dfImageTools

Release 20220312

unknown

Mar 12, 2022

CONTENTS

Collection of tools to process storage media images.

The source code is available from the project page.

CHAPTER

ONE

GETTING STARTED

To be able to use dfImageTools you first need to install it. There are multiple ways to install dfImageTools, check the following instructions for more detail.

1.1 Installation instructions

1.1.1 pip

Note that using pip outside virtualenv is not recommended since it ignores your systems package manager. If you aren't comfortable debugging package installation issues, this is not the option for you.

Create and activate a virtualenv:

```
virtualenv dfimagetoolsenv
cd dfimagetoolsenv
source ./bin/activate
```

Upgrade pip and install dfimagetools dependencies:

```
pip install --upgrade pip
pip install dfimagetools
```

To deactivate the virtualenv run:

```
deactivate
```

1.1.2 Ubuntu 18.04 and 20.04 LTS

To install dfimagetools from the GIFT Personal Package Archive (PPA):

```
sudo add-apt-repository ppa:gift/stable
```

Update and install dfimagetools:

```
sudo apt-get update
sudo apt-get install python3-dfimagetools dfimagetools-tools
```

1.1.3 Windows

The l2tbinaries contains the necessary packages for running dfimagetools. l2tbinaries provides the following branches:

- main; branch intended for the "packaged release" of dfimagetools and dependencies;
- dev; branch intended for the "development release" of dfimagetools;
- testing; branch intended for testing newly created packages.

The l2tdevtools project provides an update script to ease the process of keeping the dependencies up to date.

The script requires pywin32 and Python WMI.

To install the release versions of the dependencies run:

```
set PYTHONPATH=.

C:\Python3\python.exe tools\update.py --preset dfimagetools
```

CHAPTER

TWO

BODYFILE FORMAT

The bodyfile (or body file) format is an output format, as far as known, introduced by the SleuthKit. SleuthKit tools such as fls or ils, use a bodyfile for intermediate storage. These bodyfiles are then provided as input to the mactime tool.

The bodyfile format has been adopted by many other, non-SleuthKit tools, and does not appear to have a strict definition. This document explains the implementation used by the dfImageTools project.

The dfImageTools project uses a bodyfile format that has been derived from the format used by SleuthKit 3.0 and later. Changes have been made to overcome several shortcomings of the original format.

A bodyfile consists of one or more lines with 11 pipe-character ('|') delimited values. The SleuthKit documentation defines these values as:

```
MD5|name|inode|mode_as_string|UID|GID|size|atime|mtime|ctime|crtime
```

2.1 MD5 value

The SleuthKit documentation does not define the MD5 values. From observations the following convention is used:

- '0' if "hashing" is disabled;
- '00000000000000000000000000000000' if "hashing" is enabled but no MD5 was calculated;
- '[0-9a-f]{32}' if a MD5 was calculated.

2.2 Name value

The name value typically contains a full path of the file entry, but it can also contain a symobolic link target using the convention:

```
${PATH} -> ${SYMBOLIC_LINK_TARGET}
```

Bodyfile entries where the time values are extracted for an NTFS $FILE_NAME attribute the '($FILE_NAME)' suffix is added to the name value.

```
${PATH} (\$FILE_NAME)
```

Note that at the moment the `list_file_entries.py` script does not combine a symbolic link target and '($FILE_NAME)' suffix.

The `list_file_entries.py` script always uses forward slash ('/') as the path segment separator, even for NTFS. The following characters are escaped with a backslash ('\'):

- U+0000 - U+0019 (C0 control codes, non-printable)
- U+002f (forward slash '/', used as path segment separator)
- U+003a (colon ':', used as data stream separator)
- U+005c (backslash '\', used as escape character)
- U+007c (pipe '|', used as value delimiter)
- U+007f (delete, non-printable)
- U+0080 - U+009f (C1 control codes, non-printable)

Paths are prefixed with a partition or volume indicator if the `list_file_entries.py` script is used to list multiple partitions and/or volumes at once.

2.3 Inode value

The inode value contains an unique identifier of the file entry within the file system, which for some file systems is the inode number.

For NTFS the convention `${MFT_ENTRY}-${SEQUENCE_NUMBER}` is used instead of the non-portable metadata address used by the SleuthKit tools.

2.4 Mode_as_string value

The mode_as_string value contains a POSIX file mode represented as a string, for example 'drwxr-xr-x'.

The first character represents the file entry type:

- '-' to indicate a "regular" file (S_IFREG) or unknown type
- 'b' to indicate a block device (S_IFBLK)
- 'c' to indicate a character device (S_IFCHR)
- 'd' to indicate a directory (S_IFDIR)
- 'l' to indicate a symbolic link (S_IFLNK)
- 'p' to indicate a named-pipe (S_IFIFO)
- 's' to indicate a socket (S_IFSOCK)

The SleuthKit specific 'r' type indicator is not used by the dfImageTools project.

The remaining characters are the read, write and execute permissions for owner, group and other.

The SleuthKit specific `[-dlr]/` prefix is not used by the dfImageTools project.

2.4.1 NTFS

For NTFS dfImageTools uses the following approximation to generate a mode_as_string value.

The first character represents the file entry type:

- '-' to indicate a "regular" file or unknown type
- 'd' to indicate a directory, if the file entry has an $I30 index and is not a symbolic link
- 'l' to indicate a symbolic link, if the file entry has a $REPARSE_POINT attribute with tag 0xa000000c

The remaining characters are based on the file attribute flags and will be 'r-xr-xr-x' if FILE_ATTRIBUTE_READONLY or FILE_ATTRIBUTE_SYSTEM is set or 'rwxrwxrwx' otherwise.

2.5 Time values

Time values are provided as a number of seconds since January 1, 1970 00:00:00 (epoch) without a time zone, where negative time values predate the epoch. A fraction of second is provided if the original time value has a higher datetime value granularity.

2.6 Also see

- Forensics wiki: Bodyfile
- SleuthKit: Body file

CHAPTER

THREE

DFIMAGETOOLS PACKAGE

3.1 Submodules

3.2 dfimagetools.artifact_filters module

Helper for filtering based on artifact definitions.

class dfimagetools.artifact_filters.**ArtifactDefinitionFiltersGenerator**(*artifacts_registry*, *environment_variables*, *user_accounts*)

Bases: `object`

Generator of filters based on artifact definitions.

GetFindSpecs(*names*)

Retrieves find specifications for one or more artifact definitions.

Parameters **names** (*list[str]*) – names of the artifact definitions to filter on.

Yields *dfvfs.FindSpec* – file system (dfVFS) find specification.

3.3 dfimagetools.bodyfile module

Helper for generating bodyfile entries.

class dfimagetools.bodyfile.**BodyfileGenerator**

Bases: `object`

Bodyfile generator.

GetEntries(*file_entry*, *path_segments*)

Retrieves bodyfile entry representations of a file entry.

Parameters

- **file_entry** (*dfvfs.FileEntry*) – file entry.
- **path_segments** (*str*) – path segments of the full path of the file entry.

Yields *str* – bodyfile entry.

3.4 dfimagetools.data_stream_writer module

Helper to write data streams.

class dfimagetools.data_stream_writer.**DataStreamWriter**
Bases: object

Data stream writer.

GetDisplayPath(*source_path_segments*, *source_data_stream_name*)
Retrieves a path to display.

Parameters

- **source_path_segments** (*list[str]*) – path segment of the source file.
- **source_data_stream_name** (*str*) – name of the data stream of the source file.

Returns display path.

Return type str

GetSanitizedPath(*source_path_segments*, *source_data_stream_name*, *target_path*)
Retrieves santized a path.

This function replaces non-printable and other invalid path characters with an underscore "_".

Parameters

- **source_path_segments** (*list[str]*) – path segment of the source file.
- **source_data_stream_name** (*str*) – name of the data stream of the source file.
- **target_path** (*str*) – path of the target directory.

Returns sanitized path.

Return type str

WriteDataStream(*file_entry*, *data_stream_name*, *destination_path*)
Writes the contents of the source data stream to a destination file.

Note that this function will overwrite an existing file.

Parameters

- **file_entry** (*dfvfs.FileEntry*) – file entry whose content is to be written.
- **data_stream_name** (*str*) – name of the data stream whose content is to be written.
- **destination_path** (*str*) – path of the destination file.

3.5 dfimagetools.decorators module

Function decorators.

dfimagetools.decorators.**deprecated**(*function*)
Decorator to mark functions or methods as deprecated.

3.6 dfimagetools.environment_variables module

Windows environment variables collector.

class dfimagetools.environment_variables.**WindowsEnvironmentVariablesCollector**
Bases: object

Windows environment variables collector.

Collect(*registry*)
Collects environment variables.

Parameters **registry** (*dfwinreg.WinRegistry*) – Windows Registry.

Yields *EnvironmentVariable* – an environment variable.

3.7 dfimagetools.file_entry_lister module

Helper to list file entries.

class dfimagetools.file_entry_lister.**FileEntryLister**(**args: Any*, ***kwargs: Any*)
Bases: dfvfs.helpers.volume_scanner.VolumeScanner

File entry lister.

GetBodyfileEntries(***kwargs*)
Retrieves bodyfile entry representations of a file entry.

Parameters

- **file_entry** (*dfvfs.FileEntry*) – file entry.
- **path_segments** (*str*) – path segments of the full path of the file entry.

Returns bodyfile entry generator.

Return type generator[str]

GetWindowsDirectory(*base_path_spec*)
Retrieves the Windows directory from the base path specification.

Parameters **base_path_spec** (*dfvfs.PathSpec*) – source path specification.

Returns path of the Windows directory or None if not available.

Return type str

ListFileEntries(*base_path_specs*)
Lists file entries in the base path specifications.

Parameters **base_path_specs** (*list[dfvfs.PathSpec]*) – source path specifications.

Yields *tuple[dfvfs.FileEntry, list[str]]* – file entry and path segments.

ListFileEntriesWithFindSpecs(*base_path_specs*, *find_specs*)
Lists file entries in the base path specifications.

This method filters file entries based on the find specifications.

Parameters

- **base_path_specs** (*list[dfvfs.PathSpec]*) – source path specification.
- **find_specs** (*list[dfvfs.FindSpec]*) – find specifications.

Yields *tuple[dfvfs.FileEntry, list[str]]* – file entry and path segments.

3.8 dfimagetools.helpers module

Helper functions for CLI tools.

dfimagetools.helpers.**SetDFVFSBackEnd**(*back_end*)
Sets the dfVFS back-end.

Parameters **back_end** (*str*) – dfVFS back-end.

3.9 dfimagetools.path_resolver module

Helper for resolving paths.

class dfimagetools.path_resolver.**PathResolver**
Bases: object

Path resolver.

ExpandEnvironmentVariables(*path*, *path_separator*, *environment_variables*)
Expands environment variables.

Parameters

- **path** (*str*) – path with environment variables.
- **path_separator** (*str*) – path segment separator.
- **environment_variables** (*list[EnvironmentVariable]*) – environment variables.

Returns path with environment variables expanded.

Return type str

ExpandGlobStars(*path*, *path_separator*)
Expands globstars "**".

A globstar "**" will recursively match all files and zero or more directories and subdirectories.

By default the maximum recursion depth is 10 subdirectories, a numeric values after the globstar, such as "**5", can be used to define the maximum recursion depth.

Parameters

- **path** (*str*) – path with globstars.
- **path_separator** (*str*) – path segment separator.

Returns path with seperate globs for every globstar.

Return type str

ExpandUsersVariable(*path*, *path_separator*, *user_accounts*)
Expands a users variable, such as %%users.appdata%%.

Parameters

- **path** (*str*) – path with users variable.
- **path_separator** (*str*) – path segment separator.
- **user_accounts** (*list[UserAccount]*) – user accounts.

Returns paths for which the users variables have been expanded.

Return type list[str]

3.10 dfimagetools.resources module

Various resource classes.

class dfimagetools.resources.**EnvironmentVariable**(*case_sensitive=True*, *name=None*, *value=None*)

Bases: object

Environment variable.

case_sensitive
True if environment variable name is case sensitive.

Type bool

name
environment variable name such as "SystemRoot" as in "%SystemRoot%" or "HOME" as in "$HOME".

Type str

value
environment variable value such as "C:Windows" or "/home/user".

Type str

class dfimagetools.resources.**UserAccount**(*full_name=None*, *group_identifier=None*, *identifier=None*, *user_directory=None*, *user_directory_path_separator='/'*, *username=None*)

Bases: object

User account.

full_name
name describing the user.

Type str

group_identifier
identifier of the primary group the user is part of.

Type str

identifier
user identifier.

Type str

user_directory
path of the user (or home or profile) directory.

Type str

user_directory_path_separator
path segment separator of the user directory.

Type str

username
name uniquely identifying the user.

Type str

3.11 dfimagetools.windows_registry module

Helpers to collect information from the Windows Registry.

class dfimagetools.windows_registry.**StorageMediaImageWindowsRegistryFileReader**(**args: Any*, ***kwargs: Any*)

Bases: dfwinreg.interface.WinRegistryFileReader

Storage media image Windows Registry file reader.

Open(*path*, *ascii_codepage='cp1252'*)

Opens the Windows Registry file specified by the path.

Parameters

- **path** (*str*) – path of the Windows Registry file. The path is a Windows path relative to the root of the file system that contains the specific Windows Registry file. E.g. C:WindowsSystem32configSYSTEM
- **ascii_codepage** (*Optional[str]*) – ASCII string codepage.

Returns

Windows Registry file or None if the file cannot be opened.

Return type WinRegistryFile

class dfimagetools.windows_registry.**WindowsRegistryCollector**(*path_spec*, *windows_directory*)

Bases: object

Windows Registry collector.

CollectSystemEnvironmentVariables()

Collects the system environment variables.

Returns environment variables.

Return type list[*EnvironmentVariable*]

class dfimagetools.windows_registry.**WindowsRegistryFile**(**args: Any*, ***kwargs: Any*)

Bases: dfwinreg.interface.WinRegistryFile

Windows Registry file.

This class manages a Windows Registry file-like object.

Close()

Closes the Windows Registry file.

GetKeyByPath(*key_path*)

Retrieves the key for a specific path.

Parameters **key_path** (*str*) – Windows Registry key path.

Returns Windows Registry key or None if not available.

Return type WinRegistryKey

GetRootKey()

Retrieves the root key.

Returns Windows Registry root key or None if not available.

Return type WinRegistryKey

Open(*file_object*)
Opens the Windows Registry file using a file-like object.

Parameters **file_object** (*file*) – file-like object.

Raises

- **IOError** – if the Windows Registry file cannot be opened.
- **OSError** – if the Windows Registry file cannot be opened.

SetKeyPathPrefix(*key_path_prefix*)
Sets the Window Registry key path prefix.

Parameters **key_path_prefix** (*str*) – Windows Registry key path prefix.

3.12 Module contents

Collection of tools to process storage media images.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex

PYTHON MODULE INDEX

d

INDEX