

TZWorks® Event Log Parser (*evtwalk*) Users Guide



Abstract

evtwalk is a standalone, command-line tool used to extract records from Event logs from. **evtwalk** can be easily incorporated into any analysts' processing flow via any scripting language. All artifacts can be outputted in one of three parsable formats for easy inclusion with other forensics artifacts. **evtwalk** runs on Windows, Linux and Mac OS-X.

Copyright © TZWorks LLC

www.tzworks.com

Contact Info: info@tzworks.com

Document applies to v0.65 of **evtwalk**

Updated: Apr 25, 2025

Table of Contents

| | | |
|-------|---|----|
| 1 | Introduction | 3 |
| 2 | Event Logs and some Differences between Operating Systems..... | 3 |
| 3 | How to Use <i>evtwalk</i> | 4 |
| 3.1 | Specifying Multiple Individual Event Logs..... | 5 |
| 3.2 | Examining all the Event Logs on the Currently Running Machine | 6 |
| 3.3 | Processing Volume Shadow Copies | 6 |
| 3.4 | Processing all Event Logs in a Specified Partition | 7 |
| 3.5 | Examining Multiple Event Logs in a Directory or Subdirectories..... | 7 |
| 3.6 | Extracting Deleted Records from Slack Space | 8 |
| 4 | Event Category Reports | 8 |
| 4.1 | Password changes | 9 |
| 4.2 | System clock changes..... | 9 |
| 4.3 | Logons | 10 |
| 4.4 | Machine start and stop..... | 10 |
| 4.5 | Credential changes..... | 10 |
| 4.6 | USB Plug-n-play events | 11 |
| 5 | Pulling Statistics | 11 |
| 5.1 | Detailed Statistics for a Specific Event | 12 |
| 6 | User Defined Templates..... | 13 |
| 6.1 | EVT type logs (vice EVTX) when using templates | 15 |
| 6.1.1 | Finding the Definitions of the Parameters for each Event ID. | 15 |
| 6.2 | Using Statistics within Templates | 17 |
| 7 | Converting Segmented CSV formats into Database Friendly Formats | 17 |
| 8 | Creating a Subset EVTX log from a very large EVTX Log | 17 |
| 9 | Forwarding EVTX log Records to a Collector..... | 18 |
| 10 | Known Issues..... | 18 |
| 11 | Available Options | 19 |
| 11.1 | Event Category Report Options | 19 |
| 11.2 | Filtering Options..... | 19 |

| | | |
|------|---|----|
| 11.3 | Miscellaneous Options..... | 19 |
| 12 | Authentication and the License File..... | 22 |
| 12.1 | <i>Limited</i> versus <i>Demo</i> versus <i>Full</i> in the tool's Output Banner..... | 22 |
| 13 | References | 22 |

TZWorks EventLog Parser (*evtwalk*) Users Guide

Copyright © TZWorks LLC

Webpage: http://www.tzworks.com/prototype_page.php?proto_id=25

Contact Information: info@tzworks.com

1 Introduction

evtwalk is a command line tool that can parse Windows event logs from all versions of Windows starting with Windows XP. This includes Vista, Windows 7, Windows 8 and the server counterparts.

The output is presented with one event record per line and includes a couple of formatting options. Under the hood, **evtwalk** uses the same event log parsing engine as **evtx_view** [1] (a GUI tool to analyze event logs). As a command line tool, **evtwalk** can easily be incorporated into any analysts' processing work-flow by automating execution of **evtwalk** via any scripting language.

evtwalk allows one to generate reports of specific event log artifacts, such as USB plug-n-play events, credential changes, password changes, logon/logoff events, etc. If one of the available report options does not address an analyst's needs, there is an option for the user to generate his/her own custom report to be used and processed.

2 Event Logs and some Differences between Operating Systems

Windows event logs reside in different locations depending on whether one is on a Windows XP box, or later version, such as Windows 7 or 8. In addition to the location differences, there are also (a) naming differences in the event log file itself, and (b) significantly more event logs present starting with Vista and the later operating systems. For example, Windows 7 can have over 70 unique event logs versus the three present in Windows XP. Below are the locations for the event logs with the various Windows operating systems.

Window XP and earlier

%windir%\system32\config\[AppEvent.Evt | SecEvent.Evt | SysEvent.Evt]

Windows Vista and later (Windows 7 and Windows 8, ...)

%windir%\system32\winevt\logs\[Application.evtx | Security.evtx | System.evtx | ...]

3 How to Use *evtwalk*

While the ***evtwalk*** tool doesn't require one to run with administrator privileges, without doing so may restrict one to only looking at separately extracted event logs, depending on the version of Windows and how the permissions are setup. Therefore, it is recommended to run ***evtwalk*** with administrator privileges, if desiring to look at the event logs on a live host machine.

One can display the menu options by typing in the executable's name without parameters. A screen shot of the menu is shown below.



```
Administrator: Windows PowerShell

Usage:

evtwalk -log "log1 | log2 | .."    = pull data from extracted logs
evtwalk -livesys                  = pull data from the running OS
evtwalk -vss <num>                 = *** pull data from Volume Shadow
evtwalk -partition <drive letter> = requires volume to be traversable
dir c:\somedir\*.evtx /b /s | evtwalk -pipe
evtwalk -enumdir <folder> -num_subdirs <#> [options]

Report options
-pw                               = pull password changes [security log]
-time                            = pull clock changes or updates [security logs]
-logon                           = pull logons [security log]
-startstop                       = pull system start/stop times [system log]
-creds                           = pull credential changes [security log]
-usb                             = pull usb events [various logs]
-cmdfile <filename>              = *** custom report defined by cmd file

Processing options
-pipe                            = pipe files into evtwalk for processing
-quiet                           = don't display status during run
-filter <*partial*|*.ext>         = ** filters stdin data from -pipe option

Filter options
-eventid "id1, id2, ."           = 
-string <substring>              = 
-start_time <time UTC>           = time in "MM/DD/YYYY HH:MM:SS" format
-stop_time <time UTC>            = time in "MM/DD/YYYY HH:MM:SS" format

Basic options
-csv                             = output in comma separated value format
-csvl2t                           = log2timeline output
-bodyfile [-allparams]           = sleuthkit output

Additional options
-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
-stats <file>                    = *** generate summary statistics
-inc_slack                       = *** incl any records from slack as well

Create subset evtx logs utility options [targeting certain entries ***]
-createlog <result log> -log <srclog> -eventid "id1, id2, ..."
-createlog <result log> -log <srclog> -rec_id "rec1, rec2, ..."
-createlog <result log> -log <srclog> -rec_start <rec1> -rec_stop <recN>
```

For basic usage and to parse an individual *event log* file, use the following notation:

evtwalk -log <event log file> > results.txt

Without specifying one of the format options, the output is rendered with a custom CSV format that uses the pipe character ('|') as a delimiter versus a comma. The snapshot below is an example of what this output looks like. Notice that all similar event IDs are grouped together. This allows each grouping to have their specific unique headers (if applicable), since different events have different metadata.

| Record# | EventID | Date | Time-UTC | Level | ActivityID | Channel | Computer |
|---------|---------|------------|--------------|-------|------------|----------|----------|
| 292 | 1100 | 10/20/2012 | 18:48:34.990 | info | 0 | Security | win8tes |
| 31 | 1100 | 10/20/2012 | 21:37:30.136 | info | 0 | Security | windows |
| 1019 | 1100 | 07/10/2013 | 17:00:41.351 | info | 0 | Security | win8tes |
| 1274 | 1100 | 07/10/2013 | 16:56:56.582 | info | 0 | Security | win8tes |
| 577 | 1101 | 07/10/2013 | 16:56:56.582 | info | 0 | Security | win8tes |
| 293 | 4608 | 10/20/2012 | 18:49:12.487 | info | 0 | Security | win8tes |
| 1 | 4608 | 10/20/2012 | 21:34:55.146 | info | 0 | Security | windows |
| 32 | 4608 | 10/20/2012 | 21:38:33.760 | info | 0 | Security | win8tes |
| 578 | 4608 | 10/20/2012 | 21:38:33.760 | info | 0 | Security | win8tes |
| 1020 | 4608 | 10/20/2012 | 21:38:33.760 | info | 0 | Security | win8tes |
| 93 | 4616 | 10/20/2012 | 18:40:57.171 | info | 0 | Security | win8tes |
| 94 | 4616 | 10/20/2012 | 18:40:57.196 | info | 0 | Security | win8tes |
| 109 | 4616 | 10/20/2012 | 18:42:57.354 | info | 0 | Security | win8tes |

In the command used above, the output is redirected to a text file called 'results.txt'. Like all artifacts that have many records, and where each record has multiple fields, the output that is generated is usually very long and wide. Thus, it is recommended that one redirect the output of the command to a file.

Besides the default CSV output, one can render the output in two other formats. Switches for these other options are: (a) **-csv12t** and (b) **-bodyfile**. Each respective format option will attempt to conform to either the **log2timeline** format, or the **SleuthKit's body-file** format, as appropriate.

While parsing one event log file is useful, one will usually want to parse multiple event logs in one session. There are three ways to do this: (a) specifying individual event logs via the **-log** option where each log filename is delimited by a pipe character, (b) using the **-livesys** option, or (c) using the **-pipe** option.

3.1 Specifying Multiple Individual Event Logs

To use the **-log <event log file>** option to specify multiple event logs, use the pipe delimiter between each event log name, as shown below.

evtwalk -log "<event log1> | <event log2> | ..." > results.txt

This is useful when pulling a similar category of artifacts from multiple event logs. A good example of this is pulling USB events. The two logs needed for USB plug-n-play events are the *System* event log and *DriverFrameworks-UserMode* event log. If one extracts these two logs, one can invoke the following, rather lengthy, command to process all USB events from the two logs:

```
evtwalk -usb -log "system.evtx /  
Microsoft-Windows-DriverFrameworks-UserMode%4Operational.evtx" > results.txt
```

The *results.txt* file will contain a sorted set of groups of all 'like USB' event IDs and will provide appropriate header fields that match the record metadata for each class of event.

3.2 Examining all the Event Logs on the Currently Running Machine

For a live system, one can use the **-livesys** switch to examine all the event logs on a host machine. In this mode, **evtwalk** will determine the Windows version of the host machine, and then will scan the appropriate event log directory for that version of Windows. Below are some examples:

```
evtwalk -livesys > results.txt  
evtwalk -livesys -string "tzworks" > results.txt
```

The first example will traverse all event log files found in the Windows event log directory and parse each record for each event log encountered. The second example adds the **-string** filter option. It will also examine all the same event logs in the first example, but will only output records that contain the string "tzworks" in the one of the record fields. More information about the various filter options are discussed in a later section.

3.3 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.

To make it easier with the syntax, we've built in some 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 user hive located in the testuser account from the HarddiskVolumeShadowCopy1, the following syntax can be used:

```
evtwalk -log %vss%1\Windows\System32\winevt\logs\System.evtx > results.txt
```


In addition, one can process all the user related hives using the command **-vss <index of volume shadow>**. This option will traverse the specified volume shadow copy and look for all the event logs that are available and process them.

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 *HarddiskVolumeShadowCopy*. It is this number that is passed as an argument to the previous options.

3.4 Processing all Event Logs in a Specified Partition

One can process all the event logs on a specified volume using the **-partition <drive letter>** option. This command will look in the normal event log directory to find which logs are available and then proceed to process those logs. It is useful if mounting a collected image of a system volume as another drive letter.

3.5 Examining Multiple Event Logs in a Directory or Subdirectories

If looking at a collection of event logs that are not part of the running operating system, but gathered as part of an investigation, one can invoke the **-pipe** switch to analyze all the desired event logs in one session. The **-pipe** switch tells **evtwalk** to receive a separate path/filename per line as input and process each entry separately. By redirecting the output of the processed records to a file, one can generate a single report for all the event logs piped in.

Depending on whether one is running on Windows or Linux during the piping operation, the syntax is different. For Windows, one can use the built-in **dir** command along with some of its companion

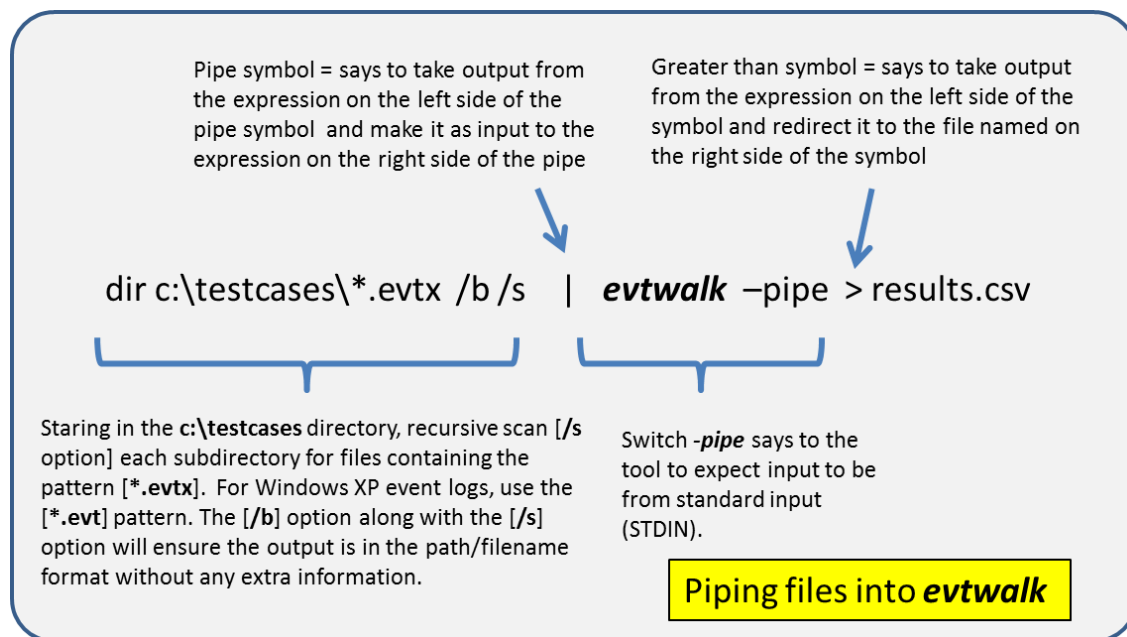
switches to get the desired result. For Linux or Mac, one can use either the built-in **ls** or **find** commands to get the desired result. Below are examples of using the pipe option:

```
dir c:\testcases\*.evtx /b /s | evtwalk -pipe > results.txt
```

```
ls -1 ~/testcases/*.evtx | ./evtwalk -pipe > results.txt
```

The above syntax will process all the *event log* files with the extension .evtx that are located anywhere in the c:\testcases directory and subdirectories.

For those not familiar with syntax that uses a **pipe** or the **dir** command line options, the figure below provides annotations to what each portion in the command is doing.



If one cannot use the **-pipe** option, one can use the experimental **-enumdir** option, which has similar functionality with more control. The **-enumdir** option takes as its parameter the folder to start with. It also allows one to specify the number of subdirectories to evaluate using the **-num_subdirs <#>** sub-option.

3.6 Extracting Deleted Records from Slack Space

For the EVT-X type log files version 0.51 adds the experimental option to examine slack space in the log sections. The option for this is **-incl_slack**. Since this option is experimental, it has not been fully exercised against all types of corrupted records. The output will be annotated with a couple of new fields, identifying which records were found in slack space and the offset the record was found at. The offset will allow the analyst to verify the record manually if desired.

4 Event Category Reports

Instead of outputting all the records contained in an event log, one may only be interested in a certain class of event data. Depending on how the host machine was configured, for event records to be present, one may need to enable the event logging for a desired class of events.

Below are the report categories currently available for this tool:

- Password changes
- Clock changes or updates
- User logon/logoff events
- System start/stop times
- User credential or permission changes
- USB events

If there are other reports an analyst wants to use that are not in the above list, or if one wishes to make modifications to the reports above, one can define one's own report via the **-cmdfile <path\file>** option. The argument passed in is a user generated text file that identifies which events to pull and which fields in the event record to output. These command files are called *User Defined Templates* and are discussed in a later section.

4.1 Password changes

The following Event IDs are examined for this category:

| Event Description | WinXP event ID | Win7/8 event ID | Log type |
|---|----------------|-----------------|--------------|
| A notification was loaded, a user changed his/her password | 518 | 4614 | Security log |
| Change Password Attempt | 627 | 4723 | Security log |
| User Account Password Reset | 628 | 4724 | Security log |
| A user account was changed | 642 | 4738 | Security log |

4.2 System clock changes

The following Event IDs are examined for this category:

| Event Description | WinXP event ID | Win7/8 event ID | Log type |
|---|----------------|-----------------|--------------|
| The System Time was Changed | 520 | 4616 | Security log |
| Service attempted to change Time | 577 | 4673 | Security log |
| Service changed Time | 578 | 4674 | Security log |

4.3 Logons

This report pulls events identifying which accounts have been used for attempted logons. Information such as date, time, username, hostname and success or failure can be extracted. The event IDs that are extracted are:

| Event Description | Win XP Event ID | Win 7/8 Event ID | Log type |
|---------------------------------|-----------------|------------------|--------------|
| Successful logon and type logon | 528, 539, 540 | 4624 | Security log |
| Failed logon | 529-537 | 4625 | Security log |
| Logoff | 538 | 4634 | Security log |
| Logon/RunAs | 552 | 4648 | Security log |

4.4 Machine start and stop

This report includes items such as when the computer started up, went to sleep, resumed, rebooted or shutdown

| Event Description | Win XP Event ID | Win 7/8 Event ID | Log type |
|-------------------|-----------------|------------------|------------|
| Reboot | 528 | 4624 | System log |
| Startup | 12 | 12 | System log |
| Shutdown | 13 | 13 | System log |
| Sleep | 42 | 42 | System log |
| Resume | 1 | 1 | System log |

4.5 Credential changes

The following Event IDs are examined for this category:

| Event Description | Win XP Event ID | Win 7/8 Event ID | Log type |
|--|-----------------|------------------|--------------|
| Special Privileges assigned to new logon | 576 | 4672 | Security log |
| User Right was assigned | 608 | 4704 | Security log |
| User Right was removed | 609 | 4705 | Security log |
| System Security Access was granted to an account | 621 | 4717 | Security log |
| System Security Access was removed from an account | 622 | 4718 | Security log |
| User Account was created | 624 | 4720 | Security log |
| User Account was enabled | 626 | 4722 | Security log |
| User Account was disabled | 629 | 4725 | Security log |
| User Account was deleted | 630 | 4726 | Security log |
| User Account was changed | 642 | 4738 | Security log |
| User Account was locked out | 644 | 4740 | Security log |
| Computer Account was created | 645 | 4741 | Security log |
| Computer Account was changed | 646 | 4742 | Security log |
| Computer Account was deleted | 647 | 4743 | Security log |

| | | | |
|---|-----|------|--------------|
| User Account was unlocked | 671 | 4767 | Security log |
| Domain Controller attempted to validate the credentials for an account | | 4776 | Security log |
| Domain Controller failed to validate the credentials for an account | 675 | 4777 | Security log |
| Name of an Account was changed | 685 | 4781 | Security log |

4.6 USB Plug-n-play events

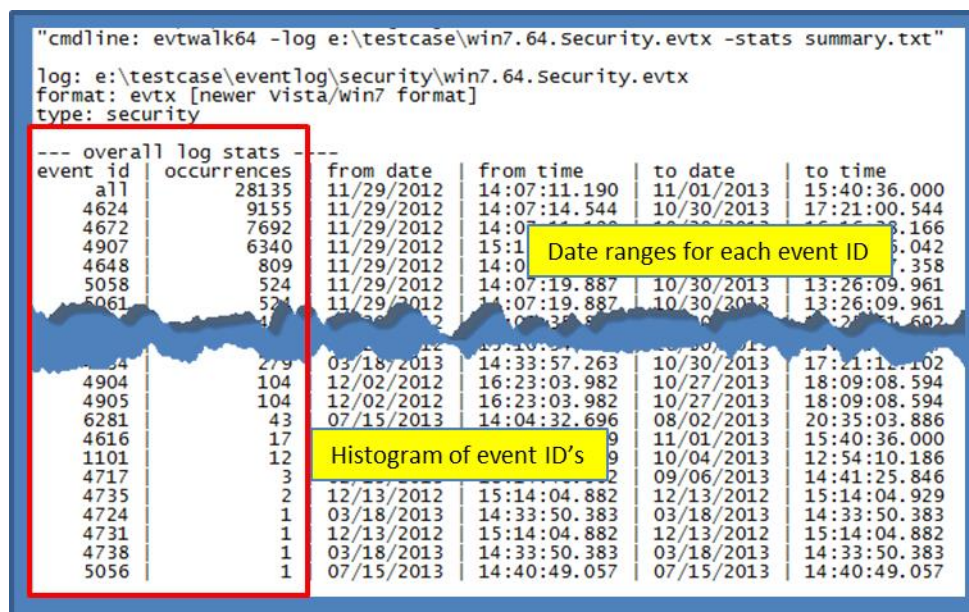
For USB events, one needs to examine the following logs: *System*, *DriverFrameworks-Usermode*, *Kernel PnP* and *Partition Diagnostics*. The following Event IDs are examined for this category:

| Event Description | Win 7+ Event ID | Log type |
|---|------------------------------|-------------------------------|
| Driver installed | 20001 | System log |
| Driver removed | 20002 | System log |
| Service added | 20003 | System log |
| Service removed | 20004 | System log |
| Device removed | 20007 | System log |
| Query to load Drivers | 2003 | DriverFrameworks-Usermode log |
| Loading Drivers for new Device | 2004, 2005 | DriverFrameworks-Usermode log |
| Successfully Loaded Driver | 2006, 2010 | DriverFrameworks-Usermode log |
| Pnp or Power Operation for a USB Device | 2100, 2101, 2102, 2105, 2106 | DriverFrameworks-Usermode log |
| Error: for Pnp or Power operation for device | 2103 | DriverFrameworks-Usermode log |
| USB Power Events | 2104, 2107, 2108, 2109 | DriverFrameworks-Usermode log |
| Device started | 410 | Kernel PnP log |
| Device deleted | 420 | Kernel PnP log |
| Inserted/Removed | 1006 | Partition Diags log |

5 Pulling Statistics

One can pull statistics in the form of a histogram for an event log. This can be done in parallel during the normal parse and pulling each event's data. The option to invoke this is **-stats** <results.txt>. The results.txt is will contain only the statistics and the normal parsed event data still outputs via the standard output which can be redirected to its own file..

The statistics report breaks out each event ID as far as, the number of occurrences, the minimum timestamp and the maximum timestamp. This resulting histogram is displayed in descending order based on the number of occurrences. Below is an example of the output.



5.1 Detailed Statistics for a Specific Event

While the **-stats** option gives overall statistics for the entire event log, perhaps one may desire to pull statistics on a specific event ID. This second option still relies on the **-stats** keyword to identify where to put the output, but the user also identifies which fields, or combination thereof, one would like to histogram. This is done with the **-hist <fields to extract>** option. For example, if one was interested in the looking at the logon activity associated with Event ID 4624, one could histogram the following fields (LogonType, TargetDomainName, TargetUserName, etc). Just for example purposes, one could see the various combinations and frequency of occurrences for the 3 fields, by prepending the following to the normal **evtwalk** command:

-eventid 4624 -stats "summary.txt" -hist "LogonType|TargetDomainName|TargetUserName"

In addition to the normal eventlog overall statistics, invoked by the **-stats** option, the **-hist** option will also look for unique combinations of the 3 fields specified as applied to event ID 4624. See the sample output below:

| | | | | | | | | | |
|--|-------------|------------|-----------|------------|----------|---------|----------------------------------|---|--|
| cmdline: evtwalk64 -log e:\testcase\win7.64.Security.evtx -stats summary.txt -eventid 4624 -hist LogonType TargetDomainName TargetUserName | | | | | | | | | |
| --- filter stats --- [refer to filter# for details] | | | | | | | | | |
| event id | occurrence# | from date | from time | to date | to time | filter# | key values | Unique combinations of requested fields | |
| 4624 | 6290 | 11/29/2012 | 14:07:14 | 10/30/2013 | 16:16:38 | 1 | 5; NT AUTHORITY; SYSTEM | LogonType; TargetDomainName; TargetUserName | |
| 4624 | 835 | 11/29/2012 | 14:07:25 | 10/30/2013 | 13:26:09 | 1 | 2; KLKJDF-PC; tzlabs | LogonType; TargetDomainName; TargetUserName | |
| 4624 | 621 | 11/29/2012 | 14:07:21 | 10/30/2013 | 17:21:00 | 1 | 3; NT AUTHORITY; ANONYMOUS LOGON | LogonType; TargetDomainName; TargetUserName | |
| 4624 | 426 | 11/29/2012 | 18:02:38 | 10/30/2013 | 13:25:51 | 1 | 0; NT AUTHORITY; SYSTEM | LogonType; TargetDomainName; TargetUserName | |
| 4624 | 426 | 11/29/2012 | 18:02:38 | 10/30/2013 | 13:25:53 | 1 | 5; NT AUTHORITY; LOCAL SERVICE | LogonType; TargetDomainName; TargetUserName | |
| 4624 | 426 | 11/29/2012 | 18:02:37 | 10/30/2013 | 13:25:52 | 1 | 5; NT AUTHORITY; NETWORK SERVICE | LogonType; TargetDomainName; TargetUserName | |
| 4624 | 80 | 03/18/2013 | 14:35:08 | 04/29/2013 | 14:11:47 | 1 | 9; KLKJDF-PC; TempUser | LogonType; TargetDomainName; TargetUserName | |
| 4624 | 51 | 11/29/2012 | 14:09:24 | 04/29/2013 | 13:40:15 | 1 | 5; KLKJDF-PC; TempUser | LogonType; TargetDomainName; TargetUserName | |

These types of statistics can be applied across event ID's assuming the fields to examine are present in each of the event ID's. For example, one can extend this example across the event ID's 4624, 4634, 4648, by eliminating the LogonType from the *-hist* fields (since LogonType is not present in the Event ID 4648).

| | | | | | | | | | |
|--|-------------|------------|-----------|------------|----------|---------|-------------------------------|----------------------------------|--|
| cmdline: evtwalk64 -log e:\testcase\win7.64.Security.evtx -stats summary.txt -eventid '4624 4634 4648' -hist TargetDomainName TargetUserName | | | | | | | | | |
| --- filter stats --- [refer to filter# for details] | | | | | | | | | |
| event id | occurrence# | from date | from time | to date | to time | filter# | key values | key fields | |
| 4624 | 6716 | 11/29/2012 | 14:07:14 | 10/30/2013 | 16:16:38 | 1 | NT AUTHORITY; SYSTEM | TargetDomainName; TargetUserName | |
| 4624 | 835 | 11/29/2012 | 14:07:25 | 10/30/2013 | 13:26:09 | 1 | KLKJDF-PC; tzlabs | TargetDomainName; TargetUserName | |
| 4624 | 621 | 11/29/2012 | 14:07:21 | 10/30/2013 | 17:21:00 | 1 | NT AUTHORITY; ANONYMOUS LOGON | TargetDomainName; TargetUserName | |
| 4624 | 426 | 11/29/2012 | 18:02:38 | 10/30/2013 | 13:25:53 | 1 | NT AUTHORITY; LOCAL SERVICE | TargetDomainName; TargetUserName | |
| 4624 | 426 | 11/29/2012 | 18:02:37 | 10/30/2013 | 13:25:52 | 1 | NT AUTHORITY; NETWORK SERVICE | TargetDomainName; TargetUserName | |
| 4624 | 131 | 11/29/2012 | 14:09:24 | 04/29/2013 | 14:11:47 | 1 | KLKJDF-PC; TempUser | TargetDomainName; TargetUserName | |
| 4634 | 198 | 08/22/2013 | 15:09:50 | 10/30/2013 | 17:21:12 | 2 | NT AUTHORITY; ANONYMOUS LOGON | TargetDomainName; TargetUserName | |
| 4634 | 81 | 03/18/2013 | 14:33:57 | 04/29/2013 | 14:11:48 | 2 | KLKJDF-PC; TempUser | TargetDomainName; TargetUserName | |
| 4648 | 418 | 11/29/2012 | 14:07:25 | 10/30/2013 | 13:26:09 | 3 | KLKJDF-PC; tzlabs | TargetDomainName; TargetUserName | |
| 4648 | 338 | 12/11/2012 | 15:44:39 | 10/30/2013 | 15:32:37 | 3 | KLKJDF-PC; davet | TargetDomainName; TargetUserName | |
| 4648 | 51 | 11/29/2012 | 14:09:24 | 04/29/2013 | 13:40:15 | 3 | KLKJDF-PC; TempUser | TargetDomainName; TargetUserName | |
| 4648 | 2 | 03/12/2013 | 20:11:00 | 03/12/2013 | 20:11:03 | 3 | KLKJDF-PC; tzworks | TargetDomainName; TargetUserName | |

6 User Defined Templates

For those cases, where one would like to extract a certain group of event ID's, templates can be useful. Once a template is defined, it can be used to ensure repeatable parsing of the same event ID's for each session run.

The templates are just text files, so they can be generated with any text editor. Care must be taken to ensure that extra control characters are not inserted into the template files. Having extra control characters will negatively affect the template parsing engine. For this reason, it is recommended that a simple text editor be used when editing a template file.

The parsing rules for these templates are as follows:

1. General Rules
 - a. Each line is parsed separately.
 - b. A line that starts with a double forward slash (eg. //) is ignored and used for comments

- c. A blank line is ignored
 - d. Any line not satisfying rule (1b) and (1c) above is assumed to be a command
 - e. All command lines are in CSV format, where the separator is a comma.
2. Command Lines

Must start with the sequence: **!cmd**, and the entire command must be on one line.

- a. The command sequence can contain the following options, using comma delimiters (in any order):

-enum_evtlog or **-enum_evtxlog**

-id, <event id to extract>

-name, <event name to use for output>

-conditions, <parameter name | value name> [note: parameter name is 'string' for old EVT logs]

-pull, <parameter data to extract> [note: syntax for EVTX type logs]

-all_data

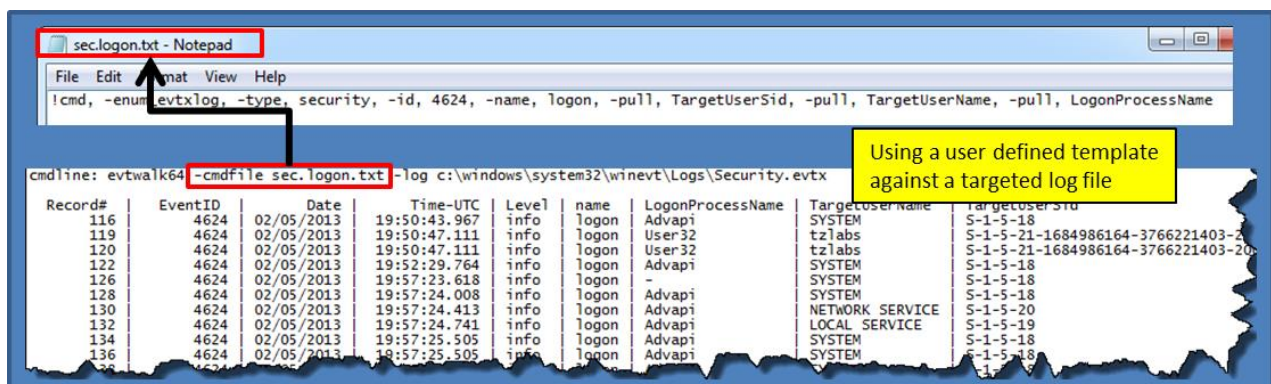
-type [system, security, application, ...]

One uses either the **-enum_evtlog** or **-enum_evtxlog**, but not both. The former specifies the target log is a Vista or Win7 (or later) log, while the latter specifies the target log is Window XP. Below are two examples (note: each command needs to be on one line, as opposed to broken into multiple lines as shown below):

```
!cmd, -enum_evtlog, -type, security, -id, 4624, -name, logon, -pull,
TargetUserSid, -pull, TargetUserName, -pull, LogonProcessName
```

```
!cmd, -enum_evtxlog, -type, system, -id, 528, -name, reboot or shutdown,
-conditions, string|winlogon.exe -all_data
```

The first example targets a Vista or Win7 log and looks for event ID 4624. The annotation of **-name, logon**, specifies to label this event as a logon event. The **-pull, TargetUserSid, ..., -pull, LogonProcessName**, tells the tool to only extract those fields from records with the event ID 4624 in the output.



The second example targets a Windows XP log and looks for event ID 528. The label annotated to this event is “reboot or shutdown”. The condition option says to only look at the *Data* field and if it has the value of “winlogon.exe” to count it as a hit. The **-all_data** option says to extract all the fields of this record in the output.

When using templates to parse event logs, one needs to be careful to use the appropriate template for type of log file. In other words, searching for a specific event ID in a system log file means one thing, and searching for the same event ID in the security log file is something different. Therefore, one should **not** blindly specify *event ID's* in a template without specifying the type of log file. For example, if targeting the *event ID# 4625*, for the *Security* log, this would translate to a *logon/failed* event. However, if looking at the *Application* log, *event ID# 4625* is the suppression of duplicate log entries. To help avoid this issue, the option **-type [system, security, application, etc]** option guides *evtwalk* to match the proper log file with the event ID specified.

The built in reports *category*, fortunately, have the necessary logic to avoid the above issue. Therefore, they can be safely thrown at many disparate log files (that may contain duplicate event ID numbers with differing meanings) and the results should be accurate.

6.1 EVT type logs (vice EVTX) when using templates

When targeting Window XP or Win2003 server type event logs (eg. EVT type logs), the parameter fields are not tagged per se, but are indicated by their position and are separated by null terminated strings. The only context one has is the position of the string relative to the start of the string array. This position is used as a parameter placement for templates stored in the resource section of certain system DLLs.

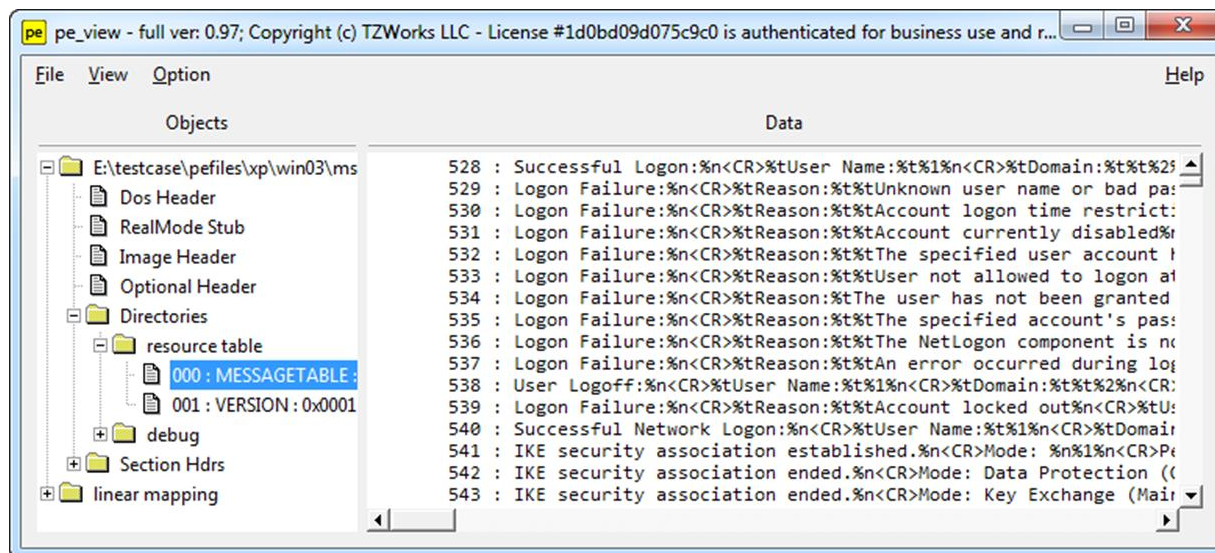
When parsing these types of logs, the output shows the headings param_01, param_02, etc. These heading are just dummy names to indicate which values go to which position. Therefore, when using the **-pull** keyword in the context of EVT logs, one can indicate the position of the field as well as the desired name to use as the header in the following syntax: **-pull, <position index> | <name to use>**.

Furthermore, when using the **-conditions** keyword, one uses it conjunction with the keyword **string**, in the following manner: **-conditions, string | <partial name that needs to be present>**. This tells the parser to look for the partial name in any of the positions that are populated, and if found, to extract that event as part of the results.

6.1.1 Finding the Definitions of the Parameters for each Event ID.

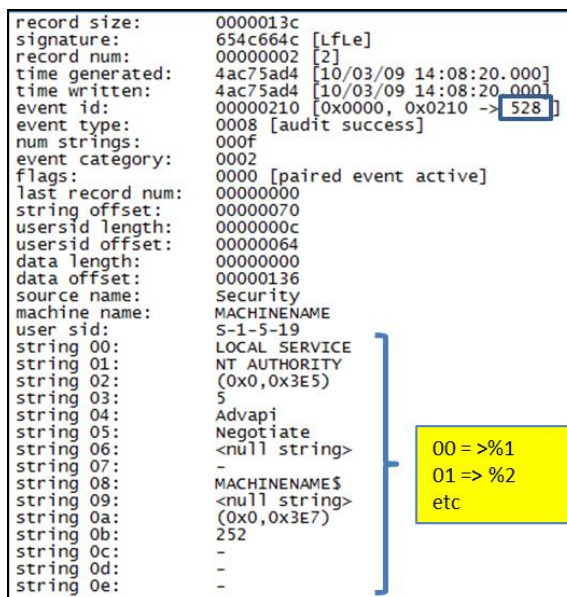
There are a number of online sources to find what each parameter is for a specific event ID. Some sources are better than others. The purpose of this section is to show you how to derive it yourself if you have access to the system directory of where the event log came from. If one examines the system registry hive, specifically in the root path: **SYSTEM\CurrentControlSet\Services\Eventlog**, it identifies various event logs. Within each event log path, there are values that are identified as EventMessageFile, CategoryMessageFile, ParameterMessageFile, etc. that point to system DLLs that have the appropriate

resource section. For this example, I picked a more common one for the security log, which is MsAuditE.dll. When looking at the resource section of this DLL, one would see the following:



Highlighted is the MESSAGETABLE in the resource section. Listed on the right are the various event ID's and their associated meaning along with the mapping of where the event string position goes to. For event ID 528 above, %1 = string at position 1, %2 = string at position 2, etc. Further, one can see what each field should be called, %1 is for "User Name", %2 is for "Domain", etc.

Looking at a raw parse of event ID 508 from *evtx_view*, one can see its structure below:



Finally, to use the syntax *-pull* for this log and substitute the positional parameter with a header, one could do the following to extract only the Username, Domain, LogonID and LogonType, assuming the Winlogon.exe was used:

```
!cmd, -enum_evtlog, -type, system, -id, 528, -name, reboot or shutdown,  
-conditions, string|winlogon.exe -pull, 1|Username,  
-pull, 2|Domain, -pull, 3|LogonID, -pull, 4|LogonType
```

6.2 Using Statistics within Templates

A template can have many filters to pull various combinations of data into one report. To supplement your report, one can also pull statistics per event by using the **-hist** keyword. The usage is similar to the command line syntax in that after the **-hist** keyword, one specifies the field or combination of fields one wishes to histogram. If using more than one field, then one delimits each of the fields with a pipe character, just like when using it via the command line. The nuance for templates, however, is one also needs to **-pull** the parameter that is used as part of the **-hist** sequence. Alternatively, one can forgo using the **-pull** option and just use the **-all_data** keyword, which is shorthand to pull all the fields for that record. Finally, one needs to specify the **-stats <filename>** option as part of the command line that invokes the template via the **-cmdfile <template file>**. This is required to set up **evtwalk** into the statistics gathering mode and identify the results file to put the data in.

7 Converting Segmented CSV formats into Database Friendly Formats

When running **evtwalk** to pull differing events from an event log into one results file, the CSV output will vary depending on the event ID that is processed. While the **-bodyfile** and **-csv/2t** formats will preserve the CSV structure, the default CSV output will show the results as segmented CSV sections. Each CSV section will represent a different event ID. This can create problems when trying to import the **evtwalk** results into other databases for analysis.

To solve this problem, one can use the **csvdx** tool to take the segmented CSV results (or any CSV results) and convert the artifact output it into either JSON or SQLite. See the **csvdx** webpage (https://tzworks.com/prototype_page.php?proto_id=34) and/or user guide (<https://tzworks.com/prototypes/csvdx/csvdx.users.guide.pdf>).

8 Creating a Subset EVT log from a very large EVT log

The option **-create_log** was initially added to the Windows binaries to assist in the analysis of problem records within a large log. There are two basic themes with this option: (a) one can either target specific event identifiers (**-eventid "id1, id2 ..."**), or (b) target specific records numbers (**-rec_id "rec1, rec2 ..."**) or (**-rec_start <rec1> -rec_stop <recN>**). In either case, **evtwalk** will examine the source EVT log and extract those records specified and create a separate EVT log with the results. The resulting EVT log should be compatible with any EVT viewer or parser.

Below is a simple example of creating an EVT log with just the event ID of 4624, which identifies some account was logged into. All the events with ID 4624 were copied from the originating log

[win10.Security.evtx] , and the a new log was created called evt4624.evtx. The original log was not modified.

```
evtwalk64 -create log evt4624.evtx -log win10.Security.evtx -eventid "4624"
```

9 Forwarding EVTX log Records to a Collector

Normally when a binary XML event log record is created and stored in the log, the use of templates are used to help save on log space. From all the local logs created on a computer, the data suggests this is the normal behavior. These templates identify the mapping of which data in the binary XML is used for variables for message table data, amongst other things. So when a record doesn't make use of a template, and also consists of variables embedded into the record, it can be problematic recreating the event log message.

For those records that are created on a client computer and then forwarded to another computer acting as a collector, there appears to be a shift in the above behavior. For these cases, empirical data suggests that the binary XML records created in these logs do not use templates. This presents some interesting problems, as stated earlier. For these forwarded records, we modified the *evtwalk* parsing engine to use some heuristics to identify these types of records and the proper mapping of variables to *message table* entries. In addition, a couple of new fields needed to be added to allow for two sets of timestamps and record numbers: (a) the first set was for the timestamp that the client created the event, and the timestamp the collector logged the record, and (b) the second set was for the record number for the client created for the event, and a record number the collector used to log the event.

10 Known Issues

For CSV (comma separated values) output, there are restrictions in the characters that are outputted. Since commas are used as a separator for *-csv/2t*, any data containing commas are replaced with a space. For the default output no changes are made to the data, since a pipe (|) character is used to delimit the fields of a record.

When using user defined templates, one needs to make sure that all text in the template does not have any special characters. The most common issue we hear about is the use of dashes preceding the keywords. If one uses notepad or some other text editor that does not insert special characters, then it should work fine. However, if using Microsoft Word to create a template file, the dashes inserted are not ASCII 0x2D encoded dashes, but some other character that looks like a dash.

11 Available Options

11.1 Event Category Report Options

| Option | Description |
|-------------------|---|
| -pw | Option to extract the appropriate event ID's for password changes. |
| -time | Option to extract the appropriate event ID's for clock changes or updates. |
| -logon | Option to extract the appropriate event ID's for user logon/logoff changes. |
| -startstop | Option to extract the appropriate event ID's for system start/stop times. |
| -creds | Option to extract the appropriate event ID's for user credential or permission changes. |
| -usb | Option to extract the appropriate event ID's for USB insertions and removals. |
| -cmdfile | Experimental. Option that allows the user to customize which event ID's to extract as well as which fields within an event record to output. The syntax is -cmdfile <filename> . |

11.2 Filtering Options

| Option | Description |
|--------------------|---|
| -eventid | Filter out the specified event ID. If more than one ID is specified, one needs to delimit each ID with a comma. The syntax is -eventid "id1, id2, ..." . |
| -string | Filter on the specified string. The string comparison engine is case insensitive. The syntax is -string "substring to target" . |
| -start_time | Filter events occurring at the specified time or later. The syntax is -start_time "time in UTC" . The time is specified in the following format MM/DD/YYYY or "MM/DD/YYYY HH:MM:SS". |
| -stop_time | Filter events occurring at the specified time or before. The syntax is -stop_time "time in UTC" . The time is specified in the following format MM/DD/YYYY or "MM/DD/YYYY HH:MM:SS". |

11.3 Miscellaneous Options

| Option | Description |
|-------------|---|
| -log | Identify which event log(s) to operate on. The syntax is: -log <eventlog to |

| | |
|-------------------|---|
| | analyze >. To operate more than one at a time, use: -log "<eventlog1> <eventlog2> ..." |
| -livesys | Option to examine all event logs from the running operating system. |
| -partition | Option to examine all event logs from an external drive (or 'dd' image) that were mounted on an analysis workstation. Syntax is -partition <drive letter> |
| -vss | Experimental. Option to analyze all events logs from a Volume Shadow. Syntax is -vss <index of volume shadow> . Only applies to Windows Vista, Win7, Win8 and beyond. Does not apply to Windows XP. |
| -stats | Experimental. Extract statistics from the event log, including time range of the records in the log as well as a histogram of the event IDs in the log. The syntax is -stats <file to store stats> . |
| -hist | Experimental. Extract statistics from certain events based on a set of specified fields. This option requires the use of the -stats option. The syntax is -stats <file to store stats> -hist <field1 field2 ..> . |
| -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. |
| -csvl2t | Outputs the data fields in accordance with the log2timeline format. |
| -bodyfile | Outputs the data fields in accordance with the 'body-file' version3 specified in the SleuthKit. The date/timestamp outputted to the body-file is in terms of UTC. So if using the body-file in conjunction with the mactime.pl utility, one needs to set the environment variable TZ=UTC. The -allparams option can be used in conjunction with the -bodyfile and tells evtwalk to output all the fields in each record. |
| -pipe | Used to pipe files into the tool via STDIN (standard input). Each file 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 <#> . |
| -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 are: 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 |
| -quiet | This option suppresses any intermediate progress during a session run |
| -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. |

12 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.

12.1 *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.

13 References

1. Introducing the Microsoft Vista event log format, by Andreas Schuster, 2007
2. Wikipedia, the free encyclopedia. [Event Viewer topic](#)
3. [TechNet, New Tools for Event Management in Windows Vista](#)
4. [Randy Franklin Smith's online encyclopedia.](#)
5. Windows Event Log Viewer, *evtx_view*, http://tzworks.com/prototype_page.php?proto_id=4
6. *SleuthKit* [Body-file](#) format, <http://wiki.sleuthkit.org>
7. Log2timeline CSV format, <http://log2timeline.net/>