

Memory Forensics

Part I

Golden G. Richard III

“Prerequisites”

- For memory forensics, strong skills in all of these areas
- "Learn on the job" as necessary

**Traditional
Digital
Forensics**

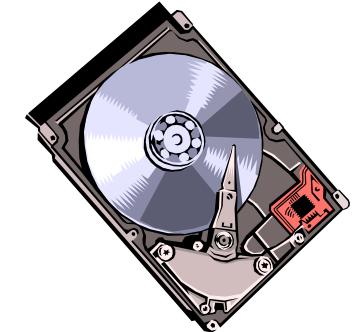
**OS
Internals**

**Data
Structures**

C

Assembler

Python 😊



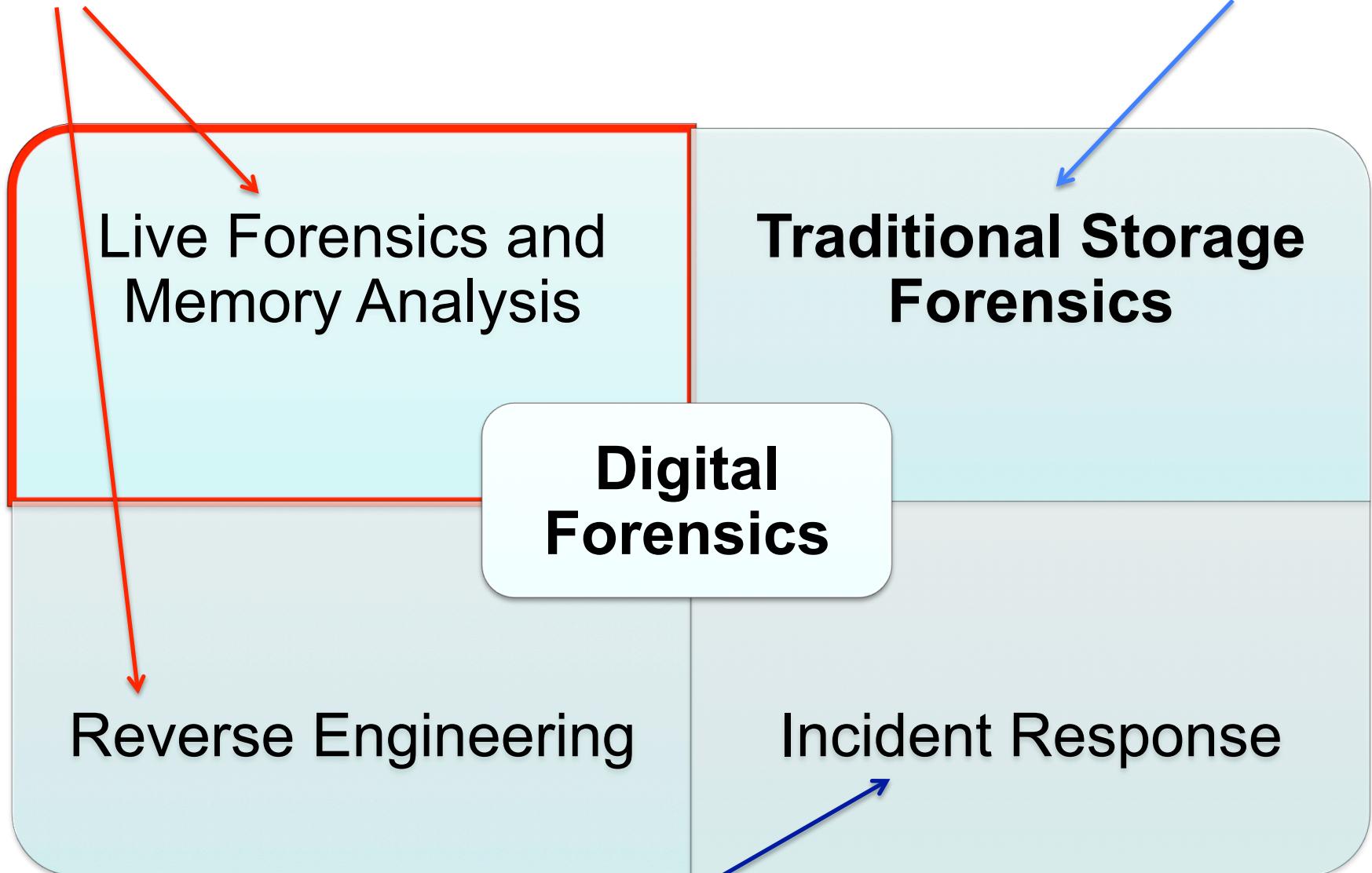
Definition: “Tools and techniques to recover, preserve, and examine digital evidence stored on or transmitted by digital devices.”

Computers, PDAs, cell phones, videogame consoles, digital cameras, copy machines, printers, digital voice recorders...



hot research areas

State of Affairs



Where's the Evidence?

Files and
Deleted Files

Filesystem
metadata

Application
metadata

Windows
registry

Print spool
files

Hibernation
files

Temp files

Log files

Slack space

Swap files

Browser
caches

Network
traces

RAM: OS
and app data
structures

Some Data is Only in RAM

Clipboard data

Volatile registry branches

Network connections

Running processes

Open files

Encryption keys

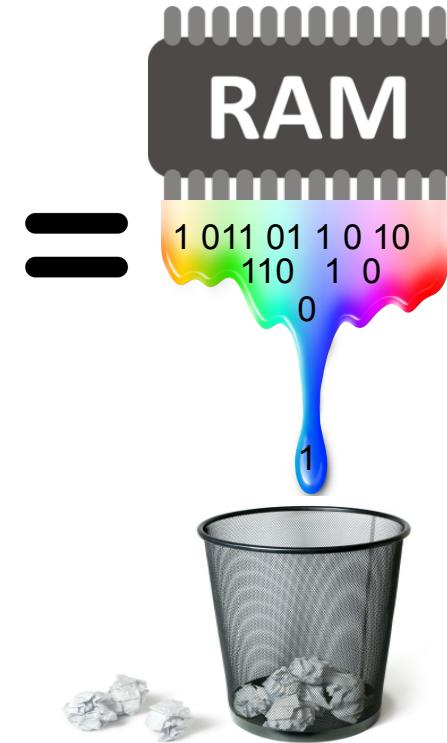
Private browsing data

Kernel structures

Application structures

...

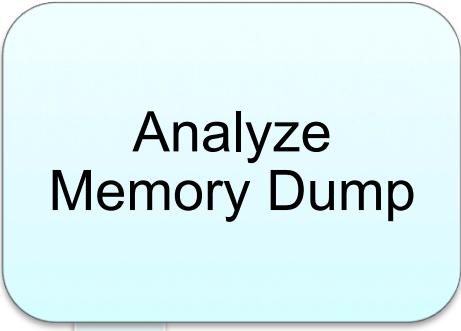
...



Memory Analysis

Capture RAM
from live
system

- physical memory dumping tool
- VM memory snapshot
- VM introspection



Analyze
Memory Dump

- strings
- carving
- Volatility
- VM introspection



Expose OS and
Application
Structures

- to yield useful
evidence

Physical Memory Dumps

- Essentially, a complete copy of RAM
- A collection of pages (4K on Intel)
- Must impose order—otherwise, essentially a random set of 4K chunks

Memory Analysis / Live Forensics

- Live forensics
 - Analysis on the box (while live)
 - Run commands (e.g., ps)
 - Record results
- Memory Analysis aka Memory Forensics
 - Capture memory dump
 - Analyze using:
 - List following techniques
 - Carving techniques
 - Hybrid approaches
- Sometimes both

Memory Analysis: Then

- strings (since dinosaurs roamed)
 - seriously—essentially no tools, circa 2004
- pt_finder (~2006)
 - Windows process enumeration
- FACE (~2008)
 - Memory analysis framework created at UNO
- ...

Memory Analysis: 2004

```
$ grep -i murder /dev/mem
```



I loved Sally, but I murdered her in
the park on...

Murder

murder

Murderous

You murdered my hamster!

Murdered

pt_finder: Andreas Schuster's Approach

- Scan memory to find kernel structures describing processes and threads
 - Normal
 - Hidden
 - **Terminated**
- Don't rely on kernel lists/tables
- Instead, search memory dump for objects that **look like** processes/threads

Schuster (2)

- Important ideas:
 - Memory is needed to store kernel objects
 - Use info about how kernel performs allocation to find blocks of allocated memory
 - Kernel objects have an OBJECT_HEADER structure
 - Further, processes and threads have a DISPATCH_HEADER, used for scheduling/synchronization
- Use these ideas to develop templates for discovering interesting structures in a Windows memory dump
- Walk memory dump in 4K steps

Schuster: POOL_TAG

```
kd> dt _POOL_HEADER          kd> dt _POOL_HEADER
→ +0x000 PreviousSize : UChar +0x000 PreviousSize : Pos 0 , 9 Bits
   +0x001 PoolIndex      : UChar +0x000 PoolIndex      : Pos 9 , 7 Bits
   +0x002 PoolType       : UChar +0x002 BlockSize      : Pos 0 , 9 Bits
→ +0x003 BlockSize       : UChar +0x002 PoolType       : Pos 9 , 7 Bits
   +0x004 PoolTag        : Uint4B +0x004 PoolTag        : Uint4B
```

Fig. 1 – Definitions of the `POOL_HEADER` structure in Windows 2000 (left) and later versions (right).

PoolTag == 0xe36f7250 for processes
PoolTag == 0xe5726854 for threads

Schuster: OBJECT_HEADER

Known values
for live/dead
processes and threads!

```
kd> dt _OBJECT_HEADER
+0x000 PointerCount      : Int4B
+0x004 HandleCount       : Int4B
+0x004 SEntry             : Ptr32
→ +0x008 Type              : Ptr32
+0x00c NameInfoOffset     : UChar
+0x00d HandleInfoOffset   : UChar
+0x00e QuotaInfoOffset    : UChar
+0x00f Flags               : UChar
+0x010 ObjectCreateInfo    : Ptr32
+0x010 QuotaBlockCharged   : Ptr32
+0x014 SecurityDescriptor : Ptr32
```

Fig. 2 – The OBJECT_HEADER structure provides information about an object's instance.

Also know information about lengths associated with name.

Schuster: Additional Tests

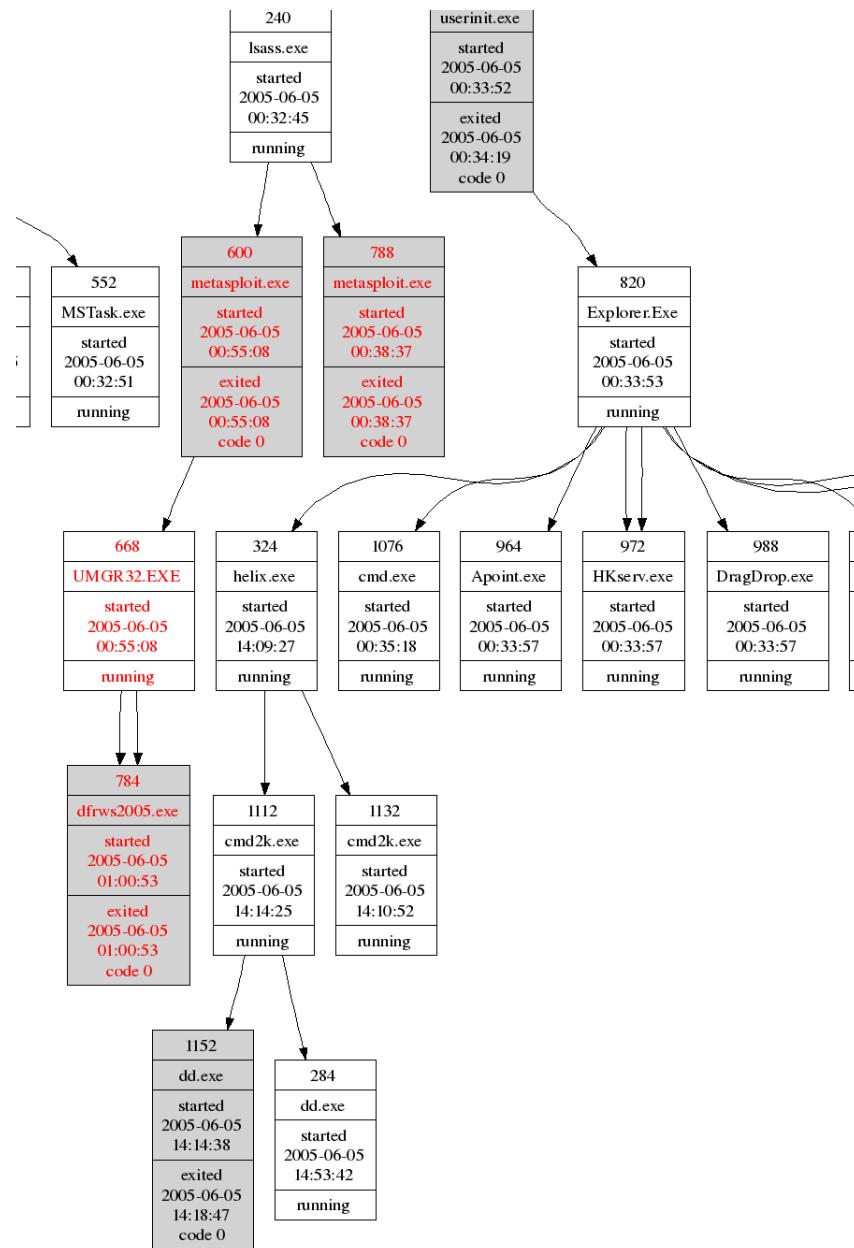
- Know some characteristics of DISPATCH_HEADER
- Know some characteristics of ETHREAD structures (e.g., pointers to owning process, DISPATCH_HEADER w/ type thread, ...)

For a certain Windows version, size field is constant for a particular object type.

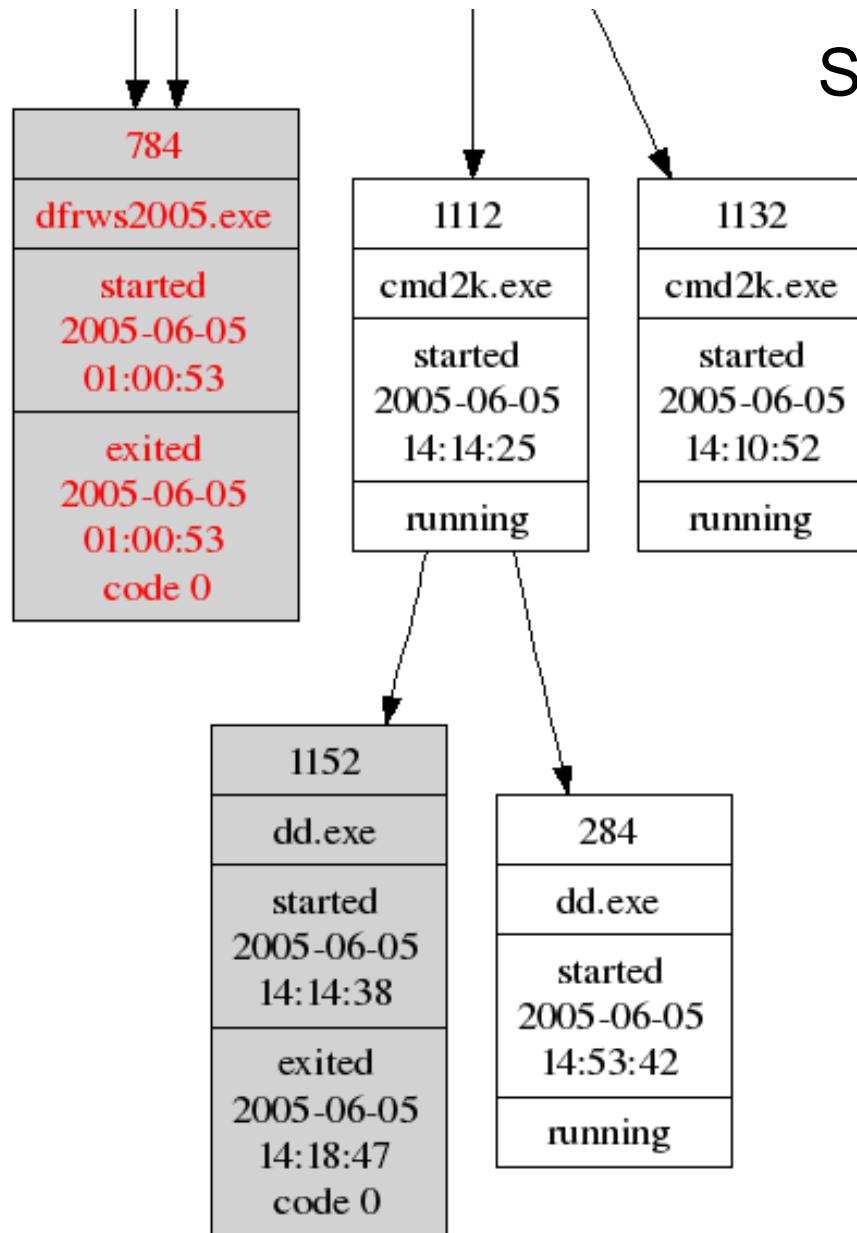
Schuster: Results

- Perl-based PTFinder
- Visualization using Graphviz

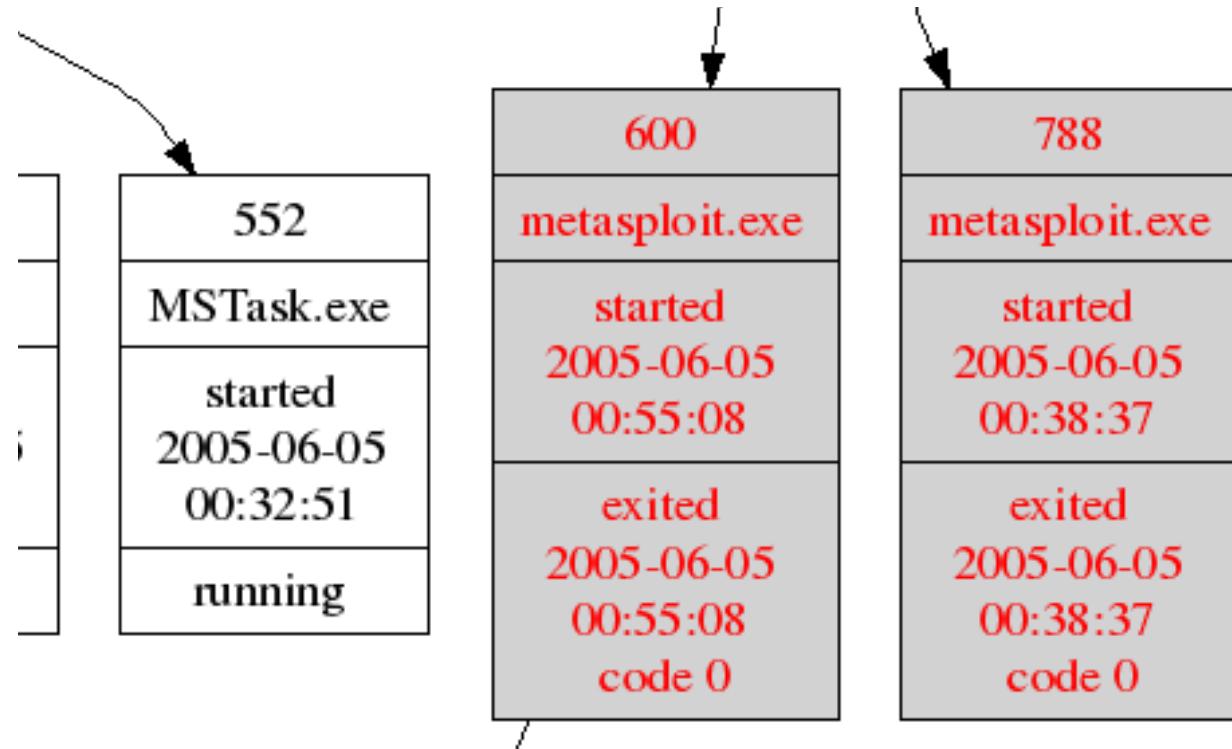




Schuster's PTfinder



Schuster's PTfinder



FACE: Linux Memory Analysis

- Andrew Case, Andrew Cristina, Lodovico Marziale, Golden G. Richard III, and Vassil Roussev. "**FACE: Automated Digital Evidence Discovery and Correlation.**" *Digital Investigation* 5 (2008): S65-S75. Deep kernel structures analysis
- Correlation between network traces, filesystem, log files, and physical memory dumps
- A step closer to frameworks, rather than "one off" tools

FACE: ps on Memory Dump

```
# ./ramparser debian.vmem -x
```

PID	UID	GID	NAME
1	0	0	init
2	0	0	migration/0
3	0	0	ksoftirqd/0
4	0	0	watchdog/0
5	0	0	migration/1
6	0	0	ksoftirqd/1
...			
...			
2425	0	43	xterm
2426	0	0	bash
2515	0	0	pdflush
2521	0	0	firefox-bin
2548	0	0	ftp

FACE: netstat on Memory Dump

```
# ./ramparser debian.vmem -N | sort
```

Proto	Local Address	Foreign Address	State	PID	Program name
TCP	192.168.20.128:45351	192.168.20.129:20	ESTABLISHED	2548	ftp
TCP	192.168.20.128:55071	192.168.20.129:80	ESTABLISHED	2521	firefox-bin
TCP	192.168.20.128:59447	192.168.20.129:21	ESTABLISHED	2548	ftp
TCP	192.168.20.128:59447	192.168.20.129:21	ESTABLISHED	2548	ftp
TCP	192.168.20.128:59447	192.168.20.129:21	ESTABLISHED	2548	ftp
UDP	0.0.0.0:111	0.0.0.0:0		1959	portmap
UDP	0.0.0.0:32768	0.0.0.0:0		2332	rpc.statd
UDP	0.0.0.0:68	0.0.0.0:0		2301	dhclient3
UDP	0.0.0.0:812	0.0.0.0:0		2332	rpc.statd
		...			
		...			
		...			
UNIX	/tmp/.X11-unix/X0			2417	Xorg
UNIX	/tmp/.X11-unix/X0			2417	Xorg
UNIX	/tmp/.X11-unix/X0			2417	Xorg
		...			
		...			
		...			

FACE: Deep Process Analysis (1)

The screenshot shows a Mozilla Firefox window titled "ftp:2548 - Mozilla Firefox". The address bar displays "file:///root/ftp_screen_shot2548.html". The page content is a process analysis report for "Process: ftp" (PID: 2548). The report includes details about the process's code, data, stack, and heap sections, each with "Hexdump" and "Raw" links. It also lists the file descriptors (FD) and their corresponding file paths or socket descriptions.

[Home](#) [Users](#) [Groups](#) [Processes](#) [Packets](#)

Process: ftp

PID: 2548

UID: [root, 0](#)

GID: [root, 0](#)

- Code: [Hexdump](#) [Raw](#)
- Data: [Hexdump](#) [Raw](#)
- Stack: [Hexdump](#) [Raw](#)
- Heap: [Hexdump](#) [Raw](#)

• **Files:**

- FD: 0 [/pts/0](#)
- FD: 1 [/pts/0](#)
- FD: 2 [/pts/0](#)
- FD: 3 [socket:\[6513\]](#)
- FD: 4 [socket:\[6513\]](#)
- FD: 5 [socket:\[6513\]](#)
- FD: 6 [/root/file2](#)
- FD: 8 [socket:\[6515\]](#)

FACE: Deep Process Analysis (2)

- **Sockets:**

- Inode: [6513](#) **192.168.20.128:59447** to **192.168.20.129:21** Send Buffer: 0 entries, Recv Buffer: 0 entries.
- Inode: [6513](#) **192.168.20.128:59447** to **192.168.20.129:21** Send Buffer: 0 entries, Recv Buffer: 0 entries.
- Inode: [6513](#) **192.168.20.128:59447** to **192.168.20.129:21** Send Buffer: 0 entries, Recv Buffer: 0 entries.
- Inode: [6515](#) **192.168.20.128:45351** to **192.168.20.129:20** Send Buffer: 243 entries, Recv Buffer: 0 entries.

- **Mappings:**

/usr/bin/netkit-ftp	Code: Hexdump Raw Data: Hexdump Raw
Anonymous	Data: Hexdump Raw
/usr/lib/gconv/gconv-modules.cache	Data: Hexdump Raw
/usr/lib/locale/locale-archive	Data: Hexdump Raw
/lib/tls/i686/cmov/libnss_nis-2.3.6.so	Code: Hexdump Raw Data: Hexdump Raw
/lib/tls/i686/cmov/libnsl-2.3.6.so	Code: Hexdump Raw Data: Hexdump Raw
Anonymous	Data: Hexdump Raw
/lib/tls/i686/cmov/libnss_compat-2.3.6.so	Code: Hexdump Raw Data: Hexdump Raw
/lib/tls/i686/cmov/libnss_files-2.3.6.so	Code: Hexdump Raw Data: Hexdump Raw
Anonymous	Data: Hexdump Raw
/lib/tls/i686/cmov/libdl-2.3.6.so	Code: Hexdump Raw Data: Hexdump Raw
/lib/tls/i686/cmov/libc-2.3.6.so	Code: Hexdump Raw Data: Hexdump Raw
Anonymous	Data: Hexdump Raw
/lib/libncurses.so.5.5	Code: Hexdump Raw Data: Hexdump Raw
Anonymous	Data: Hexdump Raw
/lib/libreadline.so.5.2	Code: Hexdump Raw Data: Hexdump Raw
Anonymous	Data: Hexdump Raw
/lib/ld-2.3.6.so	Code: Hexdump Raw Data: Hexdump Raw

Done

FACE: Socket Buffer Dumps

```
# ./ramparser debian.vmem -k 2548
# ls socks
socket0      socket15     socket21     socket28
socket34     socket40     socket47     socket53
socket1      socket16     socket22     socket29
socket54     socket60     socket67     socket73
...
# head socks/socket28
aplacental
aplanatic
aplanospore
aplasia
aplastic
aplenty
aplite
aplomb
apnoea
```



/usr/dict/words was being transmitted

Volatility Framework

- Now, most work done with a physical memory dump tool + Volatility
- Most popular memory analysis framework
- Completely open source
- Portable, written in Python 😊
- Supports analysis of Windows, Linux, Mac, Android memory dumps
- Plugins add new functionality, build on vast amount of research that's already been done
- Lots of existing plugins, fairly straightforward to add new ones

Memory Analysis: Now

Use plugins to analyze:

Running processes

Hidden processes

Hooks that hide malware

Network connections

Encryption keys

Private browsing data

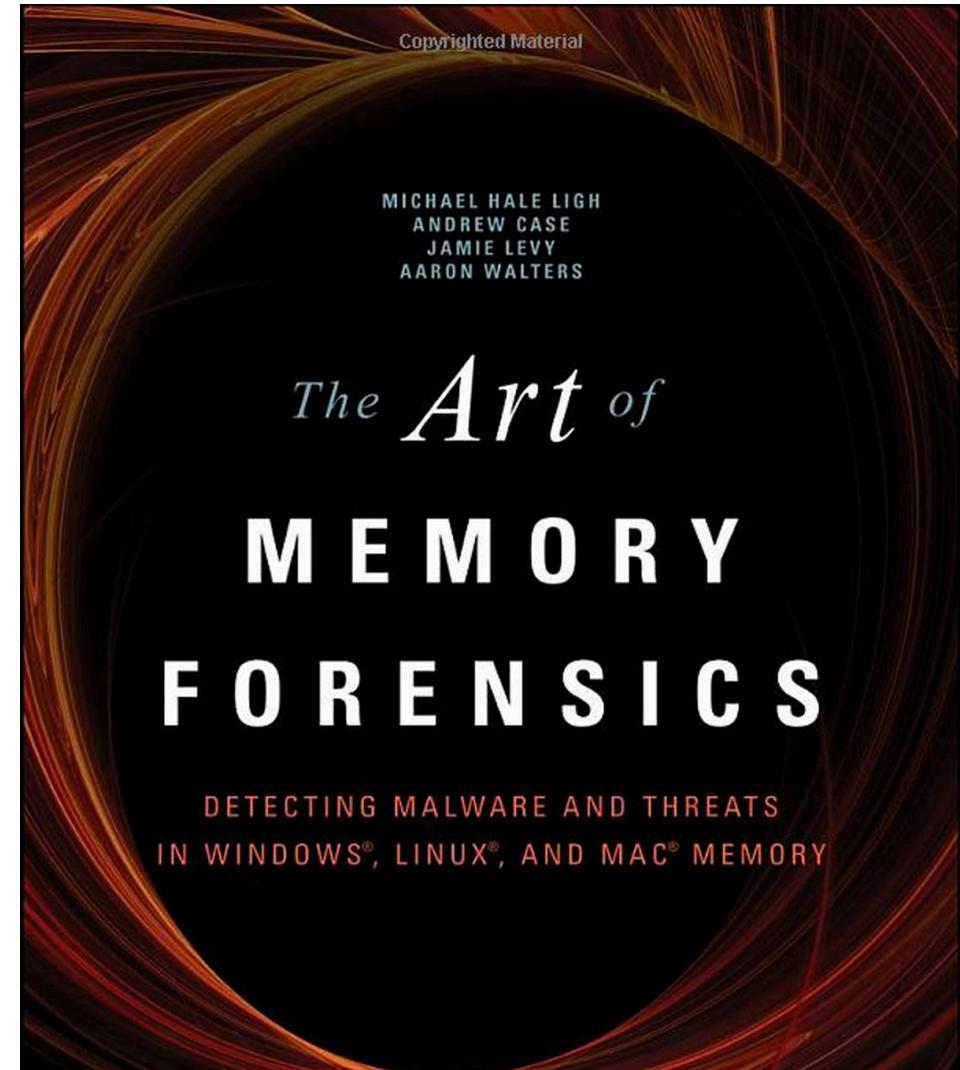
Clipboard data

Volatile registry branches

Command history

Window hierarchy

+ "easily" develop plugins



ps aux on Ubuntu 14.04

```

golden@ubuntu: ~
Software Updater
golden@ubuntu: ~

File Edit View Search Terminal Help
root      906  0.0  0.0  15656  912 ?          S    07:40   0:00 upstart-socket-bridge --daemon
root      957  0.0  0.0  23008  952 tty4        Ss+  07:40   0:00 /sbin/getty -8 38400 tty4
root      962  0.0  0.0  23008  944 tty5        Ss+  07:40   0:00 /sbin/getty -8 38400 tty5
root      970  0.0  0.0  23008  948 tty2        Ss+  07:40   0:00 /sbin/getty -8 38400 tty2
root      971  0.0  0.0  23008  952 tty3        Ss+  07:40   0:00 /sbin/getty -8 38400 tty3
root      974  0.0  0.0  23008  956 tty6        Ss+  07:40   0:00 /sbin/getty -8 38400 tty6
root     1018  0.0  0.1  61364  3044 ?          Ss    07:40   0:00 /usr/sbin/sshd -D
kernoops 1031  0.0  0.0  37144  1008 ?          Ss    07:40   0:00 /usr/sbin/kerneloops
root     1033  0.0  0.0    4368   696 ?          Ss    07:40   0:00 acpid -c /etc/acpi/events -s /var/run/acpid.socket
root     1035  0.0  0.0  23656  1008 ?          Ss    07:40   0:00 cron
whoopsie 1048  0.0  0.5  511792  11232 ?         Ssl   07:40   0:00 whoopsie
root     1063  0.0  0.2  366504  4684 ?          SLsl  07:40   0:00 lightdm
root     1130  0.0  2.4  269148  50176 tty7       Ss+  07:40   0:10 /usr/bin/X -core :0 -seat seat0 -auth /var/run/lightdm/root/:0 -
nolisten tcp vt7 -novtswitch
root     1135  0.0  0.2  302356  5156 ?          SL    07:40   0:00 /usr/lib/accountsservice/accounts-daemon
root     1191  0.0  0.0      0   0 ?              S    07:40   0:00 [kauditfd]
root     1194  0.0  0.1  75484   3396 ?          Ss    07:40   0:00 /usr/sbin/cups-browsed
vncuser  1219  0.0  0.2  22008   4908 ?          S    07:40   0:03 Xvnc4 :1 -desktop ubuntu:1 (vncuser) -auth /home/vncuser/.Xautho
rity -geometry 1024x768 -depth 16 -rfbwait 30000 -rfbauth /home/vncuser/.vnc/passwd -rfbport 5901 -pn -fp /usr/X11R6/lib/X11/font
s/Type1/,/usr/X11R6/lib/X11/fonts/Speedo/,/usr/X11R6/lib/X11/fonts/misc/,/usr/X11R6/lib/X11/fonts/75dpi/,/usr/X11R6/lib/X11/fonts
/100dpi/,/usr/share/fonts/X11/misc/,/usr/share/fonts/X11/Type1/,/usr/share/fonts/X11/75dpi/,/usr/share/fonts/X11/100dpi/ -co /etc
/X11/rgb -idleTimeout 0
nobody   1283  0.0  0.0  38216  1540 ?          S    07:40   0:00 /usr/sbin/dnsmasq --no-resolv --keep-in-foreground --no-hosts --
bind-interfaces --pid-file=/run/sendsigs.omit.d/network-manager.dnsmasq.pid --listen-address=127.0.1.1 --conf-file=/var/run/Netwo
rkManager/dnsmasq.conf --cache-size=0 --proxy-dnssec --enable-dbus=org.freedesktop.NetworkManager.dnsmasq --conf-dir=/etc/Network
Manager/dnsmasq.d
root     1521  0.0  0.0  189012  652 ?          Ssl   07:40   0:00 /usr/sbin/vmware-vmblock-fuse -o subtype=vmware-vmblock,default_
permissions,allow_other /var/run/vmblock-fuse
root     1564  0.1  0.2  165504  4676 ?          Sl    07:40   0:11 /usr/sbin/vmtoolsd
root     1584  0.0  0.2  239380  4608 ?          Sl    07:40   0:00 /usr/lib/upower/upowerd
rtkit   1592  0.0  0.0  168916  1340 ?          SNl   07:40   0:00 /usr/lib/rtkit/rtkit-daemon
vncuser 1868  0.0  0.0    4444   648 ?          S    07:40   0:00 /bin/sh /home/vncuser/.vnc/xstartup
root     1871  0.0  0.0  23008   944 tty1        Ss+  07:40   0:00 /sbin/getty -8 38400 tty1
vncuser 1882  0.0  0.0  30664  1800 ?          S    07:40   0:00 vncconfig -nowin
vncuser 1923  0.3  5.0  930572 102692 ?         Sl    07:40   0:42 chromium-browser --enable-pinch --window-size=1024,768 --window-
position=1,1
:
```

python vol.py linux_pslist

0xfffff880036b85fc0	upstart-socket-	906	0	0	0x000000006e2ff000 0
0xfffff8800798d17f0	getty	957	0	0	0x000000006e30c000 0
0xfffff880079c70000	getty	962	0	0	0x000000006e37f000 0
0xfffff88006d9047d0	getty	970	0	0	0x000000006c447000 0
0xfffff88006d9017f0	getty	971	0	0	0x000000006c455000 0
0xfffff88006e3297f0	getty	974	0	0	0x0000000079e82000 0
0xfffff88006c6d5fc0	sshd	1018	0	0	0x000000006dbf3000 0
0xfffff88006e2d0000	kerneloops	1031	106	4	0x0000000079eca000 0
0xfffff88006d87c7d0	acpid	1033	0	0	0x000000006c51f000 0
0xfffff88006c6e47d0	cron	1035	0	0	0x000000006dbcce000 0
0xfffff88006d888000	whoopsie	1048	109	116	0x000000006c70b000 0
0xfffff88006c6d0000	lightdm	1063	0	0	0x000000006dbf8000 0
0xfffff88006c6d47d0	Xorg	1130	0	0	0x000000006c5fe000 0
0xfffff88006b8147d0	accounts-daemon	1135	0	0	0x000000006c79e000 0
0xfffff880079c72fe0	kaudittd	1191	0	0	----- 0
0xfffff88006c5c8000	cups-browsed	1194	0	0	0x000000006b8e9000 0
0xfffff88006c5cc7d0	Xvnc4	1219	1001	1001	0x000000006d83a000 0
0xfffff88006b815fc0	dnsmasq	1283	65534	30	0x000000006c792000 0
0xfffff88006ba6c7d0	vmware-vmblock-	1521	0	0	0x000000006ba78000 0
0xfffff8800669e97f0	vmtoolsd	1564	0	0	0x000000006bbb4000 0
0xfffff880066b2c7d0	upowerd	1584	0	0	0x000000006699a000 0
0xfffff880066990000	rtkit-daemon	1592	107	114	0x000000006bae3000 0
0xfffff8800668e47d0	xstartup	1868	1001	1001	0x0000000079f8d000 0
0xfffff88006b93c7d0	getty	1871	0	0	0x000000006c613000 0
0xfffff88007a7e5fc0	vncconfig	1882	1001	1001	0x000000006c6b7000 0
0xfffff880066b7c7d0	chromium-browse	1923	1001	1001	0x000000007a6b9000 0
0xfffff880066bc2fe0	chromium-browse	1936	1001	1001	0x000000006c74c000 0
0xfffff880066bc17f0	chrome-sandbox	1937	1001	1001	0x0000000079fe6000 0
0xfffff880066bc47d0	chromium-browse	1939	1001	1001	0x0000000066a37000 0
0xfffff880079810000	chromium-browse	1944	1001	1001	0x000000006c7c0000 0
0xfffff88005b50dfc0	dbus-launch	1983	1001	1001	0x000000005b471000 0
0xfffff88005fecdfc0	dbus-daemon	1984	1001	1001	0x000000005b48f000 0
0xfffff88006d88c7d0	lightdm	2135	0	0	0x0000000066b58000 0
0xfffff88006b9c5fc0	colord	2154	113	121	0x0000000066b60000 0
0xfffff88005b5d17f0	gnome-keyring-d	2218	1000	1000	0x000000006bb4d000 0
0xfffff880066a497f0	init	2220	1000	1000	0x0000000036e55000 0
0xfffff8800668e0000	ssh-agent	2337	1000	1000	0x0000000036c5e000 0
0xfffff880066b78000	dbus-daemon	2339	1000	1000	0x0000000066835000 0

Installing Volatility: Linux

This recipe for installing Volatility is for Ubuntu or other Debian-based Linux distros:

```
$ sudo apt-get install build-essential -y
$ sudo apt-get install linux-headers-$(uname -r)
$ cd /usr/src
$ sudo apt-get source linux-image-$(uname -r)
$ sudo apt-get install git subversion pcregrep libpcre++-dev -y
$ sudo apt-get install python-dev -y
$ sudo apt-get install dwarfdump -y
```

Installing Volatility: Linux (2)

```
$ git clone  
https://github.com/gdabah/distorm.git  
$ cd distorm3  
$ python setup.py build  
$ sudo python setup.py build install  
$ cd
```

distorm is a disassembler library

Installing Volatility: Linux (3)

Pattern matching framework

```
$ sudo apt-get install yara python-yara -y
```

```
$ wget
```

[https://ftp.dlitz.net/pub/dlitz/crypto/pycrypto/
pycrypto-2.6.1.tar.gz](https://ftp.dlitz.net/pub/dlitz/crypto/pycrypto/pycrypto-2.6.1.tar.gz)

```
$ tar -xvzf pycrypto-2.6.1.tar.gz
```

Python crypto framework

```
$ cd pycrypto-2.6.1
```

```
$ python setup.py build
```

```
$ sudo python setup.py build install
```

```
$ cd
```

Grab latest release of Volatility

```
$ git clone
```

<https://github.com/volatilityfoundation/volatility.git>

```
$ cd volatility
```

Preliminary setup
and then sanity check

```
$ python setup.py build
```

```
$ python vol.py --info
```

Installing Volatility: Linux (4)

```
$ cd ~/volatility  
$ git pull
```

Stay up to date with latest Volatility patches and enhancements
(do this periodically)

Installing Volatility: Mac OS X

First install **macports** via

<https://guide.macports.org/chunked/installing.macports.html> . It's the best way to install Unix packages on Mac OS X. Then:

```
$ sudo port install gmp
```

Install pip by:

Downloading and saving get-pip.py via <https://bootstrap.pypa.io/get-pip.py>

```
$ sudo python get-pip.py
```

Then install pycrypto:

```
$ARCHFLAGS=-Wno-error CFLAGS=-I/opt/local/include sudo -E pip  
install pycrypto
```

Installing Volatility: Mac OS X (2)

Next install distorm by downloading **distorm3.zip** from
<https://code.google.com/p/distorm/downloads/list>

Then:

```
$ unzip distorm3.zip
$ cd distorm3
$ ARCHFLAGS=-Wno-error CFLAGS=-I/opt/local/include sudo -E
python setup.py build
$ ARCHFLAGS=-Wno-error CFLAGS=-I/opt/local/include sudo -E
python setup.py install
$ cd
```

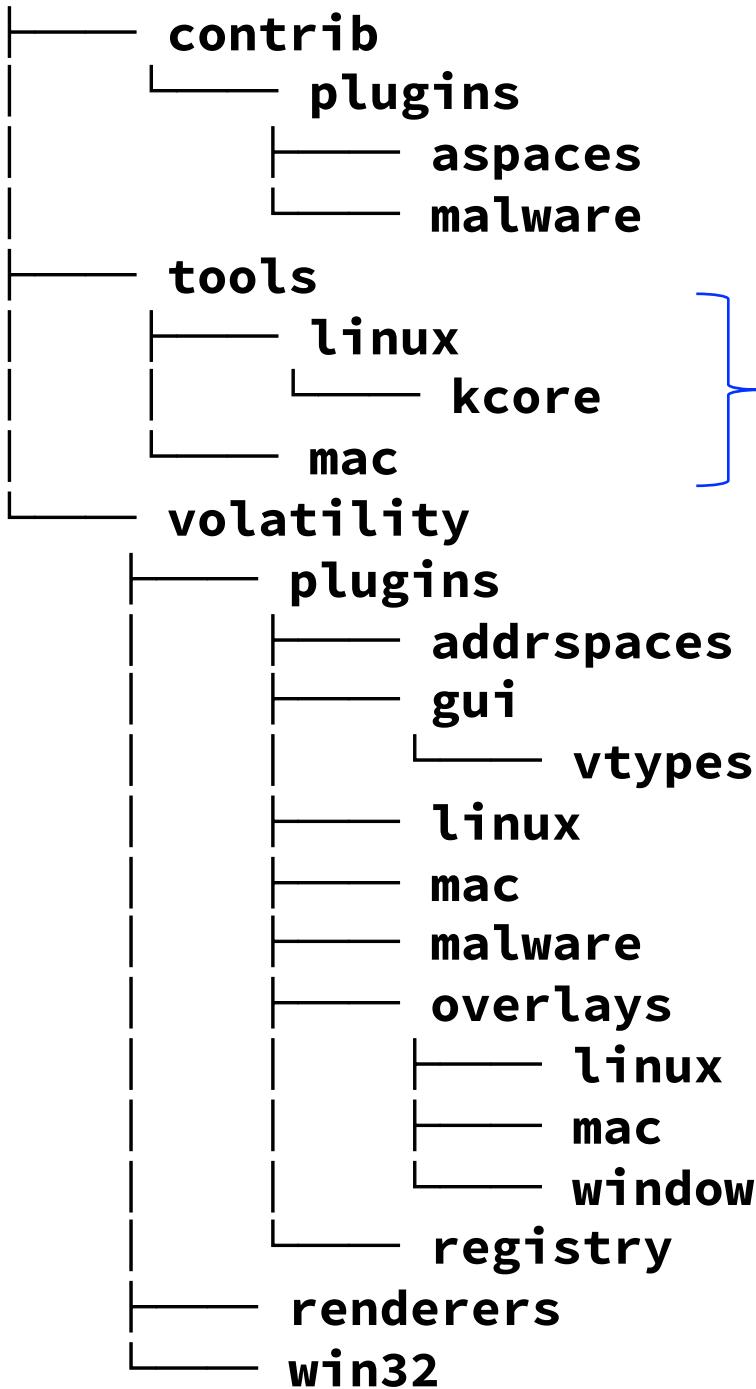
Installing Volatility: Mac OS X (3)

Finally, install yara and python-yara (which rely on libtoolize, et al for configuration, thus the extra steps):

```
$ cd  
$ sudo port install libtool  
$ sudo port install automake  
$ sudo port install autoconf  
$ git clone https://github.com/plusvic/yara.git
```

Then:

```
$ cd yara  
$ ./bootstrap.sh  
$ ./configure  
$ sudo make install  
$ cd yara-python  
$ python setup.py build  
$ sudo python setup.py install
```



Tools for generating new profiles located here

Profiles located here

Basic Volatility Usage: Simple Examples

Basic Volatility Usage

- Need physical memory dump
 - More on acquisition soon
- Need a Volatility profile for the OS that was running on the target machine
 - Defines kernel data structures, lots more
 - More on that soon
- Then choose from available commands ("plugins"), write your own, or use volshell

garfish:voltrunk golden\$ python vol.py -h
Volatile Systems Volatility Framework 2.3_alpha
Usage: Volatility - A memory forensics analysis platform.

Options:

- h, --help list all available options and their default values.
Default values may be set in the configuration file
(/etc/volatilityrc)
- conf-file=/Users/golden/.volatilityrc User based configuration file
- d, --debug Debug volatility
- plugins=PLUGINS Additional plugin directories to use (colon separated)
- info Print information about all registered objects
- cache-directory=/Users/golden/.cache/volatility Directory where cache files are stored
- cache Use caching
- tz=TZ Sets the timezone for displaying timestamps
- f FILENAME, --filename=FILENAME Filename to use when opening an image
- profile=WinXPSP2x86 Name of the profile to load
- l LOCATION, --location=LOCATION A URN location from which to load an address space
- w, --write Enable write support
- dtb=DTB DTB Address
- cache-dtb Cache virtual to physical mappings
- use-old-as Use the legacy address spaces
- output=text Output in this format (format support is module specific)
- output-file=OUTPUT_FILE write output in this file
- v, --verbose Verbose information
- g KDBG, --kdbg=KDBG Specify a specific KDBG virtual address
- k KPCR, --kpcr=KPCR Specify a specific KPCR address



Supported Plugin Commands:

apihooks	Detect API hooks in process and kernel memory
atoms	Print session and window station atom tables
atomscan	Pool scanner for _RTL_ATOM_TABLE
bioskbd	Reads the keyboard buffer from Real Mode memory
callbacks	Print system-wide notification routines
clipboard	Extract the contents of the windows clipboard
cmdscan	Extract command history by scanning for _COMMAND_HISTORY
connections	Print list of open connections [Windows XP and 2003 Only]
connscan	Scan Physical memory for _TCPT_OBJECT objects (tcp connections)
consoles	Extract command history by scanning for _CONSOLE_INFORMATION
crashinfo	Dump crash-dump information
deskscan	Poolscanner for tagDESKTOP (desktops)
devicetree	Show device tree
dlldump	Dump DLLs from a process address space
dlllist	Print list of loaded dlls for each process
driverirp	Driver IRP hook detection
driverscan	Scan for driver objects _DRIVER_OBJECT
envars	Display process environment variables
eventhooks	Print details on windows event hooks
evtlogs	Extract Windows Event Logs (XP/2003 only)
filescan	Scan Physical memory for _FILE_OBJECT pool allocations
gahti	Dump the USER handle type information
gditimers	Print installed GDI timers and callbacks
gdt	Display Global Descriptor Table
getservicesids	Get the names of services in the Registry and return Calculated SID
getsids	Print the SIDs owning each process
handles	Print list of open handles for each process
hashdump	Dumps passwords hashes (LM/NTLM) from memory
hibinfo	Dump hibernation file information
hivedump	Prints out a hive
hivelist	Print list of registry hives.
hivescan	Scan Physical memory for _CMHIVE objects (registry hives)

```
garfish:voltrunk golden$ python vol.py imageinfo -f PROSCIUTTO-VM-X-20121024-211516.raw
Volatile Systems Volatility Framework 2.3_alpha
Determining profile based on KDBG search...

WARNING : volatility.obj      : Overlay structure tty_struct not present in vtypes
WARNING : volatility.obj      : Overlay structure tty_struct not present in vtypes
    Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
        AS Layer1 : JKIA32PagedMemoryPae (Kernel AS)
        AS Layer2 : FileAddressSpace (/Users/golden/Work/voltrunk/PROSCIUTTO-VM-X-20121024
-211516.raw)
            PAE type : PAE
            DTB   : 0x335000L
            KDBG  : 0x8054d2e0
            Number of Processors : 1
            Image Type (Service Pack) : 3
                KPCR for CPU 0 : 0xffdff000
                KUSER_SHARED_DATA : 0xffffdf0000
            Image date and time : 2012-10-24 21:15:21 UTC+0000
            Image local date and time : 2012-10-24 16:15:21 -0500
garfish:voltrunk golden$
```

garfish:voltrunk golden\$ python vol.py pslist -f PROSCIUTTO-VM-X-20121024-211516.raw

Volatile Systems Volatility Framework 2.3_alpha

Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start	Exit
0x8ae39830	System	4	0	64	1322	-----	0		
0x8a65cda0	smss.exe	672	4	3	19	-----	0	2012-10-24 12:52:37	
0x8a672da0	csrss.exe	1016	672	14	760	0	0	2012-10-24 12:52:39	
0x8aa7e888	winlogon.exe	1048	672	18	572	0	0	2012-10-24 12:52:40	
0x8aab6da0	services.exe	1144	1048	15	366	0	0	2012-10-24 12:52:40	
0x8aa774b0	lsass.exe	1156	1048	23	392	0	0	2012-10-24 12:52:40	
0x8aa71da0	vmacthlp.exe	1320	1144	1	25	0	0	2012-10-24 12:52:40	
0x8a6f3500	svchost.exe	1332	1144	18	205	0	0	2012-10-24 12:52:41	
0x8aad5da0	svchost.exe	1380	1144	9	337	0	0	2012-10-24 12:52:41	
0x8aaac7e8	svchost.exe	1572	1144	68	1592	0	0	2012-10-24 12:52:41	
0x8aa6ea78	svchost.exe	1616	1144	6	85	0	0	2012-10-24 12:52:41	
0x8a709128	svchost.exe	1644	1144	13	177	0	0	2012-10-24 12:52:41	
0x8aa70020	spoolsv.exe	1956	1144	16	231	0	0	2012-10-24 12:52:43	
0x8aac4da0	explorer.exe	380	332	14	859	0	0	2012-10-24 12:52:51	
0x8a48ca90	GrooveMonitor.e	504	380	1	123	0	0	2012-10-24 12:52:52	
0x8aa72928	ISUSPM.exe	512	380	4	93	0	0	2012-10-24 12:52:52	
0x8a4b3da0	acrobat_sl.exe	520	380	0	-----	0	0	2012-10-24 12:52:52	2012-10-24 12:59:11
0x8a4c6da0	acrotray.exe	536	380	2	44	0	0	2012-10-24 12:52:52	
0x8a4cdda0	AdobeARM.exe	544	380	9	322	0	0	2012-10-24 12:52:52	
0x8aae5768	rundll32.exe	552	380	4	76	0	0	2012-10-24 12:52:52	
0x8a4b6da0	reader_sl.exe	560	380	0	-----	0	0	2012-10-24 12:52:52	2012-10-24 12:59:09
0x8aaa1a48	vmtoolsd.exe	568	380	6	303	0	0	2012-10-24 12:52:52	
0x8a497da0	jusched.exe	576	380	4	327	0	0	2012-10-24 12:52:52	
0x8a486da0	ctfmon.exe	584	380	1	76	0	0	2012-10-24 12:52:52	
0x8ad9e780	wcescomm.exe	784	380	4	134	0	0	2012-10-24 12:52:53	
0x8aa8e808	rapimgr.exe	820	1332	6	163	0	0	2012-10-24 12:52:53	
0x8aaa71d0	BTRay.exe	860	380	6	97	0	0	2012-10-24 12:52:53	
0x8aa88da0	CodeMeterCC.exe	900	380	1	202	0	0	2012-10-24 12:52:53	
0x8ad48b00	MagicDisc.exe	1092	380	1	32	0	0	2012-10-24 12:52:54	
0x8a3ae3b8	ONENOTEM.EXE	1228	380	4	113	0	0	2012-10-24 12:52:54	
0x8a650a20	CodeMeter.exe	1548	900	0	-----	0	0	2012-10-24 12:52:55	2012-10-24 12:53:13

garfish:voltrunk golden\$ python vol.py sockets -f PROSCIUTTO-VM-X-20121024-211516.raw

Volatile Systems Volatility Framework 2.3_alpha

Offset(V)	PID	Port	Proto	Protocol	Address	Create Time
0x8960ae98	880	1053	6	TCP	0.0.0.0	2012-10-24 12:54:17
0x89b94a68	880	1084	6	TCP	0.0.0.0	2012-10-24 12:54:21
0x89ba2008	880	1101	6	TCP	127.0.0.1	2012-10-24 12:54:23
0x89ae47b0	1644	1900	17	UDP	172.16.156.130	2012-10-24 19:41:20
0x89b331d8	880	1088	6	TCP	0.0.0.0	2012-10-24 12:54:21
0x89bc13d8	880	1105	6	TCP	127.0.0.1	2012-10-24 12:54:23
0x8a678e70	680	22350	17	UDP	0.0.0.0	2012-10-24 12:53:15
0x8a700b00	4	0	47	GRE	0.0.0.0	2012-10-24 12:53:33
0x896a68b8	2780	1047	6	TCP	127.0.0.1	2012-10-24 12:54:06
0x89bd11e0	880	1092	6	TCP	0.0.0.0	2012-10-24 12:54:21
0x8a55fad0	2940	22	6	TCP	0.0.0.0	2012-10-24 12:53:19
0x896cd840	880	1096	6	TCP	0.0.0.0	2012-10-24 12:54:22
0x89bca0b0	880	1051	6	TCP	127.0.0.1	2012-10-24 12:54:17
0x8a07f778	1156	500	17	UDP	0.0.0.0	2012-10-24 12:53:28
0x8a69c2f0	4	139	6	TCP	172.16.156.130	2012-10-24 19:41:20
0x8ab69b70	2300	1039	17	UDP	0.0.0.0	2012-10-24 12:53:20
0x8960ce98	880	1055	6	TCP	127.0.0.1	2012-10-24 12:54:17
0x896b48e0	880	1100	6	TCP	0.0.0.0	2012-10-24 12:54:22
0x89fc6240	736	2077	6	TCP	0.0.0.0	2012-10-24 21:14:33
0x89aed540	880	1059	6	TCP	127.0.0.1	2012-10-24 12:54:17
0x89b7db00	880	1104	6	TCP	0.0.0.0	2012-10-24 12:54:23
0x89c584d0	2556	3306	6	TCP	0.0.0.0	2012-10-24 12:53:53
0x89610918	880	1063	6	TCP	127.0.0.1	2012-10-24 12:54:17
0x89af0758	880	1108	6	TCP	0.0.0.0	2012-10-24 12:54:23
0x89bbd918	736	2085	6	TCP	0.0.0.0	2012-10-24 21:14:33
0x8a699bb0	4	445	6	TCP	0.0.0.0	2012-10-24 12:52:37
0x89a9d528	736	2089	6	TCP	0.0.0.0	2012-10-24 21:14:37
0x89af8a40	880	1067	6	TCP	127.0.0.1	2012-10-24 12:54:18
0x89bca778	964	1050	6	TCP	0.0.0.0	2012-10-24 12:54:17
0x8aaaf87e0	1380	135	6	TCP	0.0.0.0	2012-10-24 12:52:41
0x896115b0	880	1071	6	TCP	127.0.0.1	2012-10-24 12:54:18

1. bash					
Offset(V)	Pid	Handle	Access	Type	Details
0x8ae39830	4	0x4	0x1f0fff	Process	System(4)
0x8ae38020	4	0x8	0x0	Thread	TID 12 PID 4
0xe172e1b0	4	0xc	0xf003f	Key	MACHINE\SYSTEM\CONTROLSET001\CONTROL\SESSION MANAGER\MEMORY MANA
GEMENT\PREFETCHPARAMETERS					
0xe1012490	4	0x10	0x0	Key	MACHINE\SYSTEM\SETUP
0xe173c490	4	0x14	0x2001f	Key	MACHINE\HARDWARE\DESCRIPTION\SYSTEM\MULTIFUNCTIONADAPTER
0xe1747430	4	0x18	0x20019	Key	MACHINE\SYSTEM\WPA\KEY-CJ27J3P2XV9J9JCPB4DVT
0xe1730188	4	0x1c	0x20019	Key	MACHINE\SYSTEM\WPA\KEY-4F3B2RFXXC9C637882MBM
0xe173f430	4	0x20	0x20019	Key	MACHINE\SYSTEM\WPA\MEDiacenter
0xe1748430	4	0x24	0x20019	Key	MACHINE\SYSTEM\WPA\PNP
0xe1750430	4	0x28	0x20019	Key	MACHINE\SYSTEM\WPA\SIGNINGHASH-6KCM6KFTX6MD62
0xe174e430	4	0x2c	0x20019	Key	MACHINE\SYSTEM\WPA\SIGNINGHASH-V44KQMCFXKQCTQ
0xe1751430	4	0x30	0x20019	Key	MACHINE\SYSTEM\CONTROLSET001\CONTROL\PRODUCTOPTIONS
0xe176a450	4	0x34	0x2001f	Key	MACHINE\SYSTEM\CONTROLSET001\SERVICES\EVENTLOG
0xe175e430	4	0x38	0x20019	Key	TRWKWS_EVENT
0x8ae325c0	4	0x3c	0x1f0003	Event	\Device\Gpc
0x8aaefeb8	4	0x94	0x12019f	File	TID 104 PID 4
0x8ae4f458	4	0x98	0x0	Thread	TID 96 PID 4
0x8ae51da8	4	0x9c	0x1f03ff	Thread	MACHINE\SYSTEM\CONTROLSET001\SERVICES\ACPI\PARAMETERS
0xe17420e8	4	0xa0	0x20019	Key	TID 112 PID 4
0x8ad95c38	4	0xa4	0x1f03ff	Thread	VxKernel2VoldEvent
0x8ae1e2b8	4	0xa8	0x1f0003	Event	MACHINE\HARDWARE\DEVICEMAP\SCSI\SCSI PORT 2\SCSI BUS 0\INITIATOR
0xe1021d68	4	0xb0	0x2001f	Key	ID 255
0xe175b7a0	4	0xb4	0x2001f	Key	MACHINE\HARDWARE\DEVICEMAP\SCSI\SCSI PORT 2
0xe1023ed0	4	0xbc	0x2001f	Key	MACHINE\HARDWARE\DEVICEMAP\SCSI
0x8ac97cb0	4	0xc0	0x1f03ff	Thread	TID 108 PID 4
0xe1011b58	4	0xc4	0x2001f	Key	MACHINE\HARDWARE\DEVICEMAP\SCSI\SCSI PORT 2\SCSI BUS 0
0xe100a558	4	0xd0	0xf000f	Directory	WinDfs
0xe177aed8	4	0xe0	0xf000f	Directory	Harddisk0
0x8a8c3f08	4	0x358	0x12019f	File	\Device\Tcp

```
garfish:voltrunk golden$ python vol.py cmdscan -f PROSCIUTTO-VM-X-20121024-211516.raw
Volatile Systems Volatility Framework 2.3_alpha
WARNING : volatility.plugins.malware.malfind: Single strings to search_process_memory is deprecated, use a list instead
*****
CommandProcess: csrss.exe Pid: 1016
CommandHistory: 0x4f43f0 Application: ADEngineW.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x648
*****
CommandProcess: csrss.exe Pid: 1016
CommandHistory: 0x4f6120 Application: cygrunsrv.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x8ac
*****
CommandProcess: csrss.exe Pid: 1016
CommandHistory: 0x4f65a0 Application: sshd.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x90c
*****
CommandProcess: csrss.exe Pid: 1016
CommandHistory: 0x4f7840 Application: DumpIt.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x9e4
*****
CommandProcess: csrss.exe Pid: 1016
CommandHistory: 0x4f82a0 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 14 LastAdded: 13 LastDisplayed: 13
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0xa7c
Cmd #0 @ 0x4f1f68: z;
Cmd #1 @ 0x4f6b58: z:
Cmd #2 @ 0x4f6b68: cd work
Cmd #3 @ 0x4f6b80: dir
Cmd #4 @ 0x4f84c8: cd voltrunk
Cmd #5 @ 0x4f6bf0: dir
Cmd #6 @ 0x4f6c00: cv
```

```
garfish:voltrunk golden$ python vol.py iehistory -f PROSCIUTTO-VM-X-20121024-211516.raw
Volatile Systems Volatility Framework 2.3_alpha
*****
Process: 380 explorer.exe
Cache type "URL " at 0x3ea5000
Record length: 0x100
Location: Visited: Golden@http://www.moonsols.com/favicon.ico
Last modified: 2012-10-24 21:14:35
Last accessed: 2012-10-24 21:14:35
File Offset: 0x100, Data Offset: 0x0, Data Length: 0x9c
*****
Process: 380 explorer.exe
Cache type "URL " at 0x3ea5100
Record length: 0x100
Location: Visited: Golden@https://my.vmware.com/web/vmware/searchresults
Last modified: 2012-10-17 20:56:38
Last accessed: 2012-10-17 20:56:38
File Offset: 0x100, Data Offset: 0x0, Data Length: 0xa8
*****
Process: 380 explorer.exe
Cache type "URL " at 0x3ea5200
Record length: 0x100
Location: Visited: Golden@http://vmware.simplefeed.net/rss
Last modified: 2012-10-17 20:56:26
Last accessed: 2012-10-17 20:56:26
File Offset: 0x100, Data Offset: 0x0, Data Length: 0x9c
*****
Process: 380 explorer.exe
Cache type "URL " at 0x3ea5300
Record length: 0x100
Location: Visited: Golden@file:///Z:/Work/voltrunk/explorer-dump/explorer.exe.9860400.0x6fff0000-0x6fffffff.dmp
Last modified: 2012-10-24 15:31:39
Last accessed: 2012-10-24 15:31:39
File Offset: 0x100, Data Offset: 0x0, Data Length: 0xd0
*****
Process: 380 explorer.exe
Cache type "URL " at 0x3ea5400
Record length: 0x200
Location: Visited: Golden@http://www.vmware.com/support
Last modified: 2012-10-17 20:56:25
```

```
garfish:voltrunk golden$ python vol.py pstree -f PROSCIUTTO-VM-X-20121024-211516.raw
Volatile Systems Volatility Framework 2.3_alpha
```

Name	Pid	PPid	Thds	Hnds	Time
0x8aac4da0:explorer.exe	380	332	14	859	2012-10-24 12:52:51
. 0x8aa72928:ISUSPM.exe	512	380	4	93	2012-10-24 12:52:52
. 0x8a651900:cmd.exe	2080	380	1	33	2012-10-24 15:24:02
.. . 0x89bbc020:DumpIt.exe	348	2080	1	25	2012-10-24 21:15:16
. 0x8a4b3da0:acrobat_sl.exe	520	380	0	-----	2012-10-24 12:52:52
. 0x8ad9e780:wcescomm.exe	784	380	4	134	2012-10-24 12:52:53
. 0x8a4c6da0:acrotray.exe	536	380	2	44	2012-10-24 12:52:52
. 0x8aa88da0:CodeMeterCC.exe	900	380	1	202	2012-10-24 12:52:53
.. . 0x8a650a20:CodeMeter.exe	1548	900	0	-----	2012-10-24 12:52:55
. 0x8aae5768:rundll32.exe	552	380	4	76	2012-10-24 12:52:52
. 0x8a4b6da0:reader_sl.exe	560	380	0	-----	2012-10-24 12:52:52
. 0x89732020:iexplore.exe	3600	380	14	397	2012-10-24 21:14:24
.. . 0x89b23020:iexplore.exe	736	3600	33	834	2012-10-24 21:14:24
. 0x8aaa1a48:vmtoolsd.exe	568	380	6	303	2012-10-24 12:52:52
. 0x8a497da0:jusched.exe	576	380	4	327	2012-10-24 12:52:52
.. . 0x89be3020:juchck.exe	728	576	5	262	2012-10-24 12:57:53
. 0x8a4cdda0:AdobeARM.exe	544	380	9	322	2012-10-24 12:52:52
. 0x8ad48b00:MagicDisc.exe	1092	380	1	32	2012-10-24 12:52:54
. 0x8a486da0:ctfmon.exe	584	380	1	76	2012-10-24 12:52:52
. 0x8a3ae3b8:ONENOTEM.EXE	1228	380	4	113	2012-10-24 12:52:54
. 0x898c1020:idaq.exe	2820	380	0	-----	2012-10-24 12:53:42
.. . 0x8a10e020:idaq.exe	1692	2820	4	1018	2012-10-24 15:33:50
. 0x8aaa71d0:BTTTray.exe	860	380	6	97	2012-10-24 12:52:53
. 0x8a48ca90:GrooveMonitor.e	504	380	1	123	2012-10-24 12:52:52
0x8ae39830:System	4	0	64	1322	1970-01-01 00:00:00
. 0x8a65cda0:smss.exe	672	4	3	19	2012-10-24 12:52:37
.. . 0x8aa7e888:winlogon.exe	1048	672	18	572	2012-10-24 12:52:40
... . 0x8aa774b0:lsass.exe	1156	1048	23	392	2012-10-24 12:52:40
... . 0x8aab6da0:services.exe	1144	1048	15	366	2012-10-24 12:52:40
.... . 0x8a09c598:extjob.exe	3200	1144	2	33	2012-10-24 12:53:25
.... . 0x8a3d8c70:svchost.exe	2628	1144	2	63	2012-10-24 12:53:17
.... . 0x8a10e8b0:btwdins.exe	3092	1144	5	62	2012-10-24 12:53:33
.... . 0x8a951b80:prtk_supervisor	772	1144	3	31	2012-10-24 12:53:01
.... . 0x8a39c020:ADEngineS.exe	880	772	54	907	2012-10-24 12:53:01
.... . 0x8a77a308:svchost.exe	284	1144	4	108	2012-10-24 12:53:01
.... . 0x8a4acda0:cygrunsrv.exe	2672	1144	4	86	2012-10-24 12:53:17

Before and After

- Having before- and after- memory dumps is very useful

- Especially important in malware forensics

```
$ python vol.py -profile=WinXPSP3x86 -f  
before.vmem pslist > before-pslist
```

```
$ python vol.py -profile=WinXPSP3x86 -f  
after.vmem pslist > after-pslist
```

```
$ diff -y before-pslist after-pslist
```

- There's some ongoing work to automate this process—check it out and watch for improvements

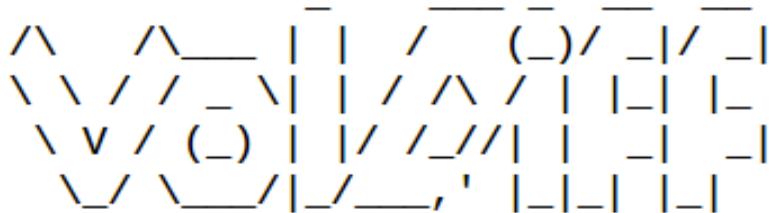
VolDiff

- Open source project to handle before/after comparisons
- Runs a bunch of plugins and generates nicely formatted report of differences
- <https://github.com/aim4r/VolDiff/wiki>
- Officially supports Linux as an investigative platform
- Can easily be made to run on Mac OS X
- And Win7 memory images
- But other things will work, but need to hack hardcoded lists of plugins to run
- (e.g., there will be problems with XP as-is unless you adapt it)
- No reason that with some effort it can't be made more flexible and configurable

remnux@remnux: ~/memory

File Edit Tabs Help

remnux@remnux:~/memory\$ VolDiff.py baseline.vmem infected.vmem Win8SP1x86



VolDiff: Malware Memory Footprint Analysis (v2.1)

Path to baseline memory image: baseline.vmem

Path to infected memory image: infected.vmem

WARNING: This script was only tested using Windows 7 profiles. The specified file (Win8SP1x86) seems different!

Running a selection of volatility plugins (time consuming):

Volatility plugin handles execution in progress...

Volatility plugin psxview execution in progress...

VolDiff_report.txt - SciTE

File Edit Search View Tools Options Language Buffers Help

1 VolDiff_report.txt

New netscan entries.

Offset(P)	Proto	Local Address	Foreign Address	State	Pid	Owner
0x7cadb858	TCPv4	172.16.240.128:49159	-:443	ESTABLISHED	2976	svchost.exe

VolDiff_Report.txt - SciTE

File Edit Search View Tools Options Language Buffers Help

1 VolDiff_Report.txt

Process svchost.exe (2976) is running in unexpected session (1 instead of 0).

Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64 Start
0x874fecc0	svchost.exe	2976	2968	5	0	1	0 2015-06-14 16:12:10 UTC+0000

These examples via:

<http://malwology.com/2015/06/25/remnux-v6-for-malware-analysis-part-1-voldiff/>

VolDiff

```
golden@ubuntu:~/volatility
File Edit View Search Terminal Help
python VolDiff.py /mnt/hgfs/m/before-republic.dmp /mnt/hgfs/m/after-republic-2.
dmp Win7SP1x64 --malware-checks
Volatility plugin consoles execution in progress...
Volatility plugin dlllist execution in progress...
Volatility plugin filescan execution in progress...
Volatility plugin shimcache execution in progress...
Volatility plugin shellbags execution in progress...
Volatility plugin sessions execution in progress...
Volatility plugin messagehooks execution in progress...
Volatility plugin eventhooks execution in progress...
Volatility plugin svcscan execution in progress...
Volatility plugin envars execution in progress...
Volatility plugin mutantscan execution in progress...
Volatility plugin symlinkscan execution in progress...
Volatility plugin atoms execution in progress...
Volatility plugin atomscan execution in progress...
Volatility plugin drivermodule execution in progress...
Volatility plugin mftparser execution in progress...
Volatility plugin driverscan execution in progress...
Volatility plugin devicetree execution in progress...
Volatility plugin modules execution in progress...
Volatility plugin modscan execution in progress...
Volatility plugin unloadedmodules execution in progress...
```

VolDiff

```
golden@ubuntu:~/volatility
File Edit View Search Terminal Help
golden@ubuntu:~/volatility$ ls -ltr VolDiff_28-06-2015_18\:10/
total 528
drwxrwxr-x 2 golden golden 4096 Jun 29 18:04 procdump
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 handles
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 psxview
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 netscan
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 iehistory
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 pslist
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 getsids
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 psscan
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 cmdline
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 consoles
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 dlllist
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 filescan
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 shimcache
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 shellbags
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 sessions
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 messagehooks
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 eventhooks
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 svcscan
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 mutantscan
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 envars
drwxrwxr-x 2 golden golden 4096 Jun 29 18:14 symlinkscan
```

volshell

volshell (`linux_volshell`, `mac_volshell`)

- Interactive environment for exploring memory dumps
- Created by Brendan Dolan-Gavitt for Windows memory analysis
- Now available for Linux and Mac, too
- Use Volatility APIs, type Python commands
- Good for prototyping plugin development
- And ad hoc exploration

Using volshell

```
$ python vol.py --info | grep volshell
```

```
Volatility Foundation Volatility Framework 2.4
```

```
linux_volshell
```

- Shell in the memory image

```
mac_volshell
```

- Shell in the memory image

```
volshell
```

- Shell in the memory image

Thus, e.g.:

```
$ python vol.py -profile=WinXPSP3x86 -f mem.dump volshell
```

Commonly Used Commands

- hh()
 - Help
- cc()
 - Set process context
 - Can use **pid**, **name**, **offset of process descriptor**
 - cc(pid=2139)
 - cc(name="xmlavipv.exe")
- sc()
 - Show current context
- dt()
 - Describe a data structure

More Commands

- `addrspace()`
 - Access current address space object
- `ps()`
 - List active processes
- `getprocs()`
 - Generator for active processes
- `modules()`
 - List kernel modules
- `getmods()`
 - Generator for kernel modules

More Commands

- db(addr)
 - Dump bytes
- dd(addr)
 - Dump doublewords (32-bit quantities)
- dq(addr)
 - Dump quadwords (64-bit quantities)
- dis(addr)
 - Disassemble at address quantities)



1. Python

```
bigjoe:volatility golden$ python vol.py --profile=WinXPSP3x86 -f /Volumes/SLOWDATA/IMAGES-THUMB-BACKUP/MEMIMAGES/eblaster.raw.2 volshell
Volatility Foundation Volatility Framework 2.4
Current context: System @ 0x8a9bd660, pid=4, ppid=0 DTB=0x33e000
Welcome to volshell! Current memory image is:
file:///Volumes/SLOWDATA/IMAGES-THUMB-BACKUP/MEMIMAGES/eblaster.raw.2
To get help, type 'hh()'
>>> ps()
Name          PID    PPID   Offset
System         4      0      0x8a9bd660
smss.exe       472    4      0x8a73b680
csrss.exe     692    472   0x8a821200
winlogon.exe   716    472   0x8a8ac740
services.exe   760    716   0x8a765d38
lsass.exe      772    716   0x8a8389b8
svchost.exe   988    760   0x8a4cd5f8
svchost.exe  1076    760   0x8a420268
svchost.exe  1116    760   0x8a7d9020
svchost.exe  1168    760   0x8a3e5320
svchost.exe  1248    760   0x8a833480
aswUpdSv.exe  1484    760   0x8a3d5b28
ashServ.exe   1536    760   0x8a3d4420
explorer.exe  1612   1596  0x8a3cc430
hpsysdrv.exe  1672   1612  0x8a6c4a00
HpqCmon.exe   1680   1612  0x8a6bebe0
KBD.EXE        1688   1612  0x8a6a6a38
tfswctrl.exe  1696   1612  0x8a7cd5b8
CCAPP.EXE      1720   1612  0x8a3a4bf8
ashDisp.exe    1752   1612  0x8a3bd020
v4brmon.exe   1768   1612  0x8a6aca18
reader_sl.exe  1776   1612  0x8a3adbff
rundll32.exe   1784   1612  0x8a6b7958
ctfmon.exe     1800   1612  0x8a6a0a18
GoogleToolbarNo 1816   1612  0x8a3a8be0
GoogleUpdate.ex 1824   1612  0x8a3b12a0
hpotdd01.exe   1836   1612  0x8a6e05a0
MemTurbo.exe   1852   1612  0x8a3b24f0
spoolsv.exe    632    760   0x8a3638b0
svchost.exe    736    760   0x8a651880
svchost.exe    908    760   0x8a6528b0
```

1. Python

```
explorer.exe    1612  1596  0x8a3cc430
hpsysdrv.exe   1672  1612  0x8a6c4a00
HpqCmon.exe    1680  1612  0x8a6bebe0
KBD.EXE        1688  1612  0x8a6a6a38
tfswctrl.exe   1696  1612  0x8a7cd5b8
CCAPP.EXE      1720  1612  0x8a3a4bf8
ashDisp.exe    1752  1612  0x8a3bd020
v4brmon.exe    1768  1612  0x8a6aca18
reader_s1.exe   1776  1612  0x8a3adbf8
rundll32.exe   1784  1612  0x8a6b7958
ctfmon.exe     1800  1612  0x8a6a0a18
GoogleToolbarNo 1816  1612  0x8a3a8be0
GoogleUpdate.ex 1824  1612  0x8a3b12a0
hpotdd01.exe   1836  1612  0x8a6e05a0
MemTurbo.exe   1852  1612  0x8a3b24f0
spoolsv.exe    632   760   0x8a3638b0
svchost.exe    736   760   0x8a651880
svchost.exe    908   760   0x8a6528b0
E_S40RP7.EXE   1012  760   0x8a368da0
FlipShareServic 1136  760   0x8a33eda0
FlipShareServer 1288  760   0x8a6ce020
taskmgr.exe    1348  716   0x8a62ada0
NTRadmin.exe   1408  760   0x8a329b28
svchost.exe    1796  760   0x8a31e990
YahooAUService. 2104  760   0x8a601020
wuauclt.exe    2272  1116  0x8a69f3b0
ashMaiSv.exe   2596  760   0x8a5fd3d0
wscntfy.exe    2780  1116  0x8a8fdaf0
unsecapp.exe   2792  988   0x8a79bda0
alg.exe         2820  760   0x8a607020
rundll32.exe   2880  1408  0x8a39f990
wmiprvse.exe   2888  988   0x8a37fda0
rundll32.exe   3128  1408  0x8a671990
xmlavipv.exe   3700  1612  0x8a3e0208
```

```
>>> cc(pid=3700) ←
```

```
Current context: xmlavipv.exe @ 0x8a3e0208, pid=3700, ppid=1612 DTB=0x179c0620
```

```
>>> sc() ←
```

```
Current context: xmlavipv.exe @ 0x8a3e0208, pid=3700, ppid=1612 DTB=0x179c0620
```

```
>>> █
```

1. Python

```
explorer.exe    1612  1596  0x8a3cc430
hpsysdrv.exe   1672  1612  0x8a6c4a00
HpqCmon.exe    1680  1612  0x8a6bebe0
KBD.EXE        1688  1612  0x8a6a6a38
tfswctrl.exe   1696  1612  0x8a7cd5b8
CCAPP.EXE      1720  1612  0x8a3a4bf8
ashDisp.exe    1752  1612  0x8a3bd020
v4brmon.exe    1768  1612  0x8a6aca18
reader_s1.exe   1776  1612  0x8a3adbf8
rundll32.exe   1784  1612  0x8a6b7958
ctfmon.exe     1800  1612  0x8a6a0a18
GoogleToolbarNo 1816  1612  0x8a3a8be0
GoogleUpdate.ex 1824  1612  0x8a3b12a0
hpotdd01.exe   1836  1612  0x8a6e05a0
MemTurbo.exe   1852  1612  0x8a3b24f0
spoolsv.exe    632   760   0x8a3638b0
svchost.exe    736   760   0x8a651880
svchost.exe    908   760   0x8a6528b0
E_S40RP7.EXE   1012  760   0x8a368da0
FlipShareServic 1136  760   0x8a33eda0
FlipShareServer 1288  760   0x8a6ce020
taskmgr.exe    1348  716   0x8a62ada0
NTRadmin.exe   1408  760   0x8a329b28
svchost.exe    1796  760   0x8a31e990
YahooAUService. 2104  760   0x8a601020
wuauclt.exe    2272  1116  0x8a69f3b0
ashMaiSv.exe   2596  760   0x8a5fd3d0
wscntfy.exe    2780  1116  0x8a8fdaf0
unsecapp.exe   2792  988   0x8a79bda0
alg.exe         2820  760   0x8a607020
rundll32.exe   2880  1408  0x8a39f990
wmiprvse.exe   2888  988   0x8a37fda0
rundll32.exe   3128  1408  0x8a671990
xmlavipv.exe   3700  1612  0x8a3e0208
```

```
>>> cc(pid=3700) ←
Current context: xmlavipv.exe @ 0x8a3e0208, pid=3700, ppid=1612 DTB=0x179c0620
>>> print proc().CreateTime ←
2014-09-24 15:57:15 UTC+0000 ←
>>> █
```

1. Python

```
>>> dt("_EPROCESS") ←
'_EPROCESS' (608 bytes)
0x0 : Pcb ←
0x6c : ProcessLock ←
0x70 : CreateTime ←
0x78 : ExitTime ←
0x80 : RundownProtect ←
0x84 : UniqueProcessId ←
0x88 : ActiveProcessLinks ←
0x90 : QuotaUsage ←
0x9c : QuotaPeak ←
0xa8 : CommitCharge ←
0xac : PeakVirtualSize ←
0xb0 : VirtualSize ←
0xb4 : SessionProcessLinks ←
0xbc : DebugPort ←
0xc0 : ExceptionPort ←
0xc4 : ObjectTable ←
0xc8 : Token ←
0xcc : WorkingSetLock ←
0xec : WorkingSetPage ←
0xf0 : AddressCreationLock ←
0x110 : HyperSpaceLock ←
0x114 : ForkInProgress ←
0x118 : HardwareTrigger ←
0x11c : VadRoot ←
0x120 : VadHint ←
0x124 : CloneRoot ←
0x128 : NumberOfPrivatePages ←
0x12c : NumberOfLockedPages ←
0x130 : Win32Process ←
0x134 : Job ←
0x138 : SectionObject ←
0x13c : SectionBaseAddress ←
0x140 : QuotaBlock ←
0x144 : WorkingSetWatch ←
0x148 : Win32WindowStation ←
0x14c : InheritedFromUniqueProcessId ←
0x150 : LdtInformation ←
```

Windows XP Kernel Version 2600 (Service Pack 3) UP Free x86 compatible
Product: WinNt, suite: TerminalServer SingleUserTS
Built by: 2600.xpssp_sp3_qfe.130704-0421
Machine Name:
Kernel base = 0x804d7000 PsLoadedModuleList = 0x805541c0
Debug session time: Mon Jun 22 20:50:06.133 2015 (UTC - 5:00)
System Uptime: 0 days 0:53:51.083
1kd> dt _ETHREAD
nt!_ETHREAD
+0x000 Tcb : _KTHREAD
+0x1c0 CreateTime : _LARGE_INTEGER
+0x1c0 NestedFaultCount : Pos 0, 2 Bits
+0x1c0 ApcNeeded : Pos 2, 1 Bit
+0x1c8 ExitTime : _LARGE_INTEGER
+0x1c8 LpcReplyChain : _LIST_ENTRY
+0x1c8 KeyedWaitChain : _LIST_ENTRY
+0x1d0 ExitStatus : Int4B
+0x1d0 OfsChain : Ptr32 Void
+0x1d4 PostBlockList : _LIST_ENTRY
+0x1dc TerminationPort : Ptr32 _TERMINATION_PORT
+0x1dc ReaperLink : Ptr32 _ETHREAD
+0x1dc KeyedWaitValue : Ptr32 Void
+0x1e0 ActiveTimerListLock : Uint4B
+0x1e4 ActiveTimerListHead : _LIST_ENTRY
+0x1ec Cid : _CLIENT_ID
+0x1f4 LpcReplySemaphore : _KSEMAPHORE
+0x1f4 KeyedWaitSemaphore : _KSEMAPHORE
+0x208 LpcReplyMessage : Ptr32 Void
+0x208 LpcWaitingOnPort : Ptr32 Void
+0x20c ImpersonationInfo : Ptr32 _PS_IMPERSONATION_INFORMATION
+0x210 IrpList : _LIST_ENTRY
+0x218 TopLevelIrp : Uint4B
+0x21c DeviceToVerify : Ptr32 _DEVICE_OBJECT
+0x220 ThreadsProcess : Ptr32 _EPROCESS
+0x224 StartAddress : Ptr32 Void
+0x228 Win32StartAddress : Ptr32 Void
+0x228 LpcReceivedMessageId : Uint4B
+0x22c ThreadListEntry : _LIST_ENTRY
+0x234 RundownProtect : _EX_RUNDOWN_REF
+0x238 ThreadLock : _EX_PUSH_LOCK
+0x23c LpcReplyMessageId : Uint4B
+0x240 ReadClusterSize : Uint4B
+0x244 GrantedAccess : Uint4B
+0x248 CrossThreadFlags : Uint4B
+0x248 Terminated : Pos 0, 1 Bit

1. Python

0x150 : LdtInformation	['pointer', ['void']]
0x154 : VadFreeHint	['pointer', ['void']]
0x158 : VdmObjects	['pointer', ['void']]
0x15c : DeviceMap	['pointer', ['void']]
0x160 : PhysicalVadList	['_LIST_ENTRY']
0x168 : Filler	['unsigned long long']
0x168 : PageDirectoryPte	['_HARDWARE_PTE']
0x170 : Session	['pointer', ['void']]
0x174 : ImageFileName	['String', {'length': 16}]
0x184 : JobLinks	['_LIST_ENTRY']
0x18c : LockedPagesList	['pointer', ['void']]
0x190 : ThreadListHead	['_LIST_ENTRY']
0x198 : SecurityPort	['pointer', ['void']]
0x19c : PaeTop	['pointer', ['void']]
0x1a0 : ActiveThreads	['unsigned long']
0x1a4 : GrantedAccess	['unsigned long']
0x1a8 : DefaultHardErrorProcessing	['unsigned long']
0x1ac : LastThreadExitStatus	['long']
0x1b0 : Peb	['pointer', ['_PEB']]
0x1b4 : PrefetchTrace	['_EX_FAST_REF']
0x1b8 : ReadOperationCount	['_LARGE_INTEGER']
0x1c0 : WriteOperationCount	['_LARGE_INTEGER']
0x1c8 : OtherOperationCount	['_LARGE_INTEGER']
0x1d0 : ReadTransferCount	['_LARGE_INTEGER']
0x1d8 : WriteTransferCount	['_LARGE_INTEGER']
0x1e0 : OtherTransferCount	['_LARGE_INTEGER']
0x1e8 : CommitChargeLimit	['unsigned long']
0x1ec : CommitChargePeak	['unsigned long']
0x1f0 : AweInfo	['pointer', ['void']]
0x1f4 : SeAuditProcessCreateInfo	['_SE_AUDIT_PROCESS_CREATION_INFO']
0x1f8 : Vm	['_MMSUPPORT']
0x238 : LastFaultCount	['unsigned long']
0x23c : ModifiedPageCount	['unsigned long']

1. Python

```
>>> dt("_KPROCESS") ←  
'_KPROCESS' (108 bytes)  
0x0 : Header ['_DISPATCHER_HEADER']  
0x10 : ProfileListHead ['_LIST_ENTRY']  
0x18 : DirectoryTableBase ['unsigned long']  
0x20 : LdtDescriptor ['_KGDTENTRY']  
0x28 : Int21Descriptor ['_KIDTENTRY']  
0x30 : IopmOffset ['unsigned short']  
0x32 : Iopl ['unsigned char']  
0x33 : Unused ['unsigned char']  
0x34 : ActiveProcessors ['unsigned long']  
0x38 : KernelTime ['unsigned long']  
0x3c : UserTime ['unsigned long']  
0x40 : ReadyListHead ['_LIST_ENTRY']  
0x48 : SwapListEntry ['_SINGLE_LIST_ENTRY']  
0x4c : VdmTrapHandler ['pointer', ['void']]  
0x50 : ThreadListHead ['_LIST_ENTRY']  
0x58 : ProcessLock ['unsigned long']  
0x5c : Affinity ['unsigned long']  
0x60 : StackCount ['unsigned short']  
0x62 : BasePriority ['unsigned char']  
0x63 : ThreadQuantum ['unsigned char']  
0x64 : AutoAlignment ['unsigned char']  
0x65 : State ['unsigned char']  
0x66 : ThreadSeed ['unsigned char']  
0x67 : DisableBoost ['unsigned char']  
0x68 : PowerState ['unsigned char']  
0x69 : DisableQuantum ['unsigned char']  
0x6a : IdealNode ['unsigned char']  
0x6b : ExecuteOptions ['unsigned char']  
0x6b : Flags ['_KEXECUTE_OPTIONS']  
>>>
```

```
>>> dt("_PEB") ←
 '_PEB' (528 bytes)
0x0 : InheritedAddressSpace      ['unsigned char']
0x1 : ReadImageFileExecOptions  ['unsigned char']
0x2 : BeingDebugged            ['unsigned char'] ←
0x3 : SpareBool                ['unsigned char']
0x4 : Mutant                   ['pointer', ['void']] ←
0x8 : ImageBaseAddress         ['pointer', ['void']] ←
0xc : Ldr                      ['pointer', ['_PEB_LDR_DATA']]
0x10 : ProcessParameters        ['pointer', ['_RTL_USER_PROCESS_PARAMETERS']]
0x14 : SubSystemData           ['pointer', ['void']]
0x18 : ProcessHeap              ['pointer', ['void']]
0x1c : FastPebLock             ['pointer', ['_RTL_CRITICAL_SECTION']]
0x20 : FastPebLockRoutine      ['pointer', ['void']]
0x24 : FastPebUnlockRoutine    ['pointer', ['void']]
0x28 : EnvironmentUpdateCount  ['unsigned long']
0x2c : KernelCallbackTable     ['pointer', ['void']]
0x30 : SystemReserved          ['array', 1, ['unsigned long']]
0x34 : AtlThunkSListPtr32       ['unsigned long']
0x38 : FreeList                 ['pointer', ['_PEB_FREE_BLOCK']]
0x3c : TlsExpansionCounter     ['unsigned long']
0x40 : TlsBitmap                ['pointer', ['void']]
0x44 : TlsBitmapBits            ['array', 2, ['unsigned long']]
0x4c : ReadOnlySharedMemoryBase  ['pointer', ['void']]
0x50 : ReadOnlySharedMemoryHeap  ['pointer', ['void']]
0x54 : ReadOnlyStaticServerData ['pointer', ['pointer', ['void']]]
0x58 : AnsiCodePageData         ['pointer', ['void']]
0x5c : OemCodePageData          ['pointer', ['void']]
0x60 : UnicodeCaseTableData    ['pointer', ['void']]
0x64 : NumberOfProcessors      ['unsigned long']
0x68 : NtGlobalFlag             ['unsigned long']
0x70 : CriticalSectionTimeout   ['_LARGE_INTEGER']
0x78 : HeapSegmentReserve       ['unsigned long']
0x7c : HeapSegmentCommit        ['unsigned long']
0x80 : HeapDeCommitTotalFreeThreshold  ['unsigned long']
0x84 : HeapDeCommitFreeBlockThreshold  ['unsigned long']
0x88 : NumberOfHeaps            ['unsigned long']
0x8c : MaximumNumberOfHeaps     ['unsigned long']
0x90 : ProcessHeaps             ['pointer', ['array', <function <lambda> at 0x110500140>, ['pointer', ['_HEAP']]]]
0x94 : GdiSharedHandleTable     ['pointer', ['void']]
0x98 : ProcessStarterHelper     ['pointer', ['void']]
0x9c : GdiDCAttributeList       ['unsigned long']
0xa0 : LoaderLock               ['pointer', ['void']]
0xa4 : OSMajorVersion            ['unsigned long']
0xa8 : OSMinorVersion            ['unsigned long']
0xac : OSBuildNumber             ['unsigned short']
```

1. Python

```
>>> proc()
[_EPROCESS _EPROCESS] @ 0x8A3E0208
>>> dt("_EPROCESS", 0x8A3E0208) ←
[_EPROCESS _EPROCESS] @ 0x8A3E0208
0x0      : Pcb                      2319319560
0x6c     : ProcessLock              2319319668
0x70      : CreateTime              2014-09-24 15:57:15 UTC+0000
0x78      : ExitTime                1970-01-01 00:00:00 UTC+0000
0x80      : RundownProtect         2319319688
0x84      : UniqueProcessId        3700
0x88      : ActiveProcessLinks      2319319696
0x90      : QuotaUsage             -
0x9c      : QuotaPeak               -
0xa8      : CommitCharge            1294
0xac      : PeakVirtualSize         47235072
0xb0      : VirtualSize             44077056
0xb4      : SessionProcessLinks    2319319740
0xbc      : DebugPort               0
0xc0      : ExceptionPort          3789084856
0xc4      : ObjectTable             3814211864
0xc8      : Token                   2319319760
0xcc      : WorkingSetLock          2319319764
0xec      : WorkingSetPage          258001
0xf0      : AddressCreationLock   2319319800
0x110     : HyperSpaceLock         0
0x114     : ForkInProgress         0
0x118     : HardwareTrigger        0
0x11c     : VadRoot                2321665488
0x120     : VadHint                2321665488
0x124     : CloneRoot              0
0x128     : NumberOfPrivatePages   1198
0x12c     : NumberOfLockedPages    0
0x130     : Win32Process            3810325000
```

1. Python

0x1a0 : ActiveThreads	2
0x1a4 : GrantedAccess	2035711
0x1a8 : DefaultHardErrorProcessing	1
0x1ac : LastThreadExitStatus	0
0x1b0 : Peb	2147332096 ←
0x1b4 : PrefetchTrace	2319319996
0x1b8 : ReadOperationCount	2319320000
0x1c0 : WriteOperationCount	2319320008
0x1c8 : OtherOperationCount	2319320016
0x1d0 : ReadTransferCount	2319320024
0x1d8 : WriteTransferCount	2319320032
0x1e0 : OtherTransferCount	2319320040
0x1e8 : CommitChargeLimit	0
0x1ec : CommitChargePeak	1294
0x1f0 : AweInfo	0
0x1f4 : SeAuditProcessCreateInfo	2319320060
0x1f8 : Vm	2319320064
0x238 : LastFaultCount	0
0x23c : ModifiedPageCount	152
0x240 : NumberOfVads	82
0x244 : JobStatus	0
0x248 : AddressSpaceInitialized	2
0x248 : BreakOnTermination	0
0x248 : CreateReported	0
0x248 : Flags	854016
0x248 : ForkFailed	0
0x248 : HasAddressSpace	1
0x248 : HasPhysicalVad	0
0x248 : InjectInpageErrors	0
0x248 : LaunchPrefetched	1
0x248 : NoDebugInherit	0
0x248 : OutswapEnabled	0
0x248 : Outswapped	0

1. Python

```
>>> dt("_PEB", 2147332096)
[CTYPE _PEB] @ 0x7FFDB000
0x0  : InheritedAddressSpace      0
0x1  : ReadImageFileExecOptions  0
0x2  : BeingDebugged            0
0x3  : SpareBool                0
0x4  : Mutant                   4294967295
0x8  : ImageBaseAddress         4194304 ←
0xc  : Ldr                      2432656
0x10 : ProcessParameters        131072
0x14 : SubSystemData            0
0x18 : ProcessHeap              1376256
0x1c : FastPebLock              2090337824
0x20 : FastPebLockRoutine       2089816064
0x24 : FastPebUnlockRoutine     2089816288
0x28 : EnvironmentUpdateCount  1
0x2c : KernelCallbackTable      2118199664
0x30 : SystemReserved           -
0x34 : AtlThunkSListPtr32        0
0x38 : FreeList                 0
0x3c : TlsExpansionCounter     0
0x40 : TlsBitmap                2090337760
0x44 : TlsBitmapBits            -
0x4c : ReadOnlySharedMemoryBase  2137980928
0x50 : ReadOnlySharedMemoryHeap  2137980928
0x54 : ReadOnlyStaticServerData 2137982600
0x58 : AnsiCodePageData         2147155968
0x5c : OemCodePageData          2147225600
0x60 : UnicodeCaseTableData    2147295232
0x64 : NumberOfProcessors      1
0x68 : NtGlobalFlag             0
0x70 : CriticalSectionTimeout   2147332208
0x78 : HeapSegmentReserve       1048576
```

1. Python

```
>>> print proc().Peb          ←  
2147332096  
>>> print proc().Peb.ImageBaseAddress ←  
4194304  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>
```

```
>>> dt("_LDR_DATA_TABLE_ENTRY") <
 '_LDR_DATA_TABLE_ENTRY' (80 bytes)
0x0 : InLoadOrderLinks      ['_LIST_ENTRY']
0x8 : InMemoryOrderLinks    ['_LIST_ENTRY']
0x10 : InInitializationOrderLinks  ['_LIST_ENTRY']
0x18 : DllBase              ['pointer', ['void']] <
0x1c : EntryPoint            ['pointer', ['void']] <
0x20 : SizeOfImage          ['unsigned long']
0x24 : FullDllName          ['_UNICODE_STRING'] <
0x2c : BaseDllName          ['_UNICODE_STRING'] <
0x34 : Flags                ['unsigned long']
0x38 : LoadCount             ['unsigned short']
0x3a : TlsIndex              ['unsigned short']
0x3c : HashLinks             ['_LIST_ENTRY']
0x3c : SectionPointer        ['pointer', ['void']]
0x40 : CheckSum              ['unsigned long']
0x44 : LoadedImports         ['pointer', ['void']]
0x44 : TimeDateStamp         ['UnixTimeStamp', {'is_utc': True}]
0x48 : EntryPointActivationContext  ['pointer', ['void']]
0x4c : PatchInformation       ['pointer', ['void']]
>>> 
```

In [4]: dt("task_struct", recursive=True)

'task_struct' (6120 bytes)

0x0	: state	['long']
0x8	: stack	['pointer', ['void']]
0x10	: usage	['__unnamed_0x346']
...	'__unnamed_0x346' (4 bytes)	
...0x0	: counter	['int']
0x14	: flags	['unsigned int']
0x18	: ptrace	['unsigned int']
0x20	: wake_entry	['llist_node']
...	'llist_node' (8 bytes)	
...0x0	: next	['pointer', ['llist_node']]
0x28	: on_cpu	['int']
0x30	: last_wakee	['pointer', ['task_struct']]
0x38	: wakee_flips	['unsigned long']
0x40	: wakee_flip_decay_ts	['unsigned long']
0x48	: wake_cpu	['int']
0x4c	: on_rq	['int']
0x50	: prio	['int']
0x54	: static_prio	['int']
0x58	: normal_prio	['int']
0x5c	: rt_priority	['unsigned int']

```
0x4d0 : thread ['thread_struct']
... 'thread_struct' (184 bytes)
... 0x0 : tls_array ['array', 3, ['desc_struct']]
... 0x18 : sp0 ['unsigned long']
... 0x20 : sp ['unsigned long']
... 0x28 : usersp ['unsigned long']
... 0x30 : es ['unsigned short']
... 0x32 : ds ['unsigned short']
... 0x34 : fsindex ['unsigned short']
... 0x36 : gsindex ['unsigned short']
... 0x38 : fs ['unsigned long']
... 0x40 : gs ['unsigned long']
... 0x48 : ptrace_bps ['array', 4, ['pointer', ['perf_event']]]
... 0x68 : debugreg6 ['unsigned long']
... 0x70 : ptrace_dr7 ['unsigned long']
... 0x78 : cr2 ['unsigned long']
... 0x80 : trap_nr ['unsigned long']
... 0x88 : error_code ['unsigned long']
... 0x90 : fpu ['fpu']

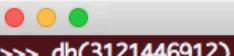
..... 'fpu' (16 bytes)
..... 0x0 : last_cpu ['unsigned int']
..... 0x4 : has_fpu ['unsigned int']
..... 0x8 : state ['pointer', ['thread_xstate']]
... 0xa0 : io_bitmap_ptr ['pointer', ['unsigned long']]
... 0xa8 : iopl ['unsigned long']
... 0xb0 : io_bitmap_max ['unsigned int']
... 0xb4 : fpu_counter ['unsigned char']
```



```
>>> for x in getmods():
...     y=x.dereference_as("_LDR_DATA_TABLE_ENTRY")
...     print y.FullName,y.DllBase,y.EntryPoint
...
\WINDOWS\system32\ntkrnlpa.exe 2152558592 2154366176
\WINDOWS\system32\hal.dll 2154631168 2154750136
\WINDOWS\system32\KDCOM.DLL 3126493184 3126496486
\WINDOWS\system32\BOOTVID.dll 3125510144 3125516402
ACPI.sys 3120009216 3120177241
\WINDOWS\System32\DRIVERS\WMILIB.SYS 3126501376 3126504320
pci.sys 3119939584 3119996932
isapnp.sys 3121250304 3121281538
compbatt.sys 3125526528 3125533907
\WINDOWS\system32\DRIVERS\BATT.CSYS 3125542912 3125550341
pciide.sys 3127312384 3127313950
\WINDOWS\System32\DRIVERS\PCIINDEX.SYS 3123871744 3123892741
viaide.sys 3126509568 3126513285
intelide.sys 3126517760 3126521605
MountMgr.sys 3121315840 3121353268
fdisk.sys 3119812608 3119924450
PartMgr.sys 3123904512 3123920011
VolSnap.sys 3121381376 3121421630
atapi.sys 3119714304 3119802871
disk.sys 3121446912 3121477805 <
\WINDOWS\System32\DRIVERS\CLASSPNP.SYS 3121512448 3121557007
fltmgr.sys 3119583232 3119702108
sr.sys 3119509504 3119574996
PxHelp20.sys 3125559296 3125560443
drvmcdb.sys 3119427584 3119481551
KSecDD.sys 3119333376 3119419827
Ntfs.sys 3118755840 3119301508
NDIS.sys 3118571520 3118739717
viaagp1.sys 3123937280 3123952320
SISAGP.sys 3123970048 3123984478
ohci1394.sys 3121577984 3121624965
\WINDOWS\System32\DRIVERS\1394BUS.SYS 3121643520 3121691781
nv_agp.sys 3125575680 3125581478
Mup.sys 3118465024 3118559542
agp440.sys 3121709056 3121745285
\SystemRoot\System32\DRIVERS\i8042prt.sys 3107885056 3107922565
\SystemRoot\System32\DRIVERS\PS2.sys 3117744128 3117754264
\SystemRoot\System32\DRIVERS\kbdclass.sys 3125084160 3125102096
\SystemRoot\System32\DRIVERS\mouclass.sys 3125116928 3125133365
\SystemRoot\System32\DRIVERS\parport.sys 3105828864 3105900293
\SystemRoot\System32\DRIVERS\serial.sys 3122036736 3122081467
\SystemRoot\System32\DRIVERS\serenum.sys 3117727744 3117740137
```

Scripting inside volshell—quick and dirty list of information about loaded kernel modules

Imagine disk.sys
is of interest...

 1. Python

79

```
>>> db(3121446912) ←  
0xba0d8000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 MZ.....  
0xba0d8010 b8 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00 .....@.....  
0xba0d8020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....  
0xba0d8030 00 00 00 00 00 00 00 00 00 00 00 00 00 c8 00 00 00 .....  
0xba0d8040 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 .....!..L.!Th  
0xba0d8050 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f is.program.canno  
0xba0d8060 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20 t.be.run.in.DOS.  
0xba0d8070 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00 mode....$.....  
>>> █
```

Dumping bytes
at module load
address reveals
PE file format

```
>>> dis(3121477805)
0xba0df8ad 8bff          MOV EDI, EDI
0xba0df8af 55            PUSH EBP
0xba0df8b0 8bec          MOV EBP, ESP
0xba0df8b2 a140a90dba    MOV EAX, [0xba0da940]
0xba0df8b7 85c0          TEST EAX, EAX
0xba0df8b9 b940bb0000    MOV ECX, 0xbb40
0xba0df8be 7404          JZ 0xba0df8c4
0xba0df8c0 3bc1          CMP EAX, ECX
0xba0df8c2 7523          JNZ 0xba0df8e7
0xba0df8c4 8b1598a40dba  MOV EDX, [0xba0da498]
0xba0df8ca b840a90dba    MOV EAX, 0xba0da940
0xba0df8cf c1e808        SHR EAX, 0x8
0xba0df8d2 3302          XOR EAX, [EDX]
0xba0df8d4 25fffff0000  AND EAX, 0xfffff
0xba0df8d9 a340a90dba    MOV [0xba0da940], EAX
0xba0df8de 7507          JNZ 0xba0df8e7
0xba0df8e0 8bc1          MOV EAX, ECX
0xba0df8e2 a340a90dba    MOV [0xba0da940], EAX
0xba0df8e7 f7d0          NOT EAX
0xba0df8e9 a33ca90dba    MOV [0xba0da93c], EAX
0xba0df8ee 5d            POP EBP
0xba0df8ef e992f7ffff    JMP 0xba0df086
0xba0df8f4 b8790000000   MOV EAX, 0x79
0xba0df8f9 0000          ADD [EAX], AL
0xba0df8fb 0000          ADD [EAX], AL
0xba0df8fd 0000          ADD [EAX], AL
0xba0df8ff 0016          ADD [ESI], DL
0xba0df901 810000082400  ADD DWORD [EAX], 0x240800
0xba0df907 0030          ADD [EAX], DH
0xba0df909 7900          JNS 0xba0df90b
0xba0df90b 0000          ADD [EAX], AL
0xba0df90d 0000          ADD [EAX], AL
0xba0df90f 0000          ADD [EAX], AL
0xba0df911 0000          ADD [EAX], AL
0xba0df913 008e84000080  ADD [ESI-0x7fffff7c], CL
0xba0df919 2300          AND EAX, [EAX]
0xba0df91b 0000          ADD [EAX], AL
0xba0df91d 0000          ADD [EAX], AL
0xba0df91f 0000          ADD [EAX], AL
0xba0df921 0000          ADD [EAX], AL
0xba0df923 0000          ADD [EAX], AL
0xba0df925 0000          ADD [EAX], AL
0xba0df927 0000          ADD [EAX], AL
0xba0df929 0000          ADD [EAX], AL
0xba0df92b 0000          ADD [EAX], AL
```

Disassembly at module entry point

Automatic Dereferencing!

```
1. Python  
=>>> for x in getmods():  
...     y=x.dereference_as("_LDR_DATA_TABLE_ENTRY")  
...     print y.FullDllName,y.DllBase,y.EntryPoint  
...  
\WINDOWS\system32\ntkrnlpa.exe 2152558592 2154366176  
\WINDOWS\system32\hal.dll 2154631168 2154750136  
\WINDOWS\system32\KDCOM.DLL 3126493184 3126496486  
\WINDOWS\system32\BOOTVID.dll 3125510144 3125516402  
ACPI.sys 3120009216 3120177241
```

```
1. Python  
=>>> for x in getmods():  
...     print x.FullDllName,x.DllBase, x.EntryPoint  
...  
\WINDOWS\system32\ntkrnlpa.exe 2152558592 2154366176  
\WINDOWS\system32\hal.dll 2154631168 2154750136  
\WINDOWS\system32\KDCOM.DLL 3126493184 3126496486  
\WINDOWS\system32\BOOTVID.dll 3125510144 3125516402  
ACPI.sys 3120009216 3120177241
```

1. Python

```
bigjoe:volatility golden$ python vol.py --profile=LinuxNilesProfilex64 -f /Volumes/SLOWDATA/IMAGES-THUMB-BACKUP/MEMIMAGES/niles.lime linux_volvshell <
Volatility Foundation Volatility Framework 2.4
Current context: process init, pid=1 DTB=0x30991e000
Welcome to volshell! Current memory image is:
file:///Volumes/SLOWDATA/IMAGES-THUMB-BACKUP/MEMIMAGES/niles.lime
To get help, type 'hh()'
>>> hh()

Use addrspace() for Kernel/Virtual AS
Use addrspace().base for Physical AS
Use proc() to get the current process object
    and proc().get_process_address_space() for the current process AS
    and proc().get_load_modules() for the current process DLLs

addrspace()                      : Get the current kernel/virtual address space.
cc(offset=None, pid=None, name=None, physical=False) : Change current shell context.
db(address, length=128, space=None)      : Print bytes as canonical hexdump.
dd(address, length=128, space=None)      : Print dwds at address.
dis(address, length=128, space=None, mode=None) : Disassemble code at a given address.
dq(address, length=128, space=None)      : Print qwords at address.
dt(objct, address=None, space=None, recursive=False, depth=0) : Describe an object or show type info.
getmods()                          : Generator for kernel modules (scripting).
getprocs()                         : Generator of process objects (scripting).
hh(cmd=None)                      : Get help on a command.
list_entry(head, objname, offset=-1, fieldname=None, forward=True) : Traverse a _LIST_ENTRY.
modules()                           : Print loaded modules in a table view.
proc()                             : Get the current process object.
ps()                               : Print active processes in a table view.
sc()                               : Show the current context.

For help on a specific command, type 'hh(<command>)'

>>> 
```

1. Python

```
>>> dt("task_struct")
'task_struct' (6120 bytes)
0x0  : state          ['long']
0x8  : stack          ['pointer', ['void']]
0x10 : usage           ['__unnamed_0x346']
0x14 : flags           ['unsigned int']
0x18 : ptrace          ['unsigned int']
0x20 : wake_entry      ['llist_node']
0x28 : on_cpu          ['int']
0x30 : last_wakee     ['pointer', ['task_struct']]
0x38 : wakee_flips    ['unsigned long']
0x40 : wakee_flip_decay_ts  ['unsigned long']
0x48 : wake_cpu        ['int']
0x4c : on_rq           ['int']
0x50 : prio            ['int']
0x54 : static_prio     ['int']
0x58 : normal_prio     ['int']
0x5c : rt_priority     ['unsigned int']
0x60 : sched_class     ['pointer', ['sched_class']]
0x68 : se               ['sched_entity']
0x1e0 : rt              ['sched_rt_entity']
0x210 : sched_task_group  ['pointer', ['task_group']]
0x218 : preempt_notifiers  ['hlist_head']
0x220 : btrace_seq      ['unsigned int']
0x224 : policy          ['unsigned int']
0x228 : nr_cpus_allowed  ['int']
0x230 : cpus_allowed     ['cpumask']
0x250 : sched_info       ['sched_info']
0x270 : tasks            ['list_head']
0x280 : pushable_tasks   ['plist_node']
0x2a8 : mm               ['pointer', ['mm_struct']]
0x2b0 : active_mm        ['pointer', ['mm_struct']]
```

```
1. Python  
->>> cc(name="bash")  
Multiple processes match name bash, please specify by PID or offset  
Matching processes:  
Name          PID    Offset  
bash           1403  0xfffff880035a3c7d0  
bash           1464  0xfffff8803098d8000  
bash           1542  0xfffff8802fec42fe0  
bash           1565  0xfffff8802fec447d0  
bash           1635  0xfffff8802ff6a5fc0  
bash           1714  0xfffff88030d9147d0  
bash           1767  0xfffff880309ee0000  
bash           1566  0xfffff880035fd8000  
->>> cc(pid=1542)  
Current context: process bash, pid=1542 DTB=0x2fec09000  
->>> print proc().pid  
1542  
->>> 
```

Acquiring Memory Dumps

Physical Memory Acquisition

- **dd** command
 - Windows, use \.\PhysicalMemory device
 - Not usable in user-space in XP SP2+
 - Linux:
 - /dev/mem
 - Physical memory
 - /proc/kcore
 - Kernel virtual memory
 - Problematic or gone
- Mostly deprecated
- Might be useful on very old machines
- Now, mostly custom tools

Generating Physical Memory Dumps

- Typical operation:
 - Load kernel driver
 - Discover ranges of physical address space associated with RAM
 - Avoid areas associated with, e.g., devices!
 - Map pages into virtual memory
 - Copy pages to removable storage
- Some tools, like LiME, dump directly from kernel module
- Others expose device and then dd is used

Physical Memory Acquisition

- Linux (all open source):
 - LiME (Joe Sylve, UNO, M.S. work)
 - Read source code to better understand dumping
 - Which we'll do...in three minutes
 - fmem
- Mac:
 - OSXPmem
 - Memoryze
 - ATC memory dumper
 - One problem is slow support for Mavericks →
- Windows gets its own slide

Physical Memory Acquisition (2)

- Mandiant Memoryze
- HBGary FastDump
- GMG Systems, Inc. KnTTools
- F-Response
- MoonSols Windows Memory Toolkit
- AccessData FTK Imager
- Belkasoft Live RAM Capturer
- ATC-NY Windows Memory Reader
- Windows pmem
- others

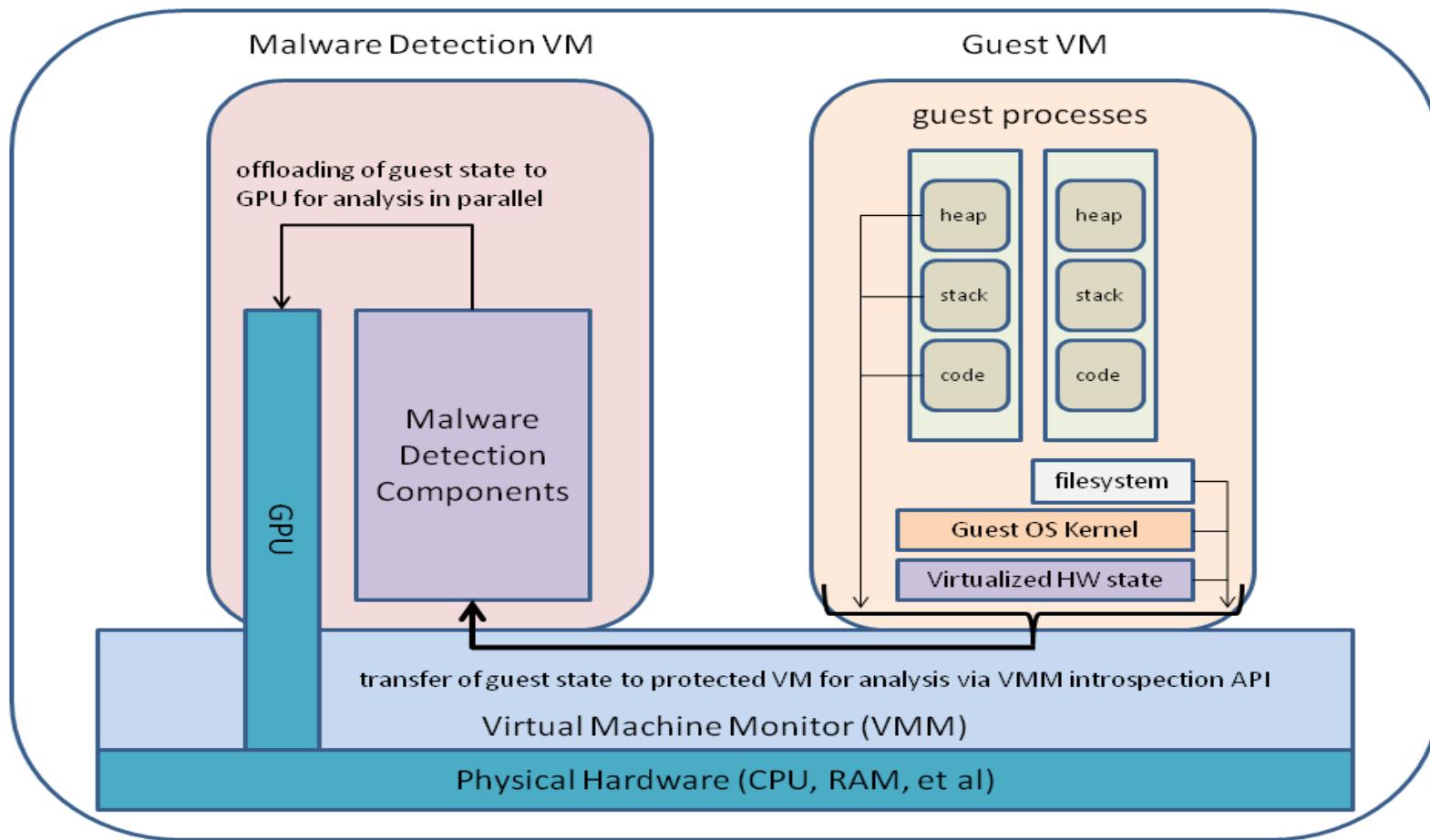
Memory Acquisition Issues

- Software-based solutions for memory imaging typically require loading software
- Probably erases some evidence
- Requires at least limited trust of the OS
- Malware, etc.
- Hardware-based solutions
 - **Tribble**
 - Brian Carrier
 - **Firewire hacks**
 - Maximillian Dornseif, metlstorm
 - **Hardware must typically be in place**

A Special Case: Virtualization

- Target is virtualized:
 - Suspending VM on VMWare Workstation / Fusion yields physical memory dump (*.vmem)
 - Can clone to investigate
 - Or use introspection (next slides)
 - And use the VM's memory dump with Volatility
 - .vmem file is also memory mapped!
- Target is a traditional computer
 - Also consider conversion of copies of physical disks to virtual disks via raw2vmdk, so a clone of system can be booted for investigative purposes

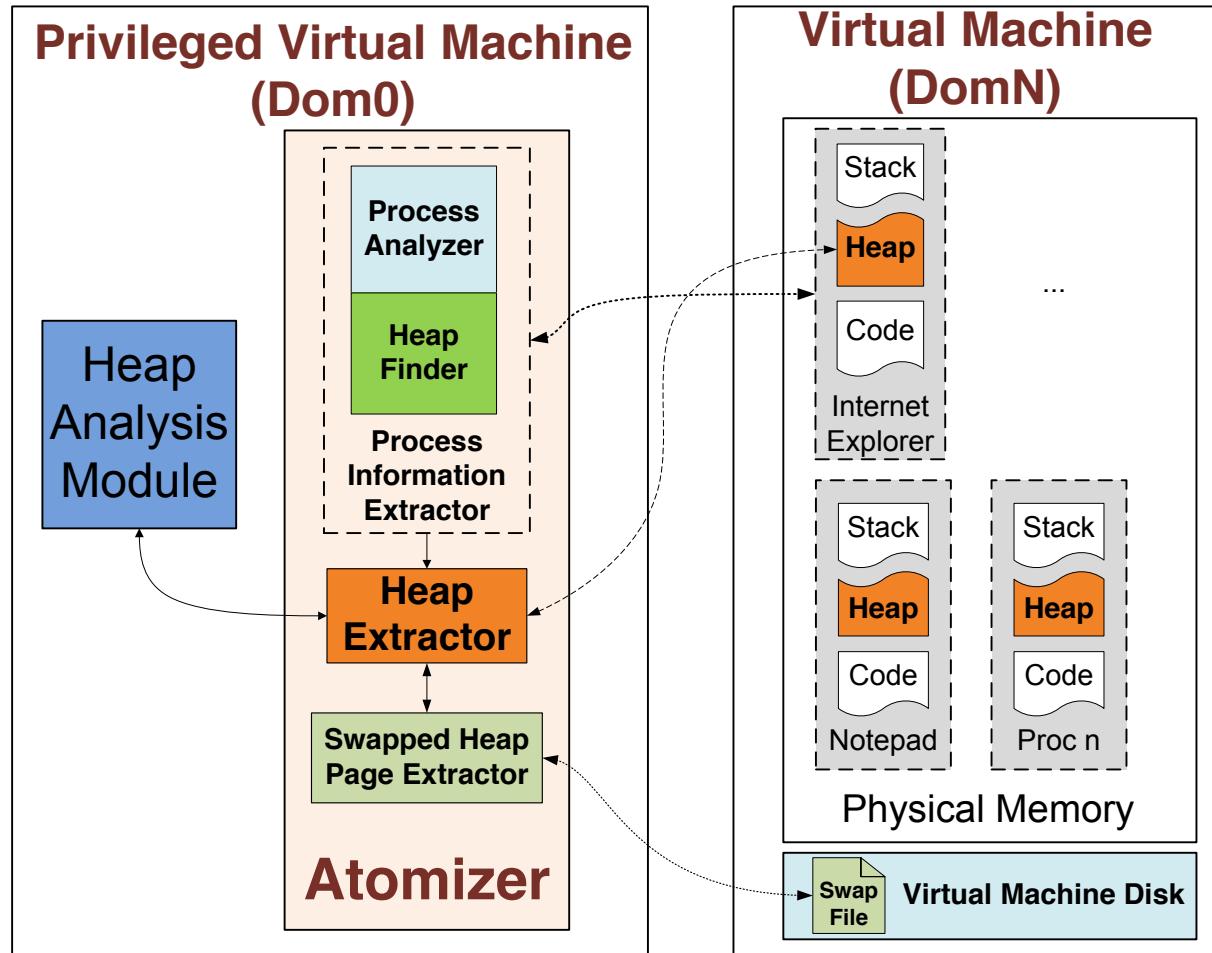
VM Introspection



“TC-Small-Virtual Machine Introspection-based Live Forensics for Detection of Software,”

National Science Foundation, PI: Golden G. Richard III.

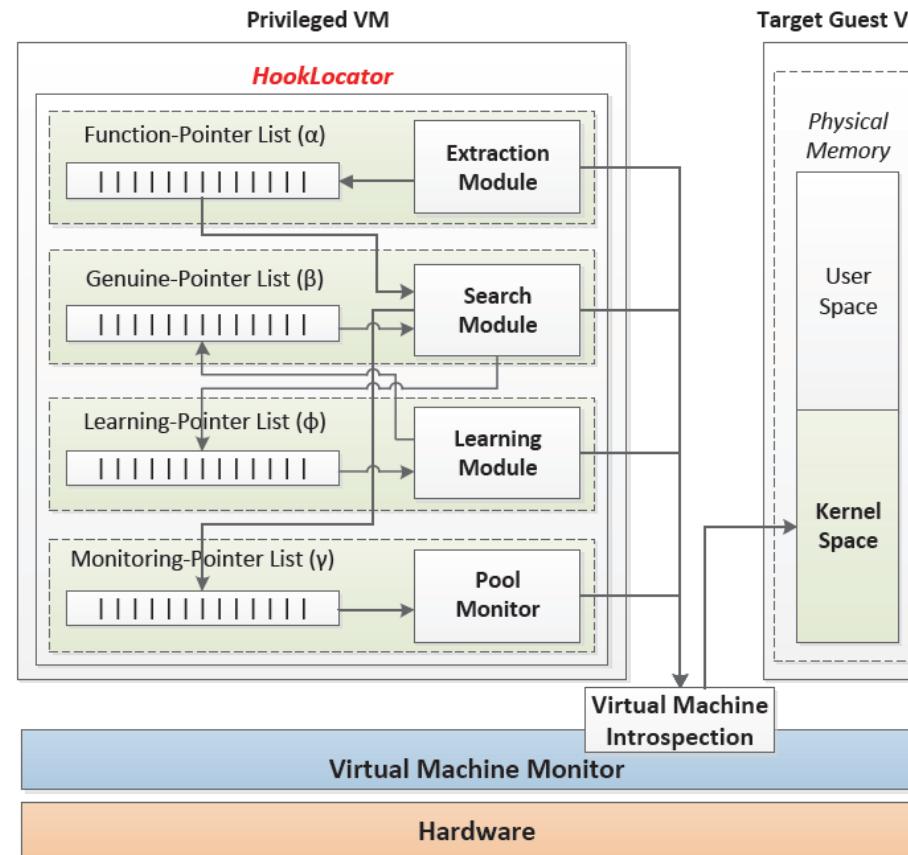
Application Analysis via VM Introspection



e.g., to detect heap spraying attacks

S. Javaid, A. Zoranic, I. Ahmed, G. G. Richard III, "Atomizer: A Fast, Scalable and Lightweight Heap Analyzer for Virtual Machines in a Cloud Environment," Proceedings of the 6th Layered Assurance Workshop (LAW'12), December 2012, Orlando, Florida.

OS Analysis via VM Introspection



I. Ahmed, G. G. Richard III, A. Zoranic, V. Roussev, "Integrity Checking of Function Pointers in Kernel Pools via Virtual Machine Introspection," Proceedings of the 16th Information Security Conference (ISC 2013), Best Paper Award, Dallas, TX.

Memory Acquisition Hazards

- Can't just scan entire physical address and copy
- Areas of physical address space dedicated to memory-mapped device I/O
- Touching these areas can freak out the devices and cause reboots, hangs, etc.
- Typically, touch only areas dedicated to OS / applications



For critical situations / real investigations:

**Test tools on similar non-critical
hardware!**

They don't all work (surprise)!

LiME Internals



**Understanding how the tools work
is important, too...**

```
$ git clone https://github.com/504ensicsLabs/LiME.git
```

```
$ less /proc/iomem
```

on a Linux machine to get an idea of what LiME is up to...

```
int init_module (void) {
    if(!path) {
        DBG("No path parameter specified");
        return -EINVAL;
    }
    if(!format) {
        DBG("No format parameter specified");
        return -EINVAL;
    }
    memset(zero_page, 0, sizeof(zero_page));
    if (!strcmp(format, "raw")) mode = LIME_MODE_RAW;
    else if (!strcmp(format, "lime"))
        mode = LIME_MODE_LIME;
    else if (!strcmp(format, "padded"))
        mode = LIME_MODE_PADDED;
    else {
        DBG("Invalid format parameter specified.");
        return -EINVAL;
    }
    method = (sscanf(path, "tcp:%d", &port) == 1)
        ? LIME_METHOD_TCP : LIME_METHOD_DISK;
    return init();
}
```

```
static int init() {
    struct resource *p;
    int err = 0;
    if((err = setup())) {
        cleanup();
        return err;
    }
    for (p = iomem_resource.child; p ; p = p->sibling) {
        if (strcmp(p->name, LIME_RAMSTR)) // "System RAM"
            continue;
        if (mode == LIME_MODE_LIME &&
            (err = write_lime_header(p))) {
            break;
        } else if (mode == LIME_MODE_PADDED &&
                   (err = write_padding(
                       (size_t)((p->start - 1)-p_last)))) {
            break;
        }
        if ((err = write_range(p))) {
            break;
        }
        p_last = p->end;
    }
    cleanup();
    return err;
}
```

```
static int write_lime_header(struct resource * res) {100
    long s;

    lime_mem_range_header header;

    memset(&header, 0, sizeof(lime_mem_range_header));
    header.magic = LIME_MAGIC;
    header.version = 1;
    header.s_addr = res->start;
    header.e_addr = res->end;

    s = write_vaddr(&header,
                    sizeof(lime_mem_range_header));

    if (s != sizeof(lime_mem_range_header)) {
        DBG("Error sending header %ld", s);
        return (int) s;
    }

    return 0;
}
```

```
static int write_range(struct resource * res) {  
    struct page *p;  
    void *v;  
    int s;  
    for (i = res->start; i <= res->end; i += is) {  
        p = pfn_to_page((i) >> PAGE_SHIFT);  
        is = min((size_t) PAGE_SIZE,  
                 (size_t) (res->end - i + 1));  
        if (is < PAGE_SIZE) {  
            write_padding(is);  
        } else {  
            v = kmap(p);  
            s = write_vaddr(v, is);  
            kunmap(p);  
        }  
    }  
    return 0;  
}
```

```
static int write_vaddr(void * v, size_t is) {102
    return (method == LIME_METHOD_TCP) ?
        write_vaddr_tcp(v, is) :
write_vaddr_disk(v, is);
}

static int setup(void) {
    return (method == LIME_METHOD_TCP) ?
        setup_tcp() :
        setup_disk();
}

static void cleanup(void) {
    return (method == LIME_METHOD_TCP) ?
        cleanup_tcp() : cleanup_disk();
}
```

```
int setup_disk() {
    mm_segment_t fs;
    int err;
    fs = get_fs();
    set_fs(KERNEL_DS);
    f = filp_open(path,
                  O_WRONLY |
                  O_CREAT |
                  O_LARGEFILE |
                  O_TRUNC |
                  O_SYNC |
                  O_DIRECT, 0444);

    ...
    set_fs(fs);
    return 0;
}
```

```
int write_vaddr_disk(void * v, size_t is) {104
    mm_segment_t fs;
    long s;
    loff_t pos;

    fs = get_fs();
    set_fs(KERNEL_DS);
    pos = f->f_pos;
    s = vfs_write(f, v, is, &pos);
    if (s == is) {
        f->f_pos = pos;
    }
    set_fs(fs);
    return s;
}
```

```
void cleanup_disk() {
    mm_segment_t fs;

    fs = get_fs();
    set_fs(KERNEL_DS);
    if(f) filp_close(f, NULL);
    set_fs(fs);
}
```

winpmem

```
1  /*
2   Copyright 2012 Michael Cohen <scudette@gmail.com>
3
4   Licensed under the Apache License, Version 2.0 (the "License");
5   you may not use this file except in compliance with the License.
6   You may obtain a copy of the License at
7
8       http://www.apache.org/licenses/LICENSE-2.0
9
10  Unless required by applicable law or agreed to in writing, software
11  distributed under the License is distributed on an "AS IS" BASIS,
12  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13  See the License for the specific language governing permissions and
14  limitations under the License.
15 */
```

```
30
31 NTSTATUS IoUnload(IN PDRIVER_OBJECT DriverObject) {
32     UNICODE_STRING DeviceLinkUnicodeString;
33     NTSTATUS NtStatus;
34     PDEVICE_OBJECT pDeviceObject = DriverObject->DeviceObject;
35
36     RtlInitUnicodeString (&DeviceLinkUnicodeString, L"\DosDevices\\"
37                           PMEM_DEVICE_NAME);
38     NtStatus = IoDeleteSymbolicLink (&DeviceLinkUnicodeString);
39
40     if (DriverObject != NULL) {
41         IoDeleteDevice(pDeviceObject);
42     }
43
44     return NtStatus;
45 }
46 }
```

```
48 /*
49  Gets information about the memory layout.
50
51  - The Physical memory address ranges.
52 */
53 int AddMemoryRanges(struct PmemMemroyInfo *info, int len) {
54     PPHYSICAL_MEMORY_RANGE MmPhysicalMemoryRange;
55     int number_of_runs = 0;
56     int required_length;
57
58     // Enumerate address ranges.
59     MmPhysicalMemoryRange = MmGetPhysicalMemoryRanges();
60
61     if (MmPhysicalMemoryRange == NULL) {
62         return -1;
63     }
64
65     /** Find out how many ranges there are. */
66     for(number_of_runs=0;
67         (MmPhysicalMemoryRange[number_of_runs].BaseAddress.QuadPart) ||
68         (MmPhysicalMemoryRange[number_of_runs].NumberOfBytes.QuadPart);
69         number_of_runs++);
70
71     required_length = (sizeof(struct PmemMemroyInfo) +
72                         number_of_runs * sizeof(PHYSICAL_MEMORY_RANGE));
73 }
```

```
74 /* Do we have enough space? */
75 if(len < required_length) {
76     return -1;
77 }
78
79 // Ensure exception bubbles up here to be caught by our caller.
80 //ProbeForWrite(info, required_length, 1);
81
82 RtlZeroMemory(info, required_length);
83
84 info->NumberOfRuns.QuadPart = number_of_runs;
85 RtlCopyMemory(&info->Run[0], MmPhysicalMemoryRange,
86                 number_of_runs * sizeof(PHYSICAL_MEMORY_RANGE));
87
88 ExFreePool(MmPhysicalMemoryRange);
89
90 return required_length;
91 };
92
```

```
94 static NTSTATUS wddCreate(IN PDEVICE_OBJECT DeviceObject, IN PIRP Irp) {
95     PDEVICE_EXTENSION ext=(PDEVICE_EXTENSION)DeviceObject->DeviceExtension;
96     PIO_STACK_LOCATION IrpStack = IoGetCurrentIrpStackLocation(Irp);
97
98     ext->MemoryHandle = 0;
99
100    Irp->IoStatus.Status = STATUS_SUCCESS;
101    Irp->IoStatus.Information = 0;
102
103    IoCompleteRequest(Irp,IO_NO_INCREMENT);
104    return STATUS_SUCCESS;
105};
106
107 static NTSTATUS wddClose(IN PDEVICE_OBJECT DeviceObject, IN PIRP Irp) {
108     PDEVICE_EXTENSION ext=(PDEVICE_EXTENSION)DeviceObject->DeviceExtension;
109     PIO_STACK_LOCATION IrpStack = IoGetCurrentIrpStackLocation(Irp);
110
111     if(ext->MemoryHandle != 0) {
112         ZwClose(ext->MemoryHandle);
113     }
114
115     Irp->IoStatus.Status = STATUS_SUCCESS;
116     Irp->IoStatus.Information = 0;
117
118     IoCompleteRequest(Irp,IO_NO_INCREMENT);
119     return STATUS_SUCCESS;
120 }
```

```
122 NTSTATUS wddDispatchDeviceControl(IN PDEVICE_OBJECT DeviceObject,
123                                     IN PIRP Irp)
124 {
125     UNICODE_STRING DestinationPath;
126     PIO_STACK_LOCATION IrpStack;
127     NTSTATUS status = STATUS_SUCCESS;
128     ULONG IoControlCode;
129     PVOID IoBuffer;
130     PULONG OutputBuffer;
131     PDEVICE_EXTENSION ext;
132     ULONG InputLen, OutputLen;
133
134     ULONG Level;
135     ULONG Type;
136
137     // We must be running in PASSIVE_LEVEL or we bluescreen here. We
138     // theoretically should always be running at PASSIVE_LEVEL here, but
139     // in case we ended up here at the wrong IRQL its better to bail
140     // than to bluescreen.
141     if(KeGetCurrentIrql() != PASSIVE_LEVEL) {
142         status = STATUS_ABANDONED;
143         goto exit;
144     }
145
146     ext = (PDEVICE_EXTENSION)DeviceObject->DeviceExtension;
147 }
```

```
148 Irp->IoStatus.Information = 0;
149
150 IrpStack = IoGetCurrentIrpStackLocation(Irp);
151
152 IoBuffer = Irp->AssociatedIrp.SystemBuffer;
153 OutputLen = IrpStack->Parameters.DeviceIoControl.OutputBufferLength;
154 InputLen = IrpStack->Parameters.DeviceIoControl.InputBufferLength;
155 IoControlCode = IrpStack->Parameters.DeviceIoControl.IoControlCode;
156
157 switch ((IoControlCode & 0xFFFFF0F)) {
158
159     // Return information about memory layout etc through this ioctl.
160 case IOCTL_GET_INFO: {
161     struct PmemMemroyInfo *info = (void *)IoBuffer;
162     int res;
163     IMAGE_DOS_HEADER *KernBase;
164
165     // Check we have enough room here.
166     if (OutputLen < sizeof(struct PmemMemroyInfo)) {
167         status = STATUS_INFO_LENGTH_MISMATCH;
168         break;
169     }
170
171     __try {
172         res = AddMemoryRanges(info, OutputLen);
173     } __except(EXCEPTION_EXECUTE_HANDLER) {
174         status = STATUS_INFO_LENGTH_MISMATCH;
175         break;
176     }
```

```
178 WinDbgPrint("Returning info on the system memory.\n");
179 if(res > 0) {
180     KernBase = KernelGetModuleBaseByPtr(NtBuildNumber, "NtBuildNumber");
181
182     // We are currently running in user context which means __readcr3() will
183     // return the process CR3. So we return the kernel CR3 we found
184     // when loading. Alternatively we could get the kernel DTB from
185     // the KDBG but that is less accurate.
186     info->CR3.QuadPart = CR3.QuadPart;
187
188     info->NtBuildNumber.QuadPart = *NtBuildNumber;
189
190     // Fill in KPCR.
191     GetKPCR(info);
192
193     // The kernel base.
194     info->KernBase.QuadPart = (uintptr_t)KernBase;
195     info->KDBG.QuadPart = (uintptr_t)KDBGScan(KernBase);
196
197     Irp->IoStatus.Information = res;
198     status = STATUS_SUCCESS;
199 } else {
200     status = STATUS_INFO_LENGTH_MISMATCH;
201 }
202 }; break;
```

```
204 case IOCTL_SET_MODE: {
205     WinDbgPrint("Setting Acquisition mode.\n");
206
207     if (InputLen == sizeof(struct PmemMemoryControl)) {
208         struct PmemMemoryControl *ctrl = (void *)IoBuffer;
209
210         ext->mode = ctrl->mode;
211
212         switch(ctrl->mode) {
213             case ACQUISITION_MODE_PHYSICAL_MEMORY:
214                 WinDbgPrint("Using physical memory device for acquisition.\n");
215                 break;
216
217             case ACQUISITION_MODE_MAP_IO_SPACE:
218                 WinDbgPrint("Using MmMapIoSpace for acquisition.\n");
219                 break;
220
221             default:
222                 WinDbgPrint("Invalid acquisition mode %d.\n", ctrl->mode);
223                 status = STATUS_INVALID_PARAMETER;
224             };
225
226     } else {
227         status = STATUS_INFO_LENGTH_MISMATCH;
228     };
229 }; break;
230 }
```

```
231 #if PMEM_WRITE_ENABLED
232     case IOCTL_WRITE_ENABLE: {
233         ext->WriteEnabled = !ext->WriteEnabled;
234         WinDbgPrint("Write mode is %d. Do you know what you are doing?\n",
235                     ext->WriteEnabled);
236     }; break;
237 #endif
238
239     default: {
240         WinDbgPrint("Invalid IOCTL %d\n", IoControlCode);
241         status = STATUS_INVALID_PARAMETER;
242     };
243 }
244
245 exit:
246     Irp->IoStatus.Status = status;
247     IoCompleteRequest(Irp, IO_NO_INCREMENT);
248     return status;
249 }
```

```
252 NTSTATUS DriverEntry (IN PDRIVER_OBJECT DriverObject,
253                         IN PUNICODE_STRING RegistryPath)
254 {
255     UNICODE_STRING DeviceName, DeviceLink;
256     NTSTATUS NtStatus;
257     PDEVICE_OBJECT DeviceObject = NULL;
258     PDEVICE_EXTENSION extension;
259
260     WinDbgPrint("WinPMEM - " PMEM_VERSION " - Physical memory acquisition\n");
261
262 #if PMEM_WRITE_ENABLED
263     WinDbgPrint("WinPMEM write support available!");
264 #endif
265
266     WinDbgPrint("Copyright (c) 2012, Michael Cohen <scudette@gmail.com> based "
267                 "on win32dd code by Matthieu Suiche <http://www.msuiche.net>\n");
268
269     RtlInitUnicodeString (&DeviceName, L"\\Device\\" PMEM_DEVICE_NAME);
270
```

```
271 // We create our secure device.  
272 // http://msdn.microsoft.com/en-us/library/aa490540.aspx  
273 NtStatus = IoCreateDeviceSecure(DriverObject,  
274                             sizeof(DEVICE_EXTENSION),  
275                             &DeviceName,  
276                             FILE_DEVICE_UNKNOWN,  
277                             FILE_DEVICE_SECURE_OPEN,  
278                             FALSE,  
279                             &SDDL_DEVOBJ_SYS_ALL ADM_ALL,  
280                             &GUID_DEVCLASS_PMEM_DUMPER,  
281                             &DeviceObject);  
282  
283 if (!NT_SUCCESS(NtStatus)) {  
284     WinDbgPrint ("IoCreateDevice failed. => %08X\n", NtStatus);  
285     return NtStatus;  
286 }  
287  
288 DriverObject->MajorFunction[IRP_MJ_CREATE] = wddCreate;  
289 DriverObject->MajorFunction[IRP_MJ_CLOSE] = wddClose;  
290 DriverObject->MajorFunction[IRP_MJ_DEVICE_CONTROL] = wddDispatchDeviceControl;  
291 DriverObject->MajorFunction[IRP_MJ_READ] = PmemRead;  
292  
293 #if PMEM_WRITE_ENABLED == 1  
294     // Support writing.  
295     DriverObject->MajorFunction[IRP_MJ_WRITE] = PmemWrite;  
296 #endif  
297     DriverObject->DriverUnload = IoUnload;
```

```
299 // Use buffered IO - a bit slower but simpler to implement, and more
300 // efficient for small reads.
301 SetFlag(DeviceObject->Flags, DO_BUFFERED_IO );
302 ClearFlag(DeviceObject->Flags, DO_DIRECT_IO );
303 ClearFlag(DeviceObject->Flags, DO_DEVICE_INITIALIZING);
304
305 RtlInitUnicodeString (&DeviceLink, L"\DosDevices\\" PMEM_DEVICE_NAME);
306
307 NtStatus = IoCreateSymbolicLink (&DeviceLink, &DeviceName);
308
309 if (!NT_SUCCESS(NtStatus)) {
310     WinDbgPrint ("IoCreateSymbolicLink failed. => %08X\n", NtStatus);
311     IoDeleteDevice (DeviceObject);
312 }
313
314 // Populate globals in kernel context.
315 CR3.QuadPart = __readcr3();
316
317 // Initialize the device extension with safe defaults.
318 extension = DeviceObject->DeviceExtension;
319 extension->mode = ACQUISITION_MODE_PHYSICAL_MEMORY;
320
321 return NtStatus;
322 }
```

```
19 static int EnsureExtensionHandle(PDEVICE_EXTENSION extension) {
20     NTSTATUS NtStatus;
21     UNICODE_STRING PhysicalMemoryPath;
22     OBJECT_ATTRIBUTES MemoryAttributes;
23
24     /* Make sure we have a valid handle now. */
25     if(!extension->MemoryHandle) {
26         RtlInitUnicodeString(&PhysicalMemoryPath, L"\Device\PhysicalMemory");
27
28         InitializeObjectAttributes(&MemoryAttributes,
29             &PhysicalMemoryPath,
30             OBJ_KERNEL_HANDLE,
31             (HANDLE) NULL,
32             (PSECURITY_DESCRIPTOR) NULL);
33
34         NtStatus = ZwOpenSection(&extension->MemoryHandle,
35             SECTION_MAP_READ, &MemoryAttributes);
36
37         if (!NT_SUCCESS(NtStatus)) {
38             WinDbgPrint("Failed ZwOpenSection(MemoryHandle) => %08X\n", NtStatus);
39             return 0;
40         }
41     };
42
43     return 1;
44 }
```

```
130 NTSTATUS PmemRead(IN PDEVICE_OBJECT DeviceObject, IN PIRP Irp) {  
131     PVOID Buf;          //Buffer provided by user space.  
132     ULONG Buflen;       //Buffer length for user provided buffer.  
133     LARGE_INTEGER BufOffset; // The file offset requested from userspace.  
134     ULONG DataLen;      //Buffer length for Driver Data Buffer  
135     PIO_STACK_LOCATION pIoStackIrp;  
136     PDEVICE_EXTENSION extension;  
137     NTSTATUS status = STATUS_SUCCESS;  
138  
139     // We must be running in PASSIVE_LEVEL or we bluescreen here. We  
140     // theoretically should always be running at PASSIVE_LEVEL here, but  
141     // in case we ended up here at the wrong IRQL its better to bail  
142     // than to bluescreen.  
143     if(KeGetCurrentIrql() != PASSIVE_LEVEL) {  
144         status = STATUS_ABANDONED;  
145         goto exit;  
146     };  
147  
148     extension = DeviceObject->DeviceExtension;  
149  
150     pIoStackIrp = IoGetCurrentIrpStackLocation(Irp);  
151     Buflen = pIoStackIrp->Parameters.Read.Length;  
152     BufOffset = pIoStackIrp->Parameters.Read.ByteOffset;  
153     Buf = (PCHAR)(Irp->AssociatedIrp.SystemBuffer);
```

```
155 switch(extension->mode) {  
156  
157     // Read using the physical memory handle.  
158     case ACQUISITION_MODE_PHYSICAL_MEMORY:  
159         status = DeviceRead(extension, BufOffset, Buf, &BufLen,  
160                             PhysicalMemoryPartialRead);  
161         Irp->IoStatus.Information = pIoStackIrp->Parameters.Read.Length;  
162         break;  
163  
164     case ACQUISITION_MODE_MAP_IO_SPACE:  
165         status = DeviceRead(extension, BufOffset, Buf, &BufLen,  
166                             MapIOPagePartialRead);  
167         Irp->IoStatus.Information = pIoStackIrp->Parameters.Read.Length;  
168         break;  
169  
170     default:  
171         WinDbgPrint("Acquisition mode %u not supported.\n", extension->mode);  
172         status = STATUS_NOT_IMPLEMENTED;  
173         BufLen = 0;  
174     }  
175  
176     exit:  
177     Irp->IoStatus.Status = status;  
178     IoCompleteRequest(Irp, IO_NO_INCREMENT);  
179  
180     return status;  
181 }
```

```
107 static NTSTATUS DeviceRead(IN PDEVICE_EXTENSION extension, LARGE_INTEGER offset,
108                             PCHAR buf, ULONG *count,
109                             LONG (*handler)(IN PDEVICE_EXTENSION, LARGE_INTEGER,
110                                         PCHAR, ULONG)) {
111     int remaining = *count;
112
113     while(remaining > 0) {
114         int result = handler(extension, offset, buf, remaining);
115
116         /* Error Occured. */
117         if(result < 0) return result;
118         /* No data available. */
119         if(result==0) break;
120
121         offset.QuadPart += result;
122         buf += result;
123         remaining -= result;
124     };
125
126     return STATUS_SUCCESS;
127 }
```

```
47 static LONG PhysicalMemoryPartialRead(IN PDEVICE_EXTENSION extension,
48                                     LARGE_INTEGER offset, PCHAR buf,
49                                     ULONG count) {
50     ULONG page_offset = offset.QuadPart % PAGE_SIZE;
51     ULONG to_read = min(PAGE_SIZE - page_offset, count);
52     PUCHAR mapped_buffer = NULL;
53     SIZE_T ViewSize = PAGE_SIZE;
54     NTSTATUS NtStatus;
55
56
57     if (EnsureExtensionHandle(extension)) {
58         /* Map page into the Kernel AS */
59         NtStatus = ZwMapViewOfSection(extension->MemoryHandle, (HANDLE)-1,
60                                       &mapped_buffer, 0L, PAGE_SIZE, &offset,
61                                       &ViewSize, ViewUnmap, 0, PAGE_READONLY);
62
63         if (NT_SUCCESS(NtStatus)) {
64             RtlCopyMemory(buf, mapped_buffer + page_offset, to_read);
65             ZwUnmapViewOfSection((HANDLE)-1, mapped_buffer);
66
67         } else {
68             WinDbgPrint("Failed to Map page at 0x%llx\n", offset.QuadPart);
69             RtlZeroMemory(buf, to_read);
70         };
71     };
72
73     return to_read;
74 }
```

```
77 // Read a single page using MmMapIoSpace.
78 static LONG MapIOPagePartialRead(IN PDEVICE_EXTENSION extension,
79                                     LARGE_INTEGER offset, PCHAR buf,
80                                     ULONG count) {
81     ULONG page_offset = offset.QuadPart % PAGE_SIZE;
82     ULONG to_read = min(PAGE_SIZE - page_offset, count);
83     PUCHAR mapped_buffer = NULL;
84     SIZE_T ViewSize = PAGE_SIZE;
85     NTSTATUS NtStatus;
86     LARGE_INTEGER ViewBase;
87
88     // Round to page size
89     ViewBase.QuadPart = offset.QuadPart - page_offset;
90
91     // Map exactly one page.
92     mapped_buffer = MmMapIoSpace(ViewBase, PAGE_SIZE, MmNonCached);
93
94     if (mapped_buffer) {
95         RtlCopyMemory(buf, mapped_buffer + page_offset, to_read);
96     } else {
97         // Failed to map page, null fill the buffer.
98         RtlZeroMemory(buf, to_read);
99     }
100
101    MmUnmapIoSpace(mapped_buffer, PAGE_SIZE);
102
103    return to_read;
104 }
```

Volatility Memory Image Support

- Volatility supports a bunch of different memory formats, to cover output of most available tools
 - Raw
 - LiME format
 - ATC-NY Mac format
 - VirtualBox, VMWare, Fusion, Qemu, etc.
 - Windows crash dump
 - Hibernation files

Virtual \leftrightarrow Physical Address Translation

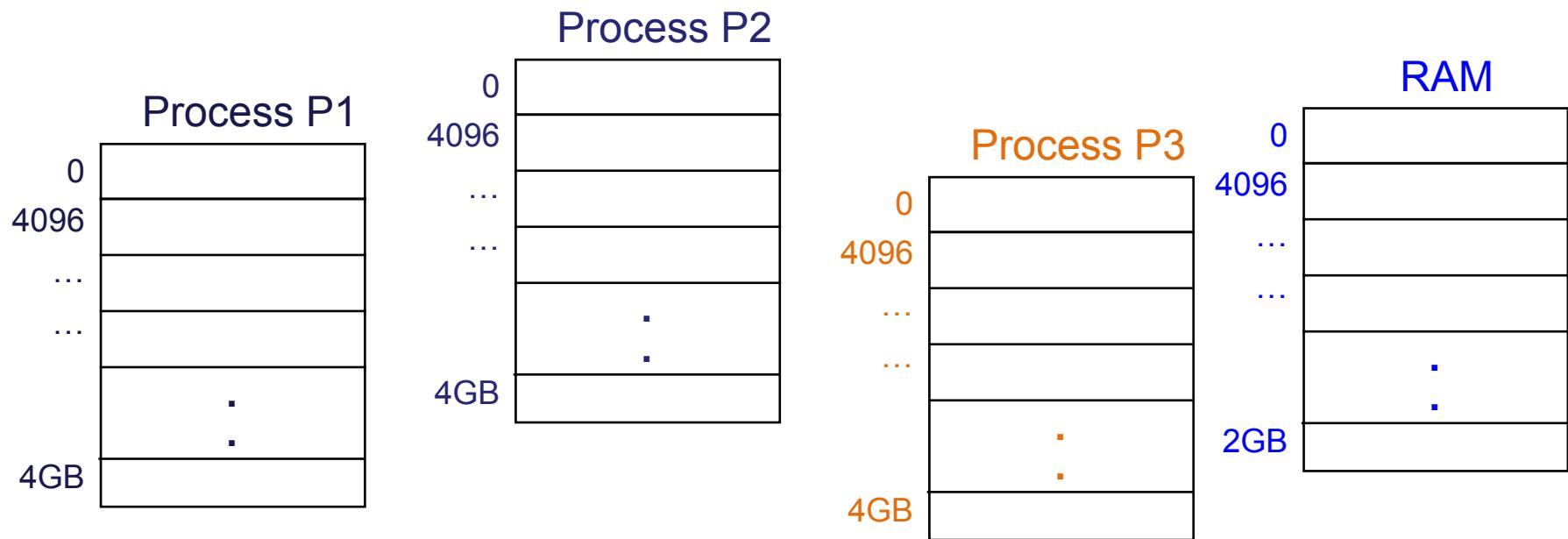
- Address translation is **crucial** in memory analysis
- Applications and OS use virtual addresses
- Must translate between virtual and physical addresses to provide context for physical memory dump regions
- Normally, largely handled by memory management hardware

Virtual Memory

- Provides contiguous, linear address space for processes
- Virtual address space is divided into fixed-size pages
 - Physical memory is divided into pages of same size
 - On x86 Intel, these are normally 4K
- OS keeps track of free and allocated pages
- Processes get only as many pages as they need

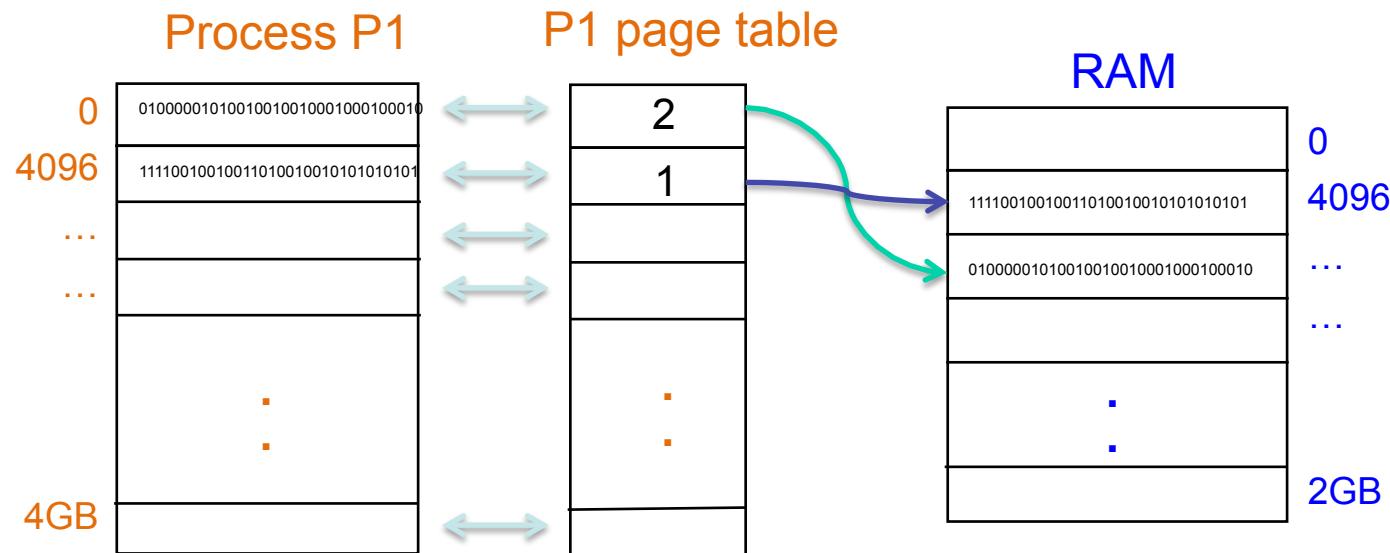
Logical vs. Physical Addresses

- Virtual address: Address whose scope is a particular process or OS kernel
- Physical address: Location in physical RAM

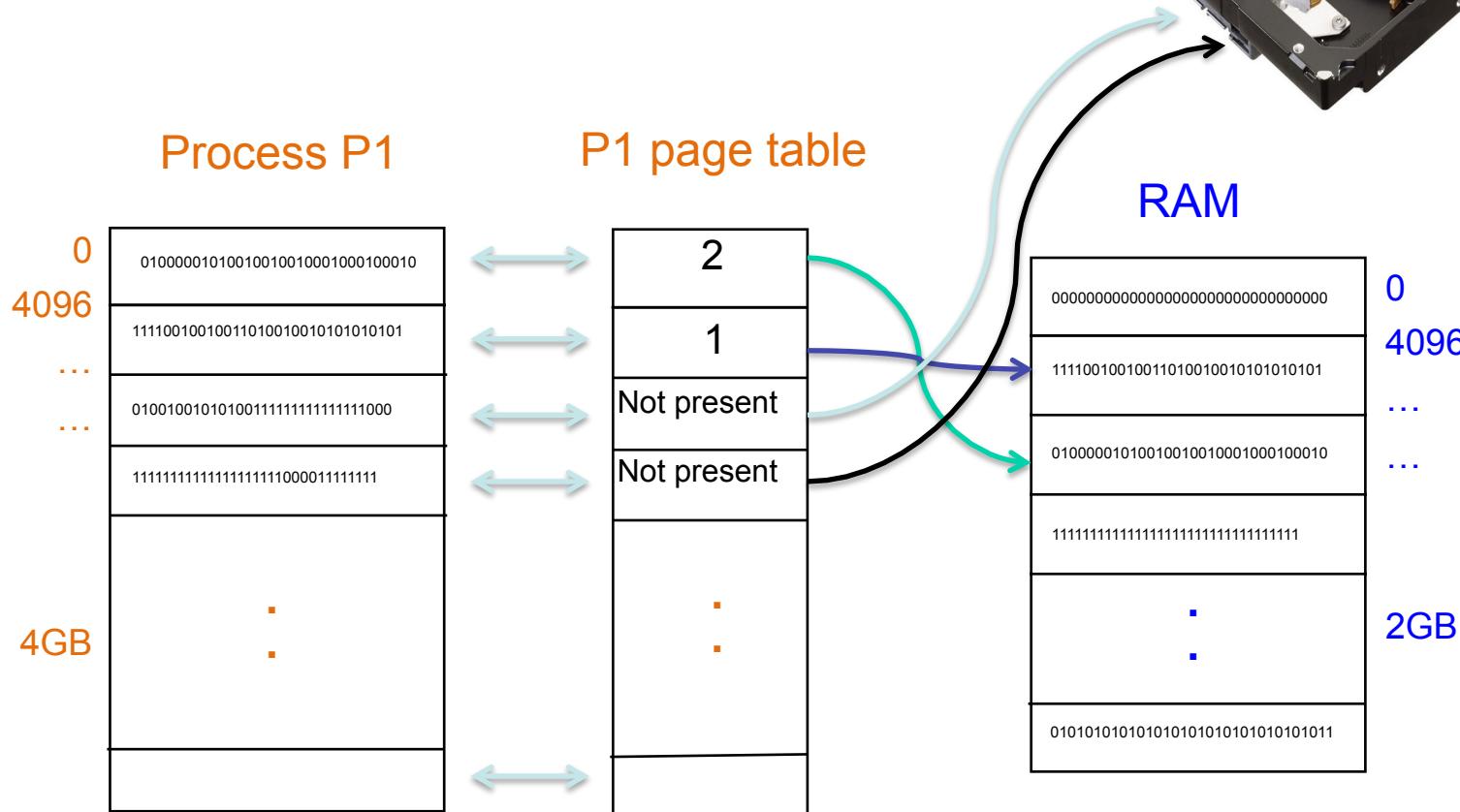


Paging: (Very) Simplified View

- Page tables (per process) are used to translate logical to physical addresses
- Pages for a particular process are generally not in contiguous order in RAM



Virtual Memory and Address Translation: Simplified Version



32-bit mode: 4K Pages

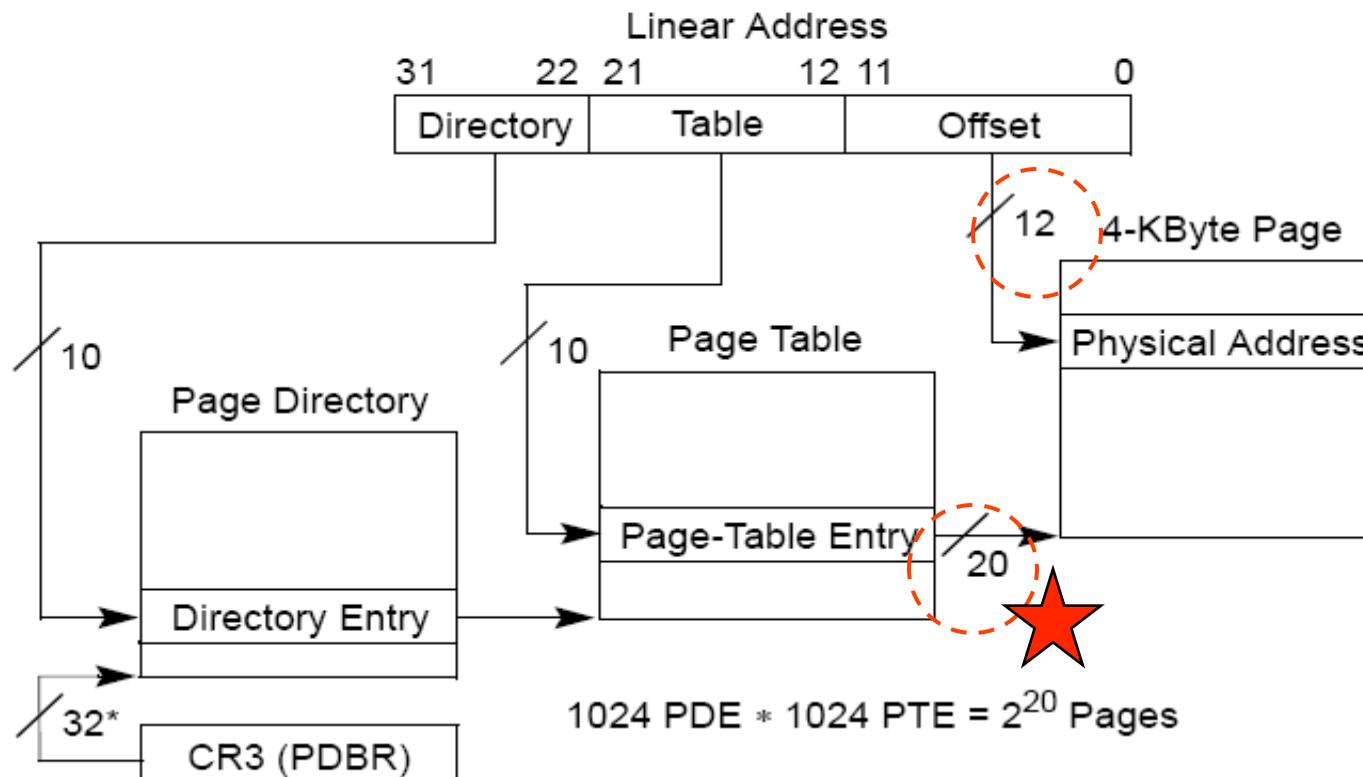
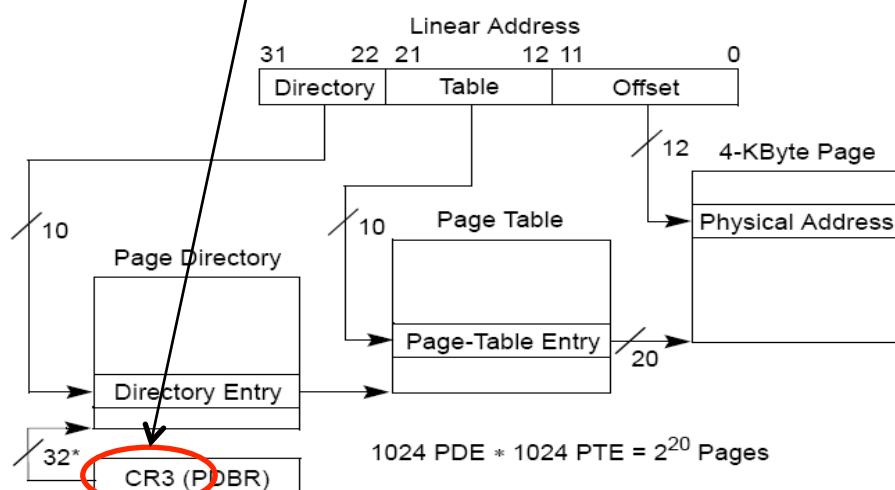


Figure 3-12. Linear Address Translation (4-KByte Pages)

```
struct task_struct {
    ...
    struct mm_struct *mm;
    ...
}
```

```
struct mm_struct {
    struct vm_area_struct *mmap;
    ...
    pgd_t *pgd;
    ...
}
```



```
struct vm_area_struct {
    unsigned long vm_start;
    unsigned long vm_end;
    ...
    struct vm_area_struct *vm_next,
    *vm_prev;
}
```

```
struct vm_area_struct {
    unsigned long vm_start;
    unsigned long vm_end;
    ...
    struct vm_area_struct *vm_next,
    *vm_prev;
}
```

• • •

32-bit mode: 4MB Pages

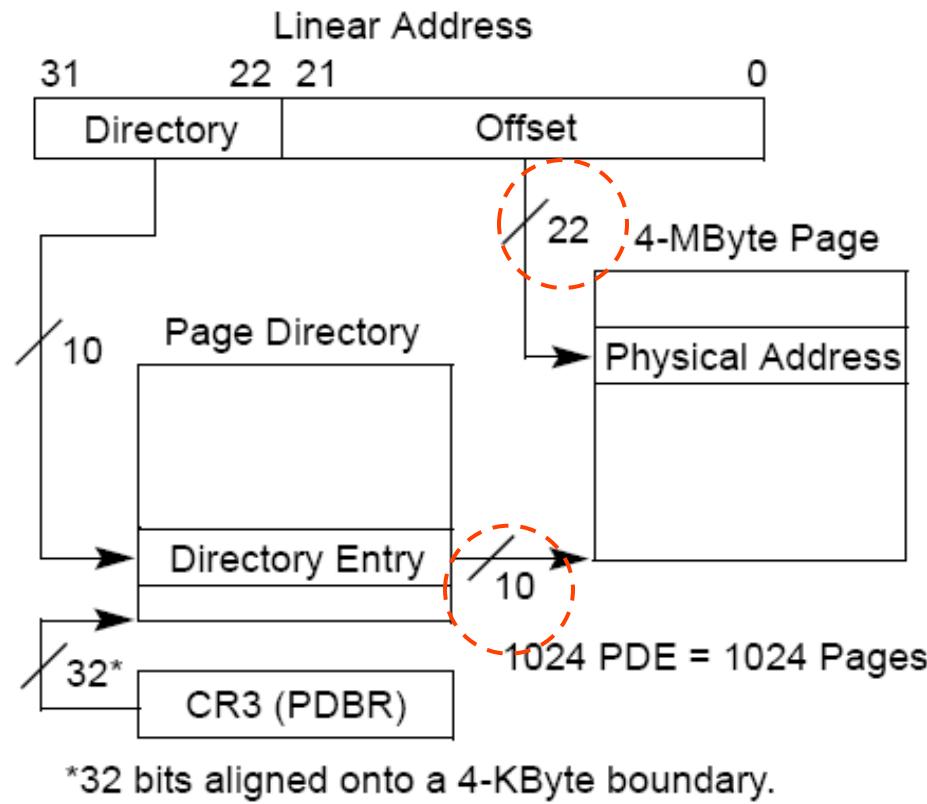


Figure 3-13. Linear Address Translation (4-MByte Pages)

32-bit Mode w/ PAE: 4K pages

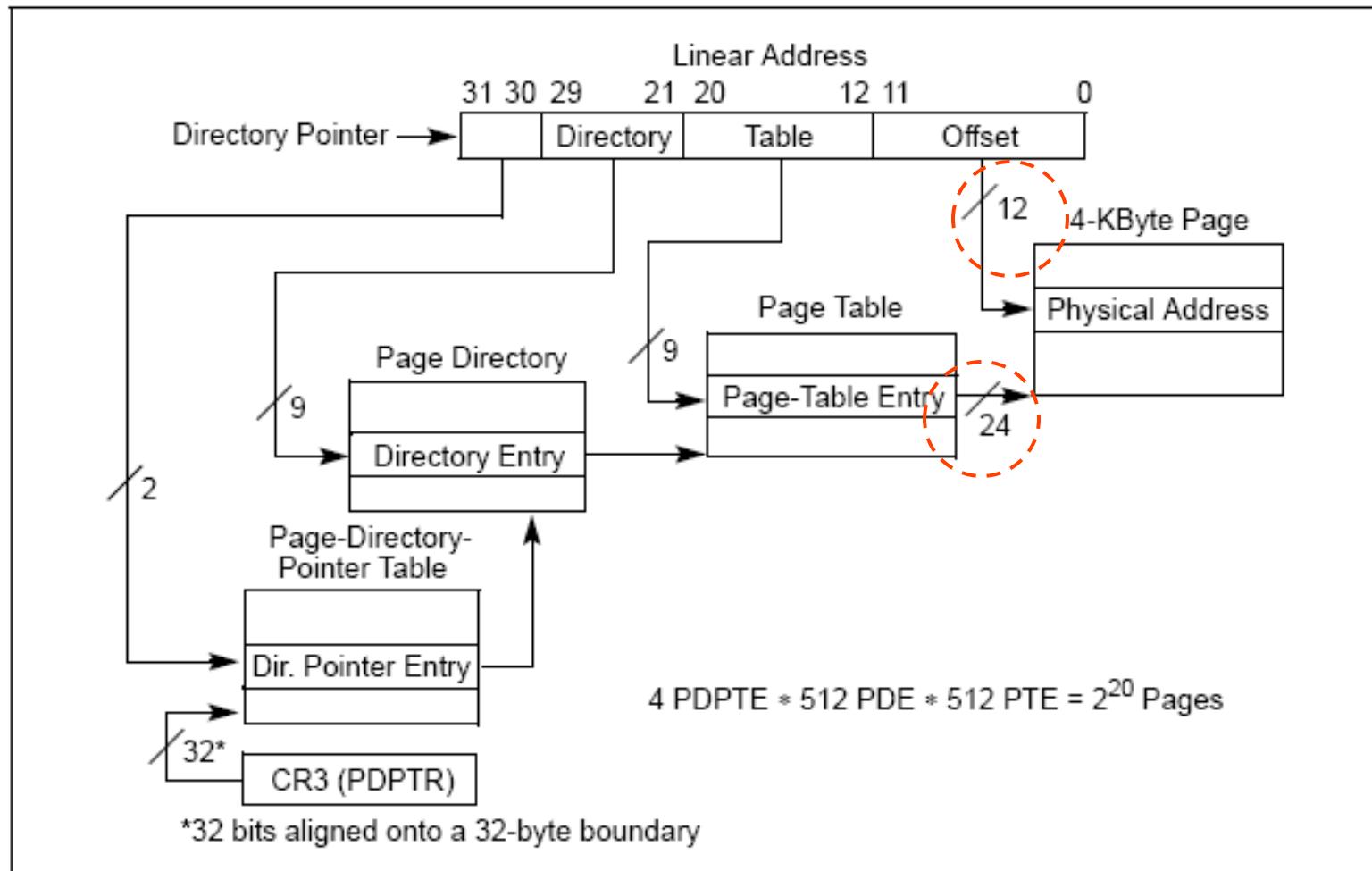


Figure 3-18. Linear Address Translation With PAE Enabled (4-KByte Pages)

32-bit Mode w/ PAE: 2MB pages

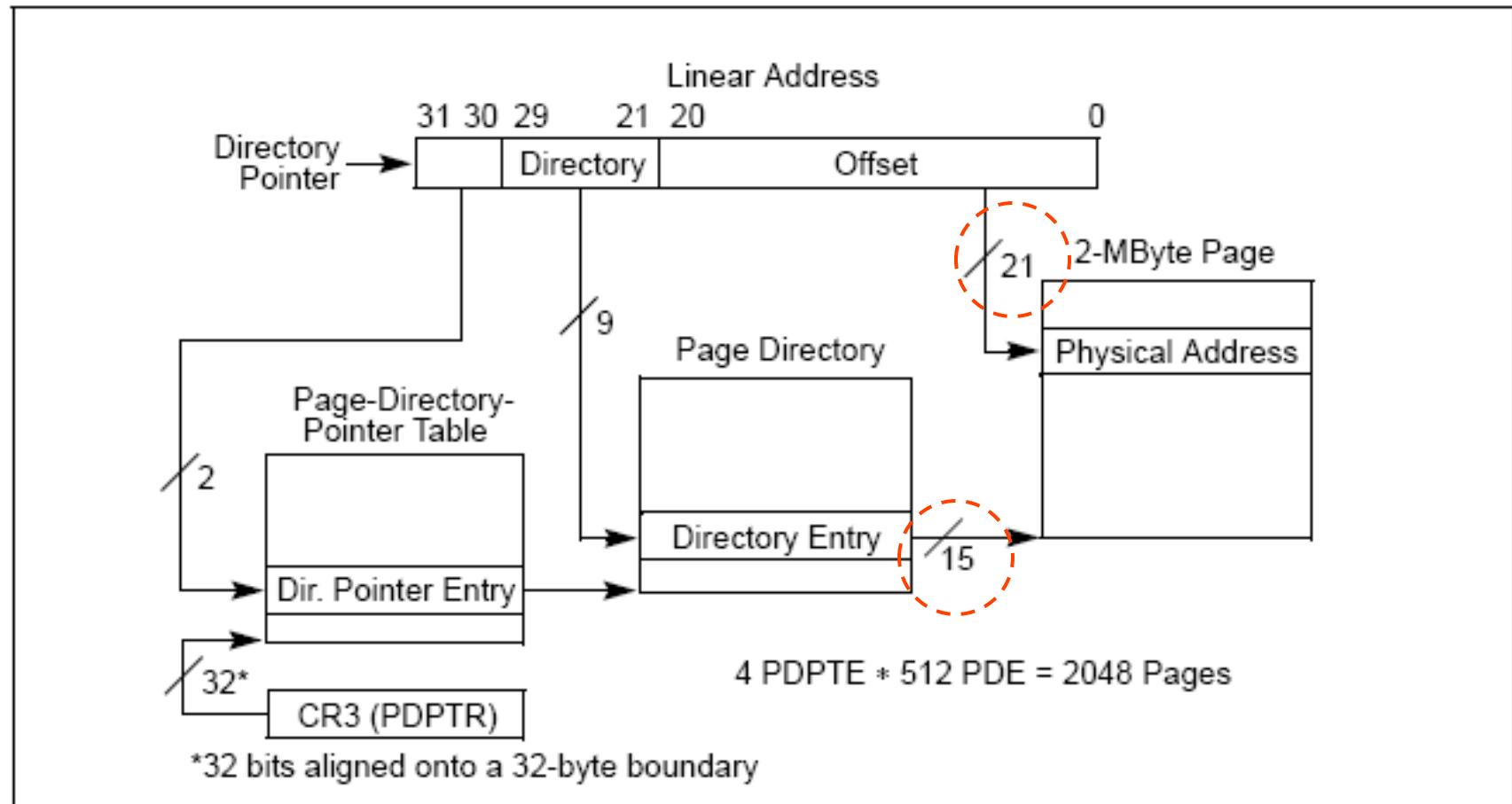


Figure 3-19. Linear Address Translation With PAE Enabled (2-MByte Pages)

64-bit Mode: 4-level Page Tables: 4K Pages

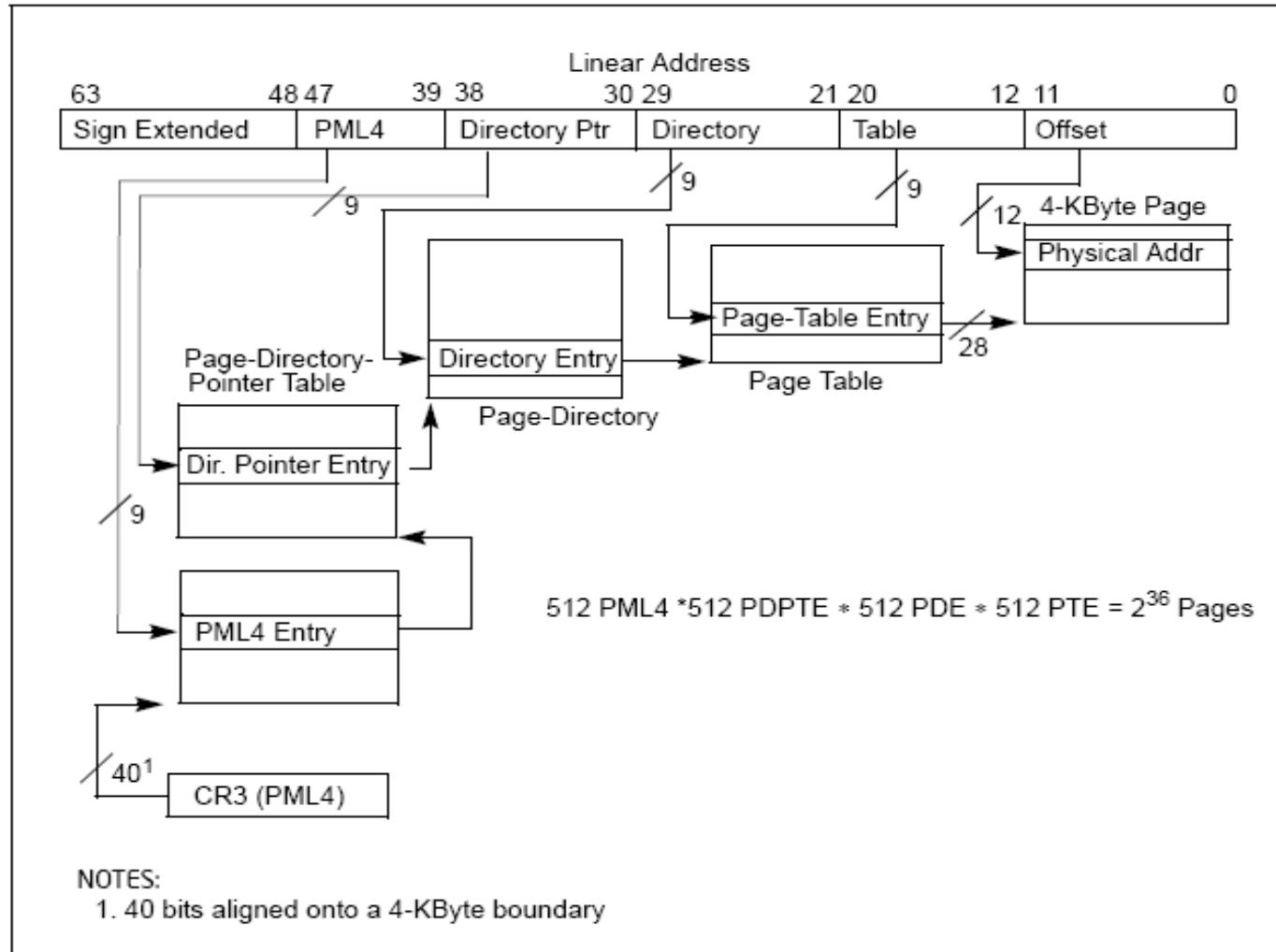


Figure 3-24. IA-32e Mode Paging Structures (4-KByte Pages)

64-bit Mode: 4-level Page Tables: 2MB Pages

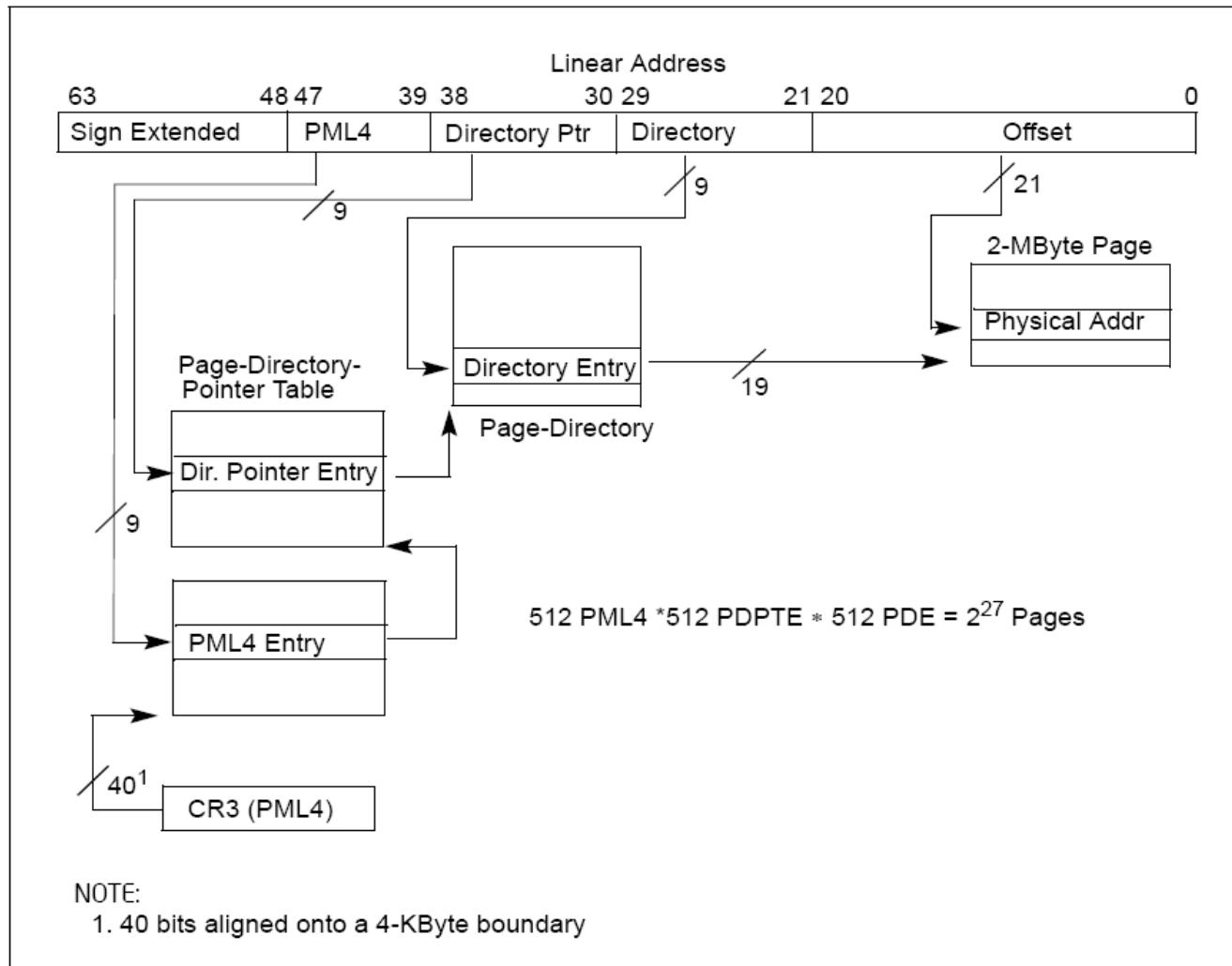


Figure 3-25. IA-32e Mode Paging Structures (2-MByte pages)

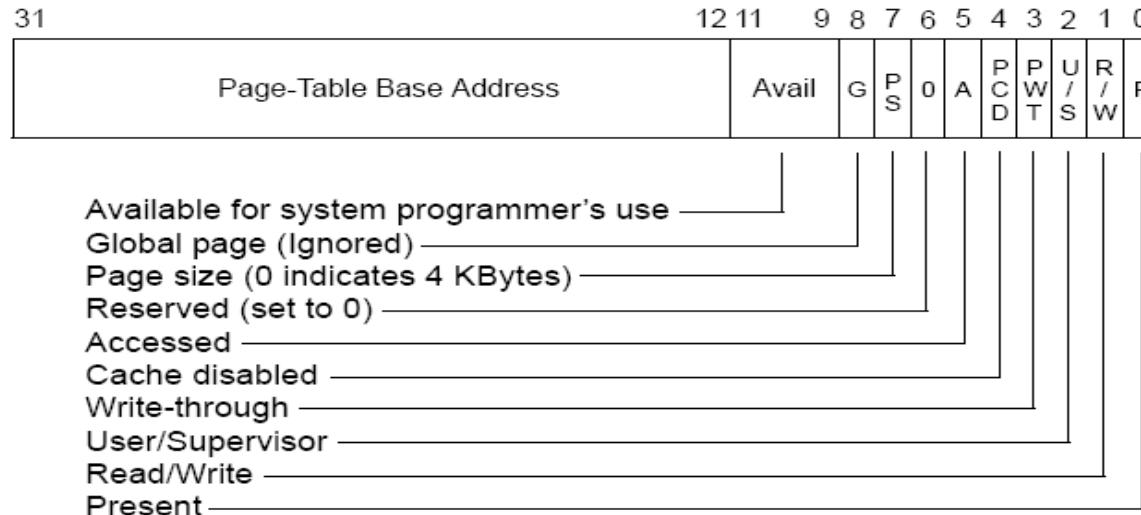
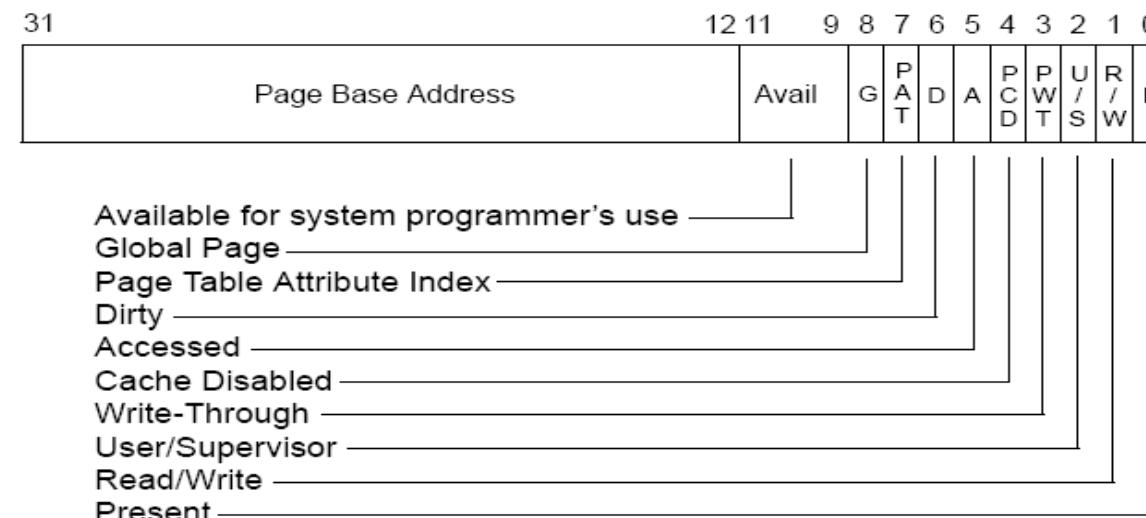
Page-Directory Entry (4-KByte Page Table)**Page-Table Entry (4-KByte Page)**

Figure 3-14. Format of Page-Directory and Page-Table Entries for 4-KByte Pages and 32-Bit Physical Addresses

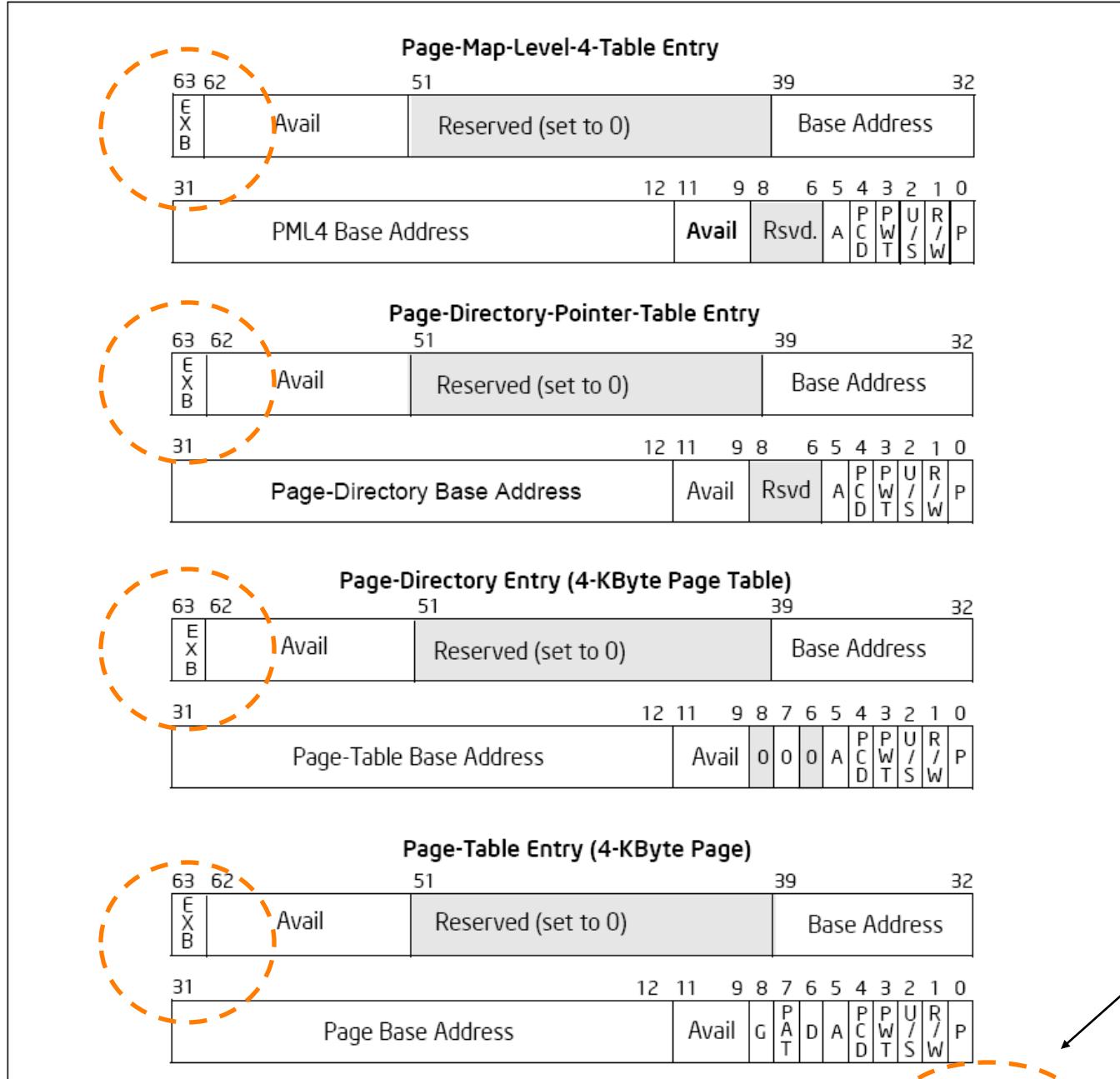


Figure 3-26. Format of Paging Structure Entries for 4-KByte Pages in IA-32e Mode

Data Structures

- Basic OS (and application) data types are often C data types
 - `char`, `int`, `unsigned int`, `short`, `long`, `pointers`, etc.
- Plus structures built on these types
- Abstract data types provide models for data and the operations that can be performed
- Accurately modeling OS and application structures is essential for memory analysis tools

Strings

- Simple null-terminated arrays of characters (ASCII or Unicode)
- Microsoft's `_UNICODE_STRING` type is a special case where length is tracked separately
- Extracting strings from memory dumps can often provide clues and useful evidence, but associating context is difficult (without virtual address translation)

Linked Lists

- Singly and doubly linked lists are commonly used for tracking collections of objects
- Track processes, DLLs, loaded modules, etc.
- Memory forensics tools often find pointers to “list heads” (beginning of the lists)
- Rootkits commonly “hide” elements by manipulating the lists, removing elements, adjusting pointers

Additional Structures

- Hash tables store data in <key, element> pairs and provide efficient searching and insertion operations
 - e.g., Linux PID Hash Table
- Trees are composed of a set of nodes that store data and links that connect the nodes
 - Hierarchical, binary trees have a root node with children that connect to at most 2 other nodes
 - The actual nodes can be any data type
 - e.g., process memory allocations on Windows are stored in a tree data structure

Memory Dump Analysis

- Once you have a memory dump, then what?
- Analyze dump to extract information about processes, threads, open files, sockets, etc.
- Most interesting things in the kernel are “objects” (e.g., structures)
- Objects likely have many pointers to other objects

Discovering Objects

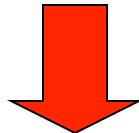
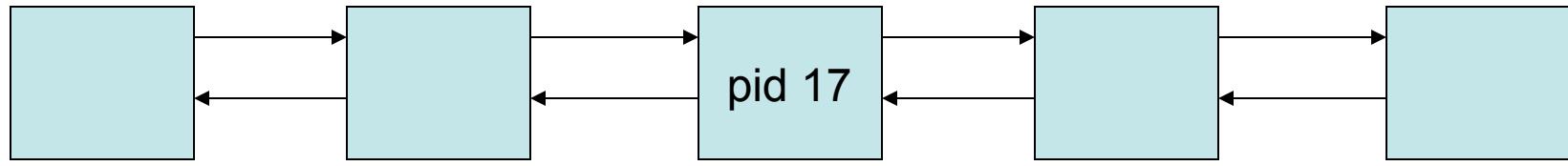
- Method # 1:
 - **List Walking:** Find start, then analyze lists/tables of kernel structures
- Method # 2:
 - **Carving:** Discover locations of individual, interesting objects
- Hybrid:
 - Combine these approaches

Finding Objects: List-Walking

- Basic idea beyond list-based memory analysis
 - From symbol table for kernel or other source, determine location of interesting tables/lists and enumerate
 - e.g., for Linux, use /boot/System.map file created when kernel is compiled
 - Can locate system call table, first process, etc.
- One challenge for list walking: **DKOM**
- Another challenge is that **deallocated structures are (typically) not in the lists**
- e.g., won't find representations of terminated processes

DKOM: FU “Rootkit” under Windows

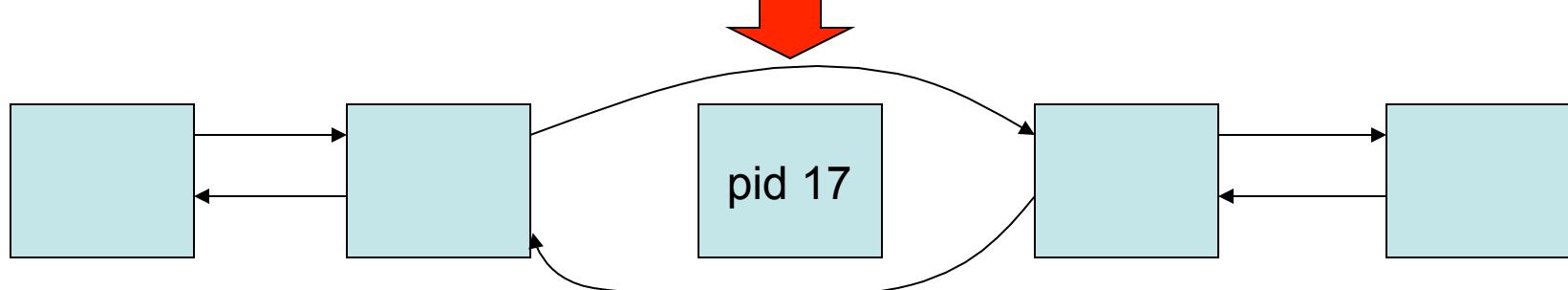
Doubly-linked process list in Windows kernel



Example of **DKOM**:
Direct Kernel Object Manipulation

C:\> fu -ph 17

Processes continue
to run because Windows
scheduler handles
threads, not processes



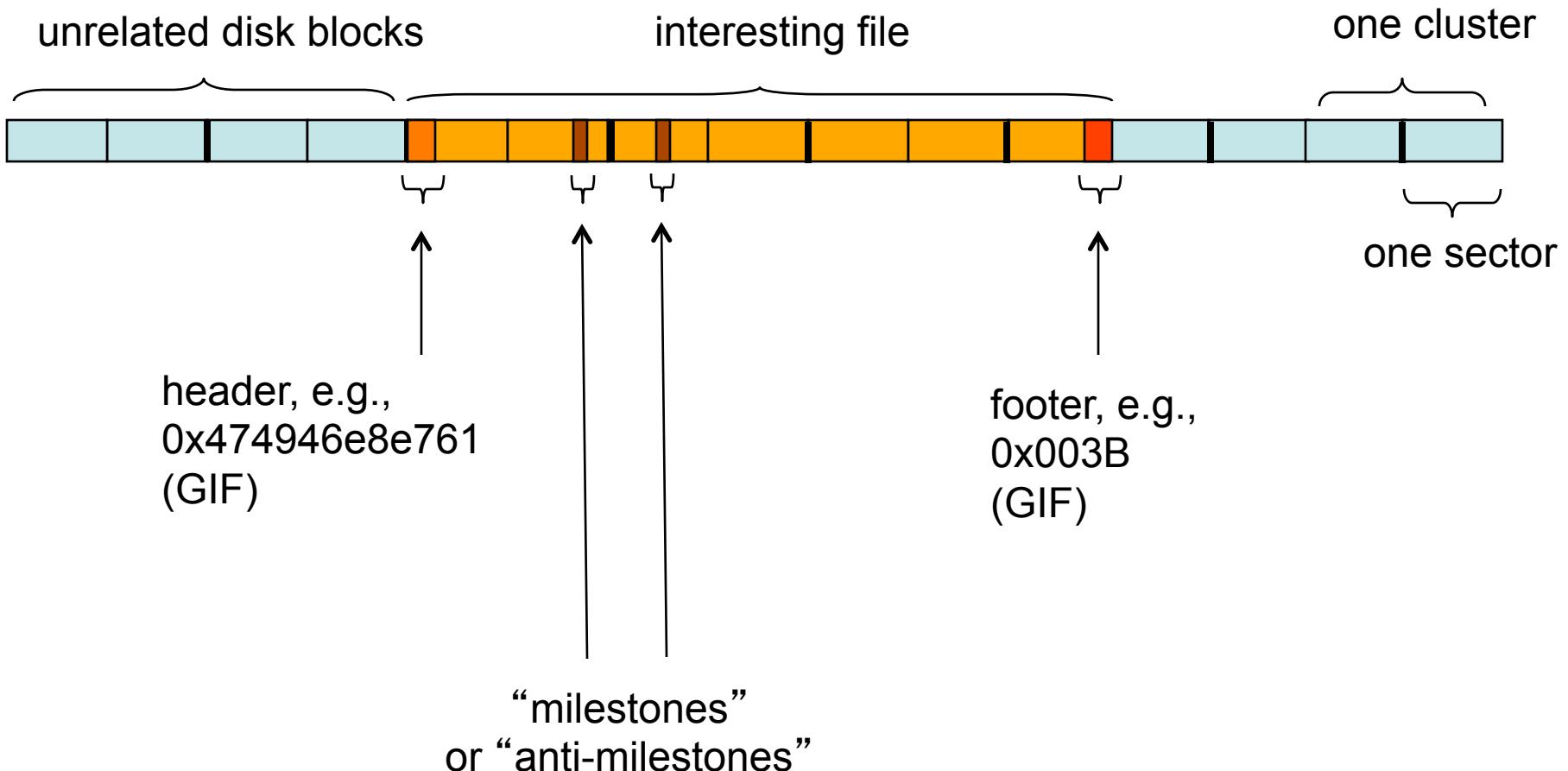
Aside: Lists and DKOM

- Not hopeless
- For process hiding case, can look deeper than the process list
- Correlate with other sources of process information (threads, handles, etc.)
- Can also supplement with carving approaches

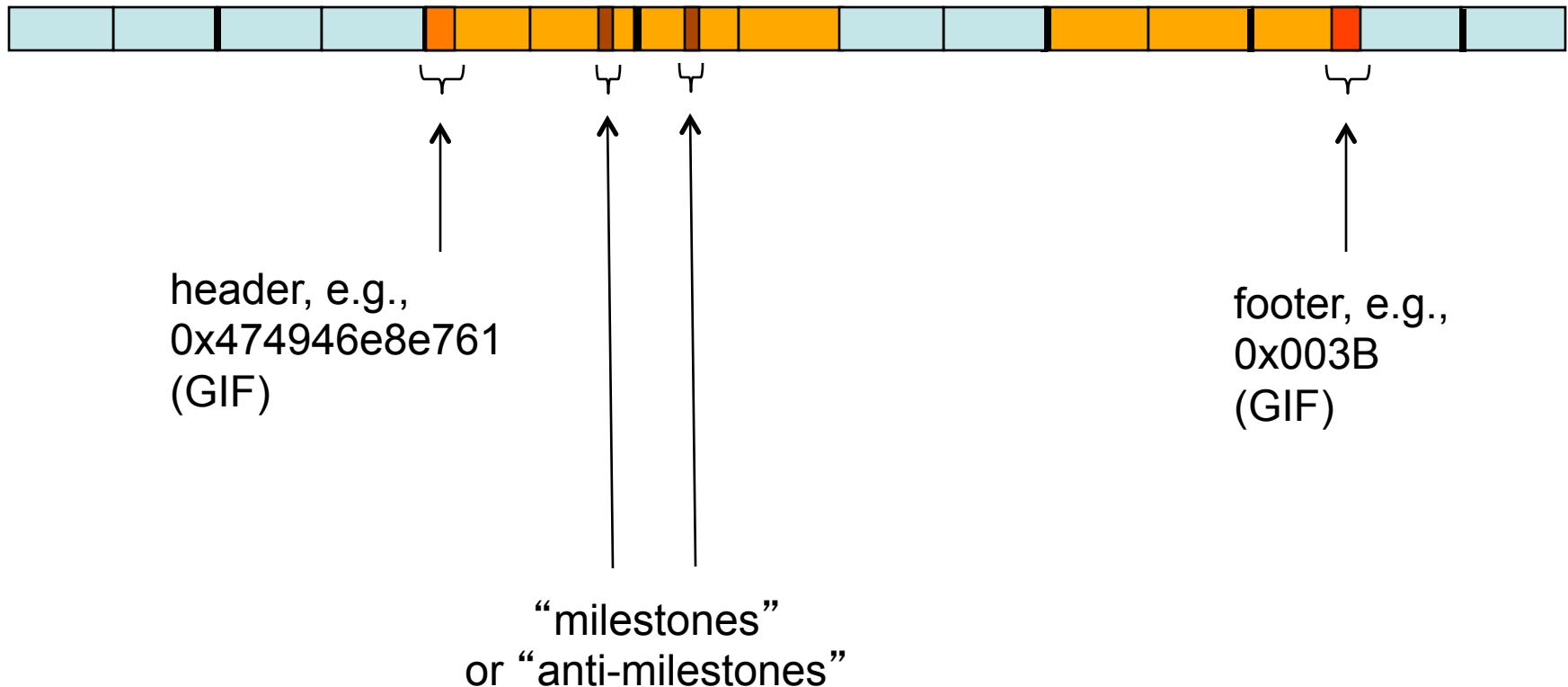
Data Carving

- In storage forensics, carving identifies begin / end of deleted files
- In memory analysis, use patterns or templates to discover structures in dump
- Not fooled by DKOM
- Can potentially identify deallocated structures
- e.g., can potentially find representations of terminated processes

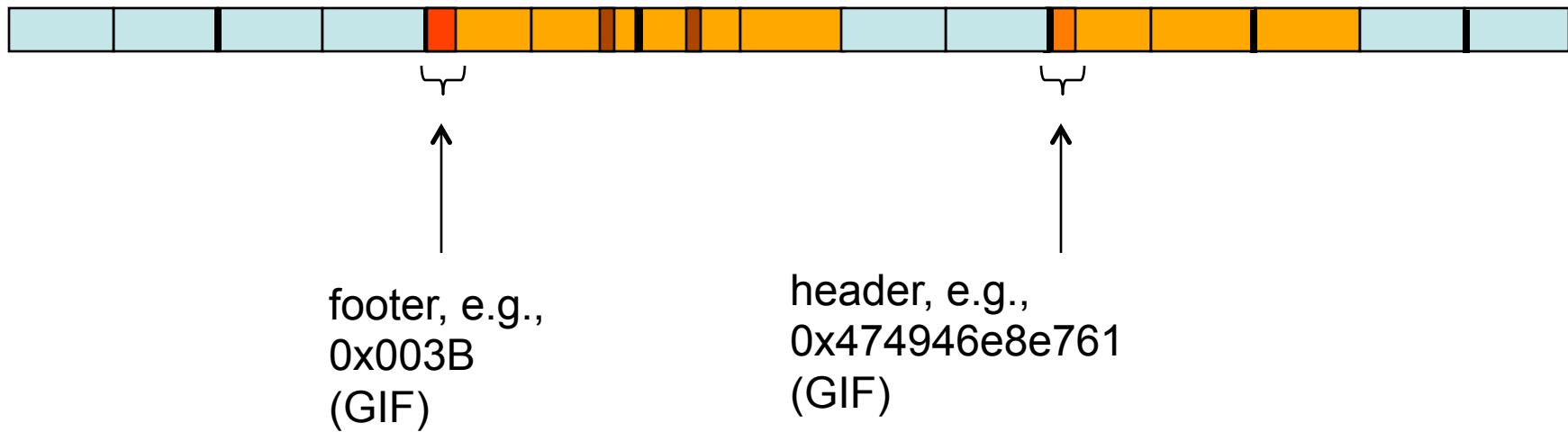
File Carving: Basic Idea



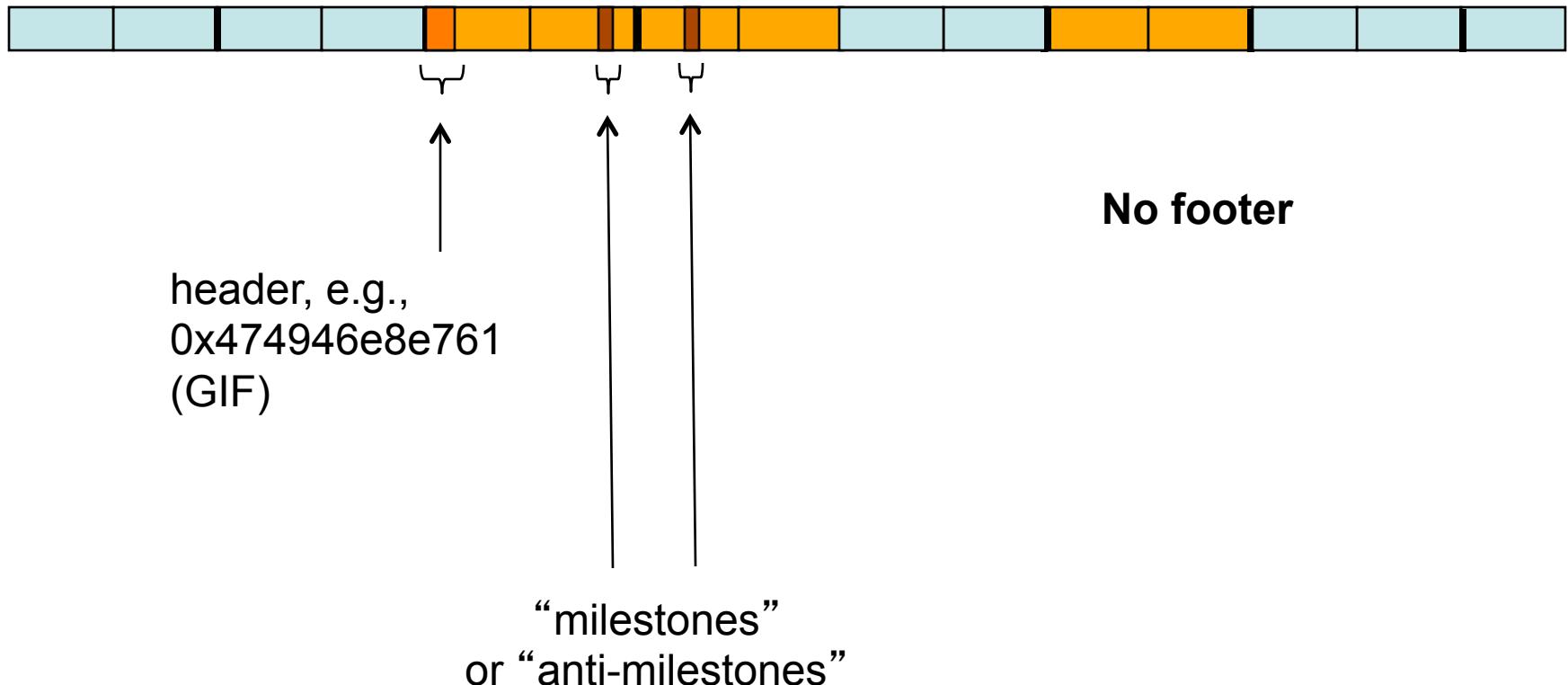
File Carving: Fragmentation



File Carving: Fragmentation



File Carving: Damaged Files



Carving Example

- Idea: Search Linux memory dump to find USB keyboard DMA buffer
- Often targeted by keystroke loggers
- Used in recent POCs of GPU malware
- Evaluate known characteristics of associated structure to discover location of buffer

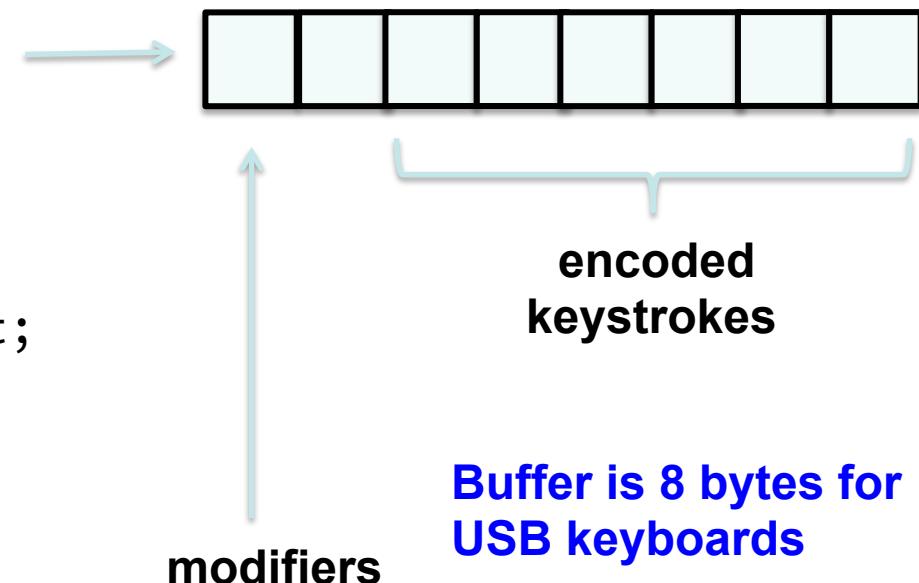
```
struct urb {  
    struct list_head urb_list;  
    struct list_head anchor_list;  
    struct usb_anchor * anchor;  
struct usb_device * dev;  
    struct usb_host_endpoint * ep;  
    unsigned int pipe;  
    unsigned int stream_id;  
    int status;  
    unsigned int transfer_flags;  
    void * transfer_buffer;  
dma_addr_t transfer_dma;  
    struct scatterlist * sg;  
    int num_mapped_sgs;  
    int num_sgs;  
u32 transfer_buffer_length;  
    u32 actual_length;  
    unsigned char * setup_packet;  
    dma_addr_t setup_dma;  
...  
};
```

Linux USB Request Buffer

```

struct urb {
    struct list_head urb_list;
    struct list_head anchor_list;
    struct usb_anchor *anchor;
struct usb_device *dev;
    struct usb_host_endpoint *ep;
    unsigned int pipe;
    unsigned int stream_id;
    int status;
    unsigned int transfer_flags;
    void *transfer_buffer;
dma_addr_t transfer_dma;
    struct scatterlist *sg;
    int num_mapped_sgs;
    int num_sgs;
u32 transfer_buffer_length;
    u32 actual_length;
    unsigned char *setup_packet;
    dma_addr_t setup_dma;
    ...
};
```

Linux USB Request Buffer is a **struct urb**



Search RAM page by page

```

for (res = iomem_resource.child; res ; res = res->sibling) {
    if (!strncmp(res->name, RAMSTR, sizeof(RAMSTR))) {
        for (i = (void *)res->start; i < (void *)res->end; i += PAGE_SIZE) {
            is = min(PAGE_SIZE, ((unsigned long)res->end - (unsigned long)i + 1));
            if (is == PAGE_SIZE) {
                p = pfn_to_page(((unsigned long)i) >> PAGE_SHIFT);
                v=kmap(p);
                if (v) {
                    for (j = v; j < v + PAGE_SIZE - sizeof(struct urb); j++) {
                        urbp = (struct urb *)j;
                        if (((unsigned long)urbp->dev) % 0x400 == 0 &&
                            __virt_addr_valid((unsigned long)urbp->dev) &&
                            ((unsigned long)urbp->transfer_dma) % 0x20 == 0 &&
                            urbp->transfer_buffer_length == 8 &&
                            __virt_addr_valid((unsigned long)urbp->dev->product)&&
                            (strnstr(urbp->dev->product, "Keyboard", 32)) {
                            usb_dma_addr = urbp->transfer_dma;
                            goto almostdone;
                        }
                    }
                }
            }
        }
    }
}

```



And remember
***physical* address of**
keyboard DMA buffer

for struct urb
pattern

```
if (((unsigned long)urbp->dev) % 0x400 == 0 &&
    __virt_addr_valid((unsigned long)urbp->dev) &&
    ((unsigned long)urbp->transfer_dma) % 0x20 == 0 &&
    urbp->transfer_buffer_length == 8 &&
    __virt_addr_valid((unsigned long)urbp->dev->product)&&
    (strnstr(urbp->dev->product, "Keyboard", 32)) {
    usb_dma_addr = urbp->transfer_dma;
    goto almostdone;
}

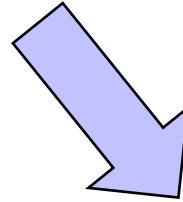
if (urbp->dev is aligned on 0x400 boundary &&
    urbp->transfer_dma is aligned on 0x20 boundary &&
    urbp->transfer_buffer_length is 8 bytes in length &&
    urbp->dev->product contains "Keyboard") {
    // found the DMA buffer for keyboard
}
```

Vtypes in Volatility

- Used to accurately represent C structures in Python
- One of the key reasons that writing plugins in Volatility is so straightforward
- Automatically generated from debugging symbols when possible
 - Microsoft PDB files
 - Linux/Mac DWARF information
- If debugging symbols aren't available,...
- Then they must be created manually via source code analysis or reverse engineering ☹
- Objects and classes subsystem allows “attaching” methods or behaviors to instances of a structure

```
struct process {  
    int pid;  
    int parent_pid;  
    char name[10];  
    char *command_line;  
    void *ptv;  
};
```

C type

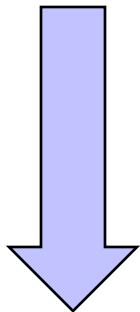


Volatility Vtype

```
'process' : [ 26, {  
    'pid' : [ 0, ['int']],  
    'parent_pid' : [ 4, ['int']],  
    'name' : [ 8, ['array', 10, ['char']]],  
    'command_line' : [ 18, ['pointer', ['char']]],  
    'ptv' : [ 22, ['pointer', ['void']]],  
}]
```

Extraction of DWARF Debugging Info

module.c



**compile "dummy" kernel module that
references all the types we want, with
debug info enabled**

module.ko

DWARF Debugging Data in ELF File

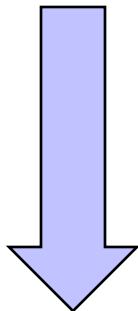
```
user@ubuntu:~/volatility/tools/linux$ objdump -h module.ko

module.ko:      file format elf64-x86-64

Sections:
Idx Name      Size    VMA          LMA          File off  Align
 0 .note.gnu.build-id 00000024  0000000000000000  0000000000000000  00000040  2**2
                CONTENTS, ALLOC, LOAD, READONLY, DATA
 1 .text       00000000  0000000000000000  0000000000000000  00000064  2**0
                CONTENTS, ALLOC, LOAD, READONLY, CODE
 2 .modinfo    00000063  0000000000000000  0000000000000000  00000064  2**0
                CONTENTS, ALLOC, LOAD, READONLY, DATA
 3 __versions  00000040  0000000000000000  0000000000000000  000000e0  2**5
                CONTENTS, ALLOC, LOAD, READONLY, DATA
 4 .data       00000000  0000000000000000  0000000000000000  00000120  2**0
                CONTENTS, ALLOC, LOAD, DATA
 5 .gnu.linkonce.this_module 00000260  0000000000000000  0000000000000000  00000120  2**5
                CONTENTS, ALLOC, LOAD, DATA, LINK_ONCE_DISCARD
 6 .bss        00002318  0000000000000000  0000000000000000  00000380  2**6
                ALLOC
 7 .debug_info 0001c567  0000000000000000  0000000000000000  00000380  2**0
                CONTENTS, RELOC, READONLY, DEBUGGING
 8 .debug_abbrev 00000642  0000000000000000  0000000000000000  0001c8e7  2**0
                CONTENTS, READONLY, DEBUGGING
 9 .debug_aranges 00000040  0000000000000000  0000000000000000  0001cf29  2**0
                CONTENTS, RELOC, READONLY, DEBUGGING
10 .debug_line  000013a3  0000000000000000  0000000000000000  0001cf69  2**0
                CONTENTS, READONLY, DEBUGGING
11 .debug_str   000104f1  0000000000000000  0000000000000000  0001e30c  2**0
                CONTENTS, READONLY, DEBUGGING
12 .comment    0000004a  0000000000000000  0000000000000000  0002e7fd  2**0
                CONTENTS, READONLY
13 .note.GNU-stack 00000000  0000000000000000  0000000000000000  0002e847  2**0
                CONTENTS, READONLY
user@ubuntu:~/volatility/tools/linux$
```

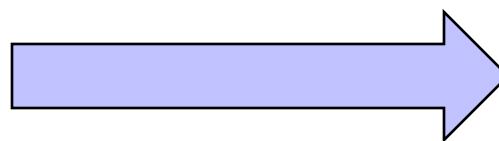
Extraction of DWARF Debugging Info

module.c



compile "dummy" kernel module that references all the types we want, with debug info enabled

module.ko



module.dwarf

Run dwarfdump on compiled kernel module to generate parseable DWARF data

DWARF (!)

...

...

```
<1><0x10eb><DW_TAG_structure_type> DW_AT_name<"task_struct"> DW_AT_byte_size<0x000006f8>
DW_AT_decl_file<0x0000000f /usr/src/linux-headers-3.2.0-4-common/include/linux/sched.h>
DW_AT_decl_line<0x000004c9> DW_AT_sibling<<0x000019e7>>
<2><0x10f9><DW_TAG_member> DW_AT_name<"state"> DW_AT_decl_file<0x0000000f /usr/src/linux-
headers-3.2.0-4-common/include/linux/sched.h> DW_AT_decl_line<0x000004ca>
DW_AT_type<<0x0000bedf>> DW_AT_data_member_location<DW_OP_plus_uconst 0>
<2><0x1108><DW_TAG_member> DW_AT_name<"stack"> DW_AT_decl_file<0x0000000f /usr/src/linux-
headers-3.2.0-4-common/include/linux/sched.h> DW_AT_decl_line<0x000004cb>
DW_AT_type<<0x000004a2>> DW_AT_data_member_location<DW_OP_plus_uconst 8>
<2><0x1117><DW_TAG_member> DW_AT_name<"usage"> DW_AT_decl_file<0x0000000f /usr/src/linux-
headers-3.2.0-4-common/include/linux/sched.h> DW_AT_decl_line<0x000004cc>
DW_AT_type<<0x000003a5>> DW_AT_data_member_location<DW_OP_plus_uconst 16>
<2><0x1126><DW_TAG_member> DW_AT_name<"flags"> DW_AT_decl_file<0x0000000f /usr/src/linux-
headers-3.2.0-4-common/include/linux/sched.h> DW_AT_decl_line<0x000004cd>
DW_AT_type<<0x00000092>> DW_AT_data_member_location<DW_OP_plus_uconst 20>
<2><0x1135><DW_TAG_member> DW_AT_name<"ptrace"> DW_AT_decl_file<0x0000000f /usr/src/linux-
headers-3.2.0-4-common/include/linux/sched.h> DW_AT_decl_line<0x000004ce>
DW_AT_type<<0x00000092>> DW_AT_data_member_location<DW_OP_plus_uconst 24>
<2><0x1144><DW_TAG_member> DW_AT_name<"wake_entry"> DW_AT_decl_file<0x0000000f /usr/src/
linux-headers-3.2.0-4-common/include/linux/sched.h> DW_AT_decl_line<0x000004d1>
DW_AT_type<<0x0000ad52>> DW_AT_data_member_location<DW_OP_plus_uconst 32>
<2><0x1153><DW_TAG_member> DW_AT_name<"on_cpu"> DW_AT_decl_file<0x0000000f /usr/src/linux-
headers-3.2.0-4-common/include/linux/sched.h> DW_AT_decl_line<0x000004d2>
```

...

...

<1><0x10eb><**DW_TAG_structure_type**> **DW_AT_name**<"task_struct">
DW_AT_byte_size<0x000006f8> DW_AT_decl_file<0x0000000f /usr/src/linux-headers-3.2.0-4-common/include/linux/sched.h>

...

<2><0x1269><**DW_TAG_member**> DW_AT_name<"mm"> DW_AT_decl_file<0x0000000f /usr/src/linux-headers-3.2.0-4-common/include/linux/sched.h> DW_AT_decl_line<0x00000509>
DW_AT_type<<0x00001d27>>

DW_AT_data_member_location<**DW_OP_plus_uconst** 424>

...


 <1><0x1d27><**DW_TAG_pointer_type**> DW_AT_byte_size<0x00000008>
DW_AT_type<<0x00001d2d>>
<1><0x1d2d><**DW_TAG_structure_type**>
DW_AT_name<"mm_struct"> **DW_AT_byte_size**<0x00000398>
 DW_AT_decl_file<0x0000000c /usr/src/linux-headers-3.2.0-4-common/include/linux/mm_types.h> DW_AT_decl_line<0x00000121> DW_AT_sibling<<0x000020ad>>
<2><0x1d3b><**DW_TAG_member**> **DW_AT_name**<"mmap">
 DW_AT_decl_file<0x0000000c /usr/src/linux-headers-3.2.0-4-common/include/linux/mm_types.h> DW_AT_decl_line<0x00000122> DW_AT_type<<0x00005fb>>
DW_AT_data_member_location<**DW_OP_plus_uconst** 0>
<2><0x1dc2><**DW_TAG_member**> **DW_AT_name**<"pgd">
 DW_AT_decl_file<0x0000000c /usr/src/linux-headers-3.2.0-4-common/include/linux/mm_types.h> DW_AT_decl_line<0x0000012f> DW_AT_type<<0x0000212e>>
DW_AT_data_member_location<**DW_OP_plus_uconst** 72>

Volatility Overlays

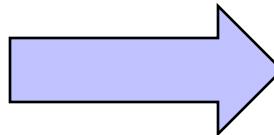
- Overlays allow automated **patching** of Vtypes
- Used to fix problematic situations where insufficient debugging information is available
- One serious offender: null pointers as structure members

```
struct process {  
    int pid;  
    int parent_pid;  
    char name[10];  
    char *command_line;  
    void *ptv;  
};
```

Overlays (2)

Analysis of source or reverse engineering may reveal that the dereferenced member (always) has a predictable type

```
struct process {  
    int pid;  
    int parent_pid;  
    char name[10];  
    char *command_line;  
    void *ptv;  
};
```

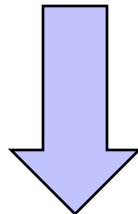


```
struct process {  
    int pid;  
    int parent_pid;  
    char name[10];  
    char *command_line;  
    struct process *ptv;  
};
```

Overlays (3)



```
'process' : [ 26, {
    'pid' : [ 0, ['int']],
    'parent_pid' : [ 4, ['int']],
    'name' : [ 8, ['array', 10, ['char']]],
    'command_line' : [ 18, ['pointer', ['char']]],
    'ptv' : [ 22, ['pointer', ['void']]],
}]
```



OVERLAY

```
'process' : [ None, {
    'ptv' : [ None, ['pointer', ['process']]],
}]
```

```
'process' : [ 26, {
    'pid' : [ 0, ['int']],
    'parent_pid' : [ 4, ['int']],
    'name' : [ 8, ['array', 10, ['char']]],
    'command_line' : [ 18, ['pointer', ['char']]],
    'ptv' : [ 22, ['pointer', ['process']]],
}]
```

Profiles

- A Volatility **profile** is a collection of **vtypes**, **overlays**, and **object classes** for a specific OS version and hardware architecture
 - System call information
 - Constant values, global variables
 - Native types
 - System.map (Linux/Mac only)
- ...all bundled up in a nice ZIP file
- Profiles for Windows are distributed with Volatility
- Generally build them for each Linux/Mac instance
- Typically follow a naming convention (for example, Win7SP1x86 or Win2003SP0x64)
- **\$ python vol.py -info** to see which are available
- Build others as needed

Creating a New Linux Profile

The following builds a kernel module solely to get debugging info:

```
$ cd ~/volatility/tools/linux  
$ make  
$ head module.dwarf
```

```
.debug_info  
<0><0+11><DW_TAG_compile_unit> ....  
<1><45><DW_TAG_typeDefinition> DW_AT_name<_s8> ...
```

Now gather the debug info and kernel symbols for the currently executing kernel into a single ZIP file. The name for the ZIP file is descriptive—use the name of the distribution for a Linux profile:

```
$ cd  
$ sudo zip ~/volatility/volatility/plugins/overlays/linux/  
Ubuntu1404.zip ~volatility/tools/linux/module.dwarf /boot/  
System.map-$(uname -r)
```

Creating a New Mac OS X Profile

Grab the kernel debug kits from Apple for the appropriate kernels via <https://developer.apple.com/downloads/>. Put them all in the same directory, e.g., ~/Desktop. While you're there, install the latest Xcode via the same link. Then type the following to install the OS X command line developer tools:

```
$ xcode-select -install
```

Then:

```
$ cd ~/volatility/tools/mac
$ mkdir tmp
$ python mac_create_all_profiles.py ~/Desktop tmp \
  ~/volatility ~/volatility/volatility/plugins/overlays/mac
```

This creates a profile for each of the debug kits discovered in the specified directory (in this case, ~/Desktop), in one step.

Checking New Profile

To check proper installation of your new profile, do:

```
$ cd ~/volatility/  
$ python vol.py --info
```

In the Profiles area, you should see the official name of your new profile:

```
...  
Profiles  
-----
```

LinuxXubuntu1404x64 - A Profile for Linux Xubuntu1404 x64

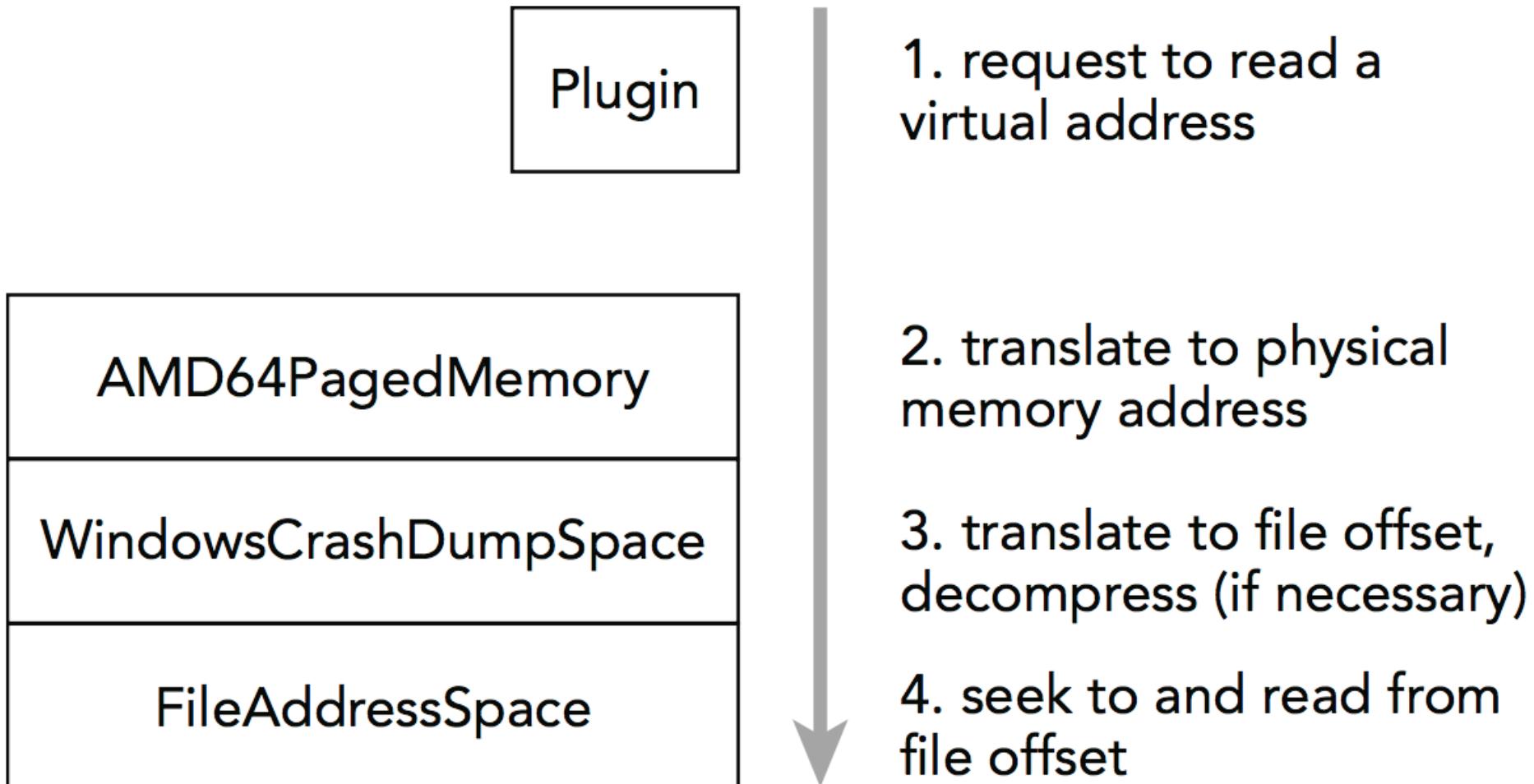
VistaSP0x64	- A Profile for Windows Vista SP0 x64
VistaSP0x86	- A Profile for Windows Vista SP0 x86
VistaSP1x64	- A Profile for Windows Vista SP1 x64

```
...
```

Address Spaces (AS)

- A Volatility address space:
 - Provides a flexible and consistent interface to data in RAM
 - Handles virtual to physical address translation as necessary
 - Transparently accounts for differences in memory dump file formats (proprietary headers, compression, etc.)
- Paged address spaces include IA32 and AMD64 (32- and 64-bit Intel)
- Volatility automatically detects the proper AS to use for a memory sample
- AS's are stacked on top of one another

Address Spaces Stack



Plugins

- Entire Volatility framework is expandable
 - Add address spaces to support new hardware architectures or memory dump formats
 - Add profiles to support new operating systems or new operating system versions
 - Add new plugins to analyze new types of data from the kernel, processes, etc.
- e.g., when LiME was developed, introduced `volatility/plugins/addrspace/lime.py`
- Understands LiME memory format headers, etc.

Plugins (2)

- Plugins have `calculate()` and `render_text()` methods
- Data gathering is performed in `calculate()` and it yields results to the rendering function
 - Can also add renderers for output to CSV, XML, database, etc.
- **Flexibility** is the most important difference between Volatility and previous memory analysis efforts

Another Carving Example

- Finding the right Windows profile
- Volatility can often (help) identify the correct Windows profile for a memory dump
- Windows XP → along with various service packs
- Uses kernel debugger data block (KDBG) structure to discover Windows version
- We'll look at kdbgscan to understand this a bit better

Identifying Windows Profiles

- Use **imageinfo** or **kdbgscan** plugins
- Some profiles look similar (for example, XP SP2 and XP SP3), use the Service Pack details in kdbgscan output to choose the right one
- If kdbgscan finds multiple KDBG structures, look at the number of processes and modules to determine the best choice
- Keep in mind the values are nonessential to the stability of the OS, so they can be manipulated by attackers

```
garfish:volatility golden$ python vol.py --profile=Win7SP1x64 -f ~/Documents/Virtual\ Machine  
s.localized/Windows\ 7\ x64/Windows\ 7\ x64-1e433ca5.vmem kdbgscan  
Volatility Foundation Volatility Framework 2.4  
*****  
Instantiating KDBG using: Kernel AS Win7SP1x64 (6.1.7601 64bit)  
Offset (V) : 0xf80002ffe0b0  
Offset (P) : 0x2ffe0b0  
KDBG owner tag check : True  
Profile suggestion (KDBGHeader): Win7SP1x64  
Version64 : 0xf80002ffe068 (Major: 15, Minor: 7601)  
Service Pack (CmNtCSDVersion) : 1  
Build string (NtBuildLab) : 7601.18798.amd64fre.win7sp1_gdr.  
PsActiveProcessHead : 0xfffff80003034590 (88 processes)  
PsLoadedModuleList : 0xfffff80003052890 (180 modules)  
KernelBase : 0xfffff80002e0d000 (Matches MZ: True)  
Major (OptionalHeader) : 6  
Minor (OptionalHeader) : 1  
KPCR : 0xfffff80002fffd00 (CPU 0)  
*****  
Instantiating KDBG using: Kernel AS Win7SP1x64 (6.1.7601 64bit)  
Offset (V) : 0xf80002ffe0b0  
Offset (P) : 0x2ffe0b0  
KDBG owner tag check : True  
Profile suggestion (KDBGHeader): Win2008R2SP1x64  
Version64 : 0xf80002ffe068 (Major: 15, Minor: 7601)  
Service Pack (CmNtCSDVersion) : 1  
Build string (NtBuildLab) : 7601.18798.amd64fre.win7sp1_gdr.  
PsActiveProcessHead : 0xfffff80003034590 (88 processes)  
PsLoadedModuleList : 0xfffff80003052890 (180 modules)  
KernelBase : 0xfffff80002e0d000 (Matches MZ: True)  
Major (OptionalHeader) : 6  
Minor (OptionalHeader) : 1  
KPCR : 0xfffff80002fffd00 (CPU 0)
```



```
*****
Instantiating KDBG using: Kernel AS Win7SP1x64 (6.1.7601 64bit)
Offset (V) : 0xf80002ffe0b0
Offset (P) : 0x2ffe0b0
KDBG owner tag check : True
Profile suggestion (KDBGHeader): Win2008R2SP0x64
Version64 : 0xf80002ffe068 (Major: 15, Minor: 7601)
Service Pack (CmNtCSDVersion) : 1
Build string (NtBuildLab) : 7601.18798.amd64fre.win7sp1_gdr.
PsActiveProcessHead : 0xfffff80003034590 (88 processes)
PsLoadedModuleList : 0xfffff80003052890 (180 modules)
KernelBase : 0xfffff80002e0d000 (Matches MZ: True)
Major (OptionalHeader) : 6
Minor (OptionalHeader) : 1
KPCR : 0xfffff80002ffffd00 (CPU 0)
```

```
*****
Instantiating KDBG using: Kernel AS Win7SP1x64 (6.1.7601 64bit)
Offset (V) : 0xf80002ffe0b0
Offset (P) : 0x2ffe0b0
KDBG owner tag check : True
Profile suggestion (KDBGHeader): Win7SP0x64
Version64 : 0xf80002ffe068 (Major: 15, Minor: 7601)
Service Pack (CmNtCSDVersion) : 1
Build string (NtBuildLab) : 7601.18798.amd64fre.win7sp1_gdr.
PsActiveProcessHead : 0xfffff80003034590 (88 processes)
PsLoadedModuleList : 0xfffff80003052890 (180 modules)
KernelBase : 0xfffff80002e0d000 (Matches MZ: True)
Major (OptionalHeader) : 6
Minor (OptionalHeader) : 1
KPCR : 0xfffff80002ffffd00 (CPU 0)
```

imageinfo Plugin

```
bigjoe:volatility golden$ python vol.py --profile=WinXPSP3x86 -f /Volumes/SLOWDATA/IMAGES-THUMB-BAC
KUP/MEMIMAGES/6623/evilprofessor.vmem imageinfo
Volatility Foundation Volatility Framework 2.4
INFO    : volatility.plugins.imageinfo: Determining profile based on KDBG search...
Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86
                  AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                  AS Layer2 : FileAddressSpace (/Volumes/SLOWDATA/IMAGES-THUMB-BACKUP/MEMIMAGES/
6623/evilprofessor.vmem)
PAE type : PAE
          DTB : 0x32a000L
          KDBG : 0x80545ce0L
Number of Processors : 1
Image Type (Service Pack) : 3
          KPCR for CPU 0 : 0xffffdff000L
          KUSER_SHARED_DATA : 0xffffdf0000L
Image date and time : 2014-10-29 17:20:02 UTC+0000
Image local date and time : 2014-10-29 12:20:02 -0500
bigjoe:volatility golden$ 
```

Quick Case Study: Memory Analysis vs. Spyware

eBlaster

- Simple but real case
- Computer infected with eBlaster
- Commercial spyware from SpectorSoft
- Very stealthy
- Uses code injection, hooks system calls, and other hiding techniques to avoid detection
- Affected party (“Jane”) noticed purchase of eBlaster on credit card statement and researched
- Jane wasn’t sure if it was installed or not
- Jane and her lawyer initiated forensic investigation

eBlaster

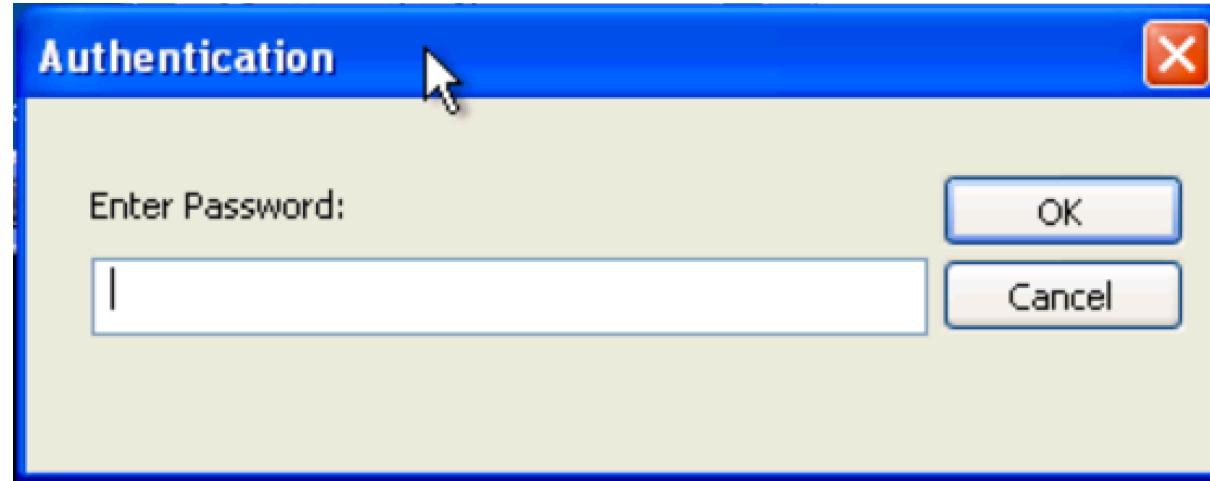
- Collects extensive information on computer activities
- Chat logs, email, web surfing, keystrokes, et al
- Emails reports to designated email address
- Reports and configuration are encrypted
- No access to configuration without password
- Questions:
 - What's being collected?
 - Where are the eBlaster reports being sent?

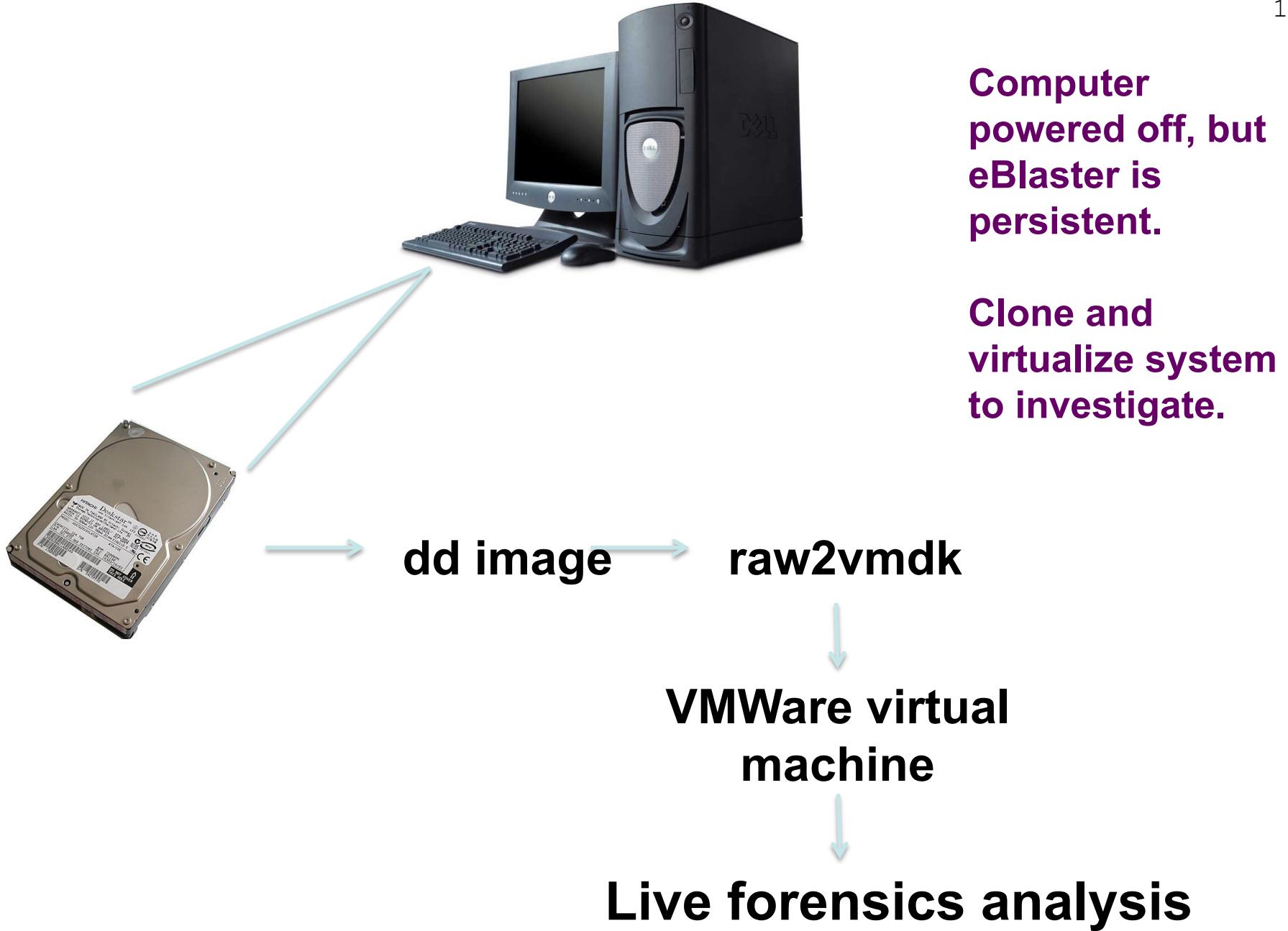
**Hotkey sequence brings up bare authentication dialog:
<CTRL><SHIFT><ALT>-T.**

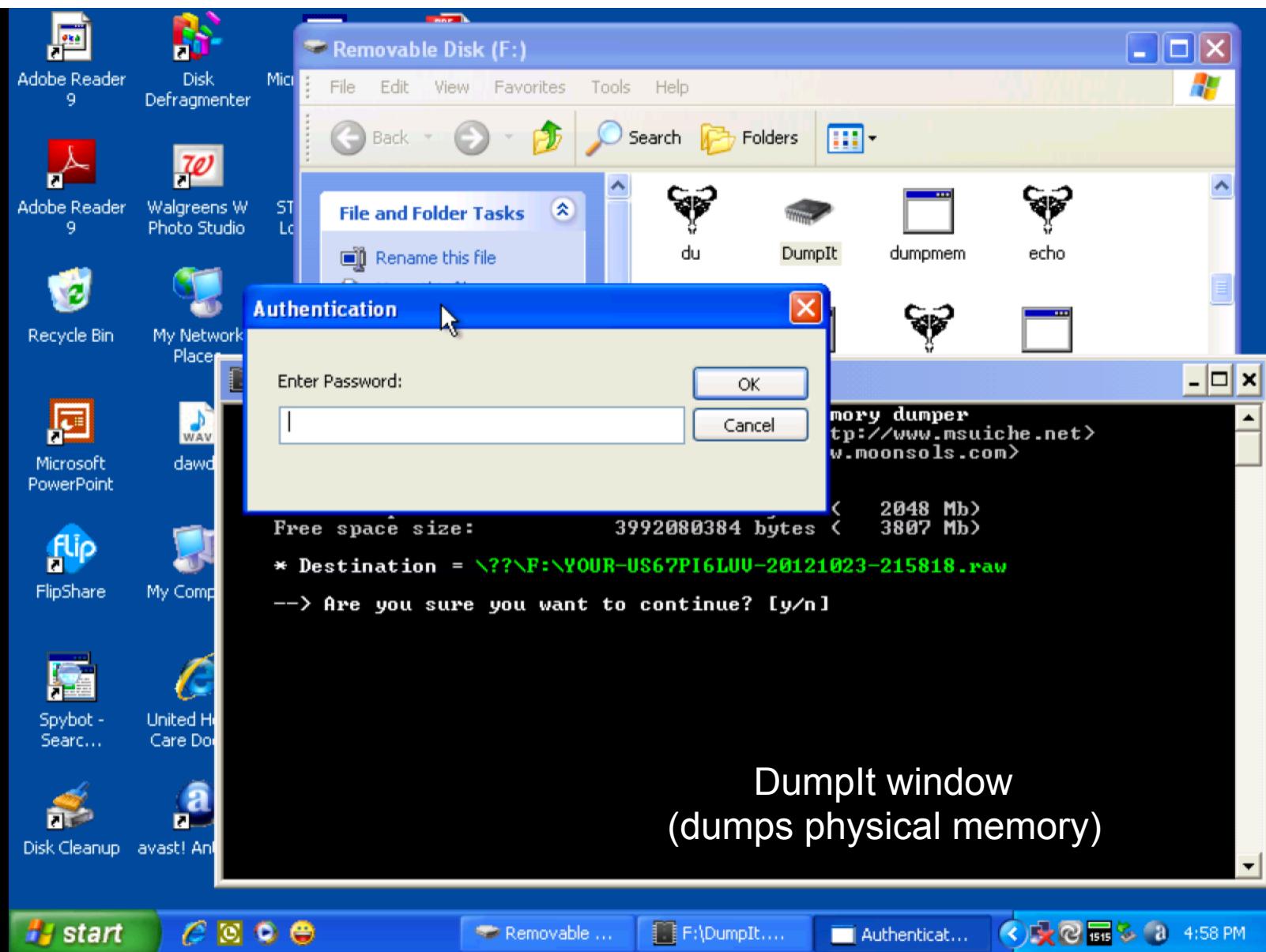
This sequence can be changed. Can't access configuration screen without the password.

Process is spawned to handle this dialog.

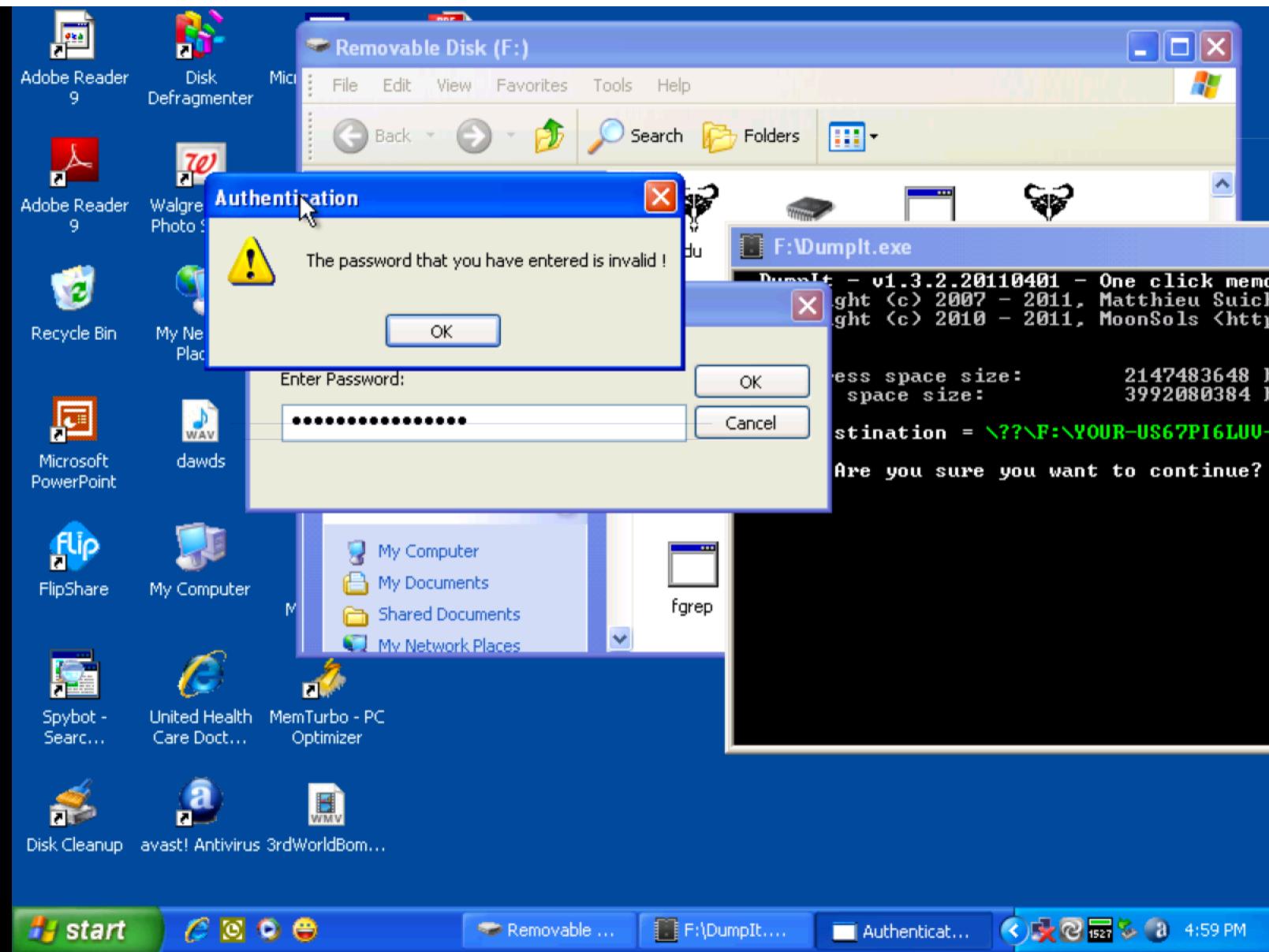
Process is terminated when dialog is dismissed.







DumpIt window
(dumps physical memory)



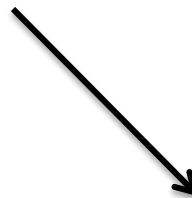
Hope: **quickly** reveal password w/o full reverse engineering effort.

Did developers leak the plaintext password?

Enter sample password “VERYUNIQUE” in password dialog.

Don’t dismiss the “invalid password” dialog.

Dump memory in the VM while the dialog box is displayed.



YOUR-US67PI6LUV-20121017-214253.raw

```
# gstrings -a -t d YOUR-US67PI6LUV-20121017-214253.raw | grep VERYUNIQUE > strings.txt
# gstrings -a -t d -e l YOUR-US67PI6LUV-20121017-214253.raw | grep VERYUNIQUE >> strings.txt
```

467105092:VERYUNIQUE

Search memory dump for invalid password VERYUNIQUE

```
# python vol.py --profile=WinXPSP3x86 -f YOUR-US67PI6LUV-20121017-214253.raw strings -s
strings.txt --output-file=stringslocation.txt -S
```

Map location of invalid pw to owning process

1bd77544 [3772:0012f544] VERYUNIQUE

PID of process whose address space includes string

```
# python vol.py --profile=WinXPSP3x86 -f YOUR-US67PI6LUV-20121017-214253.raw pslist
```

...

0x82c24bf0	xmlavipv.exe	3772	1592	2	78	0	0	2012-10-17 21:41:20
------------	--------------	------	------	---	----	---	---	---------------------

Verify that it's the process that gets spawned when dialog opens

```
# python vol.py --profile=WinXPSP3x86 -f YOUR-US67PI6LUV-20121017-214253.raw -p 3772 memdump
-D dumpdir
```

Dump memory of that process

Then run strings on dumped process memory

```
A 00004662: LEACCRC.DLL
A 00004757: WINDOWS\system32\OLEACCRC.DLL
A 00004e3c: bo 41
A 00004ec0: P0G~8
A 00004f94: Program Files\MemTurbo 4\cpurock
A 00005104: +)Bw(Z
A 0000511d: >?w(Z
A 000051d0: ! (
A 000053cf: ^? !
A 000058e5: I6ic\t4
A 00005c54: VERYUNIQUE
A 0000641c: the password that you have entered is invalid !
A 000064b8: Authentication
A 000064f1: VERYUNIQUE
A 00006544: 
A 0000667c: wdlgthis#2143
A 00006c4c: 
A 00009310: ACTUALPASSWORD mlavipv.exe"
A 000097be: !"#$%&'()*+,./0123456789:@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abc?
@defghijklmnopqrstuvwxyz{|}~^?
A 0000a4d5: ]q+D[o
A 0000ade8: C:\WINDOWS\system32\xmlavipv.exe
A 0000b0d0: WinSta0\Default
A 0000b905: r`w`J
A 0000d380: High Contrast Black (large)
A 00011bf1: & !
A 00012311: & !
A 0001253c: d0vth
A 00013b3c: %SystemRoot%\system32\mswsock.dll
A 00013c4c: temRoot%\system32\mswsock.dll
A 0001424c: %SystemRoot%\system32\mswsock.dll
A 0001454c: %CustomRoot%\Custom32\mswsock.dll
```

eBlaster Control Panel

Add Uninstall Help

eBLASTER® CONTROL PANEL

Report Delivery Sent Reports Report of Recent Activity

Activity Reports, Forwarding Services and Alerts are sent via email using the following settings:

Report Delivery Summary

Based on your current settings, eBlaster will send an Activity Report via email **every 60 minutes** to 'b...@...com'.

eBlaster will automatically forward Emails (including attachments), Chat / Instant Messages and Keyword Alerts to 'C...@...com'.

Send a Test Email

Send via Email to: [REDACTED] CC... ?

Activity Reports

Activity Report Delivery: On Off

Send a Report: Every 60 Minutes of Activity
 Once a day at 04:40 PM

Format Report as: HTML Plain Text

Email

Forward All Emails: On Off

Include Attachments: On Off

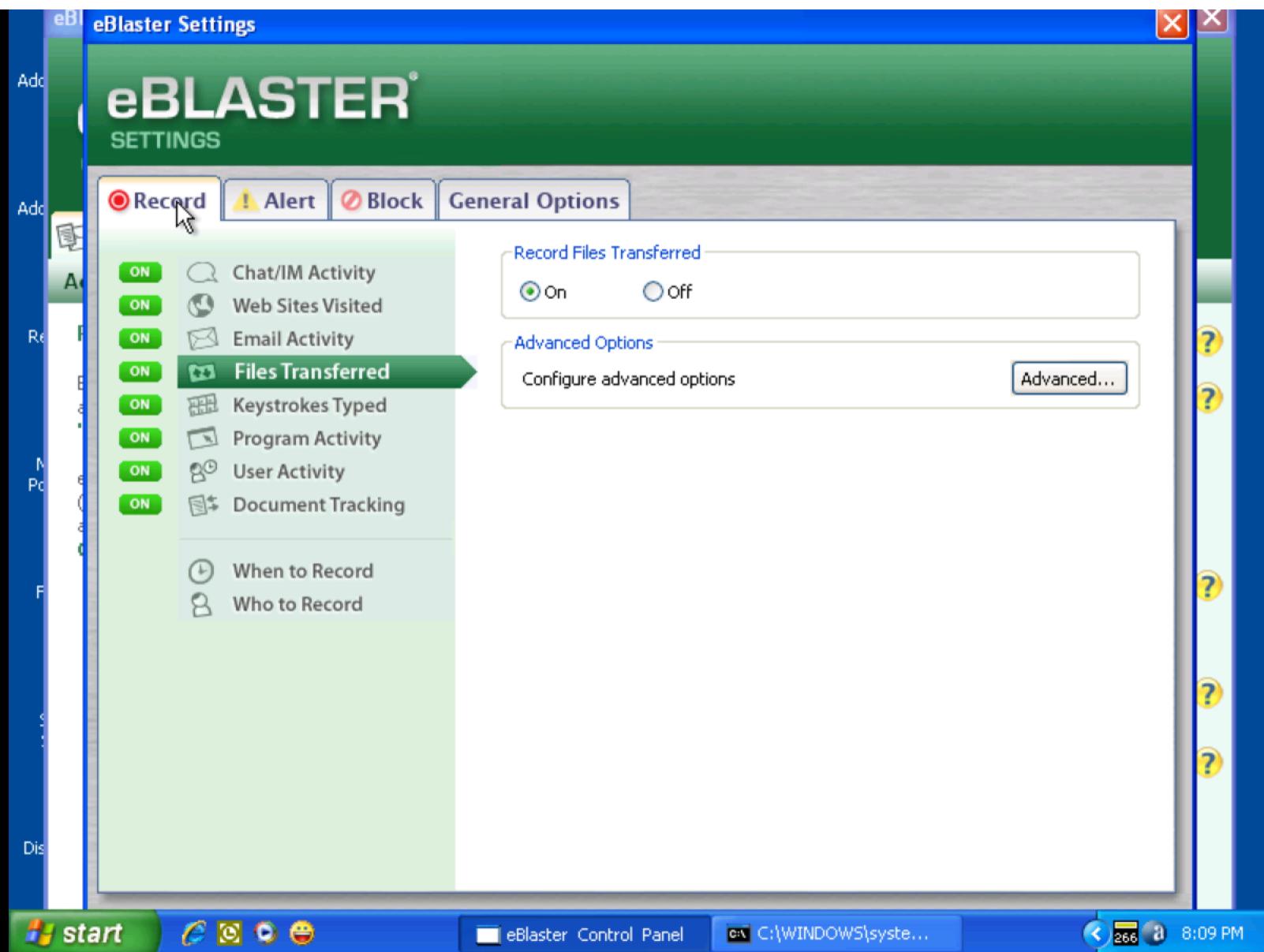
Chat / Instant Message

Forward All Chat/IMs: On Off

Alerts

Forward Keyword Alerts: On Off

start Internet Explorer Taskbar eBlaster Control Panel 247 6:36 PM



eBlaster Settings

eBLASTER[®]

SETTINGS

Record Alert Block General Options

Activity Report

Activity Report Delivery: On Off

Send a Report: Every 60 Minutes of Activity
 Once a day at 04:40 PM

Format Report as: HTML Plain Text XML

Show All IP Addresses Show MySpace Activity
 Show Online Searching Activity Show Facebook Activity

Emails

Forward All Emails: On Off
 Include Attachments
 Attach Original Email Headers

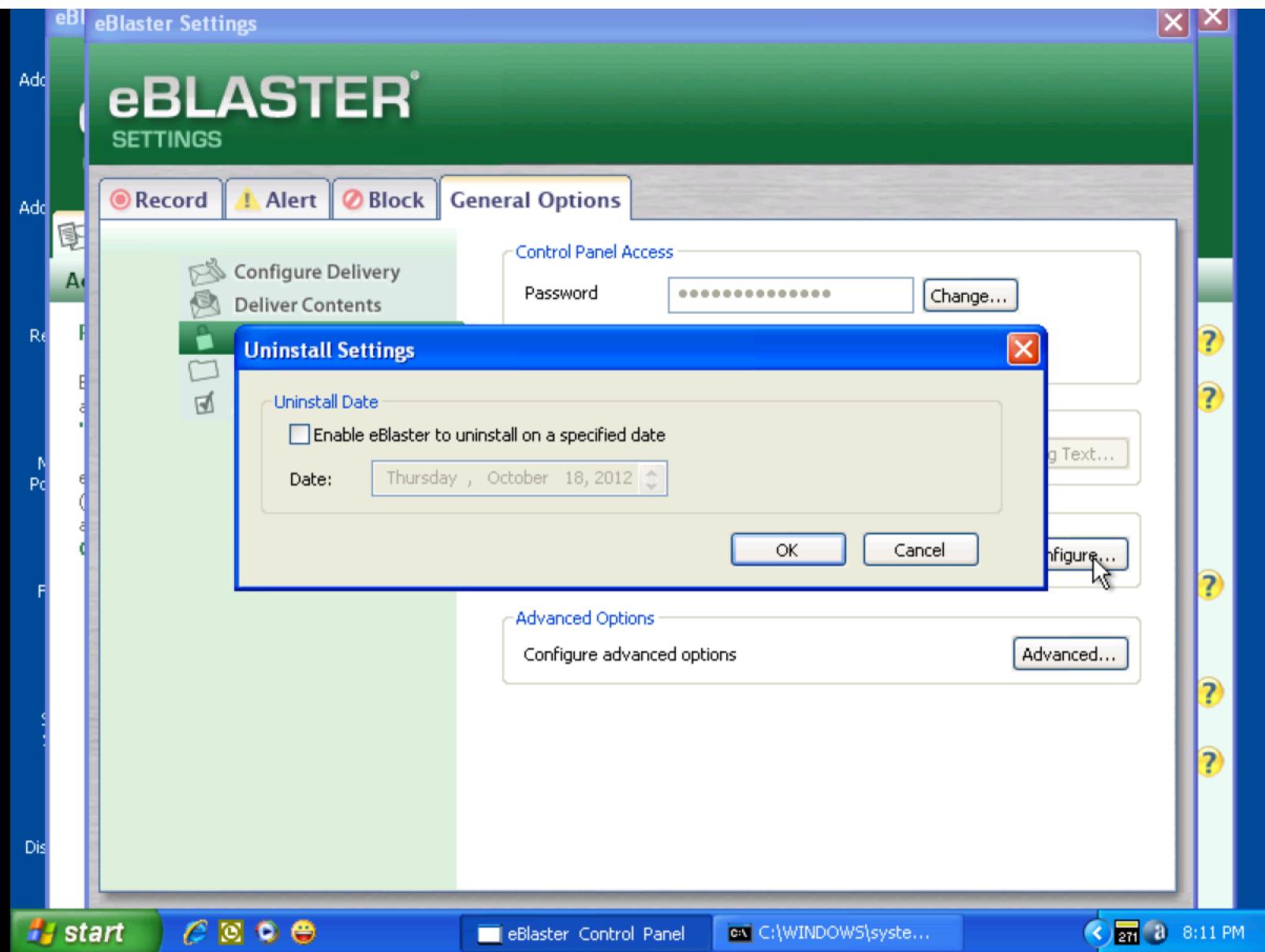
Chat / Instant Messages

Forward All Chat/IMs: On Off

Keyword Alerts

Forward Keyword Alerts: On Off

start eBlaster Control Panel C:\WINDOWS\system... 266 8:09 PM



Voila! Federal Wiretapping Charges!



Process Representations

Processes

- On all commodity operating systems, each executing application runs inside a process
- Basically, a “container” for:
 - Code
 - Data
 - Threads of execution
- Different representations on different operating systems
- But basic information is similar

Process Information

- Process state (running, stopped, etc.)
- Priority
- Links to next, previous processes
- PID
- Pointers to parent, children
- Permissions
- Accounting info
- Limits
- Filesystem stuff
- Memory management

Windows Processes

Windows 7 _EPROCESS structure

See pp. 151
in AMF

```
>>> dt("_EPROCESS")
'_EPROCESS' (1232 bytes)
0x0    : Pcb
0x160  : ProcessLock
0x168  : CreateTime
0x170  : ExitTime
0x178  : RundownProtect
0x180  : UniqueProcessId
0x188  : ActiveProcessLinks
0x198  : ProcessQuotaUsage
0x1a8  : ProcessQuotaPeak
0x1b8  : CommitCharge
0x1c0  : QuotaBlock
0x1c8  : CpuQuotaBlock
0x1d0  : PeakVirtualSize
0x1d8  : VirtualSize
0x1e0  : SessionProcessLinks
0x1f0  : DebugPort
[snip]
0x200  : ObjectTable
0x208  : Token
0x210  : WorkingSetPage
0x218  : AddressCreationLock
[snip]
0x290  : InheritedFromUniqueProcessId      ['unsigned int']
[snip]
0x2d0  : PageDirectoryPte          ['_HARDWARE_PTE']
0x2d8  : Session                ['pointer64', ['void']]
0x2e0  : ImageFileName          ['String', {'length': 16}]
0x2ef  : PriorityClass           ['unsigned char']
0x2f0  : JobLinks                ['_LIST_ENTRY']
0x300  : LockedPagesList         ['pointer64', ['void']]
0x308  : ThreadListHead        ['_LIST_ENTRY']
0x318  : SecurityPort            ['pointer64', ['void']]
0x320  : Wow64Process             ['pointer64', ['void']]
0x328  : ActiveThreads          ['unsigned long']

['_KPROCESS']
['_EX_PUSH_LOCK']
['WinTimeStamp', {'is_utc': True}]
['WinTimeStamp', {'is_utc': True}]
['_EX_RUNDOWN_REF']
['unsigned int']
['_LIST_ENTRY']
['array', 2, ['unsigned long long']]
['array', 2, ['unsigned long long']]
['unsigned long long']
['pointer64', ['_EPROCESS_QUOTA_BLOCK']]
['pointer64', ['_PS_CPU_QUOTA_BLOCK']]
['unsigned long long']
['unsigned long long']
['_LIST_ENTRY']
['pointer64', ['void']]

['pointer64', ['_HANDLE_TABLE']]
['_EX_FAST_REF']
['unsigned long long']
['_EX_PUSH_LOCK']

['unsigned int']

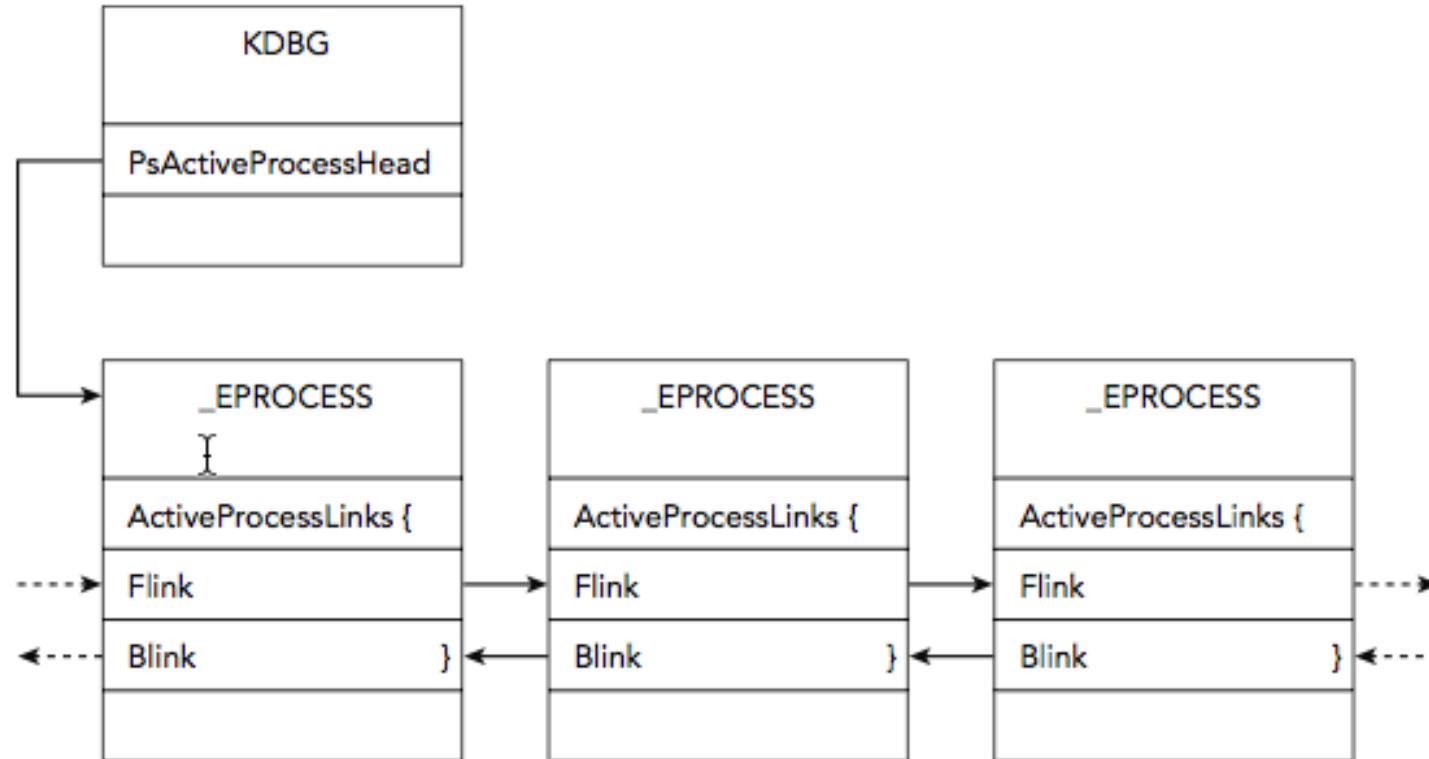
['pointer64', ['void']]
['String', {'length': 16}]
['unsigned char']
['_LIST_ENTRY']
['pointer64', ['void']]
['_LIST_ENTRY']
['pointer64', ['void']]
['pointer64', ['void']]
['unsigned long']
```

```
0x32c : ImagePathHash           ['unsigned long']
0x330 : DefaultHardErrorProcessing ['unsigned long']
0x334 : LastThreadExitStatus    ['long']
0x338 : Peb                  ['pointer64', ['_PEB']]
[snip]
0x444 : ExitStatus             ['long']
0x448 : VadRoot              ['_MM_AVL_TABLE']
0x488 : AlpcContext            ['_ALPC_PROCESS_CONTEXT']
0x4a8 : TimerResolutionLink   ['_LIST_ENTRY']
0x4b8 : RequestedTimerResolution ['unsigned long']
0x4bc : ActiveThreadsHighWatermark ['unsigned long']
0x4c0 : SmallestTimerResolution ['unsigned long']
```

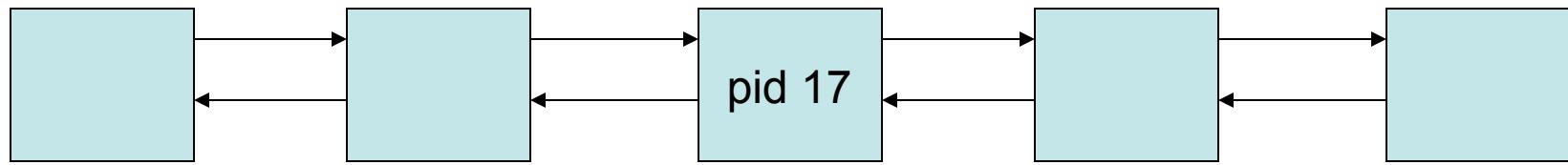
Windows Processes

- Windows organizes active processes using a doubly linked list pointed to by PsActiveProcessHead
 - Rootkits can unlink entries to hide processes from tools on the running system
 - Does not interrupt process execution because the CPU scheduler is thread-based

Linked List of Processes

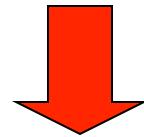


DKOM: FU “Rootkit” under Windows

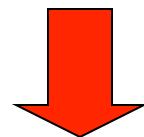


Doubly-linked process list in Windows kernel

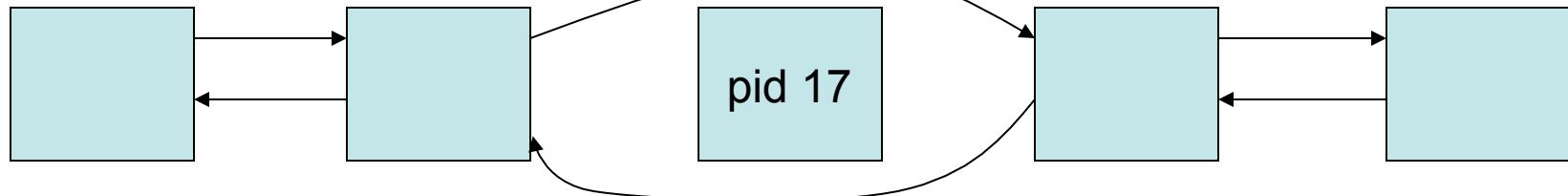
Example of **DKOM**:
Direct Kernel Object Manipulation



C:\> fu -ph 17



Processes continue
to run because Windows
scheduler handles
threads, not processes



Identifying System Processes

- Important to be able to identify (real!) system processes to eliminate them from suspicion

pp. 155-156 in AMF

- Idle and system: These are not real processes (in the sense that they have no corresponding executable on disk). Idle is just a container that the kernel uses to charge CPU time for idle threads. Similarly, System serves as the default home for threads that run in kernel mode. Thus, the System process (PID 4) appears to own any sockets or handles to files that kernel modules open.
- csrss.exe: The client/server runtime subsystem plays a role in creating and deleting processes and threads. It maintains a private list of the objects that you can use to cross-reference with other data sources. On systems before Windows 7, this process also served as the broker of commands executed via cmd.exe, so you can extract command history from its memory space. Expect to see multiple CSRSS processes because each session gets a dedicated copy; however, watch out for attempts to exploit the naming convention (csrsss.exe or cssrs.exe). The real one is located in the system32 directory.

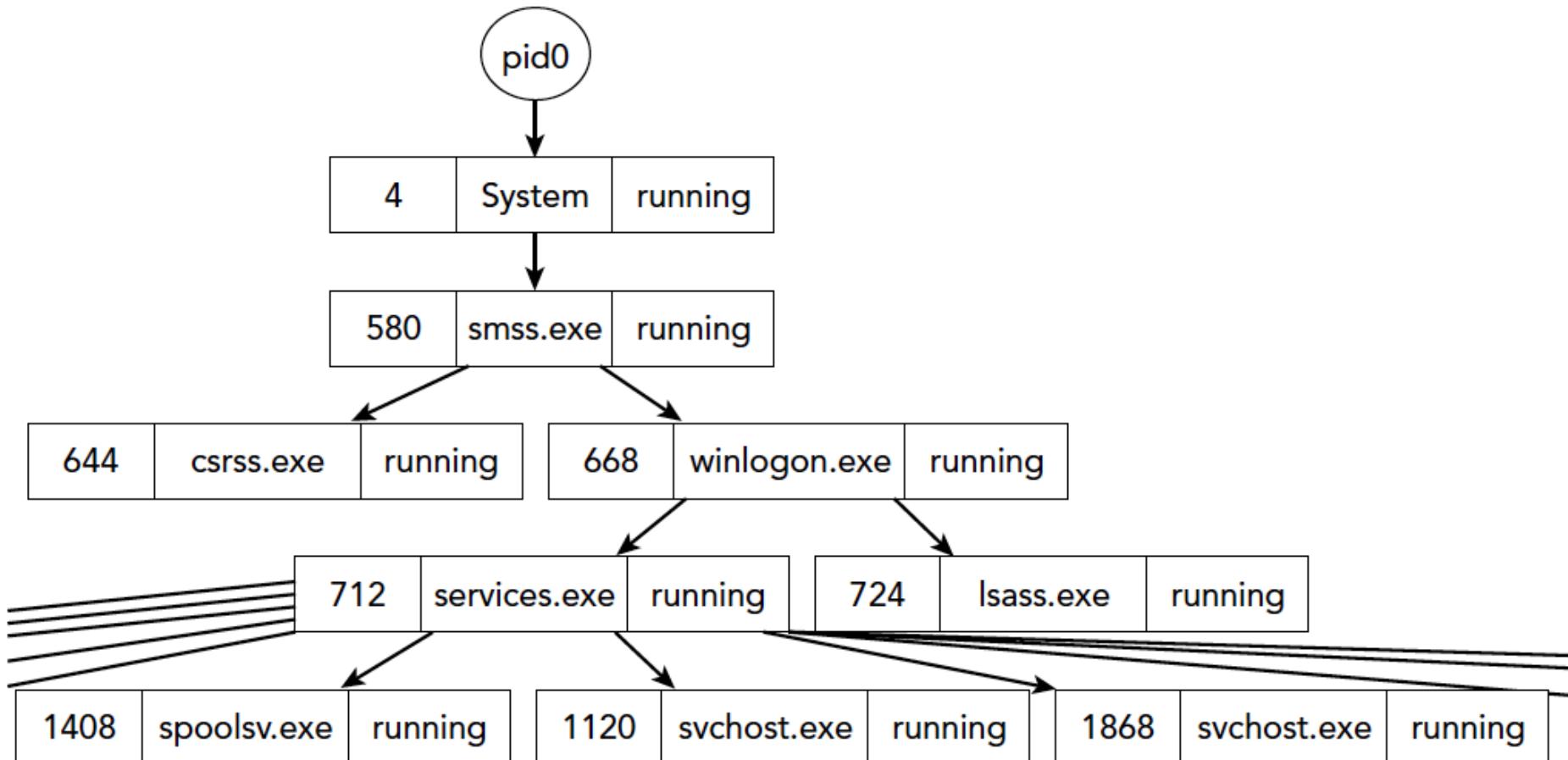
Identifying (2)

- **services.exe:** The Service Control Manager (SCM) is described more thoroughly in Chapter 12, but in short, it manages Windows services and maintains a list of such services in its private memory space. This process should be the parent for any svchost.exe (service host) instances that you see, in addition to processes such as spoolsv.exe and SearchIndexer.exe that implement services. There should be only one copy of services.exe on a system, and it should be running from the system32 directory.
- **svchost.exe:** A clean system has multiple shared host processes running concurrently, each providing a container for DLLs that implement services. As previously mentioned, their parent should be services.exe, and the path to their executable should point to the system32 directory. In his blog, Patrick identifies a few of the common names (such as scvhost.exe and svch0st.exe) used by malware to blend in with these processes.
- **lsass.exe:** The local security authority subsystem process is responsible for enforcing the security policy, verifying passwords, and creating access tokens. As such, it's often the target of code injection because the plaintext password hashes can be found in its private memory space. There should be only one instance of lsass.exe running from the system32 directory, and its parent is winlogon.exe on pre-Vista machines, and wininit.exe on Vista and later systems. Stuxnet created two fake copies of lsass.exe, which caused them to stick out like a sore thumb.

Identifying (3)

- **winlogon.exe:** This process presents the interactive logon prompt, initiates the screen saver when necessary, helps load user profiles, and responds to Secure Attention Sequence (SAS) keyboard operations such as CTRL+ALT+DEL. Also, this process monitors files and directories for changes on systems that implement Windows File Protection (WFP). As with most other critical processes, its executable is located in the system32 directory.
- **explorer.exe:** You'll see one Windows Explorer process for each logged-on user. It is responsible for handling a variety of user interactions such as GUI-based folder navigation, presenting the start menu, and so on. It also has access to sensitive material such as the documents you open and credentials you use to log in to FTP sites via Windows Explorer.
- **smss.exe:** The session manager is the first real user-mode process that starts during the boot sequence. It is responsible for creating the sessions (see Chapter 14) that isolate OS services from the various users who may log on via the console or Remote Desktop Protocol (RDP).

Visualization of Clean Systems



Windows Process Listing Plugins

- **pslist**
 - enumerates processes by walking the active list
 - will miss unlinked processes
- **psscan**
 - enumerates processes using the pool tag scanning technique
 - finds unlinked processes
 - can output a Graphviz-compatible diagram
- **pstree**
 - shows a tree of parent/child relationships
- **psxview**
 - enumerates processes 7 different ways and compares them to show discrepancies

Windows Process Lists

```
$ python vol.py -f lab.mem --profile=WinXPSP3x86 pslist
```

Volatility Foundation Volatility Framework 2.4

Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Start
0x823c8830	System	4	0	56	537	-----	
0x81e7e180	smss.exe	580	4	3	19	-----	2013-03-14 03:02:22
0x82315da0	csrss.exe	644	580	10	449	0	2013-03-14 03:02:25
0x81f37948	winlogon.exe	668	580	18	515	0	2013-03-14 03:02:26
0x81fec128	services.exe	712	668	15	281	0	2013-03-14 03:02:27
[snip]							
0x81eb4300	vmtoolsd.exe	1684	1300	6	213	0	2013-03-14 03:02:45
0x8210b9c8	IEXPLORE.EXE	1764	1300	16	642	0	2013-03-14 03:03:04
0x81e79020	firefox.exe	180	1300	27	447	0	2013-03-14 03:03:05
0x81cb63d0	wuauctl.exe	1576	1072	3	104	0	2013-03-14 03:03:40
0x81e86bf8	alg.exe	1836	712	5	102	0	2013-03-14 03:04:00
0x8209eda0	wscntfy.exe	2672	1072	1	28	0	2013-03-14 03:04:01
0x82013340	jucheck.exe	2388	1656	2	104	0	2013-03-14 03:07:45
0x81e79418	thunderbird.exe	3832	1300	30	339	0	2013-03-14 03:12:54
0x8202b398	AcroRd32.exe	3684	180	0	-----	0	2013-03-14 14:19:16
0x81ecd3c0	cmd.exe	3812	3684	1	33	0	2013-03-14 14:19:29
0x81f55bd0	a[1].php	2280	3812	1	139	0	2013-03-14 14:19:30
0x8223b738	IEXPLORE.EXE	2276	2280	7	280	0	2013-03-14 14:19:32
0x822c8a58	AcroRd32.exe	2644	180	0	-----	0	2013-03-14 14:40:16

Parent / Child Relationships

```
$ python vol.py -f lab.mem --profile=WinXPSP3x86 pstree
```

```
Volatility Foundation Volatility Framework 2.4
```

[snip]

0x82263378:explorer.exe	1300	1188	11	363	2013-03-14	03:02:42
. 0x81e85da0:TSVNCache.exe	1556	1300	7	53	2013-03-14	03:02:43
. 0x81e79020:firefox.exe	180	1300	27	447	2013-03-14	03:03:05
.. 0x8202b398:AcroRd32.exe	3684	180	0	-----	2013-03-14	14:19:16
... 0x81ecd3c0:cmd.exe	3812	3684	1	33	2013-03-14	14:19:29
.... 0x81f55bd0:a[1].php	2280	3812	1	139	2013-03-14	14:19:30
..... 0x8223b738:IEXPLORE.EXE	2276	2280	7	280	2013-03-14	14:19:32

```
bigjoe:volatility golden$ python vol.py --profile=Win7SP1x64 -f /Volumes/malware/winpmem-1.4/after-republic-2.dmp pstree
Volatility Foundation Volatility Framework 2.4
Name          Pid  PPid  Thds  Hnds Time
-----+-----+-----+-----+-----+-----+
0xfffffa8008f1eb30:iexplore.exe      1240  1100   21    384 2015-06-11 19:38:07 UTC+0000 ←
. 0xfffffa8008a12b30:taskmgr.exe     2176  1240    6    129 2015-06-11 19:38:08 UTC+0000
0xfffffa80072fe740:System           4      0   103    325 2015-06-11 19:37:53 UTC+0000
. 0xfffffa800855a170:smss.exe       264      4    2     36 2015-06-11 19:37:53 UTC+0000
0xfffffa80087a96c0:wininit.exe     392      336   3     82 2015-06-11 19:37:55 UTC+0000
. 0xfffffa80088b9420:lsm.exe        504      392   9    142 2015-06-11 19:37:56 UTC+0000
. 0xfffffa80088cf4d0:services.exe   488      392   9    198 2015-06-11 19:37:56 UTC+0000
.. 0xfffffa8008be9a30:spoolsv.exe   1036     488  12    313 2015-06-11 19:37:59 UTC+0000
.. 0xfffffa80089ffa30:svhost.exe   856      488  23    848 2015-06-11 19:37:57 UTC+0000
.. 0xfffffa80089d4060:svhost.exe   772      488  17    430 2015-06-11 19:37:57 UTC+0000
.. 0xfffffa800744a060:svhost.exe   2076     488  11    339 2015-06-11 19:40:02 UTC+0000
.. 0xfffffa800899f690:svhost.exe   676      488  6     219 2015-06-11 19:37:57 UTC+0000
.. 0xfffffa80089ed6a0:svhost.exe   812      488  20    533 2015-06-11 19:37:57 UTC+0000
... 0xfffffa8008df8860:WUDFHost.exe 1800     812  8     231 2015-06-11 19:38:02 UTC+0000
... 0xfffffa8008dceb30:dwm.exe      1660     812  3     77  2015-06-11 19:38:01 UTC+0000
... 0xfffffa80073ba060:sppsvc.exe   3032     488  4     163 2015-06-11 19:40:02 UTC+0000
... 0xfffffa8008c07b30:svhost.exe   3004     488  5     75  2015-06-11 19:40:01 UTC+0000
... 0xfffffa8008c246a0:svhost.exe   1080     488  18    333 2015-06-11 19:37:59 UTC+0000
... 0xfffffa8008d00910:taskhost.exe 1364     488  7     136 2015-06-11 19:38:01 UTC+0000
... 0xfffffa8008976060:svhost.exe   600      488  9     372 2015-06-11 19:37:57 UTC+0000
... 0xfffffa8008dd62f0:svhost.exe   1680     488  6     100 2015-06-11 19:38:01 UTC+0000
... 0xfffffa8008a7ab30:svhost.exe   996      488  10    250 2015-06-11 19:37:58 UTC+0000
... 0xfffffa8008ac56a0:svhost.exe   340      488  13    343 2015-06-11 19:37:59 UTC+0000
. 0xfffffa80088b1b30:lsass.exe     496      392  7     458 2015-06-11 19:37:56 UTC+0000
0xfffffa8008360af60:csrss.exe     344      336  11    264 2015-06-11 19:37:55 UTC+0000
0xfffffa8008738780:csrss.exe     400      384  9     147 2015-06-11 19:37:55 UTC+0000
. 0xfffffa8009020b30:conhost.exe  2660     400  2     53  2015-06-11 19:38:27 UTC+0000
0xfffffa80088a9b30:winlogon.exe   440      384  3     116 2015-06-11 19:37:56 UTC+0000
0xfffffa8008e0d060:explorer.exe   1784     1532  17    684 2015-06-11 19:38:02 UTC+0000
. 0xfffffa8009021060:cmd.exe      2652     1784  1     20  2015-06-11 19:38:27 UTC+0000
.. 0xfffffa800743bb30:winpmem_1.4.ex 2848     2652  1     29  2015-06-11 19:39:04 UTC+0000
0xfffffa8008f0a060:iexplore.exe   1916     1140  4     73  2015-06-11 19:38:06 UTC+0000
0xfffffa8008ca3b30:iexplore.exe   1908     1140  10    127 2015-06-11 19:38:06 UTC+0000

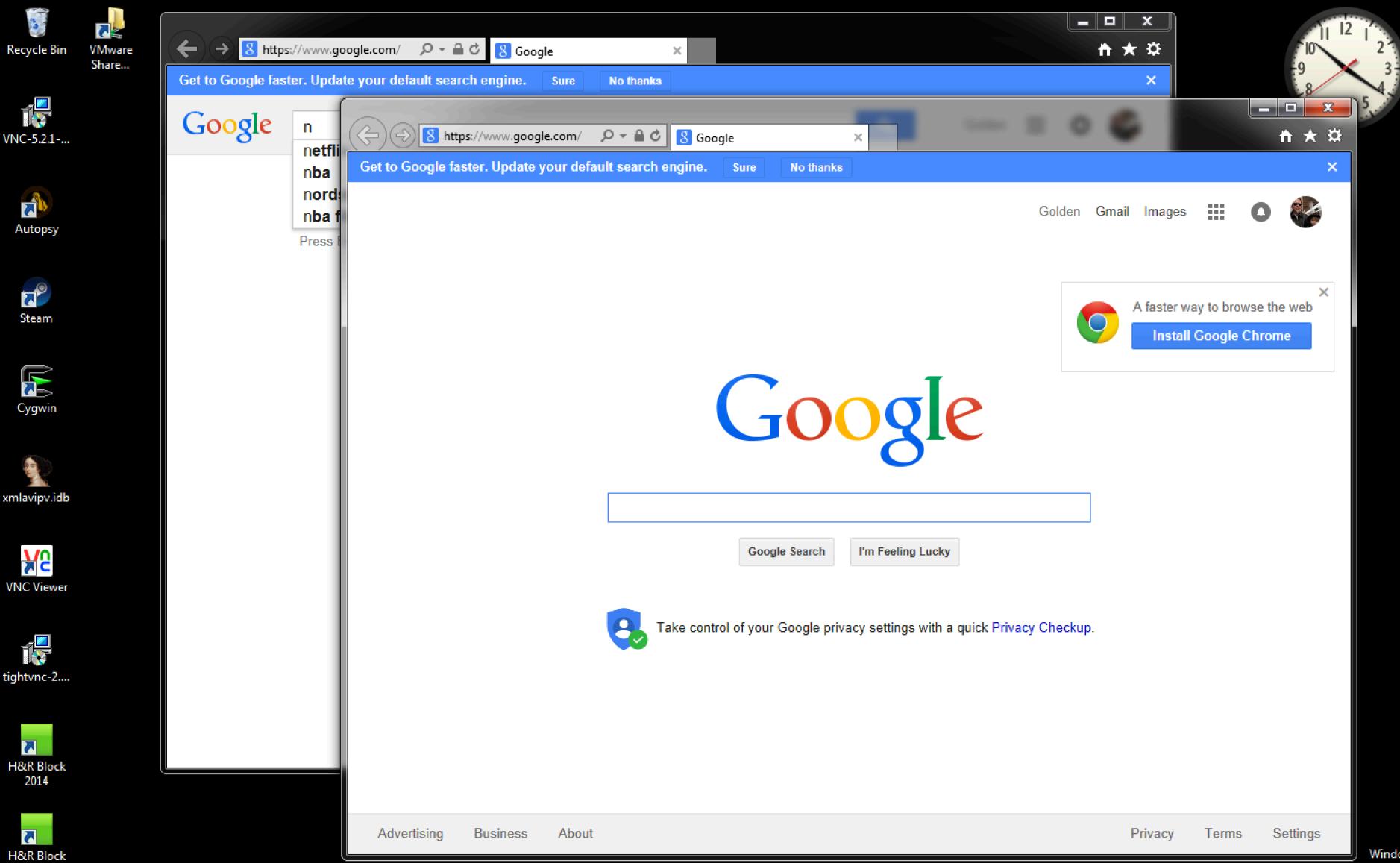
```

```
bigjoe:volatility golden$ python vol.py --profile=Win7SP1x64 -f /Volumes/malware/winpmem-1.4/after-republic-2.dmp iehistory
Volatility Foundation Volatility Framework 2.4
```

```
bigjoe:volatility golden$ mkdir DUMP && python vol.py --profile=Win7SP1x64 -f /Volumes/malware/winpmem-1.4/after-republic-2.dmp procdump -p 1240 -D DUMP
Volatility Foundation Volatility Framework 2.4
```

Process(V)	ImageBase	Name	Result
------------	-----------	------	--------

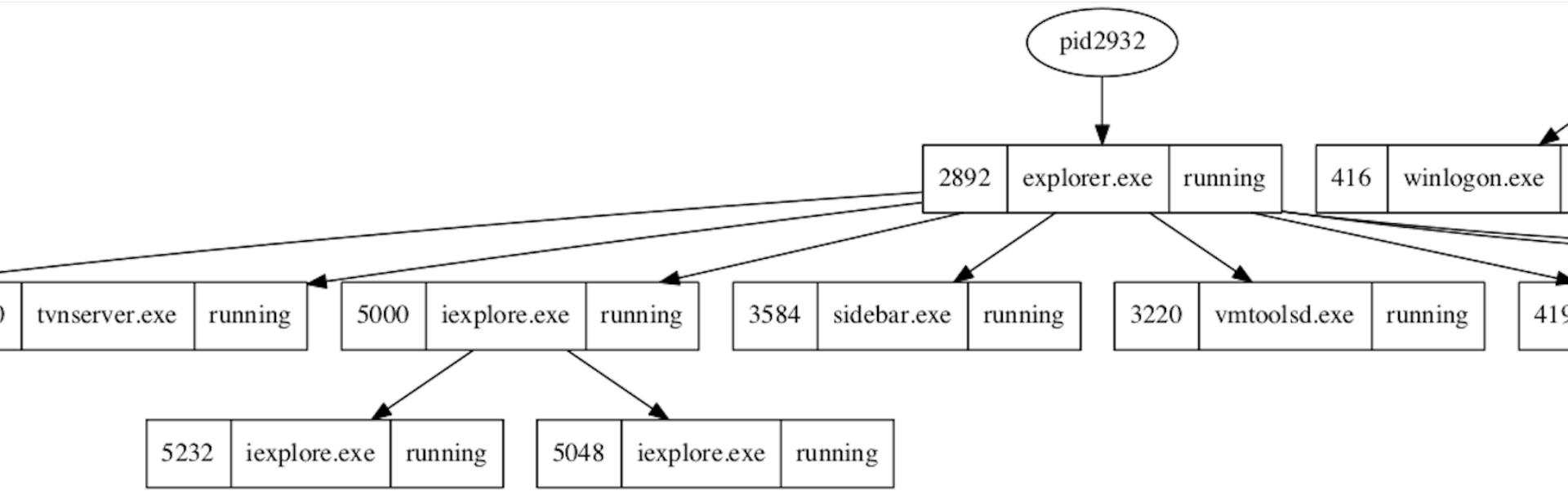
0xfffffa8008f1eb30	0x00000000008b0000	iexplore.exe	OK: executable.1240.exe
--------------------	--------------------	--------------	-------------------------



1. bash

.. 0xfffffa8049cd0060:CodeMeter.exe	1516	464	11	131	2015-06-19 03:16:05	UTC+0000
.. 0xfffffa804a229630:dllhost.exe	3328	464	17	204	2015-06-19 03:18:02	UTC+0000
.. 0xfffffa8049ecdb30:svchost.exe	2088	464	11	199	2015-06-19 03:16:08	UTC+0000
.. 0xfffffa8049dea240:rpcld.exe	1908	464	10	252	2015-06-19 03:16:06	UTC+0000
.. 0xfffffa80495b1060:eEBSvc.exe	1300	464	33	177	2015-06-19 03:16:04	UTC+0000
.. 0xfffffa8049ba9b30:svchost.exe	892	464	25	496	2015-06-19 03:16:03	UTC+0000
... 0xfffffa8048526b30:dwm.exe	4024	892	6	138	2015-06-19 03:18:40	UTC+0000
.. 0xfffffa8049f66b30:tvnserver.exe	2344	464	17	164	2015-06-19 03:16:09	UTC+0000
.. 0xfffffa8049da6b30:svchost.exe	1824	464	4	61	2015-06-19 03:16:06	UTC+0000
. 0xfffffa80495aeb30:lsass.exe	476	348	8	770	2015-06-19 03:16:02	UTC+0000
. 0xfffffa80495b6b30:lsm.exe	484	348	13	212	2015-06-19 03:16:02	UTC+0000
0xfffffa804958db30:csrss.exe	308	296	9	889	2015-06-19 03:16:01	UTC+0000
. 0xfffffa804aa7d060:conhost.exe	5540	308	1	33	2015-06-19 03:20:45	UTC+0000
0xfffffa804847c890:System	4	0	95	530	2015-06-19 03:15:57	UTC+0000
. 0xfffffa8048c2a040:smss.exe	224	4	2	30	2015-06-19 03:15:57	UTC+0000
0xfffffa804a42a630:explorer.exe	2892	2932	28	676	2015-06-19 03:18:40	UTC+0000
. 0xfffffa8048ebc060:tvnserver.exe	3600	2892	4	65	2015-06-19 03:18:45	UTC+0000
. 0xfffffa8049f54410:iexplore.exe	5000	2892	35	626	2015-06-19 03:19:29	UTC+0000
.. 0xfffffa804a54a800:iexplore.exe	5048	5000	34	686	2015-06-19 03:19:29	UTC+0000
.. 0xfffffa8049ef1060:iexplore.exe	5232	5000	32	648	2015-06-19 03:20:25	UTC+0000
. 0xfffffa8049fa0270:hpqtra08.exe	4212	2892	9	165	2015-06-19 03:18:49	UTC+0000
. 0xfffffa8049b7eb30:vmtoolsd.exe	3220	2892	9	256	2015-06-19 03:18:44	UTC+0000
. 0xfffffa8049f9c740:netsession_win	4128	2892	5	102	2015-06-19 03:18:46	UTC+0000
.. 0xfffffa804a032b30:netsession_win	4148	4128	12	339	2015-06-19 03:18:46	UTC+0000
. 0xfffffa8049cb7060:MSASCui.exe	3464	2892	15	389	2015-06-19 03:18:44	UTC+0000
. 0xfffffa804a199060:CodeMeterCC.exe	4192	2892	7	148	2015-06-19 03:18:48	UTC+0000
. 0xfffffa8049cb6b30:sidebar.exe	3584	2892	18	313	2015-06-19 03:18:45	UTC+0000
0xfffffa804a2e23f0:hpwusched2.exe	4348	4164	2	36	2015-06-19 03:18:50	UTC+0000
0xfffffa80499b49e0:acrobat_sl.exe	4268	4164	6	61	2015-06-19 03:18:49	UTC+0000
0xfffffa804a300870:crotray.exe	4288	4164	3	62	2015-06-19 03:18:49	UTC+0000
0xfffffa804a6deb30:Steam.exe	4084	4156	23	451	2015-06-19 03:19:36	UTC+0000
. 0xfffffa804a71a9e0:steamwebhelper	4012	4084	26	290	2015-06-19 03:19:37	UTC+0000
0xfffffa8049461880:winlogon.exe	416	340	6	132	2015-06-19 03:16:02	UTC+0000
0xfffffa8049421b30:csrss.exe	356	340	10	498	2015-06-19 03:16:02	UTC+0000
garfish:volatility golden\$						

```
garfish:volatility golden$ python vol.py --profile=Win7SP1x64 -f ~/Documents/Virtual\ Machines.localized/Windows\ 7\x64/Windows\ 7\x64-1e433ca5.vmem psscan --output=dot --output-file=win7-clean.dot
Volatility Foundation Volatility Framework 2.4
garfish:volatility golden$
```



Cross-referencing Kernel Structures

- Malware can easily hide from analysis tools which rely on one information source
- e.g., process list
- Expand scope by cross-referencing kernel structures that provide similar information
- **psxview** plugin accomplishes this

psxview Sources

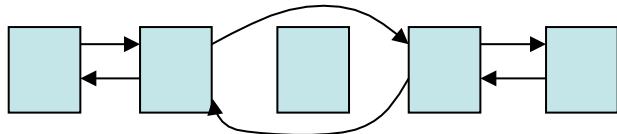
- Scan for process objects using **pool scanning** (e.g., carving)
- Scan for `_ETHREAD` objects and backtrace to `_EPROCESS`
- Structures maintained by `csrss` process
 - Maintains handles to created processes
- `PspCID` table
 - References active processes and threads
- Login session and desktop data structures
 - Also reference groups of processes

Windows XP SP3

Offset(P)	Name	PID	pslist	psscan	thrdproc	pspcid	csrss	session	deskthrd	ExitTime
0x0a96f558	GrooveMonitor.e	592	True	True	False	True	True	True	True	
0x0a72b280	GoogleUpdateSet	1492	True	True	False	True	True	True	True	
0x0a527a48	MagicDisc.exe	1476	True	True	False	True	True	True	True	
0x0a539248	AdobeARM.exe	624	True	True	False	True	True	True	True	
0x0a402da0	hasplms.exe	3228	True	True	False	True	True	True	True	
0x09f82308	jqs.exe	3392	True	True	False	True	True	True	True	
0x0a7e3c10	vmacthlp.exe	1316	True	True	False	True	True	True	True	
0x0a98e558	winlogon.exe	1084	True	True	False	True	True	True	True	
0x0a9a39a0	BTTTray.exe	948	True	True	False	True	True	True	True	
0x0a8c6da0	svchost.exe	1560	True	True	False	True	True	True	True	
0x0a5a5b20	svchost.exe	3608	True	True	False	True	True	True	True	
0x0a9a8da0	services.exe	1144	True	True	False	True	True	True	True	
0x0aa44af0	mdm.exe	3496	True	True	False	True	True	True	True	
0x0a48dda0	prt_k_worker_ser	1608	True	True	False	True	True	True	False	
0x0aa3d928	svchost.exe	1660	True	True	False	True	True	True	True	
0x0a58bda0	jusched.exe	708	True	True	False	True	True	True	True	
0x0a222830	vmtoolsd.exe	700	True	True	False	True	True	True	True	
0x0a53eac8	ctfmon.exe	716	True	True	False	True	True	True	True	
0x0a981da0	ISUSPM.exe	600	True	True	False	True	True	True	True	
0x0a81a808	svchost.exe	1616	True	True	False	True	True	True	True	
0x0a46dda0	GoogleUpdate.ex	2892	True	True	False	True	True	True	True	
0x0a7d67b8	ONENOTEM.EXE	1500	True	True	False	True	True	True	True	
0x0a49aad0	svchost.exe	1688	True	True	False	True	True	True	True	
0x0a96fda0	explorer.exe	452	True	True	False	True	True	True	True	
0x0a801da0	lsass.exe	1156	True	True	False	True	True	True	True	
0x0a370da0	rundll32.exe	360	True	True	False	True	True	True	True	
0x0a865020	acrotray.exe	616	True	True	False	True	True	True	True	
0x0a7e0648	svchost.exe	1328	True	True	False	True	True	True	True	
0x0a499c90	prt_k_supervisor	1596	True	True	False	True	True	True	False	
0x0a569248	rundll32.exe	684	True	True	False	True	True	True	True	
0x0a993558	rapimgn.exe	912	True	True	False	True	True	True	True	
0x0a973da0	WgaTray.exe	432	True	True	False	True	True	True	True	
0x09fb6250	cvpnd.exe	3164	True	True	False	True	True	True	True	
0x0a7bc020	mysqld-nt.exe	3556	True	True	False	True	True	True	True	
0x0a8045d8	svchost.exe	1708	True	True	False	True	True	True	True	
0x0a8b6da0	ADEngineW.exe	1704	True	True	False	True	True	True	True	
0x0a8bad08	ADEngineS.exe	1644	True	True	False	True	True	True	True	
0x0a36cae0	acrobat_sl.exe	608	True	True	False	True	True	True	True	
0x09f991d0	svchost.exe	3336	True	True	False	True	True	True	True	
0x09fb7120	InstallShield L	3360	True	True	False	True	True	True	True	
0x0a9892f0	svchost.exe	1420	True	True	False	True	True	True	True	
0x0a95e528	CodeMeterCC.exe	996	True	True	False	True	True	True	True	
0x0a7dd578	spoolsv.exe	2000	True	True	False	True	True	True	True	
0x0a4b6da0	CodeMeter.exe	1756	True	True	False	True	True	True	True	
0x0a846020	wcescomm.exe	840	True	True	False	True	True	True	True	
0x0a6a79e0	CodeMeter.exe	1536	True	True	False	True	False	False	False	2013-12-01 06:03:32 UTC+0000
0x0ac39830	System	4	True	True	False	True	False	False	False	
0x0a8123b8	csrss.exe	1016	True	True	False	True	False	True	True	
0x0a839020	smss.exe	672	True	True	False	True	False	False	False	

Windows XP SP3

FU on PID 2260



Offset(P)	Name	PID	pplist	psscan	thrdproc	pspcid	csrss	session	deskthrd	ExitTime
0x0a720180	prtk_worker_ser	360	True	False	True	True	True	False		
0x097e0c68	vmtoolsd.exe	3932	True	False	True	True	True	True		
0x0a4e0da0	InstallShield L	1292	True	False	True	True	True	True		
0x0a0d9888	services.exe	1152	True	False	True	True	True	True		
0x09852490	wcescomm.exe	4052	True	False	True	True	True	True		
0x09ec2d00	wmiprvse.exe	2984	True	False	True	True	True	True		
0x09f15810	wdfngr.exe	2316	True	False	True	True	True	True		
0x09f32960	sshhd.exe	2080	True	False	True	True	True	True		
0x09b49020	CodeMeterCC.exe	1372	True	False	True	True	True	True		
0x097a6c28	cmd.exe	2164	True	False	True	True	True	True		
0x09f2e518	oracle.exe	2096	True	False	True	True	True	True		
0x098185a8	rundll32.exe	3840	True	False	True	True	True	True		
0x0a926da0	vmacthlp.exe	1332	True	False	True	True	True	True		
0x0a86bb20	svchost.exe	1228	True	False	True	True	True	True		
0x0a785dd0	winlogon.exe	1056	True	False	True	True	True	True		
0x0938f370	GoogleUpdate.ex	1104	True	False	True	True	True	True		
0x09f30020	ctfmon.exe	3952	True	False	True	True	True	True		
0x0aad9600	svchost.exe	1628	True	False	True	True	True	True		
0x09f49da0	cyrunsrv.exe	1596	True	False	True	True	True	True		
0x0a607228	CodeMeter.exe	436	True	False	True	True	True	True		
0x09f32020	extjob.exe	2072	True	False	True	True	True	True		
0x0a509b30	vmtoolsd.exe	2476	True	False	True	True	True	True		
0x0a841a88	svchost.exe	1744	True	False	True	True	True	True		
0x0a640da0	svchost.exe	1432	True	False	True	True	True	True		
0x09f14020	svchost.exe	2240	True	False	True	True	True	True		
0x0a947b88	svchost.exe	396	True	False	True	True	True	True		
0x09e0ead0	svchost.exe	564	True	False	True	True	True	True		
0x09796020	MagicDisc.exe	2260	False	True	False	True	True	True		
0x09d30028	wuauctl.exe	2776	True	False	True	True	True	True		
0x097f7020	AdobeARM.exe	3804	True	False	True	True	True	True		
0x0a5fd0a0	lsass.exe	1164	True	False	True	True	True	True		
0x09e4dd020	BTTray.exe	2880	True	False	True	True	True	True		
0x09dd1d38	GoogleUpdate.ex	4028	True	False	True	True	True	True		
0x098121e8	jusched.exe	3880	True	False	True	True	True	True		
0x09c2ab20	alg.exe	3244	True	False	True	True	True	True		
0x0a562020	hasplms.exe	880	True	False	True	True	True	True		
0x0a726248	spoolsv.exe	2012	True	False	True	True	True	True		
0x0a777d00	jqs.exe	1380	True	False	True	True	True	True		
0x0981a020	TPAutoConnect.e	3752	True	False	True	True	True	True		
0x09849da0	explorer.exe	2808	True	False	True	True	True	True		
0x09e13da0	rapimgv.exe	1092	True	False	True	True	True	True		
0x0ab265d0	svchost.exe	312	True	False	True	True	True	True		
0x09c56bf8	TPAutoConnSvc.e	2276	True	False	True	True	True	True		
0x097e57b8	acrotray.exe	3780	True	False	True	True	True	True		
0x097b0c08	ONENOTE.MXE	1156	True	False	True	True	True	True		
0x09f1a58	acrobat_sl.exe	3744	True	False	True	True	True	True		
0x0a91c670	ADEngineS.exe	368	True	False	True	True	True	True		
0x0989b6f0	wscntfy.exe	240	True	False	True	True	True	True		
0x0a597d00	svchost.exe	1772	True	False	True	True	True	True		
0x0a8c93e0	prtk_supervisor	348	True	False	True	True	True	False		
0x0981ed10	GrooveMonitor.e	3264	True	False	True	True	True	True		
0x0a951b20	mysqld-nt.exe	1912	True	False	True	True	True	True		
0x0a93623d0	wuauctl.exe	2828	True	False	True	True	True	True		
0x0a72556d	svchost.exe	712	True	False	True	True	True	True		
0x0a5fb460	ADEngineW.exe	404	True	False	True	True	True	True		
0x09819020	ISUSPM.exe	3720	True	False	True	True	True	True		
0x094eb020	GoogleUpdate.ex	4064	True	False	True	True	True	True		
0x09368870	wmiprvse.exe	2392	True	False	True	True	True	True		
0x09f4c2e8	svchost.exe	2132	True	False	True	True	True	True		
0x09ea4020	btdwins.exe	192	True	False	True	True	True	True		
0x0a619578	cypnd.exe	568	True	False	True	True	True	True		
0x093bd878	wmiadap.exe	3732	True	False	True	True	True	True		
0x0a978b20	madm.exe	1884	True	False	True	True	True	True		
0x0a71fda0	svchost.exe	1348	True	False	True	True	True	True		
0x0a983680	smss.exe	688	True	False	True	False	False	False		
0x0ac39830	System	4	True	False	True	False	False	False		
0x09f3c550	cyrunsrv.exe	2056	True	False	True	False	False	False		
0x0a5ffda0	csrss.exe	1024	True	False	True	False	True	True		

Windows Process Tokens

- Process tokens store the SIDs (user and group contexts) and privileges for a process
- **getsids** and **privs** dump token information
- Helps detect privilege escalation attacks
- e.g., can see **Domain Admin** or **Enterprise Admin** in the SIDs list
- Examining privileges helps detect unusual behavior
 - Debug privilege → code injection is possible
 - Load driver privilege → obvious
 - Backup privilege → can read sensitive files

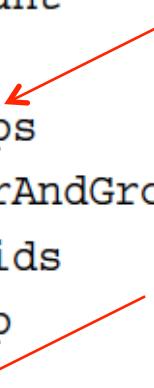
_TOKEN structure (2003 Server)

```
>>> dt("_TOKEN")
'_TOKEN' (208 bytes)

0x0      : TokenSource           ['_TOKEN_SOURCE']
0x10     : TokenId              ['_LUID']
0x18     : AuthenticationId    ['_LUID']
[snip]

0x4c     : UserAndGroupCount   ['unsigned long']
0x50     : RestrictedSidCount  ['unsigned long']
0x54     : PrivilegeCount      ['unsigned long']
[snip]

0x68     : UserAndGroups       ['pointer', ['array',
                                             lambda x: x.UserAndGroupCount, ['_SID_AND_ATTRIBUTES']]]
0x70     : RestrictedSids      ['pointer64', ['_SID_AND_ATTRIBUTES']]
0x78     : PrimaryGroup        ['pointer64', ['void']]
0x80     : Privileges           ['pointer', ['array',
                                             lambda x: x.PrivilegeCount, ['_LUID_AND_ATTRIBUTES']]]


```

_TOKEN structure (64-bit Win 7)

```
$ python vol.py --profile=Win7SP1x64 -f win7-64bit.vmem volshell
>>> dt("_TOKEN")
 '_TOKEN' (784 bytes)
0x0      : TokenSource                  ['_TOKEN_SOURCE']
0x10     : TokenId                     ['_LUID']
0x18     : AuthenticationId            ['_LUID']
0x20     : ParentTokenId               ['_LUID']
0x28     : ExpirationTime              ['_LARGE_INTEGER']
0x30     : TokenLock                   ['pointer64', ['_ERESOURCE']]
0x38     : ModifiedId                  ['_LUID']
0x40     : Privileges                ['_SEP_TOKEN_PRIVILEGES']
0x58     : AuditPolicy                 ['_SEP_AUDIT_POLICY']
0x74     : SessionId                   ['unsigned long']
0x78     : UserAndGroupCount           ['unsigned long']
0x7c     : RestrictedSidCount          ['unsigned long']
0x90     : UserAndGroups              ['pointer', ['array',
             <function <lambda> at 0x107b120c8>, ['_SID_AND_ATTRIBUTES']]]
0x98     : RestrictedSids            ['pointer64', ['_SID_AND_ATTRIBUTES']]
0xe0     : SidHash                  ['_SID_AND_ATTRIBUTES_HASH']
0x1f0    : RestrictedSidHash         ['_SID_AND_ATTRIBUTES_HASH']
```

Token Analysis

```
$ python vol.py -f grrcon.img getsids -p 1096
```

```
Volatility Foundation Volatility Framework 2.4
```

```
explorer.exe (1096): S-1-5-21-2682149276-1333600406-3352121115-500 (administrator)
explorer.exe (1096): S-1-5-21-2682149276-1333600406-3352121115-513 (Domain Users)
explorer.exe (1096): S-1-1-0 (Everyone)
explorer.exe (1096): S-1-5-32-545 (Users)
explorer.exe (1096): S-1-5-32-544 (Administrators)
explorer.exe (1096): S-1-5-4 (Interactive)
explorer.exe (1096): S-1-5-11 (Authenticated Users)
explorer.exe (1096): S-1-5-5-0-206541 (Logon Session)
explorer.exe (1096): S-1-2-0 (Local (Users with the ability to log in locally))
explorer.exe (1096): S-1-5-21-2682149276-1333600406-3352121115-519 (Enterprise Admins)
explorer.exe (1096): S-1-5-21-2682149276-1333600406-3352121115-1115
explorer.exe (1096): S-1-5-21-2682149276-1333600406-3352121115-518 (Schema Admins)
explorer.exe (1096): S-1-5-21-2682149276-1333600406-3352121115-512 (Domain Admins)
explorer.exe (1096): S-1-5-21-2682149276-1333600406-3352121115-520 (Group Policy Creator Owners)
```

User Account Privileges

via
secpol.msc

Local Security Policy

File Action View Help

Security Settings

- Account Policies
 - Password Policy
 - Account Lockout Policy
- Local Policies
 - Audit Policy
 - User Rights Assignment
 - Security Options
- Windows Firewall with Advanced Security
- Network List Manager Policies
- Public Key Policies
- Software Restriction Policies
- Application Control Policies
- IP Security Policies on Local Computer
- Advanced Audit Policy Configuration

Policy	Security Setting
Access Credential Manager as a trusted caller	Everyone,Administrators,Users,Backup Operators
Access this computer from the network	
Act as part of the operating system	LOCAL SERVICE,NETWORK SERVICE,Administrators
Add workstations to domain	Guest,Administrators,Users,Backup Operators
Adjust memory quotas for a process	Administrators,Remote Desktop Users
Allow log on locally	Administrators,Backup Operators
Allow log on through Remote Desktop Services	Everyone,LOCAL SERVICE,NETWORK SERVICE,Administrators,User...
Back up files and directories	Administrators,Backup Operators
Bypass traverse checking	Everyone,LOCAL SERVICE,NETWORK SERVICE,Administrators,User...
Change the system time	LOCAL SERVICE,Administrators
Change the time zone	LOCAL SERVICE,Administrators,Users
Create a pagefile	Administrators
Create a token object	
Create global objects	
Create permanent shared objects	
Create symbolic links	
Debug programs	
Deny access to this computer from the network	
Deny log on as a batch job	
Deny log on as a service	
Deny log on locally	
Deny log on through Remote Desktop Services	
Enable computer and user accounts to be trusted for delegation	
Force shutdown from a remote system	
Generate security audits	
Impersonate a client after authentication	
Increase a process working set	
Increase scheduling priority	
Load and unload device drivers	
Lock pages in memory	
Log on as a batch job	
Log on as a service	
Manage auditing and security log	
Modify an object label	
Modify firmware environment values	
Perform volume maintenance tasks	
Profile single process	
Profile system performance	
Remove computer from docking station	
Replace a process level token	
Restore files and directories	
Shut down the system	
Synchronize directory service data	
Take ownership of files or other objects	

Back up files and directories Properties

Local Security Setting Explain

Back up files and directories

This user right determines which users can bypass file and directory, registry, and other persistent object permissions for the purposes of backing up the system.

Specifically, this user right is similar to granting the following permissions to the user or group in question on all files and folders on the system:

- Traverse Folder/Execute File
- List Folder/Read Data
- Read Attributes
- Read Extended Attributes
- Read Permissions

Caution

Assigning this user right can be a security risk. Since there is no way to be sure that a user is backing up data, stealing data, or copying data to be distributed, only assign this user right to trusted users.

Default on workstations and servers: Administrators

For more information about security policy and related Windows features, see the Microsoft website.

OK Cancel Apply

Administrators,Backup Operators

Administrators,Users,Backup Operators

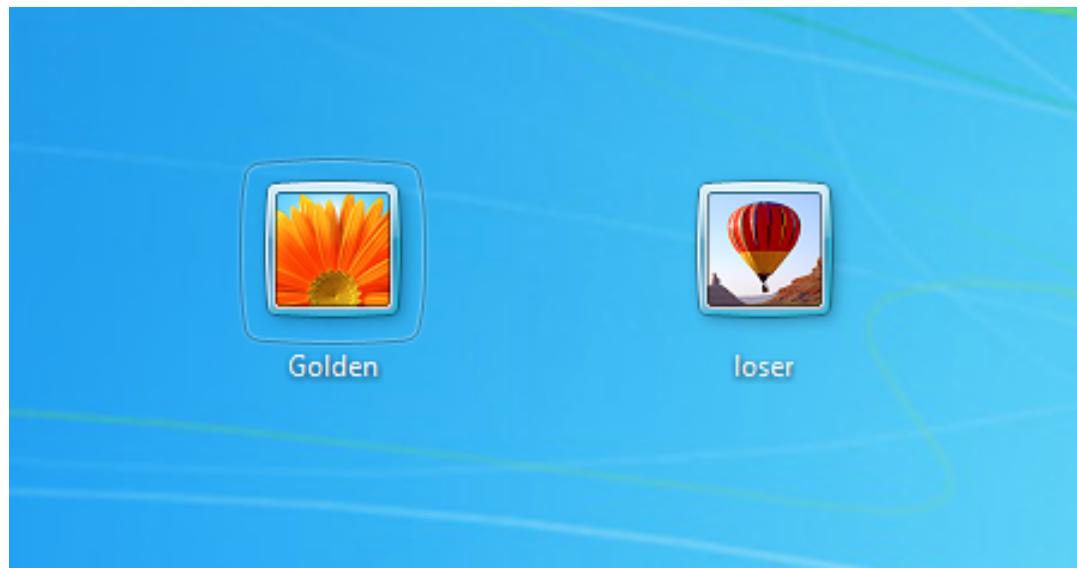
Administrators

Privilege Analysis To Detect Malware

```
$ python vol.py -f grrcon.img privs -p 1096
```

Volatility Foundation Volatility Framework 2.4

Pid	Process	Value	Privilege	Attributes
1096	explorer.exe	23	SeChangeNotifyPrivilege	Present,Enabled,Default
1096	explorer.exe	19	SeShutdownPrivilege	Present
1096	explorer.exe	25	SeUndockPrivilege	Present,Enabled
1096	explorer.exe	8	SeSecurityPrivilege	Present
1096	explorer.exe	17	SeBackupPrivilege	Present
1096	explorer.exe	18	SeRestorePrivilege	Present
1096	explorer.exe	12	SeSystemtimePrivilege	Present
1096	explorer.exe	24	SeRemoteShutdownPrivilege	Present
1096	explorer.exe	9	SeTakeOwnershipPrivilege	Present
1096	explorer.exe	20	SeDebugPrivilege	Present,Enabled
1096	explorer.exe	22	SeSystemEnvironmentPrivilege	Present
1096	explorer.exe	11	SeSystemProfilePrivilege	Present
1096	explorer.exe	13	SeProfileSingleProcessPrivilege	Present
1096	explorer.exe	14	SeIncreaseBasePriorityPrivilege	Present
1096	explorer.exe	10	SeLoadDriverPrivilege	Present,Enabled
1096	explorer.exe	15	SeCreatePagefilePrivilege	Present
1096	explorer.exe	5	SeIncreaseQuotaPrivilege	Present
1096	explorer.exe	28	SeManageVolumePrivilege	Present
1096	explorer.exe	30	SeCreateGlobalPrivilege	Present,Enabled,Default
1096	explorer.exe	29	SeImpersonatePrivilege	Present,Enabled,Default



Logged in as “golden”, administrative account

```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Golden>pslist | grep cmd

pslist v1.29 - Sysinternals PsList
Copyright (C) 2000-2009 Mark Russinovich
Sysinternals

cmd           1736   8   1   25    1976      0:00:00.015      0:00:15.184

C:\Users\Golden>
```

Pid	Process	Value	Privilege	Attributes	Description
1736	cmd.exe	2	SeCreateTokenPrivilege		Create a token object
1736	cmd.exe	3	SeAssignPrimaryTokenPrivilege		Replace a process-level token
1736	cmd.exe	4	SeLockMemoryPrivilege	Present	Lock pages in memory
1736	cmd.exe	5	SeIncreaseQuotaPrivilege	Present	Increase quotas
1736	cmd.exe	6	SeMachineAccountPrivilege		Add workstations to the domain
1736	cmd.exe	7	SeTcbPrivilege		Act as part of the operating system
1736	cmd.exe	8	SeSecurityPrivilege	Present	Manage auditing and security log
1736	cmd.exe	9	SeTakeOwnershipPrivilege	Present	Take ownership of files/objects
1736	cmd.exe	10	SeLoadDriverPrivilege	Present	Load and unload device drivers
1736	cmd.exe	11	SeSystemProfilePrivilege	Present	Profile system performance
1736	cmd.exe	12	SeSystemtimePrivilege	Present	Change the system time
1736	cmd.exe	13	SeProfileSingleProcessPrivilege	Present	Profile a single process
1736	cmd.exe	14	SeIncreaseBasePriorityPrivilege	Present	Increase scheduling priority
1736	cmd.exe	15	SeCreatePagefilePrivilege	Present	Create a pagefile
1736	cmd.exe	16	SeCreatePermanentPrivilege		Create permanent shared objects
1736	cmd.exe	17	SeBackupPrivilege	Present	Backup files and directories
1736	cmd.exe	18	SeRestorePrivilege	Present	Restore files and directories
1736	cmd.exe	19	SeShutdownPrivilege	Present	Shut down the system
1736	cmd.exe	20	SeDebugPrivilege	Present	Debug programs
1736	cmd.exe	21	SeAuditPrivilege		Generate security audits
1736	cmd.exe	22	SeSystemEnvironmentPrivilege	Present	Edit firmware environment values
1736	cmd.exe	23	SeChangeNotifyPrivilege	Present,Enabled,Default	Receive notifications of changes to files or directories
1736	cmd.exe	24	SeRemoteShutdownPrivilege	Present	Force shutdown from a remote system
1736	cmd.exe	25	SeUndockPrivilege	Present	Remove computer from docking station
1736	cmd.exe	26	SeSyncAgentPrivilege		Synch directory service data
1736	cmd.exe	27	SeEnableDelegationPrivilege		Enable user accounts to be trusted for delegation
1736	cmd.exe	28	SeManageVolumePrivilege	Present	Manage the files on a volume
1736	cmd.exe	29	SeImpersonatePrivilege	Present,Enabled,Default	Impersonate a client after authentication
1736	cmd.exe	30	SeCreateGlobalPrivilege	Present,Enabled,Default	Create global objects
1736	cmd.exe	31	SeTrustedCredManAccessPrivilege		Access Credential Manager as a trusted caller
1736	cmd.exe	32	SeRelabelPrivilege		Modify the mandatory integrity level of an object
1736	cmd.exe	33	SeIncreaseWorkingSetPrivilege	Present	Allocate more memory for user applications
1736	cmd.exe	34	SeTimeZonePrivilege	Present	Adjust the time zone of the computer's internal clock
1736	cmd.exe	35	SeCreateSymbolicLinkPrivilege	Present	Required to create a symbolic link

Logged in as “loser”, regular account

```
C:\Windows\system32\cmd.exe
C:\Users\loser>
C:\Users\loser>pslist | grep cmd
pslist v1.29 - Sysinternals PsList
Copyright (C) 2000-2009 Mark Russinovich
Sysinternals
cmd           3728   8   1   25   1960    0:00:00.000    0:00:10.159
C:\Users\loser>date
The current date is: Sat 09/06/2014
Enter the new date: (mm-dd-yy) 09/01/2014
A required privilege is not held by the client. ←
C:\Users\loser>_
```

```
C:\Windows\system32\cmd.exe
C:\Users\loser>
C:\Users\loser>pslist | grep cmd
pslist v1.29 - Sysinternals PsList
Copyright (C) 2000-2009 Mark Russinovich
Sysinternals

cmd           3728   8   1   25   1960       0:00:00.000       0:00:10.159

C:\Users\loser>date
The current date is: Sat 09/06/2014
Enter the new date: (mm-dd-yy) 09/01/2014
A required privilege is not held by the client.

C:\Users\loser>
```



Pid	Process	Value	Privilege	Attributes	Description
3728	cmd.exe	2	SeCreateTokenPrivilege		Create a token object
3728	cmd.exe	3	SeAssignPrimaryTokenPrivilege		Replace a process-level token
3728	cmd.exe	4	SeLockMemoryPrivilege		Lock pages in memory
3728	cmd.exe	5	SeIncreaseQuotaPrivilege		Increase quotas
3728	cmd.exe	6	SeMachineAccountPrivilege		Add workstations to the domain
3728	cmd.exe	7	SeTcbPrivilege		Act as part of the operating system
3728	cmd.exe	8	SeSecurityPrivilege		Manage auditing and security log
3728	cmd.exe	9	SeTakeOwnershipPrivilege		Take ownership of files/objects
3728	cmd.exe	10	SeLoadDriverPrivilege		Load and unload device drivers
3728	cmd.exe	11	SeSystemProfilePrivilege		Profile system performance
3728	cmd.exe	12	SeSystemtimePrivilege		Change the system time ←
3728	cmd.exe	13	SeProfileSingleProcessPrivilege		Profile a single process
3728	cmd.exe	14	SeIncreaseBasePriorityPrivilege		Increase scheduling priority
3728	cmd.exe	15	SeCreatePagefilePrivilege		Create a pagefile
3728	cmd.exe	16	SeCreatePermanentPrivilege		Create permanent shared objects
3728	cmd.exe	17	SeBackupPrivilege		Backup files and directories
3728	cmd.exe	18	SeRestorePrivilege		Restore files and directories
3728	cmd.exe	19	SeShutdownPrivilege	Present	Shut down the system
3728	cmd.exe	20	SeDebugPrivilege		Debug programs
3728	cmd.exe	21	SeAuditPrivilege		Generate security audits
3728	cmd.exe	22	SeSystemEnvironmentPrivilege		Edit firmware environment values
3728	cmd.exe	23	SeChangeNotifyPrivilege	Present,Enabled,Default	Receive notifications of changes to files or directories
3728	cmd.exe	24	SeRemoteShutdownPrivilege		Force shutdown from a remote system
3728	cmd.exe	25	SeUndockPrivilege	Present	Remove computer from docking station
3728	cmd.exe	26	SeSyncAgentPrivilege		Synch directory service data
3728	cmd.exe	27	SeEnableDelegationPrivilege		Enable user accounts to be trusted for delegation
3728	cmd.exe	28	SeManageVolumePrivilege		Manage the files on a volume
3728	cmd.exe	29	SeImpersonatePrivilege		Impersonate a client after authentication
3728	cmd.exe	30	SeCreateGlobalPrivilege		Create global objects
3728	cmd.exe	31	SeTrustedCredManAccessPrivilege		Access Credential Manager as a trusted caller
3728	cmd.exe	32	SeRelabelPrivilege		Modify the mandatory integrity level of an object
3728	cmd.exe	33	SeIncreaseWorkingSetPrivilege	Present	Allocate more memory for user applications
3728	cmd.exe	34	SeTimeZonePrivilege	Present	Adjust the time zone of the computer's internal clock
3728	cmd.exe	35	SeCreateSymbolicLinkPrivilege		Required to create a symbolic link

Simulating Privilege Escalation Using volshell

```
bigjoe:volatility golden$ python vol.py --profile=Win7SP1x64 -f ~/Documents/Virtual\ Machines.localized/Windows\ 7\ x64/Windows\ 7\ x64-1e433ca5.vmem volshell --write
Volatility Foundation Volatility Framework 2.4
Write support requested. Please type "Yes, I want to enable write support" below precisely (case-sensitive):
Yes, I want to enable write support
cc(Current context: System @ 0xfffffa804847c9e0, pid=4, ppid=0 DTB=0x187000
Welcome to volshell! Current memory image is:
file:///Users/golden/Documents/Virtual%20Machines.localized/Windows%207%20x64/Windows%207%20x64-1e433ca5.vmem
To get help, type 'hh()'
>>> cc(pid=3728)
Current context: cmd.exe @ 0xfffffa804a8d3060, pid=3728, ppid=3820 DTB=0x970fa000
>>> token=proc().get_token()
>>> bin(token.Privileges.Present)
'0b1100000001010001000000000000000000000000'
>>> bin(token.Privileges.Enabled)
'0b10000000000000000000000000000000'
>>> token.Privileges.Enabled=0xFFFFFFFFFFFFFFF ←
>>> token.Privileges.Present=0xFFFFFFFFFFFFFFF ←
>>> quit()
bigjoe:volatility golden$ []
```

This is for Win7 SP1—note that the trick outlined in AMF on p.174 that suggests that **Privileges.Present** need not be modified along with **Privileges.Enabled** doesn't work using this methodology.

(We're investigating.)

Logged in as “nobody”, regular account

```
C:\Windows\system32\cmd.exe
C:\Users\loser>
C:\Users\loser>pslist | grep cmd
pslist v1.29 - Sysinternals PsList
Copyright (C) 2000-2009 Mark Russinovich
Sysinternals

cmd           3728   8   1   25   1960    0:00:00.000    0:00:10.159

C:\Users\loser>date
The current date is: Sat 09/06/2014
Enter the new date: (mm-dd-yy) 09/01/2014
A required privilege is not held by the client. ←

C:\Users\loser>date
The current date is: Sat 09/06/2014
Enter the new date: (mm-dd-yy) 09/01/2014 ←

C:\Users\loser>_
```

END of Part I