Using Image Excerpts to Jumpstart Windows Forensic Analysis

There are many options available for acquiring, processing and analyzing forensic disk images. Choices range from feature-rich commercial tools that provide all-in-one solutions, to open source scripts for carrying out specific tasks. The availability of these tools and the hard work of those who contribute to the forensic community have made the job of the examiner much easier. Even with recent advances, analysis can still be time-consuming, particularly in the acquisition and processing of Windows full disk images...

Copyright SANS Institute
Author Retains Full Rights
Using Image Excerpts to Jumpstart Windows Forensic Analysis

GIAC (GCFA) Gold Certification

Author: John Brown – johns_brown@yahoo.com
Advisor: Chris Walker
Accepted: June 15, 2018

Abstract

There are many options available for acquiring, processing and analyzing forensic disk images. Choices range from feature-rich commercial tools that provide all-in-one solutions, to open source scripts for carrying out specific tasks. The availability of these tools and the hard work of those who contribute to the forensic community have made the job of the examiner much easier. Even with recent advances, analysis can still be time-consuming, particularly in the acquisition and processing of Windows full disk images. One alternative is to extract and analyze the files historically known to contain the most relevant data first. In many cases, a relatively small number of files contain the majority of information needed to perform a forensic examination. Tests were performed on Windows images to analyze some of these high-value artifacts to find an efficient approach for selectively acquiring and extracting different types of metadata. A script was then written to automate repetitive steps and leverage open source tools found on most recent Linux version of the SANS Sift virtual machine.
1. Introduction

Much like live acquisition programs\(^1\) and memory analysis tools used in incident response and threat hunting, the aim of disk image processing should be to efficiently provide an assessment or report based on some event. Obvious obstacles are the ever-increasing size of hard disks and the time needed to inspect every sector of a disk image for relevant information. Unfortunately, there is no way to quickly obtain results doing this type of data carving. The most obvious time saving strategy is to off load these drawn out tasks to another system which depending on available resource may require an additional copy of the evidence image. If carving is delayed, the processing and much of the analysis can occur well before data carving is complete. Further efficiency can be achieved by extracting an image excerpt, which consists of easily attainable files that are of high forensic interest. Once these files are collected the disk image is can temporarily be set aside or dedicated to deep inspection and the excerpted files can be immediate focus of the examination.

So, what exactly is an “image excerpt”? Where excerpt is defined as an incomplete extract of data from a media source, an image excerpt described here is a partial export of imaged forensic evidence or a collection of files. Because data is copied from an image, it lacks integrity but can still be used to quickly find answers that can later be verified using established methods and techniques. The acquisition of individual files of interest is common in threat hunting and incident response where data of interest is acquired over the network and then “stacked” to identify abnormalities. Similarly, Chris Pogue has written and delivered a number of presentations on the subject of “Sniper Forensics” that demonstrate how to perform a focused response based on a very small number of artifacts. The creation of disk image excerpts and the automation contained in the script is an illustration of these techniques.

Image excerpts are not intended to a replacement for tools like Plaso, but as a possible way to fast forward the analysis process. Once data is extracted, the reduced file set can be triaged and excerpts further examined as source data. Tools like Internet Evidence Finder (IEF) and tools like those created by Eric Zimmerman\(^2\) can be used to selectively examine relevant artifacts that may in some cases reduce the need for a full analysis of an entire disk image. One important limitation being that files lose timestamp integrity when they are copied from the mounted image to another source.

The first step in producing the contents of an image excerpt was to identify desirable files and then assign a relative score based on ease of acquisition, processing time and availability open source parsing tools. Once identification was complete, certain forensic acquisition and processing tasks were segregated into sections so they could be individually included or excluded. Limitations that prevented or made acquisition or processing impractical were also documented.

In addition to speed, the procedure attempted to consolidate outputs from parsing tools to an easily readable format. To this end, output was stored in case folders using categories based on previous descriptors identified in the SANS Windows Artifact Evidence Of Poster\(^3\), Cory Harrell’s auto_rip\(^4\) and event descriptors for TLN and Log2timeline. Available tools found on SANS Sift were scripted together to produce an interface that allowed for the selective processing of artifact types. Extraction of known significant Windows registry values were automated using RegRipper and outputted using TLN plug-ins whenever possible.

### 2. Artifact Identification

The most notable forensic artifacts in Window forensic analysis are the Windows Master File Table (\(SMFT\)) and the Windows Registry. Fortunately, both the \(SMFT\) and

\(^{2}\) https://binaryforay.blogspot.ae/


\(^{4}\) http://journeyintoir.blogspot.ae/2013/05/unleashing-autorip.html
Windows registry files were acquired and parsed with relative ease. Other files of interest included Windows event logs, browser history and cache as well as prefetch, lnk files and server logs. Some artifacts identified were considered important but not practical for inclusion because they generally required full system access to a mounted volume such as file carving, file verification, extraction of compound files, Yara IOC and antivirus scanning.

Once a simple list of forensic artifacts was identified, values were assigned for the time and effort needed to acquire and process each file type. Ratings were individually assigned with ratings from one to five. One being the easiest and fastest and five considered the slowest, most difficult or not possible. Because the rating system was used primarily as a basis for enumeration, a non-scientific rating was used.

**Estimate of Acquisition and Processing Time**

After identifying the desired files, the basic assumption was that extraction and processing times would vary based on the role of the system. For example, the time needed to copy logs from a busy web server would take longer to complete than copying all files of interest from a freshly imaged laptop. Another issue considered was how to include and de-duplicate files contained in individual Volume Shadow Copies (VSCs).

The table below is a cursory list of artifacts and ratings that was used for the development of the acquisition and processing script sifgrab.sh. The list of artifact types were ordered from the least to greatest with estimated acquisition and processing times. Proposed open source Linux parsing tools (other than Log2Timeline) were identified and tested using SANS Sift v2018.08.0⁵. The Appendix contains a summary table that shows the final set of tools used to acquire and produce output using the script.

<table>
<thead>
<tr>
<th>Artifact</th>
<th>Acquisition Time Cost</th>
<th>Predictable Output Size</th>
<th>Processing Time</th>
<th>Total Time</th>
<th>Available Sift/Linux Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registry</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>Regripper</td>
</tr>
<tr>
<td>UsrClass.dat</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>Regripper</td>
</tr>
<tr>
<td>Amcache.hve</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>Regripper</td>
</tr>
<tr>
<td>Skype History</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>Sqlite3</td>
</tr>
<tr>
<td>Chrome Artifacts</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>Sqlite3, Hindsight</td>
</tr>
<tr>
<td>Mozilla Artifacts</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>Sqlite3</td>
</tr>
</tbody>
</table>


Using Image Excerpts to Jumpstart Windows Forensic Analysis
Using Image Excerpts to Jumpstart Windows Forensic Analysis

<table>
<thead>
<tr>
<th>Artifact</th>
<th>File(s) Added to SANS Sift</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registry</td>
<td>RegRipper 2.8⁶</td>
<td><a href="https://github.com/keydet89/RegRipper2.8">https://github.com/keydet89/RegRipper2.8</a></td>
</tr>
<tr>
<td>LNK Files</td>
<td>/usr/local/bin/pylnker.py</td>
<td><a href="https://github.com/HarmJ0y/pylnker">https://github.com/HarmJ0y/pylnker</a></td>
</tr>
<tr>
<td>Prefetch</td>
<td>/usr/local/bin/w10pf_parse.py</td>
<td><a href="http://github.com/bromley/tools/tree/master/win10_prefetch">http://github.com/bromley/tools/tree/master/win10_prefetch</a></td>
</tr>
<tr>
<td>Chrome Firefox</td>
<td>Sqlite3</td>
<td>sudo apt-get install sqlite3</td>
</tr>
<tr>
<td>INDXParse</td>
<td>/usr/local/bin/INDXParse.py</td>
<td><a href="https://github.com/williballenthin/INDXParse">https://github.com/williballenthin/INDXParse</a></td>
</tr>
<tr>
<td>Sushi File</td>
<td>Sushi</td>
<td>sudo apt-get update &amp; &amp; sudo apt-get install sushi</td>
</tr>
<tr>
<td>MS-Office Docs</td>
<td>Oletools</td>
<td>sudo –H pip install –U oletools</td>
</tr>
</tbody>
</table>

Table 1: Estimated Artifact Acquisition and Processing Times

Through testing, assumptions were either confirmed or new methods were and introduced and added once they were proven beneficial. As a result the following tools were added or updated to the SANS Sift:

Table 2: Tools added to SAN Sift

⁶ https://linuxconfig.org/how-to-install-regripper-registry-data-extraction-tool-on-linux
3. Image Collection and Processing

Since data collection was essentially a number of file copy operations and a number of open source tools (Installed Linux applications, Perl and Python scripts), a Linux shell script was written to automate file copies and allow the execution of different built in processing scripts. To provide interaction a simple menu was created and individual operations were added and grouped by function and speed. A mounting option was also provided to consolidate tasks within the menu. The numbered menu interface sequentially follows the acquisition and processing workflow with individual selections for items that take longer or are specific to the goals of a particular case.

The menu was built using numeric “case” statements with functions that call each option. Functions were included or excluded by editing the script and adding or removing a comment “#”. Extensions of the script were easily achieved by adding new functions to interact with new or existing case statements. Upon completion, the script, siftgrab.sh was uploaded to GitHub.

![Image 1: Siftgrab.sh menu](https://github.com/siftgrab/siftgrab)

---

7 https://askubuntu.com/questions/1705/how-can-i-create-a-select-menu-in-a-shell-script
8 https://bash.cyberciti.biz/guide/The_case_statement
9 https://github.com/siftgrab/siftgrab
Workflow for Creating Image Excerpts
The following workflow categories were used for the creation and processing of image excerpts:

- Disk Image Mounting
- Image Excerpt Acquisition
- Image Excerpt Processing

3.1 Disk Image Mounting

To provide more of an all-in-one solution, a disk mounting menu option was written for to the script. Mounting tools included in SANS Sift, mmls, ewfmount.py, mount and vshadowmount were automated to walk the analyst through the process and echo mount commands to the screen as they executed. A similar tool is available on SANS Sift called imageMount.py but was not used in favor of providing an interactive process.

3.2 Automating Image Excerpt Acquisition

Once images were mounted, data could be extracted and stored locally or on a network share. Excerpts from the source drive were stored into folders with the following path: $CASEDIR/$COMPUTERNAME/Artifact/LOCAL. Any data from mounted volume shadow copies were stored as a sub directory of “ARTIFACT” corresponding to the VSC mount point (e.g. vss1, vss2, vss3). Files collections were created as mirrored copies using rsync. As data was collected, duplicate files found on Volume Shadow Copies were removed. An accounting of every transfer was logged to the file “$CASEDIR/$COMPUTERNAME/Acquisition.log”. 

Once the source and destination folders were set, the script repetitively executed three basic Linux commands, find\(^\text{10}\), grep and rsync with a series of while loops to mirror

\(^{10}\) A basic description of looping “find” statements and the value of using them can be found here: https://forensicaliente.blogspot.ae/2010/07/creating-timeline-wmmls-fls.html
Using Image Excerpts to Jumpstart Windows Forensic Analysis

data from source to destination.

```
find ./*/AppData/Roaming/Microsoft/Windows/Recent -type f 2>/dev/null |
grep -i \lnk$ | sed 's|\./||'| sed 's|/ /\ /g' | while read d do
  rsync -aRq "$d" "$CASEDIR/$COMPNAME/Artifact/$DRV/$USER_DIR"
  done
```

**Image 2: Acquisition “looping” using basic Linux commands**

Functions were created for each artifact type to call commands to find and copy matching files. To provide flexibility in collection choices Volatile data, $USNJRNL and Outlook Mail files were give separate menu selections so they could be easily included or excluded.

### 3.3 Automation of Image Excerpt Processing with SANS Sift

After acquisition is complete, data could then be processed and extracted into a directory called “Triage”. By selectively choosing which data to process, it was possible to process a single volume (e.g. $COMPUTERNAME/Artifact/vss3), a single case folder (e.g. $COMPUTERNAME/Artifact) or multiple cases containing other external data (e.g. /SiftShare/cases). Because the script only searched only for artifacts by name, it was possible to direct the script to search across any file path for data to process. The easiest location for storing image excerpts and output was the “cases” directory on the SANS Sift which was by default configured as a cifs share.

### 4. Proof of Concept

As the script neared completion, a proof of concept test was put together to measure the time needed to create image excerpts and determine which artifacts were missing because of data reduction. Log2timeline was ran on excerpts to measure performance and compare output from sample disks. As information was learned about an artifact or processing times the script was modified to address bottlenecks or to provide additional capability. External output from SANS Sift processing tools were also refined to determine the best format for the review and triage.
**Test Lab Configuration**

The operating environment consisted of a Windows 10 laptop installed on a Solid-State Drive (SSD) which also held the SANS Sift VM and case folders. A one-terabyte external hard drive was connected to an open USB 3.0 port and was used as the source location for the E01 evidence image files. To improve performance, virtual hard disks on the SANS Sift VM were merged from multiple segments to a single file and then changed from a dynamic disk to pre-allocated.

The test lab setup used the following hardware and VM configuration:

<table>
<thead>
<tr>
<th><strong>Host Hardware</strong></th>
<th><strong>SANS Sift VM</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>OS: Windows 10 Home 64-bit</td>
<td>OS: Ubuntu 64-bit Desktop16.04.3</td>
</tr>
<tr>
<td>VMware Workstation Pro 14</td>
<td>Virtual Machine: SANS Sift Version v2018.08.0</td>
</tr>
<tr>
<td>Intel® Core™ i7-7500U CPU® 2.70GHz 2 Core 4 Logical Processors 16 GB RAM</td>
<td>2 Processors 8 GB RAM</td>
</tr>
<tr>
<td>VMware Host Shared Folders: F:\E01 (1TB External USB 3.0)</td>
<td>Network Shared Folders: /cases /mnt</td>
</tr>
<tr>
<td>USB3.0</td>
<td>USB 3.0 Compatibility</td>
</tr>
<tr>
<td>Internal: 500GB SSD (SanDisk SD8SN8U512G1002)</td>
<td>Virtual Disk: 50 GB HDD (Pre-allocated and stored in single file)</td>
</tr>
</tbody>
</table>

Table 3: Lab Hardware Setup Used in Proof of Concept

**Logical Workflow**

SANS Sift’s default configuration has built-in cifs shares of the “cases” and “mnt” directories and were used and provided an easy and efficient means for making data available to both Linux and Windows operating systems. In addition, the VMware share created on the host machine allowed user access from the SANS Sift workstation VM to an external hard disk that held evidence items. The VMware share was configured to emulate the common scenario of incoming evidence being brought into a lab for analysis.
Acquisition and Processing Times

Once the lab setup was finished, the script was used for image excerpt creation and parsing to measure the time required to collect and generate output. Initial results from eleven sample images with no volume shadow copies were recorded. A separate measurement of Volume Shadow Copies was also completed. The procedure consisted of mounting and creating an image excerpt using the siftgrab script. Image excerpts were then further processed for analysis using Regripper, Log2timeline and other tools. As expected, data acquisition size and times for volume shadow copies were nearly the same as the primary volume. The file extraction speed was also consistent with acquisition speeds averaging slightly less than two Gigabytes per minute. For the samples analyzed, acquisition times did not vary significantly even when extracting data from larger disks containing more data.

The full results of the acquisition and processing tests are provided below.

Sample evidence images were made up from ten workstations and one server. Four images came from laptops and seven from Virtual Machines.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Volume Size</th>
<th>Data on Volume</th>
<th>Extraction Time</th>
<th>Excerpt Size</th>
<th>Siftgrab Processor</th>
<th>Log2Timeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM Windows 10-64-3</td>
<td>100GB</td>
<td>23GB</td>
<td>2:01</td>
<td>.7GB</td>
<td>2:17</td>
<td>9:22</td>
</tr>
<tr>
<td>VM Windows 7 32</td>
<td>25GB</td>
<td>18GB</td>
<td>2:27</td>
<td>5.82GB</td>
<td>2:41</td>
<td>11:40</td>
</tr>
<tr>
<td>VM Windows 10 64 -1</td>
<td>60GB</td>
<td>16GB</td>
<td>2:40</td>
<td>2.1GB</td>
<td>2:20</td>
<td>13:03</td>
</tr>
<tr>
<td>VM Win2008R2</td>
<td>32GB</td>
<td>30GB</td>
<td>3:40</td>
<td>10.1GB</td>
<td>3:53</td>
<td>17:15</td>
</tr>
<tr>
<td>VM Windows 7 64-1</td>
<td>29GB</td>
<td>24GB</td>
<td>4:04</td>
<td>6.8GB</td>
<td>4:04</td>
<td>13:21</td>
</tr>
</tbody>
</table>
Using Image Excerpts to Jumpstart Windows Forensic Analysis

Table 4: Summary of Acquisition and Processing Data

<table>
<thead>
<tr>
<th>Image</th>
<th>Excerpt</th>
<th>Acquisition Times</th>
<th>Processing Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS Windows 7 64-2</td>
<td>100GB 23GB</td>
<td>4:10 7.58GB</td>
<td>4:09 12:12</td>
</tr>
<tr>
<td>VM Windows 7-64-4</td>
<td>100GB 49GB</td>
<td>4:26 4.8GB</td>
<td>2:53 16:18</td>
</tr>
<tr>
<td>WS Windows 7 64-3</td>
<td>239GB 59GB</td>
<td>6:03 15.6GB</td>
<td>5:20 12:53</td>
</tr>
<tr>
<td>WS Windows 10 64-2</td>
<td>477GB 150GB</td>
<td>7:37 15.5GB</td>
<td>3:27 41:11</td>
</tr>
</tbody>
</table>

* USNJRNL:$J excluded

The following charts summarize the test data collected from completed image extraction and case processing. For all images examined, image excerpts were acquired in less than ten minutes. Data from all image excerpts completed processing in less than forty-five minutes.

Summary of Acquisition and Processing Tests

The only failure in the testing process occurred during an image excerpt acquisition of a Windows 10 workstation that had a single $USNJRNL file that was eighty Gigabytes (80GB) in size. Because the single file exceeds the entire size of the SANS Sift partition the collection was cancelled and the file excluded. The script was later amended to include a file size check to exclude and log the existence of files that exceed a file size of five hundred megabytes. Otherwise, data extraction and processing of mounted volumes was completed without issue in minutes.

5. Reviewing Results

After data was successfully extracted from mounted images and processed by siftgrab, output was reviewed and searched from the Windows 10 VMware host. By tweaking Windows Explorer’s preview option and advanced search options, a basic interface was available to peruse and search the Triage folder for each case. Several
open source tools available for searching and otherwise extending the capabilities of Windows explorer and were downloaded and tested on the data output. Among the tools reviewed were Agent Ransack, Directory Opus, AstroGrep and Atom text editor. Each tool had benefits and limitations with best results being achieved by using a Windows browsing tool to walk through and preview Triage output and then use another tool to grep for a date or strings. Another handy utility added was the Sushi file manager, which has a feature that emulates the Mac OS X preview on Linux systems.

Image 5: Enabling Sift Shares and Windows Explorer's Built-in Preview Options

Triage output that was generated from images that had volume shadow copies could be easily compared. For instance, by just selecting files in the Triage output in Windows explorer, it was possible to quickly scan files from a preview pane and compare RegRipper output for multiple SAM files. Individual and multiple volume shadow copies could also be selected for processing as well groups of registry or other artifacts files from different systems. This capability helps to automate repetitive processes as well as produce timelines and other searchable results.

**Adding Processing Functions**

Using available open source tools in the SANS Sift and elsewhere, it was possible to extract metadata from most artifacts of interest. Additional parsing functions were added to the script to either streamline or provide additional timeline output. After the processing functions were added to siftgrab, individual bash scripts were created for each
newly created function so individual artifacts could be extracted or analyzed from the command line outside of siftgrab.

5.1. Converting TLN to Five Columnar CSV

The siftgrab script ran many of the Regripper parsers that produced TLN. TLN was then converted on the fly to five columnar CSV with humanly readable time stamps.

Processing functions for Alternate Data Streams, the recycle bin, browser history and other artifact were also extracted using the same format. To maintain compatibility with the TLN format and to provide an easy method to convert back and forth from CSV to TLN, separate file conversion scripts were created.

5.2. Extracting Alternate Data Streams

The first custom function added was a process to find and extract alternate data streams (ADS) on mounted NTFS volumes. The “attr” package provided access to the NTFS extended file attributes using the “getfattr” command. Timestamps were identified using the “stat” command and file name were outputted as a five columnar comma separated file. The following script can be run to find files that contain alternate data streams on mounted NTFS volumes.

---

5.3. Recycle Bin Parsing

The Perl script recbin.pl on the SANS Sift was identified as a possible method to parse items in the $Recycle.bin and $Recycler directories. But after some initial research, it was decided to create a function to extract the data instead. Because Windows recycle bins consists of two file types, a metadata file and a renamed version of the original file, it was only necessary to acquire and extract the metadata ($I*.* and INFO2) files. The extraction process script created below extracts the strings from the “$I” metadata files and decodes the deletion date field. Output from the script can be either TLN or five columnar CSV.

```bash
#!/bin/bash
# http://linuxleutring.blogspot.co.uk/2009/10/processing-vista-recyclebin.html
# https://github.com/keryder/#!/tools/blob/master/source/recbin.pl
# Read "$I" entries from mounted NTFS volume's $RecycleBin and extracts to TLN with epoch "-e" or timestamp "default"
find $1/$RecycleBin -type f | grep -m 1 '[^\d]*$I*.*[^\d]' | while read d;
  echo "$d$
name=$(basename "$d")
hexsize=$(stat "$d" | awk '/textures/ {print $1}')</hexsize>)
hexdate=$(stat "$d" | awk '/textures/ {print $6}')</hexdate>)
epoch=$(echo $(((0x$(printf "%-32x", $hexsize)) * 10)) | awk 'A+1{print=($1+1)*10}1')
date=$(($epoch - 11644673600))

["$I" -- "-$e"] & echo "Filename: $name"
["$I" -- "$e"] & echo "$date, $hexdate $hexsize $epoch"
done
```

Image 9: recbin2tln.sh: Script to read and extract metadata from Windows Recycle Bin

5.4. Parsing Browser History and Skype Logs with SQLITE3

Hindsight is a tool available on the latest SANS Sift that parses Google Chrome browsing artifacts and outputs very detailed information. Output from Hindsight was included in the siftgrab processor. To create timeline entries for Chrome, Firefox and Skype, functions were created to run SQLite and extract relevant history. As with the
other scripts output was generated in five columnar CSV compatible with TLN.

```bash
#!/bin/bash
# script2csv.sh usage:script2csv.sh [path to main.db]
# Extracts Skype information from main.db to five columnar CSV (Default) or TLN (-e)
# https://gist.github.com/3ft/4466231

parse_maindb()
function parse_maindb()
    target_path="$(echo $target_path | sed 's/main.db//')"
    id=$target_path

    sql"xml" target_path="skype" full_name="full_name" display_name="display_name" from Contacts awk -F'\'| {print $1"[SKYPE]"NEW CONTACT: "$2", "$3", "$4" | sed 's/\|\|/\|/'}"
    sql"xml" target_path="skype" timestamp="timestamp" author="author" dialing_partner="dialing_partner" body="body" and from Messages awk -F'| {print $1"[SKYPE]"FROM: "$2", "$3" MESSAGE: "$4" | sed 's/\|\|/\|/'}"
    sql"xml" target_path="skype" timestamp="timestamp" dialing_partner="dialing_partner" body="body" path from Voicemails awk -F'| {print $1"[SKYPE]"FROM: "$2", "$3" MESSAGE: "$4" | sed 's/\|\|/\|/'}"
    sql"xml" target_path="skype" timestamp="timestamp" dialing_partner="dialing_partner" body="body" path from Conversations awk -F'| {print $1"[SKYPE]"CONVERSATION START: "$2" | sed 's/\|\|/\|/'}"
    sql"xml" target_path="skype" timestamp="timestamp" dialing_partner="dialing_partner" body="body" path from Transfers awk -F'| {print $1"[SKYPE]"FILE TRANSFERED: "$2" | sed 's/\|\|/\|/'}"

$["xml" target_path="skype" selects 'select * from Contacts'] | echo Usage: script2csv.sh [path to main.db] [-e]

#E
[ "$2" == "-e" ] && parse_maindb|sort -rn

Image 10: skype2csv.sh: Script to extract Skype log information from main.db

5.5. Windows Event Log Quick Reference

Siftgrab included a separate menu selection to launch evtxexport and extract Windows event logs. The result is a set of searchable XML files that can be accessed from explorer’s preview pane, Directory Opus or other more powerful tools that support regular expressions. In order to provide additional usability, a Windows Event Log Quick Reference was created to copy into the same directory as the extracted Windows Event XML file. A copy of the quick reference can be found in the Appendix B.

6. Scaling Analysis Across Multiple Endpoints

The final step taken was to update the script so that data could be processed and analyzed from multiple systems at once. Image excerpts from the eleven systems processed earlier as well as output from three additional live endpoints were processed as a single case. To test and simulate a live response, data from collection tool, CYLR was used to acquire artifacts from the three running systems. Computer names were added to individual output files and to TLN to better identify artifact entries.

Ultimately, the usability of any output depended on how efficiently the data could be extracted and used comparatively. The use of TLN provided a simple means for creating information that could be sorted and stacked to compare artifact differences. TLN could also be easily exported to a SQLite database or imported to an Elastic Stack instance via Logstash. Although other data that was not in a timeline format is could still

© 2018 The SANS Institute  Author retains full rights.
be cumulatively stored and searched for IOC strings and timestamps based on artifact types, such as program execution or USB access.

7. Limitations

Some files were acquired but not processed and there were some forensics tasks identified but not included in the script. The following table identifies these items. There were some tasks that were tested and not fully implemented. The functions for those tasks are still in the script and can be activated by removing the “#” in the menu item area. While it was technically possible to consolidate and automate most tasks, some output and parsing actions were intentionally left out because they diminished speed and performance which is the main benefit of the script.

<table>
<thead>
<tr>
<th>Artifact</th>
<th>Partially Implemented</th>
<th>Alternate solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>logfile</td>
<td>No</td>
<td>Triforce ANJP</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="https://www.gettriforce.com/product/triforce-anjp/">https://www.gettriforce.com/product/triforce-anjp/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Flash Cookie View</td>
</tr>
<tr>
<td>Flash cookies</td>
<td>No</td>
<td><a href="https://www.nirsoft.net/utils/flash_cookies_view.html">https://www.nirsoft.net/utils/flash_cookies_view.html</a></td>
</tr>
<tr>
<td>Jump list</td>
<td>No</td>
<td>JLECMD</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="https://ericzimmerman.github.io/Software/JLECmd.zip">https://ericzimmerman.github.io/Software/JLECmd.zip</a></td>
</tr>
<tr>
<td>File integrity</td>
<td>No</td>
<td>File, Md5sum, md5deep</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://md5deep.sourceforge.net/">http://md5deep.sourceforge.net/</a></td>
</tr>
<tr>
<td>Yara</td>
<td>No</td>
<td>Loki</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="https://github.com/Neo23x0">https://github.com/Neo23x0</a></td>
</tr>
<tr>
<td>MS-Office Docs</td>
<td>Yes (oletools)</td>
<td>Oletools (activate function in siftgrab)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ObjDump.py</td>
</tr>
<tr>
<td>WebcacheV01</td>
<td>Yes (esedbexport)</td>
<td></td>
</tr>
<tr>
<td>Anti-Virus Scan</td>
<td>Yes (ClamAV)</td>
<td>ClamAV (activate function in siftgrab)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Scan mounted volume from Windows Share</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NirSoft Browser Tools</td>
</tr>
<tr>
<td>Browser cache</td>
<td>Yes (Hindsight)</td>
<td><a href="http://www.nirsoft.net/web_browser_tools.html">http://www.nirsoft.net/web_browser_tools.html</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sigcheck</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="https://docs.microsoft.com/en-us/sysinternals/downloads/sigcheck">https://docs.microsoft.com/en-us/sysinternals/downloads/sigcheck</a></td>
</tr>
<tr>
<td>File Signed</td>
<td>No</td>
<td>File</td>
</tr>
<tr>
<td>Verify File Type</td>
<td>No</td>
<td>Pdffd.py, pdfparser</td>
</tr>
<tr>
<td>PDF</td>
<td>No</td>
<td><a href="https://blog.didierstevens.com/programs/pdf-tools/">https://blog.didierstevens.com/programs/pdf-tools/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Elastic Stack</td>
</tr>
<tr>
<td>Windows Logs</td>
<td>No</td>
<td>Log Parser</td>
</tr>
<tr>
<td>File Carving</td>
<td>Yes (Foremost)</td>
<td>Foremost (activate function in Siftgrab)</td>
</tr>
<tr>
<td>EVTIX</td>
<td>Yes (Evtexexport to XML)</td>
<td>Eventlog Explorer</td>
</tr>
<tr>
<td>XP Support</td>
<td>Yes</td>
<td>Legacy item intentionally left out</td>
</tr>
<tr>
<td>INDX Files</td>
<td>Yes (INDXparse.py)</td>
<td>INDXParse (activate function in Siftgrab)</td>
</tr>
</tbody>
</table>

Table 5: Artifacts and metadata not included in siftgrab.sh
8. Conclusion

By simply prioritizing forensic artifacts based on file size and processing time, you can very quickly acquire and generate a large amount of high value information from Windows disk images. Acquired files can be targeted for analysis individually using a compatible tool or through script automation. Output from data collected from multiple systems or artifacts acquired over the network or from different sources can be compared to identify anomalies.

In addition to providing a method for acquiring image excerpt data, this analysis attempted to enumerate and test possible open source parsing tools and merge results into a consolidated case file. There are many forensic tools currently available on the SANS Sift workstation and the Linux platform that were called from siftgrab.sh to generate timeline and other triage information. This output depended largely tools developed by Harlan Carvey and others who have written scripts to parse Windows files to extract forensic artifacts.

The approach described here attempts to show the value of working with a reduced data set and focused analysis of specific artifacts. Time can be efficiently managed when a second system is used to perform tasks that take longer to complete. Additionally, automation can be achieved using scripts and available tools found in the SANS Sift and other sources. Beyond current capabilities of siftgrab.sh, converting the script to python would allow for a more modular approach. A lightweight forensic artifact scanner could be built to streamline and consolidate similar processes and lower the dependency on external tools.
References


Using Image Excerpts to Jumpstart Windows Forensic Analysis


SANS Forensics and Incident Response Summit (June 2012)


## Appendix A: Tools List

Summary of tools called from Siftgrab.sh

<table>
<thead>
<tr>
<th>Artifact</th>
<th>Tools</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>$MFT</td>
<td>AnalyzeMFT.py</td>
<td><a href="https://github.com/dkovar/analyzeMFT">https://github.com/dkovar/analyzeMFT</a></td>
</tr>
<tr>
<td>$USNJRNL:$J</td>
<td>Parseusn.py</td>
<td><a href="https://code.google.com/archive/p/parser-usnjrnl/">https://code.google.com/archive/p/parser-usnjrnl/</a></td>
</tr>
<tr>
<td>Windows Registry (SAM, SYSTEM, SOFTWARE, SECURITY, NTUSER.DAT)</td>
<td>RegRipper</td>
<td><a href="https://github.com/keydet89/RegRipper2.8">https://github.com/keydet89/RegRipper2.8</a></td>
</tr>
<tr>
<td>UsrClass.dat</td>
<td>RegRipper</td>
<td><a href="https://github.com/keydet89/RegRipper2.8">https://github.com/keydet89/RegRipper2.8</a></td>
</tr>
<tr>
<td>AmCache.hve</td>
<td>RegRipper</td>
<td><a href="https://github.com/keydet89/RegRipper2.8">https://github.com/keydet89/RegRipper2.8</a></td>
</tr>
<tr>
<td>Chrome History, Cache</td>
<td>Sqlite3</td>
<td><a href="https://www.sqlite.org">https://www.sqlite.org</a></td>
</tr>
<tr>
<td>FireFox Places.sqlite, Cookie.sqlite</td>
<td>Sqlite3</td>
<td><a href="https://www.sqlite.org">https://www.sqlite.org</a></td>
</tr>
<tr>
<td>Prefetch Files</td>
<td>Prefetch.py</td>
<td><a href="https://github.com/PoorBillionaire/Windows-Prefetch-Parser">https://github.com/PoorBillionaire/Windows-Prefetch-Parser</a></td>
</tr>
<tr>
<td>Link Files</td>
<td>Pylinker</td>
<td><a href="https://github.com/HarmJ0y/pylnker">https://github.com/HarmJ0y/pylnker</a></td>
</tr>
<tr>
<td>WebCacheV.dat</td>
<td>Esedbexport</td>
<td><a href="https://github.com/libyal/libesedb/tree/master/esedbtools">https://github.com/libyal/libesedb/tree/master/esedbtools</a></td>
</tr>
<tr>
<td>IE (Index.dat)</td>
<td>Parseie.pl</td>
<td><a href="https://github.com/keydet89/Tools/tree/master/master/source">https://github.com/keydet89/Tools/tree/master/master/source</a></td>
</tr>
<tr>
<td>Anti-Virus Scan</td>
<td>ClamAV</td>
<td><a href="https://www.clamAV.net">https://www.clamAV.net</a></td>
</tr>
<tr>
<td>Outlook Mail OST, PST</td>
<td>Pffexport</td>
<td><a href="https://github.com/libyal/libpff/blob/master/pftools">https://github.com/libyal/libpff/blob/master/pftools</a></td>
</tr>
<tr>
<td>Skype History Main.db</td>
<td>Sqlite3</td>
<td><a href="https://www.sqlite.org">https://www.sqlite.org</a></td>
</tr>
<tr>
<td>Job</td>
<td>Jobparser.py</td>
<td><a href="https://raw.githubusercontent.com/gleeda/misc-scripts/master/misc_python/jobparser.py">https://raw.githubusercontent.com/gleeda/misc-scripts/master/misc_python/jobparser.py</a></td>
</tr>
<tr>
<td>ADS</td>
<td>Getfattr</td>
<td><a href="http://download.savannah.gnu.org/releases/attr/">http://download.savannah.gnu.org/releases/attr/</a></td>
</tr>
<tr>
<td>Windows Event (evt)</td>
<td>Evtxexport</td>
<td><a href="https://github.com/libyal/libevtx">https://github.com/libyal/libevtx</a></td>
</tr>
<tr>
<td>Windows Event (evtx) * .doc,*.docx, xls, xlsx, ppt, ppto</td>
<td>Oleid</td>
<td><a href="https://github.com/decalage2/oletools/">https://github.com/decalage2/oletools/</a></td>
</tr>
<tr>
<td>File Carving</td>
<td>Foremost</td>
<td><a href="http://foremost.sourceforge.net/pkg/foremost-1.5.7.tar.gz">http://foremost.sourceforge.net/pkg/foremost-1.5.7.tar.gz</a></td>
</tr>
<tr>
<td>Image Mounting</td>
<td>EWFTools</td>
<td><a href="https://github.com/libyal/libewf/">https://github.com/libyal/libewf/</a></td>
</tr>
</tbody>
</table>

*Table 6: Tools dependency for Siftgrab.sh*
Appendix B: Windows Event IDs

Siftgab.sh contains the following list of Windows Event IDs and descriptions

******************************************************************************
** WINDOWS EVENTLOG QUICK REFERENCE (EVTX) **
******************************************************************************

https://www.ultimatewindowssecurity.com/
http://www.deer-run.com/~hal/IEventLogAnalysis.pdf
https://github.com/dchad/fine-line-computer-forensics-timeline-tools
https://www.mindmeister.com/841877457/event-log
http://www.redblue.team/2015/09/spotting-adversary-with-windows-event.html
http://kb.eventtracker.com/evtpass/evtbksearch_result.asp
http://www.eventid.net/
https://github.com/iadgov/Event-Forwarding-Guidance/blob/master/Events/RecommendedEvents.csv

******************************************************************************
** SECURITY EVTXX **
******************************************************************************

FILE, NETWORK OR LOG CLEARING EVENTS
1102 - Log Clearing
4688 - Process Created (Program Execution)
4656 - Access to File or Other Object Requested
4663 - Attempt made to access a file or object
4658 - Access to a File or object closed
4697 - New Service has been Installed
5140 - Network Share Accessed

LOGIN EVENTS
4624 - Network logon
4625 - Login Failed
4634 - Logoff
4648 - Attempted Login
4672 - Administrator has Logged in
4776 - Credential Authentication (Success/Fail)
4778 - Session Reconnect (RDP or FastUser Switch)
4770 - Kerberos Ticket Renewed
4793 - Password Policy Checking API called

CHANGE TO ACCOUNT OR ACCOUNT STATUS EVENTS
4704 - User Right Assigned
4720 - New User Account Created
4722 - New User Account Enabled
4725 - User Account Disabled
4726 - User Account Deleted
4728 - Member Added to Global Group
Using Image Excerpts to Jumpstart Windows Forensic Analysis

4731 - Security-enabled Group Created
4732 - Member Added to local Group
4733 - Account removed from Local Sec. Group
4765 - SID History added to Account
4634 - Local Group Deleted
4735 - Local Group Changed
4740 - Account Locked out
4748 - Local Group Deleted
4756 - Member Added to Universal Group
4766 - SID History add attempted on Account
4767 - User Account Unlocked
4781 - Account Name Changed

CHANGES TO FIREWALL
4946 - Firewall Rule has been Added
4947 - Firewall Rule has been Modified
4948 - Firewall Rule has been Deleted
4950 - Firewall Rule has been Changed

******************************
** APPLICATION.EVTX **
******************************
865 - GPO Blocked - Exe Default Security Level
866 - GPO Blocked exe - Restricted Path
867 - GPO Blocked Exe - Certificate rule
868 - GPO Blocked Exe - zone or hash rule
882 - GPO Blocke Exe by Policy Rule
1000 - Application Error 1001 - WER Info
1001 - EMET !=Warning 2=Error
1002 - Application Hang Software Policy Events

******************************
** SYSTEM.EVTX **
******************************
1074 - System Halt
7000 - Service failed to start: did not respond to the start control request
7022 - Service hung on start
7023 - Service terminated with error
7024 - Service terminated with error
7026 - Service failed on system start
7031 - Service terminated unexpectedly
7034 - Service terminated unexpectedly
7035 - Service sent a request to Stop or Start
7036 - Service was Started or Stopped
7045 - service Installed
7040 - Service changed from "auto start" to "disabled"

*******************************************************************************
** Microsoft-Windows-TaskScheduler%4Operational.evtx **
*******************************************************************************
Using Image Excerpts to Jumpstart Windows Forensic Analysis

106 - Task scheduled
200 - Task executed
201 - Task completed
202 - Task Failed to complete
140 - Task Updated
141 - Task Deleted
142 - Task Disabled
145 - Computer woke up by TaskScheduler
300 - Task Scheduler Started
400 - Task Scheduler Service Started

****************************************************
** MICROSOFT-WINDOWS-WINDOWS DEFENDER/OPERATIONAL **
****************************************************
1005 - Scan Failed
1006 - Malware Detected
1008 - Action on Malware Failed
2000 - Signature Updated
2001 - Signature Update Failed
2003 - Engine Update Failed
2004 - Reverting to Last Known Gadd Signatures
3001 - Real-Time Protection Stopped
5008 - Unexpected Error

****************************************************
** Microsoft-Windows-TerminalServices-RemoteConnectionManager.EVTX **
****************************************************
261 - Terminal Service Received Connection
1006 - Large Number of Connection Attempts
1149 - User authenticated

****************************************************
** Microsoft-Windows-TerminalServices-LocalSessionManager.EVTX **
****************************************************
21 - logon success
23 - logoff
24 - disconnect

****************************************************
** PtH Detection for lateral movement: **
****************************************************
Event ID: 4624
Event level: Information
LogonType: 3
Logon method NTLM
A local logon that is not ANONYMOUS

Table 7: Windows event log quick reference included in Siftgrab.sh
<table>
<thead>
<tr>
<th>Event Name</th>
<th>Location</th>
<th>Dates</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS Cyber Defence Bangalore 2018</td>
<td>Bangalore, IN</td>
<td>Jul 16, 2018 - Jul 28, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Riyadh July 2018</td>
<td>Riyadh, SA</td>
<td>Jul 28, 2018 - Aug 02, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Pittsburgh 2018</td>
<td>Pittsburgh, PAUS</td>
<td>Jul 30, 2018 - Aug 04, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Hyderabad 2018</td>
<td>Hyderabad, IN</td>
<td>Aug 06, 2018 - Aug 11, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS August Sydney 2018</td>
<td>Sydney, AU</td>
<td>Aug 06, 2018 - Aug 25, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Boston Summer 2018</td>
<td>Boston, MAUS</td>
<td>Aug 06, 2018 - Aug 11, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS New York City Summer 2018</td>
<td>New York City, NYUS</td>
<td>Aug 13, 2018 - Aug 18, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Virginia Beach 2018</td>
<td>Virginia Beach, VAUS</td>
<td>Aug 20, 2018 - Aug 31, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Prague 2018</td>
<td>Prague, CZ</td>
<td>Aug 20, 2018 - Aug 25, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS San Francisco Summer 2018</td>
<td>San Francisco, CAUS</td>
<td>Aug 26, 2018 - Aug 31, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS SEC504 @ Bangalore 2018</td>
<td>Bangalore, IN</td>
<td>Aug 27, 2018 - Sep 01, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Copenhagen August 2018</td>
<td>Copenhagen, DK</td>
<td>Aug 27, 2018 - Sep 01, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Amsterdam September 2018</td>
<td>Amsterdam, NL</td>
<td>Sep 03, 2018 - Sep 08, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Wellington 2018</td>
<td>Wellington, NZ</td>
<td>Sep 03, 2018 - Sep 08, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Tokyo Autumnn 2018</td>
<td>Tokyo, JP</td>
<td>Sep 03, 2018 - Sep 15, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Tampa-Clearwater 2018</td>
<td>Tampa, FLUS</td>
<td>Sep 04, 2018 - Sep 09, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS MGT516 Beta One 2018</td>
<td>Arlington, VAUS</td>
<td>Sep 04, 2018 - Sep 08, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>Threat Hunting &amp; Incident Response Summit &amp; Training 2018</td>
<td>New Orleans, LAUS</td>
<td>Sep 06, 2018 - Sep 13, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Baltimore Fall 2018</td>
<td>Baltimore, MDUS</td>
<td>Sep 08, 2018 - Sep 15, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Alaska Summit &amp; Training 2018</td>
<td>Anchorage, AKUS</td>
<td>Sep 10, 2018 - Sep 15, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Munich September 2018</td>
<td>Munich, DE</td>
<td>Sep 16, 2018 - Sep 22, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS London September 2018</td>
<td>London, GB</td>
<td>Sep 17, 2018 - Sep 22, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Network Security 2018</td>
<td>Las Vegas, NVUS</td>
<td>Sep 23, 2018 - Sep 30, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>Oil &amp; Gas Cybersecurity Summit &amp; Training 2018</td>
<td>Houston, TXUS</td>
<td>Oct 01, 2018 - Oct 06, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANSFIRE 2018</td>
<td>OnlineDCUS</td>
<td>Jul 14, 2018 - Jul 21, 2018</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS OnDemand</td>
<td>Books &amp; MP3s OnlyUS</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
</tbody>
</table>