Advancing Mac OS X Rootkit Detection

Andrew Case (@attrc)
Volatility Foundation

Golden G. Richard III (@nolaforensix)
University of New Orleans
State of Affairs

Live Forensics and Memory Analysis

Traditional Storage Forensics

Digital Forensics

Reverse Engineering

Incident Response

Increasingly encompasses all the others

Copyright 2015 by Andrew Case and Golden G. Richard III
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

Copyright 2015 by Andrew Case and Golden G. Richard III
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://easier saidblogdotcom.files.wordpress.com/2013/02/hot_dogger.jpg
http://i5.walmartimages.com/dfw/dce07b8c-bb22/k2_95ea6c25-e9aa-418e-a3a2-8e48e62a9d2e.v1.jpg

Copyright 2015 by Andrew Case and Golden G. Richard III
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

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

• ...

Copyright 2015 by Andrew Case and Golden G. Richard III
Memory Analysis: Now

- 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 Data Structures
  - to yield useful evidence

Copyright 2015 by Andrew Case and Golden G. Richard III
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

+ "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

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

C:\> fu –ph 2260

Copyright 2015 by Andrew Case and Golden G. Richard III
<table>
<thead>
<tr>
<th>Offset(?)</th>
<th>Name</th>
<th>PID</th>
<th>pslist</th>
<th>pscscope</th>
<th>thdproc</th>
<th>pspid</th>
<th>carss</th>
<th>session</th>
<th>deskthrd</th>
<th>ExitTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x08739000</td>
<td>pritk_worker.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08740000</td>
<td>vmtoolsd.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08750000</td>
<td>InstallShield L</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08760000</td>
<td>services.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08770000</td>
<td>wscom.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08780000</td>
<td>wsprys.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08790000</td>
<td>mdm.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x087a0000</td>
<td>inside.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x087b0000</td>
<td>CodeMeter.CC.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x087c0000</td>
<td>CodeMeter.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x087d0000</td>
<td>oracle.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x087e0000</td>
<td>rundll32.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x087f0000</td>
<td>vmmctblip.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08800000</td>
<td>svchost.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08810000</td>
<td>winlogon.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08820000</td>
<td>GoogleUpdate.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08830000</td>
<td>citmon.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08840000</td>
<td>True.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08850000</td>
<td>svchost.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08860000</td>
<td>cygtrunsrv.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08870000</td>
<td>CodeMeter.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08880000</td>
<td>extjob.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08890000</td>
<td>wscom.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x088a0000</td>
<td>svc.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x088b0000</td>
<td>svchost.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x088c0000</td>
<td>svchost.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x088d0000</td>
<td>svchost.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x088e0000</td>
<td>svchost.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x088f0000</td>
<td>svchost.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08900000</td>
<td>svchost.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x08910000</td>
<td>svchost.exe</td>
<td></td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

FU on PID 2260

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

Copyright 2015 by Andrew Case and Golden G. Richard III
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
"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!
<table>
<thead>
<tr>
<th>Address</th>
<th>Event Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xffffffff803c2198f8</td>
<td>EVFILT_VM</td>
<td>NOTE_EXIT</td>
</tr>
<tr>
<td>0xffffffff803cc97c88</td>
<td>EVFILT_USER</td>
<td>NOTE_EXIT</td>
</tr>
<tr>
<td>0xffffffff803c21a6a0</td>
<td>EVFILT_PROC</td>
<td>NOTE_NSECONDS, NOTE_ABSOLUTE</td>
</tr>
<tr>
<td>0xffffffff803cc97ac0</td>
<td>EVFILT_MACHPORT</td>
<td></td>
</tr>
<tr>
<td>0xffffffff803b91bf78</td>
<td>EVFILT_MACHPORT</td>
<td></td>
</tr>
<tr>
<td>0xffffffff803b91bdfb0</td>
<td>EVFILT_TIMER</td>
<td>NOTE_NSECONDS, NOTE_ABSOLUTE</td>
</tr>
<tr>
<td>0xffffffff803a33c018</td>
<td>EVFILT_TIMER</td>
<td>NOTE_NSECONDS, NOTE_ABSOLUTE</td>
</tr>
<tr>
<td>0xffffffff8039e0cef0</td>
<td>EVFILT_PROC</td>
<td>NOTE_EXIT</td>
</tr>
<tr>
<td>0xffffffff803b91c568</td>
<td>EVFILT_VNODE</td>
<td>NOTE_DELETE, NOTE_WRITE, NOTE_EXTEND, NOTE_RENAME</td>
</tr>
<tr>
<td>0xffffffff803b91c140</td>
<td>EVFILT_VNODE</td>
<td>NOTE_RENAME</td>
</tr>
<tr>
<td>0xffffffff803b91c270</td>
<td>EVFILT_VNODE</td>
<td>NOTE_RENAME</td>
</tr>
<tr>
<td>0xffffffff803c21e48</td>
<td>EVFILT_VNODE</td>
<td>NOTE_RENAME</td>
</tr>
<tr>
<td>0xffffffff8039fa7b50</td>
<td>EVFILT_VNODE</td>
<td>NOTE_RENAME</td>
</tr>
<tr>
<td>0xffffffff803c46ad10</td>
<td>EVFILT_VNODE</td>
<td>NOTE_RENAME</td>
</tr>
<tr>
<td>0xffffffff803b91c4d0</td>
<td>EVFILT_READ</td>
<td></td>
</tr>
<tr>
<td>0xffffffff803a451440</td>
<td>EVFILT_VM</td>
<td></td>
</tr>
<tr>
<td>0xffffffff803cc96f78</td>
<td>EVFILT_USER</td>
<td>NOTE_EXIT</td>
</tr>
<tr>
<td>0xffffffff803b91c308</td>
<td>EVFILT_PROC</td>
<td>NOTE_EXIT</td>
</tr>
<tr>
<td>0xffffffff803b91bb50</td>
<td>EVFILT_MACHPORT</td>
<td></td>
</tr>
<tr>
<td>0xffffffff8039bb98f8</td>
<td>EVFILT_MACHPORT</td>
<td></td>
</tr>
<tr>
<td>0xffffffff803c46d00</td>
<td>EVFILT_TIMER</td>
<td>NOTE_NSECONDS, NOTE_ABSOLUTE</td>
</tr>
<tr>
<td>0xffffffff803bc6d000</td>
<td>EVFILT_PROC</td>
<td>NOTE_EXIT</td>
</tr>
<tr>
<td>0xffffffff803a451b60</td>
<td>EVFILT_MACHPORT</td>
<td></td>
</tr>
</tbody>
</table>
```bash
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

<table>
<thead>
<tr>
<th>PID</th>
<th>File Descriptor</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>390</td>
<td>0</td>
<td>/Macintosh HD/dev/null</td>
</tr>
<tr>
<td>390</td>
<td>1</td>
<td>/Macintosh HD/dev/null</td>
</tr>
<tr>
<td>390</td>
<td>2</td>
<td>/Macintosh HD/dev/null</td>
</tr>
<tr>
<td>390</td>
<td>4</td>
<td>/Macintosh HD/Applications/iTunes.app/Contents/Resources/iTunes.rscc</td>
</tr>
<tr>
<td>390</td>
<td>5</td>
<td>/Macintosh HD/System/Library/Frameworks/Carbon.framework/Versions/A/Frameworks/HIToolbox.framework/Versions/A/Resources/HIToolbox.rscc</td>
</tr>
<tr>
<td>390</td>
<td>6</td>
<td>/Macintosh HD/System/Library/Frameworks/Carbon.framework/Versions/A/Frameworks/HIToolbox.framework/Versions/A/Resources/English.lproj/Localized.rscc</td>
</tr>
<tr>
<td>390</td>
<td>10</td>
<td>/Macintosh HD/Users/nssal/Music/iTunes/iTunes Library Extras.itdb</td>
</tr>
<tr>
<td>390</td>
<td>11</td>
<td>/Macintosh HD/Users/nssal/Music/iTunes/iTunes Library Genius.itdb</td>
</tr>
<tr>
<td>390</td>
<td>12</td>
<td>/Macintosh HD/System/Library/Frameworks/Carbon.framework/Versions/A/Frameworks/HIToolbox.framework/Versions/A/Resources/Extra2.rscc</td>
</tr>
<tr>
<td>390</td>
<td>13</td>
<td>/Macintosh HD/private/var/folders/m3/kpdlbsjn7k97rhv9bd9b8vz26h00000gp/T/etilqs_XCRMSilycspqVfb</td>
</tr>
<tr>
<td>390</td>
<td>14</td>
<td>/Macintosh HD/dev/urandom</td>
</tr>
<tr>
<td>390</td>
<td>15</td>
<td>/Macintosh HD/Applications/iTunes.app/Contents/Resources/Images.rscc</td>
</tr>
<tr>
<td>390</td>
<td>16</td>
<td>/Macintosh HD/Applications/iTunes.app/Contents/Resources/Images2.rscc</td>
</tr>
<tr>
<td>390</td>
<td>17</td>
<td>/Macintosh HD/private/var/folders/m3/kpdlbsjn7k97rhv9bd9b8vz26h00000gp/T/etilqs_RNXE3E12V5FShauC</td>
</tr>
<tr>
<td>390</td>
<td>18</td>
<td>/Macintosh HD/Applications/iTunes.app/Contents/Resources/DeviceImages.rscc</td>
</tr>
<tr>
<td>390</td>
<td>19</td>
<td>/Macintosh HD/Users/nssal/Library/Caches/com.apple.iTunes/Cache.db</td>
</tr>
<tr>
<td>390</td>
<td>20</td>
<td>/Macintosh HD/Users/nssal/Library/Caches/com.apple.iTunes/Cache.db</td>
</tr>
<tr>
<td>390</td>
<td>21</td>
<td>/Macintosh HD/Users/nssal/Library/Caches/com.apple.iTunes/Cache.db-wal</td>
</tr>
<tr>
<td>390</td>
<td>22</td>
<td>/Macintosh HD/Users/nssal/Library/Caches/com.apple.iTunes/Cache.db-shm</td>
</tr>
<tr>
<td>390</td>
<td>23</td>
<td>/Macintosh HD/Users/nssal/Library/Caches/com.apple.iTunes/Cache.db-wal</td>
</tr>
<tr>
<td>390</td>
<td>29</td>
<td>/Macintosh HD/Users/nssal/Music/iTunes/iTunes Media/Automatically Add to iTunes.localized</td>
</tr>
<tr>
<td>390</td>
<td>30</td>
<td>/Macintosh HD/Users/nssal/Music/iTunes/iTunes Media</td>
</tr>
<tr>
<td>390</td>
<td>31</td>
<td>/Macintosh HD/Users/nssal/Music/iTunes</td>
</tr>
<tr>
<td>390</td>
<td>32</td>
<td>/Macintosh HD/Users/nssal/Music</td>
</tr>
<tr>
<td>390</td>
<td>33</td>
<td>/Macintosh HD/Users/nssal</td>
</tr>
<tr>
<td>390</td>
<td>34</td>
<td>/Macintosh HD/Users</td>
</tr>
<tr>
<td>390</td>
<td>40</td>
<td>/Macintosh HD/dev/random</td>
</tr>
</tbody>
</table>
```
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/](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

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>PID</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xffffffffdfb6008</td>
<td>coreservicesd</td>
<td>36</td>
<td>DELETE, RENAME</td>
</tr>
<tr>
<td>0xffffffffdf4e2008</td>
<td>fseventsd</td>
<td>39</td>
<td>CREATE_FILE, DELETE, STAT_CHANGED, RENAME, CONTENT_MODIFIED, EXCHANGE, FINDER_INFO_CHANGED, CREATE_DIR, CHOWN, XATTR_MODIFIED, XATTR_REMOVED</td>
</tr>
<tr>
<td>0xffffffffdf69e008</td>
<td>mds</td>
<td>63</td>
<td>CREATE_FILE, DELETE, STAT_CHANGED, RENAME, CONTENT_MODIFIED, EXCHANGE, FINDER_INFO_CHANGED, CREATE_DIR, CHOWN, XATTR_MODIFIED, XATTR_REMOVED</td>
</tr>
<tr>
<td>0xffffffffdfc21008</td>
<td>fslogger</td>
<td>8495</td>
<td>CREATE_FILE, DELETE, STAT_CHANGED, RENAME,CONTENT_MODIFIED, EXCHANGE, FINDER_INFO_CHANGED, CREATE_DIR, CHOWN, XATTR_MODIFIED, XATTR_REMOVED</td>
</tr>
</tbody>
</table>
```
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