Registry Analysis and Memory Forensics: Together at Last

Brendan Dolan-Gavitt
Georgia Institute of Technology
Who I Am

- Developer on Volatility project
- Grad student and researcher at Georgia Tech
- Author of “Push the Red Button” blog
Overview
Overview

- Registry Analysis
Overview

- Registry Analysis
- Memory Forensics
Overview

- Registry Analysis
- Memory Forensics
- Combining the fields
Overview

- Registry Analysis
- Memory Forensics
- Combining the fields
- Lots of examples throughout
Windows Registry

• Centralized, hierarchical configuration database
• Structured like a filesystem
  • Keys = Directories, Values = Files
• Rich source of forensic information
Windows Registry (cont.)

- Registry appears as unified tree, but is made up of distinct *hives*
- Standard set of systemwide hives, as well as per-user hives
- Note: Registry contains some *volatile* data available only at runtime
Registry Data

- Removable media / USB keys
- Artifacts from P2P programs
- Malware persistence artifacts
- Password hashes (encrypted)
Registry Forensics

Tools

- RegLookup
  - Fast, robust registry parser
- Parse::Win32Registry
  - Lower level perl module
  - Good for rolling your own tools
- RegRipper
  - Undisputed king of registry analysis
  - Produces reports of forensic info
Memory Forensics

• Analysis of volatile data based on memory snapshots
• Allows extraction of live data, with
  • Higher integrity (smaller attack surface)
• Repeatable results
• Ability to ask new questions later
Memory Forensics Tools

- Memoryze
- HBGary Responder
- Volatility
  - GPL, collaborative development
  - Supports plugins
  - My favorite ;)

GPL, collaborative development
Data in Memory

- Current tools can extract a ton of information from memory images
- Processes
- Threads
- Drivers
- Open files
- Loaded DLLs
- Rootkit behavior
- Injected code
- Encryption keys
Combining Registry and Memory Analysis

• Windows keeps registry data in memory
• By figuring out the internals of the Windows Configuration Manager...
• We can combine these two fields!
Design Goals

- Access stable registry data
- Access volatile registry data
- Speed up incident response
- Apply existing registry analyses
Solution: VolReg

- Suite of plugins for Volatility 1.3
- Provides API to access registry data from XPSP2 memory images
- Adds new commands for:
  - Showing keys / values
  - Dumping registry as CSV
  - Extracting password hashes
VolReg: Hivescan

- Hivescan: finds raw offsets in memory image of registry hives
- Not very useful by itself, but needed for other plugins
- Once we have one hive offset, we can get a nicer list of all hives in memory

```
$ volatility hivescan \
   -f image.dd

<table>
<thead>
<tr>
<th>Offset</th>
<th>(hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>42168328</td>
<td>0x2837008</td>
</tr>
<tr>
<td>42195808</td>
<td>0x283db60</td>
</tr>
<tr>
<td>47598392</td>
<td>0x2d64b38</td>
</tr>
<tr>
<td>155764592</td>
<td>0x948c770</td>
</tr>
<tr>
<td>155973608</td>
<td>0x94bf7e8</td>
</tr>
<tr>
<td>208587616</td>
<td>0xc6ecb60</td>
</tr>
<tr>
<td>208964448</td>
<td>0xc748b60</td>
</tr>
<tr>
<td>234838880</td>
<td>0xdff5b60</td>
</tr>
<tr>
<td>243852936</td>
<td>0xe88e688</td>
</tr>
<tr>
<td>251418760</td>
<td>0xefc5888</td>
</tr>
<tr>
<td>252887048</td>
<td>0xf12c008</td>
</tr>
<tr>
<td>256039736</td>
<td>0xf42db38</td>
</tr>
<tr>
<td>269699936</td>
<td>0x10134b60</td>
</tr>
<tr>
<td>339523208</td>
<td>0x143cb688</td>
</tr>
<tr>
<td>346659680</td>
<td>0x14a99b60</td>
</tr>
<tr>
<td>377572192</td>
<td>0x16814b60</td>
</tr>
</tbody>
</table>
```
VolReg: Hivelist

• Given one of the offsets from hivescan, finds all other hives in memory
• Keep this list around!
• The addresses in this list are what we will use for the other registry plugins
HIVELIST EXAMPLE

$ volatility hivelist -f image.dd -o 0x2837008

Address      Name
0xe2610b60   \Documents and Settings\S----\Local Settings\[...]\UsrClass.dat
0xe25f0578   \Documents and Settings\S----\NTUSER.DAT
0xe1d33008   \Documents and Settings\LocalService\Local Settings\[...]\UsrClass.dat
0xe1c73888   \Documents and Settings\LocalService\NTUSER.DAT
0xe1c04688   \Documents and Settings\NetworkService\Local Settings\[...]\UsrClass.dat
0xe1b70b60   \Documents and Settings\NetworkService\NTUSER.DAT
0xe1658b60   \WINDOWS\system32\config\software
0xe1a5a7e8   \WINDOWS\system32\config\default
0xe165cb60   \WINDOWS\system32\config\SAM
0xe1a4f770   \WINDOWS\system32\config\SECURITY
0xe1559b38   [no name]
0xe1035b60   \WINDOWS\system32\config\system
0xe102e008   [no name]
**VolReg: Printkey**

- Given a hive address and a key, prints the key and its subkeys and values
- Shows last modified time, formats values according to type
- Includes *volatile* keys & values as well
- Good for just looking around
Printkey Example

$ volatility printkey -f image.dd -o 0xe1035b60
Key name: $$$PROTO.HIV (Stable)
Last updated: Mon Jul  4 18:16:59 2005

Subkeys:
  ControlSet001 (Stable)
  ControlSet002 (Stable)
  LastKnownGoodRecovery (Stable)
  MountedDevices (Stable)
  Select (Stable)
  Setup (Stable)
  WPA (Stable)
  CurrentControlSet (Volatile)

Values:

$ volatility printkey -f image.dd -o 0xe1035b60 CurrentControlSet
Key name: CurrentControlSet (Volatile)
Last updated: Mon Jul  4 18:16:59 2005

Subkeys:

Values:
REG_LINK  SymbolicLinkValue : \Registry\Machine\System\ControlSet001 (Volatile)
**VolReg: Hashdump**

- Local account password hashes are stored in the registry (encrypted)
- Hashdump module decrypts and prints these hashes
- If LanMan hash is in use, rainbow tables can recover password in minutes
- Use this power for good :)


Hashdump Example

System hive

SAM hive

$ volatility hashdump -f image.dd -y 0xe1035b60 -s 0xe165cb60

Administrator:500:08f3a52bdd35f179c81667e9d738c5d9:ed88ccc8bc08d1c18bcde3317112555f4::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0::
HelpAssistant:1000:ddd4c9c883a8ecb2078f88d729ba2e67:e78d693bc40f92a534197dc1d3a6d34f::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:8bfd47482583168a0ae5ab020e1186a9::
phoenix:1003:07b8418e83fad948aad3b435b51404ee:53905140b80b6d8cbe1ab5953f7c1c51::
ASPNET:1004:2b5f61b079400df84f9346ce3e830467:ae73a8bb65a0f01d9470fad55a411c::
S----:1006:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0::

Hashes can now be cracked using John the Ripper, rainbow tables, etc.
Others (not shown)

- **Cachedump**: dumps cached domain credentials
- **Lsadump**: dumps LSA protected storage (can contain passwords)
- **Hivedump**: dumps CSV of all keys and (optionally) values
Design Goals

• Access *stable* registry data [✓]
• Access *volatile* registry data [✓]
• Speed up incident response [?]
• Apply existing registry analyses [X]
Solution: VolRip

- Python → Perl interface that makes VolReg look like Parse::Win32Registry
- Now we can run RegRipper against memory!
- Existing analysis plugins just work
VolRip Setup

- Linux only (sorry!)
- Requirements: Inline::Python, VolReg
- Extract VolRip tarball into Volatility directory
- Run rip.pl with the memory image and address of the hive:

```bash
perl rip.pl -r image.dd@0xe1035b60 -f system
```
VolRip Example: Tracking USB Keys

$ perl rip.pl -r image.dd@0xe1035b60 -p usbstor
Launching usbstor v.20080418
USBStor
ControlSet001\Enum\USBStor

Disk&Ven_Generic&Prod_STORAGE_DEVICE&Rev_0.01 [Sun Jun 27 05:58:42 2004]
  S/N: 6&a344881&0 [Wed Jun 30 00:23:07 2004]
    FriendlyName  : Generic STORAGE DEVICE USB Device
    ParentIdPrefix: 7&192c45c3&0

  S/N: 0A4F1011180132130804&0 [Mon Jul  4 18:17:50 2005]
    FriendlyName  : LEXAR JUMPDRIVE SPORT USB Device
    ParentIdPrefix: 7&2930a404&0

$ perl rip.pl -r image.dd@0xe1035b60 -p usbstor2
SPLATITUDE,Disk&Ven_Generic&Prod_STORAGEDEVICE&Rev_0.01,6&a344881&0,
  1088554987,Generic STORAGE DEVICE USB Device,7&192c45c3&0,\DosDevices\E:
SPLATITUDE,Disk&Ven_LEXAR&Prod_JUMPDRIVE_SPORT&Rev_1000,0A4F1011180132130804&0,
  1120501070,LEXAR JUMPDRIVE SPORT USB Device,7&2930a404&0,\DosDevices\D:
**Design Goals**

- Access *stable* registry data [✓]
- Access *volatile* registry data [✓]
- Speed up incident response [✓]
- Apply existing registry analyses [✓]
Caveats

• Since analysis is done on memory, some things might be missing

• Can be especially annoying with hash dumping -- both System and SAM must be in memory

• Possible future direction: dump all available registry data, then use fragment recovery
Conclusions

- Applying registry analysis to memory can give you powerful new tools!
- Some existing registry tools can be coaxed into working with memory
- Still need tools to analyze volatile registry data (new RegRipper plugins?)
Thanks!

- To you all, for listening
- To AArorn Walters and the folks in #volatility
- To Harlan Carvey, Tim Morgan, and many others for their work and insight on all matters registry-related

...any questions?