Vol.2020-CSEC-91 No.2
Vol.2020-SPT-40 No.2
Vol.2020-EIP-90 No.2
2020/11/25

NTFSにおけるUsnJrnlを活用したタイムライン分析手法の提案と実装

阿部 拓真[1,a)]　上原 哲太郎[2,b)]

概要：コンピュータ犯罪や不正といったインシデントの解析手法としてタイムライン分析があるが，従来のタイムライン分析ツールの多くはOS内で発生した各種イベントを文字の状態で時系列に並べるのみであり，容易に詳細な分析をすることが困難であった．本研究ではNTFSのメタデータやUsnJrnl，Windows 10の機能である「Recent Files」(最近開かれたファイル)の履歴を活用してファイルの作成・変更・削除の履歴を過去に遡って調査し，その結果を用いてタイムライン図で描画するタイムライン分析手法の提案を行う．

Design and Implementation of Timeline Analysis for NTFS Forensics Utilizing UsnJrnl

1. はじめに

コンピュータ犯罪や不正といったインシデントが発生した際の調査や解析の手法として，コンピュータに保存されたデータをデジタル・フォレンジックの技術を用いて保全し，それを解析することが一般的となってきている．コンピュータに接続される補助記憶装置として，Hard Disk Drive(以下，「HDD」とする)に比べて読み書きに掛かる速度が速いことが特徴であるSolid State Drive(以下，「SSD」とする)を使用することが増加している．一方で，SSDには新たなファイルを書き込もうとする領域に存在する過去のデータを予め削除した上で書き込まなければならないという仕様上の制約があり，一般的なSSDにおいてはOSによってファイルが削除されたと同時にSSDに保存されたデータの一部が削除される．そのため，SSDのデジタル・フォレンジックにおいて，HDDに対するデジタル・フォレンジックで多用されてきた「OSによって削除されたファイルの復元を行う」手法を用いることは不可能となった．

そこで，本研究ではファイルの復元を前提とせず，SSD内に保存され続けるデータを活用してデジタル・フォレンジックを行うことを目指した．そこで，NTFSのメタデータでありNTFSでフォーマットされた論理ドライブ内に保存された全てのファイルやフォルダの情報を保存する「MFTファイルレコード」や，NTFSの変更ジャーナルを記録する「$UsnJrnl」ファイル，Windows OSで保存されるログのうち，直近にExplorer.exeで開かれたファイルを記録する「Recent files」機能に使用されるログを活用し，これらのデータを解析するファスト・フォレンジックを行うことでタイムスタンプを抽出し，その結果を元にタイムライン図を描画するタイムライン分析方法を提案し，その実装を行う．

[1] 立命館大学大学院 情報理工学研究科
[2] 立命館大学 情報理工学部
[a)] abe@cysec.cs.ritsumei.ac.jp
[b)] t-uehara@fc.ritsumei.ac.jp

2. 研究背景

2.1 SSDのTrim命令

1章で述べたように，SSDでは新たなデータを書き込む前に過去のデータをブロック単位で削除する必要がある．そこで，多くのSSDでは書き込み速度の高速化のために「Trim命令」と呼ばれる機能が自動的に動作する設定となっている．これは，OSによって当該のファイルやフォルダが削除された際に，OSならびにSSDのファームウェアによってそれらの実データが保存された領域を削除し空の状態を予め作成する操作を自動で行う機能である．そのため，Trim命令が有効となっているSSDに対する削除されたファイルやフォルダのデータのうち，そのファイルやフォルダ内に含まれる実データの復元は不可能である．一

ⓒ 2020 Information Processing Society of Japan

方で，削除されたファイルやフォルダに関するメタデータの一部は Trim 命令によって削除されない．また，OS やアプリケーションに依存するログの形態によっては，削除されたファイルやフォルダに関係のある他のファイルやフォルダが存在する場合がある．そこで，これらのデータを活用するデジタル・フォレンジック手法が求められている．

2.2 ファスト・フォレンジック

ファスト・フォレンジックとは，「早急な原因究明・侵入経路や不正な挙動を把握するため，必要最低限のデータを抽出及びコピーし，解析すること」[1] である．一般に，ファスト・フォレンジックの利点として，結果が出るまでの時間が短いことが挙げられる．一方で，補助記憶装置内に含まれるデータの全てや論理ドライブ内に含まれるデータの全てを解析対象とし削除ファイルの復元をするような従来の手法に比べて解析結果に対する証拠能力が低いことが欠点であるとされている．しかし，SSD に含まれるデータに対する消去ファイルの復元を前提とした従来の手法で解析を行うことは，2.1 節で挙げた理由により実現不可能である．そこで，SSD に含まれる全てのデータに対する保全を諦め，解析時点で SSD に存在する部分的なデータのみを保全し，それに対して解析することで SSD に対応したファスト・フォレンジックを実現できると考えられる．

2.3 NTFS とメタデータ

NTFS とは，主に Windows NT 系で用いられているファイルシステムである．NTFS の特徴として，OS によってファイルが削除する操作がされても即座に実データやメタデータが削除されず，新たなファイルが同じ領域に上書きされるまで残り続けることが挙げられる．また，全てのファイルやフォルダをファイルとして扱い，そのファイルごとに MFT ファイルレコードが生成されることも特徴の一つである．

2.3.1 NTFS の構造

NTFS で管理された論理ドライブは必ず 1 つの MFT(Master File Table) と呼ばれるメタデータが存在しており，その中には論理ドライブ内に保存されたそれぞれのファイルやフォルダに紐付いた「MFT ファイルレコード」が格納されている．また，MFT ファイルレコード自身もファイルであり，各エントリは 1KB の固定長である．各ファイルのメタデータの多くはこの MFT ファイルレコードに保存されているが，格納しきれない情報は別の特殊ファイルとして格納されている場合がある．これを「NTFS メタデータファイル」と呼ぶ．MFT 自身も NTFS メタデータファイルとして管理されている．

2.3.2 MFT ファイルレコードに含まれる情報

MFT ファイルレコードは，1 つの「Attribute Header」と，複数の「Attribute」と呼ばれるフォーマットによって構成されている．

Attribute Header は，主にその MFT ファイルレコードに含まれる Attribute の情報が含まれている．Attribute Header には，ファイルの実データが MFT ファイルレコード内に保存されているかを示す「Resident」属性，ならびにファイル名に関する情報がその MFT ファイルレコードに含まれるかを示す「Name」属性によって 4 種類が存在する．そのうち，最も一般的である「No Resident, Named」属性である MFT ファイルレコードにて使用される Attribute Header には，表 1 に示す情報が格納されている [2].

表 1 Attributes Header の内容 [2]

Offset	Size	Value	Description
0x00	4		Attribute Type (e.g. 0x80, 0xA0)
0x04	4		Length (including this header)
0x08	1	0x01	Non-resident flag
0x09	1	N	Name length
0x0A	2	0x40	Offset to the Name
0x0C	2		Flags
0x0E	2		Attribute Id (a)
0x10	8		Starting VCN
0x18	8		Last VCN
0x20	2	2N+0x40	Offset to- the Data Runs (b)
0x22	2		Compression- Unit Size (c)
0x24	4	0x00	Padding
0x28	8		Anyocated size- of the attribute (d)
0x30	8		Real size of the attribute
0x38	8		Initialized data size of- the stream (e)
0x40	2N		Unicode The Attribute's Name
2N+0x40	...		Data Runs (b)

次に，Attribute の一覧を表 2 で示す [2]．Attribute には 17 種類のタイプが存在しているが，主にテキストファイルやプログラムといった，一般にコンピュータの利用者が使うようなデータに紐付く MFT ファイルレコードには「$STANDARD_INFORMATION」,「$FILE_NAME」,「$DATA」の 3 種類の Attribute が含まれている [2].

$STANDARD_INFORMATION は, そのファイルやフォルダの基本情報が含まれる Attribute である．$STANDARD_INFORMATION Attribute には，表 3 に示す情報が格納されている [2]．このうち，「File Creation Time」,「File Altered Time」,「MFT Changed Time」,「File Read Time」はタイムスタンプであり，それぞれ「ファイルが作成された時刻」,「ファイルが変更された時刻」,「MFT ファイルレコードが変更された時刻」,「ファイルが読み込まれ

表 2 Attributes 一覧 [2]

Type	OS	Name
0x10	Any	$STANDARD_INFORMATION
0x20	Any	$ATTRIBUTE_LIST
0x30	Any	$FILE_NAME
0x40	NT	$VOLUME_VERSION
0x40	2K	$OBJECT_ID
0x50	Any	$SECURITY_DESCRIPTOR
0x60	Any	$VOLUME_NAME
0x70	Any	$VOLUME_INFORMATION
0x80	Any	$DATA
0x90	Any	$INDEX_ROOT
0xA0	Any	$INDEX_AnyOCATION
0xB0	Any	$BITMAP
0xC0	NT	$SYMBOLIC_LINK
0xC0	2K	$REPARSE_POINT
0xD0	Any	$EA_INFORMATION
0xE0	Any	$EA
0xF0	NT	$PROPERTY_SET
0x100	2K	$LOGGED_UTILITY_STREAM

表 3 $STANDARD_INFORMATION Attribute に含まれる情報 [2]

OS	Description
Any	File Creation
Any	File Altered
Any	MFT Changed
Any	File Read
Any	DOS File Permissions
Any	Maximum Number of Versions
Any	Version Number
Any	Class Id
2K	Owner Id
2K	Security Id
2K	Quota Charged
2K	Update Sequence Number (USN)

た時刻」を示す．これらのタイムスタンプは，ユーザレベルのプロセスによって変更されるため，ユーザが利用したことによって更新されるタイムスタンプが反映されることから有意である．一方で，ユーザによる変更や改竄がしやすいことが欠点として挙げられる．

表 3 にある「DOS File Permissions」は，表 4 で示すフラグが保存されている．このフラグを確認することで，ファイルやフォルダを判別したり，削除済みファイルか否かの確認をしたりすることが可能となる [2].

$FILE_NAME は，主にそのファイルやフォルダの名前情報が含まれる Attribute である．$FILE_NAME の Attribute は，そのファイルやフォルダの名前に「拡張子以外に 9 バイト以上」，もしくは「拡張子に 4 バイト以上」の文字が含まれている場合に限り 2 つ目の$FILE_NAME Attribute が存在する場合がある．これは，Windows OS にて用いられる「8.3 形式」と呼ばれる短い名前が自動生成されるためである．$FILE_NAME Attribute には，表 5 に示す情報が格納されている [2]．このうち，「Filename namespace」はファイル名が Unicode で格納されている．また，$FILE_NAME Attribute には$STANDARD_INFORMATION Attribute と同様の意味を持つタイムスタンプ群が格納されているが，これらのタイムスタンプは，システムカーネルによってのみ変更が可能であるため，ユーザレベルのプロセスで更新されたタイムスタンプは反映されない．一方で，これらのタイムスタンプの変更や改竄はユーザレベルで変更することができないことから困難である利点がある [3][4].

表 4 DOS File Permissions フラグ一覧 [2]

Flag	Description
0x0001	Read-Only
0x0002	Hidden
0x0004	System
0x0020	Archive
0x0040	Device
0x0080	Normal
0x0100	Temporary
0x0200	Sparse File
0x0400	Reparse Point
0x0800	Compressed
0x1000	Offline
0x2000	Not Content Indexed
0x4000	Encrypted

表 5 $FILE_NAME Attribute に含まれる情報 [2]

OS	Description
Any	File reference to the parent directory.
Any	File Creation
Any	File Altered
Any	MFT Changed
Any	File Read
Any	Anyocated size of the file
Any	Real size of the file
Any	Flags, e.g. Directory, compressed, hidden
Any	Used by EAs and Reparse
Any	Filename length in characters
Any	Filename namespace

そのため，タイムライン分析では基本的に$STANDARD_INFORMATION Attribute のタイムスタンプを用いるものの，併せて$FILE_NAME Attribute も活用することが有効であると言える．

$DATA は，ファイル内にある実データに関する情報が含まれている Attribute である．NTFS では，容量が小さいファイルでは，それに紐付く Attribute Header の Resident 属性が「Resident」に設定され，実データは MFT ファイルレコードに格納される．また，容量が大きいファイルでは，それに紐付く Attribute Header の Resident 属性が

「No Resident」に設定され，$DATA Attribute 内に実データが保存された場所が保存される．

2.3.3 $UsnJrnl

メタデータのうち$UsnJrnl(Update Sequence Number Journal,「更新ジャーナル」) は，ファイルシステムに加えられた変更の記録を保存する NTFS のメタデータである [5]．また，Windows 10 で使用されている USN_RECORD_V2 は，Windows 8.1，Windows Server 2012 R2 以降の Windows OS において標準で有効である．$UsnJrnl に含まれる情報は，以下の 9 項目である．

- 変更時刻
- 変更理由
- ファイル・フォルダの属性
- ファイル・フォルダの名前
- ファイル・フォルダの MFT ファイルレコード番号
- ファイルの親フォルダのファイルレコード番号
- セキュリティ ID
- レコードの更新シーケンス番号 (USN)
- 変更のソースに関する追加情報

これらのうち，「変更理由」は，表 6 で示す 22 種類のフラグによって管理されている [6]．これを解析することにより，コンピュータがファイルシステムに対して行った操作の履歴を辿ることが可能となる．また，2.3.2 節で示した MFT ファイルレコードにて取得したタイムスタンプは，最新のタイムスタンプに限った解析が可能である．そのため，過去に遡ったタイムスタンプの取得は不可能である．一方で，$UsnJrnl は NTFS が更新される度にレコードが生成され，このデータが残り続けることから，過去のタイムスタンプについて解析することが可能となる．そのため，$UsnJrnl を活用したデジタル・フォレンジックは有効であると言える．

表 6 $UsnJrnl の「変更理由」の一覧 [6]

USN_REASON	内容
DATA_OVERWRITE	データ上書き
DATA_EXTEND	データ追記
DATA_TRUNCATION	データ切り詰め
NAMED_DATA_OVERWRITE	ストリームデータ上書き
NAMED_DATA_EXTEND	ストリームデータ追記
NAMED_DATA_TRUNCATION	ストリームデータ切り詰め
FILE_CREATE	作成
FILE_DELETE	削除
EA_CHANGE	拡張属性（EA）の変更
SECURITY_CHANGE	アクセス権の変更
RENAME_OLD_NAME	ファイル名の変更（前）
RENAME_NEW_NAME	ファイル名の変更（後）
INDEXABLE_CHANGE-NOT_INDEXED	属性の変更
BASIC_INFO_CHANGE	属性/タイムスタンプの変更
HARD_LINK_CHANGE	ハードリンクの追加/削除
COMPRESSION_CHANGE	圧縮状態の変更
ENCRYPTION_CHANGE	暗号状態の変更
OBJECT_ID_CHANGE	オブジェクト ID の変更
REPARSE_POINT_CHANGE	リパースポイントの変更
STREAM_CHANGE	ストリームの変更
TRANSACTED_CHANGE	ストリーム変更（TxF）
INTEGRITY_CHANGE	INTEGRITY 変更（ReFS）
CLOSE	クローズ

2.4 Recent Files とショートカットファイル

「Recent Files」は，Explorer.exe で開かれたファイルのショートカットを自動的に `%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Recent\` フォルダに作成する，Windows OS の機能である．Recent Files のログは一部のコンピュータでは無効となっている場合がある．そのため，本ツールを使用する際はこの機能が有効に設定されている必要がある．

ショートカットファイル内には，「実ファイルへのパス」や，MFT ファイルレコード内に含まれるタイムスタンプとは別に「Creation time」,「Last access time」,「Last modification time」の 3 種類のタイムスタンプ等の情報が含まれている [7]．

2.5 タイムライン分析とその信頼性

タイムライン分析とは，デジタル・フォレンジックにてよく用いられている分析手法の一つであり，ログやファイルやフォルダのタイムスタンプなどを時系列で整理することで，コンピュータ上で起こったイベントを分析する手法 [8] である．この手法により，判明している挙動を起点に，前後のシステムの動きを確認することで何が行われたかをより容易に解析することが可能となる．一方で，タイムライン分析に用いるログやファイルやフォルダのタイムスタンプは改竄される可能性があるが，部分的な改竄が行われた場合はタイムライン分析を行った結果に矛盾が生じることが予想される．そのため，メタデータの部分的な改竄が行われた際にはタイムライン分析を行うことで見いだせる可能性がある．

3. 関連研究

3.1 SSD 内の消去ファイルにおける復元の困難性

山前らは「SSD 上の消去ファイルの復元可能性の実験と評価」[9] という論文を発表している．この論文では，SSD の Trim 命令の有無，削除後の利用状況の差異，OS の差異に伴うデータの復元に対する確率について検証している．この論文によると，Trim 命令が無効の際はデータの多くが削除してから一日しか残っておらず，Trim 命令が有効の際は削除直後でもデータが残っていないことが明らかと

なった．これより，SSD におけるファイルの復元を前提としたデジタル・フォレンジックは不可能であると言える．

3.2 タイムライン分析

log2timeline や Plaso[10] は，オープンソースのスーパータイムラインを生成するツールである．このツールでは，様々なログを収集することが可能である．一方で，解析結果は csv 形式で出力されることから，結果が文字列で表示されることとなるため，タイムライン解析に慣れていない人が解析を行うことは難しいと言える．

Autopsy[11] は、補助記憶装置内に含まれるデータを調査する際に使用されているフリーのデジタル・フォレンジックツールである。このツールには「タイムライン」と呼ばれる機能があり，調査時点で補助記憶装置に保存されているファイルに紐付いたタイムスタンプを元にタイムライン図を描画する機能がある．しかし，MFT ファイルレコード内に格納された$STANDARD_INFORMATION の Attribute 内に存在する「File Creation Time」,「File Altered Time」,「MFT Changed Time」,「File Read Time」に限ったタイムスタンプを活用した図であり，詳細なタイムライン分析を行うことは困難である．また，MFT ファイルレコードに含まれるタイムスタンプは最新のものに限って保持することから，過去のタイムスタンプを遡ることは不可能である．

4. 提案手法

4.1 想定する状況

本研究では，コンピュータのユーザによってファイルが削除されたり，ファイル自身やファイルのタイムスタンプが変更されたりする状況を想定する．

4.2 本研究で提案する手法

本研究では，4.1 節で示した状況を解決するファスト・フォレンジック手法として NTFS の MFT と$UsnJrnl, Recent Files 内のショートカットに含まれるタイムスタンプを解析し，その結果をタイムラインとして図に描画することで，タイムライン分析をより容易にすると同時に，2.5 節で示したようなタイムスタンプの改竄を発見しやすくすることを目指すツールを実装し，手法の提案を行うこととした．

本研究で提案するツールの仕様は，以下の通りとした．

(1) Windows 10 で動作するコンピュータを対象とすること
(2) MFT ファイルレコードの解析によって，タイムスタンプを取得する機能を持つこと
(3) MFT ファイルレコードの解析によって得られた情報を元にファイルパスを生成する機能を持つこと
(4) $UsnJrnl の解析によって，タイムスタンプと$UsnJrnl が持つ「変更理由」の 2 つを取得する機能を持つこと
(5) ショートカットファイルに含まれるタイムスタンプと，実データがある場所のパスを取得する機能を持つこと
(6) 各種ファイルに解析によって得られたタイムスタンプからタイムライン分析が可能な図を描画すること

上記で示したツールの仕様のうち 3 番目に示したファイルパスを生成する機能を仕様とすることについて，以下の理由が挙げられる．

(1) MFT ファイルレコードは，ファイル名のみを保持し，ファイルパスについては文字列として保持しない
(2) ショートカットファイルでは，実データの場所をファイルパスで保持している

ファイルパスを生成する機能を仕様とする理由のうち 1 番目については，表 5 で示したように MFT ファイルレコードの$FILE_NAME Attribute が親ファイル・フォルダに割り当てられたエントリ番号 (File reference to the parent directory) とファイル名 (Filename namespace) の情報を保持する一方で，ファイルパスは生成されていない．そのため，本ツールにてこれらの情報を解析することでファイルパスを生成することが可能となる．また，2 番目については，ショートカットファイル内にエントリ番号が保存されない仕様となっている．これらの理由により，MFT ファイルレコードの解析によって得られた情報とショートカットファイルを結びつけるにはファイルパスが必要となる．そのため，本ツールにてファイルパスを生成する機能を保持することが必須となる．

5. 実装

5.1 概要

本研究で実装するツールは，機能別に以下の 4 つのプログラムに分割して作成した．また，ツール開発にあたって，プログラミング言語として Python を使用した．また，MFT ファイルレコードの解析には Libyal/Libfsntfs(Pyfsntfs) ライブラリ [12] を，ショートカットファイルの解析には Libyal/Liblnk(Pylnk) ライブラリ [13] を，タイムライン図の描画には Matplotlib[14] をそれぞれ使用した．

5.1.1 MFT ファイルレコードの解析

MFT ファイルレコードの解析は，解析対象となる論理ドライブに保存された MFT を対象に行った．この際，全ての MFT ファイルレコードを解析することによりファイルとフォルダの親子関係を求めた．

5.1.2 $UsnJrnl の解析

$UsnJrnl の解析は，解析対象となる論理ドライブに保存された$UsnJrnl を対象に行った．この際，全ての$UsnJrnl レコードを解析することにより，MFT ファイルレコードで取得することが難しい過去のタイムスタンプの収集をより容易することとした．

ⓒ 2020 Information Processing Society of Japan

5.1.3 ショートカットファイルの解析

Recent Files のショートカットの解析は，該当のディレクトリ内に保存された全てのショートカットファイルを対象にそれぞれ行い，ショートカットファイルの中から実ファイルへのパスと，3 種類のタイムスタンプを取得した．

5.1.4 解析した情報の結合とタイムライン図の作成

5.1.1 節，5.1.2 節，5.1.3 節で収集・解析した情報を元に，以下のタイムスタンプをファイルごとに収集した．なお，解析対象とするファイルに関する MFT ファイルレコード内に存在するタイムスタンプ以外のタイムスタンプは，状況によっては存在しない場合がある．その際は，取得可能なタイムスタンプに限って取得することとする．

(1) 解析対象とするファイルに関するタイムスタンプ
- MFT ファイルレコード内の$STANDARD_INFORMATION Attribute に保存された「File Creation Time」，「File Altered Time」，「MFT Changed Time」，「File Read Time」の各タイムスタンプ
- MFT ファイルレコード内の$FILE_NAME Attribute に保存された「File Creation Time」，「File Altered Time」，「MFT Changed Time」，「File Read Time」の各タイムスタンプ
- $UsnJrnl 内に保存された「変更理由」とそれが発行されたタイムスタンプ

(2) 解析対象とするファイルに紐付く Recent Files のショートカット
- MFT ファイルレコード内の$STANDARD_INFORMATION Attribute に保存された「File Creation Time」，「File Altered Time」，「MFT Changed Time」，「File Read Time」の各タイムスタンプ
- MFT ファイルレコード内の$FILE_NAME Attribute に保存された「File Creation Time」，「File Altered Time」，「MFT Changed Time」，「File Read Time」の各タイムスタンプ
- $UsnJrnl 内に保存された「変更理由」とそれが発行されたタイムスタンプ
- そのショートカット内に保存された「Creation time,「Last access time,「Last modification time」の各タイムスタンプ

これらのタイムスタンプを収集したのち，それぞれのファイルやフォルダごとに 1 つのタイムライン図を描画した．

6. 評価

6.1 評価方法

本研究では，解析対象とするコンピュータとして，表 7 に示す仮想マシンを用いて検証した．

表 7 解析対象のコンピュータ

OS	Windows 10 バージョン 2004
マシン	Oracle VM Virtualbox バージョン 6.1.12
ディスク	30GB vhd ファイル

本研究ではファイルを以下のように操作した．

(1) `%USERPROFILE%\Desktop\` に `hoge.txt` を作成する
(2) `hoge.txt` を Windows 10 に付属している Notepad.exe を用いて開き，文字の入力を行う
(3) `hoge.txt` を保存し，Notepad.exe を終了する

これらの操作を行った後に仮想マシンを終了し，以下の操作を行った．

(1) 仮想マシンを動作していたホストコンピュータに，仮想マシンで使用した仮想ドライブをホストコンピュータの G ドライブとしてマウントした
(2) FTK Imager[15] バージョン 4.3.0.18 を使用し仮想ドライブ内から MFT ファイルレコードと$UsnJrnl を抽出した
(3) Recent Files は，G ドライブ内の所定の位置にあるショートカットファイルを解析対象とした
(4) 本研究で提案するツールを用いて解析した

6.2 結果

本ツールを用いてタイムライン分析を行った際に得られたタイムラインの図を，図 1 に示す．

図 1 では，タイムスタンプの種別を以下のように示した．
- MFT ファイルレコード内の$STANDARD_INFORMATION から取得したタイムスタンプの先頭には「$SN」を付けた
- MFT ファイルレコード内の$FILE_NAME から取得したタイムスタンプの先頭には「$FN」を付けた
- $UsnJrnl から取得したタイムスタンプの先頭には「USN/」を付けた
- Recent Files にあるショートカットから取得したタイムスタンプの先頭には「<R>」を付けた

6.3 考察

6.2 節に示した結果より，複数の情報源から取得したタイムスタンプを 1 つのタイムライン図に図示することで，各タイムスタンプの前後関係が解析しやすくなった．これにより，タイムスタンプを含むログを文字列として並べる方式を用いたタイムライン分析に比べてより容易にタイムライン分析を行うことが可能であると言える．また，ユーザによってタイムスタンプが改竄された際に，タイムスタンプの順序が変わるような改竄が行われた場合は図 1 のようなタイムライン図を用いて解析することで，より容易にタイムスタンプの発見することが可能になると言える．

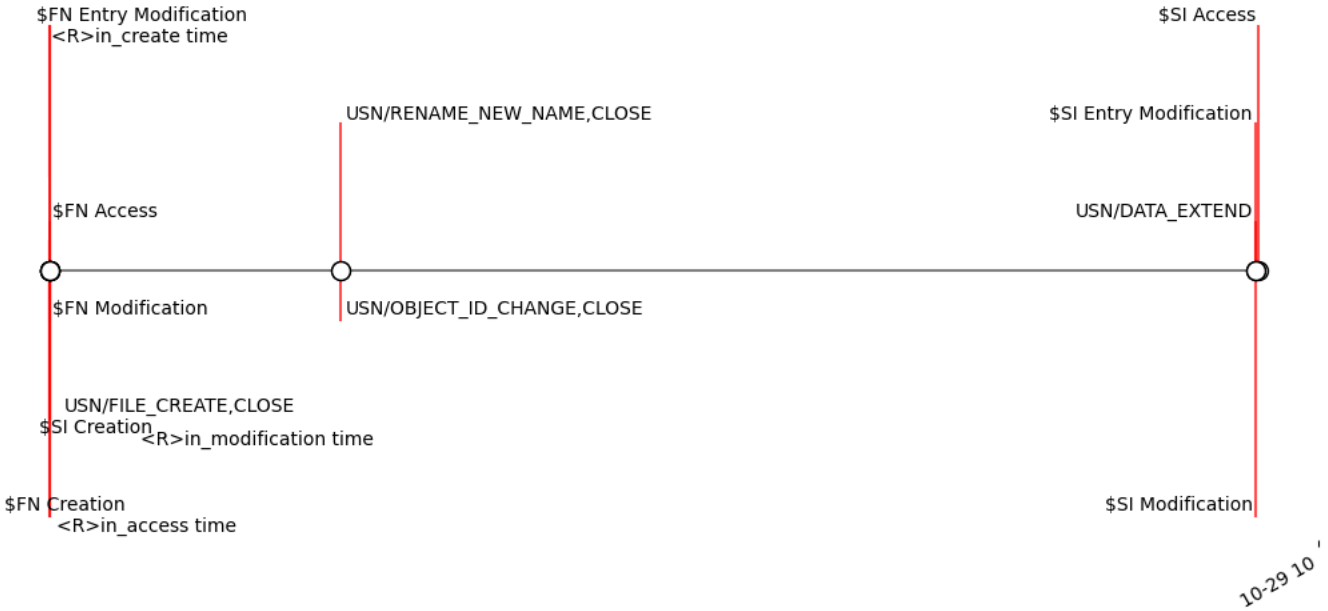

図 1 タイムライン図の描画結果

7. 今後の課題

本研究で提案した手法により，昨今一般的に行われているようなタイムライン分析手法である「ログを文字のまま表に並べ，それを時系列順に解析する」ことよりも容易な解析手法を示すことが可能となった．また，Autopsy を始めとする既存のデジタル・フォレンジックツールでは MFT ファイルレコードに保存されたタイムスタンプに限って解析を行うタイムライン分析が主流であったため，過去のタイムスタンプの取得は難しかったが，本研究では$UsnJrnl や Recent Files の活用により過去に遡ったタイムスタンプの取得方法を示すことが可能であることが示された．

一方で，コンピュータには本研究にて使用したタイムスタンプ以外にも様々なログが残っている．具体例として，以下のファイルが挙げられる．

(1) Windows 10 のイベントログで収集される各種ログ
(2) Windows 10 のレジストリに保存されたデータ
(3) Windows 10 のタスクビューで表示される最近使われたファイルの「アクティビティ履歴」
(4) exe 形式のプログラムが実行された際に生成される Prefetch ファイル
(5) ブラウザソフトに保存された Web ページの閲覧履歴
(6) IME を使用して入力された文字のログ

一般に，コンピュータインシデントが発生する以前に，初期設定では収集・保存しないログを収集するために設定が変更される状況は少ないと考えられる．そこで，上記に挙げたような「初期設定のまま使用した場合に収集・保存されるログ」を活用したツールを開発し，デジタル・フォレンジックに活用することが望まれる．また，それらのログを組み合わせて解析することで，各タイムスタンプ単独では算出することが難しい「ファイルやディレクトリを開いた時刻」を求め，ガントチャートのような図で描画することも今後の課題として挙げられる．

参考文献

[1] デジタル・フォレンジック研究会. 証拠保全ガイドライン第 8 版. https://digitalforensic.jp/wp-content/uploads/2020/06/guideline_8thv1.10.pdf. (参照:2020-9-15).
[2] Richard Russon,Yuval Fledel. NTFS Documentation. http://dubeyko.com/development/FileSystems/NTFS/ntfsdoc.pdf, 2000. (参照:2019-1-15).
[3] Tony Knutson. SANS Institute Information Security Reading Room, Filesystem Timestamps: What Makes Them Tick? https://www.sans.org/reading-room/whitepapers/forensics/filesystem-timestamps-tick-36842. (参照:2020-9-15).
[4] Brian Carrier. *File System Forensic Analysis*. Addison-Wesley Professional, 2005.
[5] Windows Dev Cnter. USN_RECORD_V2 structure. https://docs.microsoft.com/en-us/windows/win32/api/winioctl/ns-winioctl-usn_record_v2. (参照:2020-9-15).
[6] 山崎 輝. USN ジャーナル解析の追求 (Japan Security Analyst Conference 2018). https://www.jpcert.or.jp/present/2018/JSAC2018_03_yamazaki.pdf. (参照:2020-10-28).
[7] Microsoft. [MS-SHLLINK]: Shell Link (.LNK) Binary File Format. https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-shllink/16cb4ca1-9339-4d0c-a68d-bf1d6cc0f943. (参照:2020-10-25).
[8] 佐々木 良一, 上原 哲太郎, 櫻庭 信之, 白濱 直哉, 野崎 周作, 八槇 博史, 山本清子. デジタル・フォレンジックの基礎と実践. 東京電機大学出版局, 2017.
[9] 山前 碧，小林 裕太，上原 哲太郎，佐々木 良一. SSD 上

ⓒ 2020 Information Processing Society of Japan

の消去ファイルの復元可能性の実験と評価. 研究報告マルチメディア通信と分散処理 (DPS), Vol. 2015, No. 39, pp. 1–7, 2015.
[10] log2timeline/plaso. https://github.com/log2timeline/plaso. (参照:2020-9-15).
[11] Basis Technology. Autopsy. https://www.autopsy.com. (参照:2020-9-15).
[12] libyal/libfsntfs. https://github.com/libyal/libfsntfs. (参照:2020-10-28).
[13] libyal/liblnk. https://github.com/libyal/liblnk. (参照:2020-10-28).
[14] Matplotlib. https://matplotlib.org/. (参照:2020-10-28).
[15] Access Data. FTK Imager. https://accessdata.com/products-services/forensic-toolkit-ftk. (参照:2020-10-28).

ⓒ 2020 Information Processing Society of Japan