

TZWorks® Portable Executable Scanner (*pescan*) Users Guide



Abstract

pescan is a standalone, command-line tool that scans portable executable (PE) files and identifies how they were constructed and if they are considered abnormal. Abnormal is defined to be different than Microsoft built PE files. ***pescan*** runs on Windows, Linux and macOS.

Copyright © TZWorks LLC

www.tzworks.com

Contact Info: info@tzworks.com

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

Updated: Apr 15, 2024

Table of Contents

1	Introduction	2
2	How to Use <i>pescan</i>	2
2.1	Default mode: Pulling statistics from PE files	4
2.1	Volume Shadow Copies	5
2.2	<i>PEiD</i> signature mode.....	6
2.3	Computing a hash	6
2.4	Anomaly detection.....	7
2.5	Extraction of Message Tables from select PE files.....	8
2.6	Extraction of Message Templates from select PE files	9
3	Available Options	10
4	Authentication and the License File.....	13
4.1	<i>Limited</i> versus <i>Demo</i> versus <i>Full</i> in the tool's Output Banner.....	13
5	References	13

TZWorks® PE Scanner (*pescan*) Users Guide

Copyright © TZWorks LLC

Webpage: http://www.tzworks.com/prototype_page.php?proto_id=15

Contact Information: info@tzworks.com

1 Introduction

pescan is a command line tool used to scan portable executable (PE) files to identify how they were constructed. Various metadata is displayed, identifying items such as:

- Compile timestamp
- MACB timestamps
- File size and type of executable
- Target OS and whether binary is 32 or 64 bit
- Linker version used
- Entry point address and desired image base address
- Whether an X509 certificate was used and who the author is.
- Whether there is a checksum present and does it match the binary
- Optional analysis of the PE internals to generate an abnormality score which compares the internal construction to the standard operating system files. Higher scores equate to larger differences.
- Optional MD5 and/or SHA1 hashes of the file can be generated

As an option, one can also use any standard *PEiD* signature file and pass it to **pescan** during its scan. Additional statistics will be displayed based on the *PEiD* signatures found. One can use this option to generate one's own signatures to help identify certain classes of binaries, instead of just using it to identify compiler version and/or packer version.

While not normally used, **pescan** also has to options to extract *event log messages and templates* from certain categories of PE files. More discussion on this topic is presented later.

Internally, **pescan** uses the same PE libraries that are used in **pe_view** to analyze and display the PE internals. Like **pe_view**, **pescan** uses cross platform C++, and hence, there are compiled versions for Windows, Linux and macOS, both in 32-bit and 64-bit binaries.

2 How to Use *pescan*

The command menu shows most of the options available. **pescan** can operate on: (a) a single PE file or (b) a collection of PE files using the **-pipe** option. The ability to process many PE files in one session was what **pescan** was designed for.

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.

There are two basic forms of output: (a) unstructured text, displaying multiple lines per PE file analyzed and (b) structured, delimited text, displaying one line per PE file that is analyzed. The output is really a function whether one is analyzing one file or many files in one session. For single file analysis, the output will be unstructured, multi-lined format. For files processed via STDIN (standard input), via the pipe option, the output will be one line per file. Furthermore, the structured, delimited output can be either (a) cleanly formatted using pipe delimiters or (b) packed fields using a comma delimiter. The first is the default, while the latter is used with the **-csv** option.

The structured delimited output can generate some very long lines, and therefore, one should redirect the output to an archive file to ensure all the data can be read in a text editor or spreadsheet application.

Aside from the formatting options, there are a few options for selecting different modes for **pescan** to operate in when processing the data. Each of these modes discussed in their respective sections below.

```
Administrator: Windows PowerShell

Usage

pescan <pefile> [options]
pescan <pefile> -peid <peid file> > results.txt
pescan <pefile> -anomalies > results.txt
dir *.exe /b /s | pescan -pipe -csv > results.csv
pescan -enumdir <folder> -num_subdirs <#> [options]

Basic options
-csv                = output in comma separated value format
-out <results file> = send output to a file
-peid <peid file>   = use this PEiD file during scan (see readme)

Additional options
-pipe              = use stdin to pipe in files to parse
-hostname <name>   = display hostname in output
-base10           = output in base10 vice hex [only with -csv]
-string_resource  = extract string resources [not avail w/ -csv]
-msg_table        = extract any msg tables [not avail w/ -csv]
-wevt_temp        = extract eventlog templates [not avail w/ -csv]
-dump_entrypt <# bytes> = dump the bytes at entrypoint [up 0x50 bytes]
-md5              = output MD5 hash of binary
-sha1             = output SHA1 hash of binary
-appid            = compute AppID [one of several possible ones]
-dateformat mm/dd/yyyy = "yyyy-mm-dd" is the default
-timeformat hh:mm:ss  = "hh:mm:ss" is the default
-pair_datetime     = *** combine date/time into 1 field for csv
-no_whitespace     = remove whitespace between csv delimiter
-quiet            = don't display status during run
-csv_separator "|"  = use a pipe char for csv separator
-anomalies        = enum PE attributes differing from norm
-rating <min rating> = display results for ratings at or above
-filter <*partial*|.ext> = *** filters stdin data from -pipe option
```

2.1 Default mode: Pulling statistics from PE files

One can either pull statistics from one file or a collection of files from a directory. Below are two examples of the syntax for each option. For processing many files, one should redirect the output into a separate file due to the amount of data outputted into one line.

Example1: `pescan <pefile>`

Example2: `dir c:\<somedir>* /b /s | pescan -pipe [-csv] > results.txt`

When processing many files in one session, if one does not use the CSV option the results file will be easy to read in any text editor, since the fields are reasonably spaced to allow for better viewability. This default option also has the benefit that it uses pipes as delimiters which means there is no chance of namespace collision with filenames should one want to import this file into Excel. On the other hand, if one uses the CSV option, the individual fields are packed between comma delimiters. This is most useful for spreadsheet applications and less useful for text editors to view the data. In this case, if a filename contains a comma, **pescan** will change the comma in the name to a space to keep the fields aligned with the proper delimiter count. Below is example of the output for the two cases above.

```
pescan - full ver: 0.22; Copyright (c) TZworks LLC
License is authenticated and valid
run time: 10/11/2013 03:48:50 [UTC]
cmdline: pescan64 c:\windows\system32\ntoskrnl.exe

File selected: c:\windows\system32\ntoskrnl.exe
Company name: Microsoft Corporation
Compile date: 08/29/2013 01:13:25 [UTC]
Create date: 10/10/2013 01:09:08 [UTC]
Access date: 10/10/2013 01:09:08 [UTC]
Modify date: 08/29/2013 02:17:48 [UTC]
File size: 0x05549504 [5549504]
PE type: 64 bit - native
Linker version: 9.0
Min OS version: win7 or win8/R2
Entrypoint VA: 0x002b36f0 [2832112] - FileAddr: 0x0022
Imagebase: 0x0000000140000000 [5368709120]
Message Table exists
Digital signature embedded into PE
Checksum: validated
```

```
pescan - full ver: 0.21; Copyright (c) TZworks LLC
run time: 10/10/2013 23:08:49 [UTC]
cmdline: <filelist> ... | pescan64 -pipe

compiled | time-utc | created | time-utc | cpu | linker | min OS | entry rva | entryfaddr | imagebase | cert | chksum | file
11/16/2011 | 17:47:43 | 11/16/2011 | 18:48:36 | 32 bit | 10.0 | winXP | 0x0003f1e4 | 0x0003e5e4 | 0x0000400000 | no | yes | c:\tools\dot
11/16/2011 | 17:35:14 | 11/16/2011 | 18:35:39 | 32 bit | 10.0 | winXP | 0x00028cd6 | 0x000280d6 | 0x0000400000 | no | yes | c:\tools\ne
05/16/2011 | 08:14:06 | 11/17/2011 | 17:51:21 | 32 bit | 9.0 | win2K | 0x0000212d | 0x0000152d | 0x0000400000 | no | no | c:\tools\gr
05/16/2011 | 08:14:04 | 11/17/2011 | 17:51:21 | 32 bit | 9.0 | win2K | 0x000026e4 | 0x00001ae4 | 0x0000400000 | no | no | c:\tools\gr
05/16/2011 | 08:14:02 | 11/17/2011 | 17:51:21 | 32 bit | 9.0 | win2K | 0x0000361a | 0x00002a1a | 0x0000400000 | no | no | c:\tools\gr
05/09/2011 | 17:58:19 | 11/17/2011 | 17:51:21 | 32 bit | 9.0 | win2K | 0x00005b07 | 0x00004f07 | 0x0010000000 | no | no | c:\tools\gr
05/16/2011 | 08:12:11 | 11/17/2011 | 17:51:21 | 32 bit | 9.0 | win2K | 0x0000cfed | 0x0000c3ed | 0x0010000000 | no | no | c:\tools\gr
05/23/2003 | 17:51:21 | 11/17/2011 | 17:51:21 | 32 bit | 9.0 | win2K | 0x00001149 | 0x00001149 | 0x0010000000 | no | no | c:\tools\g
```

2.1 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 4 of the volume shadow copy, one would prepend the keyword and index, like so, **%vss%4** to the normal path of the file. For example, to access a notepad located in the Windows\System32 directory from the *HarddiskVolumeShadowCopy4*, the following syntax can be used:

```
pescan64 %vss%4\windows\system32\notepad.exe

File selected:  \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy4\windows\system32\notepad.exe
Company name:   Microsoft Corporation
Compile date:   07/13/2009 23:56:35 [UTC]
Create date:    [UTC]
Access date:    [UTC]
Modify date:    [UTC]
File size:      0x00193536 [193536]
PE type:        64 bit - exe
Linker version: 9.0
Min OS version: Win7 or Win08/R2
Entrypoint VA:  0x00003570 [13680] - FileAddr: 0x00002b70 [11120]
Imagebase:      0x0000000100000000 [4294967296]
Checksum:       Validated
```

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 appended to the **%vss%** keyword.

2.2 PEiD signature mode

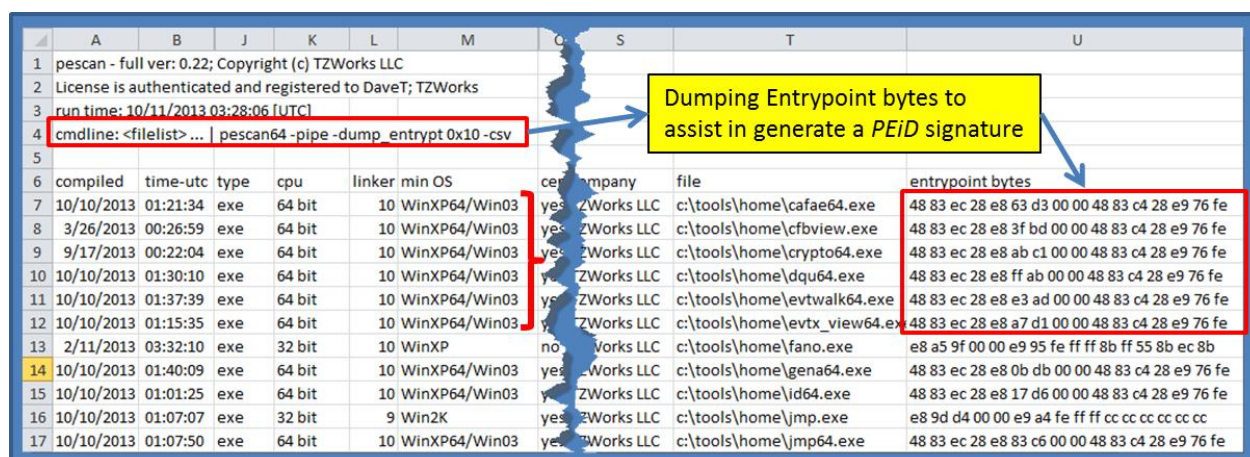
PEiD is a separate GUI tool used to scan and detect common packers, encryptors, and compilers for PE files. There were three different scanning modes in *PEiD*: (a) *Normal mode* – which scans the PE file at its entry point for all defined signatures, (b) *Deep Mode* – which scans the PE file section containing the entry point, and (c) *Hardcore Mode* – which does a complete scan of the entire PE file.

pescan can make use of the *PEiD* signature file and perform a similar scan as *PEiD*, using the **-peid <signature file>** option. *pescan* operates in what *PEiD* calls the 'normal mode' as the default option and will only scan the entry point of the PE file. There are two other sub-options to include a deeper and hardcore scan. To scan more than just entry points, one can use the **-peid <signature file> -section** sub-option to scan the entire section encompassing the entry point. While not recommended, there is also an option to scan the entire file, in what *PEiD* call 'hardcore mode'. This latter option is invoked with the combination **-peid <signature file> -full**. Be forewarned, this last option takes much longer to process and empirically gives many false positives. Below is an example of using the *PEiD* option with *pescan*.

```
dir c:\ksomedir\* /b /s | pescan -pipe -peid <signature file> -csv > results.csv
```

Many *PEiD* signature files are available for download via the Internet, and many of them are dated. Therefore, to assist one in generating one's own set of signatures that can be incorporated in a *PEiD* type file, *pescan* has an option to dump the bytes of the entry point of the file under analysis. This option is

-dump_entrypt <# bytes>. Using this option, one is limited up to 80 bytes to be dumped as part of the output. Armed with this data, one can generate their own *PEiD* signatures using the standard *PEiD* notation. See the snapshot below to see how the data is presented to the user.



	A	B	J	K	L	M		S	T	U
1	pescan - full ver: 0.22; Copyright (c) TZWorks LLC									
2	License is authenticated and registered to DaveT; TZWorks									
3	run time: 10/11/2013 03:28:06 [UTC]									
4	cmdline: <filelist> ... pescan64 -pipe -dump_entrypt 0x10 -csv									
5										
6	compiled	time-utc	type	cpu	linker	min OS	company	file	entrypoint bytes	
7	10/10/2013	01:21:34	exe	64 bit	10	WinXP64/Win03	yes	ZWorks LLC	c:\tools\home\cafae64.exe	48 83 ec 28 e8 63 d3 00 00 48 83 c4 28 e9 76 fe
8	3/26/2013	00:26:59	exe	64 bit	10	WinXP64/Win03	yes	ZWorks LLC	c:\tools\home\cfbview.exe	48 83 ec 28 e8 3f bd 00 00 48 83 c4 28 e9 76 fe
9	9/17/2013	00:22:04	exe	64 bit	10	WinXP64/Win03	yes	ZWorks LLC	c:\tools\home\crypto64.exe	48 83 ec 28 e8 ab c1 00 00 48 83 c4 28 e9 76 fe
10	10/10/2013	01:30:10	exe	64 bit	10	WinXP64/Win03	yes	ZWorks LLC	c:\tools\home\dqu64.exe	48 83 ec 28 e8 ff ab 00 00 48 83 c4 28 e9 76 fe
11	10/10/2013	01:37:39	exe	64 bit	10	WinXP64/Win03	yes	ZWorks LLC	c:\tools\home\evtwalk64.exe	48 83 ec 28 e8 e3 ad 00 00 48 83 c4 28 e9 76 fe
12	10/10/2013	01:15:35	exe	64 bit	10	WinXP64/Win03	yes	ZWorks LLC	c:\tools\home\evtx_view64.exe	48 83 ec 28 e8 a7 d1 00 00 48 83 c4 28 e9 76 fe
13	2/11/2013	03:32:10	exe	32 bit	10	WinXP	no	ZWorks LLC	c:\tools\home\fano.exe	e8 a5 9f 00 00 e9 95 fe ff ff 8b ff 55 8b ec 8b
14	10/10/2013	01:40:09	exe	64 bit	10	WinXP64/Win03	yes	ZWorks LLC	c:\tools\home\gena64.exe	48 83 ec 28 e8 0b db 00 00 48 83 c4 28 e9 76 fe
15	10/10/2013	01:01:25	exe	64 bit	10	WinXP64/Win03	yes	ZWorks LLC	c:\tools\home\id64.exe	48 83 ec 28 e8 17 d6 00 00 48 83 c4 28 e9 76 fe
16	10/10/2013	01:07:07	exe	32 bit	9	Win2K	yes	ZWorks LLC	c:\tools\home\jmp.exe	e8 9d d4 00 00 e9 a4 fe ff ff cc cc cc cc cc cc
17	10/10/2013	01:07:50	exe	64 bit	10	WinXP64/Win03	yes	ZWorks LLC	c:\tools\home\jmp64.exe	48 83 ec 28 e8 83 c6 00 00 48 83 c4 28 e9 76 fe

2.3 Computing a hash

Sometimes, one wishes to identify a PE file in a way to compare it with other databases that collate binary hashes, such as Virus Total or Bit 9. To aid in this, *pescan* can either compute the MD5 hash or SHA2 hash or both using the options

-md5 and **-sha1** separately or together. The hash is added as another field (or two) in the output displayed. See the example in the next section below.

2.4 Anomaly detection

As an option, *pescan* has what we call abnormality detection. In the context of *pescan*, this consists of looking at the PE file structure/packing and comparing it to what normally is available with a Windows operating system executable, dynamic link library or driver file. Based on the differences found, a score is assigned. The higher the score, the more differences were found in the construction of the binary file under analysis from the norm. Furthermore, a high score does not mean the binary is malware or malicious, it just means it was constructed using tools and/or methods that are different than what was normal to other Windows operating system files. It could be that the author of the binary wanted to obfuscate the internals of the file from reverse engineers, or it could have been built from some non-standard compiler/linker. If there was a high score assigned to a binary, one needs to also look at the reasons for the high score. Here are a handful of things *pescan* looks at during an anomaly scan while making a determination:

- Is the entry point in a standard location or not
- Does the file make use of thread local storage
- Is the import table present
- Does any of the sections in the PE have high entropy
- Are any of the PE section headers abnormal
- Was there evidence of strings being obfuscated
- Were the exports obfuscated
- Was there a checksum mismatch
- Do any resources contain possible embedded PE files
- Was there evidence of PE patching
- Was any of the resources modified after compilation
- etc.

While the list above is not exhaustive, it lists some of the major items *pescan* will look for. The higher the number that is generated just means *pescan* encountered more of the items listed above in one PE file.

One can invoke the anomaly detection by using the option **-anomalies** or **-rating <minimum rating>**. The first option will compute and display the rating for all files requested, while the latter will only display those files with a rating at or above what was specified. Usually when using this option, one will want to have the MD5 or SHA1 hash computed in parallel.

Below is a sample output using the hashing options and anomaly detection. If a binary scored a high rating, then one can either analyze the file in more detail using other tools available or just submit the MD5/SHA1 hash to Virus Total, Bit 9 or some other service for their take.

base	cert	chksum	md5	sha1	company	file	rating	no
30	no	no	6d974e18042a5a52433944c4db8bfd	1d5b2b164cf356dfcdd47cc82db0f860b916d6e	<unk>	c:\tools\grp\bin\gltZpr0	12	Ver
0000	no	yes	fe5ec4b2a0702c70a4cd3aa09ce0c571	1d1b202c0e8f5dce80a2793a6121b909eec38397	<unk>	c:\tools\grp\bin\freety	9	Ver
000	no	yes	86fc9bcaa15f979b0102c86e2c75737	b59258d842b5818bdd9e0e58031c1b6306ba84cf	The GLib develop	c:\tools\grp\bin\libgthr	7	No R
000	no	yes	f78ea80cec007b2c32fb10f9c6ca9f39	b9febc13e5b0e64554f160d0607f08a3c5547ddc	Free Software Fo	c:\tools\grp\bin\gettext	6	No
0000	no	yes	eb2d4c4d4a527bc88a69a16cc99afcf5	b326ec4919e1ec9595c064b24853b1e6b71530a3	Free Software Fo	c:\tools\grp\bin\intl.dll	6	No
0000	no	yes	7cb03aa1126d4024a2c9745f3952e383	d3b874825488cb9eea14894b03a99fb21b68895	Sun Microsystem	c:\tools\grp\bin\libatk-1	6	No R
0000	no	yes	961e0ceadfb66e99310bf4dfc441ccd8	5b6841e8ee724f72877474ac56c7464fe1738b80	The GTK develop	c:\tools\grp\bin\libgdk	6	No R
0000	no	yes	835daf325a93ca18d78c9a2a292567ca	19fa77fe8c21257af24f73414bf80e9a4634e673	The GTK develop	c:\tools\grp\bin\libgdk	6	No R
0000	no	yes	8cc667edd3415395e21d79c3f83745ab	1fc826ea07ff3f836474d566857326cc240de247	The GLib develop	c:\tools\grp\bin\libglib	6	No R
00000	no	yes	47417c314255bb58b28f77c880a6c810	81caee84cd632693f317382e836eb3a60e3490f0	The GLib develop	c:\tools\grp\bin\libgmo	6	No R
80000	no	yes	4cf9a23aad2825fc9f2c9876919a5f0	f0d5e2458ea0c64062fb6804d5bf937c4d0688d	The GTK develop	c:\tools\grp\bin\libgtk-3	6	No R
80000	no	yes	1cb61fa3691a368			libpan	6	No R
0000	no	yes	f3f3e42a9ba9358			libpan	6	No R
0000	no	yes	0922506efb7d32e906c01c83190c204	81230da0ab11e0001119067c0a2ae1903e22c	Red Hat Software	c:\tools\grp\bin\libpan	6	No R
000	no	yes	03384ab6368b68ed16ecb9e6352539af	4ec6112fc29ca32be5f1066b75ce8ba5f82d94a4	Free Software Fo	c:\tools\grp\bin\ngette	6	No R
0000	no	yes	7ff4d84576e75dba209d614976355a0c	f5df451309fafd85890e4dc194f697a63c907191	The GLib develop	c:\tools\grp\lib\debug\	5	No

2.5 Extraction of Message Tables from select PE files

Certain PE files contain message tables that map event ID to some message. Services and drivers in conjunction with the event log service use these tables to log event IDs to record noteworthy events. The mapping of event ID to a definition is recorded in the appropriate PE's message table. Extracting this mapping can be done with **pescan** using the option **-msg_table**. Using this option will cause the output to be rather verbose, and thus, this option cannot be used with the CSV option.

Below is an example of using this option on the infamous Windows **ntdll.dll** file.

```
pescan - full ver: 0.22; Copyright (c) TZworks LLC
License is authenticated and registered to DaveT; TZworks
run time: 10/11/2013 03:13:36 [UTC]
cmdline: pescan64 c:\windows\system32\ntdll.dll -msg_table

File selected: c:\windows\system32\ntdll.dll
Company name: Microsoft Corporation
Compile date: 08/29/2013 02:17:08 [UTC]
Create date: 10/10/2013 01:09:04 [UTC]
Access date: 10/10/2013 01:09:04 [UTC]
Modify date: 08/29/2013 02:16:35 [UTC]
File size: 0x01732032 [1732032]
PE type: 64 bit - dll
Linker version: 9.0
Min OS version: win7 or win8/R2
Entrypoint VA: 0x00000000 [0] - FileAddr: 0x00000000 [0]
Imagebase: 0x0000000078e50000 [2028273664]
Message Table exists
Digital signature embedded into PE
Checksum: validated

-----
Message Table follows:
event id                                     msg
-----
0 : STATUS_WAIT_0<CR>
1 : STATUS_WAIT_1<CR>
2 : STATUS_WAIT_2<CR>
3 : STATUS_WAIT_3<CR>
63 : STATUS_WAIT_63<CR>
128 : STATUS_ABANDONED_WAIT_0<CR>
191 : STATUS_ABANDONED_WAIT_63<CR>
192 : STATUS_USER_APC<CR>
256 : STATUS_KERNEL_APC<CR>
257 : STATUS_ALERTED<CR>
258 : STATUS_TIMEOUT<CR>
259 : The operation that was requested is pending completion
260 : A reparse should be performed by the Object Manager
261 : Returned by enumeration APIs to indicate more data
262 : Indicates that all privileges or process refs
```

Extracting any Message
Tables used in EventLogs

2.6 Extraction of Message Templates from select PE files

Similar to message tables, certain PE files contain message *templates* that are used to help construct messages and are used primarily for boiler plate information and/or syntax. They are creations starting with the Vista operating system and are unique to binary XML format used in the new event log. **pescan** also has the ability to extract these message templates using the **-wevt_temp** option. Using this option will cause the output to be rather verbose, and thus, this option cannot be used with the CSV option.

Below is an example of using this option on the infamous Windows *wevtscv.dll* file.

	point. Finally, the -full sub-option allows non-entry point signatures to be scanned anywhere in the PE file (this is a very slow option).
-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.
-hostname	Option is used to populate the output records with a specified hostname. The syntax is -hostname <name to use> .
-msg_table	Extract any message tables that are embedded into the PE file. This option is not available with the -csv option.
-wevt_temp	Extract any event log templates that are embedded into the PE file. This option is not available with the -csv option.
-base10	Ensure all size/address output is displayed in base-10 format vice hexadecimal format. Default is hexadecimal format.
-dump_entrypt	Dump the specified number of bytes starting at the PE's entry point. This dump is limited to the first 0x50 bytes. The syntax is: -dump_entrypt <# bytes>
-md5	Output the MD5 hash of the PE file.
-sha1	Output the SHA1 hash of the PE file
-appid	Compute the Application Identifier (AppID) that can be used in Jump Lists. (note) AppID's can be manually set by the application itself to be any arbitrary value. This is just an implementation of the algorithm used to compute the AppID using the application's path.
-anomalies	Look for anomalous PE construction and packing. Anomalous here is defined to be those items that are differing from the normal. High scores outputted here warrant deeper analysis of the PE file.

<i>-rating</i>	Similar to the <i>-anomalies</i> option in functionality, but only output those records with a score at or above the specified rating. The syntax is: <i>-rating <min rating></i> .
<i>-no_whitespace</i>	Used in conjunction with <i>-csv</i> option to remove any whitespace between the field value and the CSV separator.
<i>-csv_separator</i>	Used in conjunction with the <i>-csv</i> option to change the CSV separator from the default comma to something else. Syntax is <i>-csv_separator "/"</i> to change the CSV separator to the pipe character. To use the tab as a separator, one can use the <i>-csv_separator "tab"</i> OR <i>-csv_separator "\t"</i> options.
<i>-dateformat</i>	Output the date using the specified format. Default behavior is <i>-dateformat "yyyy-mm-dd"</i> . 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.
<i>-timeformat</i>	Output the time using the specified format. Default behavior is <i>-timeformat "hh:mm:ss.xxx"</i> One can adjust the format to microseconds, via <i>"hh:mm:ss.xxxxxx"</i> or nanoseconds, via <i>"hh:mm:ss.xxxxxxxxxx"</i> , or no fractional seconds, via <i>"hh:mm:ss"</i> . 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.
<i>-pair_datetime</i>	Output the date/time as 1 field vice 2 for csv option
<i>-quiet</i>	This option suppresses status output as each file is processed.
<i>-utf8_bom</i>	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. Microsoft Portable Executable and Common Object File Format [Specification](#).
2. An In-Depth Look into the Win32 Portable Executable File Format, by Matt Pietrik, MSDN Magazine.
3. Wikipedia, the free encyclopedia. [PE format](#)
4. [FOX-toolkit](#) version 1.6.43 [5] PEiD references: [PEiD](#), [PEiD Forum](#), Example [userdb.txt](#)
5. Portable Executable Identifier, PEiD.