5

Computer Investigation Process

Being a digital forensic examiner requires you to have a plan to conduct the investigation. For instance, there is the kitchen sink approach – where the person requesting the examination states, *I want it all*. However, this is not practical when the smallest drive might contain hundreds of thousands of pages or events. So while the kitchen sink approach is a plan, it may not be the most efficient.

In reality, your search method will depend on the crime you are investigating and whether there are limitations to the scope of the search. For example, in some investigations, the judicial authority may restrict an investigator's access to digital evidence to only email messages, or you may be limited to a specific date and time within the forensic image.

This chapter will first go through timeline analysis, where a user's activity is analyzed *temporally*. Then, we will examine the storage containers used by the user. You will also learn about string search, in which you search a dataset using matching strings of characters. Finally, in the last section, we will analyze data that has been deleted from the filesystem.

In this chapter, we will learn about the following topics:

- Timeline analysis

- Media analysis
- String search
- Recovering deleted data

Timeline analysis

During the investigation, you may find artifacts that appear to show the accused's guilt or innocence. However, we cannot construe the mere presence of the artifact as a sign of the suspect's guilt or innocence. Instead, the artifact needs to be placed within the user and system activity context.

For example, I was brought in as a consultant on a case; they accused the suspect of physically abusing their child. One piece of evidence that was considered against the suspect was the high number of Google searches about how to treat an injury. They attributed the searches to the accused, who was the father. The most challenging piece of evidence is to prove the user's identity behind the keyboard when the contested actions occurred. Since the items were present in the internet history (we will go into much greater detail in *Chapter 9*, *Internet Artifacts*), I wanted to check the context of when the searches were made. The wife was the primary owner of the laptop, but the husband was also a frequent user of the laptop. So, how do you attribute the searches to a specific user, especially when you have multiple people using the same laptop with the same user account?

A person's internet viewing habits can almost be as distinctive as a fingerprint. As I reviewed the one million-plus lines of internet history, I could differentiate the two different users on the laptop. I could correlate social media use with each user and attribute the Google searches to the child's mother. When she was confronted with the findings, the mother admitted that she searched for how to treat her child's injuries. After being presented with the evidence and testimony of the mother, the jury found the client not guilty of child abuse.

Suppose they had done a timeline analysis before making the charging decision. In that case, I believe the father would not have been charged, as the only evidence against him was the digital evidence found on the wife's laptop.

Your ability to create a timeline to analyze the system and actions of the user allows you to develop a much deeper and more thorough understanding of digital evidence. When I first started in the field, timelines were rudimentary and were typically based on the MAC times of the filesystem. **MAC** times refer to the **Modified, Accessed, and Created** times that are records created by the filesystem as created, edited, or accessed. The downside to only using MAC times for timeline analysis is that the recorded times may not be accurate. For example, this can happen when files are moved from one volume to another or if a user uses a third-party tool to change the timestamps and the timestamps are dependent on the system time.

We will now use multiple sources to help us to determine the context of what is happening on a system regarding a specific artifact. These additional sources may not be as easily manipulated as the MAC times and can determine any irregularities in the timestamps. For example, using multiple resources found within the forensic image, we can see when the user logs in, launches an executable, and accesses a file associated with the executable. This method of accessing multiple sources helps us confirm and validate the information provided by the MAC times.

Applying multiple frames of reference to the event being investigated allows us to support our hypothesis about the event. For example, can we determine whether the investigated incident results from user activity or is it a system process? In addition, using all of the available sources such as event logs, filesystem logs, or internet history captured by the system allows us to get into the small details to see the context of the event.

By gathering data points from multiple sources, you can create what Rob Lee from the SANS Institute calls a super timeline because of the sheer amount of data points you will have to sort through.

Hard drive capacity is not getting smaller. Instead, it is increasing at a phenomenal rate. Users and developers use this increased capacity to store more data and increase the number of logs that can track what occurs in a system. In some investigations, you may not need to examine the content of the files; for example, in an investigation dealing with illicit images, I need not see the visual depiction of the file. Instead, to answer whether a user knew about the existence of a specific file, I can use timeline analysis to make that determination.

Commercial forensic (and open-source) tools have made many advances when it comes to creating timelines. For example, at one time, you had to use many tools to extract data to create a timeline. Now you can use just a single tool to create a timeline.

Note

In this chapter, we will be discussing date-times, which will be converted into UTC/GMT. Always be aware of which time zone your dataset is operating in and the time zone it is stored. I use GMT/UTC as a standard when conducting an examination.

In this chapter, I will demonstrate the use of several tools for you to see the difference in the outputs and discuss where the tools pull the information from.

X-Ways

X-Ways Forensics has a very robust timeline-creation utility built in, called an **event list**. X-Ways compiles multiple sources such as timestamps at the filesystem level, internal timestamps, browser histories, event logs, registry hives, emails, and many other sources. When you start an event list, the data will be presented chronologically, creating a timeline. The event list is a very detailed timeline with copious amounts of information, which allows you to see the sequence of events of the incident you are investigating.

Note

As you explore the features of a new tool, remember to validate the tool against a known dataset. We will use a forensic image offered by Digital Corpora for this lab. You can visit `https://digitalcorpora.org/` and go to the 2008 M–57 Jean scenario for more information.

In this scenario, you are investigating a data leak. Someone has posted a spreadsheet containing an organization's confidential information onto a competitor's website, and the spreadsheet came from the computer of the CFO, Jean. During her interview, Jean stated that she emailed the spreadsheet to the president, Allison, at her request. The spreadsheet is `m57plan.xls` and can be found on the desktop of Jean's account. It has an MD5 hash value of `e23a4eb7f2562f53e88c9dca8b26a153` and a modified time of **2008-JUL-20 01:28:03 GMT** that also corresponds to Jean's statement regarding when she emailed the spreadsheet.

The filename and time frame give us a starting point for conducting the timeline analysis. When you are in the user environment of X-Ways Forensics, select the icon for the event list:

Figure 5.1: X-Ways

As you can see in the preceding screenshot, when you select the **Calendar** option, it will show you the calendar interface so that you can drill down to a specific day. If I do not filter any of the results on the event list, I have over one million entries that I will need to parse through. My preferred workflow method is to start big and then filter the results to meet the needs of my investigation.

When I filter down to July 20, I have reduced my results to a much more manageable 4,052 events.

Once we filter the results, let's search for the filename and see what activity has occurred. One of the first results shows that at 01:27:42, the system created a link file for the spreadsheet. In the following screenshot, you can see the user activity from 01:27 to 01:28. A pre-fetch file (`EXCEL.EXE-1C75F8D6.pf`) was created for Excel at 01:27, which shows the user starting the Excel program and then opening the spreadsheet, which corresponds to the creation of a link file:

Figure 5.2: Filter results

When you view the event list, you can see where the forensic tool is getting the information that is being displayed. The creation of the pre-fetch file starts with a change in the NT `user.dat` file. The tool follows along from gathering information from the internal file metadata to the operating system artifact. We can follow along and observe what occurs at the user and system levels as the user activity is being recorded.

If you look at timestamp 01:28:00, you can see that Jean sent a message out. In the **Name** column, we can see the subject of the email, and when we double-click on it, we can view the email itself:

Figure 5.3: Jean's email

We can see that Jean has emailed what appears to be `allison@M57.biz`, but, in reality, it is going to `tuckgorge@gmail.com`. We can then filter by file type, in this case, the `.eml` files, and you can see the results as follows:

Graphical user interface, text, application, email Description automatically generated

Figure 5.4: Jean's email header

When you look at the **Sender** and **Recipients** columns, and when the data is sorted chronologically, you can get a good idea about the email communication between the attacker and Jean. It appears they have compromised Allison's account, as we can see the name "Alex" and the email account `tuckgorge@gmail.com` associated with the account.

Using the event list feature of X-Ways Forensics allows us to pinpoint when the file was compromised and from what vector. Now we can direct our investigation to Allison's computer to determine whether the attacker compromised her system. Based on these initial results, I believe the attacker targeted Jean in a phishing attack.

What I like about X-Ways Forensics is its ability to gather the dates and times from traditional sources and combine them with the actual artifacts, in this case, the emails. This gives you another level of granularity and context for your investigation.

The X-Ways Forensics documentation lists the following as sources of information for the event list feature:

Table Description automatically generated

As you can see, this shows a very diverse list of sources. However, when used for analysis, it can give the investigator the confidence to rely on the date timestamps they are reporting in their investigation.

I have found that forensic suites also include timeline analysis with their products. I have discussed X-Ways Forensics and its ability to create a timeline for analysis with its event list feature. I have included a list of some additional forensic suites that you may use to analyze timeline data. The following list is not inclusive of all the forensic suites that are available:

- Belkasoft Evidence Center: `belkasoft.com/ec`
- Autopsy: `www.sleuthkit.org/autopsy`
- Recon Lab: `sumuri.com/software/recon-lab`
- PALADIN: `sumuri.com/software/paladin`

X-Ways is not the only tool you can use to create timelines; there are also several open-source tools that you can utilize. One of the most common is **Plaso/log2timeline**, which we will discuss next.

Plaso (Plaso Langar Að Safna Öllu)

Plaso (Plaso Langar Að Safna Öllu) is a Python backend and framework for the `log2timeline` tool. `log2timeline` is a forensic tool that pulls out timestamps from a system and creates a database of all the events, also known as a super timeline.

Note

You can download Plaso at https://github.com/log2timeline/plaso.

Plaso will work on most operating systems and was initially designed to replace the Perl version of `log2timeline`. However, the development has now shifted to modules, and they have created several CLI tools supported by the Plaso backend.

The tools supported by Plaso are activated by the **command-line interface** (**CLI**). While the CLI can intimidate the user, if you take your time and proceed slowly, you will take the mystique out of the CLI. Many open-source tools use the CLI instead of the **graphical user interface** (**GUI**). The very core of the CLI consists of two parts: the executable and the modifiers. Once you learn the specific modifiers for the CLI command, you will see that it all falls into place.

Let’s talk about the tools included with Plaso:

- `image_export`
- `log2timeline`
- `pinfo`
- `psort`

- `psteal`

image_export

`image_export` will export file content from a device, media image, or forensic image. There are several parameters that you can use to define the information you wish to extract.

In the Windows version of the executable, the executable will end with `.exe`. With macOS, you may see it end in `.sh`.

Using `–h` or `--help` will give you the full list of parameters:

Figure 5.5: image_export

Further down the screen, you will see detailed explanations for the modifiers. Note that I will only cover the most used options; there is additional documentation that we will not discuss here:

- `--names NAMES`: The filter on filenames. This option accepts a comma-separated string denoting all filenames, for example, `x NTUSER.DAT,UsrClass.dat`.
- `-w PATH`, `--write PATH`: The directory in which extracted files should be stored.
- `--data PATH`: The path to a directory containing the data files.
- `-x EXTENSIONS`, `--extensions EXTENSIONS`: The filter on filename extensions. This option accepts multiple comma-separated values, for example, `csv`, `docx`, and `pst`.

If you use the following command, it will export the `.xls` file to the `files` folder:

```
image_export --names 'm57plan.xls' C:\tools\plaso\image\jean.001
```

You can see the breakdown of the preceding command as follows:

Figure 5.6: CLI map

Here, with the `image_export` command, we are using the `names` modifier to look for a specific file. In this case, it is `M57plan.xls`.

Now, you can tell the executable where to search; in this command, we are searching in the forensic image, `jean.001` (make sure that you include the full path to where the forensic image is located). Next, you can indicate where you want the exported files to be sent. The `-w` modifier will specify the write location.

You will find that the modifiers have some commonality with the commands within the Plaso framework.

log2timeline

`log2timeline` is a CLI tool that is designed to extract chronological-based events from files, directories, forensic images, and devices. It will create a database file (`.plaso`) that can then be analyzed by a variety of tools.

As you can see in the following screenshot, the `-h` modifier (help) will display the options for the command. As before, there are detailed explanations not displayed that will give you additional context for these commands. You should be able to recognize some of them from the previous command we looked at:

Figure 5.7: log2timeline

Try using the `info` modifier, as follows:

```
c:\tools\plaso>log2timeline.exe --info
```

You will get a list of all of the supported plugins, parsers, and output modules:

Figure 5.8: Results of the info modifier

From the preceding output, you can see that some of the presets include collecting artifacts from many filesystems.

At a very basic level, you can use the following command structure:

```
log2timeline OUTPUT INPUT
```

One idiosyncrasy of `log2timeline` is that the output file is the first modifier to the executable and then you specify the input:

```
log2timeline C:\tools\plaso\export\files\jean.plaso C:\tools\pla
```

When the command executes, you should see the following output on the screen:

Figure 5.9: Output

As the command executes, it locates the data folder that contains the dependencies for the executable, and then it searches for the files that contain the information about the artifacts that may be stored within the system. This is a default folder and is installed when you install `plaso`.

We now have a `.plaso` file that we can find in the `files` folder. In some cases, you might not want to create the database file with every option, that is, the kitchen sink. Rather, you may wish to do a targeted examination of the timeline, in which case you would need to employ filters. Using the `-f` modifier will allow you to do that.

Note

If you want to download some premade filters, you can do so at https://github.com/mark-hallman/plaso_filters.

I downloaded the premade filters and created a folder, named `filter`, within the path of the `plaso` installation. As you see from the following screenshot, I have installed `plaso` in a folder called `tools` at the root of my `C` drive:

```
log2timeline -f filter_windows.txt C:\tools\plaso\export\files\j
```

And, as you can see in the following screenshot, the tool was able to locate my filter within the `artifacts` folder and created a new Plaso database file:

Figure 5.10: Filter

So far, we have covered several commands; however, we still have more to cover. The next command in the framework is `pinfo`.

pinfo

`pinfo` is a command line that is used to display information about the Plaso database file (`.plaso`).

The `plaso` database file will contain the following information:

- When the user executed the tool
- What options were used when the tool was run
- What information was obtained by the tool during the pre-processing stage
- The database metadata
- What was parsed and the parameters that were used
- The number of events extracted
- Tagged events

To learn more about the preceding options, execute the command with the `-h` modifier. While the options are similar, you will have a far smaller selection than with the other tools, as shown in the following screenshot:

Figure 5.11: pinfo

When you use the `pinfo` command in its simplest form, you will get the following results:

```
---------------------------------------------------------------
*********************** Plaso Storage Information **************
Filename: jeanfilter.plaso
```

```
Format version: 20190309
Serialization format: JSON
-------------------------------------------------------------
********************************** Sessions ******************
276a7520-999e-428b-a6b4-11fcf9cf987d : 2019-07-19T22:19:36.0927
-------------------------------------------------------------
```

As you can see in the preceding output, you get the storage information about the file and how many sessions were used to create it.

You can send the results to the standard output, that is, the monitor, or you can use the `-w` modifier to create a text file with the results. The use of the additional tools on the `.plaso` file will create the GUID and the date timestamp of when the analysis was conducted.

The tool can also provide system information about the source system you are now examining:

```
--------------------------------------------------------------
******* System configuration: 276a7520-999e-428b-a6b4-11fcf9cf98
Hostname: N/A
Operating system: Windows NT
Operating system product: Microsoft Windows XP
Operating system version: 5.1
Code page : cp1252
Keyboard layout: N/A
Time zone: GMT
--------------------------------------------------------------
```

After verifying the information in the database file, you can move on to the next command.

psort

`psort` is a CLI tool that allows you to filter, sort, and conduct analysis on the contents of the `plaso` database file. Just like with the previous commands, the `-h` modifier will show you all the options for the command. In the following `psort` screenshot, you can see the available options, and you should be able to recognize the commonality of the options between all of the commands in the `plaso` architecture:

Figure 5.12: psort

Let's discuss some of the new options:

```
-o FORMAT, --output_format FORMAT, --output-format FORMAT
```

Below is a list of the available output formats:

Name	**Description**
`dynamic`	Output events to a delimiter (comma by default) separated value output format, that supports a dynamic selection of fields.
`elastic`	Output events to an ElasticSearch database. Requires elasticsearch-py.
`elastic_ts`	Output events to an ElasticSearch database for use with Timesketch. Requires elasticsearch-py. Solely intended to be used by the Timesketch backend.

Name	**Description**
`json`	Output events to JSON format.
`json_line`	Output events to JSON line format.
`kml`	Output events with geography data into a KML format.
`l2tcsv`	Output events to `log2timeline.pl` legacy CSV format, with 17 fixed fields.
`l2ttln`	Output events to `log2timeline.pl` extended TLN format, with 7 fixed fields.
`null`	Do not output events.
`rawpy`	Output events in “raw” (or native) Python format.
`tln`	Output events to TLN format, with 5 fixed fields.
`xlsx`	Output events to an Excel spreadsheet (XLSX).

As you are processing with `psort`, you can export your findings outside of the `plaso` database. There are a wide variety of options that you can use to export the data for analysis. One of the more common formats for exporting is `l2tcsv`, which is the legacy format for `log2timeline` and is a `.csv` worksheet.

A potential issue you may run into when creating the `.csv` worksheet is that if the file you create is too large, some tools may not analyze it, nor will you be able to open it with your favorite spreadsheet program.

`--analysis list`: `psort` comes with analysis plugins installed by default (you can still create your own custom plugins) to allow you to go through the database file and extract and analyze the contents.

You can use the `--analysis list` modifier to view the complete list of plugins:

Figure 5.13: List of analysis plugins

If we run the command, it will go through the `plaso` database file, tagging the specific events that have been identified in the `tag_windows.txt` file (which is part of the default installation and can be found in the `data` directory):

```
psort -o null --analysis tagging --tagging-file tag_windows.txt
```

On completion of the process, it will show you how many tags were applied to the database:

```
************************** Analysis report: 0 ******************
String: Report generated from tagging
Generated on:2019-07-20T20:04:46.000000Z
Report text: Tagging plugin produced 9754 tags
---------------------------------------------------------------
```

Additionally, you can filter out extraneous data using the `--slice` modifier.

Note

5 minutes is the default value. If you want a longer or shorter time slice, you can add the amount after `DATE TIME` with `--slice_size <VALUE>`.

If you find the `GET` event, you may want to place that event into context by observing what occurred before and afterward:

```
psort -q --slice '2008-07-20 01:26:17' c:/tools/plaso/export/fil
```

The command will create a `csv` file, which contains events 5 minutes before and 5 minutes after the timestamp placed in the CLI.

The final tool in the framework is `psteal`, which we will discuss next.

psteal

`psteal` is the final CLI command in the plaso framework. It combines the `log2timeline` and `psort` commands to extract and process events in a single step. It is very much the kitchen sink approach, otherwise known as “I want it ALLLLLL”, and it has a limited selection of modifiers when compared to the other CLI commands within the framework.

Once again, `-h` will provide you with a list of options for the command, which are displayed in the following screenshot:

Figure 5.14: psteal

At a minimum, specify the source and the output. The process will create the plaso database file and place it in the root of the plaso installation. This location allows you to perform additional tagging, filtering, or analysis after the command completes. The naming convention for the database file created is `<timestamp>-<source>.plaso`.

Here's the command. It creates a `.csv` file that is almost 1 GB in size. However, if I change the output to `.xlsx`, it reduces the size to 35 MB. So, keep in mind that you are processing and analyzing your datasets:

```
psteal --source C:/tools/plaso/image/jean.001 -o l2tcsv -w c:/tc
```

I am using a relatively small forensic image of a 20 GB hard drive. Just imagine if you were using a 500 GB or a 1 TB hard drive and it had been active for an extended period.

Now that we have created our database file and have exported the datasets we find relevant to the investigation, what do we do now? It is time to analyze the datasets to find the evidence that will either prove or disprove the allegation. The tools you use for analysis can simply be the spreadsheet reader of your favorite Office suite or a commercial open-source tool designed for that specific purpose.

It is not possible to cover all the tool options that are available to an examiner in this book. I will highlight several options that are available and summarize the tools for you. Ultimately, the analysis of the data is where the examiner eyeballs the dataset and reviews the findings. Once again, it comes back to the verification/validation of your forensic tools to ensure they are providing accurate results.

Here are a few tools:

- **ELK stack**: This can be found at https://www.elastic.co. It is an acronym for three open-source projects: Elasticsearch, Logstash, and Kibana. Elasticsearch is the search and analytical engine. Logstash is the data processor and ingest engine, while Kibana is the visualizer. You have the option to download the three engines and install them in the operating system of your choice. You have options for macOS, Windows, and Linux. There is also the option to pay for the cloud environment if you do not wish to host the systems within your environment.
- **TimelineMaker Pro**: This can be found at www.timelinemaker.com. It is a commercial product specifically designed for creating timeline charts. With this tool, you can import the CSV files created with the plaso framework.
- **TimeSketch**: This can be found at https://github.com/google/timesketch. It is an open-source forensic timeline-analysis tool. It is Linux-based. I have installed it in a virtual environment so that I can use it as needed. It can also be worked on collaboratively by different members of your team. You can also import from a variety of plaso framework output options.
- **Aeon Timeline**: This can be found at www.aeontimeline.com. It is a commercial product specifically designed for creating visual timelines. It will allow you to view relationships among events. It was initially designed for authors, but it can also be used to analyze super timelines. You can import the CSV files created using the plaso framework.

- **Timeline Explorer**: This can be found at `ericzimmerman.github.io/#!index.md`. Timeline Explorer is an open-source platform created by Eric Zimmerman, who wanted a tool to read MAC time and plaso-generated CSV files without the need to use Microsoft Excel. It is not designed to examine very large CSV files; in fact, Zimmerman recommends explicitly that it is best to open smaller, targeted timelines than one giant one.

Media analysis

You can use timeline analysis on several vectors, such as network analysis, media analysis, software analysis, and hardware analysis. Network analysis is where you analyze log files, trace files, and the communication content between users and their devices. Media analysis is analyzing physical storage devices such as hard drives, SSD drives, thumb drives, or optical storage disks. You will examine the content, allocated space, and slack space. Finally, when performing software analysis, you reverse-engineer malicious code and analyze the protection code for potential exports.

So, let's look at media analysis. The primary source for your digital investigation will be the forensic images of storage devices such as hard drives, SSDs, USB devices, optical disks, and mobile devices such as smartphones. Depending on your organization, you may be the person responsible for creating the forensic image, or the forensic image may be provided to you from another part of the organization. Remember, the forensic image is a bit-for-bit copy of the source device. In most cases, you do not want to use a backup as the source of your digital forensic investigation because a backup will not contain all of the information on the storage device.

The storage device may contain four different data types that you want to examine:

- **Allocated space**: This is the space on the storage device that a file occupies. The filesystem recognizes the storage space as being used.
- **Unallocated space**: This is the space on the storage device that is not occupied by a file. The filesystem recognizes the storage space as being available for use.
- **Slack space**: When the data is stored in a cluster, if the file does not completely fill a cluster, the remaining space not used by the file is referred to as slack space.
- **Bad blocks/sectors/clusters**: This is the space on the disk that has been marked bad by the filesystem because of a defect. It can also be used by a user to hide data from a casual inspection.

Brian Carrier describes the progression of media analysis in his paper "Defining Digital Forensic Examination and Analysis Tools" as follows:

- **Disk**: Physical storage devices such as a hard disk drive, SSD, or flash media.
- **Volume**: A container comprising a single disk or multiple disks. You may find numerous volumes on a single disk or a volume may span across multiple discs. You may see the term "volume" used interchangeably with the term "partition." Brian Carrier defines a partition as being restricted to a single physical disk, whereas a volume is a collection of one or more partitions.
- **Filesystem**: This is used within the boundaries of a volume and tracks file allocation and cluster use.
- **Data unit**: The smallest allocation unit available to the filesystem. In most cases, this will be clusters, or, in a UNIX-based system, it will be

blocks.

- **Metadata**: This is the data about data. This includes the modified, accessed, and created date-time stamps, as well as any other information about the file that the filesystem and some applications track.

The goal of media analysis in your digital forensic investigation is to find relevant artifacts that will either prove or disprove the allegations you are investigating. In addition, as you conduct the digital forensic investigation, you may find artifacts that will direct your focus to other locations.

We will now discuss some different analysis techniques that you might use during your digital forensic investigation.

String search

A search method you might use during your digital forensic investigation is a string or byte search. This search technique is utilized when you have a keyword list of specific terms that you wish to search for. Most commercial and open-source forensic tools allow for string searches and will search the allocated, unallocated, and file slack spaces. You can use specific words, symbols, or strings of letters as the search criteria. Generally, you will want to have some predefined keyword lists before you start your digital forensic investigation.

Your keyword lists will fall into one of the following categories:

- **Generic keyword list**: This is a keyword list that you will use in every case. This list can also be further categorized by the subject of the investigation. For example, you may have one keyword list for digital forensic investigations into fraudulent activity and a different keyword list for digital forensic investigations into illicit images.

- **Case-specific keyword list**: This is a keyword list that you will use for a specific digital forensic investigation. As you prepare to conduct your digital forensic investigation, you will identify keywords based on the participants, locations, and, sometimes, the slang used by the participants. For example, you could have keywords based on usernames, email addresses, physical addresses, phone numbers, credit card numbers, and more.

Note

You should avoid keyword terms that are generic or have additional meanings. For example, if you were investigating a homicide, the word “kill” seems to be a valid term to search for. Unfortunately, “kill” is also a term used in the programming language(s) you will find in a computer system. This will leave you with a large number of false positives. Ideally, the goal is to have the keyword list to help filter out non-pertinent data so that you can focus your efforts efficiently.

You may encounter different encoding schemes as you are conducting your searches on forensic images, such as the following:

- **American Standard Code for Information Interchange** (**ASCII**) is a character-encoding scheme based initially on U.S. English and is limited to 256-character codes.

- **Unicode** was developed to overcome the limitations of ASCII. Each character has a unique 2-byte value resulting in the ability to define over 65,000 characters.

While keyword searching can be very powerful, there is a downside to this, as it is very literal when searching for content based on the keyword. For example, if you search for a word, it will not find an alternative spelling; that is, if you are searching for `ally`, the filter will not find `alley`. Luckily, there is an alternative search methodology known as pattern matching/regular expressions.

A regular expression uses character strings to create a search pattern, and it will find all instances that match the pattern. Here are some common symbols and their meanings when used to create a regular expression:

- **The asterisk symbol (`*`)**: Match the preceding character(s) for `X` amount of times. For example, `ca*t` will cause positive hits for `ct`, `cat`, `caat`, and `caaat`.
- **The pound sign (`#`)**: This will match a number (0-9).
- **The backslash (`\`)**: The following character will be interpreted literally. `\.` will be construed as a period.
- **Caret (`^`)**: Match the start of the text. For example, `^123` will cause the positive hits to start with `123`.
- **The dollar sign (`$`)**: Match the end of the text. For example, `123$` will cause positive hits to end with `123`.
- **Plus symbol (`+`)**: Repeat the preceding character(s) for one or more times. For example, `ca+t` will cause positive hits for `cat`, `caat`, and `caaat`.
- **Curly brackets** `{…}`: Repeat the preceding character(s) for `X` times (depending on the value in the bracket).

- **Brackets** `[...]`: This will match a single character in the brackets. For example, `[b,c,d]` will match on b, c, or d.
- **Brackets w/ ^ [^...]**: This will match any single character not in the brackets. For example, `[^b,c,d]` will match any character other than b, c, or d.
- **Brackets (range) [..-..]**: This will match any character within the given range. `[0-9]` will match any character from 0 to 9.
- **Dot** (`.`): The dot can take the place of any character.
- **Question mark (`?`)**: The preceding character may/may not be present. For example, `.e01?` will return `.e0(x)` values. `x` shows it may find any value after`.e0`.
- **Pipe (`|`)**: This matches any one-character set separated by the pipe (`|`) character. For example, `br(ead|ake|east)` will return matches for bread or brake or breast.

The following are some common examples of pattern matching that you may find helpful.

To search for an IP address, you can use the following regular expression:

```
\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}
```

The `\d` specifies that the following will match on a digit (number). The curly brackets, `{1,3}`, indicate the number can be from one to three digits. `\.` specifies a search for the `.` character. The `\d{1,3}` pattern then repeats an additional three times until it has the value for an IPv4 address.

To search for a US phone number, you can use the following expression:

```
((\(\d{3}\) ?)|(\d{3}-))?\d{3}-\d{4}
```

The `\(` will match the open bracket. `\d{3}` will match on a three-digit number. `\)` will match on the closed bracket. This pattern will give you the area code, `(###)`, in this format. The remaining regular expression will give you the first three digits, `\d{3}`, the dash, `-`, and the final four digits, `\d{4}`, of the US phone number. If the phone number is not formatted as `(###) ###-####` or `###-###-####`, you will not get a hit.

Regular expressions are a powerful tool, but they can also be very complicated to craft. I like to use the regular expression library (which can be found at http://regexlib.com/Default.aspx) to help me with my regular expression skills.

So, what happens when the user deletes a file or folder from the media? Let's discuss what happens when a file or folder is deleted next.

Recovering deleted data

When a file is deleted in the FAT filesystem, the data itself does not get changed. The first character of the directory entry will have it changed to a `xE5` and the file allocation table entries are reset to `x00`. When the filesystem reads the directory entries, and it encounters the `xE5`, it will skip that entry and start reading from the subsequent entries.

To recover deleted files, we need to reverse the process the filesystem used to delete the files. Remember, it has not changed the file contents; they still physically reside in their assigned clusters. We now need to reverse-engineer the deletion and recreate the file entry and the entries in the file allocation table. To do this, we need to find the first cluster of the file, the size of the file, and the size of the clusters in the volume:

Figure 5.15: Deleted entry

In the preceding screenshot, we have a directory entry showing that a file has been deleted. We see the `xE5` at the start of the directory entry. (This will require the use of a hex editor to make the changes.) Then, we have to determine the starting cluster, `x00 x08` (which is shown as `x08 x00`), which is cluster number `8`. To determine the file size, look at the last four bytes (remember that the FAT filesystem stores data in little-endian format, which means that the least-significant byte is on the left, so we would read that value as `x00 x00 x00 x27`, not as it is displayed, `x27 x00 x00 x00`), and when we convert the hexadecimal value to a decimal, we get the value of 39 bytes for the file size.

Now we have to determine how many sectors make up a cluster and what the sector size is. You will need to go to the boot record to get that information. The boot record shows that there are 512 bytes per sector, and there are 8 sectors per cluster, which gives us a cluster size of 4,096 bytes:

Figure 5.16: Boot record

This means that our file will only occupy a single cluster. We then go to the file allocation table and look at the entry for cluster 8 and see that it is zeroed out:

Figure 5.17: Deleted FAT

To recover the deleted file, perform the following steps:

1. You need to change the entry in the file allocation table from `x0000 x0000` to `xFFFF FFF8` or `xFFFF FF0F`. If this were a larger file, you

would need to change the file allocation table entry to point to the next cluster until you reach the cluster that contains the end of the file. Should you find an entry marked as allocated before you reach the end of the file, you may be dealing with a fragmented file. Another possibility is when the clusters were made available for use when the file was deleted, the data from a new file was placed in the available space. This would cause the old data to be overwritten with the data from the new file.

2. The next step is to go back to the directory entry and replace `xE5` with another character. When replacing the `xE5` character of the filename in the directory entry, be careful not to guess what the character is. If you select an incorrect character, you could change the meaning or create a bias with the new filename, and that would be improper.

I recommend that when you recover a deleted file, you replace that first character with an underscore or a dash so there is no misunderstanding about the filename.

When recovering a file with a long filename, it is important to relink the long filename to the short filename. This is because when the additional directories are created to accommodate the long filename, the system creates a checksum based on the data of the short filename. When you changed the `xE5` value on the short filename entry, you also want to use the same replacement character for the subsequent `xE5` entries for the long filename directory entries. The reason for linking the long filename to the short filename is that the short filename directory entry contains information such as the date and times, the starting cluster, and the file size.

As we discussed in *Chapter 4*, *Computer Systems*, when a file/directory is created on an NTFS volume, the system creates an entry in the `$MFT` file.

The MFT record will contain the metadata about the file/directory; if the contents of the file are non-resident, then the `$Bitmap` file will be updated to show the clusters occupied by the file are allocated.

When a file/directory is deleted, then the sequence count in the MFT file record's header is incremented by one digit. The allocation status for the record will change from allocated to unallocated. If the file data is non-resident, the system will update the `$Bitmap` file to show the clusters occupied by the file are now unallocated.

Every MFT file entry will start with the file signature of the file, which you can use as a search term to locate MFT file entries in unallocated space. Until the clusters containing the data on the disk are overwritten, we can recover the data.

If the MFT file record is unused, then you can reverse the steps and recover the file. You can decipher the file record, as we discussed in *Chapter 4, Computer Systems*. If the file is resident within the file record, you will recover the data when you retrieve the MFT file record. If the data is non-resident, then you will have to decipher the MFT file record to determine whether the data runs and identify the occupied clusters.

If the system has overwritten the MFT file record, then you cannot recover the deleted MFT file record data or any resident data. You may recover the non-resident data, but that will depend on the size of the files and the fragmentation. Once the MFT record has been overwritten, you will lose any information regarding the data runs and which clusters contain the data.

Summary

In this chapter, we discussed, in detail, timeline creation and timeline analysis with open-source and commercial forensic tools. We took an in-depth look at utilizing the commercial forensic tool, X-Ways Forensics, and the open-source plaso framework for `log2timeline`. We also touched upon using the kitchen sink approach or a targeted examination of the dataset. Remember, we are not analyzing the contents of files, just the timelines associated with the files and other events within the operating system and filesystems.

In the next chapter, we will discuss the contents of files, specifically, Windows artifacts.

Questions

1. It is important for the examiner to know the time zone in which the evidence was collected.

   a. True
   b. False

2. You can do timeline analysis with X-Way Forensics when you create a(n) _______ list.

   a. Timeline
   b. Date/time
   c. Event
   d. Party

3. Plaso is a framework for how many tools?

   a. One
   b. Three

    c. Five
    d. Seven

4. `pinfo` will give you what information?
    a. Information about the examiner
    b. Information about the database file
    c. Information about the forensic machine
    d. Information about the suspect
5. `log2timeline` is a ____________ -based tool.
    a. CLI
    b. GUI
    c. VFD
    d. XYZ
6. `psort` will give you the ____________.
    a. Ability to sort
    b. Ability to filter
    c. Ability to connect
    d. All of the above
7. You can do a timeline analysis with an Excel spreadsheet.
    a. True
    b. False

Further reading

You can refer to the following links for more information on the topics covered in this chapter:

- *T. P. P. A. (2019, July 8). Plaso Documentation. Retrieved from The Plaso Project*: https://buildmedia.readthedocs.org/media/pdf/plaso/latest/plaso.pdf
- *Carvey, H. (2014). Windows forensic analysis toolkit: Advanced analysis techniques for Windows 8; Waltham, MA: Syngress*. Available at: https://www.abebooks.com/servlet/SearchResults?sts=t&cm_sp=SearchF-_-home-_-Results&an=&tn=Windows+forensic+analysis+toolkit&kn=&isbn=

Exercise

Data set

`Chapter 5 Emails.xlsx`

`Chapter 5 Carving.dd`

Software needed

Timeline Explorer - https://ericzimmerman.github.io/#!index.md

Microsoft .NET 6 or newer is required. You will get errors without at least .NET 6. When in doubt, install it! Make sure you get the **Desktop** runtime if you plan on running any of the GUI programs.

Autopsy - https://www.autopsy.com/

Email exercise

An individual outside of m57.biz purchased a laptop from Craigslist. The laptop the individual purchased contained child pornography and they decided to inform the police about it.

Investigators were able to trace the laptop back to m57.biz. When the police contacted the CEO of m57.biz, the CEO reported that the laptop, as well as other items, had been stolen from the m57 inventory.

The m57 CEO gave consent for the police investigators to search m57.biz and image all of the m57.biz computers, company phones, as well as USB drives.

Analyze the emails found in the `Chapter 5 emails.xlsx` spreadsheet and identify potential suspects and a timeline of their activity.

Data carving exercise

1. Load Autopsy and start a new case.
2. Select **Disk Image** or **VM file** for the data source.
3. Navigate to the folder where you stored the image `Chapter 5 Carving.dd`. Select only the following Ingest Modules:
    - PhotoRec Carver Embedded File Extractor
4. From the drop-down menu, select **All files**, **Directory**, and **Unallocated Space**.

Analyze the results.

Join our community on Discord