What is Rekall?

- An advanced memory analysis solution.
  - Historically a fork of the Volatility memory analysis framework
    - Most code re-written/updated.
  - Fully open source and GPL - all commits are public.
  - Focus on:
    - code quality - public code reviews.
    - performance.
    - ease of use as a library - Integrated into other tools.
Rekall is integrated in GRR: Remote memory forensics at scale.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Name</th>
<th>type</th>
<th>size</th>
<th>stat.st_size</th>
<th>stat.st_mtime</th>
<th>stat.st_ctime</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>amosor-140691456.52</td>
<td>RekallResponseCollection</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>2014-08-19 23:00:16</td>
</tr>
<tr>
<td></td>
<td>scudette-140649156.21</td>
<td>RekallResponseCollection</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>2014-08-19 23:01:48</td>
</tr>
<tr>
<td></td>
<td>scudette-140649278.37</td>
<td>RekallResponseCollection</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>2014-08-25 13:06:07</td>
</tr>
<tr>
<td></td>
<td>scudette-1406979133.94</td>
<td>RekallResponseCollection</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>2014-09-11 12:26:47</td>
</tr>
</tbody>
</table>

### pslist

<table>
<thead>
<tr>
<th>PPID</th>
<th>Thds</th>
<th>Hnds</th>
<th>Sass</th>
<th>Wow64</th>
<th>Start</th>
<th>Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>71</td>
<td>510</td>
<td>None</td>
<td>False</td>
<td>2014-08-19 22:41:37+0000</td>
<td>-</td>
</tr>
<tr>
<td>460</td>
<td>3</td>
<td>34</td>
<td>0</td>
<td>False</td>
<td>2014-08-19 22:41:37+0000</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>26</td>
<td>None</td>
<td>False</td>
<td>2014-08-19 22:41:37+0000</td>
<td>-</td>
</tr>
<tr>
<td>340</td>
<td>9</td>
<td>415</td>
<td>0</td>
<td>False</td>
<td>2014-08-19 22:41:37+0000</td>
<td>-</td>
</tr>
<tr>
<td>340</td>
<td>3</td>
<td>75</td>
<td>0</td>
<td>False</td>
<td>2014-08-19 22:41:37+0000</td>
<td>-</td>
</tr>
<tr>
<td>378</td>
<td>7</td>
<td>276</td>
<td>1</td>
<td>False</td>
<td>2014-08-19 22:41:37+0000</td>
<td>-</td>
</tr>
<tr>
<td>378</td>
<td>3</td>
<td>132</td>
<td>1</td>
<td>False</td>
<td>2014-08-19 22:41:37+0000</td>
<td>-</td>
</tr>
<tr>
<td>378</td>
<td>3</td>
<td>132</td>
<td>1</td>
<td>False</td>
<td>2014-08-19 22:41:37+0000</td>
<td>-</td>
</tr>
<tr>
<td>384</td>
<td>8</td>
<td>180</td>
<td>1</td>
<td>False</td>
<td>2014-08-19 22:41:37+0000</td>
<td>-</td>
</tr>
<tr>
<td>384</td>
<td>7</td>
<td>544</td>
<td>0</td>
<td>False</td>
<td>2014-08-19 22:41:37+0000</td>
<td>-</td>
</tr>
<tr>
<td>384</td>
<td>10</td>
<td>145</td>
<td>0</td>
<td>False</td>
<td>2014-08-19 22:41:37+0000</td>
<td>-</td>
</tr>
<tr>
<td>384</td>
<td>11</td>
<td>355</td>
<td>0</td>
<td>False</td>
<td>2014-08-19 22:41:37+0000</td>
<td>-</td>
</tr>
<tr>
<td>460</td>
<td>12</td>
<td>118</td>
<td>0</td>
<td>False</td>
<td>2014-08-19 22:41:37+0000</td>
<td>-</td>
</tr>
<tr>
<td>460</td>
<td>19</td>
<td>257</td>
<td>1</td>
<td>False</td>
<td>2014-08-19 22:41:37+0000</td>
<td>-</td>
</tr>
</tbody>
</table>
How is it different from X?

- Other memory forensic frameworks rely on guessing global symbols through signature scanning.
- Rekall uses a different design philosophy:
  - Exact symbol information for the analyzed system
    - e.g. Fetch from Microsoft Symbol Server.
  - Store profiles in a public profile repository
    - Rekall fetches the required profile at runtime.
      - We have an index of kernel profiles.
    - We have over 200 different kernels in the public repository.
How is it different from X?

- This means we do not need to guess or try to deduce global symbols.
  - This makes Rekall much faster, more efficient and more accurate.
  - For example, Rekall does not use the Kernel Debugger Block
    - This can easily be overwritten by malware. Or newer versions of Windows.
  - This is similar to the way the kernel debugger works - much more reliable.
How is it different from X?

- Rekall distributes and supports a complete memory acquisition solution.
  - We have synergy between acquisition and analysis.
  - Support all major operating systems:
    - Windows - Winpmem tool.
    - Linux - pmem tool + LMAP tool (No need to precompile on target system).
    - OSX - OSXPmem tool (supports 10.9.4+).
  - Rekall acquisition tools allow for live system analysis (Triaging etc).
The Rekall User interfaces

- Rekall has 3 user interfaces:
  - Command line - single shot, run and exit.
  - Interactive console - IPython based (text only).
  - Webconsole - most powerful.
- The same plugin works in all environments!
  - Writing a plugin is easy
    - One does not need to think about output formatting - the framework does it all!
Text Interactive Console

- Fast and efficient
- Great for interactively exploring data types.
- Great for scripting complex analysis (no need to write plugin).
The Rekall Web Console interface

A GUI is not just a pretty thing!

- The Rekall Webconsole GUI helps drive analysis by:
  - Allowing the user to annotate her analysis
    - Notebook interface creates a mini "report" format.
    - Hides/Compacts long analysis to improve document flow.
  - Persistent file storage allows results to be managed and shared (based on Zip files).
  - Rekall files contain plugin output in JSON format
    - Machine readable - can be exported
# Rekall Memory Forensics

The web console is used to write notes about the analysis.

There is more:

- Markdown formatting:
  - Can provide links, lists, images etc.
- Persistent document file
  - All results from analysis are stored in the file.
  - You do not need the original image to view the data.
  - Can include arbitrary files inside the Rekall document.

---

<table>
<thead>
<tr>
<th>Name</th>
<th>PPid</th>
<th>Thds</th>
<th>Hnds</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>wininit.exe</td>
<td>340</td>
<td>3</td>
<td>75</td>
<td>Oct 1, 2012 11:39:57 PM</td>
</tr>
<tr>
<td>System</td>
<td>0</td>
<td>84</td>
<td>511</td>
<td>Oct 1, 2012 11:39:51 PM</td>
</tr>
<tr>
<td>srv.exe</td>
<td>2</td>
<td>29</td>
<td></td>
<td>Oct 1, 2012 11:39:51 PM</td>
</tr>
<tr>
<td>csrss.exe</td>
<td>8</td>
<td>192</td>
<td></td>
<td>Oct 1, 2012 11:39:57 PM</td>
</tr>
<tr>
<td>winlogon</td>
<td>4</td>
<td>4</td>
<td>125</td>
<td>Oct 1, 2012 11:39:58 PM</td>
</tr>
<tr>
<td>expl0.exe</td>
<td></td>
<td></td>
<td></td>
<td>Oct 1, 2012 11:40:04 PM</td>
</tr>
<tr>
<td>VBox.exe</td>
<td>8</td>
<td>8</td>
<td>80</td>
<td>Oct 1, 2012 11:40:05 PM</td>
</tr>
<tr>
<td>Cons.exe</td>
<td>4</td>
<td>188</td>
<td></td>
<td>Oct 1, 2012 11:40:18 PM</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>2616</td>
<td>2</td>
<td>66</td>
<td>Oct 1, 2012 11:40:20 PM</td>
</tr>
<tr>
<td>vol.exe</td>
<td>2644</td>
<td>1</td>
<td>19</td>
<td>Oct 1, 2012 11:41:03 PM</td>
</tr>
<tr>
<td>vol.exe</td>
<td>2912</td>
<td>4</td>
<td>169</td>
<td>Oct 1, 2012 11:41:03 PM</td>
</tr>
</tbody>
</table>

---

- HexDump
- Disassemble
- Struct
- Information
- Vad
- Process Info
Can Launch any Rekall plugins from UI.

- Markdown formatting,
  - Can provide links, lists, images etc.
- Persistent document file
  - All results from analysis are stored in the file.
    - You do not need the original image to view the data.
  - Can include arbitrary files inside the Rekall document.

Context aware plugin arguments allow customized UI.
Context sensitive Actions can analyze in a modal box - for quick drilling.

<table>
<thead>
<tr>
<th>Address</th>
<th>Rel Op Codes</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xf80002746d20</td>
<td>0 48894c2408</td>
<td>MOV [RSP+0x8],RCX</td>
<td></td>
</tr>
<tr>
<td>0xf80002746d25</td>
<td>5 53</td>
<td>PUSH RBX</td>
<td></td>
</tr>
</tbody>
</table>
Plugins that dump files can produce a zip file.

Dumped files are also persistent in the Rekall file.
Analysis files can be restored at will.

The UI works directly with the analysis file so no need to "save" the document.

It is possible to download a current snapshot of the document at any time (e.g. for backup).
The Rekall file is just a zip file which can be viewed/processed.

Each cell is a directory in the file.
What makes it work?

- The UI uses Rekall's data export facility.
  - Rekall exports **structured, semantically aware** data:
    
    ```
    $ rekal -v -f ~/images/win7.elf pslist -r data | json_pp
    "_EPROCESS" : {
      ...
      "Cybox" : {
        "Image_Info" : {
          "Path" : "C:\Windows\system32\csrss.exe",
          "File_Name" : "\\Device\\HarddiskVolume2\\Windows\\System32\csrss.exe",
          "type" : "ProcessObj:ImageInfoType",
          "Command_Line" : "%SystemRoot%\system32\csrss.exe ObjectDirectory=\\Windows SharedSection=1024,20480,768 Windows=On SubSystemType=Windows ServerDll=basesrv,1 ServerDll=winsrv: UserServerDllInitialization,3 ServerDll=winsrv:ConServerDllInitialization,2 ServerDll=sxssrv,4 ProfileControl=Off MaxRequestThreads=16"
        }
      }
    }
    ```

    Rekall uses Cybox "like" format to describe higher level objects in JSON.
Rekall Export System

- Rekall has a rich and highly customizable export system
  - Output format chosen by "Renderer"
    - Text renderer is default.
    - Data Export renderer produces rich JSON (used by the UI).
    - XLS renderer produces Excel sheets.
  - If we can make the GUI work with the exported data, any application can work with it!
    - This means you do not have to use Rekall as a library. Can be part of arbitrary pipeline.
http://www.rekall-forensic.com/

Sorry, Quaid. Your whole life is just a dream.

See you at the party, Richter!