Incident Response - Filesystem Timeline Generation

Taz Wake | Halkyn Consulting Ltd | Originally published 2024-09-09

There is no doubt that a well-generated and well-analysed timeline of events is key to understanding any intrusion. Building a timeline is a key skill for anyone working in Digital Forensics or Incident Response (DFIR).

The most basic method (*and the one we all used to rely on until fairly recently*) was for the investigator to copy/paste timestamps into a spreadsheet and then add some text explaining what happened. This definitely works well, but is the opposite of fast or efficient.

Speed and efficiency are so crucial in DFIR that you need to use tooling to generate timelines today. The difference is pretty stark. Instead of spending 2-3 hours to create a manual timeline, you can generate a filesystem timeline in a few minutes at most.

In this post, I will look at some of the common command line tools used to generate timelines. There are, as with most things, lots of other choices, but these tools are an excellent starting point. The main thing to remember is that your choice of tool is likely to be constrained by the filesystem you are analysing!

Note: *Throughout this, I am going to assume you are analysing the output as a CSV and opening it in a tool such as Timeline Explorer. I will also assume you can install any tools you need.*

Windows - NTFS Timelines

Using MFTECmd.exe and TSK

This is probably the best, and fastest, approach for NTFS timelines.

Requirements

- A copy of the MFT. This can be a full disk image or just a copy of the $MFT file itself (such as from a KAPE extract)
- Access to the mactime script from The Sleuth Kit (TSK)

Workflow

First, run mftecmd.exe against the $MFT to generate a "bodyfile". An example of the command line for this is: (assuming the evidence is the MFT for the C:\ drive of a target system)

```
```

In use, it will look something like this:

```
G:\> MFTECmd.exe -f "E:\C\$MFT" --body "G:\Case1234\C_drive_timeline" --bodyf suspect_system.body --blf --bdl C:
MFTECmd version 1.2.2.1

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/MFTECmd

Command line: -f E:\C\$MFT --body G:\Case1234\C_drive_timeline --bodyf suspect_system.body --blf --bdl C:

File type: Mft

Processed E:\C\$MFT in 3.8171 seconds

E:\C\$MFT: FILE records found: 357,501 (Free records: 30,511) File size: 379MB
Path to G:\Case1234\C_drive_timeline doesn't exist. Creating...
       Bodyfile output will be saved to G:\Case1234\C_drive_timeline\suspect_system.body
```

Example of MFTECmd.exe running

This will produce a "body" file timeline. Bodyfiles are a machine-readable timeline. We can use this as our "final" timeline but it isn't always the easiest thing to read - for example, the entries look like this:

```
 0|c:/$MFT|0-128-6|r/rrwxrwxrwx|0|0|397410304|1661977933|1661977933|1661977933|1661977933
0|c:/$MFT ($FILE_NAME)|0-48-3|r/rrwxrwxrwx|0|0|397410304|1661977933|1661977933|1661977933|1661977933
0|c:/$MFTMirr|1-128-1|r/rrwxrwxrwx|0|0|4096|1661977933|1661977933|1661977933|1661977933
0|c:/$MFTMirr ($FILE_NAME)|1-48-2|r/rrwxrwxrwx|0|0|4096|1661977933|1661977933|1661977933|1661977933
0|c:/$LogFile|2-128-1|r/rrwxrwxrwx|0|0|67108864|1661977933|1661977933|1661977933|1661977933
0|c:/$LogFile ($FILE_NAME)|2-48-2|r/rrwxrwxrwx|0|0|67108864|1661977933|1661977933|1661977933|1661977933
0|c:/$Volume|3-128-3|r/rrwxrwxrwx|0|0|0|1661977933|1661977933|1661977933|1661977933
0|c:/$Volume ($FILE_NAME)|3-48-1|r/rrwxrwxrwx|0|0|0|1661977933|1661977933|1661977933|1661977933
0|c:/$AttrDef|4-128-1|r/rrwxrwxrwx|0|0|2560|1661977933|1661977933|1661977933|1661977933
0|c:/$AttrDef ($FILE_NAME)|4-48-2|r/rrwxrwxrwx|0|0|2560|1661977933|1661977933|1661977933|1661977933
```

Example bodyfile data

To make life easier, we can convert this to a human-readable CSV document with the mactime command from The Sleuth Kit. The syntax is

The arguments used here are:

- -z The timezone from where the data was collected.
- -y Display datetimes in ISO8601 format.
- -d Display output in CSV format.
- -b location of the body file.
- STARTDATE and ENDDATE should be specified as YYYY-MM-DD..YYY-MM-DD.

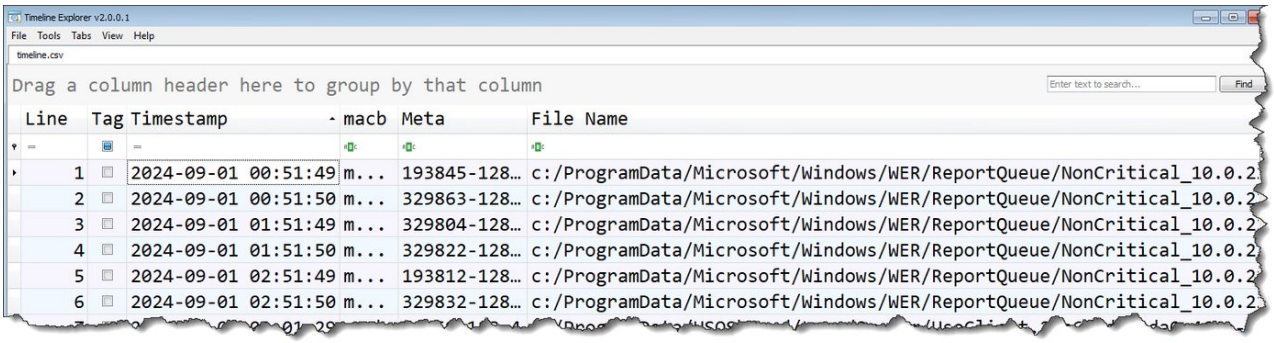
In use, the output will look something like this.

```
sansforensics@SANS-SIFT:/mnt/g/Case1234/C_drive_timeline$ mactime -z UTC -y -d -b suspect_system.body 2024-09-01..2024-09-07 > timeline.csv


sansforensics@SANS-SIFT:/mnt/g/Case1234/C_drive_timeline$ head -n3 timeline.csv
Date,Size,Type,Mode,UID,GID,Meta,File Name
2024-09-01T00:51:49Z,24576,m...,r/rrwxrwxrwx,0,0,193845-128-1,"c:/ProgramData/Microsoft/Windows/WER/ReportQueue/NonCritical_10.0.22000.1165_9874eb93b2d8f87ea
5bede9daf445af5c5a59328_00000000_cab_22966a2e-7e5f-4bcb-acda-9eeeabbce00e/MoUsoCoreWorker.5e3003f9-4ec7-4f85-934d-b70d817ee2bb.1.etl"
2024-09-01T00:51:50Z,8192,m...,r/rrwxrwxrwx,0,0,329863-128-1,"c:/ProgramData/Microsoft/Windows/WER/ReportQueue/NonCritical_10.0.22000.1165_9874eb93b2d8f87ea5
bede9daf445af5c5a59328_00000000_cab_22966a2e-7e5f-4bcb-acda-9eeeabbce00e/MoUxCoreWorker.916b580a-9d7e-4dfa-8009-bf7376ed717c.1.etl"
```

Running mactime and then checking the first three lines of the resulting CSV file.

Now you have a CSV version of the timeline, you can easily open it in Timeline Explorer and analyse away!



Opening a CSV in Timeline Explorer.

An alternative approach - The Sleuthkit Only

This actually takes about the same time to generate, however, it does require you to have a full disk image (E01, RAW, AFF). Collecting this can take a lot longer than a Triage collection of the $MFT.

Requirements

- A full disk image.
- Access to The Sleuth Kit
- This is probably best run from a Linux-based investigation host.

Workflow

The main change is that instead of using mftecmd.exe, you use fls to create the timeline. The command would be something similar to: (this example assumes you are running it from a Linux investigation system)

```
```

(note: there is a space between the drive letter - C: here - and the start of the path)

Once you have built the bodyfile, you convert it into a human-readable (normally CSV) format with mactime as shown above.

Linux Filesystems.

Generating timelines from Linux evidence is a little bit more complicated, and this is really because there isn't really an equivalent for the $MFT. Most of the time, this means we need to run it against a live system or a full disk image, rather than having the relative ease of using a triage collection or just capturing the $MFT via EDR.

Another important consideration is that there are always limitations in the tools we use, and different filesystems lead to different issues. Some key considerations include:

- EXT3 does not have a file creation timestamp.
- EXT4 does have a file creation timestamp but this is not always identified by our tools. Often this is down to the tool making a call to something like **os.stat()** in Python which doesn't recover the creation timestamp. Until the tools get updated, we won't really see the full picture.

- Lots of tools simply don't work against XFS (especially The Sleuth Kit) and even more don't work against Btrfs/ZFS filesystems.
- Sometimes you might just have to buy a commercial tool. If you have this option then at the very least you will have an account manager you can complain to when it lies to you.

With that out of the way, the approaches we will consider are: Using The Sleuth Kit and using a triage collection tool called UAC. The second option (UAC), gives us an effective way to build timelines on XFS, Btrfs and other Linux files systems even if they aren't supported by TSK.

Using The Sleuth Kit against EXT4

If you have a full disk image, this is very, very quick to run. It works best on EXT4 but in principle, it can be used on EXT3 (you just won't get creation timestamps).

Requirements

- A full disk image in E01, AFF or raw format.
- A copy of The Sleuth Kit including mactime.

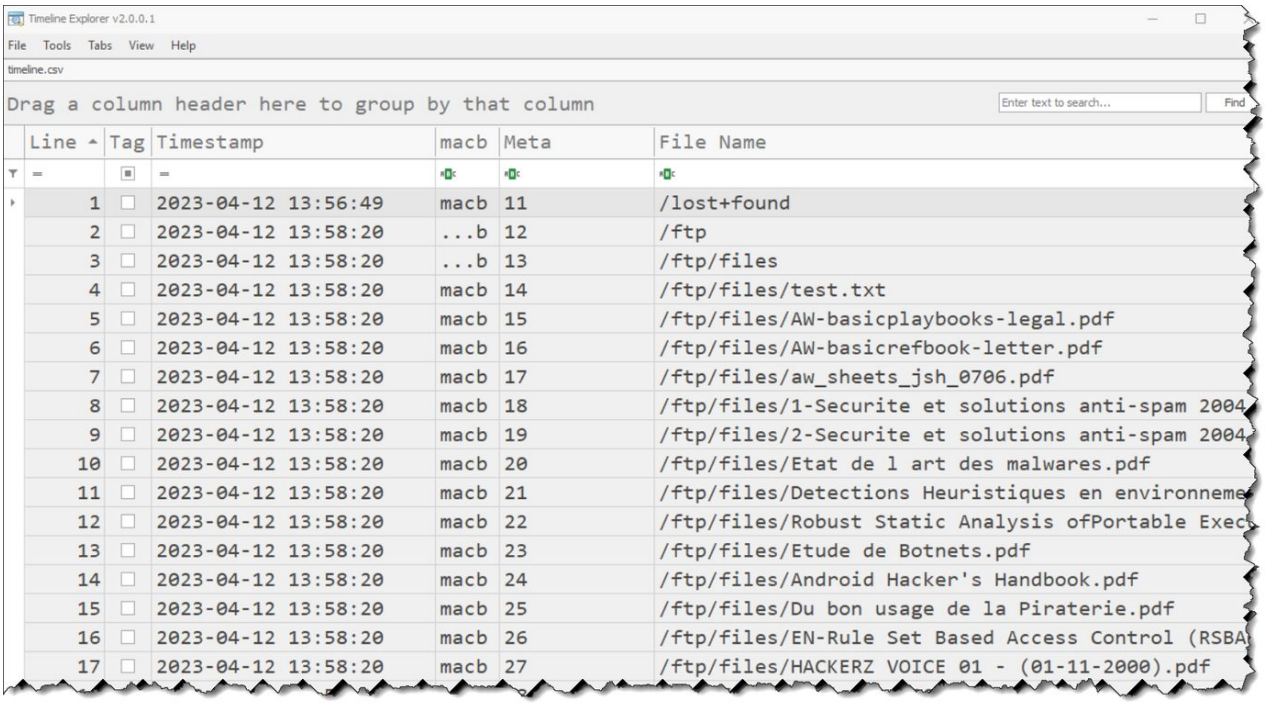
Workflow

This is very similar to how you would run it against MS Windows evidence. First run fls to create the bodyfile, then convert it with mactime.

An example of this is shown below.

Creating an EXT4 timeline.

As with Windows, it is often better to use Timeline Explorer from the Zimmerman tools to read this.



Line	Tag	Timestamp	macb	Meta	File Name
1		2023-04-12 13:56:49	macb	11	/lost+found
2		2023-04-12 13:58:20	...b	12	/ftp
3		2023-04-12 13:58:20	...b	13	/ftp/files
4		2023-04-12 13:58:20	macb	14	/ftp/files/test.txt
5		2023-04-12 13:58:20	macb	15	/ftp/files/AW-basicplaybooks-legal.pdf
6		2023-04-12 13:58:20	macb	16	/ftp/files/AW-basicrefbook-letter.pdf
7		2023-04-12 13:58:20	macb	17	/ftp/files/aw_sheets_jsh_0706.pdf
8		2023-04-12 13:58:20	macb	18	/ftp/files/1-Securite et solutions anti-spam 2004
9		2023-04-12 13:58:20	macb	19	/ftp/files/2-Securite et solutions anti-spam 2004
10		2023-04-12 13:58:20	macb	20	/ftp/files/Etat de l art des malwares.pdf
11		2023-04-12 13:58:20	macb	21	/ftp/files/Detections Heuristiques en environneme
12		2023-04-12 13:58:20	macb	22	/ftp/files/Robust Static Analysis ofPortable Exec
13		2023-04-12 13:58:20	macb	23	/ftp/files/Etude de Botnets.pdf
14		2023-04-12 13:58:20	macb	24	/ftp/files/Android Hacker's Handbook.pdf
15		2023-04-12 13:58:20	macb	25	/ftp/files/Du bon usage de la Piraterie.pdf
16		2023-04-12 13:58:20	macb	26	/ftp/files/EN-Rule Set Based Access Control (RSBA
17		2023-04-12 13:58:20	macb	27	/ftp/files/HACKERZ VOICE 01 - (01-11-2000).pdf

Using Timeline Explorer to read the EXT4 timeline

Using UAC against live systems or captured data (works on XFS/Btrfs)

UAC (Unix-like Artifacts Collector) is an incredible tool for rapidly collecting data from live systems or mounted images. **The biggest advantage here is that it will analyse XFS and Btrfs data if you can mount the system**.

At a very high level, UAC uses a collection profile (a predefined list of artifacts) to collect data from a target.

The profiles currently available are:

- full
- ir_triage
- offline

Full and ir_triage are similar, although the ir_triage profile collects less data and can be faster to run. Both also include recovering data from the /proc filesystem (memory), which is pointless on a mounted evidence item. However, as you can imagine, the offline profile skips this and is much more effective running against a mounted image.

In use, the syntax is pretty consistent.

With an offline profile, also use --mount-point to specify where the evidence is mounted.

This is an example of the output when running UAC an offline collection profile against an XFS formatted evidence item mounted at /mnt/xfs and storing the data in /tmp.

```
root@siftworkstation:/opt/uac-2.9.0# ./uac -p offline --mount-point /mnt/xfs /tmp
--------------------------------------------------------------------------------
 Unix-like Artifacts Collector 2.9.0
--------------------------------------------------------------------------------
Operating System    : linux
System Architecture : x86_64
Hostname            : unknown
Mount Point         : /mnt/xfs
Running as          : root
Temp Directory      : /tmp/uac-data.tmp
--------------------------------------------------------------------------------
Artifacts collection started...
[001/103] 2024-09-07 17:53:00 +0000 bodyfile/bodyfile.yaml
[002/103] 2024-09-07 17:53:04 +0000 chkrootkit/chkrootkit.yaml
[003/103] 2024-09-07 17:53:04 +0000 hash_executables/hash_executables.yaml
[004/103] 2024-09-07 17:53:05 +0000 files/applications/itunes_backup.yaml
[005/103] 2024-09-07 17:53:05 +0000 files/applications/teamviewer.yaml
[006/103] 2024-09-07 17:53:05 +0000 files/applications/rustdesk.yaml
[007/103] 2024-09-07 17:53:05 +0000 files/applications/addressbook.yaml
[008/103] 2024-09-07 17:53:06 +0000 files/applications/wget.yaml
[009/103] 2024-09-07 17:53:06 +0000 files/applications/microsoft_office_mru.yaml
[010/103] 2024-09-07 17:53:06 +0000 files/applications/box.yaml
```

UAC running against /mnt/xfs

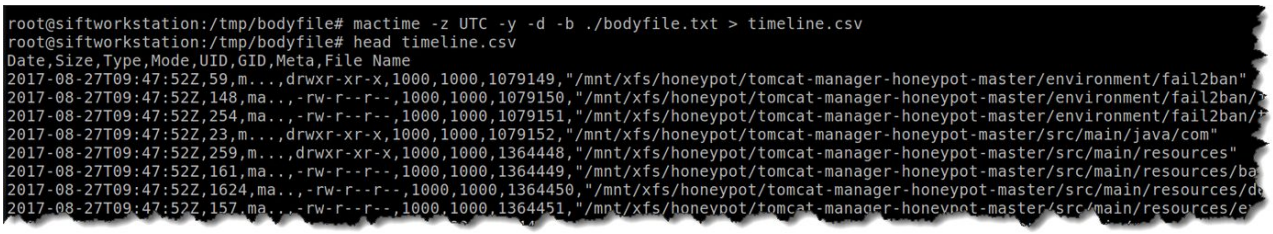
When it finishes, there will be a .tar.gz file in the output folder (/tmp in our example above) which contains a bodyfile folder. In here there is *bodyfile.txt* - which should look like this.

```
root@siftworkstation:/tmp/bodyfile# cat bodyfile.txt
0|/mnt/xfs/.nothingtoseehere|1364621|drwxr-xr-x|0|0|6|1681340335|1681340335|1681340335|1681340335
0|/mnt/xfs/advanced/physics/001199.text|1572995|-rw-rw-r--|1000|1000|6072|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001200.text|1572996|-rw-rw-r--|1000|1000|11212|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001201.text|1572997|-rw-rw-r--|1000|1000|57307|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001202.text|1572998|-rw-rw-r--|1000|1000|11889|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001204.text|1572999|-rw-rw-r--|1000|1000|27302|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001254.text|1573000|-rw-rw-r--|1000|1000|147433|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001351.text|1573001|-rw-rw-r--|1000|1000|44850|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001352.text|1573002|-rw-rw-r--|1000|1000|35278|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001353.text|1573003|-rw-rw-r--|1000|1000|4047|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001415.text|1573004|-rw-rw-r--|1000|1000|120264|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001416.text|1573005|-rw-rw-r--|1000|1000|416785|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001417.text|1573006|-rw-rw-r--|1000|1000|124043|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001434.text|1573007|-rw-rw-r--|1000|1000|501907|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001435.text|1573008|-rw-rw-r--|1000|1000|501907|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001437.text|1573009|-rw-rw-r--|1000|1000|127903|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001438.text|1573010|-rw-rw-r--|1000|1000|41895|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001439.text|1573011|-rw-rw-r--|1000|1000|45852|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001440.text|1573012|-rw-rw-r--|1000|1000|11349355|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001441.text|1573013|-rw-rw-r--|1000|1000|1731700|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001442.text|1573014|-rw-rw-r--|1000|1000|340034|1681339712|1681339712|1681340680|1681339712
0|/mnt/xfs/advanced/physics/001443.text|1573015|-rw-rw-r--|1000|1000|137436|1681339712|1681339712|1681340680|1681339712
```

Bodyfile.txt from UAC

This is a timeline of the system data in bodyfile format. Now you can use mactime to convert this as previously identified.

When it runs, you should end up with CSV formatted output.



Converting bodyfile.txt to a CSV timeline.

Using UAC against a live system is similar, but the arguments change - for example:

This command will run the bodyfile artifact against the live system it is run on and store the data in /tmp.

Tools

- The Sleuth Kit (fls, mactime, etc): https://www.sleuthkit.org/sleuthkit/download.php
- UAC: https://github.com/tclahr
- Zimmerman Tools (MFTECmd, Timeline Explorer, etc): https://ericzimmerman.github.io/#!index.md

Summary

Timelines are effectively a "forensic superpower," and being able to quickly build one is an essential skill for any incident responder. While you may have a go-to commercial tool, it definitely helps if you have the ability to rapidly build one using alternative methods.

When it comes to DFIR, the more tools and techniques you have available, the better you will be able to adapt to a given incident and resolve any issues quickly. If you are working in a Linux or multi-OS environment, one of the biggest challenges is finding support for filesystems other than EXT!

In Windows, we have a lot of options. One of the most efficient is to pull the $MFT with a triage tool and then use MFTECmd.exe and mactime. If you have a full disk image, then you can also just use The Sleuth Kit.

Linux is more complex. If it is an EXT family full disk image, then The Sleuth Kit is still an option. However, a faster way (and possibly the only way if you have XFS/Btrfs etc) might be to use UAC to build the bodyfile, then mactime to convert it.

#DFIR #timeline #infosec #cybersecurity #forensics #incidentresponse #cyber #security

About Halkyn Consulting

Halkyn Consulting Ltd is a UK-based security and risk management consultancy providing 24/7 incident response, digital forensics, threat hunting, virtual security leadership (vCISO) and security compliance support to organisations worldwide. We deliver clear, practical, vendor-independent advice based on real-world threats — no hype, no upsell.

More free security resources are available at www.halkynconsulting.co.uk/security-resources.

Web: www.halkynconsulting.co.uk | **Email:** info@halkynconsulting.co.uk | **Tel:** +44 1244 940858