Timeline Analysis

Timeline Analysis 101 (1)

- What is Timeline Analysis?
  - Chronologically arranged data of various artifacts such as filesystem metadata entries and registry entries.

- Why Timeline Analysis?
  - It helps us to know what occurred before or after a given event.
    - For example, if we know the initial infection time, we can use timelines to know what file was created, modified, and deleted before the time. That can show us what caused the infection. We can also use timelines to estimate what attackers did after the infection.

Timeline Analysis 101 (2)

- How to analyze timeline (1)
  - Timelines often become large. In real case, each timeline contains millions of lines and that could be several hundred Mbytes in size. It is impossible for humans to view a whole timeline.
  - The most important purpose of timeline analysis is to know what occurred before or after a given event. Those events which investigators focus on are called "pivot points".

Timeline Analysis 101 (3)

- How to analyze timeline (2)
  - First, you should find “pivot points” by other way such as examining auto-start locations or program execution artifacts. Then you can examine timelines by investigating around those lines as starting points.
  - Usually “pivot points” indicate certain time frames and/or file paths. Thus we can decrease lines to investigate by applying those filters to the timelines.

Timeline Analysis 101 (4)

- How to analyze timeline (3)
  - Sometimes investigators build a "super-timeline" by gathering many kinds of timestamps such as registry, event logs, program execution artifacts first. But it could be filled with unnecessary events and it could become too large.
  - We recommend you to investigate each kind of timeline for each purpose such as revealing file manipulation events and program execution events. Then we can concatenate the results of each timeline analysis to reveal details of incidents. It is not necessary to analyze all timelines at the same time.

- In this section, we will build timelines from filesystem metadata and registry entries.

Analyzing Filesystem Timelines

NTFS 101

- To examine Windows computers, we handle NTFS volumes.
- The filesystem has various kinds of metadata. The following metadata is useful for digital forensics.
  - $MFT
  - $UsnJrnl
  - $Logfile
  - $I30

NTFS 101: MFT (1)

- The Master File Table (MFT) contains information about all files and folders as its entries.
- Windows does not erase MFT entries immediately when files or folders are deleted. It just changes the "in-use" flag to zero in the MFT entry corresponding to the deleted file or folder. Later, when newer files or folders are created, Windows overwrites those MFT entries.
- Thus we can get information of deleted files or folders from those MFT entries.

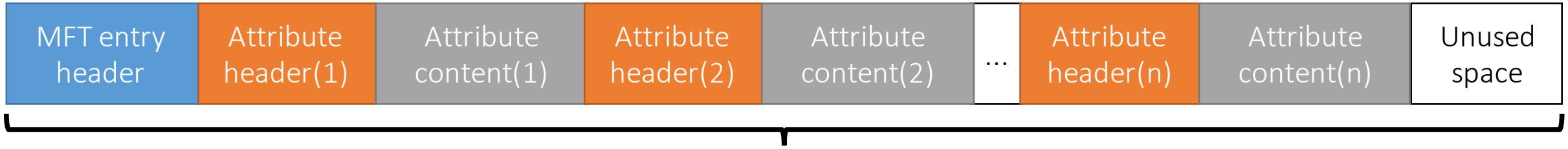
NTFS 101: MFT (2)

- Each MFT entry has a MFT entry header and multiple "attributes".
- MFT entry headers start with the ASCII signature "FILE". The signature typically becomes "BAAD" when the entry is corrupt.
- Each MFT entry contains several pieces of important information such as flags ("in-use" and "folder"), link count, MFT record ID and so on.

NTFS 101: MFT (3)

- All MFT entries are exactly 1024 bytes or 4096 bytes in size.
- It depends on the physical sector size of the disk storage.

Physical Sector Size	Length of Each MFT Entry
512 bytes	1024 bytes
4096 bytes	4096 bytes



NTFS 101: MFT Attributes (0)

- MACB times are essential information for timeline analysis.
- It means:
  - the last Modification time
  - the last Access time
  - the last Change time (the last modified time of the related MFT entry)
  - the Birth time (the creation time)

NTFS 101: MFT Attributes (1)

- There are many standard attribute types. We often focus on the following four attributes for forensic investigation.
  - $STANDARD_INFORMATION ($SI)
  - $FILE_NAME ($FN)
  - $DATA
  - $EA (Extended Attributes)

NTFS 101: MFT Attributes (2)

- $STANDARD_INFORMATION ($SI)
  - It contains general information such as MACB times, the owner and its security ID. These timestamps can be changed by Windows APIs.

- $FILE_NAME ($FN)
  - It contains a file name in Unicode (UTF-16LE). And it also contains timestamps like $SI. Timestamps in this attribute can not be changed by user-mode Windows APIs.
  - On timestamp manipulation, some attackers modify only $SI attributes. So we can find those timestamp manipulations by examining the differences between $FN and $SI timestamps.

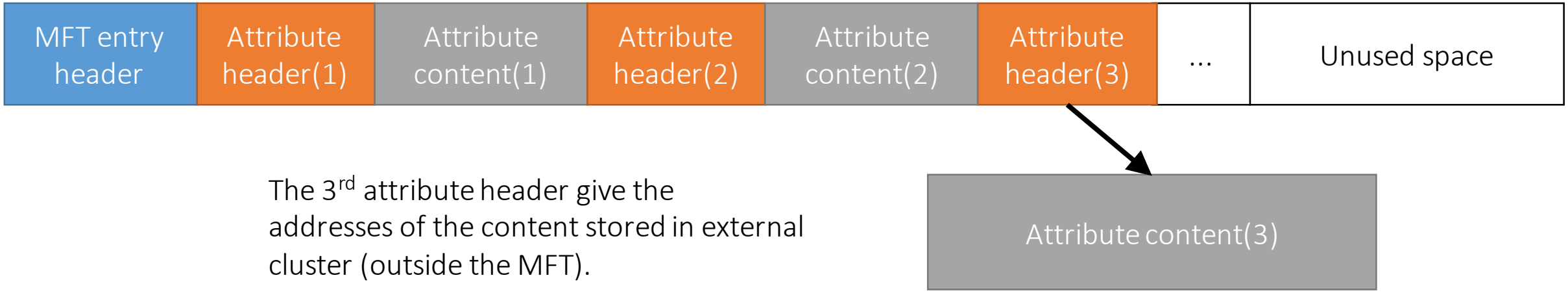
NTFS 101: MFT Attributes (3)

- $DATA
  - It is a file content. Each file can have multiple $DATA attributes. When a file has more than one $DATA attributes, the each additional one must have unique name, and those are called "alternative data stream" (ADS).
  - For example, Windows uses ADS for "Zone.Identifier". It indicates that the file was downloaded from the internet.

- $EA (Extended Attributes)
  - It was designed for backward compatibility with OS/2 applications.
  - Variants of the Trojan Zeroaccess uses $EA for storing malicious payload.
  - Windows 8 or later also use this attribute on many system binaries as a part of the secure boot components.

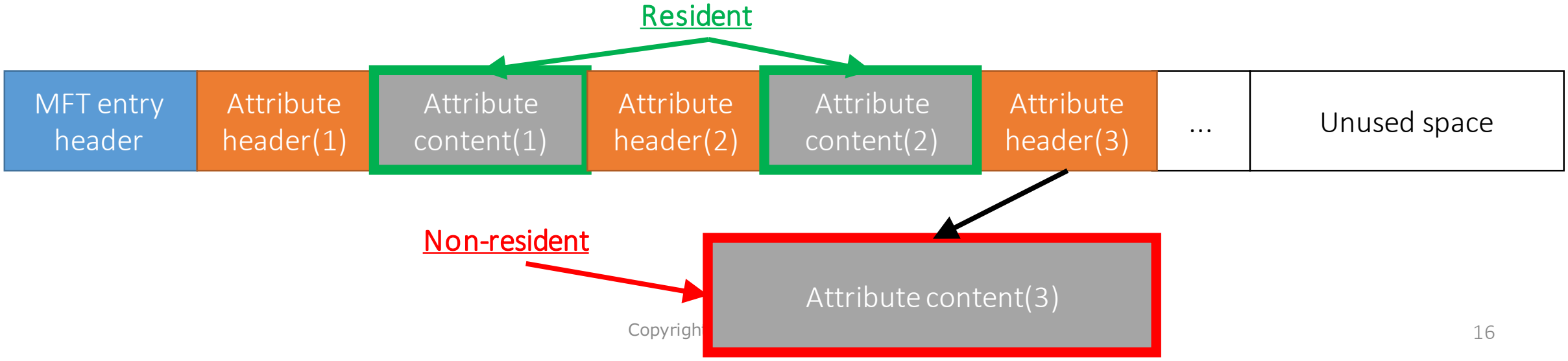
NTFS 101: MFT Attributes (4)

- Since the length of each MFT entry is fixed, attribute contents are stored in external cluster in the filesystem when they are too large to be stored in a MFT entry.
- For example, If the length of each MFT entry is fixed to 1024 bytes, a $DATA attribute over about 700 bytes is stored in external cluster.



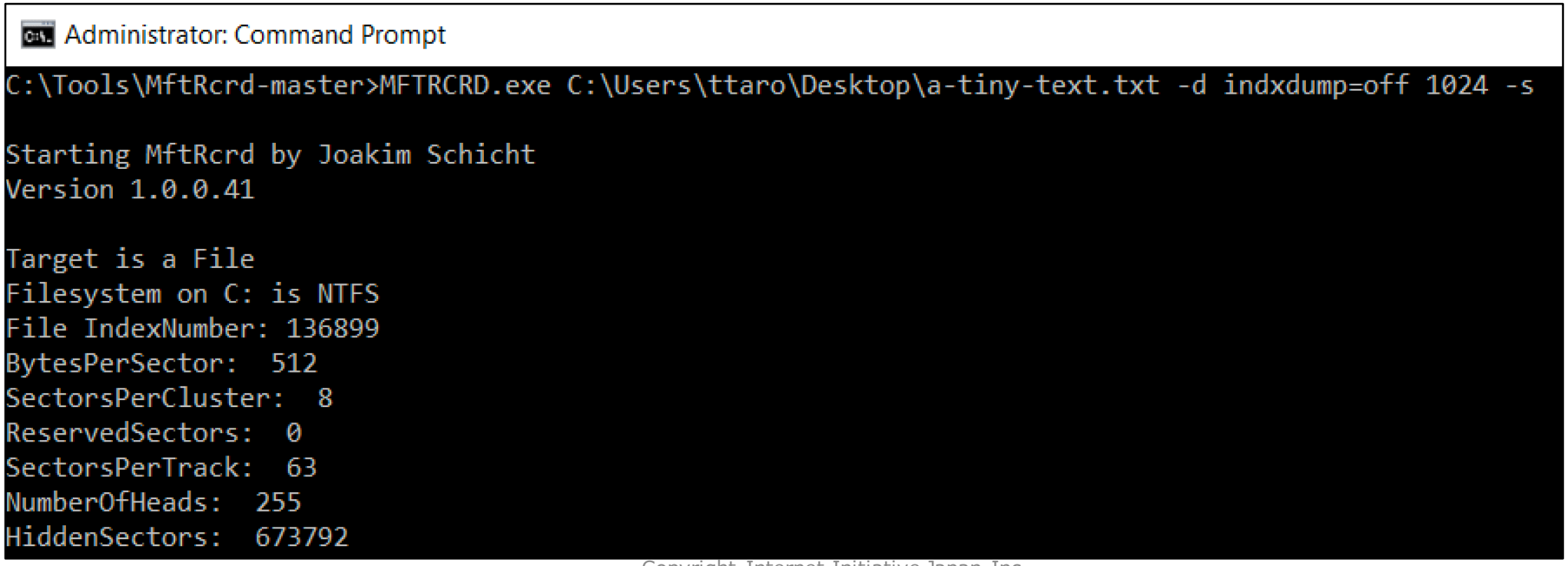
NTFS 101: MFT Attributes (5)

- When an attribute content is stored in external cluster, the attribute is called the "non-resident" attribute.
- On the other hand, when a content is small enough, a whole content is stored in a MFT entry. It is called the "resident" attribute.
- So, we can restore a deleted file from a MFT entry, if the $DATA attribute was the "resident" attribute.



MFT Parsing Tools (1)

- MFTRCRD [1]
  - MFTRCRD is a command line MFT record decoder for online filesystem.
  - You can use it to know the information contained in MFT easily.



NTFS 101: Demo (1)

NTFS 101: Demo (2)

NTFS 101: Demo (3)

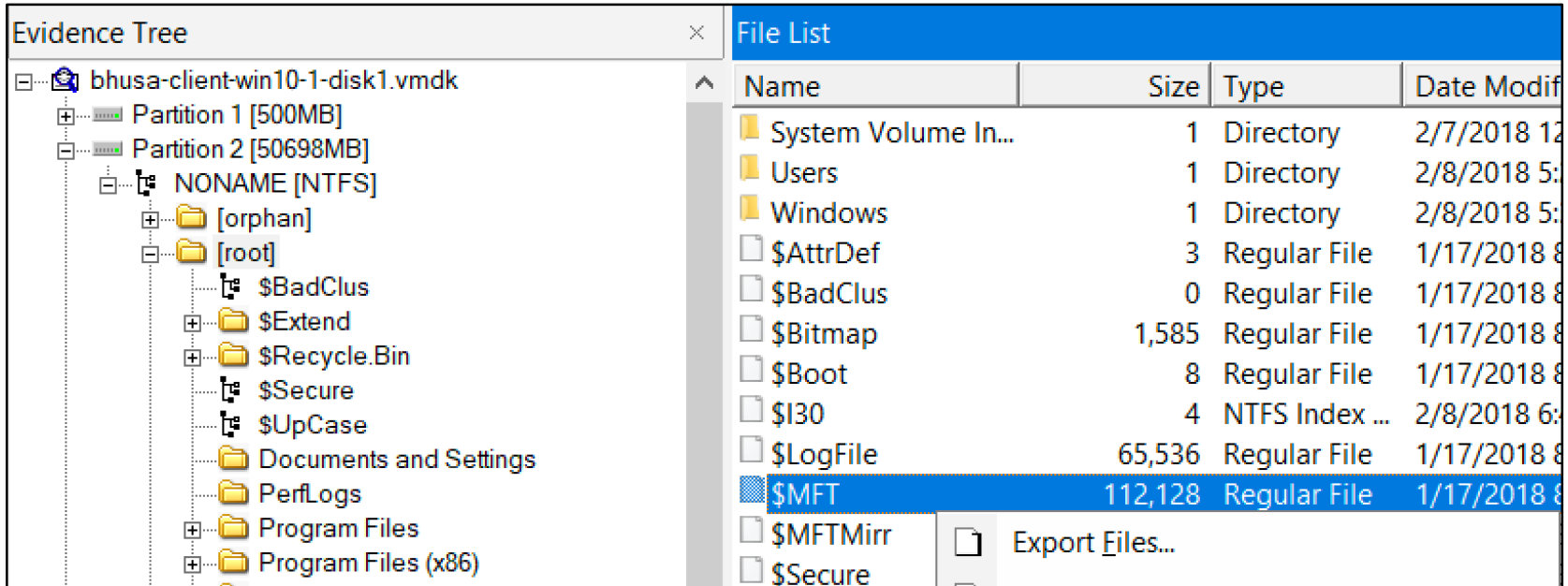
NTFS 101: Demo (4)

NTFS 101: Demo (5)

NTFS 101: Demo (6)

NTFS 101: MFT (4)

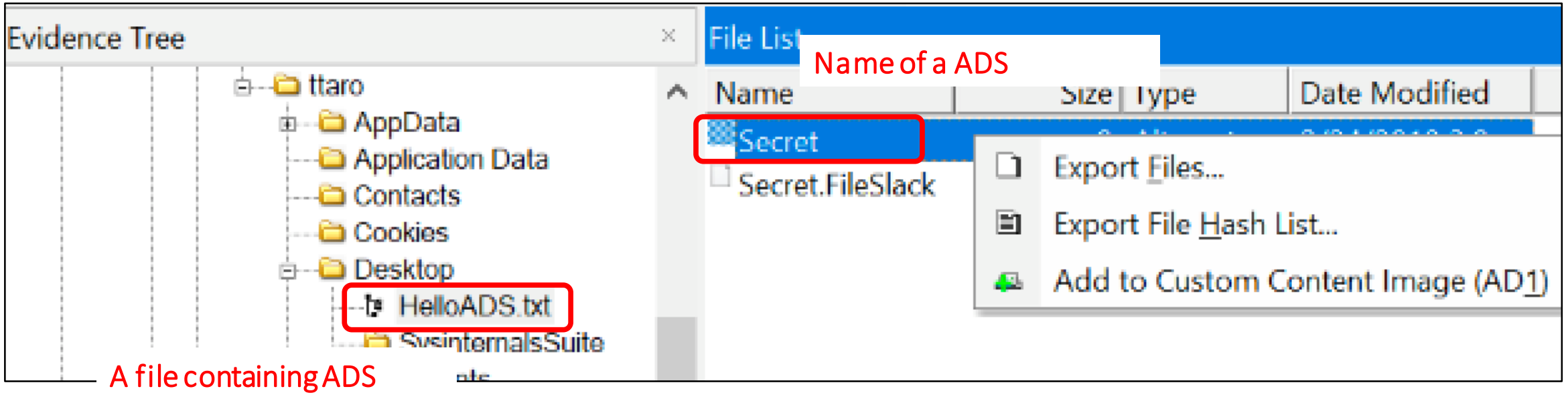
- In real case, you should examine MFT of acquired disk images instead of online volumes.
- MFT is stored in the root of the NTFS volume as a file named "$MFT".
- We can extract the file from disk images by using several image mounting/parsing tools.



In this case, we used the FTK Imager.

NTFS 101: MFT (5)

- We can also confirm existence of ADS and extract it easily by using file parsing/mounting tools such as FTK imager.



MFT Parsing Tools (2)

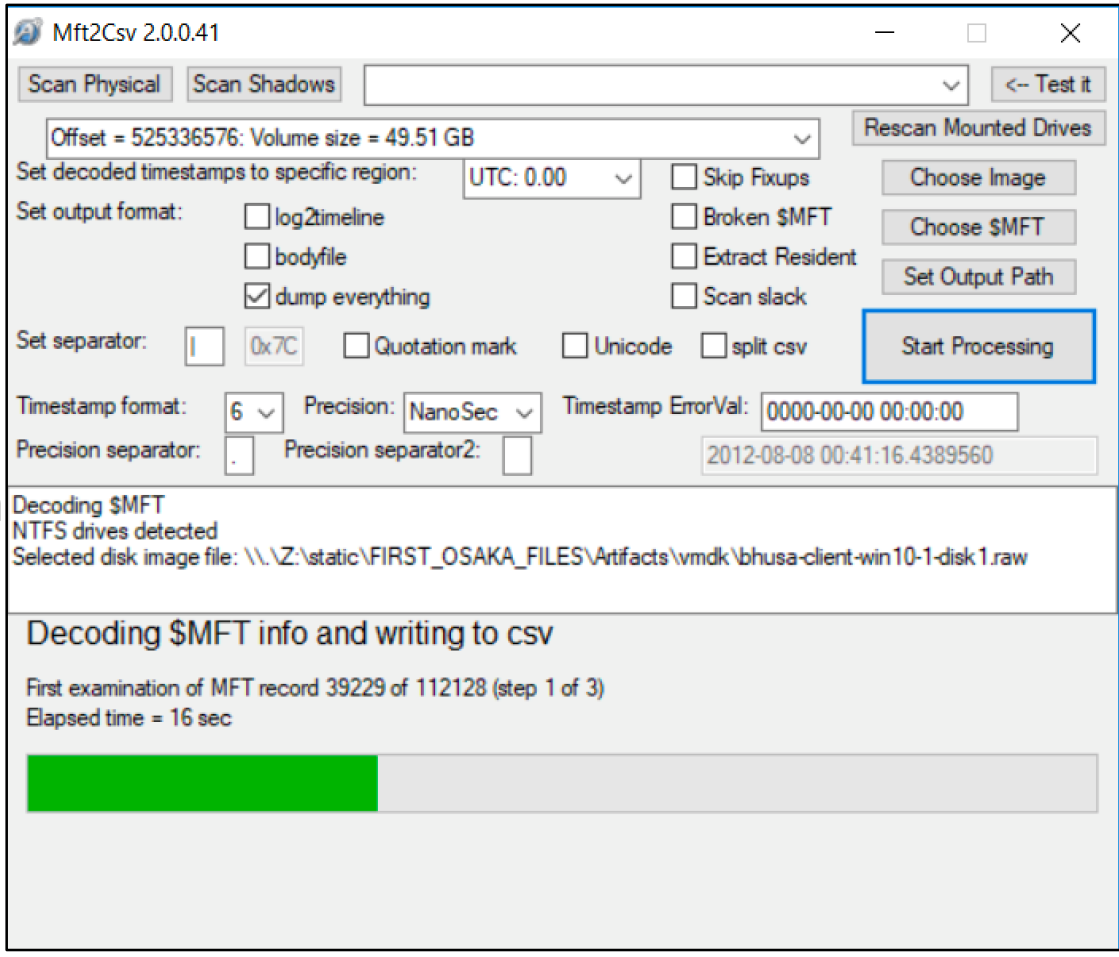
- analyzeMFT.py [2]
  - analyzeMFT.py can parse $MFT files and summarize those information.
  - It shows information contained in MFT entry header, $SI, $FN and, $DATA attributes. It also shows other standard attributes such as $EA exist or not.
  - If a MFT entry has more than one $FN attribute, analyzeMFT.py will parse all of those.
  - It is also capable of anomaly detection for $SI and $FN timestamps.

C:\Windows\system32\cmd.exe - more analyzed-mft-win10-1-full.csv

```
"Record Number","Good","Active","Record type","Sequence Number","Parent File Rec. #","Parent File Rec. Seq. #","Filename
 #1","Std Info Creation date","Std Info Modification date","Std Info Access date","Std Info Entry date","FN Info Creatio
n date","FN Info Modification date","FN Info Access date","FN Info Entry date","Object ID","Birth Volume ID","Birth Obje
ct ID","Birth Domain ID","Filename #2","FN Info Creation date","FN Info Modify date","FN Info Access date","FN Info Entr
y date","Filename #3","FN Info Creation date","FN Info Modify date","FN Info Access date","FN Info Entry date","Filename
 #4","FN Info Creation date","FN Info Modify date","FN Info Access date","FN Info Entry date","Standard Information","At
tribute List","Filename","Object ID","Volume Name","Volume Info","Data","Index Root","Index Allocation","Bitmap","Repars
e Point","EA Information","EA","Property Set","Logged Utility Stream","Log/Notes","STF FN Shift","uSec Zero","ADS"
"0","Good","Active","File","1","5","5","/$MFT","=""2018-01-17 20:31:28.720869""","=""2018-01-17 20:31:28.720869""","=""2
018-01-17 20:31:28.720869""","=""2018-01-17 20:31:28.720869""","=""2018-01-17 20:31:28.720869""","=""2018-01-17 20:31:28
.720869""","=""2018-01-17 20:31:28.720869""","=""2018-01-17 20:31:28.720869""","","","","","","","","","","","","","",""
,"","","","","","True","False","True","False","False","False","False","False","False","True","False","False","False","Fa
lse","False","","N","N","N"
"1","Good","Active","File","1","5","5","/$MFTMirr","=""2018-01-17 20:31:28.720869""","=""2018-01-17 20:31:28.720869""","
=""2018-01-17 20:31:28.720869""","=""2018-01-17 20:31:28.720869""","=""2018-01-17 20:31:28.720869""","=""2018-01-17 20:3
```

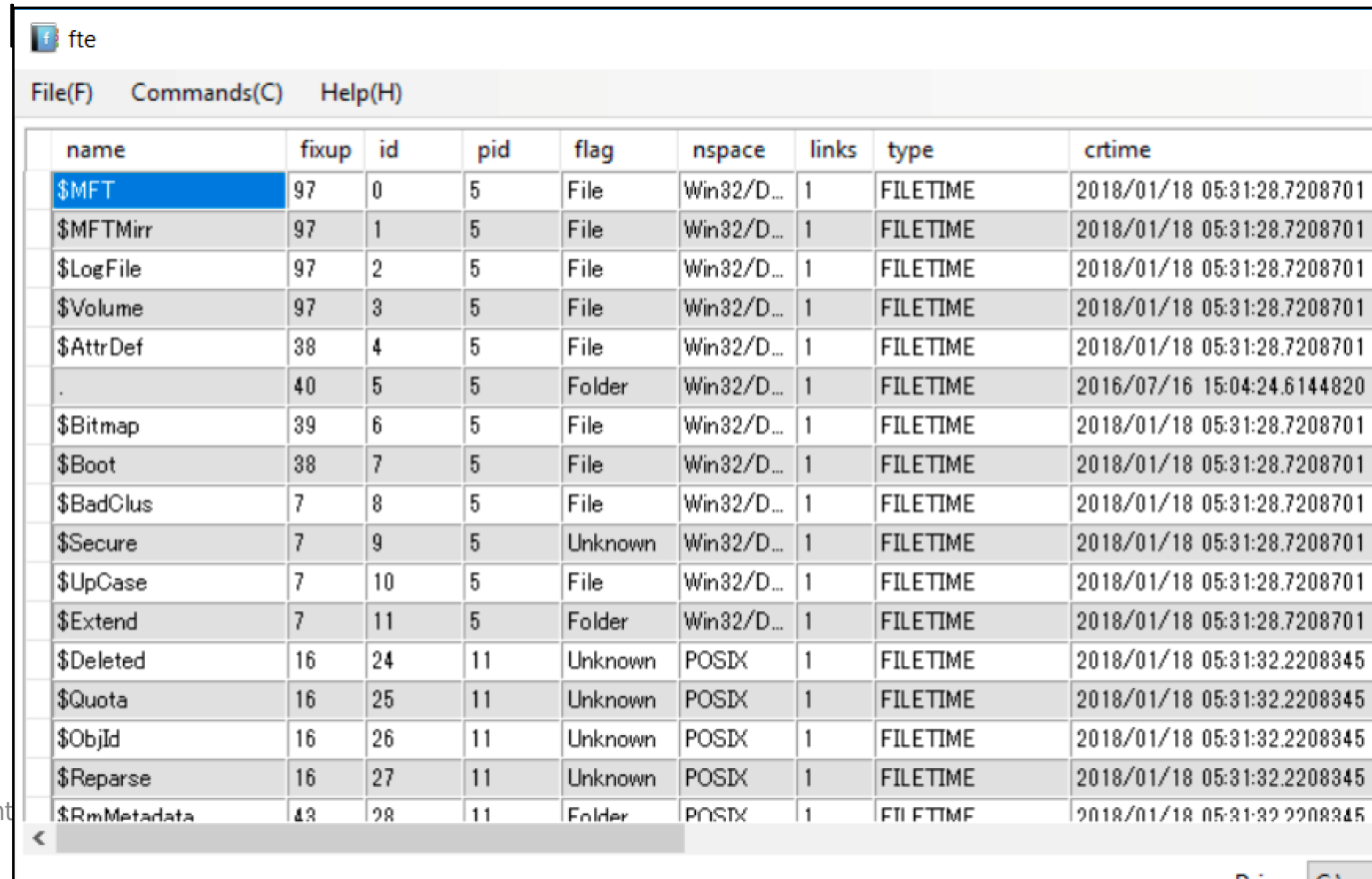
MFT Parsing Tools (3)

- Mft2Csv [3]
  - Mft2Csv can parse $MFT files and raw disk images. It can extract resident files from $MFT.
  - Its output contains details of several types of information such as $EA entries. That is useful to seek suspicious contents stored in $EA.
  - Mft2Csv can also parse recovered $MFT entries by MftCarver.



MFT Parsing Tools (4)

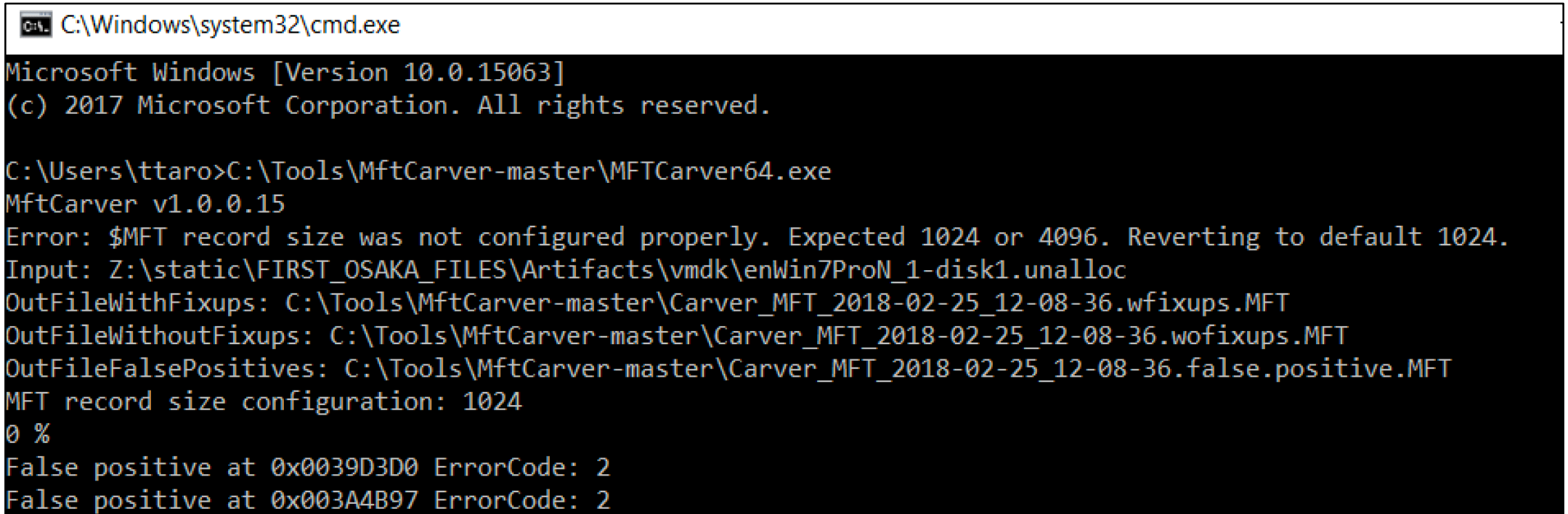
- fte (FILETIME Extractor) [4]
  - fte can parse not only $MFT but also INDX attributes which we will mention later.
  - It has simple GUI viewer.



name	fixup	id	pid	flag	nspace	links	type	crtime
$MFT	97	0	5	File	Win32/D...	1	FILETIME	2018/01/18 05:31:28.7208701
$MFTMirr	97	1	5	File	Win32/D...	1	FILETIME	2018/01/18 05:31:28.7208701
$LogFile	97	2	5	File	Win32/D...	1	FILETIME	2018/01/18 05:31:28.7208701
$Volume	97	3	5	File	Win32/D...	1	FILETIME	2018/01/18 05:31:28.7208701
$AttrDef	38	4	5	File	Win32/D...	1	FILETIME	2018/01/18 05:31:28.7208701
.	40	5	5	Folder	Win32/D...	1	FILETIME	2016/07/16 15:04:24.6144820
$Bitmap	39	6	5	File	Win32/D...	1	FILETIME	2018/01/18 05:31:28.7208701
$Boot	38	7	5	File	Win32/D...	1	FILETIME	2018/01/18 05:31:28.7208701
$BadClus	7	8	5	File	Win32/D...	1	FILETIME	2018/01/18 05:31:28.7208701
$Secure	7	9	5	Unknown	Win32/D...	1	FILETIME	2018/01/18 05:31:28.7208701
$UpCase	7	10	5	File	Win32/D...	1	FILETIME	2018/01/18 05:31:28.7208701
$Extend	7	11	5	Folder	Win32/D...	1	FILETIME	2018/01/18 05:31:28.7208701
$Deleted	16	24	11	Unknown	POSIX	1	FILETIME	2018/01/18 05:31:32.2208345
$Quota	16	25	11	Unknown	POSIX	1	FILETIME	2018/01/18 05:31:32.2208345
$ObjId	16	26	11	Unknown	POSIX	1	FILETIME	2018/01/18 05:31:32.2208345
$Reparse	16	27	11	Unknown	POSIX	1	FILETIME	2018/01/18 05:31:32.2208345
$RmMetadata	43	28	11	Folder	POSIX	1	FILETIME	2018/01/18 05:31:32.2208345

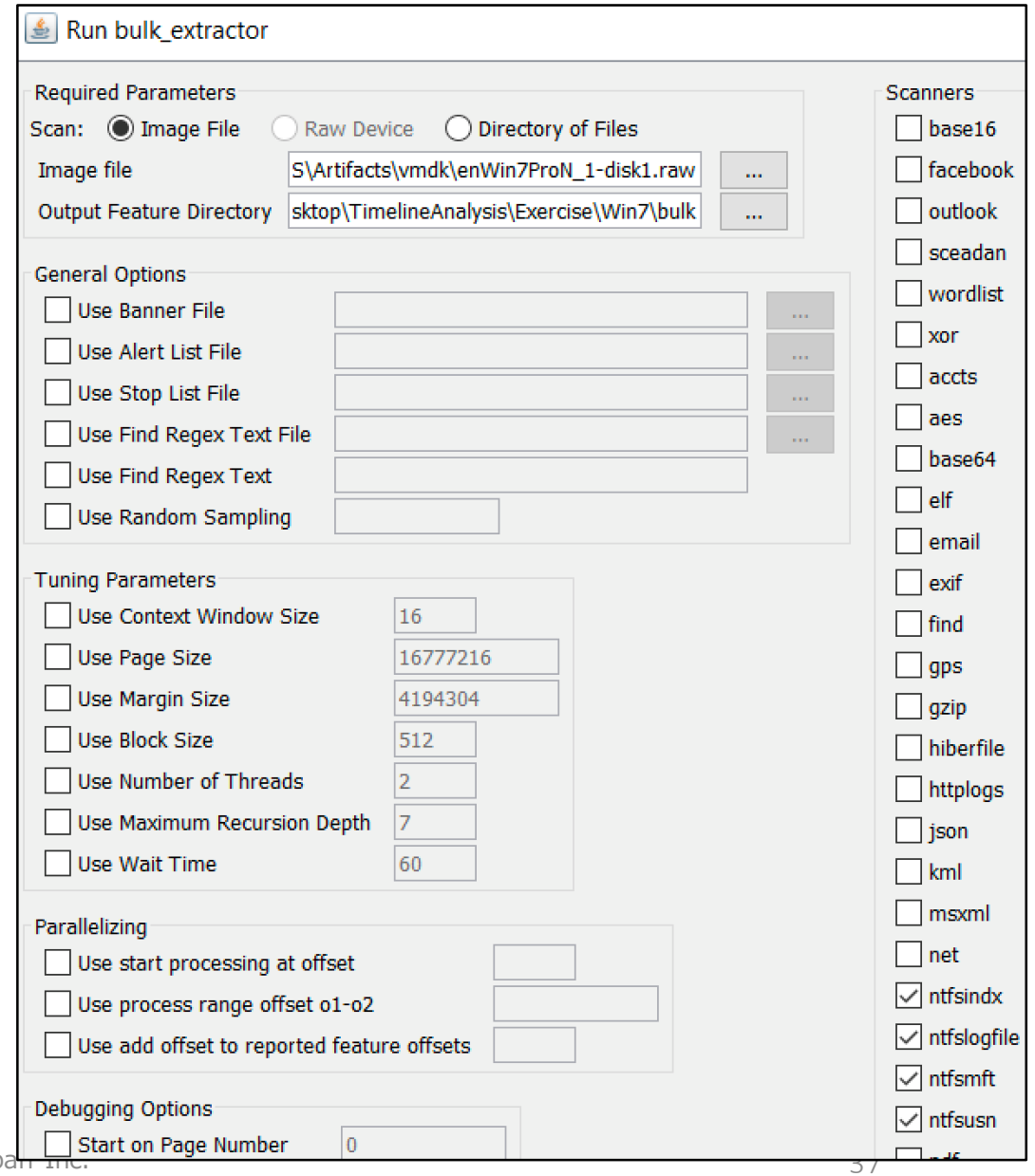
MFT Carving Tools (1)

- MftCarver [5]
  - This tool dumps individual MFT entries. It can scan unallocated spaces, file slacks, memory dumps, and so on.
  - We can recover old MFT entries that are not listed in current MFT.



MFT Carving Tools (2)

- Bulk Extractor with Record Carving [6]
  - It is an enhanced version from original Bulk Extractor. It contains scanner plug-ins for records of $MFT, $LogFile, $UsnJrnl:$J, $INDEX_ALLOCATION, and utmp structure.
  - It can recover those records from disk images.



Timeline Analysis Exercise 1-1 (1)

View MFT entries and examine a certain folder

- The purpose of this exercise:
  - By parsing $MFT, list up the files and folders which placed in the desktop folder of user "ttaro".
- We parsed a $MFT of ttaro's disk with analyzeMFT.py. And the following is the result. Let's open it with CSVFileView.
  - "Training_Materials\TimelineAnalysis\Win7\analyzeMft-output.csv".
- The CSVFileView binaries are placed in the following paths.
  - Training_Materials\Tools\csvfileview-x64\CSVFileView.exe (for 64 bit)
  - Training_Materials\Tools\csvfileview\CSVFileView.exe (for 32 bit)
- The below is command line sample for analyzeMft.py

```
analyzeMFT.py -f artifact\$MFT -a -e -o analyzeMft-output.csv
```

Timeline Analysis Exercise 1-1 (2)

View MFT entries and examine a certain folder

- An output of analyzeMFT.py has 54 columns. Most of those are same as we viewed in demo 0. But follows are useful columns added by analyzeMFT.py.
  - STF FN Shift
    - If "Y" (means YES), the $FN creation time is after the $SI creation time. It implies that the timestamps in $SI could have been manipulated.
  - uSec Zero
    - If "Y", the micro second (uSec) value of $SI creation time is zero. It also implies that the timestamp could have been manipulated.
  - ADS
    - If "Y", the MFT entry contains alternative data stream (ADS).
  - EA
    - If "Y", the MFT entry contains $EA attribute.

Timeline Analysis Exercise 1-1 (3)

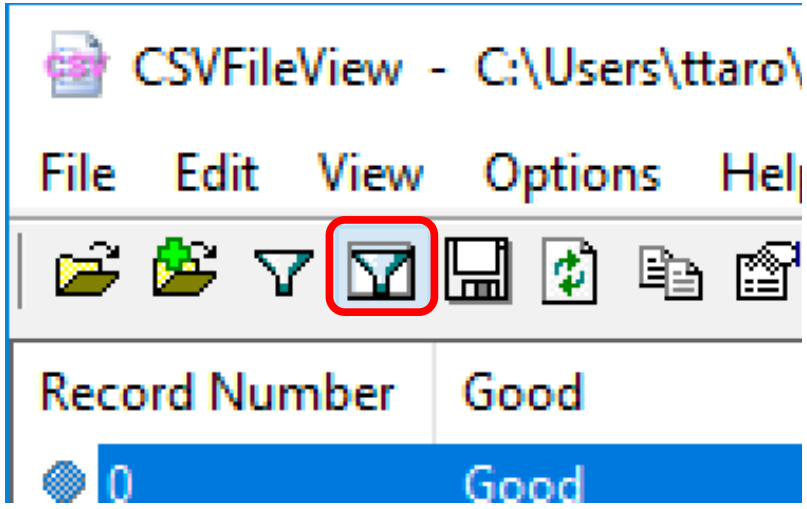
View MFT entries and examine a certain folder

- Notice:
  - Original zip format contains the creation time of files on the 10 millisecond scale. And it fills the values under 10 millisecond with zero when extraction of that files. So old zip archives can trigger false positives by this uSec Zero detection.
  - But these days, ordinary zip archivers use extra fields to support higher-resolution timestamps. So we hardly face those false positives.

Timeline Analysis Exercise 1-1 (4)

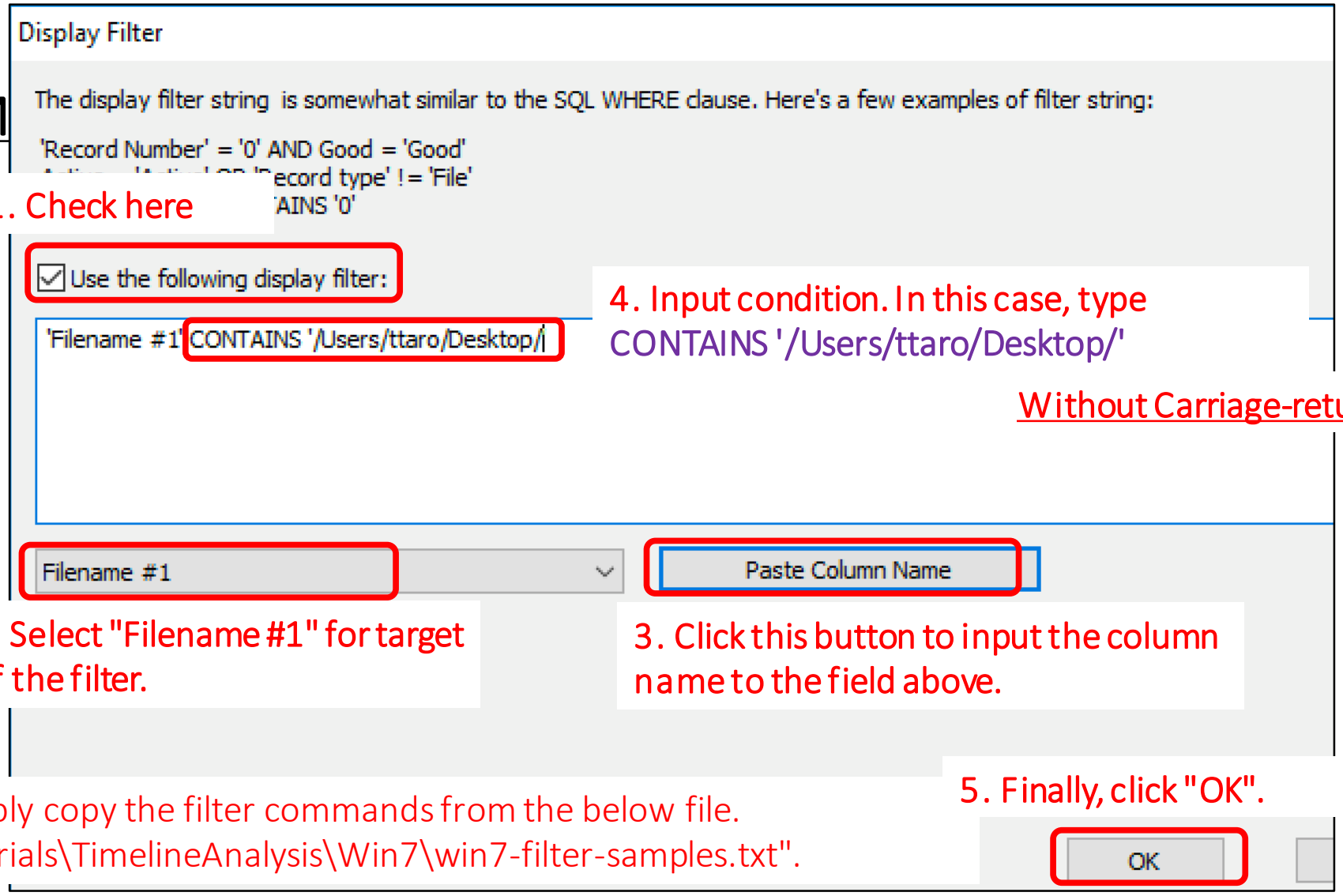
View MFT entries and examine a certain folder

- Let's apply the filter to list files and folders placed in user ttaro's desktop.
- First, click the "Edit Display Filter" button.



Timeline Analysis Exercise 1-1 (5)

View M



1. Check here

2. Select "Filename #1" for target of the filter.

3. Click this button to input the column name to the field above.

4. Input condition. In this case, type CONTAINS '/Users/ttaro/Desktop/'

Without Carriage-return

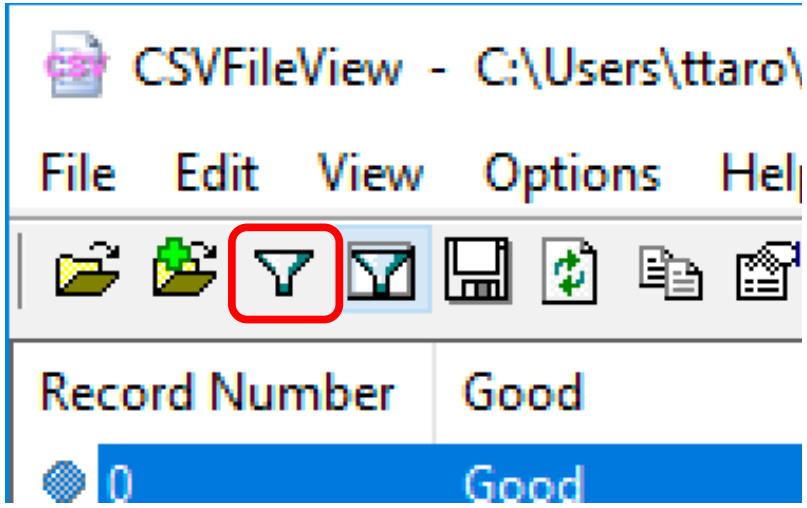
5. Finally, click "OK".

Or, you can simply copy the filter commands from the below file.
"Training_Materials\TimelineAnalysis\Win7\win7-filter-samples.txt".

Timeline Analysis Exercise 1-1 (6)

View MFT entries and list files placed in ttaro's Desktop folder

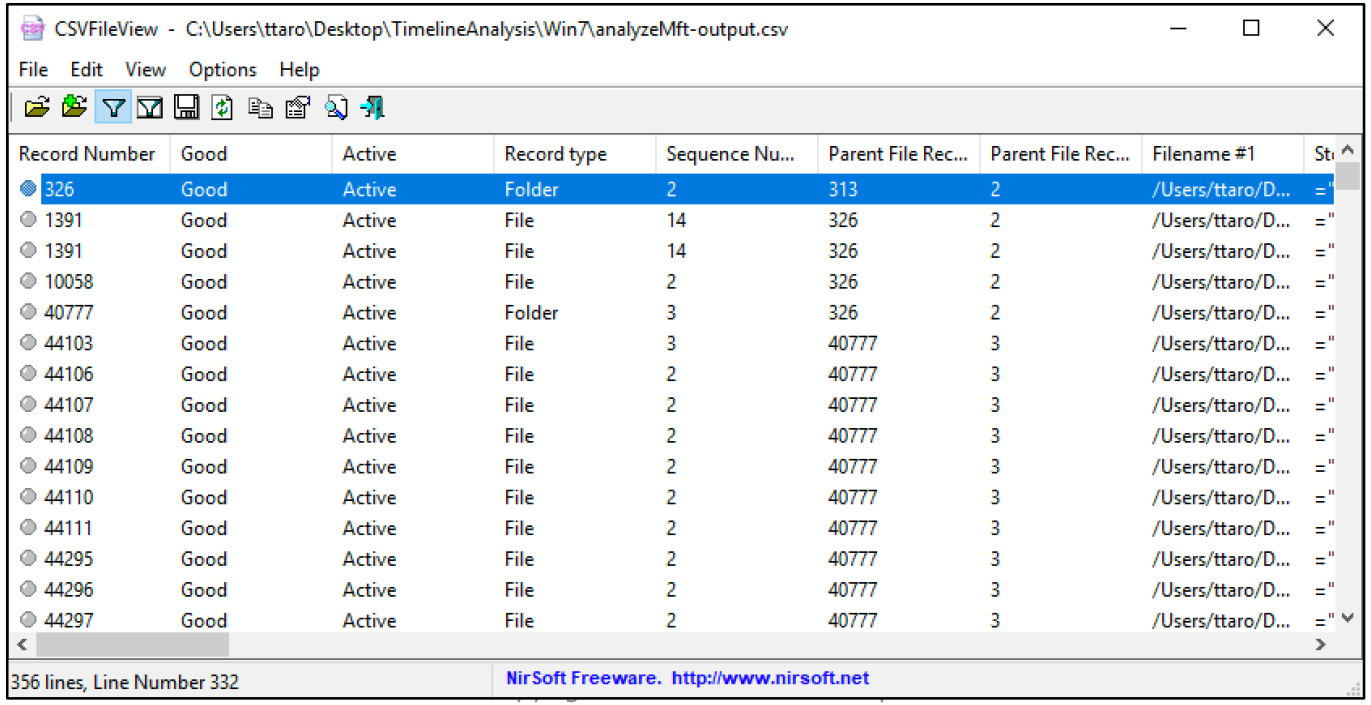
- To apply the filter, activate the "Use Display Filter" button.



Timeline Analysis Exercise 1-1 (7)

View MFT entries and examine a certain folder

- After applying the filter, you can confirm 355 entries contained in ttaro's desktop.
- These include deleted files which do not exist in the NTFS volume.

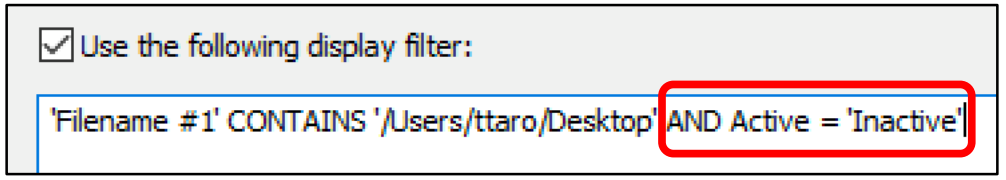


Record Number	Good	Active	Record type	Sequence Nu...	Parent File Rec...	Parent File Rec...	Filename #1	St
326	Good	Active	Folder	2	313	2	/Users/ttaro/D...	="
1391	Good	Active	File	14	326	2	/Users/ttaro/D...	="
1391	Good	Active	File	14	326	2	/Users/ttaro/D...	="
10058	Good	Active	File	2	326	2	/Users/ttaro/D...	="
40777	Good	Active	Folder	3	326	2	/Users/ttaro/D...	="
44103	Good	Active	File	3	40777	3	/Users/ttaro/D...	="
44106	Good	Active	File	2	40777	3	/Users/ttaro/D...	="
44107	Good	Active	File	2	40777	3	/Users/ttaro/D...	="
44108	Good	Active	File	2	40777	3	/Users/ttaro/D...	="
44109	Good	Active	File	2	40777	3	/Users/ttaro/D...	="
44110	Good	Active	File	2	40777	3	/Users/ttaro/D...	="
44111	Good	Active	File	2	40777	3	/Users/ttaro/D...	="
44295	Good	Active	File	2	40777	3	/Users/ttaro/D...	="
44296	Good	Active	File	2	40777	3	/Users/ttaro/D...	="
44297	Good	Active	File	2	40777	3	/Users/ttaro/D...	="

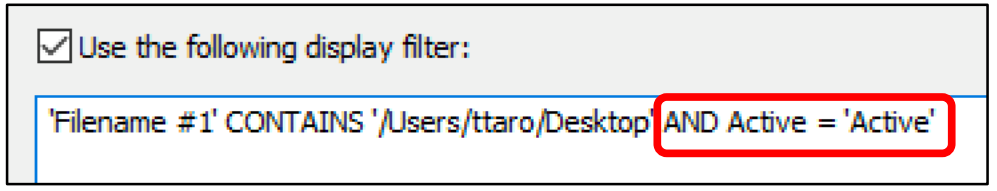
Timeline Analysis Exercise 1-1 (8)

View MFT entries and examine a certain folder

- "Active" column indicate the file exist or not. It is result of parsing in-use flag in a MFT entry header.
- So, we can focus on the deleted files by applying the filter to display rows with the column as "Inactive". Then we can confirm 202 deleted files and folders on ttaro's desktop.



- We can also confirm 153 files and folders which exist in the folder by applying the filter to display rows with the column as "Active".



Timeline Analysis Exercise 1-2 (1)

Find suspicious timestamps

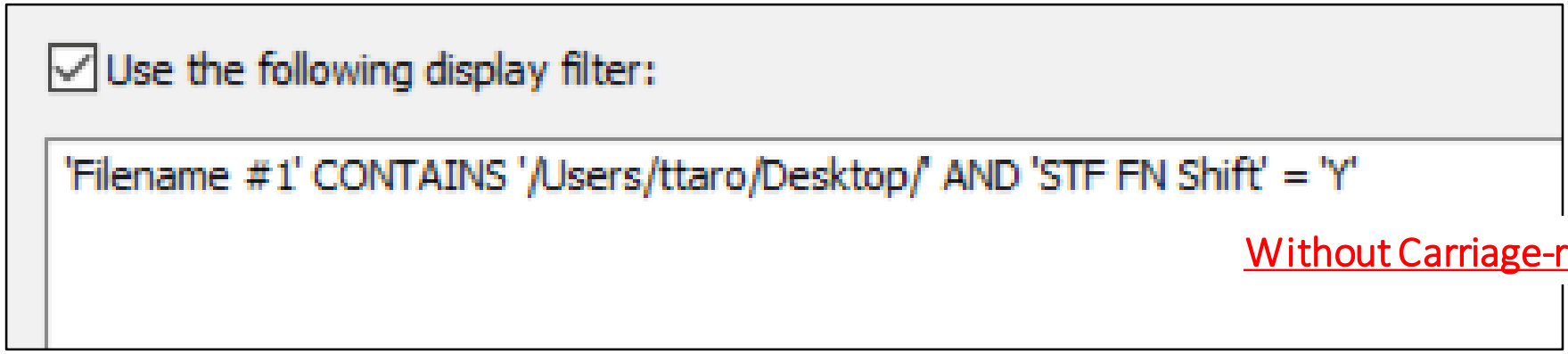
- Find the files which are suspicious of timestamp manipulation in ttaro's Desktop by checking "STF FN Shift" and "uSec Zero" columns.

- Notice: ZIP archivers manipulate $SI timestamps for the purpose of recovering original timestamps.

Timeline Analysis Exercise 1-2 (2)

Find suspicious timestamps

- Let's use display filter to check "STF FN Shift" column.



You can simply copy the filter commands from the file "Training_Materials\TimelineAnalysis\Win7\win7-filter-samples.txt".

Timeline Analysis Exercise 1-2 (3)

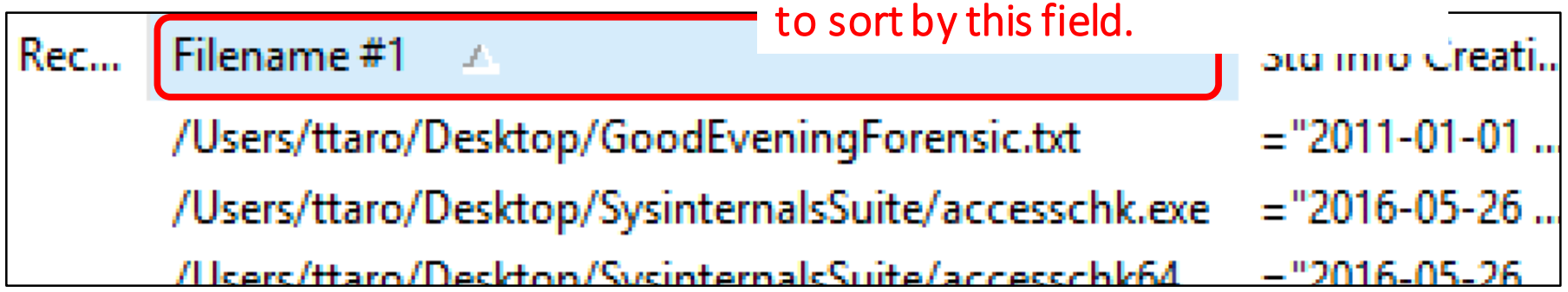
Find suspicious timestamps

- You can confirm 148 entries with applying the filter to display rows with "STF FN Shift" column as 'Y'. 147 of those are placed under the "SysinternalsSuite" folder.
- It indicates the possibility that the folder was extracted from a archive file such as zip. And SysinternalsSuite is a famous windows utility package which is distributed as a zip archive file.

Timeline Analysis Exercise 1-2 (4)

Find suspicious timestamps

- Thus, we should focus on the last one file which is not contained SysinternalsSuite folder first. Its name is "GoodEveningForensic.txt"



Timeline Analysis Exercise 1-2 (5)

Find suspicious timestamps

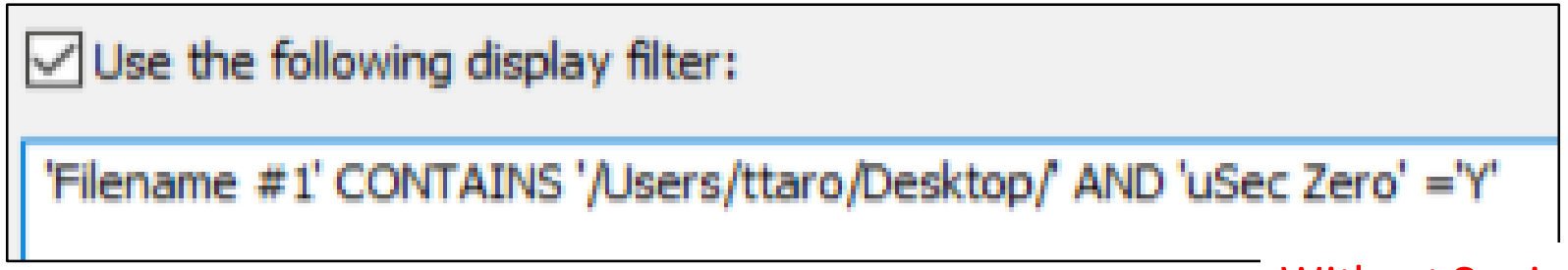
- You can see that the $SI timestamps are before $FN timestamps. It implies that someone probably manipulated the file's timestamps after creating the file.

Filename #1	Std Info Creation date	S	S	S	FN Info Creation date
/Users/ttaro/Desktop/GoodEveningForensic.txt	="2011-01-01 06:00:01"	=	=	=	="2018-02-24 14:58:24.282444"

Timeline Analysis Exercise 1-2 (6)

Find suspicious timestamps

- Next, let's use display filter to check uSec Zero column.



Without Carriage-return

You can simply copy the filter commands from the file "Training_Materials\TimelineAnalysis\Win7\win7-filter-samples.txt".

Timeline Analysis Exercise 1-2 (7)

Find suspicious timestamps

- You can confirm one entry by applying filter to display rows with "uSec Zero" column as 'Y'. The file is "GoodNightForensic.txt".
- All timestamps contained in the entry are same. But analyzeMFT.py detected that micro second values of those are zero.

Filename #1	Std Info Creation date	S	S	S	FN Info Creation date
/Users/ttaro/Desktop/GoodNightForensic.txt	="2001-01-01 12:00:00"	=	=	=	="2001-01-01 12:00:00"

Timeline Analysis Exercise 1-2 (8)

Find suspicious timestamps

- It is not natural. It implies that someone probably manipulated the file's timestamps.
- For example, "SetMACE.exe" can manipulate all timestamps contained in both of $SI and $FN.
- If attackers set those timestamps with non-zero micro second values, it becomes more difficult to detect.

Timeline Analysis Exercise 1-3 (1)

Recover files from $MFT

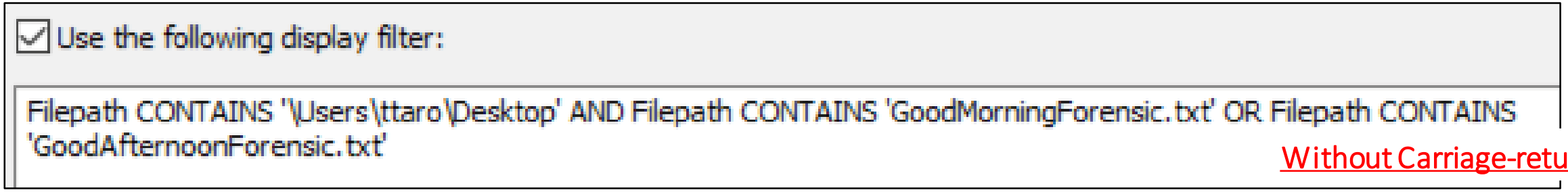
- There are two entries for deleted files which were in ttaro's desktop.
- The names of the files are:
  - GoodMorningForensic.txt
  - GoodAfternoonForensic.txt

- Can we recover these contents?
  - File carving may recover them. But it consumes a long time.
  - If contents of those files are stored in MFT entries (in other words, those $DATA attributes were "resident"), we can recover them from the $MFT!

Timeline Analysis Exercise 1-3 (2)

Recover files from $MFT

- Let's check the resident flags for $DATA attribute in the target files.
- We executed Mft2Csv for the same $MFT as before. And the parsed list of MFT entries are saved as the file below. Let's open it with CSVFileView!
  - "TimelineAnalysis\Win7\Mft2Csv-output\Mft_2018-02-25_00-18-39.csv"
- Then, apply the filter to display target files.



You can simply copy the filter commands from the file "Training_Materials\TimelineAnalysis\Win7\win7-filter-samples.txt".

Timeline Analysis Exercise 1-3 (3)

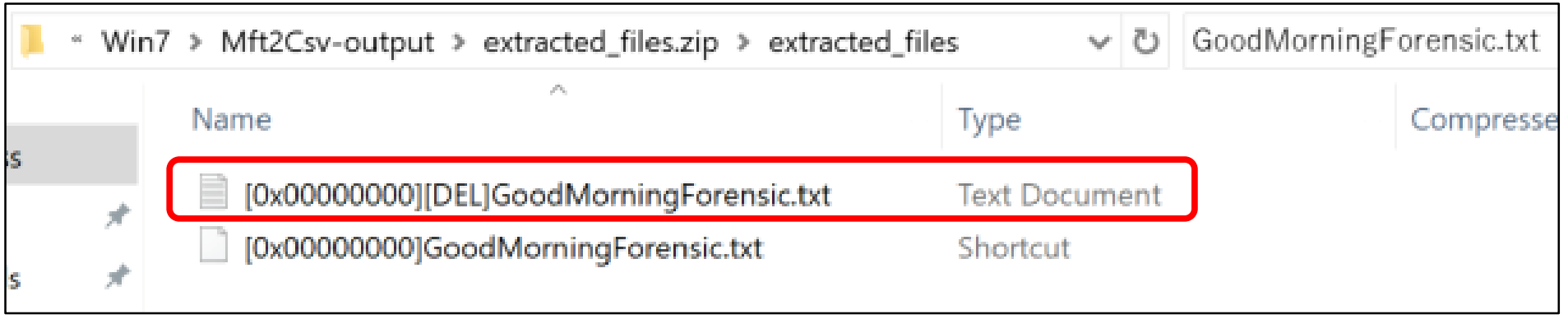
Recover files from $MFT

- A MFT entry list of Mft2Csv output has 124 columns and those are similar to result of analyzeMFT.py, but the column names are different.
- Mft2Csv has some additional columns such as "DATA_NonResidentFlag". If its value is "0", it means resident. In other words, a $DATA content of the entry can be extracted from $MFT.

Timeline Analysis Exercise 1-3 (4)

Recover files from $MFT

- When Mft2Csv is executed with "Extract Resident" option, it saves the recovered files in the output folder. And Mft2Csv set "[0x00000000]" as prefix of recovered files' file names.
- Since the number of extracted files is over tens of thousands, we archived those files to extracted_files.zip.
- So you should browse names of files that are contained in the archive file without extracting those all files (since it takes long time). If you can find the file you look for, extract only the target file.



Timeline Analysis Exercise 1-4 (1)

Find suspicious $EA attributes

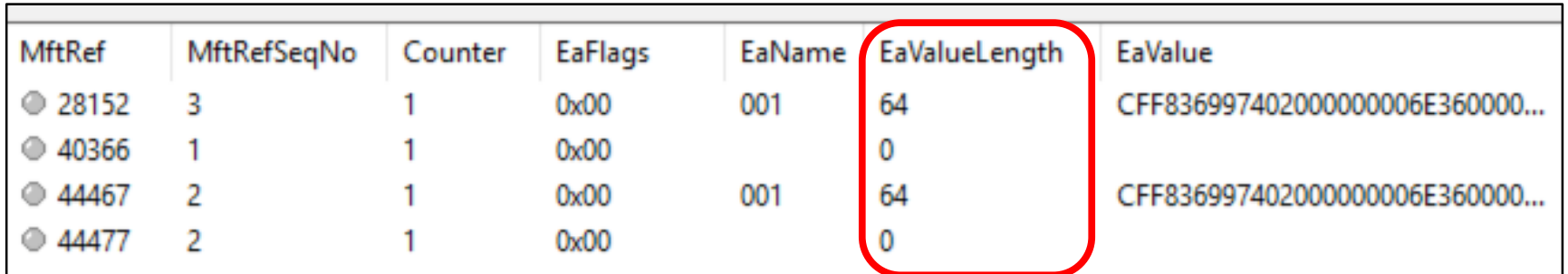
- $EA attribute is not used regularly. So it is unnatural that files contain $EA attribute.
- When the $EA attribute is non-resident, it become more suspicious. Because malware usually needs least tens of kilo bytes in size. Non-resident $EA attribute can have enough size to hide malware.

- Let's find files which has non-resident $EA attributes.

Timeline Analysis Exercise 1-4 (2)

Find suspicious $EA attributes

- "Mft-Ea-Entries_2018-02-25_00-18-39.csv" in Mft2Csv output folder is list of MFT entries which contain $EA attributes.
- But separator of the header line in the original file is incorrect. So use modified one. The name of the modified version is "Mft-Ea-Entries_2018-02-25_00-18-39_mod.csv ". Let's open this!
- This list has the "EaValueLength" column. When its value is "0", it means that the content of the $EA attribute is "non-resident". In other words, those $EA attributes have data over hundreds of bytes.



MftRef	MftRefSeqNo	Counter	EaFlags	EaName	EaValueLength	EaValue
28152	3	1	0x00	001	64	CFF836997402000000006E360000...
40366	1	1	0x00		0	
44467	2	1	0x00	001	64	CFF836997402000000006E360000...
44477	2	1	0x00		0	

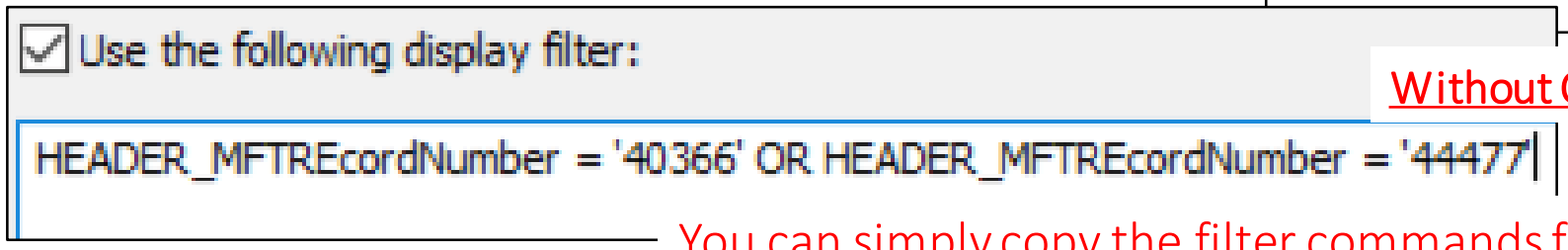
Timeline Analysis Exercise 1-4 (3)

Find suspicious $EA attributes

- You can find the names of files which contain non-resident $EA attribute by matching "MftRef" column in the "Mft-Ea-Entries_2018-02-25_00-18-39.csv" with "HEADER_MFTREcordNumber" column in the "Mft_2018-02-25_00-18-39.csv".

HEADER_MFTREcordNumber	FilePath
40366	:\Windows\CSC\v2.0.6
44477	:\Windows\System32\services.exe

MftRef	MftRefSeqNo	Counter	EaFlags	EaName	EaValueLength
28152	3	1	0x00	001	64
40366	1	1	0x00		0
44467	2	1	0x00	001	64
44477	2	1	0x00		0



Without Carriage-return

You can simply copy the filter commands from the file "Training_Materials\TimelineAnalysis\Win7\win7-filter-samples.txt".

Timeline Analysis Exercise 1-4 (4)

- Some variants of a Trojan Zeroaccess are known for hiding malicious payload in $EA attribute of "\Windows\System32\services.exe".
- If it was a real case, you should extract the content of $EA by using disk image parsing tool such as the sleuthkit. And check it.

- A folder named "\Windows\CSC\v2.0.6" also has non-resident $EA attribute. From the path, the folder seems to be related to client-side caching feature. Since we confirmed that the attribute was set on several freshly installed Windows environments, it can be said that we can regard it as benign.

Timeline Analysis Exercise 1-4 (5)

- Notice:
  - Microsoft uses $EA attribute of system binaries for secure booting. Then, thousands of system binaries have $EA attribute in Windows 8 or later.
  - But, all $EA attributes of those system binaries are "resident". They contain short string value less than about 100 bytes.
  - Thus files containing non-resident $EA attributes should be considered to be suspicious.

NTFS 101: $Logfile

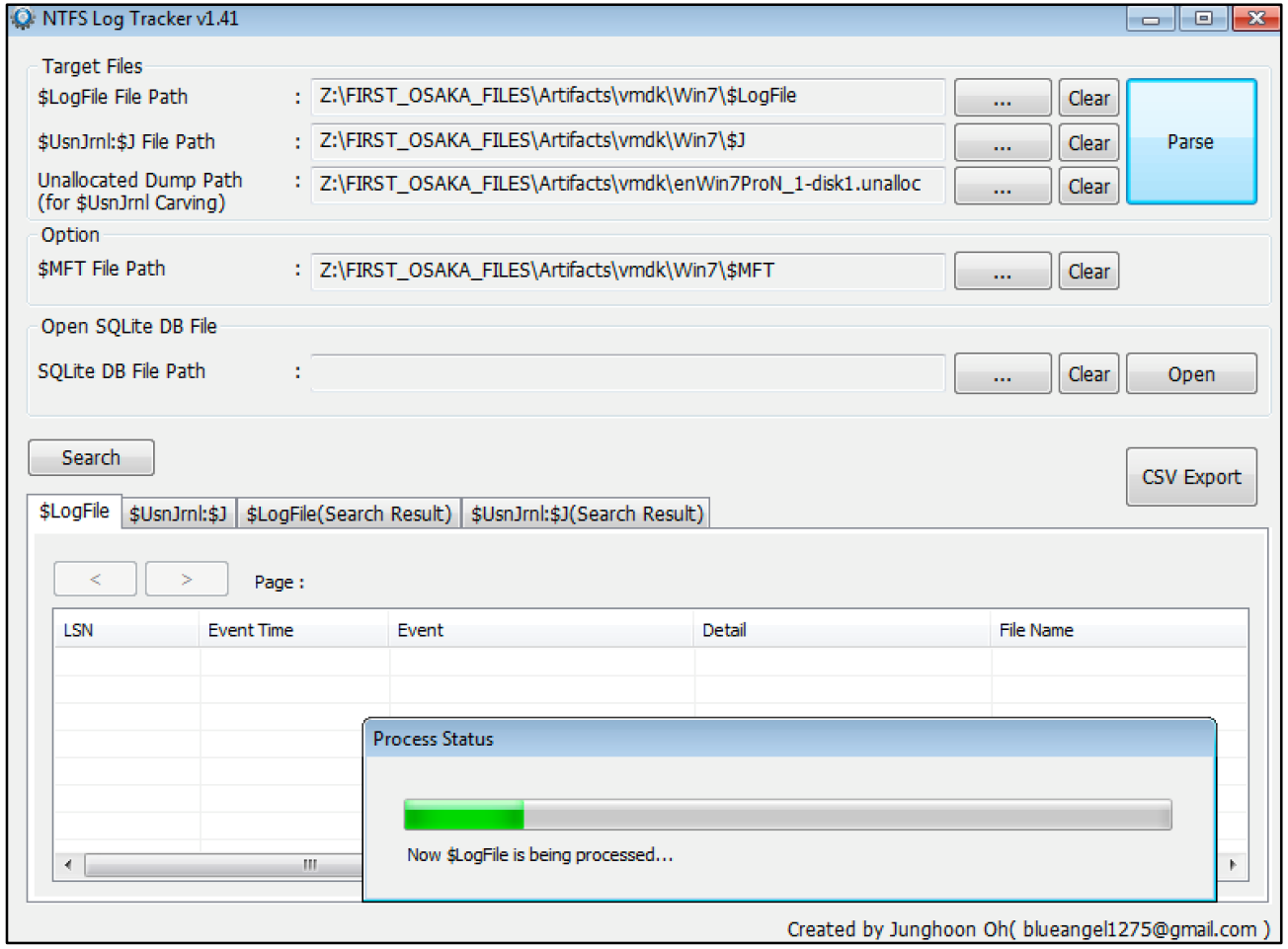
- $Logfile
  - This is a transaction log for recovering filesystem after such as accidental power fail.
  - Windows records events below in $Logfile.
    - Creation, deletion, and modification of files and folders.
    - Modification of $MFT entries.
  - This file is placed in the root of NTFS volumes.

NTFS 101: $UsnJrnl

- $UsnJrnl
  - This is a journal log. In other words, it is change log for files and folders.
  - It is used to determine history of a specific file or folder. "File history" feature of Windows 8 or later uses this function.
  - $UsnJrnl contains "$Max" and "$J".
    - $Max is metadata of this journal log.
    - $J is actual change log records.
  - $UsnJrnl:$J usually contains information of filesystem history more longer than $Logfile.

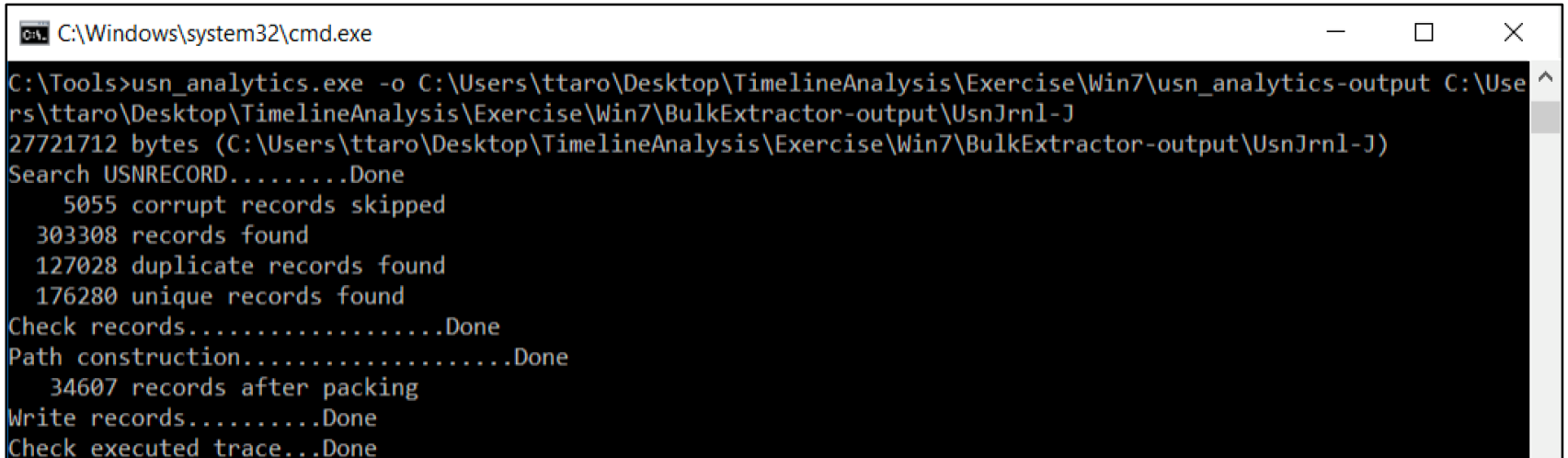
NTFS Log Parsing Tools (1)

- NTFS Log Tracker [7]
  - It parses $Logfile and $UsnJrnl:$J.
  - It can dump $UsmJrnl records from unallocated disk spaces.
  - GUI viewer with simple filter is implemented.
  - It resolves file paths by parsing $MFT.
  - It can store parsed data as SQLite DB file and load it again.
  - It parses TimeStamp fields on the second time scale. Sometimes that is not enough to sort journal events because journal events can be logged over hundreds times within a second.



NTFS Log Parsing Tools (2)

- USN Analytics [8]
  - It can parse records of $UsnJrnl that were extracted by "Bulk Extractor with Record Carving".
  - It generates not only parsed journal log, but also useful report that summarize prefetch entries, opened file entries, executable files, and so on.
  - It parses TimeStamp fields on the millisecond time scale.



Timeline Analysis Exercise 2-1 (1)

Reveal the sequence of the infection

- What did we find in the exercise before?
  - The system could be infected with a variant of Trojan Zeroaccess.
  - $EA attribute of \Windows\System32\services.exe could have been injected malicious payload by the trojan.
- Let's find which file caused the infection and when it happened by examining the output of NTFS Log Tracker.

Timeline Analysis Exercise 2-1 (2)

Reveal the sequence of the infection

- Open the below file with CSVFileView.
- "TimelineAnalysis\Win7\ntfs-log-tracker-output\UsnJrnl.csv"
- This is results of parsing $UsnJrnl:$J.
- In this case, there are few entries in "Logfile.csv" which is results of parsing $Logfile. So we check UsnJrnl.csv only.
  - These csv files are converted from SQLite DBs which are created by NTFS Log Tracker.

Timeline Analysis Exercise 2-1 (3)

Reveal the sequence of the infection

- UsnJrnl.csv has following 7 columns.
  - Timestamps
  - Filename
  - FullPath
  - EventInfo
  - File Attribute
  - USN
  - Sourceinfo

Timeline Analysis Exercise 2-1 (4)

Reveal the sequence of the infection

- Let's find the file which caused the initial infection and reveal the sequence of events.
- Hints:
  - According to the result of exercise 1-4, we suspect that the file "\Windows\System32\services.exe" is infected with Trojan Zeroaccess.
  - In this case, we focus on the files placed in ttaro's desktop. Not always. Generally we get those kind of information by other way such as examining auto-start locations or program execution artifacts.
  - Prefetch files imply program execution history. They are placed in "\Windows\Prefetch"

Timeline Analysis Exercise 2-1 (5)

Reveal the sequence of the infection

- In this case, we should apply Display Filter like this.
- This filter can display events we are interested in.

Use the following display filter:

```
FullPath CONTAINS '\Users\ttaro\Desktop' OR FullPath CONTAINS '\Windows\Prefetch\' OR FullPath = '\Windows\System32\services.exe'
```

Without Carriage-return

Operator	Field name	Condition	Value
	FullPath	Contains	\Users\ttaro\Desktop
OR	FullPath	Contains	\Windows\Prefetch\
OR	FullPath	=	\Windows\System32\services.exe

Timestamp	Target File	Action	What does it mean?
2018-02-25 00:00:52	\Users\ttaro\Desktop\ea.zip	File was created.	
2018-02-25 00:01:01	\Users\ttaro\Desktop\ea.exe	File was created.	Since its own name, the file seems to be extracted from zip file above.
2018-02-25 00:01:39	\Windows\Prefetch\EA.EXE-67BB4897.pf	File was created.	ea.exe was executed. And this is the first execution of ea.exe.
2018-02-25 00:01:39	\Windows\Prefetch\CONSENT.EXE-531BD9EA.pf	File was modified	consent.exe is related to UAC. It seems that the ea.exe required admin rights.
2018-02-25 00:01:40	\Users\ttaro\Desktop\ea.exe	File was deleted.	ea.exe was deleted immediately after its execution.
2018-02-25 00:01:40	\Windows\Prefetch\CMD.EXE-4A81B364.pf	File was modified	cmd.exe was executed. It could be launched by ea.exe since its execution time. Malware sometimes launch such as cmd.exe.
2018-02-25 00:01:40	\Windows\Prefetch\CONHOST.EXE-1F3E9D7E.pf	File was modified	conhost.exe was executed. It could be launched by ea.exe since its execution time. Malware sometimes launch such as conhost.exe with cmd.exe.
2018-02-25 00:01:40	\Windows\System32\services.exe	$EA attr was changed.	$EA attribute of the target file was modified.

Timeline Analysis Exercise 2-1 (7)

Reveal the sequence of the infection

- ea.exe was deleted immediately after it was executed.
- Its execution, deletion, and the modification of $EA on services.exe happened almost the same time.
- cmd.exe and conhost.exe were executed between execution of ea.exe and modifiction of $EA on services.exe. It seems to be that they are launched by ea.exe and did something such as manipulation of services.exe and so on.
- Execution of consent.exe seems to be UAC for ea.exe. ea.exe could have required the administrative rights.

- We can assume the ea.exe is related to the infection of services.exe.

Timeline Analysis Exercise 2-2

View the summary report of USN Analytics

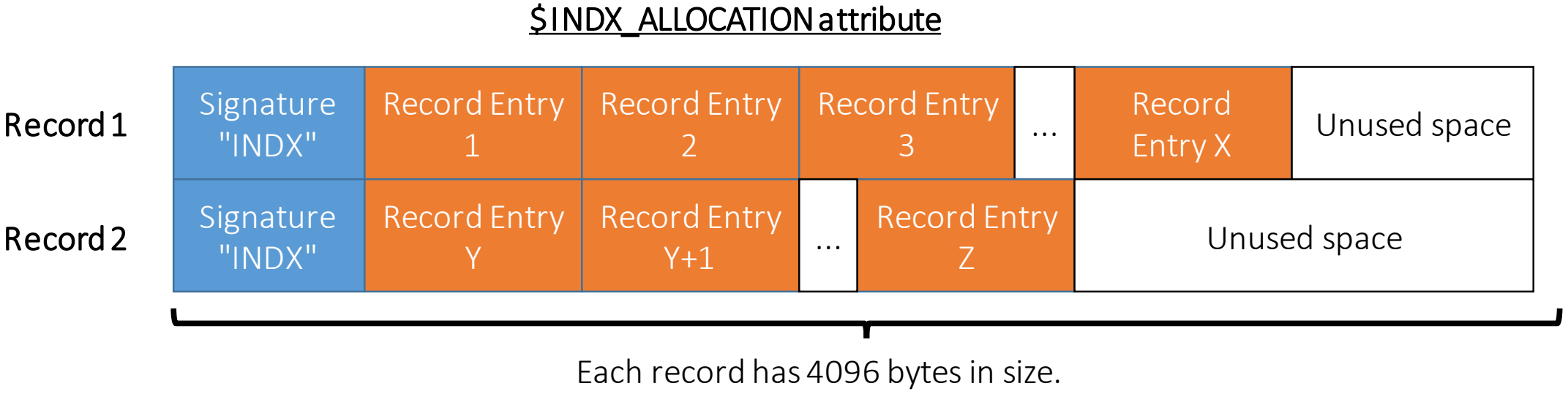
- The below is output folder of USN Analytics.
  - "TimelineAnalysis\Exercise\Win7\usn-analytics-output"
- It contains useful summary report named "usn-analytics-report.txt".
- Let's open the file with notepad or your favorite text editor/viewer. You can find lists as below.
  - prefetch exe, opened files, job, exe, dll, scr, ps1, vbe/vbs, bat, tck, PSEXESVC
- The report helps you to get essence of the journal logs briefly.

NTFS 101: $I30 and INDX (1)

- $I30 is metadata file which placed in each folder.
- It contains $INDX_ROOT and $INDX_ALLOCATION attributes. These attributes have information about files and folders in the folder such as name, timestamps, and size.
- Windows uses only $INDX_ROOT when number of files and folders placed in the folder is small. If the number becomes larger, Windows uses both of those.

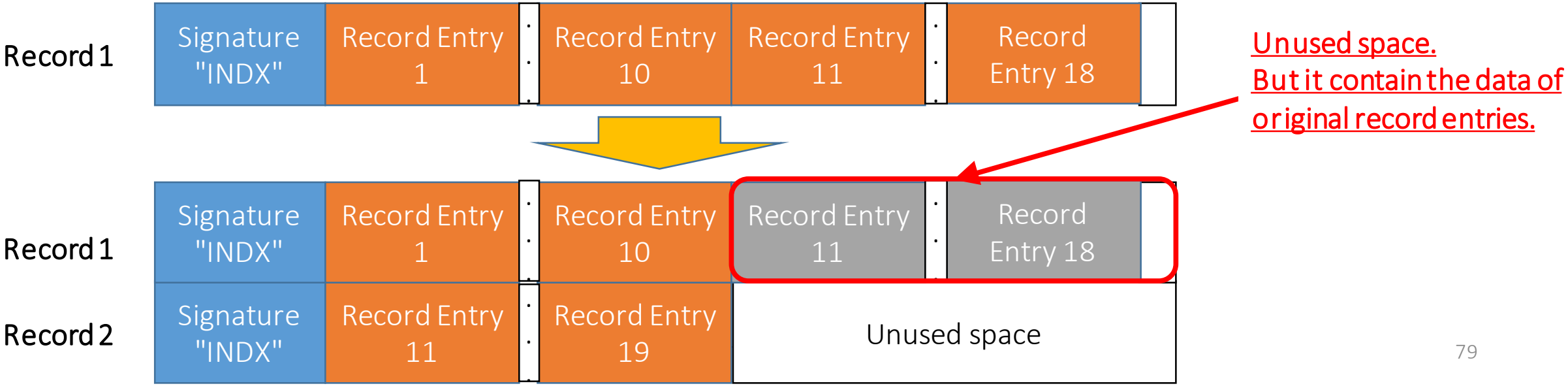
NTFS 101: $I30 and INDX (2)

- $INDX_ALLOCATION attributes consist of the "records". Each record contains multiple "record entry". Each record entry has information of the file or folder like $FN attribute such as name, timestamps, and size.
- Each record is fixed to 4096 bytes in size.



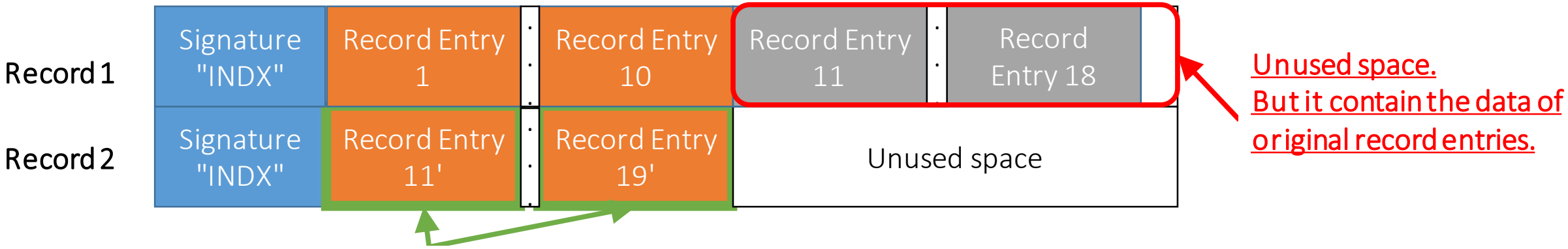
NTFS 101: $I30 and INDX (3)

- When the total size of record entries within a record become larger than 4096 bytes, The record is divided into two.
- At the time, the posterior half of the record entries are moved to the new record. Then the space after the last record entry become "unused space". But data of the original record entries remain until Windows overwrite those with other record entries. Even if the files or folders related to the record entries will be deleted.



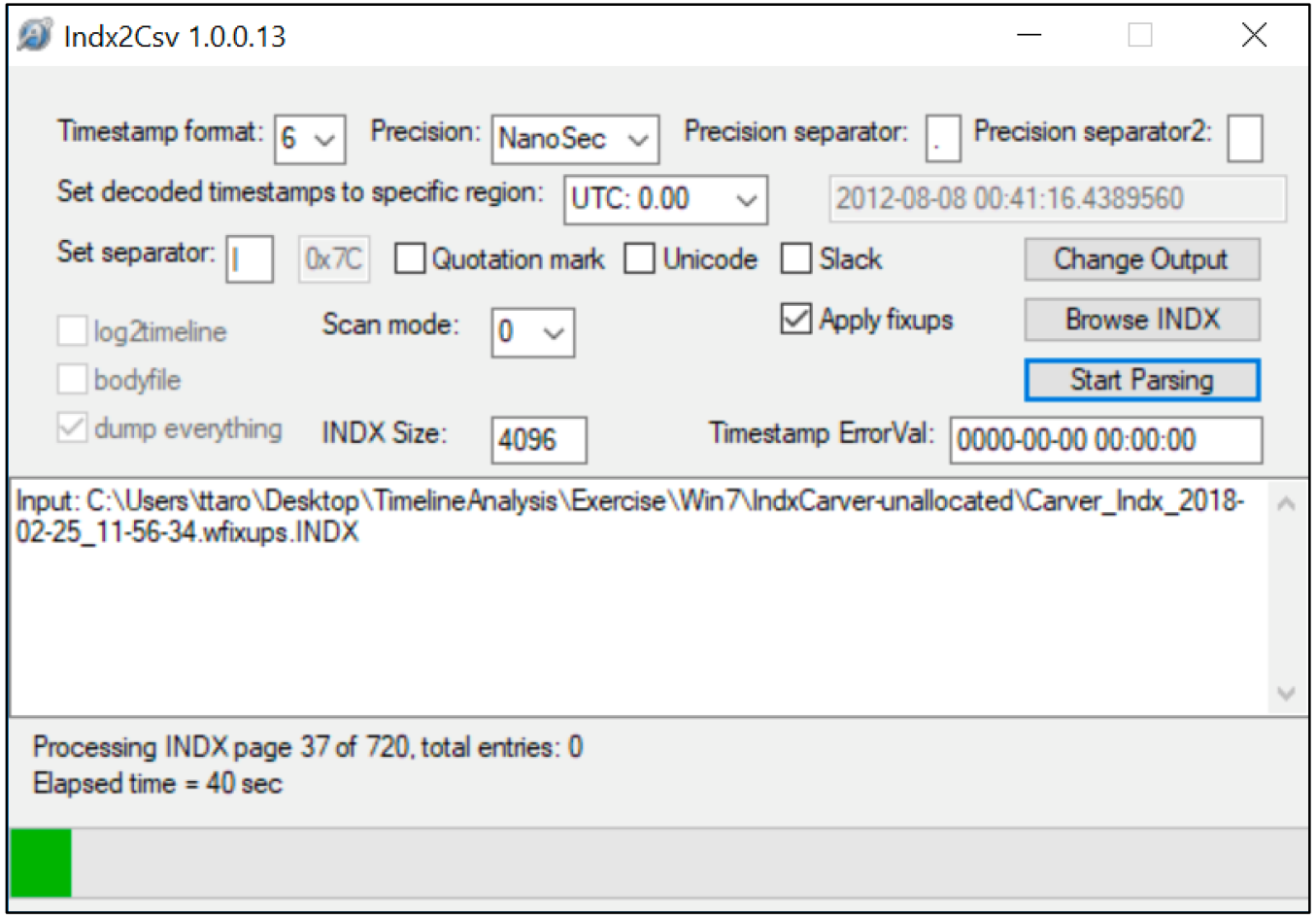
NTFS 101: $I30 and INDX (4)

- We can get information about deleted files and folders from old record entries placed in "unused spaces".
- That is sometimes useful. When $MFT, $Logfile, and $UsnJrnl contain no evidence about a file you seek for, you should examine $INDX_ALLOCATION attributes.



INDX parsing tools (1)

- Indx2Csv [9]
  - Indx2Csv is a parser for INDX records.
  - Its output contains details of several types of information such as names, timestamps, sizes, and so on.
  - The author recommend to use IndxCarver to collect data of INDX records from disk images.



INDX parsing tools (2)

- INDXParse [10]
  - It parses a single $I30 metadata file placed in each folder.
  - It is useful for understanding $I30.

C:\Windows\system32\cmd.exe

```
C:\Users\ttaro\Desktop\TimelineAnalysis\Exercise>C:\Tools\INDXParse-master\INDXParse.py $I30
FILENAME,       PHYSICAL SIZE,  LOGICAL SIZE,   MODIFIED TIME,  ACCESSED TIME,  CHANGED TIME,   CRE
desktop.ini,    288,    282,    2018-02-22 12:10:03.454479,     2018-02-22 12:10:03.454479,     201
79,     2018-02-22 12:10:03.454479
GoodEveningForensic.txt,        64,     61,     2011-01-01 06:00:01,    2011-01-01 06:00:01,    201
15,     2011-01-01 06:00:01
GOODEV~1.TXT,   64,     61,     2011-01-01 06:00:01,    2011-01-01 06:00:01,    2018-02-24 14:59:13
01 06:00:01
GoodNightForensic.txt,  56,     55,     2001-01-01 12:00:00,    2001-01-01 12:00:00,    2001-01-01
01 12:00:00
GOODNI~1.TXT,   56,     55,     2001-01-01 12:00:00,    2001-01-01 12:00:00,    2001-01-01 12:00:00

HelloADS.txt,   32,     28,     2018-02-24 15:00:27.115707,     2018-02-24 15:00:14.074165,     201
```

INDX Carving Tools (1)

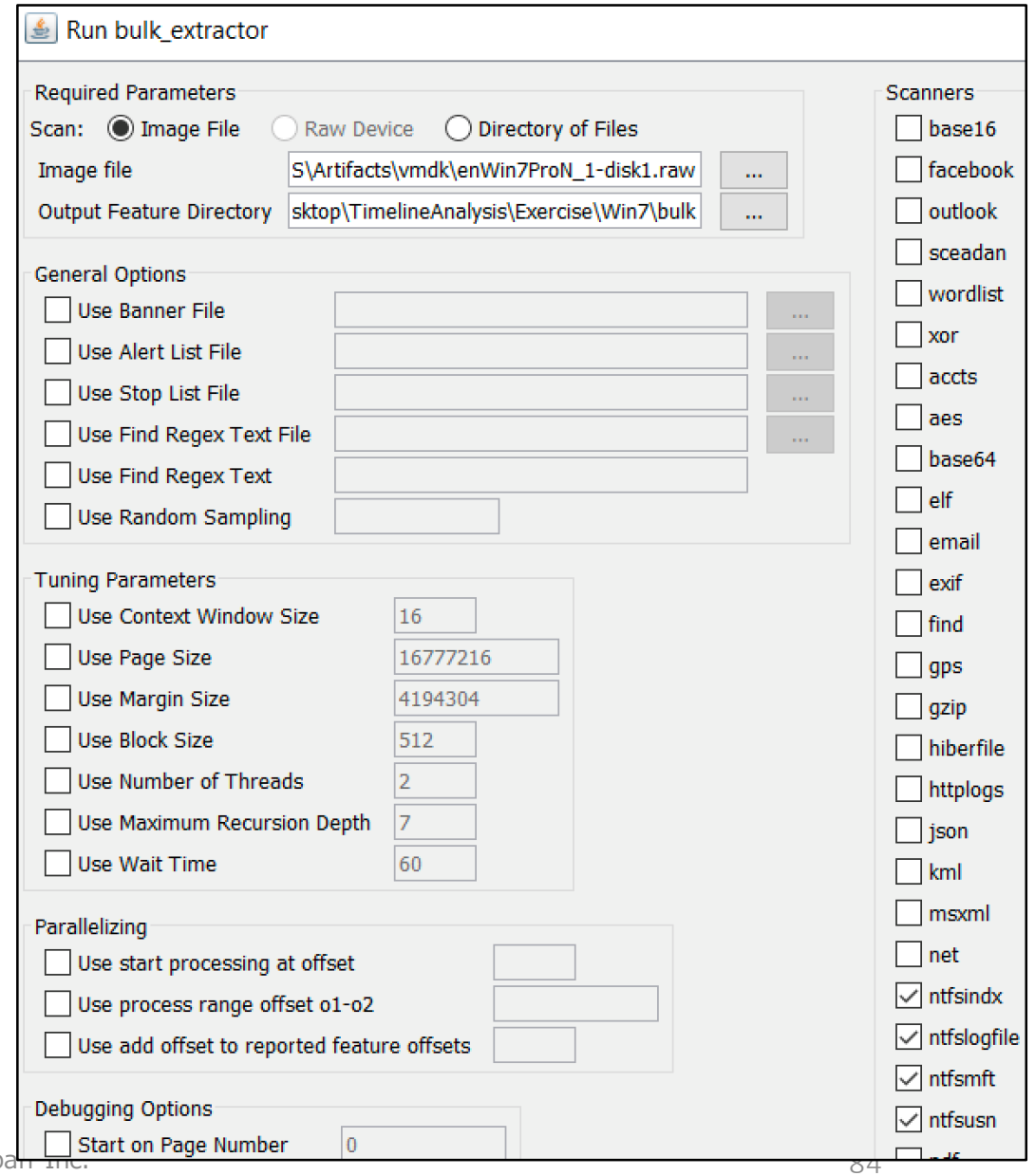
- IndxCarver [11]
  - This tool dumps individual INDX records from unallocated spaces.
  - We can recover old INDX records to get information about deleted files and folders with this tool.

C:\Tools\IndxCarver-master\IndxCarver64.exe

```
IndxCarver v1.0.0.5
Input: Z:\static\FIRST_OSAKA_FILES\Artifacts\vmdk\enWin7ProN_1-disk1.raw
Input filesize: 16106127360 bytes
OutFileWithFixups: C:\Tools\IndxCarver-master\Carver_Indx_2018-03-02_20-46-34.wfixups.INDX
OutFileWithoutFixups: C:\Tools\IndxCarver-master\Carver_Indx_2018-03-02_20-46-34.wofixups.INDX
OutFileFalsePositives: C:\Tools\IndxCarver-master\Carver_Indx_2018-03-02_20-46-34.false.positive.INDX
INDX size configuration: 4096
0 %
```

INDX Carving Tools (2)

- Bulk Extractor with Record Carving [6]
  - We mentioned it before as a $MFT carving tool. It also contains scanner plug-ins for records of $MFT, $LogFile, $UsnJrnl$J, $INDEX_ALLOCATION, and utmp structure.
  - It can recover those records from disk images.



Timeline Analysis Exercise 3 (1)

- Let’s assume these conditions are given.
  - You are investigating a compromised Windows client.
  - You have already known that attackers installed a RAT to the client. And they also installed some utility programs for their action such as checking environment, lateral movement, and so on. (Since you found that from other artifacts.)
  - Those utility programs were installed to the path below.
    - "\ProgramData\s"

Timeline Analysis Exercise 3 (2)

- Let's answer to the questions below by investigating artifacts placed in "Training_Materials\TimelineAnalysis\Win10".
  1. How can you confirm that the folder "\ProgramData\s" really existed?
  2. Are there any suspicious point in creation or deletion of the folder? If there are, what do they mean?

Timeline Analysis Exercise 3 (3)

How do you confirm that the folder "\ProgramData\s" really existed?

- There are no entry related to the folder "\ProgramData\s" in $MFT and $Logfile, but $UsnJrnl contains it. So you can find the deletion log in output of ntfs-log-tracker.
- Let's check data under the following folder.
  - Training_Materials\TimelineAnalysis\Win10\ntfs-log-tracker-outpunt

Timeline Analysis Exercise 3 (4)

How do you confirm that the folder "\ProgramData\s" really exist?

- In this case, UsnJrnl.csv is too large to open with CSVFileView and other CSV viewers. (Ordinary CSV viewers can not handle logs over about 320,000 lines well.)
- To handle the large CSV data, we should use ElasticSearch and Kibana.

Timeline Analysis Exercise 3 (5)

<u>View the journal log with ElasticSearch and Kibana.</u>

1. Double-click the bat file below to launch ElasticSearch and Kibana.
   - "elasticsearch\es_kibana.bat"
2. Generate configuration file for loading the CSV file "UsnJrnl.csv" into ElasticSearch by executing following command at the folder "elasticsearch".

```
em bulk.bat guess .\seed-ntfs-log-tracker.yml -o config-ntfs-log-tracker.yml
```

This is the seed file which contains the path to the CSV file, some definitions and so on.

This is the name of the file to generate.

```
in:
  type: file
  path_prefix: "../Training_Materials/TimelineAnalysis/Win10/ntfs-log-tracker-output/UsnJrnl.csv"
out:
  type: elasticsearch
  index: ntfslogtracker-win10
  index_type: ntfslogtracker
  nodes:
    - host: localhost
```

Timeline Analysis Exercise 3 (

View the journal log with ElasticSearch and Kibana.

3. Modify the generated configuration file "config-ntfs-log-tracker.yml" like following.
   - **Just add this line** since we handle the timestamps as JST (UTC+9) in this case.

default_timezone: 'Asia/Tokyo'

```
in:
  type: file
  path_prefix: ../Training_Materials/TimelineAnalysis/Win
  parser:
    charset: UTF-8
    newline: CRLF
    type: csv
    delimiter: ','
    quote: '"'
    escape: '"'
    trim_if_not_quoted: false
    skip_header_lines: 1
    allow_extra_columns: false
    allow_optional_columns: false
    default_timezone: 'Asia/Tokyo'
    columns:
    - {name: TimeStamp, type: timestamp, format: '%Y-%
    - {name: USN, type: long}
    - {name: Filename, type: string}
    - {name: FullPath, type: string}
    - {name: EventInfo, type: string}
    - {name: SourceInfo, type: string}
    - {name: File Attribute, type: string}
out:
```

Timeline Analysis Exercise 3 (7)

View the journal log with ElasticSearch and Kibana.

4. Test the modified configuration file by following command. You can check the output format.

```
em bulk.bat preview config-ntfs-log-tracker.yml
```

5. Load data from the CSV file into ElasticSearch by executing command below.

```
em bulk.bat run config-ntfs-log-tracker.yml -c diff.yml
```

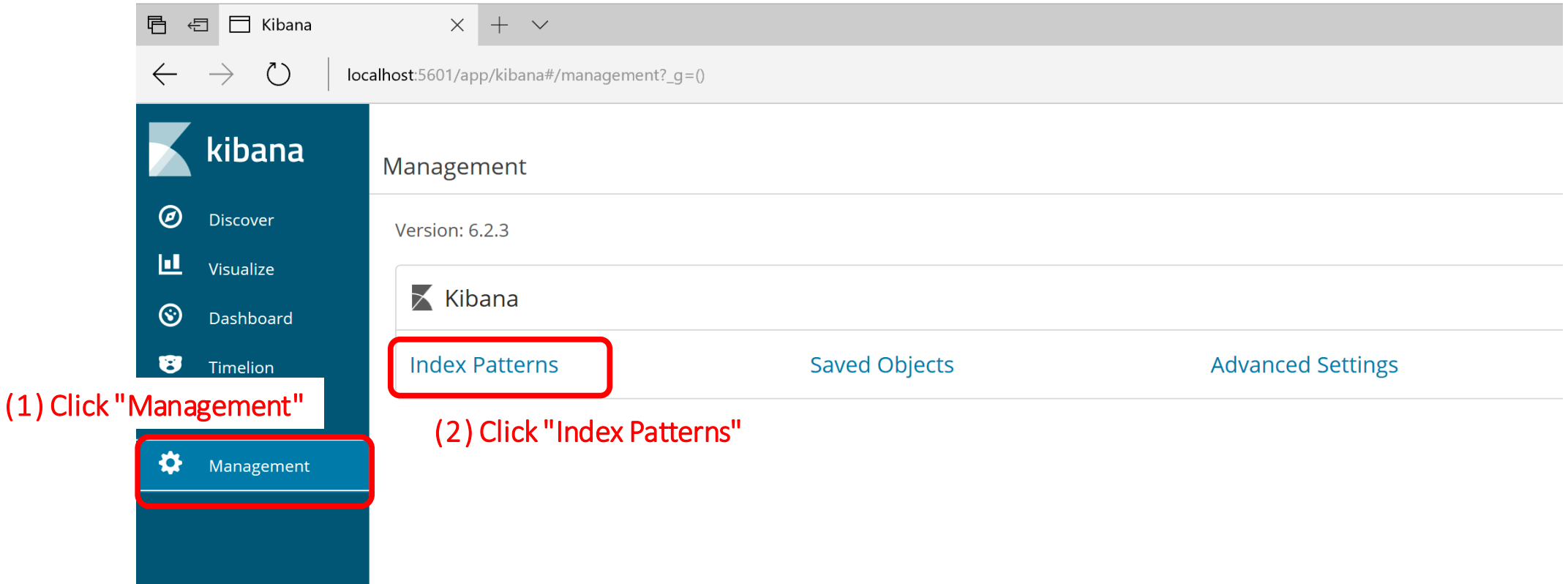
This file is to read and write the next configuration diff.
By using this file, you can avoid import duplication.

6. Finally, open the following URL with Web browser such as Edge.
   - http://localhost:5601/

Timeline Analysis Exercise 3 (8)

View the journal log with ElasticSearch and Kibana.

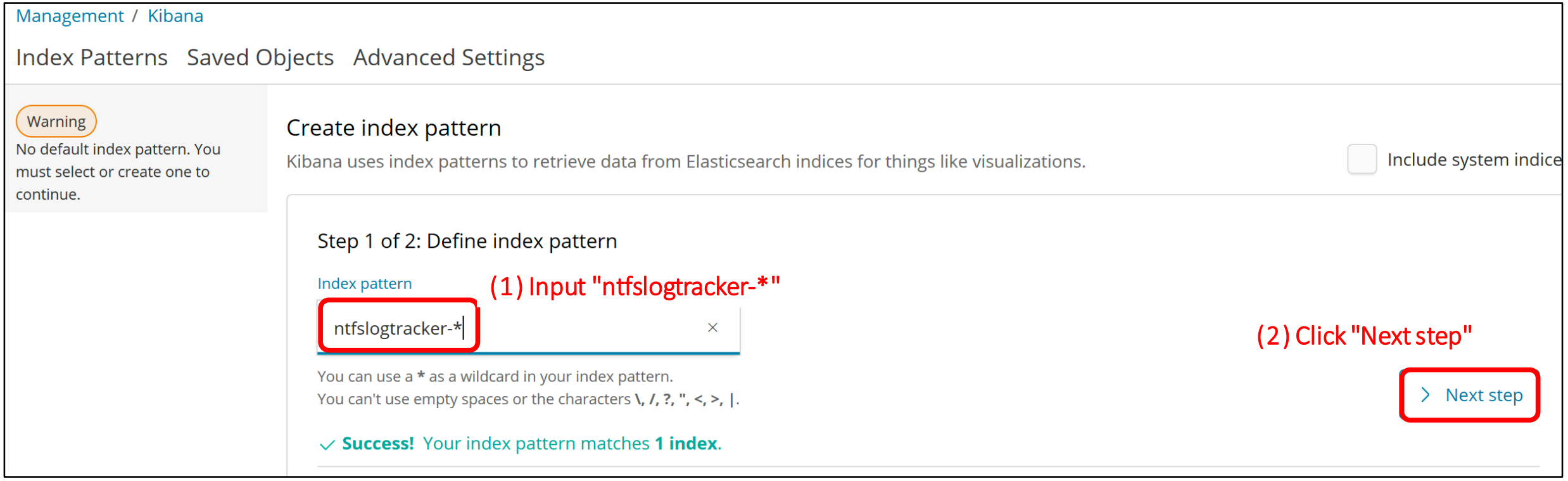
7. Click "Management" in left menu, then click "Index Patterns" to move to the "Index Patterns" page.



Timeline Analysis Exercise 3 (9)

View the journal log with ElasticSearch and Kibana.

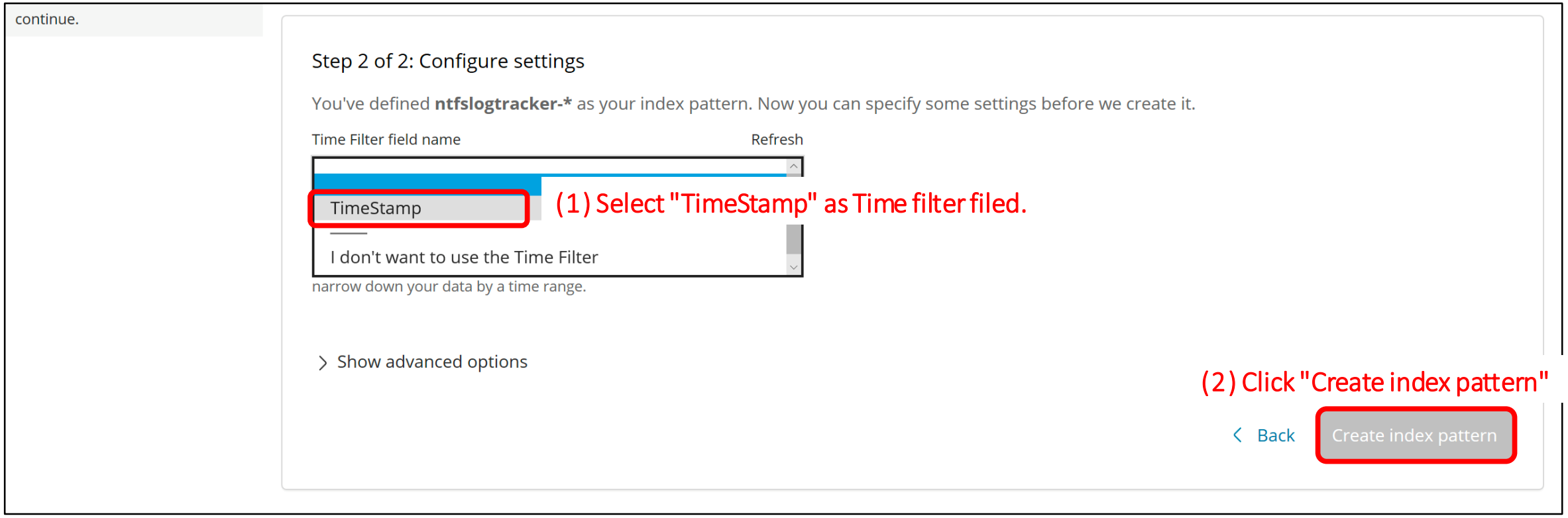
8. Input string "ntfslogtracker-*" as index pattern, then click "Next step" to create index for imported data. This string indicate the indexes which we use.



Timeline Analysis Exercise 3 (10)

View the journal log with ElasticSearch and Kibana.

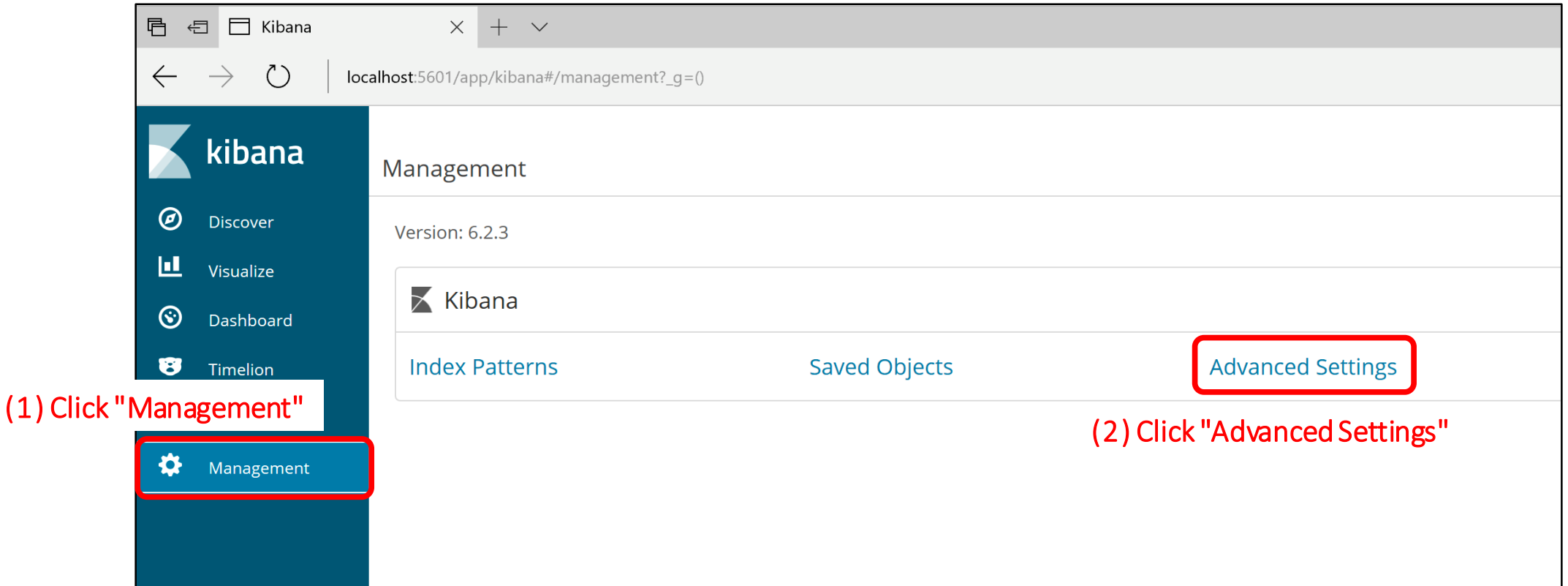
9. Select "TimeStamp" and click "Create index pattern" to define time filter field.



Timeline Analysis Exercise 3 (11)

View the journal log with ElasticSearch and Kibana.

10. Move to the "Advanced Settings" page to set some options.

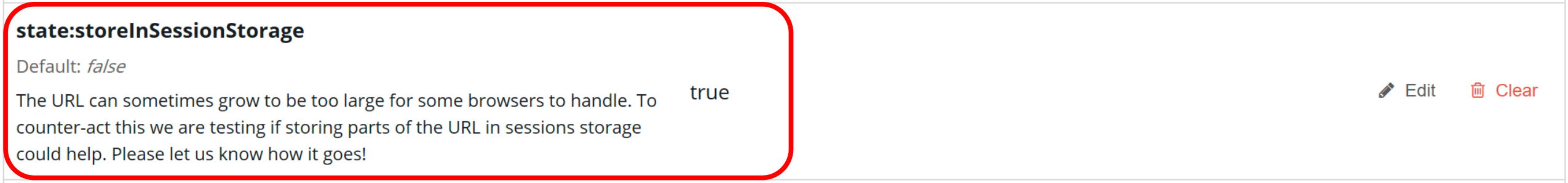


Timeline Analysis Exercise 3 (12)

View the journal log with ElasticSearch and Kibana.

11. Modify options like below.
    - Change discover:sampleSize from 500 to 10000.

    - Change state:storeInSessionStorage from false to true.



    - OK, we have finished the setting. Let's back to the exercise 3.

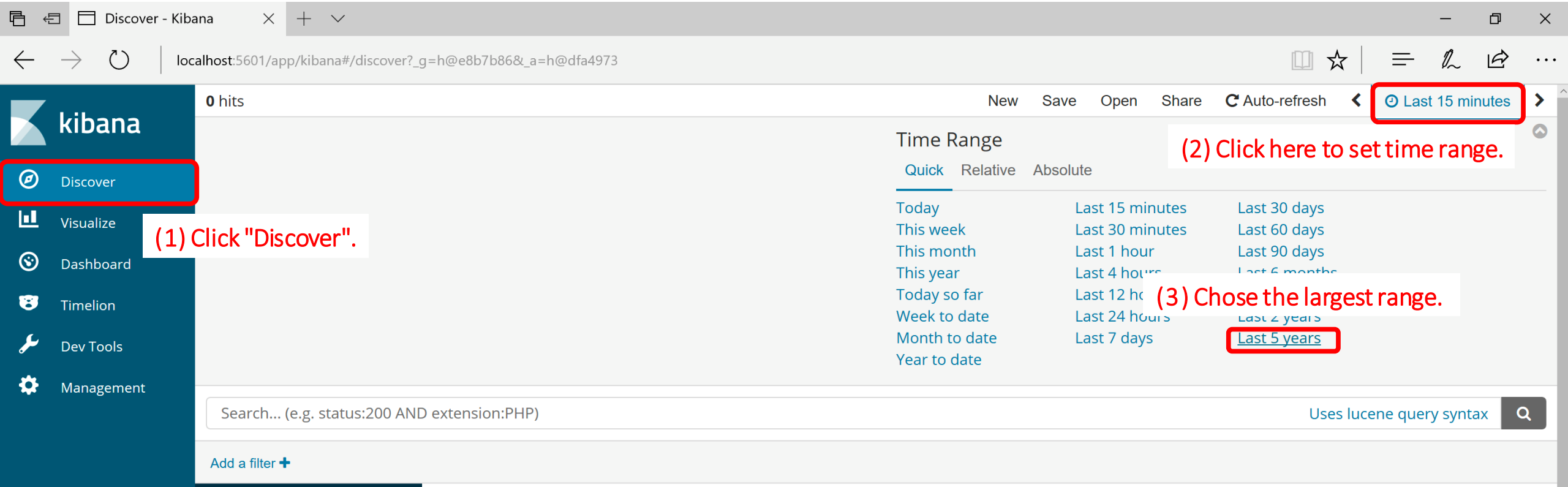
Timeline Analysis Exercise 3 (13)

- Back to the exercise 3. Let's reconfirm questions below.
  1. How can you confirm that the folder "\ProgramData\s" really existed?
  2. Are there any suspicious point in creation or deletion of the folder? If there are, what do they mean?

Timeline Analysis Exercise 3 (14)

How can you confirm that the folder "\ProgramData\s" really exist?

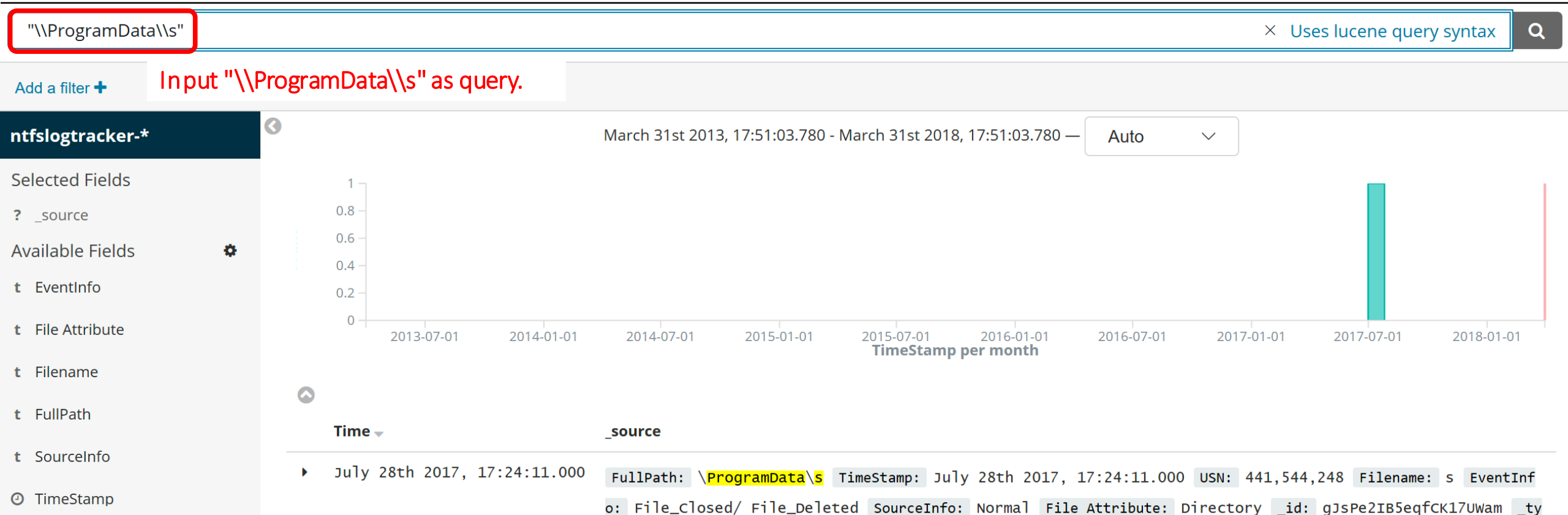
- First, we should specify the time range to search.
- In this case, we use the largest range.



Timeline Analysis Exercise 3 (15)

How can you confirm that the folder "\ProgramData\s" really exist?

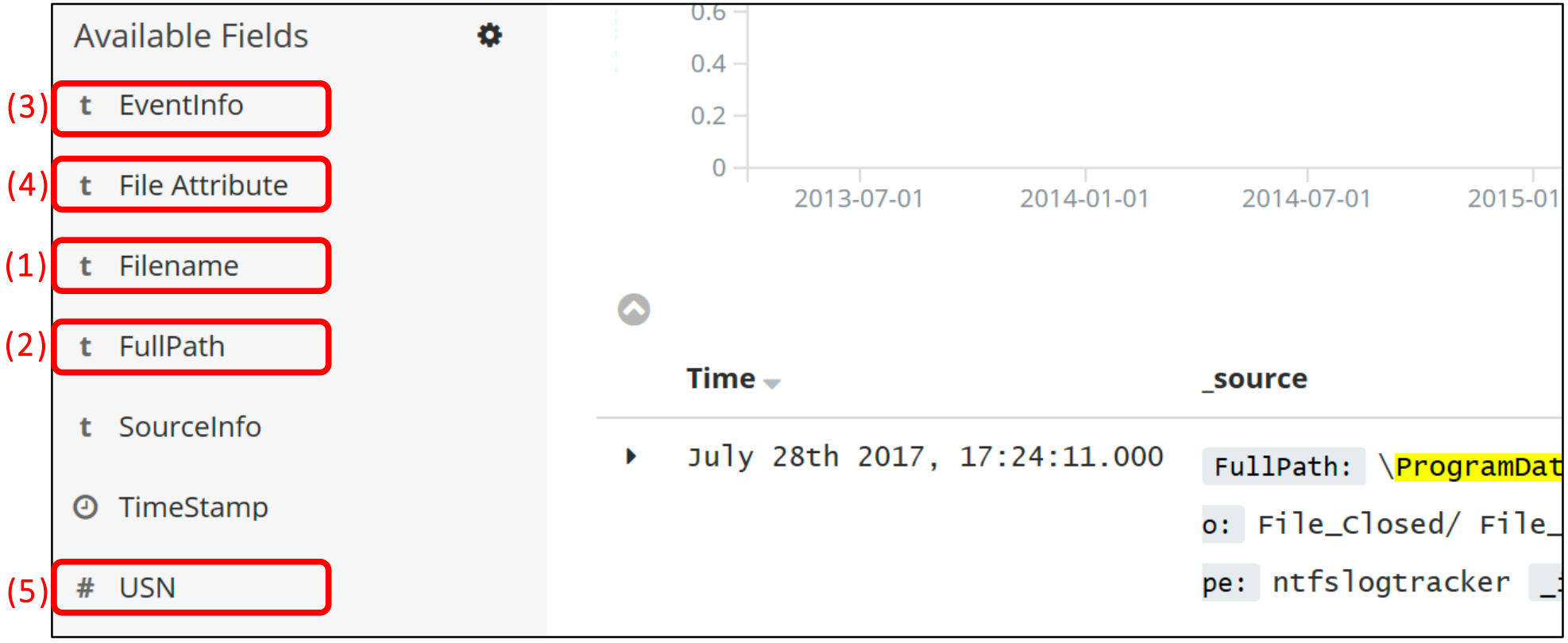
- To search the data with the path, input **"\\ProgramData\\s"** as query and execute.



Timeline Analysis Exercise 3 (16)

How can you confirm that the folder "\ProgramData\s" really exist?

- To make the result easy to view, add fields by clicking the link in order below.



Timeline Analysis Exercise 3 (17)

How can you confirm that the folder "\ProgramData\s" really exist?

- This is the deletion event of the folder. It can be evidence of that the folder existed.

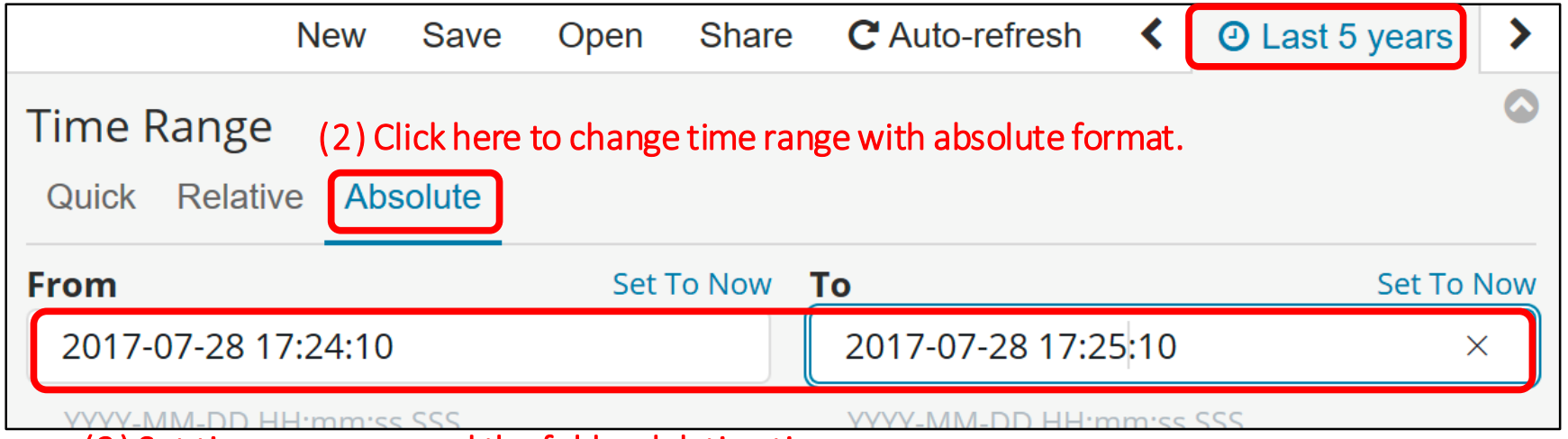
Time	Filename	FullPath	EventInfo	File Attribute	USN
July 28th 2017, 17:24:11.000	s	\ProgramData\s	File_Closed/ File_Deleted	Directory	441,544,248

Timeline Analysis Exercise 3 (18)

Are there any suspicious point in creation or deletion of the folder? If there are, what do they mean?

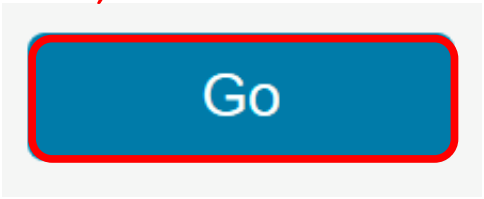
- Let's view the events logged around the deletion time.

(1) Click here to change the time range.



(3) Set time range around the folder deletion time.

(4) Then, click the "Go" button.



(5) Finally, clear the query and execute it.

Timeline Analysis Exercise 3 (19)

Are there any suspicious point in creation or deletion of the folder? If there are, what do they mean?

- Sort data by USN since TimeStamp field does not have enough time resolution.
  - NTFS Log Tracker parses TimeStamp fields on the second time scale. That is not enough to sort journal events because journal events can be logged over hundreds times within a second.
- Then, let's check events which were logged at around the deletion event of the folder.

Click this upper arrow to sort data by USN.

	Filename	FullPath	EventInfo	File Attribute	USN «
7:25:10.000	www.j-cast[1].xml	\Users\kfuma\AppData\LocalLow\Microsoft\Internet Explorer\DOMStore\N2R52GOT\www.j-cast[1].xml	File_Added/ File_Truncated	Archive/ Not_Content_Indexed	441,613,312
7:25:10.000	www.j-cast[1].xml	\Users\kfuma\AppData\LocalLow\Microsoft\Internet Explorer\DOMStore\N2R52GOT\www.j-cast[1].xml	File_Added/ Data_Overwritten/ File_Truncated	Archive/ Not_Content_Indexed	441,613,408

Sort by USN

wizSafe

July 28th 2017, 17:24:11.000	w.vbs		Data_Overwritten/ File_Closed	Archive	441,535,888
July 28th 2017, 17:24:11.000	w.vbs		File_Renamed_Old	Archive	441,535,960
July 28th 2017, 17:24:11.000	prograAAAAAAAAA.AAA	\prograAAAAAAAAA.AAA	File_Renamed_New	Archive	441,536,032
July 28th 2017, 17:24:11.000	prograAAAAAAAAA.AAA	\prograAAAAAAAAA.AAA	File_Renamed_New/ File_Closed	Archive	441,536,136
July 28th 2017, 17:24:11.000	prograAAAAAAAAA.AAA	\prograAAAAAAAAA.AAA	File_Renamed_Old	Archive	441,536,240
					441,536,344
					441,536,512
July 28th 2017, 17:24:11.000	prograBBBBBB	\prograBBBBBBBBB.BBB	File_Renamed_Old	Archive	441,536.
		<snip>			
July 28th 2017, 17:24:11.000	prograZZZZZZZZZ.ZZZ	\prograZZZZZZZZZ.ZZZ	File_Renamed_New	Archive	441,543,936
July 28th 2017, 17:24:11.000	prograZZZZZZZZZ.ZZZ	\prograZZZZZZZZZ.ZZZ	File_Renamed_New/ File_Closed	Archive	441,544,040
July 28th 2017, 17:24:11.000	prograZZZZZZZZZ.ZZZ	\prograZZZZZZZZZ.ZZZ	File_Closed/ File_Deleted	Archive	441,544,144
July 28th 2017, 17:24:11.000	s	\ProgramData\s	File_Closed/ File_Deleted	Directory	441,544,248

The logs show that files were overwritten, then renamed many times, and deleted.

This is the deletion log.

Timeline Analysis Exercise 3 (21)

Are there any suspicious point in creation or deletion of the folder? If there are, what do they mean?

- Consequently, we can say that many files were rewritten, then renamed many times and deleted.
- Such operations are known as the deletion method of some data-erasing tools, such as SDelete and CClearner.
- In this case, we can determine that attackers used kinds of data-erasing tool to delete their tools.

To be continued…