Trace Evidence Analysis

Jihun Joun, Sangjin Lee, Jungheum Park

We identified and analyzed traces of deleted files through our proposed experiment. Some files’ format and definition have been described in previous studies and official documents. However, it is impossible to establish ESI spoliation based on previous studies only. In particular, there is no research on whether the traces are still left in internal artifacts even after the actual files are permanently deleted. To establish ESI spoliation, exact details of the traces of deleted files left behind in the identified artifacts must be known. Table 1 summarizes the result of trace evidence analysis for digital forensics. In this table, the metadata column shows the types of metadata that can be found in artifacts. We defined types of metadata by six features: Filename (N), path (P), timestamp (T), size (S), content (C), and the file itself (I).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Type | Artifact | Novelty | Filename | Metadata | | | | | |
| **N** | **P** | **T** | **S** | **C** | **I** |
| OS | Spotlight | 1 | com.apple.spotlight.Shortcuts | O | O | O |  |  |  |
| 1 | Parsecd | O |  |  |  |  |  |
| 1 | JournalAttr.[filename] | O |  |  |  |  |  |
| 3 | Store.db | O | O | O |  |  |  |
| Diagnostic | 1 | [random].tracev3 | O | O | O |  |  |  |
| 1 | [year].[month].[day].asl | O |  | O |  |  |  |
| Recent Files | 3 | com.apple.textedit(preview).sfl2 | O | O |  |  |  |  |
| 3 | ~RecentDocuments.sfl2 | O | O |  |  |  |  |
| Document Revision | 3 | db.sqlite | O |  | O |  |  |  |
| 3 | 1 |  |  |  |  | O |  |
| Printer | 3 | c00001 | O |  | O |  |  |  |
| 3 | d00001-[random] |  |  |  |  | O |  |
| Web | 2 | ~QuarantineEventsV2 | O |  | O |  |  |  |
| QuickLook | 3 | index.sqlite | O | O | O | O |  |  |
| Desktop Services Store | 3 | .DS\_Store | O | O |  | O |  |  |
| FSEvent | 3 | .fseventsd | O | O | O |  |  |  |
| KnowledgeC | 2 | knowledgeC.db | O |  |  |  |  |  |
| Document | MS Office | 1 | com.microsoft.Word.plist | O | O |  |  |  |  |
| 1 | ComRPCDB | O | O | O |  |  |  |
| 1 | MicrosoftRegistractionDB\_.reg | O | O | O | O |  |  |
| 1 | ~WRF{[random]} | O |  | O | O | O | O |
| 1 | ~WRS{[random]} |  |  |  |  | O |  |
| 1 | windows.plist | O |  |  |  |  |  |
| 3 | ~securebookmarks.plist | O | O | O |  |  |  |
| Notes | 3 | [filename] | O | O | O | O | O | O |
| 3 | NoteStore.sqlite | O |  | O |  |  |  |
| 1 | CloudKitOperationalInfo | O |  |  |  |  |  |
| Cloud Storage | Google Drive | 1 | (mirror) metadata\_sqlite\_db | O |  | O | O |  |  |
| 1 | psid.db | O |  |  |  |  |  |
| 1 | structured\_log\_[random] | O |  |  |  |  |  |
| 1 | finder\_ext\_[random].txt | O | O | O |  |  |  |
| 1 | drive\_fs.txt | O | O | O |  |  |  |
| iCloud | 3 | [filename] | O | O | O | O | O | O |
| 3 | client.db, server.db | O |  | O | O |  |  |
| Messenger | Microsoft Teams | 3 | [random]\_0 | O | O | O | O | O | O |
| 3 | [random].log | O |  |  |  |  |  |
| 3 | [random].ldb | O | O | O |  |  |  |
| 3 | [random].ldb | O | O | O |  |  |  |
| Slack | 2 | [random]\_0 | O | O | O | O | O | O |
| 1 | [random] | O |  |  | O |  |  |
| 1 | webapp-console.log | O |  | O |  |  |  |
| iChat | 2 | chat.db | O | O | O |  |  |  |
| 2 | [phone num] on [date].chat | O |  |  |  |  |  |
| Web | Safari | 2 | Downloads.plist | O |  | O |  |  |  |
| 1 | [random], [random]-blob | O |  |  |  |  |  |
| 2 | History.db | O |  |  |  |  |  |
| 2 | Favicons.db | O |  |  |  |  |  |
| 2 | RecentlyClosedTabs.plist | O |  | O |  |  |  |
| Mail | Mail (App) | 1 | 1.qdat | O |  |  |  |  |  |
| 2 | Envelope Index | O |  | O |  |  |  |
| Gmail | 2 | [random].partial.emlx | O |  | O |  |  |  |
| 2 | [filename] | O | O | O | O | O | O |

*Table 1: Result of trace evidence analysis for digital forensics*

# **Operating System Artifact**

Operating system artifact is an artifact that is created and recorded at the system level in macOS. This artifact records all the information related to the file, regardless of the application.

## **Spotlight**

There are numerous valuable data in Spotlight on macOS. Spotlight is a desktop search technology first released with Mac OS X 10.4 (Tiger) and is included in recent macOS versions. Spotlight has already been studied for digital forensic analysis. In particular, the “store.db” file is well known, but other related files have not been analyzed yet. We discovered traces of deleted files in Spotlight shortcuts, parsecd, and journalAttr files.

The shortcuts file “com.apple.spotlight.Shortcuts” contains data in XML file format and stores the list searched by Finder. Even if the searched file is deleted, the searched history is not deleted, so it can be useful for digital forensics. As shown in Fig. 2, this file stores the filename, path, and searched timestamp. The searched keyword is stored in the ‘key’ value, and related information is stored in the ‘dictionary’ value. In the ‘display\_name’ in the ‘dictionary’ value, the filename retrieved through keyword search is stored in a string data type. The searched timestamp is stored in the ‘last\_used’ key, and the path is stored in the URL.

텍스트이(가) 표시된 사진

자동 생성된 설명

*Figure 2: Internal structure of “com.apple.spotlight.Shortcuts” file contains filename, path, and timestamp*

Parsecd is a temp log file used for Spotlight, messages, lookup, and Safari. This file’s information and structure are not accessible to the public and have not been analyzed yet. The filename is saved with names as “session.[random].open” or “session.[random].close” in “~/Library/Caches/com.apple.parsecd/”. As can be seen in Fig. 3, we discovered that the filename with or without its extension is stored in this file through the experiment. It is challenging to analyze this file because the file structure is not studied yet. Still, the filename of a deleted file can be identified via keyword search using a regular expression.

텍스트이(가) 표시된 사진

자동 생성된 설명

*Figure 3: Internal structure of “session.[random].open” file contains the filename*

JournalAttr is a journal file for the Spotlight indexing function, and it is not analyzed yet. The filename is saved as “JournalAttr.[random]” in “~/Library/Metadata/CoreSpotlight/index.spotlightV3” and “/.Spotlight-V100/Store-V2/[random]”. Figure 4 shows that the filename and information related to the document file are stored temporarily when the file is uploaded or downloaded from the iCloud. The file structure is not analyzed yet; therefore, the trace of deleted document files can be identified by keyword search using a regular expression.

텍스트이(가) 표시된 사진

자동 생성된 설명

*Figure 4: Internal structure of “JournalAttr.[random]” file contains the filename*

Databases containing details of Spotlight searches have already been analyzed by many researchers. These databases contain details of file system characteristics, metadata, and indexed textual content that helped us to find files and information [28]. Spotlight has already been studied from the perspective of digital forensic analysis. “Store.db,” which is the filename of this artifact, stores Spotlight data containing a filename, path, size, and timestamps [29]. Atwal studied the persistence of records for deleted files in the storage for Spotlight and checked whether the deleted database pages are recoverable from unallocated space [13]. Khatri analyzed the structure of the Spotlight metadata cache database and developed a script to explore and read the database [29].

## **Diagnostics log**

Information related to document files can be found in the diagnostics log. The existence of the diagnostics log is already known, but it has not been thoroughly studied. We found several log files that include information related to document files. The diagnostics log has two versions: Apple Unified Log (AUL) and Apple System Log (ASL). ASL is the old version of the diagnostics log, whereas AUL is the latest and currently used version. Most of the diagnostics information is written in the AUL file, but a few log data are still written in the ASL file, so both types of diagnostics logs should be checked.

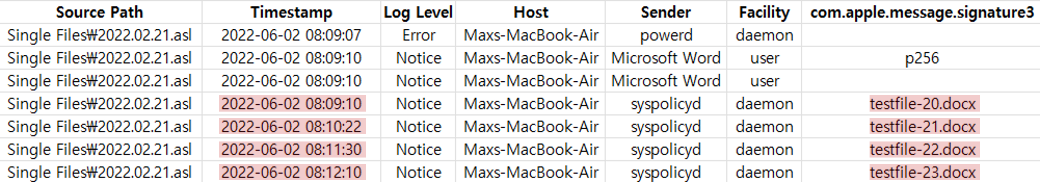
AUL is a new log storing method in macOS applied since macOS 10.12 Sierra. It consists of multiple binary format files with a ‘.tracev3’ extension. AUL files are created in several locations, but diagnostic logs related to document files are stored in the ‘Persist’ and ‘Special’ folders in “/private/var/db/diagnostics” with filenames of random numbers. As shown in Fig. 5, the filename and path are saved in the “[random].trace3” file in the ‘Persist’ and ‘Special’ folder, but the timestamp is saved in the “/private/var/db/diagnostics/timesync”. Although the ‘.tracev3’ file structure was not officially disclosed, Khatri analyzed the file format and developed a parser [30]. Alarms related to information, debugging, warnings, and errors are recorded in the log along with messages; the filename and path are stored in this message. As they are stored in an unstructured way for each log level, traces of document files are found by keyword search using regular expressions during analysis.

텍스트이(가) 표시된 사진

자동 생성된 설명

*Figure 5: Internal structure of “[random].tracev3” file contains the filename, path, and timestamp*

ASL is the old log file format recorded by macOS; filenames are saved as “[year].[month].[day].asl” in the “/private/var/log/DiagnosticMessages”. As shown in Fig. 6, the filename and timestamp are stored. The file structure has not been officially disclosed, but OpenText has developed EnScript to parse this log file format [31]. The filename and description are recorded in the message data when looking inside the file. The filename is recorded in the ‘com.apple.message.signature3’ table, and the timestamp is recorded in the ‘timestamp’ table. Regular expressions may also be used to search for the filename and timestamp instead of using the tool.

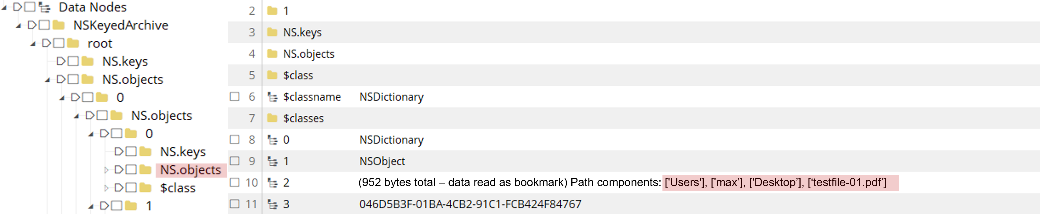


*Figure 6: Internal structure of “[year].[month].[day].asl” file contains the filename and timestamp*

## **Recent files**

Recent opened file lists are stored in several artifacts at the OS level: “com.apple.LSSharedFileList.RecentDocuments.sfl2”, “com.apple.textedit.sfl”, and “com.apple.preview.sfl2”. These files are in the binary plist file format; it is simple to analyze the contents of the file. In this study, we used Plist parser developed by OpenText [32].

The file “com.apple.LSSharedFileList.RecentDocuments.sfl2” stores recently accessed files created by all applications. This file is saved in “~/Library/com.apple.sharedfilelist”. The file “com.apple.TextEdit.sfl” stores recently accessed files written by the TextEdit application. The file “com.apple.preview.sfl2” is the file to provide a recent file’s preview. Information related to files such as pictures or PDF files is saved when the files are opened or modified. The files “com.apple.textedit.sfl” and “com.apple.preview.sfl2” are saved in “~/Library/Application Support/com.apple.sharedfilelist/com.apple.LSSharedFileList. ApplicationRecentDocuments”. As can be seen in Fig. 7, the filename and path are saved. Even though the actual file is deleted, the record of these files is not erased.



*Figure 7: Internal structure of “com.apple.textedit.sfl” file contains the filename and path*

## **Document Revision**

Revision history can be restored if the document file is created by Apple’s TextEdit application. As shown in Fig. 8, file metadata, such as filename and timestamp, is stored in “db.sqlite”. “db.sqlite” is saved in “./DocumentRevisions-V100/db-V1”. In addition, the file list in iCloud is stored in this database too. Deleted file metadata can be found in “db.sqlite-wal” file. In particular, TextEdit records the revision history in a specific filename, “1”. This file contains the content of the previous and current files. Figure 9 shows the actual deleted content in this file. The file is stored in “/.DocumentRevisions-V100/.cs/ChunkStorage/0/0/0”.

텍스트이(가) 표시된 사진

자동 생성된 설명

*Figure 8: Internal structure of “db.sqlite” file contains the filename, path, and timestamp*

테이블이(가) 표시된 사진

자동 생성된 설명

*Figure 9: Internal structure of “1” file contains the deleted file content*

## **Printer**

Garijo studied the digital forensics analysis method for macOS, including printer-related artifacts [33]. When a file is printed, file metadata is stored in the file under a name such as “c00001” or “c00002”. The filename and timestamp of the printed file are recorded and remain for a certain period after printing. The content of the actual printed file is saved under a name such as “d0001”. As this file is in the PDF format, it can be easily viewed using a PDF viewer. The file content is stored inside the PDF file, and the filename can be found by analyzing the structure of the PDF file. These files are stored in “/private/var/spool/cups”. In addition to this, several logs related to printing are recorded in “/private/var/log/cups”, but the actual filename is not recorded.

## **Web artifacts**

The file “com.apple.LaunchServices.QuarantineEventsV2” keeps a log of the file list downloaded. This artifact is already well known; however, we analyzed necessary information to track the ESI spoliation. It is an SQLite file saved in the “~/Library/Preferences” file. As shown in Fig. 10, the downloaded time is recorded in ‘LSQuarantine TimeStamp’. The downloaded URL is saved in ‘LSQuarantineDataURLString’, and the downloaded filename is saved in this URL. Even if there is no actual file in the download folder, the name of the downloaded file and the path of the file are saved in this database.

### **QuickLook Thumbnail Cache**

Quicklook thumbnail cache is a type of cache file that shows the preview of the file before accessing the file. This artifact is already well known and has been analyzed to glean necessary information for tracking ESI spoliation. The file “index.sqlite”, which is the filename of this artifact, is saved in the SQLite format in a folder, “/private/var/folders/[random]/[random]/C/com.apple.QuickLook.thumbnailcache”. In this database, the ‘file’ table contains the filename, path, timestamp (not created or accessed time, generated by opening the parent folder), and size of the actual file [34]. As it is a database in the SQLite format, traces of deleted files can be found by checking the column where the metadata described above is stored without additional analysis. Furthermore, the file’s thumbnail can be seen as this is saved in “thumbnails.data” file.

테이블이(가) 표시된 사진

자동 생성된 설명

*Figure 10: Internal structure of “∼.LaunchServices.QuarantineEventsV2” file contains filename and timestamp*

## **Desktop services store**

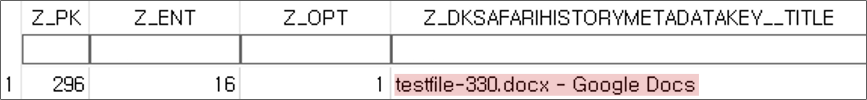
“.DS\_Store”, the desktop services store, helps to read data more quickly when accessing through the Finder application. This file is similar to the ‘desktop.ini’ file in Windows OS. This artifact is already well known and necessary information to track the ESI spoliation has been previously analyzed. Apple developed the “.DS Store” file, and the file structure is open to the public. Sebastian Neef studied the structure of “.DS\_Store” [35]. OpenText developed an EnScript to parse the content in “.DS\_Store” file [36]. The file list in the directory, filename, size, and icon coordination are recorded in this file. Furthermore, the file path can be predicted because this file is saved in each directory. Specifically, this file is created in the recycle bin and external storage device; deleted files and existing files can be identified by checking the file list in this file. Even if the files in the recycle bin and external storage device are deleted, the file list in “.DS\_Store” file continues to exist unless the data are overwritten.

## **File System Events (FSEvents)**

FSEvents is the notification API for the macOS file system [13]. This artifact is a framework that records the file changes, similar to the ‘$LogFile’ and ‘$usnJrnl’ files in Windows OS. The artifact records all the file system activity, including file creation, deletion, rename, and change. FSEvents is used in Spotlight function and in time machine, the macOS backup system. This artifact is saved in the ‘/.fseventsd’ folder as gzip format log file. The filename of this log file is stored with a name as the last Event ID stored in the FSEvents log file plus 1. Ibrahim analyzed FSEvents log file format and developed a parser [37], [38]. A single log file saves the filename, path, and date without a timestamp. To investigate ESI spoliation, the file traces are listed by parsing the filename, path, and date in the studied format.

## **KnowledgeC**

KnowledgeC is a database that stores application usage, application activities, and device power status. “KnowledgeC.db”, which is the filename of this artifact, is saved in the SQLite format in folder, “/private/var/db/CoreDuet/Knowledge”. This file is already analyzed, but it is not known that an accessed document’s filename can exist in this file. Figure 11 shows the result of the experiment. The document’s filename, which is used over the Internet, can be identified in this database file. For instance, the filename is saved in ‘Z\_DKSAFARIHISTORYMETADATAKEY\_\_TITLE’ column in ‘zstructuredmetadata’ table. As this file is a database format, historical data can be found in the “KnowledgeC.db-wal” file.



*Figure 11: Internal structure of “KnowledgeC.db” file contains filename*

## **Finder**

This study mainly focuses on the traces left of a file that has been deleted. However, there is the possibility that a folder has been deleted instead of a file. This artifact stores recently opened folders’ names, connected external storage device names, and external share folder information. This artifact is already well known, and necessary information to track the ESI spoliation is analyzed. “com.apple.finder.plist”, which is the filename of this artifact, is saved as plist format in the folder “~/Library/Preference”. In the ’Name’ tag, the names of the remote shared folder, external storage device directory, and folders recently viewed by the user are recorded. In the “\_CFURLAliasData” tag, a string value pointing to the full path of the directory opened by the user is encoded in Base64 and stored.

# **Document Application Artifacts**

Document application creates specific logs and files to manage the document files. Each application has distinctive logs and files, so the log and file applicable to the application should be checked. This research focused on MS Office and Notes (macOS default note application) among the document applications.

## **Microsoft Office**

In this study, the target version was MS Office 2007+, which is in the Open Office XML (OOXML) format. MS Office 2007+ includes Word, PowerPoint, and Excel. MS Word was analyzed intensively because other applications have similar logs and files.

The file “com.microsoft.Word.plist” stores various information such as application version at boot time, installed language, and last accessed folder name. This file has been previously analyzed, but the fact that traces of deleted files exist in this file had not yet been discovered. As a result of the experiment, the document’s filename and path can be identified in this database file as shown in Fig. 12. “com.microsoft.Word.plist,” which is one of the filenames of MS-Office-related files, is saved as a plist format in the folder “~/Library/Containers/com.microsoft.Word/Data/Library/Preferences”. From the ESI spoliation perspective, the filename and path are saved in this file. Inside the file, the key with a name that starts with NSWindows Frame records the size of an application displayed on the screen. The rest of the key name contains the accessed Word file’s path and name. It is saved under a name such as “NSWindow Frame CUIDocumentShellWindowFrameNameURL[file path]/[filename]”.

텍스트이(가) 표시된 사진

자동 생성된 설명

*Figure 12: Internal structure of “com.microsoft.Word.plist” file contains filename and path*

“ComRPCDB” is a database file that has not been analyzed. This file is saved in “~/Library/Group Containers/UBF8T346G9.Office/ComRPC32”. This file records information about an object when saving an embedded object within a document file. As shown in Fig. 13, the filename and path are stored from the offset ‘0x10’ in the ‘moniker\_eq\_buff’ column of the ‘rot’ table in the database. In the ‘mod\_time’ column of the same table, the creation time of the embedded file is stored as a Windows 64 timestamp. As it is a database format, traces of deleted files can be found in the “ComRPCDB-wal” file.

텍스트이(가) 표시된 사진

자동 생성된 설명

*Figure 13: Internal structure of “COMRPCDB” file contains filename and path*

“MicrosoftRegistrationDB\_[random].reg” is a database file that has not been analyzed. This database contains information about files created or downloaded directly in the PC. “MicrosoftRegistrationDB\_[random].reg” file is a database file stored in the folder, “~/Library/Group Containers/UBF8T346G9.Office/MicrosoftRegistrationDB”. As shown in Fig. 14, the table “HKEY\_CURRENT\_USER\_values” in the database contains information about files created or downloaded directly in the PC. Each file is separated by the value of the ‘node\_id’ column. After sorting by ‘node\_id’ in the table, check the row data with ‘node\_id’. The metadata type is written in the name column of the item with the same ‘node\_id’, and if you check the value column of the same row, the value of the corresponding metadata is written. The data types of the name columns are ‘DocumentURL’, ‘FileName’, ‘FileSizeinBytes’, ‘FutureAccessToken’, ‘isPinned’, ‘Path’, and ‘Timestamp’, and so on. Even if the actual file is deleted, the metadata remains in this database. In addition, although the amount of time of data overwriting is not set, we estimate on the basis of experiments that the data remain for more than one year.

테이블이(가) 표시된 사진

자동 생성된 설명

*Figure 14: Internal structure of “MicrosoftRegistrationDB\_[random].reg” contains filename, path, timestamp, and size*

The “~WRF[random]” and “~WRS[random]” files are temporary MS Office files and have not been studied yet. These files are stored in the “~/Library/Containers/com.microsoft.Word/Data/Library/Application Support/Microsoft/Temp”. As a result of the analysis, the “~WRF[random]” file is a temporary file in which the actual file is stored. It is not saved the same as the actual file, but the beginning portion (approximately ‘0x600’ size) is in compound file format; later, the actual file is saved and can be restored. As the actual file is stored, all content and metadata can be restored. As can be seen in Fig. 15, the “~WRS[random]” file is also a temporary file, but the actual file is not saved; only the content of the file. The content of several document files is stored in a single file.

텍스트이(가) 표시된 사진

자동 생성된 설명

*Figure 15: Internal structure of “~WRS[random]” file contains file content*

The file “com.microsoft.Word.securebookmarks.plist” exists in the “~/Library/Containers/com.microsoft.Word/Data/Library/Preferences” folder. TheMS Office application displays the recent file list by referring to this file. This file has already been analyzed; therefore, this study does not describe the precise analysis method. The recent file’s path, name, and timestamp are saved in this file.

## **Notes**

Notes is the default notetaking application developed by Apple. This artifact is already well known, but we analyzed it in detail to find more evidence. All information related to the created note is recorded in the “NoteStore.sqlite” file, and the metadata of the deleted note can remain in the “NoteStore.sqlite-wal” file. It is saved in “~/Library/Group Containers/group.com.apple.notes”. As shown in Fig. 16, the filename, path, size, timestamp, and content are saved in this database. We focus on the database tables that contain the information related to the shared file to the Notes application. Information about files synced to iCloud is stored in the ‘ZICCLOUDSYNCINGOBJECT’ table in the database. The filename is stored in the ‘ZTITLE’ column, and the share time is stored in ‘ZCREATIONDATE’. Furthermore, the files associated with Notes are stored in the “∼/Library/Group Containers/group.com.apple.notes/Media/random”.

테이블이(가) 표시된 사진

자동 생성된 설명

*Figure 16: Internal structure of “NoteStore.sqlite” file contains filename, timestamp, and size*

# **Cloud Storage Artifacts**

ESI spoliation investigation targets e-mail, external storage devices, messengers, and cloud storage. Cloud storage is often used during crimes because it is simple to upload and download. This paper checks the traces of files left when using Google Drive and iCloud.

## **Google Drive**

Google Drive creates databases and logs to manage the files in cloud storage. These files can be used for digital forensic investigations.

The file list stored in Google Drive is recorded on “metadata\_sqlite\_db” and “mirror\_metadata\_sqlite.db” files in the path “~/Library/Application Support/GoogleDriveFS/random”. These files have not been studied or analyzed previously; therefore, we analyzed these files to find evidence to establish ESI spoliation. The “metadata\_sqlite\_db” database stores a list of all files and folders that currently exist. The “mirror\_metadata\_sqlite.db” database has the same structure as “metadata\_sqlite\_db,” and it can be said to be a file of the journal function. Even if it is completely deleted and does not remain in the recycle bin, it has traces of the file before it is deleted or modified, so it is essential to analyze.

Figure 17 shows the stored metadata we found through the experiment. The filenames, paths, and sizes are stored in several tables in these two databases. In the ‘items’ table, the filename is stored in the ‘local\_title’ column, and the size exists in the ‘file\_size’ column. The last modified time is stored in ‘modified\_date’, and the last accessed time by the user is stored in ‘viewed\_by\_me\_date’. The filename is stored in another location in this table, i.e., the ‘proto’ column. The filename also exists in the value in the ‘item\_properties’ table. It is stored in the ‘value’ column in the row where the ‘key’ column value is ‘local-title’. Next, in the ‘proto’ column in the ‘deleted\_items’ table, information related to the document file that has been deleted and moved to the recycle bin is stored.

테이블이(가) 표시된 사진

자동 생성된 설명

*Figure 17: Internal structure of “metadata\_sqlite\_db” file contains filename, timestamp, and size*

The “psid.db” file has not been studied yet, and although the file’s internal signature is SQLite, it is difficult to analyze using an incomplete format. As shown in Fig. 18, we estimate that it is a file that indexes the list of files stored in Google Drive; the list of saved filenames can be found. Even if the actual file is deleted, the filename remains. This file exists in the “~/Library/Application Support/Google/DriveFS/[random]/local\_folders/spotlight-index/Store-V2/[random]” folder.

텍스트이(가) 표시된 사진

자동 생성된 설명

*Figure 18: Internal structure of “psid.db” file contains filename*

The “structured\_log\_[random]” and “finder\_ext\_20.txt” files in the “~/Library/Application Support/Google/DriveFS/Logs” folder contain certain traces of deleted file information. These files are logs generated by Google Drive, which have not yet been disclosed or studied. As shown in Fig. 19, filenames of all files uploaded or downloaded from the Google Drive web, synchronized in a local synchronization folder, viewed, or modified are saved. However, as the structure of the “structured\_log\_[random]” file has not yet been analyzed, it is analyzed via keyword search. The “finder\_ext\_20.txt” file is a TEXT file and can be analyzed without needing a separate application. As shown in Fig. 20, this file is expected to record logs of errors and warnings that occur when viewing or accessing files in Google Drive. The accessed filename and the path where the file exist in Google Drive are stored. It also stores the timestamp of the error or warning that occurred. It is structured in text form rather than as a complex structure to be analyzed line-by-line or by keyword search.

텍스트이(가) 표시된 사진

자동 생성된 설명

*Figure 19: Internal structure of “structured\_log\_[random]” file contains filename*

테이블이(가) 표시된 사진

자동 생성된 설명

*Figure 20: Internal structure of “finder\_ext\_20.txt” file contains filename, path, and timestamp*

## **iCloud**

iCloud is a cloud developed by Apple and is used by many Apple users. All the information to be saved in iCloud Drive, files uploaded from this device, and devices that can sync with this iCloud Drive may be found in the client.db and server.db databases. Both databases are saved in the “~/Library/Application Support/CloudDocs/session/db” folder. “server.db” and “client.db” have the same structure, but only data after synchronization are stored in “server.db”. When using the cloud, it is necessary to use cloud storage without internet access, so log information is stored in “client.db”. This file records behavior information such as upload, modification, and synchronization on storage and time information changes as the behavior occurs. As shown in Fig. 21, the ‘client\_items’ table contains the filename recorded in ‘item\_filename’, while the file synchronization time is stored in ‘item\_birthtime’. ‘version\_mtime’ stores the file’s modification time, while ‘version\_size’ stores the size.

테이블이(가) 표시된 사진

자동 생성된 설명

*Figure 21: Internal structure of “client.db” file contains filename, timestamp, and size*

**Messenger Artifacts**

Numerous users send and receive document files through messengers. This study experimented with Microsoft Teams, Slack, and iChat messenger.

## **Microsoft Teams**

For the Microsoft Teams application, Cache, IndexedDB, and Local Storage folders exist in the “~/Library/Application Support/Microsoft/Teams”, and traces related to document files can be found in these folders.

When a file is downloaded from the Internet, it is saved in the Download folder. Similarly, when a file is downloaded from the Microsoft Teams application, it is saved in the download folder. However, the Microsoft Teams application saves the actual file that is viewed and downloaded as the specific format in the Cache. This file has a similar composition to the Chrome cache file. The name of this cache file is saved in the format “[random]\_0”. By substituting the document file format into the file’s internal structure, the actual document file can be restored from this cache file.

IndexedDB is a client-side data storage technology used primarily in web browsers since 2012. “[random].log” and “[random].ldb” are the names of the IndexedDB artifacts. Furkan Paligu analyzed the Microsoft Teams desktop application utilizing IndexedDB storage, but in Windows OS [39]. Through experiments, the author confirmed that information such as accounts, teams, chat messages, and video calls are stored in IndexedDB. When the ‘.log’ file reaches its size limit, the data are passed to the ‘.ldb’ file, and a new version of the ‘.log’ file appears. Information related to the transmitted and received document files are also stored in the chat record with attached media. The downloaded or viewed filename is saved in Unicode (UTF-16 little-endian) encoding. The time value for sending and receiving this document is also stored.

Another “[random].ldb” file is saved in the ‘Local Storage’ folder. In this file, information about the program, such as when the Microsoft Teams application was run and ended, is stored. As information about the downloaded file is stored, the filename, path, and timestamp can be checked.

## **Slack**

Like Microsoft Teams, the Slack application also stores the opened files in the cache file in the “~/Library/Application Support/Slack/Cache” folder. It is saved under a name in the form “[random]\_0”. There is an actual document file inside the file; content and metadata can be found in this cache. Another cache named ‘a’ stores the information related to the document file in “~/Library/Application Support/Slack/IndexedDB/https \_app.slack.com\_0.indexeddb.blob/1/00”. The filename and size are saved in this file. It must be analyzed through keyword search because the file structure has not yet been analyzed.

## **iChat**

iChat application is an instant messenger on macOS. The actual exchanged messages and attribute information are saved in the “chat.db” database file in the “~/Library/Messages” folder. Like other database files, “chat.db-wal” file contains previous data before file deletion. In the ‘attachment’ table in this database, the filename and path are recorded in the ‘filename’ column, and the timestamp is stored in the ‘created\_date’ column. Even if the actual file is deleted, the records in this database are not reflected; consequently, this file can be the evidence to track the document file.

There is a file with a name consisting of the phone number, date, timestamp, and filename, e.g., “[phone number]\_[year-month-day]\_[hour:minute:second]\_[filename].ichat”. This file is created when an attachment is sent via iChat and saved in “~/Library/Messages/Archive” folder. The filename and timestamp are on the file’s name, and the same filename is stored inside the file too. Even though there is a developed parser [40], it would be better to use a keyword search to find the filename. Furthermore, iChat saves actual attached files in “~/Library/Messages/Attachments/30/00/C3CF80BF52C5-4615-941E-C6C6BE0488EE”.

# **Web Artifacts**

Digital forensic techniques for analyzing web browsers are well known. “History.db” and “Favicons.db” are databases that store Internet history and the icons used, and the traces related to files are stored. As it is a database file, it can be easily analyzed, and the previous record might still be in ‘wal’ file. “Downloads.plist” stores the list of downloaded files; the filename and timestamp are stored in this file. “RecentlyClosedTabs.plist” stores information about recently closed tabs, but the filename and timestamp are saved when a file is opened or downloaded from the Internet.

WebKitCache is a cache file that keeps Safari records. As shown in Fig. 22, the filename remains in this cache because information such as search terms and upload and download files is also saved. This file is saved with a name as a random character in “~/Library/Containers/com.apple.Safari/Data/Library/Caches/com.apple.Safari/WebKitCache/Version 16/Records/[random]/Resource”. WebKitCache created a blob file together as a temporary file; older data might still be in this file. The structure has not yet been analyzed; the filename and timestamp must be searched for by using a keyword search.

텍스트, 전자기기, 여러개이(가) 표시된 사진

자동 생성된 설명

*Figure 22: Internal structure of “WebKitCache” file contains filename*

# **Mail**

As mentioned above, information is sometimes leaked through email and the cloud or messenger. In this study, Gmail, the most popular, and Mail, which is the default mail application of macOS, were analyzed.

## **Mail (macOS application)**

The “envelope index” database stored data related to e-mail, which is saved in “~/Library/Mail/V[random]/MailData”. When sending an e-mail with a document file attached, the filename and timestamp are saved in the “Envelop Index” and “Envelop Index-wal” databases. This database has previously been analyzed, but it was not known that the attached document’s filename can exist in “1.qdat.” The “1.qdat” file is stored in the “~/Library/Suggestions/pending” folder as a temporary file. The filename in this file is completely or partially recorded; a keyword search can identify the filename.

## **Gmail**

In Gmail, the content of the email is saved in “[random].partial.emlx” in “~/Library/Mail/V[random]/[random]/[random].mbox/[random]/[random]/Messages”. The attached filename and timestamp are saved in this file. The actual attached file is saved in the “~/Library/Mail/V6/[random]/[random].mbox/[random]/Data/[random]/Attachments/[random]/[random]” folder.

# **References**

[1] M. Luoma, V. Luoma, After five years of e-discovery missteps: Sanctions or safe harbor?, Annual ADFSL Conference on Digital Forensics, Security and Law.

[2] T. Y. Allman, Managing preservation obligations after the 2006 federal e-discovery amendments, Rich. JL & Tech. 13 (2006) 1.

[3] R. Durrant, Vii. spoliation of discoverable electronic evidence, Loyola of Los Angeles Law Review 38 (4) (2005) 1803.

[4] D. B. Garrie, Digital forensic evidence in the courtroom: understanding content and quality, Nw. J. Tech. & Intell. Prop. 12 (2014) i.

[5] L. Daniel, Digital forensics for legal professionals: understanding digital evidence from the warrant to the courtroom, Elsevier, 2011.

[6] D. B. Oh, K. H. Park, H. K. Kim, De-wipimization: Detection of data wiping traces for investigating ntfs file system, Computers & Security 99 (2020) 102034.

[7] I. Mitchell, T. Anandaraja, S. Hara, G. Hadzhinenov, D. Neilson, Deconstruct and preserve (dap): a method for the preservation of digital evidence on solid state drives (ssd), in: International Conference on Global Security, Safety, and Sustainability, Springer, 2017, pp. 3–11.

[8] S. Institute, Windows forensics analysis, online; accessed 15-March-2022 (2021).  
URL https://www.sans.org/security-resources/posters/windows-forensic-analysis/170/download

[9] D. Jeong, S. Lee, Study on the tracking revision history of ms word files for forensic investigation, Digital Investigation 23 (2017) 3–10.

[10] J. Joun, H. Chung, J. Park, S. Lee, Relevance analysis using revision identifier in ms word, Journal of Forensic Sciences 66 (1) (2021) 323–335.

[11] S. Schoenhardt, Macintosh apfs forensic software assessment: Blackbag technologies blacklight 2019 r3, Ph.D. thesis, Utica College (2020).

[12] E. Casey, Digital evidence on macintosh systems, Digital Evidence and Computer Crime (2011) 587.

[13] T. S. Atwal, M. Scanlon, N.-A. Le-Khac, Shining a light on spotlight: Leveraging apple’s desktop search utility to recover deleted file metadata on macos, Digital Investigation 28 (2019) S105–S115.

[14] K. Kafadar, The need for objective measures in forensic evidence, Significance 16 (2) (2019) 16–20.

[15] J. Andress, S. Winterfeld, Chapter 12 - legal system impacts, in: J. Andress, S. Winterfeld (Eds.), Cyber Warfare, Syngress, 2011, pp. 207–223.

[16] H. C. Black, B. A. Garner, Black’s law dictionary, 11th Edition, St. Paul, MN, 2019.

[17] Union pacific r.r. co. v. barber.

[18] The federal rule of civil procedure.

[19] Kronisch v. united states.

[20] Silvestri v. gen. motors corp.

[21] J. Oh, S. Lee, H. Hwang, Ntfs data tracker: Tracking file data history based on $logfile, Forensic Science International: Digital Investigation 39 (2021) 301309.

[22] H. Chung, J. Park, S. Lee, Forensic analysis of residual information in adobe pdf files, in: Future Information Technology, Springer, 2011, pp. 100–109.

[23] J. Park, S. Lee, Forensic investigation of microsoft powerpoint files, Digital Investigation 6 (1-2) (2009) 16–24.

[24] B. Maddu, P. Rao, Os x artifact analysis, International Journal of Recent Technology and Engineering 7 (2019) 26–32.

[25] S. Bunting, Forensic analysis of spoliation and other discovery violations part 1: Macintosh examinations, eForensics Magazine. URL https://eforensicsmag.com/

[26] K. A. Schuler, E-discovery: Creating and managing an enterprisewide program: A technical guide to digital investigation and litigation support, Syngress, 2011.

[27] Y. Liu, M. Xu, J. Xu, N. Zheng, X. Lin, Sqlite forensic analysis based on wal, in: International Conference on Security and Privacy in Communication Systems, Springer, 2016, pp. 557–574.

[28] Spotlight overview (Aug 2013). URL  
https://developer.apple.com/library/archive/documentation/Carbon/Conceptual/MetadataIntro/MetadataIntro.html

[29] Y. Khatri, Investigating spotlight internals to extract metadata, Digital Investigation 28 (2019) 96–103.

[30] Y. Khatri, Unifiedlogreader (1 2020). URL https://github.com/ydkhatri/UnifiedLogReader

[31] G. Software, Apple system log (asl) file parser. URL https://security.opentext.com/appDetails/Apple-System-Log-ASL-File-Parser

[32] G. Software, Plist parser. URL https://security.opentext.com/appDetails/Plist-Parser

[33] J. M. Garijo, Mac os x forensics, Tech. rep., Technical Report, RHUL-MA-2015-8 (2015).

[34] S. Newcomer, L. Martin, Determining user actions in os x based on quicklook thumbnail cache database entries., Issues in Information Systems 15 (2).

[35] S. Neef, Parsing the .ds store file format (Mar 2018). URL https://0day.work/parsing-the-ds\\_store-file-format/

[36] G. Software, Ds store parser. URL https://security.opentext.com/appDetails/DS-Store-Parser

[37] N. Ibrahim, Understanding macos file system events with fseventsparser, in: Proc. of the 8th Annual Open Source Digital Forensics Conference, Herndon, VA, 2017.

[38] N. Ibrahim, Fseventparser (2019). URL https://github.com/dlcowen/FSEventsParser

[39] F. Paligu, C. Varol, Microsoft teams desktop application forensic investigations utilizing indexeddb storage, Journal of Forensic Sciences.

[40] G. Software, ichat message parser. URL https://security.opentext.com/appDetails/iChat-Message-Parser