

TZWorks® Windows Event Log Viewer (*evtx_view*) Users Guide



Abstract

evtx_view is a standalone, GUI tool used to extract and parse Event Logs and display their internals. The tool allows one to export all records into a text file, generate 'canned – reports' and provides filtering options to display only event records of interest. ***evtx_view*** runs on Windows, Linux and Mac OS-X.

Copyright © TZWorks LLC

www.tzworks.com

Contact Info: info@tzworks.com

Document applies to v1.21 of ***evtx_view***

Updated: Apr 15, 2024

Table of Contents

1	Introduction	2
2	Event Logs and some Differences between Operating Systems.....	2
3	How to Use <i>evtx_view</i>	3
3.1	Analyzing Event Logs in Volume Shadows	6
4	Event Category Reports	7
4.1	Password changes.....	7
4.2	System clock changes.....	8
4.3	Logons	8
4.4	Credential changes.....	8
4.5	USB Plug-n-play events	9
5	Extracting Records using Filters	9
6	Examining the Record Internals	10
7	User Defined Templates.....	11
7.1	EVT type logs (vice EVTX) when using templates	12
7.1.1	Finding the Definitions of the Parameters for each Event ID.	12
8	X-Window Dependencies.....	14
9	Authentication and the License File.....	15
9.1	<i>Limited</i> versus <i>Demo</i> versus <i>Full</i> in the tool's Output Banner.....	15
10	References	15

TZWorks® EventLog Viewer (*evtx_view*)

Users Guide

Copyright © TZWorks LLC

Webpage: http://www.tzworks.com/prototype_page.php?proto_id=4

Contact Information: info@tzworks.com

1 Introduction

evtx_view is a GUI based 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 counter parts.

Originally inspired by the paper by Andreas Schuster on *Introducing the Microsoft Vista event log format*, **evtx_view** was an attempt to take the concepts introduced in the paper and implement a Windows API independent parsing engine. Written entirely in C++, the **evtx_view** parse engine has been ported over to Windows, Linux and Mac OS-X.

While Schuster's paper described the key elements of the new Microsoft event log file format in Vista and Windows 7, his paper included notes on the proprietary binary encoding of XML. Further research was necessary, however, in order fully breakout the binary XML structure used in Windows Vista and beyond. **evtx_view** is a culmination of this research. Under the hood, **evtx_view** uses the same event log parsing engine as **evtwalk** [7].

evtx_view also makes use of the [FOX-toolkit](#). FOX is a C++ based Toolkit for developing GUI applications that can easily run across various platforms by compiling the source for the appropriate OS. FOX is distributed under the GNU Lesser General Public License (LGPL), with a FOX Library License addendum.

The output is presented as a tree-view where one can select the components of an event log and display their internal structure. The tool allows one to generate reports for certain specific event log categories, such as USB plug-n-play events, credential changes, password changes, etc. If one of the available reports does not address an analyst's needs, there is an option for a custom, user-generated 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 a Windows 7 box. 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

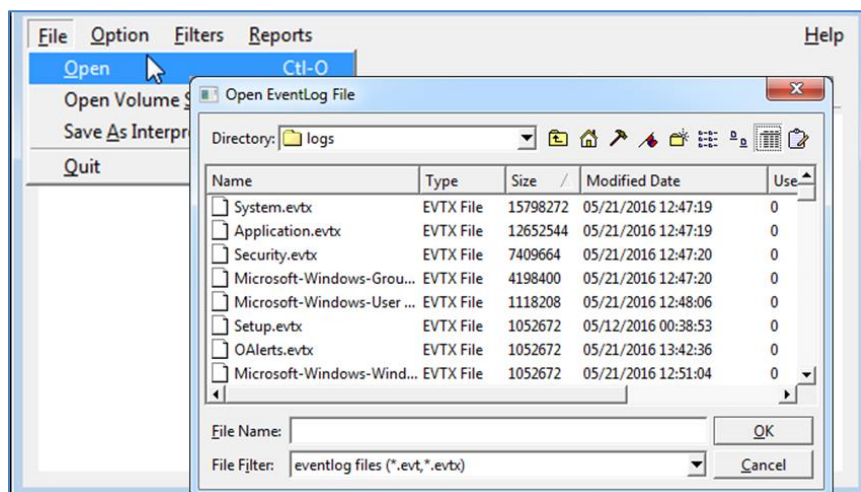
%windir%\system32\config\AppEvent.Evt / SecEvent.Evt / SysEvent.Evt]

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

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

3 How to Use *evtx_view*

Once ***evtx_view*** is launched, the tool will try to establish administrator privileges. This is to allow it to access the Windows event log directory on a live host machine. To analyze an event log, one just navigates to the desired log one wishes to analyze, and ***evtx_view*** will attempt to parse each record in the log. Since ***evtx_view*** uses the FOX (Free Objects for X) as the GUI engine, the main GUI panel and dialog boxes are different from the normal Windows operating system ones. While the differences are minor, the FOX toolkit allows developers to easily port GUI tools from one operating system to another relatively easily.

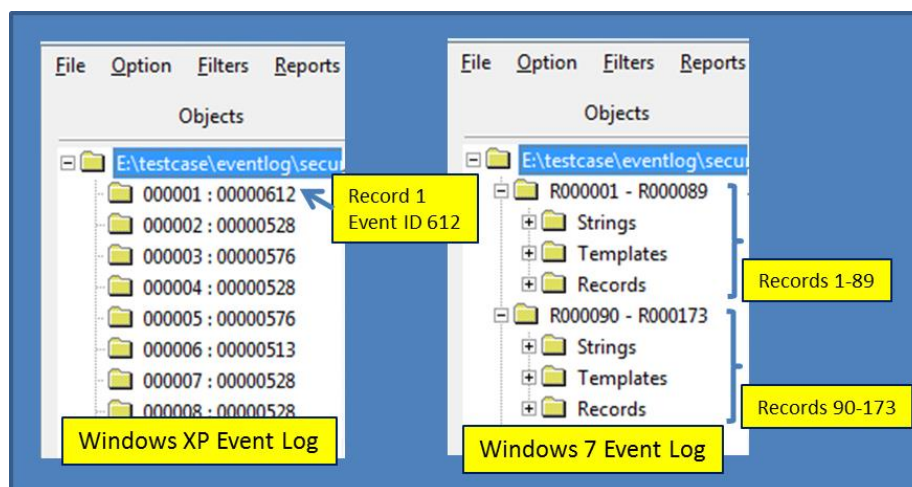


Once the data is parsed, a set of statistics is computed and shown. The statistics report breaks out each event ID as far as, the number of occurrences, the minimum timestamp and the maximum timestamp. This resulting histogram is displayed in descending order based on the number of occurrences. Below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

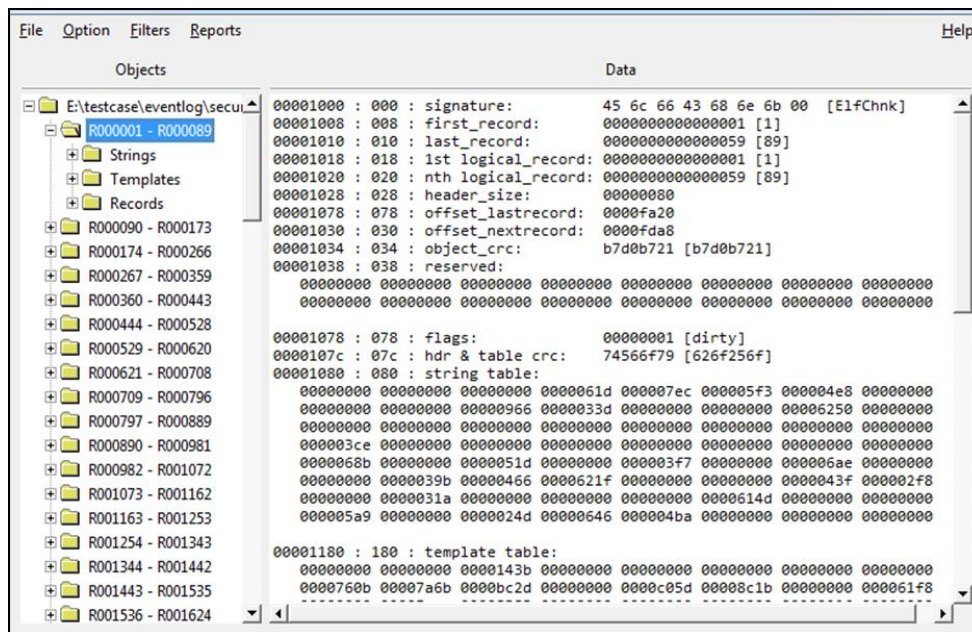
--- overall log stats ---

event id	occurrences	from date	from time	to date	to time
all	23687	09/11/2014	20:29:12.716	01/24/2015	19:19:08.205
4624	8344	09/11/2014	20:29:14.213	01/24/2015	19:19:08.204
4672	6546	09/11/2014	20:29:14.213	01/24/2015	19:19:08.205
4907	2180	09/11/2014	22:21:09.573	01/14/2015	14:39:03.381
5058	1032	09/12/2014	03:41:03.059	01/24/2015	14:56:07.858
5061	1032	09/12/2014	03:41:03.059	01/24/2015	14:56:07.858
4648	504	09/11/2014	20:29:15.141	01/24/2015	14:56:11.070
4608	497	09/11/2014	22:14:17.415	01/24/2015	14:56:05.500
4902	497	09/11/2014	22:14:17.415	01/24/2015	14:56:05.500
5024	497	09/11/2014	22:14:17.415	01/24/2015	14:56:05.500
5033	497	09/11/2014	22:14:17.415	01/24/2015	14:56:05.500
1100	496	09/11/2014	22:13:59.817	01/24/2015	03:56:52.492
5056	490	09/11/2014	22:33:28.174	01/24/2015	14:56:06.358
4647	484	09/11/2014	22:13:59.302	01/24/2015	03:56:51.447
4634	340	09/13/2014	02:56:15.204	01/24/2015	18:50:25.952
4904	49	09/11/2014	20:57:18.231	01/21/2015	13:51:32.659
4905	49	09/11/2014	20:57:18.231	01/21/2015	13:51:32.660
4616	47	09/11/2014	20:29:12.716	01/18/2015	14:08:53.976
4735	45	09/11/2014	20:29:12.716	12/10/2014	19:31:45.751
4731	14	09/11/2014	20:29:12.716	12/10/2014	19:31:45.751
4781	14	09/11/2014	23:28:18.428	09/11/2014	23:28:18.428
4732	8	09/11/2014	20:29:12.747	09/12/2014	03:52:24.175
4738	8	09/11/2014	20:29:12.762	09/11/2014	23:28:18.444
4717	7	09/12/2014	03:40:55.920	12/10/2014	19:31:45.767
4723	2	09/13/2014	02:57:14.708	10/03/2014	14:48:50.702
4734	2	11/08/2014	13:47:48.027	12/10/2014	19:30:18.422
1101	1	09/12/2014	02:12:38.267	09/12/2014	02:12:38.267
4720	1	09/11/2014	20:29:12.747	09/11/2014	20:29:12.747
4722	1	09/11/2014	20:29:12.762	09/11/2014	20:29:12.762
4724	1	09/11/2014	20:29:12.809	09/11/2014	20:29:12.809
4728	1	09/11/2014	20:29:12.747	09/11/2014	20:29:12.747
4733	1	09/11/2014	20:29:12.794	09/11/2014	20:29:12.794

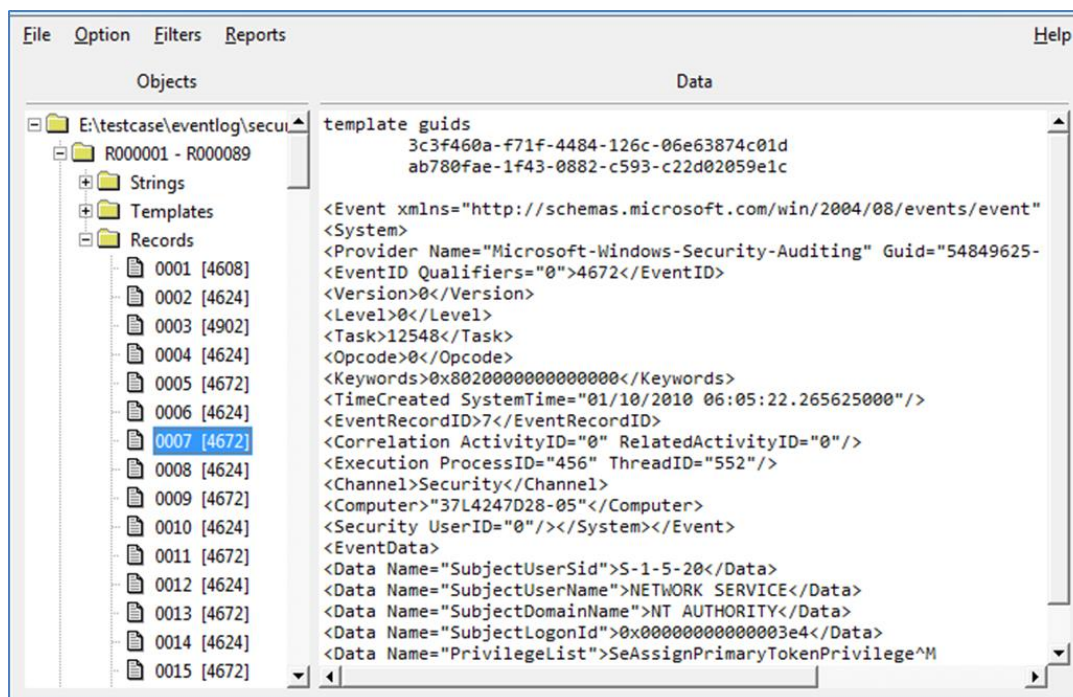
One can navigate to each record or structure from the tree-view pane. Depending on which type of log file is analyzed, this portion of the GUI is broken up into chunks of records for Windows 7 logs or individual event records for Windows XP. Below is a screenshot of the how Windows XP is rendered vice Windows 7. The rendering difference is a reflection of how the two operating systems internally structure their respective event log. Windows XP is just a sequence of records, while Windows 7 encapsulates a chunk of records in what is called an *ElfChnk*. **evt_x_view** preserves this by displaying each log's internals.



From the tree-view, one can select individual records to display, or in the case of Windows 7, display the associated *ElfChnk*. Below is an example of looking at selected Windows 7 *ElfChnk*. As one can see, all the internals of the structure as well as the file offset are displayed. While not normally used by the analyst, this type of breakout is useful for the reverse engineer.



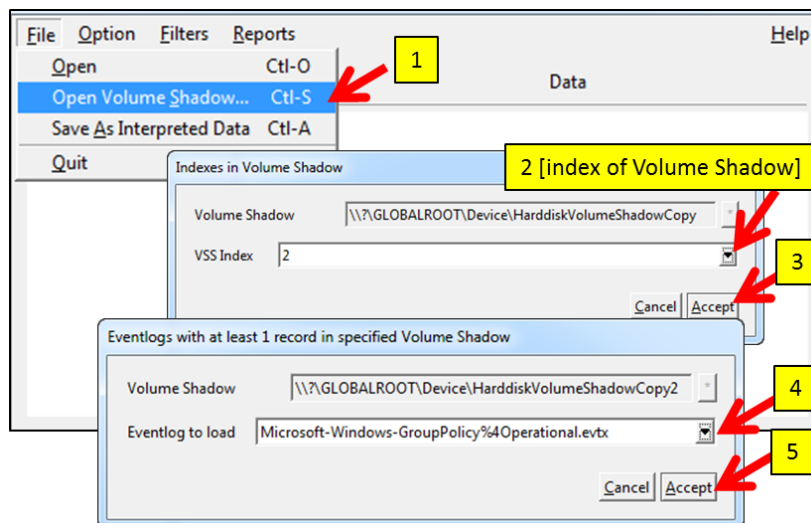
Navigating down to an individual record, one can display the associated data for that record. Unlike the data for Windows XP, for Windows 7, the event log metadata is expressed as binary XML. The screenshot below shows the re-generated XML data for a Windows 7 record.



3.1 Analyzing Event Logs in Volume Shadows

Volume Shadow copies, as is discussed here, only applies to Windows Vista, Win7, Win8 and beyond. It does not apply to Windows XP.

To analyze a Volume Shadow, one first needs to identify the index of the volume shadow copy one wishes to analyze. Fortunately, *evtx_view* figures out which snapshots are available and presents them to the user in a dropdown selection. This process is shown in the figure below. Each step is numbered in the order that it is executed.

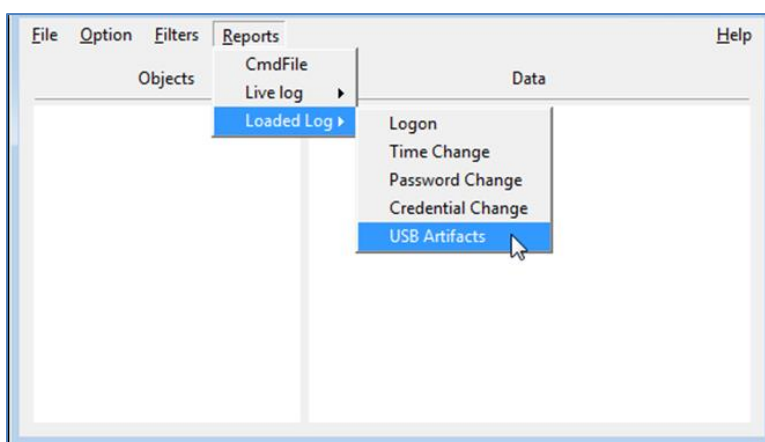


The last dialog box shown (with steps 4 and 5 annotated) gets populated based on the Volume Shadow index that is selected. Only event logs that have at least one record will be shown in the selection combination window.

4 Event Category Reports

While viewing the record data is useful, pulling out specific events and packing a timeline of events into a report is usually more useful to the average analyst. For this use-case, *evtx_view* has a couple of options to pull specific events based on category type.

The screen shot below shows the current categories that are available. The difference between the “Loaded Log” and the “Live Log,” is the former targets the event log that is loaded into the GUI and the latter targets the appropriate event log(s) on the live host machine.



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. More information on this is 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 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		4776	Security log

account			
Domain Controller failed to validate the credentials for an account	675	4777	Security log
Name of an Account was changed	685	4781	Security log

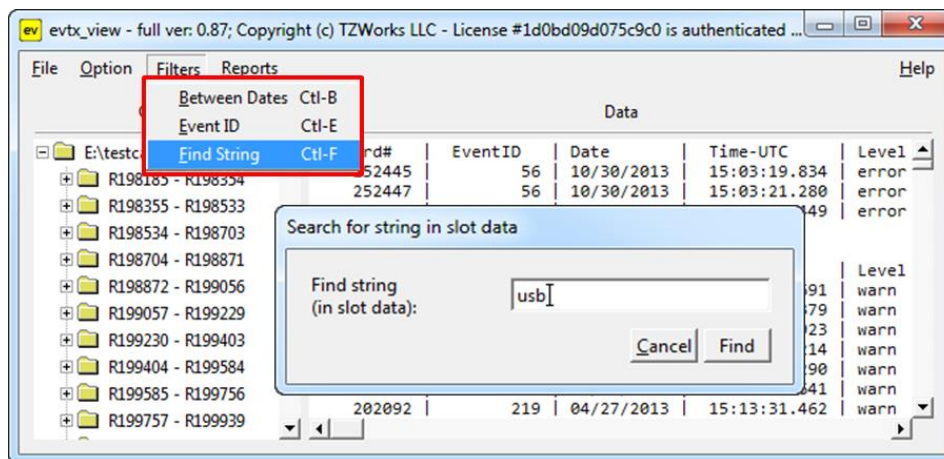
4.5 USB Plug-n-play events

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

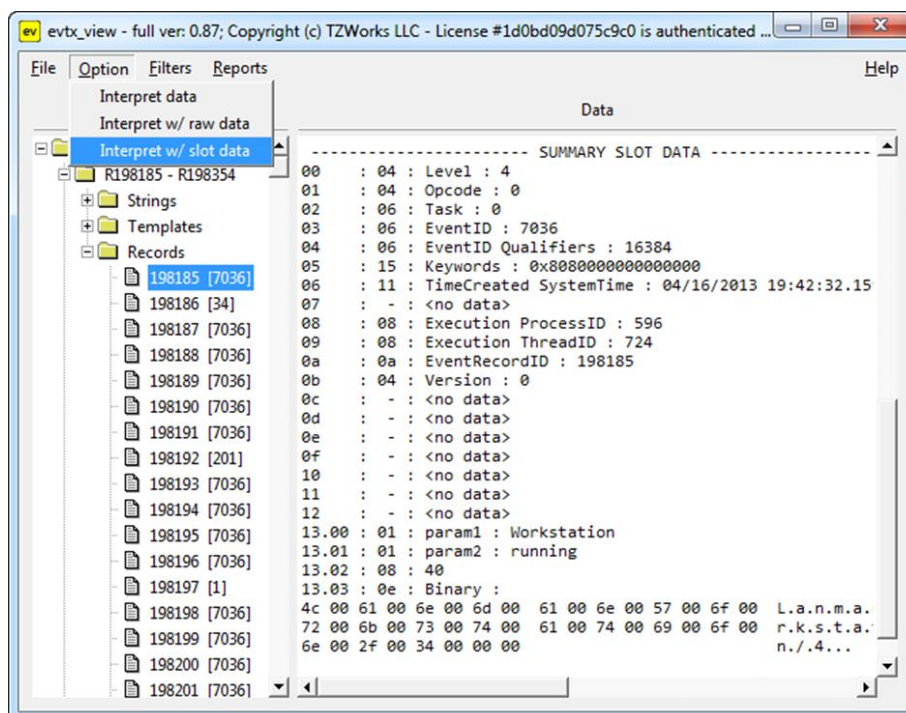
5 Extracting Records using Filters

One can use the available filters to find certain records of interest. Currently, the **evtx_view** filter menu has options for: (a) date range, (b) event id(s), or (c) a string pattern. The example below searches for the string "usb" in the record data. The result is also shown. Each 'like record' type is grouped together in the output, so that each specific event ID header can be included as part of the grouping.



6 Examining the Record Internals

Occasionally it is necessary to examine the raw data of the event record to verify the XML records generated are correct. To look at the raw data, one has two options: (a) either to look at the record with the associated hexadecimal data or (b) to look at the record with the raw slot data (Vista and later). Below is a screenshot of the latter option. The bottom of the right pane shows the slot data associated with the record.



7 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 text 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.

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
 - a. Must start with the sequence: **!cmd**
 - b. And contain the following options, using comma delimiters (in any order):

-enum_evtxlog or **-enum_evtlog**

-id, <event id to extract>

-name, <event name to use for output>

-conditions, <parameter name / value name> [note: parameter name = '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_evtxlog** or **-enum_evtlog**, 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 WinXP. Below are two examples:

```
!cmd, -enum_evtxlog, -type, security, -id, 4624, -name, logon, -pull, \
    TargetUserId, -pull, SubjectUserId, -pull, TargetUserName, -pull, \
    LogonProcessName
```

```
!cmd, -enum_evtlog, -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, TargetUserId, ... -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 WinXP 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. One cannot blindly use a template for uncommon types of log files, unless you are aware each event ID in your template is unique to that event log. 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 **-type [system, security, application, etc]** option guides *evtwalk* to match the proper log file with the event ID specified.

The built-in 'category' reports, however, 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.

7.1 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.

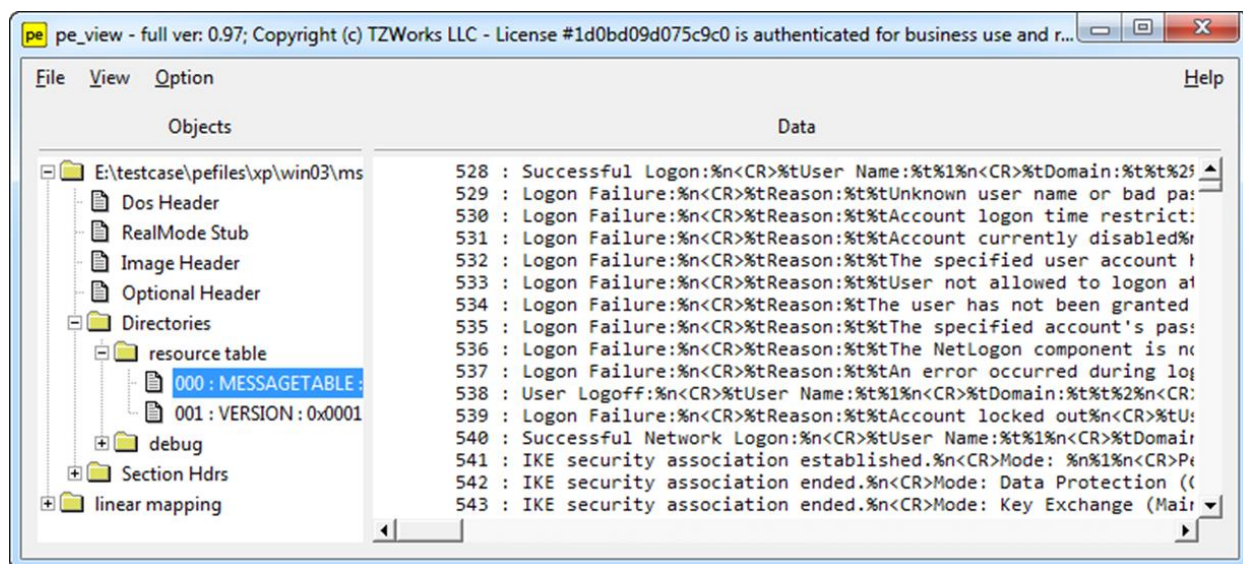
When parsing these types of logs, the output shows the headings param_01, param_02, etc. These heading are just dummy names to indicate which values go to which position. Therefore, when using the **-pull** keyword in the context of EVT logs, one can indicate the position of the field as well as the 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.

7.1.1 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:


```

record size: 0000013c
signature: 654c664c [LfLe]
record num: 00000002 [2]
time generated: 4ac75ad4 [10/03/09 14:08:20.000]
time written: 4ac75ad4 [10/03/09 14:08:20.000]
event id: 00000210 [0x0000, 0x0210 -> 528]
event type: 0008 [audit success]
num strings: 000f
event category: 0002
flags: 0000 [paired event active]
last record num: 00000000
string offset: 00000070
usersid length: 0000000c
usersid offset: 00000064
data length: 00000000
data offset: 00000136
source name: Security
machine name: MACHINENAME
user sid: S-1-5-19
string 00: LOCAL SERVICE
string 01: NT AUTHORITY
string 02: (0x0,0x3E5)
string 03: 5
string 04: Advapi
string 05: Negotiate
string 06: <null string>
string 07: -
string 08: MACHINENAME$
string 09: <null string>
string 0a: (0x0,0x3E7)
string 0b: 252
string 0c: -
string 0d: -
string 0e: -

```

00 => %1
 01 => %2
 etc

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
```

8 X-Window Dependencies

For this tool to work, the X Window System libraries are required for both Linux and macOS (they are not required for Windows). These libraries use the X11 protocol and graphics primitives to render the graphical user interface components. These libraries are common on Unix-like OS's.

If one is unfamiliar with X Windows or the libraries associated with it, one can download an installer package from XQuartz.org, which is an open-source effort to develop a version of the X Windows System that runs on Linux and macOS.

After the X11 libraries are installed, one needs to ensure they are running prior to running this tool.

9 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.

9.1 *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.

10 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. [FOX-toolkit](#) version 1.6.43
5. [Randy Franklin Smith's online encyclopedia](#).
6. Windows Event Log Viewer, *evt_x_view*, http://tzworks.com/prototype_page.php?proto_id=4
7. Windows Event Log Parser, *evt_walk*, http://tzworks.com/prototype_page.php?proto_id=25
8. [X Window System Libraries](#) by XQuartz.org.