Advancing Mac OS X Rootkit Detection

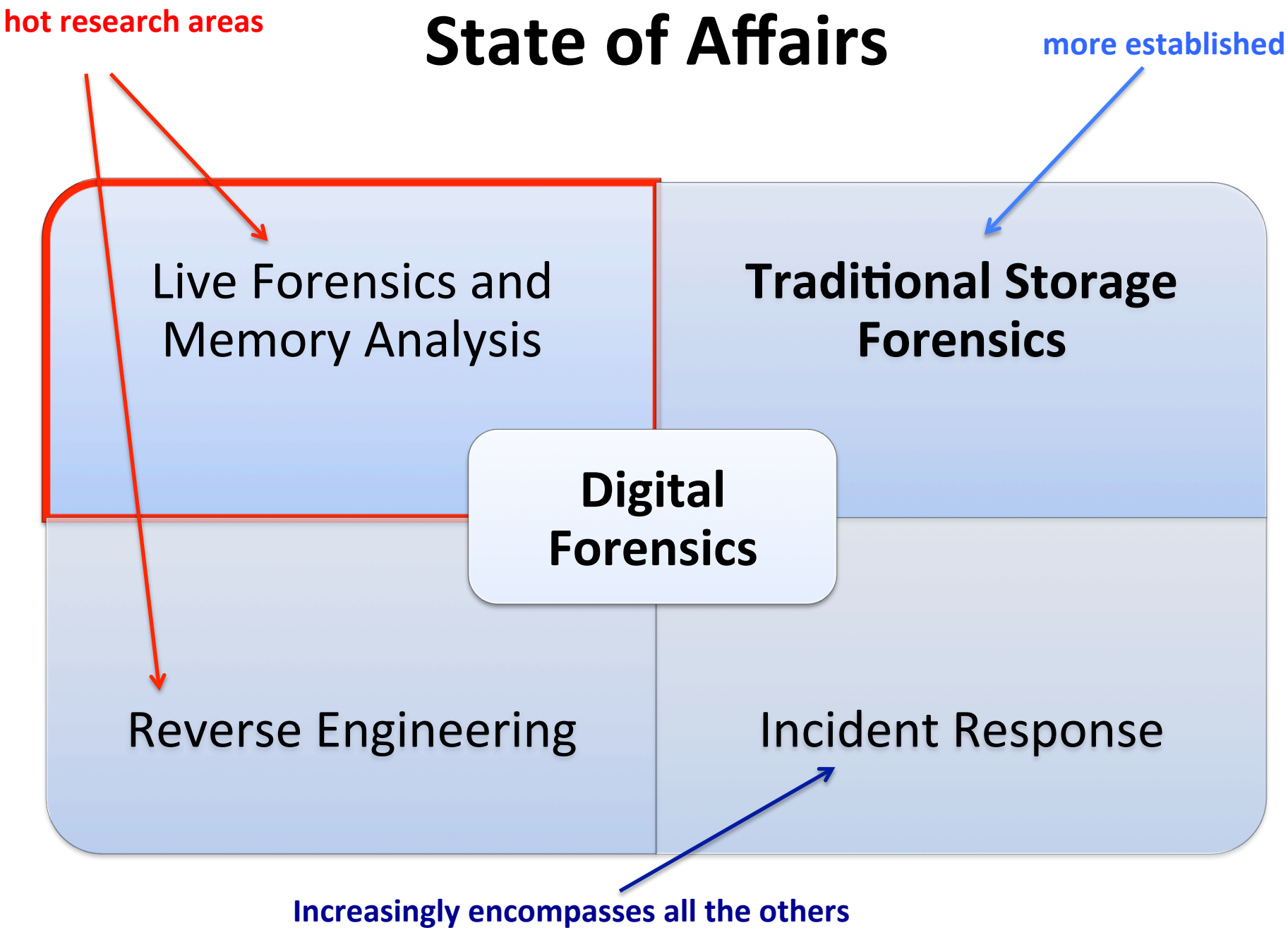
Andrew Case (@attrc)

Volatility Foundation

Golden G. Richard III (@nolaforensix)

University of New Orleans

State of Affairs



Where's the Evidence?

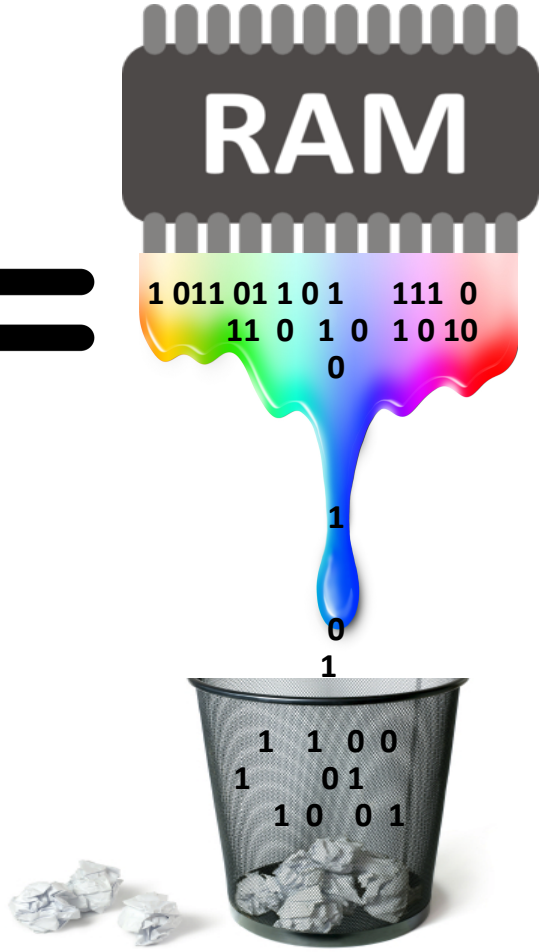
Files and Deleted Files	Filesystem metadata	Application metadata	Windows registry
Print spool files	Hibernation files	Temp files	Log files
Slack space	Swap files	Browser caches	Network traces

RAM: OS and app data structures

Volatile Evidence

Volatile Evidence

=



Awesomeness Progression: File Carving

Chaos: can't carve files → **Can carve files, but not very well** → **Faster** → **More accurate** → **Almost Hurray!**

Manual hex editor stuff

Tools appear, but have issues

Multithreading, better design

File type aware carving, et al

Fragmentation, damned spinning disks!

Images: https://easiersaidblogdotcom.files.wordpress.com/2013/02/hot_dogger.jpg
http://cdn.bigbangfish.com/555/Cow/Cow-6.jpg, http://f.tqn.com/y/bbq/1/W/U/i/Big_green_egg_large.jpg
http://i5.walmartimages.com/dfw/dce07b8c-bb22/k2-_95ea6c25-e9aa-418e-a3a2-8e48e62a9d2e.v1.jpg

Awesomeness Progression: Memory Forensics

Chaos: run strings?

Pioneering efforts show great promise

Beyond Windows

More, more, more

??

Manual, run strings, little context

`pt_finder` et al awesome but limited functionality

Mac, Linux, BSD

More attention to malware, filling in the gaps

...

Images: https://s-media-cache-ak0.pinimg.com/736x/75/5a/37/755a37727586c57a19d42caa650d242e.jpg,, http://img.photobucket.com/albums/v136/Hell2Pay77/SS-trucks.jpg http://skateandannoy.com/wp-content/uploads/2007/12/sportsbars.jpg, http://gainesvillescene.com/wp-content/uploads/2013/03/dog-longboard.jpg

Memory Analysis: 2004

$ grep -i murder /dev/mem

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

Murder

Murderer! Blood is on your shoulders!

Murderous

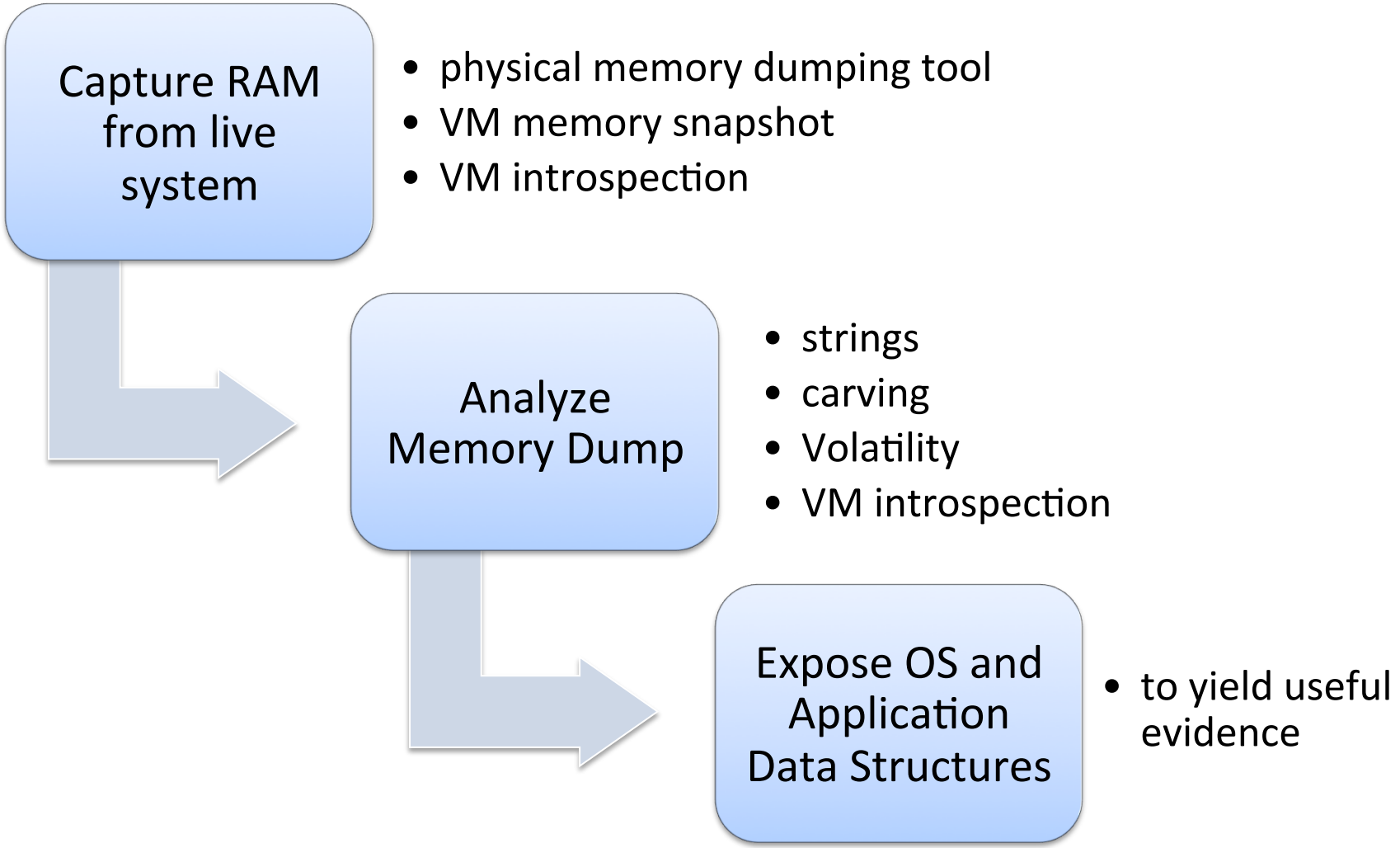
You murdered my hamster!

Murdered

Memory Analysis: Then

- `strings`
  - essentially no tools besides this, circa 2004
- `pt_finder` (~2006)
  - Windows process and thread enumeration
- FACE (~2008)
  - Memory analysis framework created at UNO
  - Correlates evidence in memory, network stack, network traces, filesystem
  - A bit closer to a framework rather than one-off tools
- …

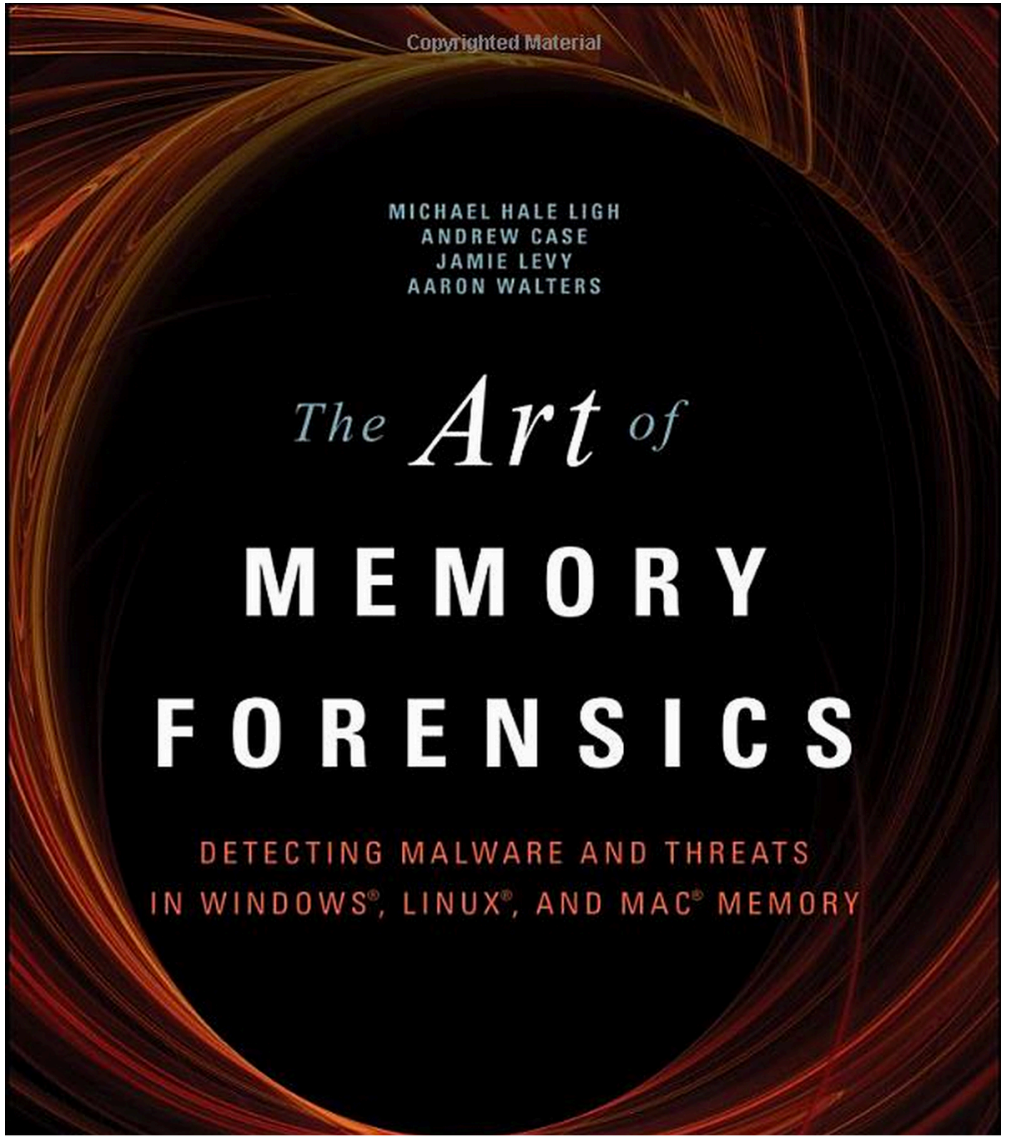
Memory Analysis: Now



Memory Analysis: Now

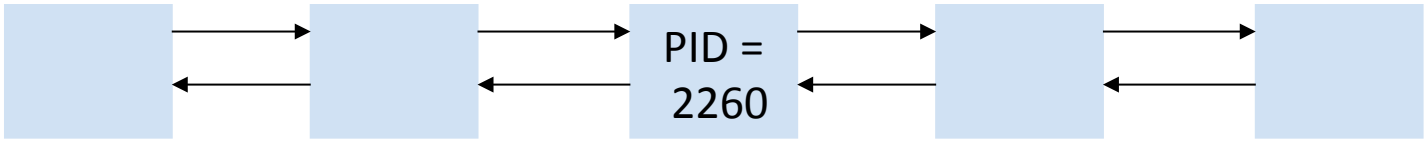
Use plugins to analyze:

- 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 new plugins



Detecting Hidden Resource Utilization

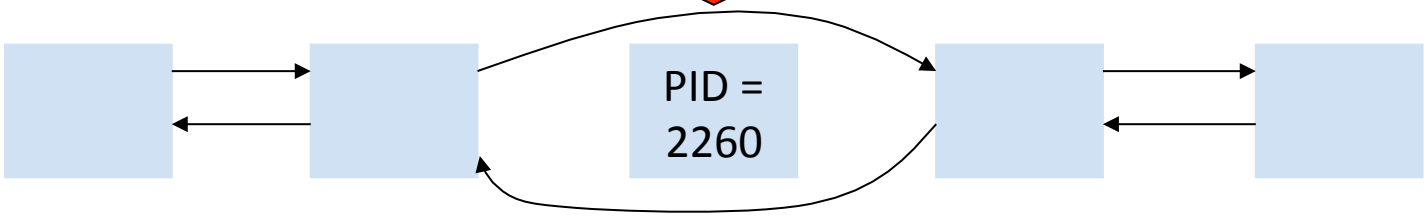
- Adversary: Direct Kernel Object Manipulation (DKOM)
- Strategy: Deep analysis and cross-correlation of data kernel data structures to reveal hidden resource utilization

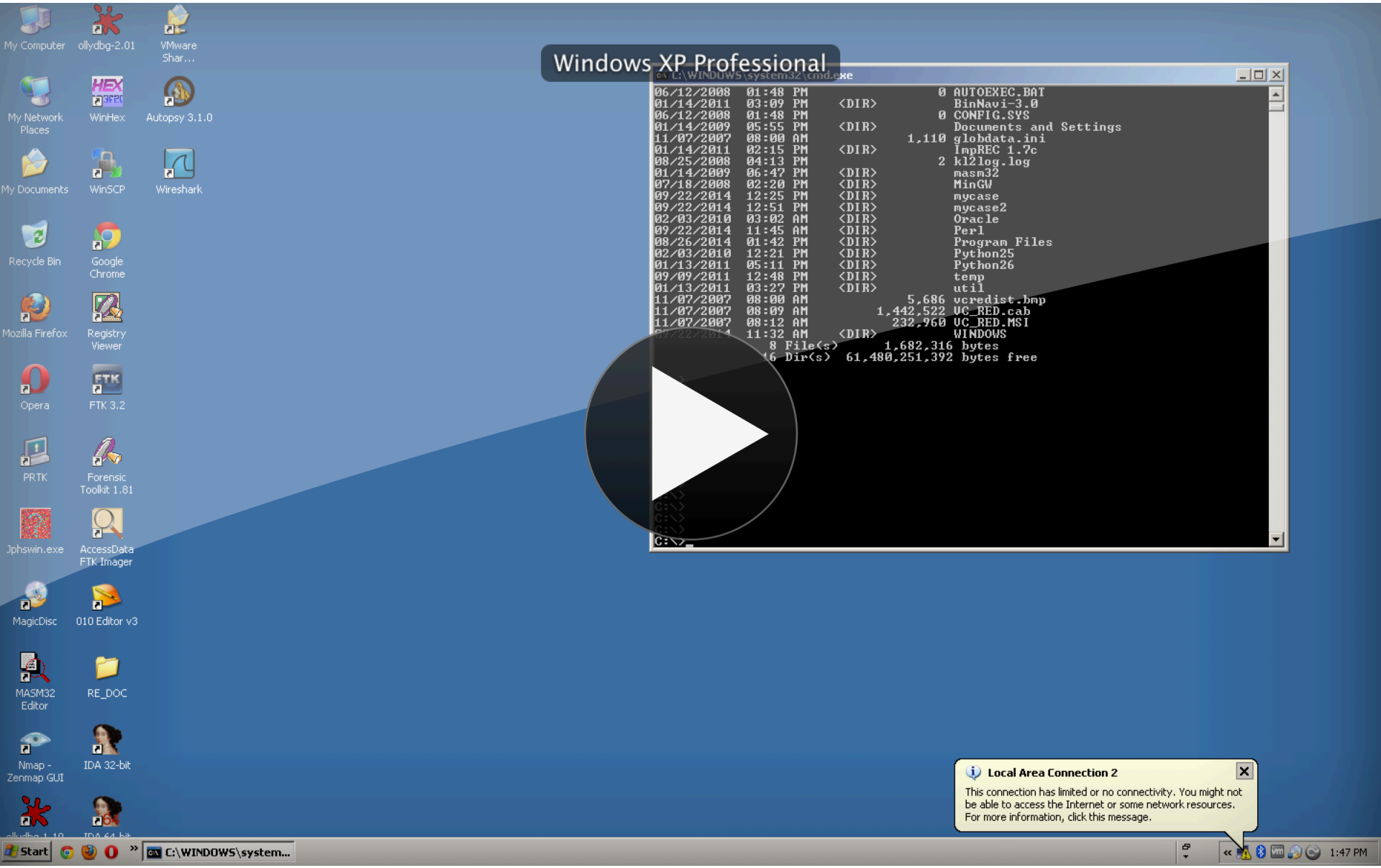


Doubly-linked process list in Windows kernel

C:\> fu –ph 2260

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





FU on PID 2260

Offset(P)	Name	PID	pslist	psscan	thrdproc	pspcid	csrss	session	deskthrd	ExitTime
0x0a720180	prtk_worker_ser	360	True	True	False	True	True	True	False	
0x097e0c68	vmtoolsd.exe	3932	True	True	False	True	True	True	True	
0x0a4e0da0	InstallShield L	1292	True	True	False	True	True	True	True	
0x0aad9888	services.exe	1152	True	True	False	True	True	True	True	
0x09852490	wcescomm.exe	4052	True	True	False	True	True	True	True	
0x09ec2da0	wmiprvse.exe	2984	True	True	False	True	True	True	True	
0x09f15810	wdfmgr.exe	2316	True	True	False	True	True	True	True	
0x09f32960	sshd.exe	2080	True	True	False	True	True	True	True	
0x09b49020	CodeMeterCC.exe	1372	True	True	False	True	True	True	True	
0x097a6c28	cmd.exe	2164	True	True	False	True	True	True	True	
0x09f2e518	oracle.exe	2096	True	True	False	True	True	True	True	
0x098185a8	rundll32.exe	3840	True	True	False	True	True	True	True	
0x0a926da0	vmacthlp.exe	1332	True	True	False	True	True	True	True	
0x0a86bb20	svchost.exe	1228	True	True	False	True	True	True	True	
0x0a785da0	winlogon.exe	1056	True	True	False	True	True	True	True	
0x0938f370	GoogleUpdate.ex	1104	True	True	False	True	True	True	True	
0x09f30020	ctfmon.exe	3952	True	True	False	True	True	True	True	
0x0aad9608	svchost.exe	1628	True	True	False	True	True	True	True	
0x09f49da0	cygrunsrv.exe	1596	True	True	False	True	True	True	True	
0x0a607228	CodeMeter.exe	436	True	True	False	True	True	True	True	
0x09f32020	extjob.exe	2072	True	True	False	True	True	True	True	
0x0a509b30	vmtoolsd.exe	2476	True	True	False	True	True	True	True	
0x0a841a88	svchost.exe	1744	True	True	False	True	True	True	True	
0x0a640da0	svchost.exe	1432	True	True	False	True	True	True	True	
0x09f14020	svchost.exe	2240	True	True	False	True	True	True	True	
0x0aa947b8	svchost.exe	396	True	True	False	True	True	True	True	
0x0a76eda0	svchost.exe	564	True	True	False	True	True	True	True	
0x09796020	MagicDisc.exe	2260	False	True	False	True	True	True	True	
0x09cd0928	wuauclt.exe	2776	True	True	False	True	True	True	True	
0x097f7020	AdobeARM.exe	3804	True	True	False	True	True	True	True	
0x0a5fdda0	lsass.exe	1164	True	True	False	True	True	True	True	
0x09e4d020	BTTray.exe	2880	True	True	False	True	True	True	True	
0x09dd1d38	GoogleUpdate.ex	4028	True	True	False	True	True	True	True	
0x098121e8	jusched.exe	3880	True	True	False	True	True	True	True	
0x09c2ab20	alg.exe	3244	True	True	False	True	True	True	True	
0x0a562020	hasplms.exe	880	True	True	False	True	True	True	True	
0x0a726248	spoolsv.exe	2012	True	True	False	True	True	True	True	
0x0a777da0	jqs.exe	1380	True	True	False	True	True	True	True	
0x0981a020	TPAutoConnect.e	3752	True	True	False	True	True	True	True	
0x09849da0	explorer.exe	2808	True	True	False	True	True	True	True	
0x09e13da0	rapimgr.exe	1092	True	True	False	True	True	True	True	
0x0ab265d0	svchost.exe	312	True	True	False	True	True	True	True	
0x09c56bf8	TPAutoConnSvc.e	2276	True	True	False	True	True	True	True	
0x097e57b8	acrotray.exe	3780	True	True	False	True	True	True	True	
0x097b0c08	ONENOTEM.EXE	1156	True	True	False	True	True	True	True	
0x097f1a58	acrobat_sl.exe	3744	True	True	False	True	True	True	True	
0x0a91c670	ADEngineS.exe	368	True	True	False	True	True	True	True	
0x0989b6f0	wscntfy.exe	240	True	True	False	True	True	True	True	
0x0a597da0	svchost.exe	1772	True	True	False	True	True	True	True	
0x0a8c93e0	prtk_supervisor	348	True	True	False	True	True	True	False	
0x0981ed10	GrooveMonitor.e	3264	True	True	False	True	True	True	True	
0x0a951b20	mysqld-nt.exe	1912	True	True	False	True	True	True	True	
0x093623a0	wuauclt.exe	2828	True	True	False	True	True	True	True	
0x0a725568	svchost.exe	712	True	True	False	True	True	True	True	
0x0a5fb460	ADEngineW.exe	404	True	True	False	True	True	True	True	
0x09819020	ISUSPM.exe	3720	True	True	False	True	True	True	True	
0x094eb020	GoogleUpdate.ex	4064	True	True	False	True	True	True	True	
0x09368870	wmiprvse.exe	2392	True	True	False	True	True	True	True	
0x09f4c2e8	svchost.exe	2132	True	True	False	True	True	True	True	
0x09ea4020	btwdins.exe	192	True	True	False	True	True	True	True	

DKOM Hidden Process Detection in Volatility

Windows, Mac, Linux

- Windows and Linux memory forensics techniques are fairly mature
- Lots of functionality
- More work to do, but good malware / rootkit detection
- > 115 Windows plugins for Volatility
- ~60 Mac plugins for Volatility
  - Some tackle the same thing on the BSD / Mach sides
- Mac OS X stuff lags behind and we're trying to fix that
- Requires OS internals work, which we both like

Mac OS X

- Many APIs similar to those on Windows and Linux are commonly abused by malware on other platforms
- Existing Mac plugins do not check these subsystems
- In addition, many Mac-specific features are vulnerable to abuse by rootkits / malware
- These aren't addressed by existing memory forensics tools, either

Facility: Kernel Event Callbacks

- **Use:** Allows registration of callbacks to be executed before specific events occur, e.g., process execution, system shutdown, hibernation, et al
- **Abuse:** Prevent security tools from loading, inject code into process before it can start, maintain persistence during reboot or shutdown
- Windows API: `PsSetCreateProcessNotifyRoutine`
- Windows detection plugin: `callbacks`

Mac OS X Power Related Events

- On Mac OS X, IOKit provides an API to register interest in power-related events
- Callback is triggered for notification
  - `kIOMessageSystemWillPowerOff`
  - `kIOMessageSystemWillRestart`
  - `kIOMessageSystemWillSleep`
  - `kIOMessageDeviceHasPoweredOn`
  - …
- Defined in `iokit/IOKit/IOMessage.h`

`mac_interest_handlers` Plugin

- Find root of IOKit device tree
- Walk tree and check for power-related interests, stored in `IORegistryEntry` structures
- Enumerate handlers in associated `IOCommand` structure and **verify that handlers are:**
  - **in code section of running kernel --or--**
  - **in address space of a loaded kernel extension**
- **Marked suspicious if not**
- **Importantly: location of handler also hints at malware location**
- **Can then target specific regions of memory and examine**

Facility: Driver ←→ Userland Communication

- **Use:** Provide an interface for userland processes to request services from a kernel driver (read, write, change configuration, etc.)
- **Abuse:** Provides a mechanism for a userland component to hide data, protect files, communicate with kernel-level malware
- Linux API: `devfs, ioctl`
- Linux detection plugin: `linux_check_fop`

Mac OS X Driver Communication

- Mac OS X provides several mechanisms for drivers to communicate with user space
  - IOKit APIs that allow userspace to search for interesting devices
  - *devfs*
- For this talk, look only at *devfs*
- Traditional filesystem interface
- Device registers handlers for `open()`, `close()`, `read()`, `write()`, `ioctl()`, etc.

Crisis: Notorious Mac OS X Malware

- Patches Activity Monitor to hide
- Takes screenshots
- Captures audio
- Captures video
- Connects to WiFi hotspots to transmit collected data
- Basically, all the stuff you're scared malware will do to you

Crisis

- Legitimate device: **`/dev/pmCPU`**
- "Evil" Crisis device: **`/dev/pfCPU`**
- Kernel components live behind this device
- Userspace components of Crisis use `ioctl()` calls to communicate with kernel components
- Lots of complex hiding mechanisms in Crisis
- Quickly locating handlers for kernel module can help direct static analysis

mac_devfs plugin

```
$ python vol.py --profile=MacLion_10_7_3_AMDx64 -f crisis-infected.snap mac_devfs
```

Offset	Path	Member	Handler	Module	Handler Sym
0xffffff8000859000	/dev/pfCPU	d_mmap	0xffffff800055e480	__kernel__	_enodev
0xffffff8000859000	/dev/pfCPU	d_ioctl	0xffffff7f808049c6	com.apple.mdworker	
0xffffff8000859000	/dev/pfCPU	d_strategy	0xffffff800055e490	__kernel__	_enodev_strat
0xffffff8000859000	/dev/pfCPU	d_select	0xffffff800055e480	__kernel__	_enodev
0xffffff8000859000	/dev/pfCPU	d_read	0xffffff800055e480	__kernel__	_enodev
0xffffff8000859000	/dev/pfCPU	d_write	0xffffff800055e480	__kernel__	_enodev
0xffffff8000859000	/dev/pfCPU	d_reserved_1	0xffffff800055e48	__kernel__	_enodev
0xffffff8000859000	/dev/pfCPU	d_open	0xffffff7f808049b6	com.apple.mdworker	
0xffffff8000859000	/dev/pfCPU	d_close	0xffffff7f808049be	com.apple.mdworker	
0xffffff8000859000	/dev/pfCPU	d_reset	0xffffff800055e480	__kernel__	_enodev
0xffffff8000859000	/dev/pfCPU	d_reserved_2	0xffffff800055e480	__kernel__	_enodev
0xffffff8000859000	/dev/pfCPU	d_stop	0xffffff800055e480	__kernel__	_enodev

"Good" mdworker is associated with Spotlight search indexing. This "evil" one can now be dumped and analyzed.

Facility: Kernel Timers

- **Use:** Register callback to be executed after a certain amount of time
- **Abuse:** Allow malicious code to run periodically without needing to hook functions, check persistence, check C&C server status, periodically exfiltrate data
- Windows API: `KeSetTimer`
- Windows detection plugin: `timers`

`mac_timers` Plugin

- Timer data is stored in per-CPU variables
- Plugin analyzes `cpu_data` structure for each processor
- `cpu_data->rtclock_timer` maintains queue of timers
- For each timer, output:
  - **Registering module**
  - Time to elapse
  - Address of each parameter
  - **Address of handler function**
  - **Suspicious?**

Facility: Userland Event Monitoring

- **Use:** Allows userland processes to monitor events on files, processes, signals, etc.
- **Abuse:** Stop processes from executing, determine when security tools are installed, detect when persistence mechanisms are removed
- Windows API: `FindFirstChangeNotification`, `RegNotifyChangeKeyValue`
  - `FILE_NOTIFY_CHANGE_FILE_NAME`
  - `FILE_NOTIFY_CHANGE_SIZE`
- Windows/Linux detection plugins: None (hint)

KQueue Monitoring

- Mac OS X facility that dates back to FreeBSD 4.1
- **Use:** Monitor file events, process creation, signals, etc.
- **Abuse:** Prevent processes from loading, react to process termination, etc.
- Interfaced from userland through *kqueue* and *kevent*
- `man kqueue / man kevent`

mac_kevents Plugin

- Enumerates all `kevent` structures…
- …(data structures kung fu in the paper)…
- …in the kernel and reports:
  - Which processes are being monitored (by PID) and which events are being monitored (`fork`, `exec`, `exit`, etc.)
  - Which file descriptor operations (`delete`, `write`, `rename`, etc.) are being monitored and by whom
  - …
- `kqueue/kevent` use is pervasive
- Burden is largely on the investigator to figure out what's cool and what's not
- If you have better ideas, share or hack!

mac_kevents

```
garfish:volatility golden$ python vol.py --profile MacMavericks_10_9_5_AMDx64 -f ~/Documents/Virtual\ Machines.localized/Mac\ OS\
 X\ 10.9.vmwarevm/Mac\ OS\ X\ 10.9-53f72337.vmem mac_kevents | grep -i iTunes
Volatility Foundation Volatility Framework 2.4
0xffffff803c2198f8 iTunesHelper             357      0 EVFILT_VM
0xffffff803cc97c88 iTunesHelper             357      1 EVFILT_USER
0xffffff803c21a6a0 iTunesHelper             357     14 EVFILT_PROC         NOTE_EXIT
0xffffff803cc97ac0 iTunesHelper             357   7683 EVFILT_MACHPORT
0xffffff803b91bf78 iTunesHelper             357  10499 EVFILT_MACHPORT
0xffffff803b91bdb0 iTunesHelper             357 18...2 EVFILT_TIMER        NOTE_NSECONDS, NOTE_ABSOLUTE
0xffffff803a33c018 iTunesHelper             357 18...0 EVFILT_TIMER        NOTE_NSECONDS, NOTE_ABSOLUTE
0xffffff8039e0cef0 iTunesHelper             357    357 EVFILT_PROC         NOTE_EXIT
0xffffff803b91c568 iTunes                   390     29 EVFILT_VNODE        NOTE_DELETE, NOTE_WRITE, NOTE_EXTEND, NOTE_RENAME
0xffffff803b91c140 iTunes                   390     30 EVFILT_VNODE        NOTE_RENAME
0xffffff803b91c270 iTunes                   390     31 EVFILT_VNODE        NOTE_RENAME
0xffffff803c218e48 iTunes                   390     32 EVFILT_VNODE        NOTE_RENAME
0xffffff8039fa7b50 iTunes                   390     33 EVFILT_VNODE        NOTE_RENAME
0xffffff803c46ad10 iTunes                   390     34 EVFILT_VNODE        NOTE_RENAME
0xffffff803b91c4d0 iTunes                   390     35 EVFILT_READ
0xffffff803a451440 iTunes                   390      0 EVFILT_VM
0xffffff803cc96f78 iTunes                   390      1 EVFILT_USER
0xffffff803b91c308 iTunes                   390     14 EVFILT_PROC         NOTE_EXIT
0xffffff803b91bb50 iTunes                   390   6403 EVFILT_MACHPORT
0xffffff8039bb98f8 iTunes                   390   7427 EVFILT_MACHPORT
0xffffff803c46d0b0 iTunes                   390 18...0 EVFILT_TIMER        NOTE_NSECONDS, NOTE_ABSOLUTE
0xffffff803bcbd900 iTunes                   390    390 EVFILT_PROC         NOTE_EXIT
0xffffff803a451b60 iTunes                   390   3083 EVFILT_MACHPORT
```

mac_lsof

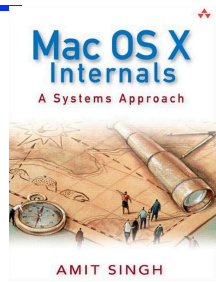
```
garfish:volatility golden$ python vol.py --profile MacMavericks_10_9_5_AMDx64 -f ~/Documents/Virtual\ Machines.localized/Mac\ OS\
 X\ 10.9.vmwarevm/Mac\ OS\ X\ 10.9-53f72337.vmem mac_lsof -p 390
Volatility Foundation Volatility Framework 2.4
PID      File Descriptor File Path
-------- --------------- ---------
     390               0 /Macintosh HD/dev/null
     390               1 /Macintosh HD/dev/null
     390               2 /Macintosh HD/dev/null
     390               4 /Macintosh HD/Applications/iTunes.app/Contents/Resources/iTunes.rsrc
     390               5 /Macintosh HD/System/Library/Frameworks/Carbon.framework/Versions/A/Frameworks/HIToolbox.framework/Versi
ons/A/Resources/HIToolbox.rsrc
     390               6 /Macintosh HD/System/Library/Frameworks/Carbon.framework/Versions/A/Frameworks/HIToolbox.framework/Versi
ons/A/Resources/English.lproj/Localized.rsrc
     390              10 /Macintosh HD/Users/nssal/Music/iTunes/iTunes Library Extras.itdb
     390              11 /Macintosh HD/Users/nssal/Music/iTunes/iTunes Library Genius.itdb
     390              12 /Macintosh HD/System/Library/Frameworks/Carbon.framework/Versions/A/Frameworks/HIToolbox.framework/Versi
ons/A/Resources/Extras2.rsrc
     390              13 /Macintosh HD/private/var/folders/m3/kpdlbsjn7k97rvhvd9b8vz6h0000gp/T/etilqs_XCRmSiLycspqVfb
     390              14 /Macintosh HD/dev/urandom
     390              15 /Macintosh HD/Applications/iTunes.app/Contents/Resources/Images.rsrc
     390              16 /Macintosh HD/Applications/iTunes.app/Contents/Resources/Images2.rsrc
     390              17 /Macintosh HD/private/var/folders/m3/kpdlbsjn7k97rvhvd9b8vz6h0000gp/T/etilqs_RNXE3ElXVfShauC
     390              18 /Macintosh HD/Applications/iTunes.app/Contents/Resources/DeviceImages.rsrc
     390              19 /Macintosh HD/Users/nssal/Library/Caches/com.apple.iTunes/Cache.db
     390              21 /Macintosh HD/Users/nssal/Library/Caches/com.apple.iTunes/Cache.db
     390              22 /Macintosh HD/Users/nssal/Library/Caches/com.apple.iTunes/Cache.db-wal
     390              23 /Macintosh HD/Users/nssal/Library/Caches/com.apple.iTunes/Cache.db-shm
     390              24 /Macintosh HD/Users/nssal/Library/Caches/com.apple.iTunes/Cache.db-wal
     390              29 /Macintosh HD/Users/nssal/Music/iTunes/iTunes Media/Automatically Add to iTunes.localized
     390              30 /Macintosh HD/Users/nssal/Music/iTunes/iTunes Media
     390              31 /Macintosh HD/Users/nssal/Music/iTunes
     390              32 /Macintosh HD/Users/nssal/Music
     390              33 /Macintosh HD/Users/nssal
     390              34 /Macintosh HD/Users
     390              40 /Macintosh HD/dev/random
```

Facility: File System Monitoring

- **Use:** Provide an interface for userland processes to request services from a kernel driver (e.g., read X bytes from file Y)
- **Abuse:** Provides a mechanism for a userland component to closely monitor filesystem operations, hide data, prevent modifications to data
- Windows API: `FindFirstChangeNotification`
- Windows/Linux detection plugins: None!

Mac Filesystem Hooking: Another Facility

- Userland tools can monitor file system activity by issuing `ioctl()` calls against */dev/fsevents*
- The registered callback will be executed upon file creation, deletion, renaming, and more
- See: http://osxbook.com/software/fslogger/
- Unlike many of the other facilities we've examined, this one is private and intended for use by Spotlight
- Limited number of "subscribers" is supported



```
$ python vol.py --profile=MacMavericks_10_9_5_AMDx64 -f fslogger.dump mac_vfsevents
```

Offset	Name	PID	Events
0xffffff80df4b6008	coreservicesd	36	DELETE, RENAME
0xffffff80df4e2008	fseventsd	39	CREATE_FILE, DELETE, STAT_CHANGED, RENAME, CONTENT_MODIFIED, EXCHANGE, FINDER_INFO_CHANGED, CREATE_DIR, CHOWN, XATTR_MODIFIED, XATTR_REMOVED
0xffffff80df69e008	mds	63	CREATE_FILE, DELETE, STAT_CHANGED, RENAME, CONTENT_MODIFIED, EXCHANGE, FINDER_INFO_CHANGED, CREATE_DIR, CHOWN, XATTR_MODIFIED, XATTR_REMOVED
0xffffff80dfc21008	fslogger	8495	CREATE_FILE, DELETE, STAT_CHANGED, RENAME, CONTENT_MODIFIED, EXCHANGE, FINDER_INFO_CHANGED, CREATE_DIR, CHOWN, XATTR_MODIFIED, XATTR_REMOVED

Final Thoughts

- We now have a solid foundation in memory forensics, with Volatility, Rekall, et al
- Focus on mining remaining sources of kernel / application data with a special emphasis on malware mitigation
- Work like this is largely "minding the gap" hacking, but it's still important
- Be old and have tenure (me) or be famous (Andrew) first?
- Your choice ☺

?

golden@cs.uno.edu / @nolaforensix
andrew@dfir.org / @attrc