Improving the `bodyfile` format: a suggestion

Jan Starke <jan.starke@t-systems.com>

April 2, 2022

Abstract

In this paper we describe the famous `bodyfile` format and, especially, a lot of its shortcomings. But because this format is widely known and used, we discuss the benefits of `bodyfile` and take a look at the requirements that an improved fileformat should fulfill.

1 The `bodyfile` format

A lot of forensic tools collect data which are related with time information. We can see sich information as sets of events. Those data must be related to each other, which normally means that they are ordered chronologically, so that one can see which event occured after another. Unfortunately, a lot of events are extracted from many different sources, such as various filesystems, the Windows Registry, several log files and so on. Common tools which work with the `bodyfile` format include

- `fls`, which is part of The Sleuth Kit ([1])
- `evtx2bodyfile` can be used to convert Windows Event Log (`evtx`) files into `bodyfile`
- `mactime` is the standard to which is used to sort `bodyfile`s and generate human readable text- as well as csv-files
- `plaso`, which has its own SQLite based timeline format, is also able to consume `bodyfile`s

So, it seems that we have a common standard of how to describe events and their relation in time. It seems that we can use this standard to correlate a lot of different event sources to find interdependencies between events from different sources. It seems that because of the broad use, `bodyfile` is as base for sound forensic work. We will see that this is not the case.

1.1 Format of `bodyfile` files

The `bodyfile` format is text based, which means that nearly every commandline tool and text editor can work with it. A file in this format consists of a list von lines, where each line describes a single file which may be found on the filesystem under analysis, together with some metainformation about the file as well as the timestamps related to the file. Each line consists of a list of fields which are separated by the pipe character. The following fields are being used:

`MD5` MD5 hashsum of the file, or the number 0

`name` name of the file. Normally the fully qualified path is specified here

`inode` inode number or, on systems where no inodes are being used

`mode` describes *who* is permitted to do *what* with this file. This field must be specified in the common unixoid syntax, e.g. `rwxr-xr-x` specifies that the owner of the file is allowed to read (`r`), write (`w`) and execute (`x`) the file, while members of the owning group as well as all others may read or execute the file, but are not allowed to write it. This field only makes sense when the filesystem has such a simple permission theme. As far as it comes to access control lists, this syntax is not sufficient anymore.

`UID` (numerical) identifier of the files owner, or 0 if this cannot be specified. For example, Windows dos not have numerical user identifiers, but uses Security Identifiers[1] to identify users.

`GID` (numerical) identifier of the group which owns the file. the same restriction as with the *UID* applies here as well.

`size` size of the file in bytes

`atime` specifies at which time the file has lastly been *accessed*, as epoch time (number of seconds since January 1st, 1970). It is possible that this fields contains -1, which means that this value must be ignored. The same also applies to the other three timestamps.

`mtime` time of the last *modification* of the file, as epoch time

`ctime` time of the last *change* to the file's *meta-information*, as epoch time

`crtime` time of the files *creation* (birthtime), as epoch time

[1] `https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/security-identifiers`

1.2 What's wrong with `bodyfile`?

A lot of people have found one or more simple problems which occured during their work with the `bodyfile` format. Fortunately, a lot of those problems are collected at the forensikswiki page ([3]):

1.2.1 Missing timezone information

Despite a `bodyfile` entry has multiple timestamps, there is no information about from which timezone these timestamps has been collected. Also, there is no information if daylight savings should be applied. Most unixoid systems store their timestamps in Universal Time Coordinated (UTC), but it is – especially in forensic software – bad practice to silently assume such information. There are a lot of timestamp formats (e.g. [2]) which also contain a timezone information and should better be used

1.2.2 Unclear file encoding

When `bodyfile` was created, textfiles where basically ASCII encoded files. Also, filesystems used ASCII to encode filenames, so there was no need to support other encodings. But modern operating system support a lot of different encoding to be efficiently used by native speakers all over the world. So, a textfile now could be encoded in ASCII, UTF-8, UTF-16LE or in any other single- or multi-byte encoding. This makes it hard to correctly interpret filenames in the `bodyfile`.

References

[1] Brian Carrier. The Sleuth Kit website. `https://www.sleuthkit.org/sleuthkit/`, 2003-2020. [Online; accessed 02-April-2022].

[2] G. Clyne and C. Newman. Date and Time on the Internet: Timestamps. RFC 3339, RFC Editor, July 2002.

[3] Joachim Metz. Bodyfile. `https://forensicswiki.xyz/wiki/index.php?title=Bodyfile`, 2021. [Online; accessed 02-April-2022].