TZWorks® Windows 'index.dat' (id) Parser Users Guide

Abstract

id is a standalone, command-line tool that can parse Windows 'index.dat' type files. The 'index.dat' file functions as a repository for web URL's, search queries and recently opened files. ***id*** can operate on a live volume or a *VMWare* volume. ***id*** runs on Windows, Linux and Mac OS-X.

www.tzworks.net

Contact Info: info@tzworks.net

*Document applies to v0.77 of **id***

Updated: Feb 19, 2018

Table of Contents

TZWorks® Index.dat Parser (id) Users Guide

Webpage: http://www.tzworks.net/prototype_page.php?proto_id=6
Contact Information: info@tzworks.net

1 Introduction

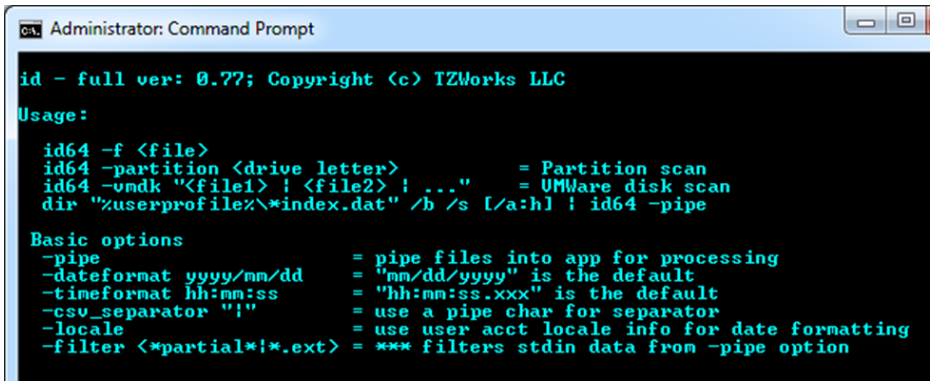
id is a command line version of a *index.dat* parser. While there are other *index.dat* parsers freely available, we wanted a tool that was fast, cross-platform and flexible in architecture for additional enhancements.

From Wikipedia, "the *index.dat* file functions as an active database, which runs as long as a user is logged on... It functions as a repository of redundant information, such as web URLs, search queries and recently opened files. Its role is similar to that of an index file in the field of databases, where a technique called 'indexing' stores the contents of a database in a different order to help speed up query responses".

Since the Windows operating system uses these files as databases, they are locked. This makes it is difficult to remove them. Coupled with the fact they (a) record user history, (b) store URLs that are visited and (c) cookies, these files are useful in computer forensics.

2 How to Use *id*

Below is a screen shot of the command menu that shows which options are available. From the options, one can: (a) parse an individual *index.dat* file, (b) operate on an entire volume (or VMWare VMDK volume) scanning for *index.dat* signatures and parsing them, or (c) handling a collection of *index.dat* files in one session by piping in the desired files to parse via STDIN (standard input).



2.1 Some Common Locations of index.dat files and what their contents contain

2.1.1 Windows XP and 2003 (Pre-Vista) Operating Systems

History files
%userprofile%\Local Settings\History\History.IE5\
%userprofile%\Local Settings\History\History.IE5\MSHist*\

Cache
%userprofile%\Local Settings\Temporary Internet Files\Content.IE5\
%userprofile%\Local Settings\Application Data\Microsoft\Feeds Cache\
%userprofile%\Local Settings\Application Data\Microsoft\Internet Explorer\UserData\
%userprofile%\Local Settings\Application Data\Microsoft\Internet Explorer\DOMStore\
%userprofile%\IECompatCache\
%userprofile%\IETldCache\
%userprofile%\PrivacIE\
%userprofile%\UserData\

Cookies
%userprofile%\Cookies

2.1.2 Windows Vista and Later Windows Operating Systems

History files
%userprofile%\AppData\Local\Microsoft\Windows\History\History.IE5
%userprofile%\AppData\Local\Microsoft\Windows\History\History.IE5\MSHist*\
%userprofile%\AppData\Local\Microsoft\Windows\History\Low\History.IE5 (unprivileged)

Cache
%userprofile%\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5
%userprofile%\AppData\Local\Microsoft\Windows\Temporary Internet Files\Low\Content.IE5
%userprofile%\AppData\Local\Microsoft\Windows\Feeds Cache\
%userprofile%\AppData\Roaming\Microsoft\Windows\IECompatCache\
%userprofile%\AppData\Roaming\Microsoft\Windows\IECompatCache\Low\
%userprofile%\AppData\Roaming\Microsoft\Windows\IETldCache\
%userprofile%\AppData\Roaming\Microsoft\Windows\IETldCache\Low\
%userprofile%\AppData\Roaming\Microsoft\Windows\IETldCache\Low\

Cookies
%userprofile%\AppData\Roaming\Microsoft\Windows\Cookies\
%userprofile%\AppData\Roaming\Microsoft\Windows\Cookies\Low\
%userprofile%\AppData\LocalLow\Microsoft\Internet Explorer\DOMStore\
%userprofile%\AppData\Roaming\Microsoft\Internet Explorer\UserData\Low\

2.2 Piping Files into *id*

When trying to pipe the contents of a directory listing using the Windows ***dir*** command, ensure the correct switches are used with the ***dir*** to enumerate the index.dat files. Depending on Windows explorer viewing settings, you may need to use the attribute to display hidden files.

2.2.1 Windows XP Example:

To pipe all occurrences of index.dat within the 'Documents and Settings' directory, do the following: If the index.dat has the hidden attribute set, then use:

dir "c:\Documents and Settings*index.dat" /a:h /b /s | id.exe -pipe

If the index.dat files don't have the hidden attribute set, then use:

dir "c:\Documents and Settings*index.dat" /b /s | id.exe –pipe

2.2.2 Windows Vista, Win7, Win8 Example:

To pipe all occurrences of index.dat within the 'Users' directory, do the following: The case shown assumes the index.dat files have the hidden attribute set.

dir "c:\Users*index.dat" /a:h /b /s | id.exe -pipe

dir "c:\Users*index.dat" /b /s | id.exe -pipe

To redirect output from STDOUT (standard output) to a file

dir "c:\Users*index.dat" /a:h /b /s | id.exe -pipe > c:\dump\parsed.txt

dir "c:\Users*index.dat" /b /s | id.exe -pipe > c:\dump\parsed.txt

2.2.3 Linux/Mac OS-X Example:

When trying to pipe a set of index.dat files in various directories, one can use the 'find' command and pipe the output into **id** and redirect the final output to the results file, like this:

```
find /home/<some dir> -name *.dat -type f | ./id -pipe > results
```

2.3 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 file. For example, to access an *index.dat* file located in the *testuser* account from the *HarddiskVolumeShadowCopy1*, the following syntax can be used:

```
id -f "%vss%1\Users\testuser\AppDataRoaming\Microsoft\Windows\Temporary Internet Files\Content.IE5\index.dat" > out.txt
```

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

```
vssadmin list shadows
```

To filter some of the extraneous detail, type

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

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

```
Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1
Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2
. . .
```

From the above, notice the number after the word *HarddiskvolumeShadowCopy*. It is this number that is appended to the ***%vss%*** keyword.

3 Available Options

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

Option	Extra	Description
-f		Specifies the file to parse. Syntax is ***-f <filename>***
-locale		Use the system local of the user account to perform date formatting. [this option is very experimental]. Invoking this option for a en-US system would be

		MM/DD/YYYY. For a fr-FR system, the format should be DD/MM/YYYY. Time remains at HH:MM:SS.MSEC, independent of the system locale.
-pipe		Used to pipe files into the tool via STDIN (standard input). Each file passed in is parsed in sequence.
-filter	***	Filters data passed in via STDIN via the ***-pipe*** option. The syntax is ***-filter <"*.ext | *partialname* | ...">***. The wildcard character '*' is restricted to either before the name or after the name.
-partition		Windows only option. Extract artifacts from a mounted Windows volume. The syntax is ***-partition <drive letter>***.
-vmdk		Extract artifacts from a VMWare monolithic NTFS formatted volume. The syntax is ***-vmdk "disk"***. For a collection of VMWare disks that include snapshots, one can use the following syntax: ***-vmdk "disk1 | disk2 | ..."***
-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 "mm/dd/yyyy"***. This allows more flexibility for a desired format. For example, one can use this to show year first, via ***"yyyy/mm/dd"***, or day first, via ***"dd/mm/yyyy"***, or only show 2 digit years, via the ***"mm/dd/yy"***. The restriction with this option is the forward slash (/) 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.xxxxxxxxx"***, or no fractional seconds, via ***"hh:mm:ss"***. The restrictions with this option is a colon (:) symbol needs to separate hours, minutes and seconds, a period (.) symbol needs to separate the seconds and fractional seconds, and the repeating symbol 'x' is used to represent number of fractional seconds. (Note: the fractional seconds applies only to those time formats that have the appropriate precision available. The Windows internal filetime has, for example, 100 nsec unit precision available. The DOS time format and the UNIX 'time_t' format, however, have no fractional seconds). Some of the times represented by this tool may use a time format without fractional seconds, and therefore, will not

		show a greater precision beyond seconds when using this option.

4 Authentication and the License File

This tool has authentication built into the binary. There are two authentication mechanisms: (a) the digital certificate embedded into the binary and (b) the runtime authentication. For the first method, only the Windows and Mac OS-X (if available) versions have been signed by an X-509 digital code signing certificate, which is validated by Windows (or OS-X) during operation. If the binary has been tampered with, the digital certificate will be invalidated.

For the second (runtime authentication) method, the authentication does two things: (a) validates that the tool has a valid license and (b) validates the tool's binary has not been corrupted. 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. The runtime binary validation hashes the executable that is running and fails the authentication if it detects any modifications.

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. Wikipedia, the free encyclopedia. Windows index.dat files topic.
2. Forensic Analysis of Internet Explorer Activity Files, Keith Jones, 3/19/03
3. Is Your index.dat File LEAKing, Mike Murr, 9/18/09
4. SANS Computer Forensics Essentials [408 course], Rob Lee, 1/2010