

TZWorks Windows AppCompatibility Cache Utility (wacu) Users Guide



Abstract

wacu is a standalone, command-line tool that targets the AppCompatibility Cache data in the registry. This data is useful for analysts in determining which applications were run on the system in the past. **wacu** can operate on a live volume, an extracted system hive, an image of a volume or a *VMWare* volume. The parsed results can be outputted in one of three formats for easy inclusion with other forensics artifacts. **wacu** has binaries that run on Windows, Linux and Mac OS-X.

Copyright © TZWorks LLC

www.tzworks.com

Contact Info: info@tzworks.com

Document applies to v0.53 of **wacu**

Updated: Apr 25, 2025

Table of Contents

| | | |
|-----|---|----|
| 1 | Introduction | 2 |
| 2 | How to use <i>wacu</i> | 3 |
| 2.1 | Processing the System Hive from the System Volume | 3 |
| 2.2 | Processing a Specified System Hive | 3 |
| 2.3 | Processing Volume Shadow Copies | 4 |
| 2.4 | Understanding the Output..... | 4 |
| 2.5 | Slack Space in the AppCompatibility Cache | 6 |
| 2.6 | Extracting the Raw <i>AppCompatCache</i> data from the System Hive | 7 |
| 3 | Available Options | 8 |
| 4 | Authentication and the License File..... | 10 |
| 4.1 | <i>Limited</i> versus <i>Demo</i> versus <i>Full</i> in the tool's Output Banner..... | 10 |
| 5 | References | 11 |

TZWorks AppCache Parser (*wacu*) Users Guide

Copyright © TZWorks LLC

Webpage: http://www.tzworks.com/prototype_page.php?proto_id=29

Contact Information: info@tzworks.com

1 Introduction

wacu is a command line tool that targets the Windows system registry hive *AppCompatibility Cache* subkey. It was designed to: (a) extract this type of data from the registry and archive the binary data to a file for later analysis, and (b) parse the target data from various sources (live or exported system hive, image of the system volume or from a file containing solely the binary data).

As background, the *Application Compatibility Cache* is used by the Windows operating system to help identify application compatibility issues with the goal of trying to resolve them, so legacy applications can run on the newer version of the operating system. A search through the registry will show this data is spread across a number of hives, including the user, system and software hives. The Windows operating system looks at the combination of data collected, and/or set by the user, to determine which applications require shims to be used for compatibility purposes.

For the case of **wacu**, only data from the *system* hive is analyzed. Specifically, **wacu** looks to one of the following registry subkeys depending on the version of the operating system:

For Windows XP:

HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatibility\AppCompatCache

For versions beyond Windows XP

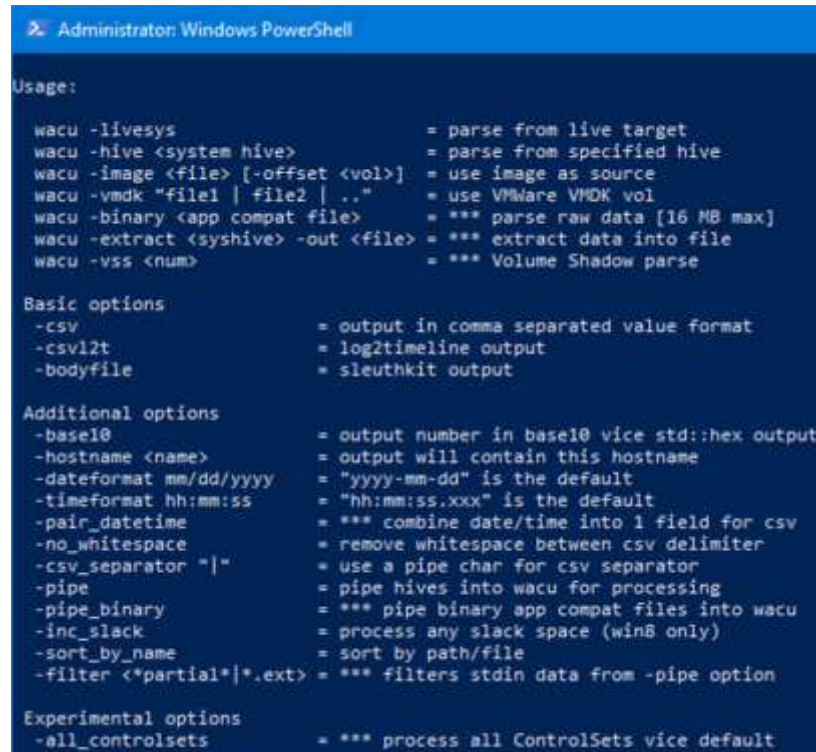
HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache\AppCompatCache

From a forensics standpoint, the data in the above subkeys gives the investigator insight into which applications were executed on the system. One should note, however, that the presence of a filename in this data does not necessarily mean that the file was executed. With this caveat, the data can be useful in tracking the initial infection vector of malware. This is because if a user mode application was run on the system, the chances are good it was recorded in this cache data.

For those user mode executables recorded, one gets the modification file time for the target binary along with other miscellaneous data.

2 How to use *wacu*

wacu has a number of command line switches that can be displayed at the command prompt. Below is a screen shot of the menu displayed. Each option is explained in the section on available option.



```
Administrator: Windows PowerShell

Usage:

wacu -livesys                = parse from live target
wacu -hive <system hive>     = parse from specified hive
wacu -image <file> [-offset <vol>] = use image as source
wacu -vmdk "file1 | file2 | .." = use VMWare VMDK vol
wacu -binary <app compat file> = *** parse raw data [16 MB max]
wacu -extract <syshive> -out <file> = *** extract data into file
wacu -vss <num>              = *** Volume Shadow parse

Basic options
-csv                = output in comma separated value format
-csv12t             = log2timeline output
-bodyfile           = sleuthkit output

Additional options
-base10             = output number in base10 vice std::hex output
-hostname <name>    = output will contain this hostname
-dateformat mm/dd/yyyy = "yyyy-mm-dd" is the default
-timeformat hh:mm:ss = "hh:mm:ss.xxx" is the default
-pair_datetime      = *** combine date/time into 1 field for csv
-no_whitespace      = remove whitespace between csv delimiter
-csv_separator "|"   = use a pipe char for csv separator
-pipe               = pipe hives into wacu for processing
-pipe_binary        = *** pipe binary app compat files into wacu
-inc_slack          = process any slack space (win8 only)
-sort_by_name       = sort by path/file
-filter <*partial*|*.ext> = *** filters stdin data from -pipe option

Experimental options
-all_controlsets    = *** process all ControlSets vice default
```

2.1 Processing the System Hive from the System Volume

When running **wacu** on a Windows machine, one can access that machine's system hive directly by using the **-livesys** switch. For this case, since one is targeting a volume mounted on a live Windows system, one needs to be run **wacu** with administrator privileges. Below is an example:

```
wacu -livesys > results.txt
```

2.2 Processing a Specified System Hive

Other options are designed to work on either extracted system hives or NTFS system volume images that contain a system hive. When working with images, one needs to use the offset of the system volume containing the registry hives. Below are some examples:

```
wacu -image "my_disk_image.dd" -offset 0x019db00000 -csv > results.csv
```

```
wacu -hive k:\host_xyz\system.bin -csv > results.csv
```

2.3 Processing Volume Shadow Copies

For starters, to access Volume Shadow copies, one needs to be running with administrator privileges. Also, Volume Shadow copies, as is discussed here, only applies to Windows Vista, Win7, Win8 and beyond. It does not apply to Windows XP.

There are 2 options available to pointing to the system hive on a Volume Shadow. The first option follows the format of the other tools and uses the built-in shortcut syntax to access a specified Volume Shadow copy, via the **%vss%** keyword. This internally gets expanded into `\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy`. Thus to access index 1 of the volume shadow copy, one would prepend the keyword and index, like so, **%vss%1** to the normal path of the hive. For example, to access a System hive located on *HarddiskVolumeShadowCopy1*, the following syntax can be used:

```
wacu -hive %vss%1\Windows\System32\Config\System > results.txt
```

The second option is much easier and uses the **-vss <index of Volume Shadow>** syntax. Below yields the same result as the first one above.

```
wacu -vss 1 > results.txt
```

To determine which indexes are available from the various Volume Shadows, one can use the Windows built-in utility **vssadmin**, as follows:

```
vssadmin list shadows
```

To filter some of the extraneous detail, type

```
vssadmin list shadows | find /i "volume"
```

While the amount of data can be voluminous, the keywords one needs to look for are names that look like this:

```
Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1
```

```
Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2
```

```
...
```

From the above, notice the number after the word *HarddiskVolumeShadow*. It is this number that is passed as an argument to the previous options.

2.4 Understanding the Output

There are two main forms of output, both of which use one line per entry parsed. The default output is formatted text, where the fields are nicely space with a pipe delimiter between fields. This is useful when trying to view the output with a *notepad* type application. The second output is the standard comma separated value (CSV) output, and it is invoked via the **-csv** switch.

One caveat with the CSV option is since the field separators are commas, there is the possibility that filenames may have a comma(s) as part of their name. If this happens, then the field separators will be misaligned for that entry. Therefore, to maintain alignment of the data, **wacu** will internally replace any commas in the name to spaces. This behavior is only done if commas are used as the field delimiter. If this is a concern, one is recommended to change the default comma delimiter (in the **-csv** option) to a pipe '|' delimiter, via the **-csv_separator "|"** option.

The other thing to understand about the output is it varies depending on the version of the operating system the hive came from. This is because certain artifacts available in one version of Windows may or may not be available in another version of Windows, as it relates to *AppCompatibility Cache* artifacts. Therefore, the output generated will be a function of what version of Windows the system hive originated from. Furthermore, some of the data parsed is not fully understood (*as far as what is the meaning of the data*) by the forensics community at large. For these cases, the raw data will be shown along with a 'best guess' of the interpretation of the data. This 'guess' is based on empirical results, and as more samples are observed over time, the better the refinement of the interpretation. Finally, to ensure clarity, this data will be identified as "experimental" in the column header and will only be available with licensed versions.

The common output between all the operating systems is the target binary's file modification time, entry number parsed, and target path/filename.

Below are two examples. The first is the output from a Windows XP image used in the older forensics 408 by SANS Institute. The second example demonstrates the parsing of the system hive from a live Windows 7, 64 bit system. Note the differences in the output between the two operating systems. In the Windows XP case, the entries have an ordering as well as the file size of the target binary. In the Windows 7 case, these entries are not present, however, two other entries designated as flags are, along with a 'best guess' of the meaning of the flags.

WinXP, 32 bit
Example targets an image

WinXP includes file size as well as ordering

| file mod | time-UTC | xupdate | time-UTC | entry | order | file sz | Path/File |
|------------|--------------|------------|--------------|-------|-------|------------|---|
| 06/30/2007 | 22:51:17.313 | 06/30/2007 | 22:51:17.363 | 13 | 33 | 0x0049c2e8 | ...Content.IE5\B79M1DCT\3hw3hw[1].exe |
| 07/01/2007 | 15:50:28.092 | 07/01/2007 | 15:50:28.222 | 30 | 14 | 0x00c1648 | ...Content.IE5\B79M1DCT\Install ADM[1].exe |
| 06/30/2007 | 22:56:48.890 | 06/30/2007 | 22:56:48.820 | 17 | 29 | 0x00df568 | ...Content.IE5\B79M1DCT\winsta120[1].exe |
| 07/01/2007 | 15:49:16.509 | 07/01/2007 | 15:49:16.529 | 28 | 16 | 0x01540d28 | ...Content.IE5\B79M1DCT\SkypeSetup[1].exe |
| 10/21/2008 | 17:16:18.080 | 07/01/2007 | 15:50:59.517 | 32 | 11 | 0x00000019 | ...Program Files\ADM Search\register.bat |
| 10/07/2008 | 19:09:18.080 | 07/01/2007 | 15:53:58.835 | 33 | 2 | 0x00022528 | ...Program Files\ain toolbar\ainToolbarServer.exe |
| 10/21/2008 | 17:09:59.944 | 07/01/2007 | 15:53:58.835 | 37 | 7 | 0x0000c520 | ...Program Files\ADM\ain6.exe |
| 10/08/2007 | 21:50:56.000 | 07/01/2007 | 15:54:00.798 | 38 | 6 | 0x0000a360 | ...Program Files\ADM\ainsoftware.exe |
| 07/01/2007 | 15:51:13.277 | 07/01/2007 | 15:51:13.287 | 35 | 3 | 0x00037928 | ...Program Files\ADM\ainrator.exe |
| 07/01/2007 | 15:51:13.267 | 07/01/2007 | 15:53:58.544 | 36 | 8 | 0x00000036 | ...Program Files\ADM\ainrm.exe |
| 09/26/2008 | 16:40:32.000 | 07/01/2007 | 15:51:12.135 | 34 | 10 | 0x0002e328 | ...Program Files\Common Files\Software Update Utility\dmu.exe |
| 06/30/2007 | 22:57:06.565 | 06/30/2007 | 22:57:06.515 | 19 | 27 | 0x000210e8 | ...Program Files\Google\Common\Google Updater\GoogleUpdaterService.exe |
| 06/30/2007 | 22:57:07.737 | 06/30/2007 | 23:03:16.686 | 20 | 23 | 0x000290e8 | ...Program Files\Google\Google Toolbar\Notifier\1.2.1328.5462\GoogleToolbarNotifier.exe |
| 02/28/2006 | 12:00:00.000 | 06/30/2007 | 22:38:04.133 | 7 | 35 | 0x00034600 | ...Program Files\Internet Explorer\Connection Wizard\wincom1.exe |
| 02/28/2006 | 12:00:00.000 | 06/30/2007 | 22:36:14.405 | 4 | 36 | 0x00011e00 | ...Program Files\Outlook Express\setup50.exe |
| 11/07/2008 | 18:31:38.000 | 07/01/2007 | 15:49:54.734 | 29 | 15 | 0x014a1920 | ...Program Files\Skype\Phone\Skype.exe |
| 11/07/2008 | 18:31:48.000 | 07/04/2007 | 16:10:43.831 | 39 | 5 | 0x00012bc8 | ...Program Files\Skype\Phone\SkypePM.exe |
| 01/04/2007 | 21:30:08.082 | 07/01/2007 | 15:50:59.217 | 31 | 13 | 0x0000004c | ...Program Files\Skype\Phone\SkypeService.exe |
| 09/10/2008 | 16:00:00.000 | 06/30/2007 | 22:57:54.775 | 21 | 26 | 0x00510560 | ...Program Files\WinZip\WINZIP32.EXE |
| 09/10/2008 | 16:00:00.000 | 06/30/2007 | 22:57:54.775 | 22 | 27 | 0x00000000 | ...Program Files\WinZip\WINZIP32.EXE |

wacu - full ver: 0.10; Copyright (c) TZWorks LLC
License is authenticated and registered to Da
run time: 11/21/2013 02:04:57 [UTC]
cmdline: wacu64 -livesys -csv

Windows 7, 64 bit
 Example targets a running system's hive

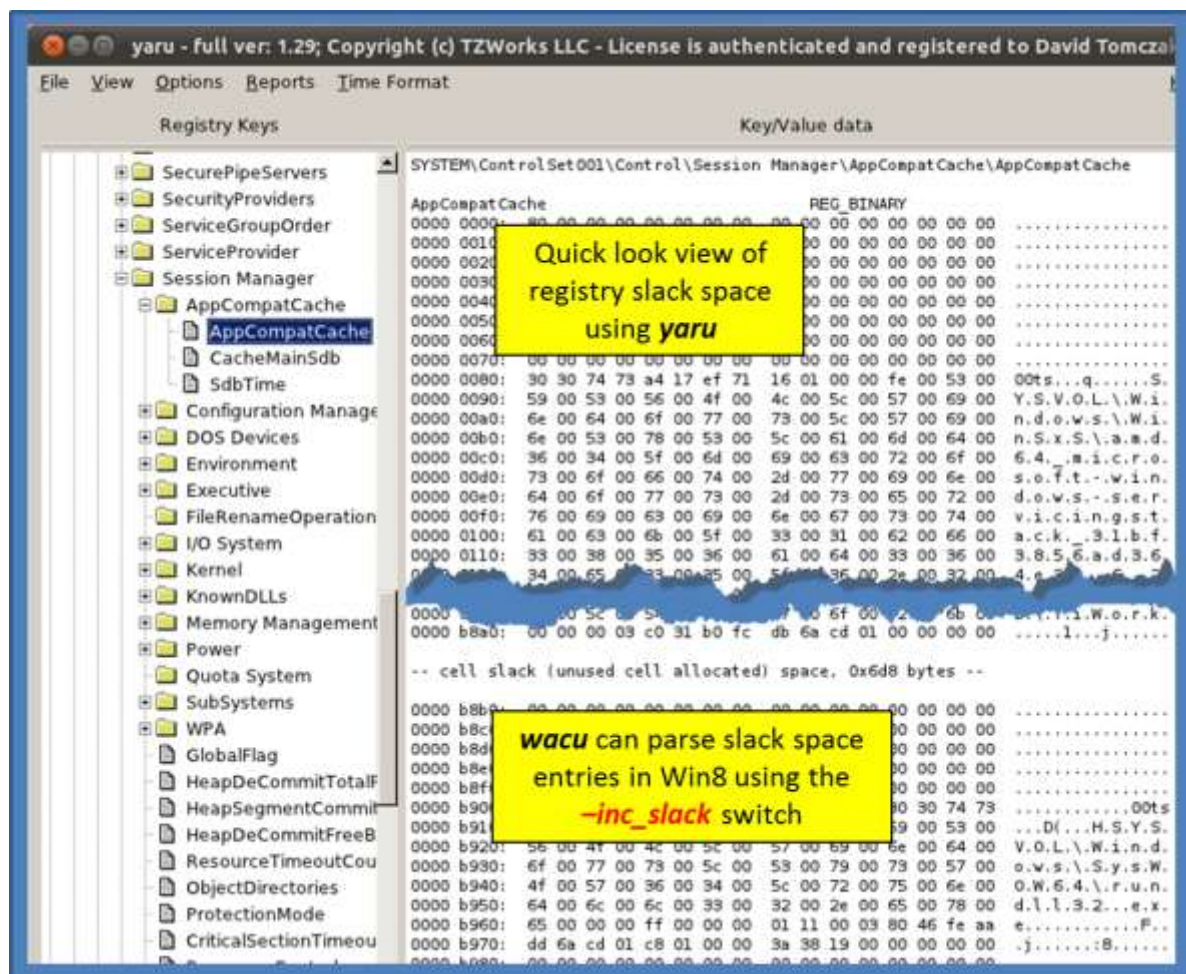
| file mod | time-UTC | entry | flags1 | flags2 | meaning (exper | Path/file |
|------------|--------------|-------|------------|------------|----------------|--|
| 11/20/2010 | 13:25:23.778 | 0 | 0x00000007 | 0x00000000 | exec | \\??C:\Windows\servicing\TrustedInstaller.exe |
| 11/20/2010 | 13:24:52.859 | 1 | 0x00000007 | 0x00000100 | exec | \\??C:\Windows\system32\LogonUI.exe |
| 5/4/2011 | 05:19:28.224 | 2 | 0x00000007 | 0x00000100 | exec | \\??C:\Windows\system32\SearchFilterHost.exe |
| 5/4/2011 | 05:19:28.505 | 3 | 0x00000007 | 0x00000100 | exec | \\??C:\Windows\system32\SearchProtocolHost.exe |
| 11/23/2012 | 03:13:57.304 | 4 | 0x00000007 | 0x00000000 | exec | \\??C:\Windows\system32\taskhost.exe |
| 11/19/2013 | 17:55:38.998 | 5 | 0x00000005 | 0x00000100 | | \\??C:\Windows\System32\ieframe.dll |
| 12/7/2012 | 13:15:31.608 | 6 | 0x00000005 | 0x00000100 | | \\??C:\Windows\System32\gameux.dll |
| 7/14/2009 | 01:39:46.503 | 7 | 0x00000007 | | | C:\Windows\system32\svchost.exe |
| 11/20/2010 | 13:24:26.183 | 8 | 0x00000007 | | | C:\Windows\system32\aitagent.EXE |
| 7/14/2009 | 01:39:02.948 | 9 | 0x00000007 | | | C:\Windows\system32\defrag.exe |
| 7/14/2009 | 01:39:31.433 | 10 | 0x00000007 | 0x00000100 | exec | \\??C:\Windows\system32\rundll32.exe |
| 10/23/2012 | 09:12:55.889 | 11 | 0x00000007 | 0x00000100 | exec | \\??C:\Program Files\Common Files\Microsoft Shar |

Interpretation of flags
 (still experimental)

2.5 Slack Space in the AppCompatibility Cache

Since the size allocated in the system hive for the *AppCompatibility Cache* is rather large compared to the rest of the registry, the potential for large slack space is more probable. Slack space defined here is the left over (unused space) that was allocated for that hive cell. To see this visually, one can use **yaru** v1.29 or above and navigate to the appropriate location. After scrolling towards the end of the data dump, one should see a hex dump for cell slack space. For this particular hive, there was enough slack space to hold 3 additional entries. Below is an example screenshot when looking at a Windows 8 system hive.

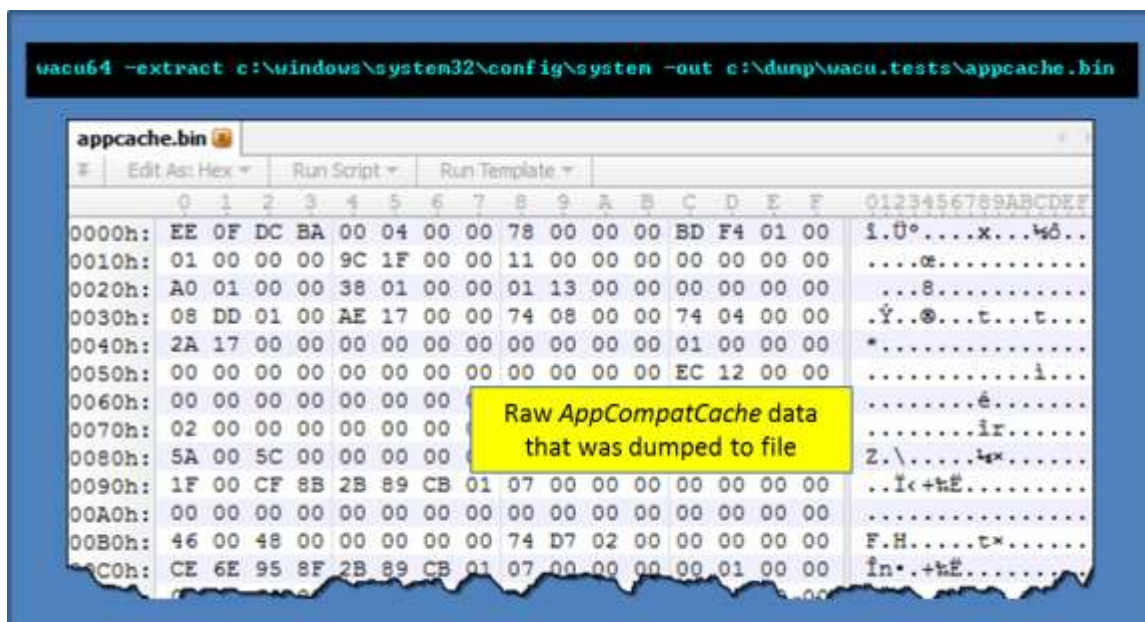
Due to the structure of the *AppCompatibility Cache* on Windows 8 and Server 2012, **wacu** can sparse these entries on a best effort basis, using the **-inc_slack** switch.



2.6 Extracting the Raw AppCompatCache data from the System Hive

For the analyst wanting to perform a more detailed analysis on the raw data, **wacu** has an **-extract** option to pull the raw *AppCompatibility Cache* data from the system hive into a designated file. This allows further analysis using one's own parser, if desired. Below is an example of the command to use to pull the data from the local system hive and dump it to a file. Also shown is the dumped file displayed in a hex editor.

If one has the raw AppCompatCache data, one can then use the **-binary** switch to parse the data into intelligible data.



3 Available Options

Added with version 0.14 is the **-vss <num>** option to point **wacu** to a Volume Shadow copy specified by its index, to pull *AppCompatibility Cache* data from.

| Option | Description |
|-----------------|--|
| -hive | Use this specified target hive to process artifacts from. Syntax is -hive <hive file> |
| -livesys | Targets the active system hive of the running computer. Requires administrative privileges. |
| -binary | Parse the application cache raw data from a file. This assumes one has extracted just the data portion of the application cache from the appropriate registry value and saved it to a file. One can use the -extract option to do this. Syntax is -binary <app compat file> |
| -extract | Extract just the application cache data from the system hive and save the binary data to a file. Syntax is -extract <system hive> -out <file to save data> |
| -image | Extract application compatibility cache artifacts from a volume specified by an image and volume offset. The image is assumed to be in 'dd' (bit-for-bit identical) format to the original volume. The offset specified, should be the volume containing the Windows directory. The syntax is -image <filename> -offset <volume offset> |
| -vmdk | Extract artifacts from a VMWare monolithic NTFS formatted |

| | |
|-----------------------|--|
| | volume. The syntax is -vmdk <disk name> . For a collection of VMWare disks that include snapshots, one can use the following syntax: -vmdk "<disk1> <disk2> ..." |
| -vss | Experimental. Extract artifacts from Volume Shadow. The syntax is -vss <index number of shadow copy> . Only applies to Windows Vista, Win7, Win8 and beyond. Does not apply to Windows XP. |
| -csv | Outputs the data fields delimited by commas. Since filenames can have commas, to ensure the fields are uniquely separated, any commas in the filenames get converted to spaces. |
| -base10 | Ensure all size/address output is displayed in base-10 format vice hexadecimal format. Default is hexadecimal format. |
| -hostname | Option is used to populate the output records with a specified hostname. The syntax is -hostname <name to use> . |
| -inc_slack | Experimental option. If there is any slack in the application cache value in the registry, it will try to parse that data as well. Currently only works with Windows 8. |
| -sort_by_name | Sorts the entries outputted by name vice sequence found in the data. |
| -pipe | Used to pipe system hives into the tool via STDIN (standard input). Each system hive passed in is parsed in sequence. |
| -enumdir | Experimental. Used to process files within a folder and/or subfolders. Each file is parsed in sequence. The syntax is -enumdir <folder> - num_subdirs <#> . |
| -pipe_binary | Used to pipe binary app compat raw files into tool via STDIN (standard input). Each app compat raw file passed in is parsed in sequence. |
| -filter | Filters data passed in via STDIN via the -pipe or -enumdir options. The syntax is -filter <"*.ext *partialname* ..."> . The wildcard character '*' is restricted to either before the name or after the name. |
| -no_whitespace | Used in conjunction with -csv option to remove any whitespace between the field value and the CSV separator. |
| -csv_separator | Used in conjunction with the -csv option to change the CSV separator from the default comma to something else. Syntax is - csv_separator "/" to change the CSV separator to the pipe character. To use the tab as a separator, one can use the -csv_separator "tab" OR - csv_separator "\t" options. |
| -dateformat | Output the date using the specified format. Default behavior is - dateformat "yyyy-mm-dd" . Using this option allows one to adjust the format to mm/dd/yy, dd/mm/yy, etc. The restriction with this option is the forward slash (/) or dash (-) symbol needs to separate month, day and |

| | |
|-------------------------|--|
| | year and the month is in digit (1-12) form versus abbreviated name form. |
| -timeformat | Output the time using the specified format. Default behavior is -timeformat "hh:mm:ss.xxx" One can adjust the format to microseconds, via "hh:mm:ss.xxxxxx" or nanoseconds, via "hh:mm:ss.xxxxxxxxxx" , or no fractional seconds, via "hh:mm:ss" . The restrictions with this option is a colon (:) symbol needs to separate hours, minutes and seconds, a period (.) symbol needs to separate the seconds and fractional seconds, and the repeating symbol 'x' is used to represent number of fractional seconds. (Note: the fractional seconds applies only to those time formats that have the appropriate precision available. The Windows internal filetime has, for example, 100 nsec unit precision available. The DOS time format and the UNIX 'time_t' format, however, have no fractional seconds). Some of the times represented by this tool may use a time format without fractional seconds, and therefore, will not show a greater precision beyond seconds when using this option. |
| -pair_datetime | Output the date/time as 1 field vice 2 for csv option |
| -all_controlsets | Process all ControlSet00x's vice the default, which is <i>CurrentControlSet</i> |
| -utf8_bom | All output is in Unicode UTF-8 format. If desired, one can prefix an UTF-8 byte order mark to the CSV output using this option. |

4 Authentication and the License File

This tool has authentication built into the binary. The primary authentication mechanism is the digital X509 code signing certificate embedded into the binary (Windows and macOS).

The other mechanism is the runtime authentication, which applies to all the versions of the tools (Windows, Linux and macOS). The runtime authentication ensures that the tool has a valid license. The license needs to be in the same directory of the tool for it to authenticate. Furthermore, any modification to the license, either to its name or contents, will invalidate the license.

4.1 Limited versus Demo versus Full in the tool's Output Banner

The tools from *TZWorks* will output header information about the tool's version and whether it is running in *limited*, *demo* or *full* mode. This is directly related to what version of a license the tool authenticates with. The *limited* and *demo* keywords indicates some functionality of the tool is not available, and the *full* keyword indicates all the functionality is available. The lacking functionality in the *limited* or *demo* versions may mean one or all of the following: (a) certain options may not be available, (b) certain data may not be outputted in the parsed results, and (c) the license has a finite lifetime before expiring.

5 References

1. Leveraging the Application Compatibility Cache in Forensic Investigations, by Andrew Davis, Mandiant
2. [http://msdn.microsoft.com/en-us/library/bb432182\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/bb432182(v=vs.85).aspx)
3. <http://www.alex-ionscu.com/?p=41>