

TZWorks® Prefetch Parser (*pf*) Users Guide



Abstract

pf is a standalone, command-line tool that can parse Windows prefetch files. From a forensics perspective, the prefetch file offers the analyst information about the applications that were executed, their location and the frequency. ***pf*** runs on Windows, Linux and Mac OS-X.

Copyright © TZWorks LLC

www.tzworks.com

Contact Info: info@tzworks.com

Document applies to v1.40 of ***pf***

Updated: Apr 25, 2025

Table of Contents

1	Introduction	2
2	How to Use <i>pf</i>	2
2.1	Handling Volume Shadow Copies	5
2.2	Other Output Options	6
3	Known Issues.....	7
4	Available Options	8
5	Authentication and the License File.....	10
5.1	<i>Limited</i> versus <i>Demo</i> versus <i>Full</i> in the tool's Output Banner	10
6	References	10

TZWorks® Prefetch Parser (*pf*) Users Guide

Copyright © TZWorks LLC

Webpage: http://www.tzworks.com/prototype_page.php?proto_id=1

Contact Information: info@tzworks.com

1 Introduction

pf is a command line tool that parses Windows prefetch files. Using the definition in Wikipedia, “the prefetcher is a component of ... Microsoft Windows starting with Windows XP... that speeds up the Windows boot process and shortens the amount of time it takes to start up programs. In Windows Vista, SuperFetch and ReadyBoost extend upon the prefetcher and attempt to accelerate application and boot launch times...” [1]. A good source for discussion on the internals of the mechanics of prefetching is given in the MSDN article written by Mark Russinovich and David Solomon [2].

The prefetcher behavior is controlled by the Windows registry value "EnablePrefetcher" located in the following registry path: HKLM\ System\CurrentControlSet\Control\Session\Manager\ Memory Management\ PrefetchParameters. The value for “EnablePrefetcher” can have one of the following values [1]:

Value	Meaning
0	Disabled
1	Application launch prefetching enabled
2	Boot prefetching enabled
3	Application launch and boot enabled (default)

Based on various references, Windows maintains the prefetch folder up to 128 entries.

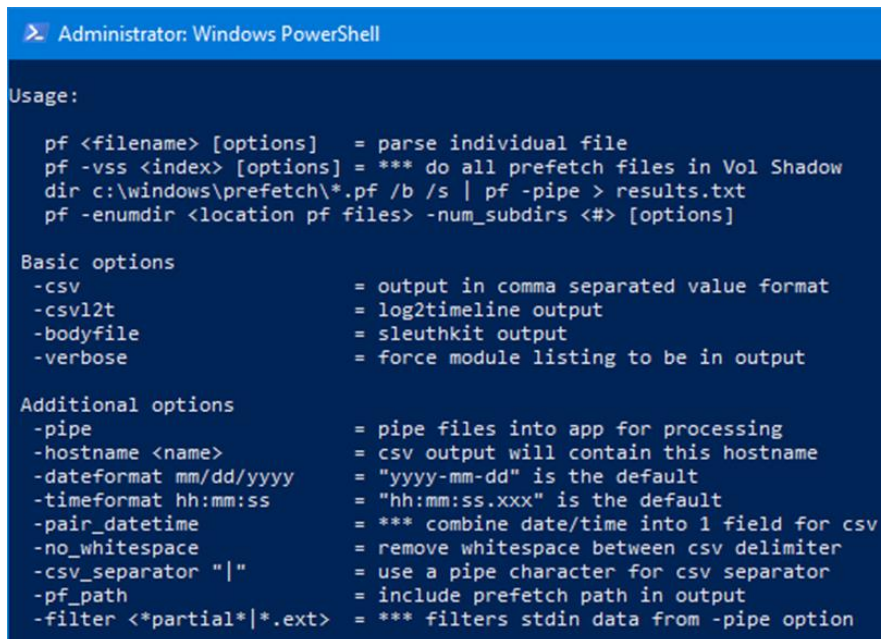
From a forensics standpoint, the prefetch file offers the analyst some information about the applications that were executed, the location of the application, and the frequency that it was run. Specifically, the prefetch file contains information such as: (a) filename, (b) file location, (c) timestamps related to the prefetch entry (created, modified and accessed), (d) the number of times a certain application was run, (e) the last run time, (f) which modules were loaded with the application, and (g) which volumes were used in access the application or the modules used.

2 How to Use *pf*

pf has a number of command line switches. The screenshot below shows the available options. There are two available options that tell **pf** how much data to display to the analyst. The first is the default

mode, which is the verbose option and displays as much information that **pf** can parse. The second is a variety of CSV options that output one line per prefetch file.

The one-line-per-entry behavior will output the (a) application name and path, (b) number of times the application was run, (c) the last time run, and (d) the prefetch file MAC timestamps. The verbose option includes the same information the one-line option outputs, plus module dependencies loaded and volumes used to run the application.



```
Administrator: Windows PowerShell

Usage:

pf <filename> [options] = parse individual file
pf -vss <index> [options] = *** do all prefetch files in Vol Shadow
dir c:\windows\prefetch\*.pf /b /s | pf -pipe > results.txt
pf -enumdir <location pf files> -num_subdirs <#> [options]

Basic options
-csv = output in comma separated value format
-csvl2t = log2timeline output
-bodyfile = sleuthkit output
-verbose = force module listing to be in output

Additional options
-pipe = pipe files into app for processing
-hostname <name> = csv output will contain this hostname
-dateformat mm/dd/yyyy = "yyyy-mm-dd" is the default
-timeformat hh:mm:ss = "hh:mm:ss.xxx" is the default
-pair_datetime = *** combine date/time into 1 field for csv
-no_whitespace = remove whitespace between csv delimiter
-csv_separator "|" = use a pipe character for csv separator
-pf_path = include prefetch path in output
-filter <*partial*|*.ext> = *** filters stdin data from -pipe option
```

When desiring detailed data on a certain prefetch entry, the verbose option is the best choice. However, with this option, the data outputted per prefetch record can be large. Therefore, the unstructured output may be the preferred choice instead of the CSV formatting. The unstructured mode will represent multiple lines per prefetch record while the CSV mode will force all output to be on one line. Below is an example of how much data can be outputted on a typical prefetch file in verbose mode.

pf - full ver: 1.04; Copyright (c) TZworks LLC
License is authenticated for business use and registered to Dave T; TZworks
time: 03/26/2014 15:24:52 (UTC)

cmdline: c:\Windows\Prefetch\NOTEPAD.EXE-D8414F97.pf -v

Analyzing file: c:\Windows\Prefetch\NOTEPAD.EXE-D8414F97.pf
modified: 03/26/14 15:24:04
accessed: 10/22/12 19:21:09
created: 10/22/12 19:21:09

target file: NOTEPAD.EXE, run 14 times
Last run: 03/26/14 15:24:01.198
Prev run: 03/01/13 22:36:25.733
Prev run: 03/01/13 20:58:39.704
Prev run: 11/10/12 16:29:18.114
Prev run: 11/05/12 19:35:16.302
Prev run: 11/05/12 19:30:15.780
Prev run: 11/05/12 19:29:09.994
Prev run: 11/05/12 18:50:22.405
path: \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\NOTEPAD.EXE

Verbose output with -v switch

Win8 has previous timestamps run as well as the normal last time run

----- files mapped -----

```
001 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\NTDLL.DLL
002 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\NOTEPAD.EXE
003 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\KERNEL32.DLL
004 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\KERNELBASE.DLL
005 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\LOCALE.NLS
006 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
007 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\GDI32.DLL
008 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\USER32.DLL
009 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\MSVCRT.DLL
010 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\COMDLG32.DLL
011 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\SHELL32.DLL
012 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\WINSPPOOL.DRV
013 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\OLE32.DLL
014 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\RPCRT4.DLL
015 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
016 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
017 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
018 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
019 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
020 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
021 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
022 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
023 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
024 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
025 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
026 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
027 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
028 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
029 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
030 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
031 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
032 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
033 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
034 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
035 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
036 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
037 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
038 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
039 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\ADVAPI32.DLL
040 : \DEVICE\HARDDISKVOLUME2\WINDOWS\SYSTEM32\EN-US\PROPSYS.DLL.MUI
041 : \DEVICE\HARDDISKVOLUME2\DUMP\TOOLS\LP64.V.0.54.WIN\RESULTS.CSV
042 : \DEVICE\HARDDISKVOLUME2\WINDOWS\REGISTRATION\R0000000000012.CLB
043 : \DEVICE\HARDDISKVOLUME2\DUMP\TOOLS\JMP64.V.0.17.WIN\RESULTS.CSV
044 : \DEVICE\HARDDISKVOLUME2\USERS\TZLABS\DESKTOP\USB_WIN8.CMD.TXT
045 : \DEVICE\HARDDISKVOLUME2\WINDOWS\REGISTRATION\R000000000000C.CLB
046 : \DEVICE\HARDDISKVOLUME2\PROGRAMDATA\MICROSOFT\WINDOWS\CACHES\{6AF0698E-
047 : \DEVICE\HARDDISKVOLUME3\TZTOOLS\DUMP\JP8_RESULTS.TXT
048 : \DEVICE\HARDDISKVOLUME3\TZTOOLS\DUMP\PF8_RESULTS.TXT
049 : \DEVICE\HARDDISKVOLUME3\TZTOOLS\USB_WIN7.CMD.TXT
```

----- directories mapped -----

```
vol path: \DEVICE\HARDDISKVOLUME2
time created: 10/22/12 20:37:14.444
serial num: e205-d6a3

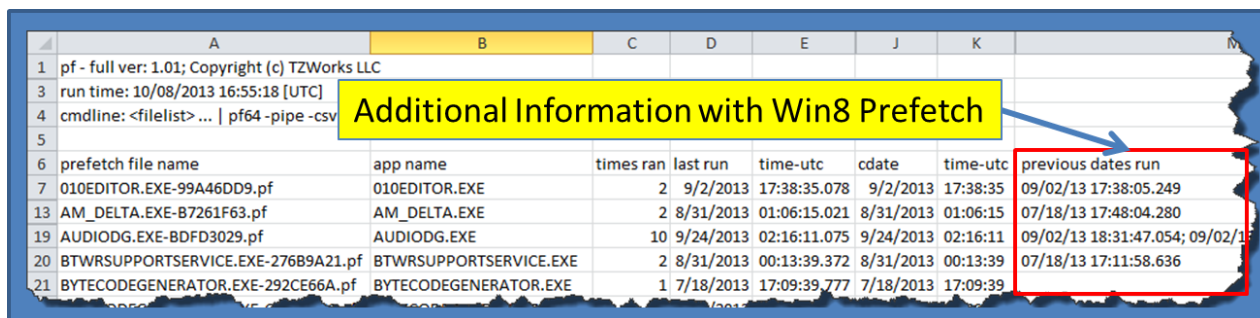
001 : \DEVICE\HARDDISKVOLUME2\PROGRAMDATA
002 : \DEVICE\HARDDISKVOLUME2\PROGRAMDATA\MICROSOFT
003 : \DEVICE\HARDDISKVOLUME2\PROGRAMDATA\MICROSOFT\WINDOWS
004 : \DEVICE\HARDDISKVOLUME2\PROGRAMDATA\MICROSOFT\WINDOWS\CACHES
```

To analyze a bunch of prefetch files in a directory, the **-pipe** switch is available. For example, running this on the c:\Windows\Prefetch directory is a common use of this option. The screenshot below shows this mode using the minimum output (not verbose) and using **-csv**:

3	run time: 10/08/2013 16:44:14 [UTC]											
4	cmdline: <filelist> ... pf64 -pipe -csv											
5												
6	prefetch file name	app name	times ran	last run	time-utc	mdate	time-utc	adate	time-utc	cdate	time-utc	path/appname
8	AM_DELTA_PATCH_1.155.205	AM_DELTA_PATCH.EXE	1	8/14/2013	14:23:47.7	8/14/2013	14:23:57	8/14/2013	14:23:57	8/14/2013	14:23:47	\DEVICE\HARDDISKV
9	AUDIODG.EXE-BDFD3029.pf	AUDIODG.EXE	2099	9/14/2013	18:32:22.5	9/14/2013	18:32:32	2/16/2013	15:52:00	9/14/2013	18:32:22	\DEVICE\HARDDISKV
12	BSCMAKE.EXE-A6DC0796.pf	BSCMAKE.EXE	26	9/14/2013	18:16:14.5	9/14/2013	18:16:16	8/14/2013	16:19:00	9/14/2013	18:16:14	\DEVICE\HARDDISKV
13	CALC.EXE-77FDF17F.pf	CALC.EXE	14	9/12/2013	15:05:00.5	9/12/2013	15:05:10	8/13/2013	18:04:17	9/12/2013	15:05:00	\DEVICE\HARDDISKV
14	CL.EXE-15285A53.pf	CL.EXE	429	9/14/2013	18:15:52.5	9/14/2013	18:15:57	6/25/2013	15:32:09	9/14/2013	18:15:52	\DEVICE\HARDDISKV
15	CL.EXE-18180C6C.pf	CL.EXE	2099	9/13/2013	14:50:09.5	9/13/2013	14:50:10	2/16/2013	16:01:10	9/13/2013	14:50:09	\DEVICE\HARDDISKV
16	CL.EXE-18180C6C.pf	CL.EXE	2099	9/13/2013	14:50:09.5	9/13/2013	14:50:10	2/16/2013	16:01:10	9/13/2013	14:50:09	\DEVICE\HARDDISKV

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.

With Windows 8 prefetch files, older timestamps are recorded in the prefetch file. Therefore, **pf** has an additional option to scan for other times in the prefetch file. This option is now on by default.



prefetch file name	app name	times ran	last run	time-utc	cdate	time-utc	previous dates run
010EDITOR.EXE-99A46DD9.pf	010EDITOR.EXE	2	9/2/2013	17:38:35.078	9/2/2013	17:38:35	09/02/13 17:38:05.249
AM_DELTA.EXE-B7261F63.pf	AM_DELTA.EXE	2	8/31/2013	01:06:15.021	8/31/2013	01:06:15	07/18/13 17:48:04.280
AUDIODG.EXE-BDFD3029.pf	AUDIODG.EXE	10	9/24/2013	02:16:11.075	9/24/2013	02:16:11	09/02/13 18:31:47.054; 09/02/13
BTWRSUPPORTSERVICE.EXE-276B9A21.pf	BTWRSUPPORTSERVICE.EXE	2	8/31/2013	00:13:39.372	8/31/2013	00:13:39	07/18/13 17:11:58.636
BYTECODEGENERATOR.EXE-292CE66A.pf	BYTECODEGENERATOR.EXE	1	7/18/2013	17:09:39.777	7/18/2013	17:09:39	

2.1 Handling Volume Shadow Copies

One can parse the *prefetch* files stored in a Volume Shadow via the **-vss <index of volume shadow>** option. This command will look for prefetch files in the standard location, and parse all the files found in one session.

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 from that above command, 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 **-vss** option. Below is an example of running **pf** with the argument of **-vss 1**. By using the switch **-pf_path** as well, one can see the path of each of the *prefetch* files parsed in the session.

	A	B	C	D	E
4	cmdline pf64 -vss 1 -csv -csv_separator " " -pf_path				
5					
6	prefetch file name	app name	times r	last run	time-utc
7	\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\Prefetch\ATBROKER.EXE-2E15A492.pf	ATBROKER.EXE	1	8/10/2014	13:40:16.093
8	\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\Prefetch\AUDIODG.EXE-BDFD3029.pf	AUDIODG.EXE	322	8/12/2014	02:14:44.741
9	\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\Prefetch\AUTHGEN.EXE-BF7631FD.pf	AUTHGEN.EXE	146	8/12/2014	01:41:38.163
10	\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\Prefetch\CAFAE.EXE-0DA38C11.pf	CAFAE.EXE	2	8/12/2014	01:04:34.151
11	\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\Prefetch\CALC.EXE-77E0E17F.pf	CALC.EXE	1	8/10/2014	17:00:51.480
12	\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\Prefetch\CL.EXE-44444444.pf	CL.EXE	128	8/12/2014	01:41:38.272
13	\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\Prefetch\CL.EXE-44444444.pf	CL.EXE	23	8/12/2014	01:08:11.230
14	\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\Prefetch\CMD.EXE-AC113AA8.pf	CMD.EXE	12	8/12/2014	02:15:29.756
15	\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\Prefetch\CMD.EXE-AC113AA8.pf	CMD.EXE	149	8/12/2014	01:41:38.131
16	\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\Prefetch\COMPMGMTLAUNCHER.EXE-D8C6	COMPMGMTLAUNCHER.EXE	1	8/12/2014	02:17:15.202
17	\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\Prefetch\CONHOST.EXE-1F3E9D7E.pf	CONHOST.EXE	315	8/12/2014	02:15:29.766
18	\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\Prefetch\CONSENT.EXE-5318D9EA.pf	CONSENT.EXE	141	8/12/2014	02:20:04.281
19	\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\Prefetch\CONTROL.EXE-817F8F1D.pf	CONTROL.EXE	3	8/12/2014	02:20:47.699
20	\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\Prefetch\CSC.EXE-BE9AC2DF.pf	CSC.EXE	4	8/10/2014	18:43:32.912

-vss 1 = enumerate prefetch folder and parse all prefetch files in VolumeShadowCopy1

In addition to the **-vss** option, 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 *prefetch* path/file. For example, to access the `CMD.EXE-00F4F355.pf` file located in the *prefetch* folder from the *HarddiskVolumeShadowCopy1*, the following syntax can be used:

pf %vss%1\Windows\Prefetch\CMD.EXE-00F4F355.pf -v > out.txt

2.2 Other Output Options

Added with version 1.04 is the ability to output results in Log2Timeline and Body-file formats. The options are **-csvl2t** and **-bodyfile**, respectively. Below is a same output of using **-csvl2t**.

pf - full ver: 1.04; Copyright (c) TZWorks LLC License is authenticated for business use and registered to Dave T; TZWorks run time: 03/25/2014 17:21:10 [UTC] cmdline: <filelist> ... pf64 -pipe -csvl2t -hostname testbox								
date	time	timez	MACB	source	sourcetype	host	short	desc
3/25/2014	16:30:56.000	UTC	.A.B	PREFETCH	Win8 Prefetch	testbox	NOTEPAD.EXE-D8414F97.pf [NOTEPAD.EXE executed]	Module list: NTDLL.DLL; NOTEPAD.EXE; KE
3/25/2014	16:30:36.000	UTC	.A.B	PREFETCH	Win8 Prefetch	testbox	CMD.EXE-AC113AA8.pf [CMD.EXE executed]	Module list: NTDLL.DLL; WOW64.DLL; WO
3/25/2014	16:29:59.000	UTC	.A.B	PREFETCH	Win8 Prefetch	testbox	010EDITORWIN32INSTALLER50.EXE-C2A5A109.pf [010EDIT	Module list: NTDLL.DLL; WOW64.DLL; WO
3/25/2014	16:28:48.000	UTC	.A.B	PREFETCH	Win8 Prefetch	testbox	AM_BASE_PATCH1.EXE-FC84E7C0.pf [AM_BASE_PATCH1.i	Module list: NTDLL.DLL; AM_BASE_PATCH
3/25/2014	16:28:48.000	UTC	.A.B	PREFETCH	Win8 Prefetch	testbox	AM_BASE_PATCH_129.EXE-10518282.pf [AM_BASE_PATCH	Module list: NTDLL.DLL; AM_BASE_PATCH
3/25/2014	16:28:48.000	UTC	.A.B	PREFETCH	Win8 Prefetch	testbox	AM_DELTA.EXE-B7261F63.pf [AM_DELTA.EXE executed]	Module list: MPSIGSTUB.EXE; NTDLL.DLL; A
3/25/2014	14:29:48.000	UTC	.A.B	PREFETCH	Vista/Win7 Prefetch	testbox	DEFRAG.EXE-738093E8.pf [DEFRAG.EXE executed]	Module list: VIRTDISK.DLL; FLTLib.DLL; SX
10/30/2013	13:51:21.000	UTC	M...	PREFETCH	Vista/Win7 Prefetch	testbox	DEFRAG.EXE-738093E8.pf [DEFRAG.EXE executed]	Module list: VIRTDISK.DLL; FLTLib.DLL; SX
9/2/2013	17:46:44.000	UTC	M...	PREFETCH	Win8 Prefetch	testbox	NOTEPAD.EXE-D8414F97.pf [NOTEPAD.EXE executed]	Module list: NTDLL.DLL; NOTEPAD.EXE; KE
9/2/2013	17:46:33.322	UTC	.C.	PREFETCH	Win8 Prefetch	testbox	NOTEPAD.EXE-D8414F97.pf [NOTEPAD.EXE executed]	Module list: NTDLL.DLL; NOTEPAD.EXE; KE
9/2/2013	17:37:46.884	UTC	MAC.	PREFETCH	Win8 Prefetch	testbox	NOTEPAD.EXE-D8414F97.pf [NOTEPAD.EXE executed]	Modules for last run are at record w/ same
9/2/2013	17:37:26.000	UTC	M...	PREFETCH	Win8 Prefetch	testbox	010EDITORWIN32INSTALLER50.EXE-C2A5A109.pf [010EDIT	Module list: NTDLL.DLL; WOW64.DLL; WO
9/2/2013	17:37:14.000	UTC	C.	PREFETCH	Win8 Prefetch	testbox	010EDITORWIN32INSTALLER50.EXE-C2A5A109.pf [010EDIT	Module list: NTDLL.DLL; WOW64.DLL; WO

Normally, the Log2Timeline format has at most 4 entries when considering the MACB and standard information data in a file's timestamp, however with *prefetch* files, there are some additional entries that may (or may not) be added, specifically with Win8 prefetch files. Recall that Windows 8 prefetch includes additional information that has previous timestamps on when the target file was run. To use these additional timestamps in the Log2Timeline output, **pf** will create new entries. Using the example given in a previous section, when **pf** was run against the notepad prefetch file in the verbose, long output mode, the same file parsed using the **-csvl2t** option will yield the following output. One can see, in addition to the MACB entries, where the 'C' is used for the last run timestamp and MAB are used for the file standard attribute timestamps, there are 7 additional entries for the 'previously run' timestamps, for a total of 11 entries just for one file.

	date	time	timez	MACB	source	type	short
1	pf - full ver: 1.04; Copyright (c) TZWorks LLC						
2	d registered to Dave T; TZWorks						
3	Timestamps from Std Info data						
4	cmdline: pf64 c:\Windows\Prefetch\NOTEPAD.EXE-D8414F97.pf -csvl2t -hostname abs_64test -timeformat hh:mm:ss						
5							
6	3/26/2014	15:24:04	UTC	M...	PREFETCH	Last Modified	NOTEPAD.EXE; KERNEL32.DLL; KERNELBASE.DLL; LOCALI
7	3/26/2014	15:24:01	UTC	..C.	PREFETCH	Last Run	NOTEPAD - Module list: NTDLL.DLL; NOTEPAD.EXE; KERNEL32.DLL; KERNELBASE.DLL; LOCALI
8	3/1/2013	22:36:25	UTC	MAC.	PREFETCH	Modified; Access; Run	NOTEPAD - Modules for last run are at record w/ same name at time: 03/26/14 15:24:01
9	3/1/2013	20:58:39	UTC	MAC.	PREFETCH	Modified; Access; Run	record w/ same name at time: 03/26/14 15:24:01
10	11/10/2012	16:29:18	UTC	MAC.	PREFETCH	Modified; Access; Run	record w/ same name at time: 03/26/14 15:24:01
11	11/5/2012	19:35:16	UTC	MAC.	PREFETCH	Modified; Access; Run	record w/ same name at time: 03/26/14 15:24:01
12	11/5/2012	19:30:15	UTC	MAC.	PREFETCH	Modified; Access; Run	record w/ same name at time: 03/26/14 15:24:01
13	11/5/2012	19:29:09	UTC	MAC.	PREFETCH	Modified; Access; Run	record w/ same name at time: 03/26/14 15:24:01
14	11/5/2012	18:50:22	UTC	MAC.	PREFETCH	Modified; Access; Run	record w/ same name at time: 03/26/14 15:24:01
15	10/22/2012	19:21:09	UTC	..A.B	PREFETCH	Last Access; Created	NOTEPAD - Modules for last run are at record w/ same name at time: 03/26/14 15:24:01
16							NOTEPAD - Module list: NTDLL.DLL; NOTEPAD.EXE; KERNEL32.DLL; KERNELBASE.DLL; LOCALI

3 Known Issues

When using a spreadsheet application (such as Microsoft Excel) to view the output of CSV type results, sometimes the cell data will wrap to a new line. This is because a cell in the spreadsheet has a maximum number of characters it can handle. Typically, this is 1024 characters for the newer spreadsheet applications. Therefore, when parsing very large prefetch files, the number of modules loaded by an application can be large and consequently will have an output of greater than 1024 characters for a module list. When this happens, the data will be rendered on multiple cells/lines.

For Linux and Mac builds, the *file cdate & time* reported in the output is the date and time of the metadata change of the file (not the creation time of the file). This behavior is different in Windows, where the *file cdate & time* reported in the output is the date and time of the creation of the file.

4 Available Options

The options labeled as 'Extra' require a separate license for them to be unlocked.

Option	Description
-verbose	Verbose output. This option will include the module listing within the output. This was the default option when parsing single pf files (not using the -pipe command) that produced multiple lines in the output. This was added for additional CSV functionality.
-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.
-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> .
-vss	Experimental. Parse all prefetch files from specified Volume Shadow. The syntax is -vss <index number of shadow copy> . Only applies to Windows Vista, Win7, Win8 and beyond. Does not apply to Windows XP.
-prevtimes	[Deprecated – on by default if using a commercial license]. Scans the prefetch file for other dates that may be present. For example, Windows 8

	has other dates that document previous run times besides the last run time.
<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>-utf8_bom</i>	All output is in Unicode UTF-8 format. If desired, one can prefix an UTF-8 byte order mark to the output using this option.

5 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.

5.1 *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.

6 References

1. Windows Forensic Analysis, by Harlan Carvey, Syngress 2007
2. MSDN, "Windows XP: Kernel Improvements Create a More Robust, Powerful, and Scalable OS", discussion on prefetch, by Mark Russinovich and David Solomon
3. Wikipedia, the free encyclopedia. Prefetch topic.
4. Pre-fetching of pages prior to a hard page fault sequence, US patent 6,633,968, dtd 10/14/03, by Zwiegincew, et al.