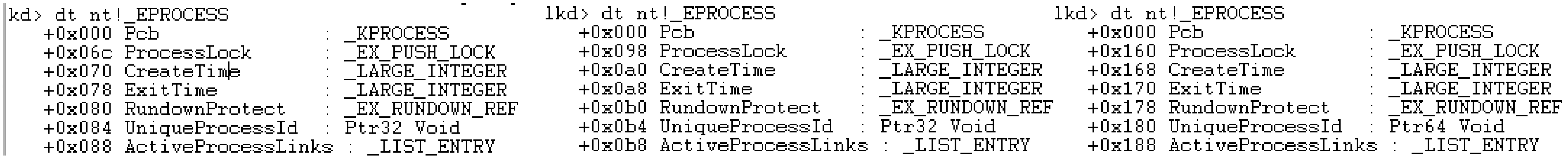
- Hook installed and memory dump detected - what’s next?
- Memory is read and written to image in pages or page-multiples
- Wait and scan every buffer being written for our target objects (i.e. allocations)?
  - OK, but slow and inefficient
- Solution
  - Build a (sorted) list of all (physical) addresses somehow related to our target objects
    - if the buffer being written contains those addresses – hide them (change or delete)

INFIGO
Information security

Dementia – Hiding? (2)

- That sounds fine…
- .. but we’re dealing with undocumented kernel structures, functions, sizes and offsets

Win XP x86 Win 7 x86 Win 7 x64



```
kd> dt 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
```

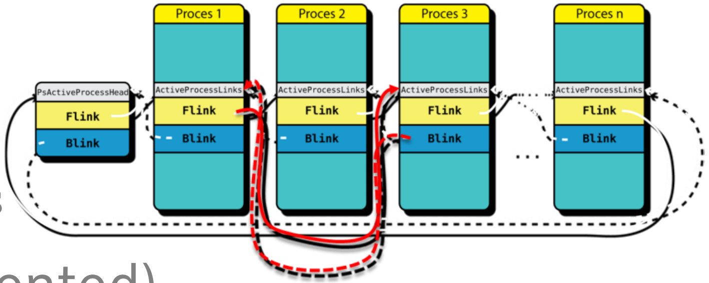
```
lkd> dt nt!_EPROCESS
   +0x000 Pcb                  : _KPROCESS
   +0x098 ProcessLock          : _EX_PUSH_LOCK
   +0x0a0 CreateTime           : _LARGE_INTEGER
   +0x0a8 ExitTime             : _LARGE_INTEGER
   +0x0b0 RundownProtect       : _EX_RUNDOWN_REF
   +0x0b4 UniqueProcessId      : Ptr32 Void
   +0x0b8 ActiveProcessLinks : _LIST_ENTRY
```

```
lkd> dt nt!_EPROCESS
   +0x000 Pcb                  : _KPROCESS
   +0x160 ProcessLock          : _EX_PUSH_LOCK
   +0x168 CreateTime           : _LARGE_INTEGER
   +0x170 ExitTime             : _LARGE_INTEGER
   +0x178 RundownProtect       : _EX_RUNDOWN_REF
   +0x180 UniqueProcessId      : Ptr64 Void
   +0x188 ActiveProcessLinks : _LIST_ENTRY
```

- If WinDBG can do it, we can do it too!
  - Use Microsoft PDB symbols and DbgHelp API
  - Kernel sends the list of needed symbols
  - UM fills the gaps – addresses, offsets and sizes

Dementia – Hiding Processes

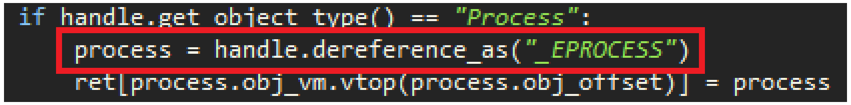
- Get the target process `EPROCESS` block
- „Unlink” the process from the various process lists
  - `ActiveProcessLinks`
  - `SessionProcessLinks`
  - Job list (not yet implemented)



- Clear the entire „*Proc*” allocation
  - Remember, we’re doing it in the dump only
- Hide related data
  - Threads, handles, memory allocations (VADs), etc.

Dementia – Hiding Processes (2)

- Hiding processes is deceptively simple
- However, traces of process activity are everywhere and difficult to remove completely!
  - will see some artifacts in the next couple of slides
- **Volatility note:** deleting just the „*Proc*” allocation will fool most of the plugins (*psscan*, even *psxview*!)



```
if handle.get_object_type() == "Process":
    process = handle.dereference_as("_EPROCESS")
    ret[process.obj_vm.vtop(process.obj_offset)] = process
```

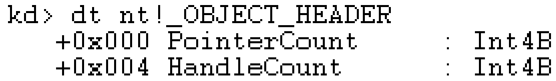
  - don’t rely on `EPROCESS` block existance and validity – maybe better to show it as-is

Dementia – Hiding Threads

- All threads of target process are hidden
  - Clear „*Thre*” allocations
  - Remove thread handle from `PspCidTable`
- It is still possible to detect „unusual entries”
  - Hanging thread locks, various lists (`PostBlockList`, `AlpcWaitListEntry`, ...) etc.
- No analysis application will detect these threads

Dementia – Hiding Handles and Objects

- Rather deep cleansing
- Hide process handle table
  - Unlink it from the `HandleTableList` and delete the „*Obtb*” allocation
- Hide process-exclusive handles/objects
  - Handles to objects opened exclusively by the target process (counts == 1)

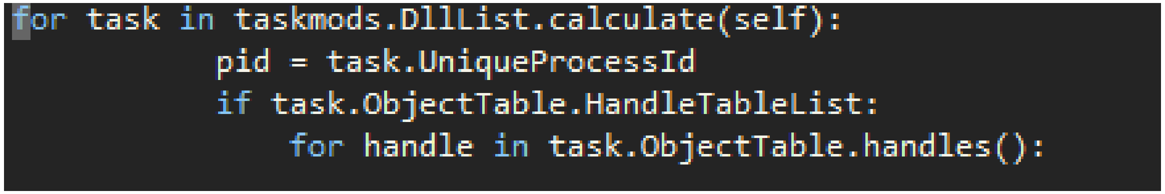


```
kd> dt nt!_OBJECT_HEADER
   +0x000 PointerCount      : Int4B
   +0x004 HandleCount       : Int4B
```

  - Hide the `HANDLE_TABLE_ENTRY` and the object itself
- Decrement the count for all other handles/objects
  - And hide the `HANDLE_TABLE_ENTRY`

Dementia – Hiding Handles and Objects (2)

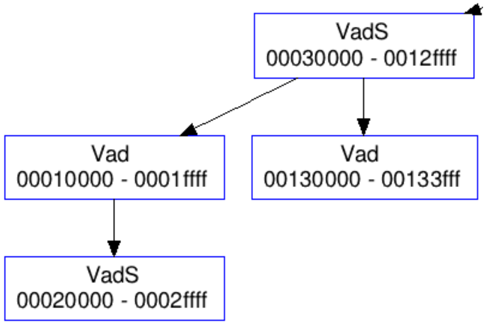
- Wait, there is more!
  - `PspCidTable` and csrss.exe handle table contain handle to our target process
    - find the target handle and remove it from the table
- Handle hiding can be difficult
  - **Volatility note:** don't enumerate the handles starting from the `EPROCESS` and using the `HandleTableList` – scan for „*Obtb*" allocations!



```
for task in taskmods.DllList.calculate(self):
            pid = task.UniqueProcessId
            if task.ObjectTable.HandleTableList:
                for handle in task.ObjectTable.handles():
```

Dementia – Hiding Memory Allocations

- All process memory allocations are described by VADs – *Virtual Address Descriptors*
- VADs are stored in an AVL tree
  - Root of the tree is in `VadRoot` in `EPROCESS`
- Hide algorithm
  - Traverse the tree
  - Hide the „*VadX*” allocation (X == -,S or M)
  - If VAD describes private memory || VAD describes process image (EXE)
    - clear the entire memory region
  - If VAD describes shared section
    - check if opened exclusively – clear if yes, along with potential mapped files (i.e. `FILE_OBJECTS`)



VadS
00030000 - 0012ffff

Vad
00010000 - 0001ffff

Vad
00130000 - 00133fff

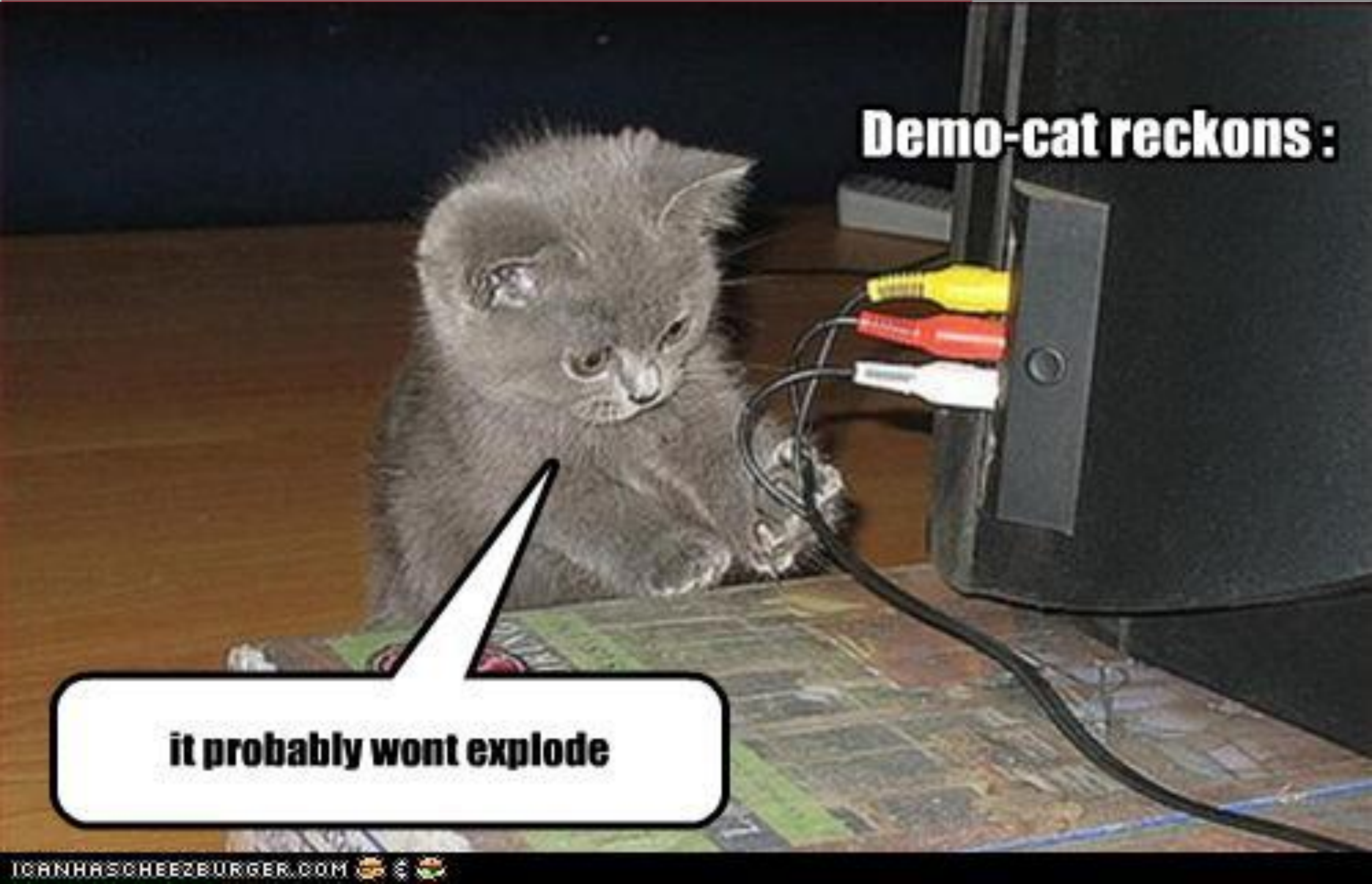
VadS
00020000 - 0002ffff

Dementia – Hiding Drivers

- Apart from the process hiding, drivers can be hidden too
  - Unlink from the `PsLoadedModuleList`
  - Delete the `LDR_DATA_TABLE_ENTRY` allocation („*MmLd*”)
  - Clear the driver image from the memory
- Rudimentary, but effective
- Needs improvement
  - Kernel allocations, symlinks, …

INFIGO Information security

Finally!


Demo-cat reckons :
it probably wont explode
ICANHASCHEEZBURGER.COM

You’re doing it wrong!

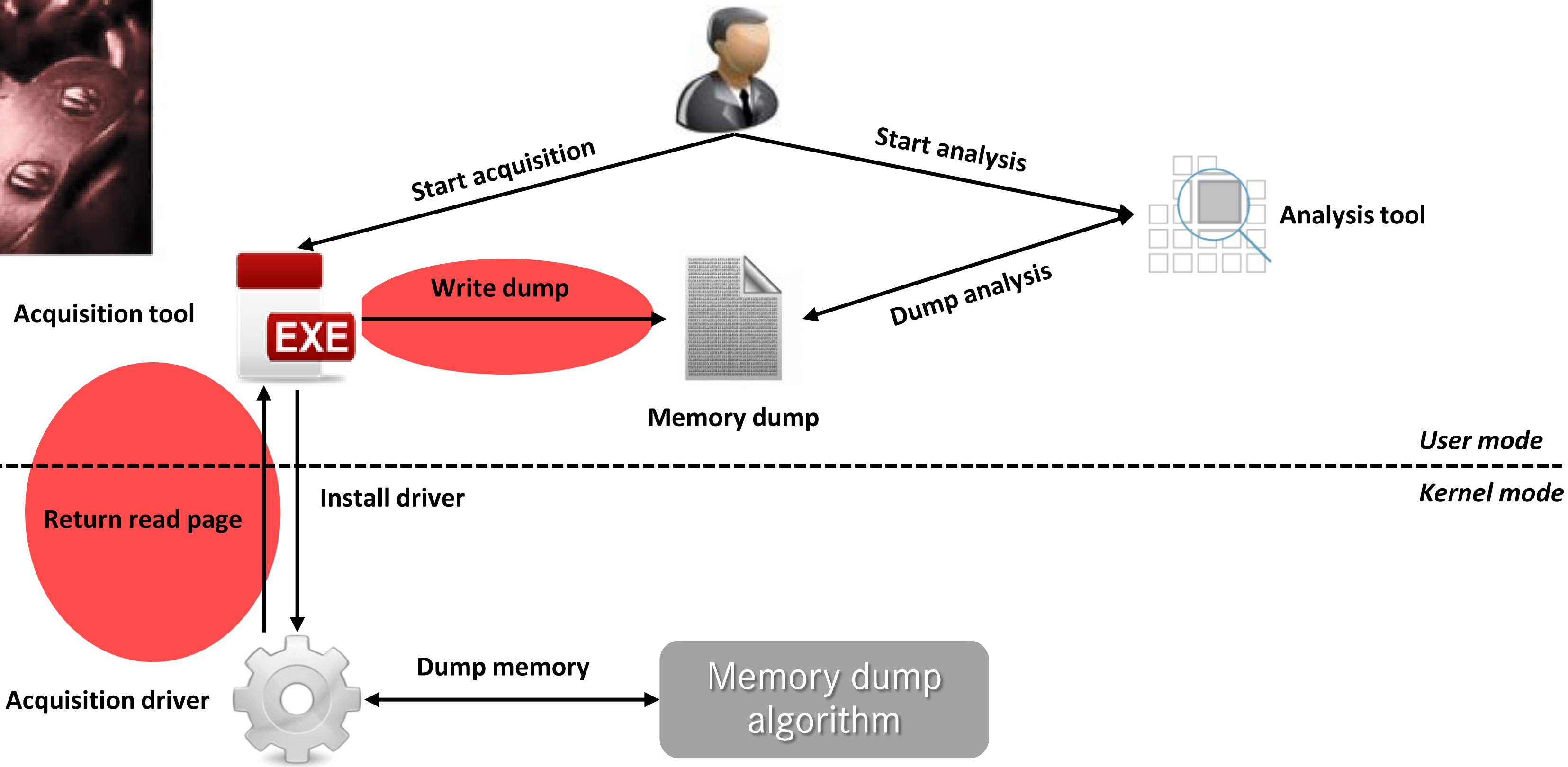
- Remember these columns?

Tool	Handle	Buffer
FTK Imager	UM	UM
MDD	UM	UM
Memoryze	UM	UM
OSForensics	KM	UM
Win32DD	KM	KM
Winen (EnCase)	UM	UM
Winpmem	UM	UM

- Handle == UM
  - Memory dump file opened in user mode
    - vulnerable to `WriteFile()/NtWriteFile()` hooks in user mode
- Buffer == UM
  - Buffer passed back to user mode (usually coupled with Handle == UM)
    - vulnerable to `DeviceIoControl()/NtDeviceIoControlFile()` hooks

You’re doing it wrong! (2)

- Almost all tools are doing it wrong!



1. Reading `\\Device\\PhysicalMemory`
2. Physical space mapping (`MmMapIoSpace()`)
3. Other (for example `MmMapMemoryDumpMdl()`)

So what?

- Attacker can now modify dump from the user mode☺
- Dementia module
  - Hiding target process, process threads and connections
    - completely from the user mode, no driver used
    - need to be admin unfortunately (because acquisition app runs as admin)
  - Injects DLL to forensic app process
    - currently only Memoryze, but extensions are easy
  - Hooks `DeviceIoControl()` and sanitizes buffers on the fly

Dementia user mode - internals

- Sounds simpler than the kernel mode
  - Actually, it is much harder!
    - no knowledge of kernel addresses
    - no V2P translation, determine everything from the dump
    - partial knowledge - only single pages of the dump
  - Search the current buffer for interesting allocations (processes, threads, connections)
    - if target object encountered – delete the allocation
    - if object related to a target object (thread, connection) – delete the allocation
  - So far so good…

Dementia user mode – internals (2)

- What about the process/thread list unlinking?
  - Difficult part
    - don’t know where next/prev object is, just their (kernel) virtual address
    - what if that object was already written to file – we can’t easily reach that buffer anymore☹
  - Solution
    - determine virtual address of the object using self-referencing struct members (for example, `ProfileListHead`)
    - „cache” the object in a dictionary with VA as the key, and remember the physical offset of that buffer in the dump
    - fix the next/prev pointers either in the current buffer, or move the file pointer, write new value and restore the file pointer

Demo again!

Dementia limitations

- Focus on kernel module
  - Plenty of other artifacts not hidden
    - connections
    - registry keys and values
    - arbitrary DLLs
  - Improve driver hiding functionality
  - Self-hiding
    - it’s useless in your rootkit arsenal without it☺
  - Complete port to x64
  - Work in progress!
- No motives for user mode module, probably won’t update

Conclusions & possible solutions

- Acquisition tools should utilize drivers correctly!
  - Current method is both insecure and slow!
- Use hardware acquisition tools
  - Firewire -what about servers?
- Use crash dumps (native!) instead of raw dumps
  - Entirely different OS mechanisms, difficult to tamper with
- Perform anti-rootkit scanning before acquisition?
- Live with it
  - Live forensic is inherently insecure!

Thank you!

http://code.google.com/p/dementia-forensics/

INFIGO
Information security