2013 Open Source Digital Forensics Conference

Doing More with Less

Willi Ballenthin
Consultant
MANDIANT
Introduction: Willi Ballenthin

- Professionally
  - Incident response
  - Malware analysis
  - MANDIANT

- Personally:
  - Programming
  - Running
  - NYC, USA

- Contact:
  - @williballenthin
Effective enterprise-scale IR must be intelligence-driven

• You cannot keep up with an attacker if you deep-dive everything
  – Closing the door on first introductions probably won’t get you anywhere
  – They can tunnel faster than you can image

• Manual review does not scale across an enterprise
  – Codify knowledge
  – Distribute jobs and queries across the environment
  – Go hunting (with a plan)
Funnel the initial analysis results into the intel cycle

• Analysis of a box should yield actionable intelligence
  – It *may* help paint a picture
  – It *must* help you be in a better place tomorrow
    • Understand scope of compromise
    • Know how to find evil elsewhere
Funnel the initial analysis results into the intel cycle

- Questions to answer:
  - Network, host indicators?
    - Infection vector?
    - Compromised accounts?
    - Malware & persistence?
  - Timeframe?
  - Is more analysis needed?
Enterprise-grade Incident Response

Triage analysis should not take too long

• Prepare for dozens or hundreds of compromised boxes
  – Your time is valuable, and time is of the essence
  – Answer questions and move along

• Upshot: recognize patterns across hosts
  – Bad indicators are verbatim features of a binary (but they’re cheap & cost effective)
  – Great indicators describe patterns and generalize well
    • Only humans can do this

• (scripting can help)
Enterprise tools or preparation are required

• Have a remote forensic access route ready
  – Agent-based systems can be a great start
  – Custom solutions work too, just make sure they are be maintained

• Acquire data, analyze, and move along!
  – You don’t want to wait for images
  – Respond while attackers are still on the system
Hacking (in the RMS sense)

Let's brainstorm a balanced set of data to collect for triage

• Desirables:
  – High “bang for the buck”
  – Low resource requirements
    • Bandwidth
    • Storage
    • Worry
  – Pareto principle
Hacking (in the RMS sense)

Lets brainstorm a balanced set of data to collect for triage

<table>
<thead>
<tr>
<th>Class</th>
<th>Data</th>
<th>Resources Req’d</th>
<th>Completeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>File System</td>
<td>MFT</td>
<td>50 MB</td>
<td>OK</td>
</tr>
<tr>
<td>Configuration</td>
<td>Registry Hives</td>
<td>20 MB</td>
<td>Good</td>
</tr>
<tr>
<td>Logs</td>
<td>EVT/EVTX</td>
<td>50 MB?</td>
<td>OK</td>
</tr>
<tr>
<td>Memory</td>
<td>???</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
These formats compress very well

In compressed 75MB, you can capture the triage data for a typical workstation
What to do with an MFT file?

• AnalyzeMFT.py
• Sanderson Forensics MFTView (not OSS)
• TZWorks ntfswalk (not OSS)
• MFT tools descending from INDXParse
  – Pure Python
  – Backed by module MFT.py for easy reuse
  – https://github.com/williballenthin/INDXParse
list_mft.py

- MFT timelining utility
- Concise code: around 200 lines
- Supports $SI, multiple $FN, resident INDX records
- Constant memory usage

- How?
  - Each record has an $FN attribute with a name. The $FN attribute has a link to the parent record. Iterate the records and walk up the parent links.
list_mft.py
MFTView.py

- Unoriginal name, sorry.
- Pure Python
- Interactive inspection of a file system using only an MFT
- Some features
  - Integrated INDX record parsing
  - Strings, hex view
  - Data extraction and cluster run calculations
MFTView.py

Explore a NTFS file system from just the MFT
MFTView.py Metadata
MFTView.py Resident Data
NOTE: Check Disk Geometry

These byte offsets assume the following disk geometry. Please double check the geometry and update it here.

<table>
<thead>
<tr>
<th>Volume Offset (bytes)</th>
<th>32256</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster Size (bytes)</td>
<td>4096</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cluster Run</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset (clusters)</td>
</tr>
<tr>
<td>Length (clusters)</td>
</tr>
<tr>
<td>Offset (bytes)</td>
</tr>
<tr>
<td>Length (bytes)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cluster Run</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset (clusters)</td>
</tr>
<tr>
<td>Length (clusters)</td>
</tr>
<tr>
<td>Offset (bytes)</td>
</tr>
<tr>
<td>Length (bytes)</td>
</tr>
</tbody>
</table>
### MFTView.py INDX Records

#### Volume Offset (bytes) 32256  Cluster Size (bytes) 4096

<table>
<thead>
<tr>
<th>INDX Record Information</th>
<th>CWINDO~1.LNK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename</td>
<td>CWINDO~1.LNK</td>
</tr>
<tr>
<td>Size (bytes)</td>
<td>587</td>
</tr>
<tr>
<td>Created</td>
<td>2011-08-10T17:30:11.728249Z</td>
</tr>
<tr>
<td>Modified</td>
<td>2011-08-10T17:30:11.728249Z</td>
</tr>
<tr>
<td>Changed</td>
<td>2011-08-10T17:30:11.728249Z</td>
</tr>
<tr>
<td>Accessed</td>
<td>2011-08-10T17:30:11.728249Z</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INDX Record Information</th>
<th>GETTEX~1.LNK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename</td>
<td>GETTEX~1.LNK</td>
</tr>
<tr>
<td>Size (bytes)</td>
<td>693</td>
</tr>
<tr>
<td>Created</td>
<td>2011-08-11T15:23:26.862000Z</td>
</tr>
</tbody>
</table>
MFTView.py Strings
MFT Analysis

get_file_info.py

- Check the MFT structures for interesting files:
  - Timeline all embedded timestamps
  - Extract strings from MFT stack
  - Review (active & slack) INDX records
  - Pull out many other fields
Git/INDXParse - [master] » python get_file_info.py MFT.copy0 1
MFT Record: 123
Path: \system32\usmt

Metadata:
  Active: 1
  Type: directory
  Flags:
    $SI Modified: 2010-12-16 11:44:38.578125
    $SI Changed: 2010-12-16 11:44:38.578125
    $SI Birthed: 2010-12-16 11:41:33.343750
  Owner ID: 0
  Security ID: 278
  Quota charged: 0
  USN: 0

Filenames:
  Type: WIN32 + DOS 8.3
    Name: usmt
    Flags: has-index
    Logical size: 0
    Physical size: 0
    Modified: 2010-12-16 11:41:33.343750
    Accessed: 2010-12-16 11:41:33.343750
    Changed: 2010-12-16 11:41:33.343750
    Birthed: 2010-12-16 11:41:33.343750
    Parent reference: 29
    Parent sequence number: 1

Attributes:
  Type: $STANDARD INFORMATION
    Name: <none>
    Flags: has-index
    Resident: True
    Data size: 0
    Allocated size: 0
    Value size: 72
  Type: $FILENAME INFORMATION
    Name: <none>
    Flags: has-index
    Resident: True
    Data size: 0
    Allocated size: 0
    Value size: 74
  Type: $INDEX ROOT
    Name: $I30
    Flags: has-index
    Resident: True
    Data size: 0
    Allocated size: 0
    Value size: 56
  Type: $INDEX ALLOCATION

get_file_info.py

Page 1 of 2
Value size: 0
INDX root entries: <none>
INDX root slack entries: <none>
Timeline:
2010-12-16 11:41:33.343750  birthed  $SI  usmt
2010-12-16 11:41:33.343750  birthed  $FN  usmt
2010-12-16 11:41:33.343750  accessed $FN  usmt
2010-12-16 11:41:33.343750  modified $FN  usmt
2010-12-16 11:41:33.343750  changed  $FN  usmt
2010-12-16 11:44:38.578125  modified  $SI  usmt
2010-12-16 11:44:38.578125  changed  $SI  usmt

ASCII strings:
FILE0
<FH/0
<FH/0
<FH/0
<FH/0

Unicode strings:
USMT
$SI300
$SI30
$SI30
cobramsg.dll
guitrn.dll
guiltrna.dll
iconlib.dll
log.dll

get_file_info.py

Page 2 of 2
• Naturally: start with a timeline
• Manually review entries with MFTView
• Always rinse and repeat
What to do with a Registry hive?

• RegRipper
• MiTeC Windows Registry Analyzer (WRA, not OSS)
• Registry Decoder
• Registry tools included with python-registry
  – Pure Python
  – Backed by stable API in python-registry
  – https://github.com/williballenthin/python-registry
regview.py

Kinda like regedit.exe. Except read-only. And cross platform.
findkey.py

Search keys, values, paths with various queries.
### timeline.py

Timeline Registry key modification timestamps

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Event</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sat Aug 10</td>
<td>22:03:06</td>
<td>Registry</td>
<td>CreateHive(2A7FB991-7BBE-4F9D-B91E-7CB5)</td>
</tr>
<tr>
<td>Sat Aug 10</td>
<td>22:03:26</td>
<td>Registry</td>
<td>CreateHive(2A7FB991-7BBE-4F9D-B91E-7CB5)</td>
</tr>
<tr>
<td>Sat Aug 10</td>
<td>22:03:27</td>
<td>Registry</td>
<td>CreateHive(2A7FB991-7BBE-4F9D-B91E-7CB5)</td>
</tr>
<tr>
<td>Sat Aug 10</td>
<td>22:03:30</td>
<td>Registry</td>
<td>CreateHive(2A7FB991-7BBE-4F9D-B91E-7CB5)</td>
</tr>
<tr>
<td>Sat Aug 10</td>
<td>22:03:38</td>
<td>Registry</td>
<td>CreateHive(2A7FB991-7BBE-4F9D-B91E-7CB5)</td>
</tr>
<tr>
<td>Sat Aug 10</td>
<td>22:06:02</td>
<td>Registry</td>
<td>CreateHive(2A7FB991-7BBE-4F9D-B91E-7CB5)</td>
</tr>
<tr>
<td>Sat Aug 10</td>
<td>22:06:13</td>
<td>Registry</td>
<td>CreateHive(2A7FB991-7BBE-4F9D-B91E-7CB5)</td>
</tr>
</tbody>
</table>
• *Windows Registry Analysis* by H. Carvey
What to do with Event Log files?

- Parse-EVTX
- GrokEVT
- libevtx
- Carving library used by LfLe.py
  - Carve for Records (not files) – Jeff Hamm @ SANS DFIR Summit 2012
  - Pure Python module Evt.py
  - https://github.com/williballenthin/LfLe
- Parsing library python-evtx
  - Even more Python
  - https://github.com/williballenthin/python-evtx
Sneak peek: Event Log Viewer

• Open source GUI for reviewing event log files
• 
• Indexed search across events
• Multiple views of same data (searching, filtering, sorting)
• Quite slow.
• Lets collaborate!
### Event Viewer: Overview

<table>
<thead>
<tr>
<th>#</th>
<th>EID</th>
<th>Timestamp</th>
<th>Category</th>
<th>Subcategory</th>
<th>Message Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>91740</td>
<td>4672</td>
<td>2013-09-24T07:02:38....</td>
<td>Privilege Use</td>
<td>Sensitive Privilege Use / Non ...</td>
<td>Special privileges assigned to new logon.</td>
</tr>
<tr>
<td>91741</td>
<td>4688</td>
<td>2013-09-24T07:02:38....</td>
<td>Detailed Tracking</td>
<td>Process Creation</td>
<td>A new process has been created.</td>
</tr>
<tr>
<td>91742</td>
<td>4688</td>
<td>2013-09-24T07:02:39....</td>
<td>Detailed Tracking</td>
<td>Process Creation</td>
<td>A new process has been created.</td>
</tr>
<tr>
<td>91743</td>
<td>4624</td>
<td>2013-09-24T07:03:02....</td>
<td>Logon/Logoff</td>
<td>Logon</td>
<td>An account was successfully logged on.</td>
</tr>
<tr>
<td>91744</td>
<td>4672</td>
<td>2013-09-24T07:03:02....</td>
<td>Privilege Use</td>
<td>Sensitive Privilege Use / Non ...</td>
<td>Special privileges assigned to new logon.</td>
</tr>
<tr>
<td>91745</td>
<td>4688</td>
<td>2013-09-24T07:03:07....</td>
<td>Detailed Tracking</td>
<td>Process Creation</td>
<td>A new process has been created.</td>
</tr>
<tr>
<td>91746</td>
<td>4624</td>
<td>2013-09-24T07:03:10....</td>
<td>Logon/Logoff</td>
<td>Logon</td>
<td>An account was successfully logged on.</td>
</tr>
<tr>
<td>91747</td>
<td>4672</td>
<td>2013-09-24T07:03:10....</td>
<td>Privilege Use</td>
<td>Sensitive Privilege Use / Non ...</td>
<td>Special privileges assigned to new logon.</td>
</tr>
<tr>
<td>91748</td>
<td>4688</td>
<td>2013-09-24T07:03:10....</td>
<td>Detailed Tracking</td>
<td>Process Creation</td>
<td>A new process has been created.</td>
</tr>
<tr>
<td>91749</td>
<td>4688</td>
<td>2013-09-24T07:03:10....</td>
<td>Detailed Tracking</td>
<td>Process Creation</td>
<td>A new process has been created.</td>
</tr>
<tr>
<td>91750</td>
<td>4624</td>
<td>2013-09-24T07:03:10....</td>
<td>Logon/Logoff</td>
<td>Logon</td>
<td>An account was successfully logged on.</td>
</tr>
<tr>
<td>91751</td>
<td>4672</td>
<td>2013-09-24T07:03:10....</td>
<td>Privilege Use</td>
<td>Sensitive Privilege Use / Non ...</td>
<td>Special privileges assigned to new logon.</td>
</tr>
<tr>
<td>91752</td>
<td>4688</td>
<td>2013-09-24T07:03:10....</td>
<td>Detailed Tracking</td>
<td>Process Creation</td>
<td>A new process has been created.</td>
</tr>
<tr>
<td>91753</td>
<td>4688</td>
<td>2013-09-24T07:03:11....</td>
<td>Detailed Tracking</td>
<td>Process Creation</td>
<td>A new process has been created.</td>
</tr>
<tr>
<td>91754</td>
<td>4688</td>
<td>2013-09-24T07:03:17....</td>
<td>Detailed Tracking</td>
<td>Process Creation</td>
<td>A new process has been created.</td>
</tr>
<tr>
<td>91755</td>
<td>4689</td>
<td>2013-09-24T07:03:20....</td>
<td>Detailed Tracking</td>
<td>Process Termination</td>
<td>A process has exited.</td>
</tr>
<tr>
<td>91756</td>
<td>4624</td>
<td>2013-09-24T07:03:26....</td>
<td>Logon/Logoff</td>
<td>Logon</td>
<td>An account was successfully logged on.</td>
</tr>
<tr>
<td>91757</td>
<td>4672</td>
<td>2013-09-24T07:03:26....</td>
<td>Privilege Use</td>
<td>Sensitive Privilege Use / Non ...</td>
<td>Special privileges assigned to new logon.</td>
</tr>
<tr>
<td>91758</td>
<td>4688</td>
<td>2013-09-24T07:03:26....</td>
<td>Detailed Tracking</td>
<td>Process Creation</td>
<td>A new process has been created.</td>
</tr>
<tr>
<td>91759</td>
<td>4688</td>
<td>2013-09-24T07:03:29....</td>
<td>Detailed Tracking</td>
<td>Process Creation</td>
<td>A new process has been created.</td>
</tr>
</tbody>
</table>
Event Viewer: Details
Event Viewer: Structure
Event Viewer: Binary
Recap

In the last 30 mins, we covered:

• GUI MFT browser
• CLI MFT timeliner and inspector
• GUI Registry explorer
• CLI Registry timeliner & utils
• GUI EVT(X) tool

...and these work great for rapid triage!
Questions?