File System Timeline Analysis Appendix

Extra Exercise: MFT Basics 1

Launching Program From ADS With PowerShell

Extra Exercise: MFT Basics 1

Launching Program From ADS With PowerShell (1)

- Since it was possible to load program from ADS, attacker often launch malware from ADS.
- Therefore, Microsoft restricted starting program from ADS on Windows 7 or later.
- However, we can load program from ADS by using reflective PE injection technique.
- I will show you a demo for executing a program from ADS!
- In addition, I will show you the usage of an MFT parsing tool at the beginning of the demo.

Extra Exercise: MFT Basics 1

Launching Program From ADS With PowerShell (2)

This operation works only on NTFS volumes.

- This command is to save the PowerShell code to an ADS of a target file.

```
TYPE m64.ps1 > confidential.txt:mimi
```

  - "TYPE" command works similar to "cat" on Linux.
  - This command saves the content of a script file "m64.ps1" to an ADS of the target file "confidential.txt". And the name of the ADS is set to "mimi".

Extra Exercise: MFT Basics 1

Launching Program From ADS With PowerShell (3)

- We can confirm the existence of the ADS by executing "DIR" command with "/R" option.

```
C:\Users\ttaro\Desktop\Training_Materials\TimelineAnalysis\example>DIR /R
 Volume in drive C has no label.
 Volume Serial Number is 90EC-0BBA

 Directory of C:\Users\ttaro\Desktop\Training_Mater

03/15/2018  12:45 PM    <DIR>          .
03/15/201                <DIR>          ..
03/15/201                              32 confidential.txt
                             1,504,240 confidential.txt:mimi:$DATA
02/28/2018  08:56 PM         1,101,182 m32.ps1
02/28/2018  08:55 PM         1,504,240 m64.ps1
               3 File(s)      2,605,454 bytes
               2 Dir(s)  42,052,702,208 bytes free
```

This entry shows that a ADS is contained in "confidential.txt" and the name of the stream is "mimi".

Same size!

Extra Exercise: MFT Basics 1

Launching Program From ADS With PowerShell (4)

- We can also confirm the existence of the ADS and other information by using MFTRCRD.
  - You can view the result of the MFTRCRD command for the file "confidential.txt" by opening the file below.
  - "E:\Artifacts\other_timeline_analysis\example\output-MFTRCRD.txt".

  - This is a command line sample for MFTRCRD command.

```
MFTRCRD <full-path-to>confidential.txt -d  indxdump=off 1024 -s
```

This operation requires Admin rights.

"1,024" is the physical sector size of the disk storage. You should chose 1,024 or 4,096 for this option.

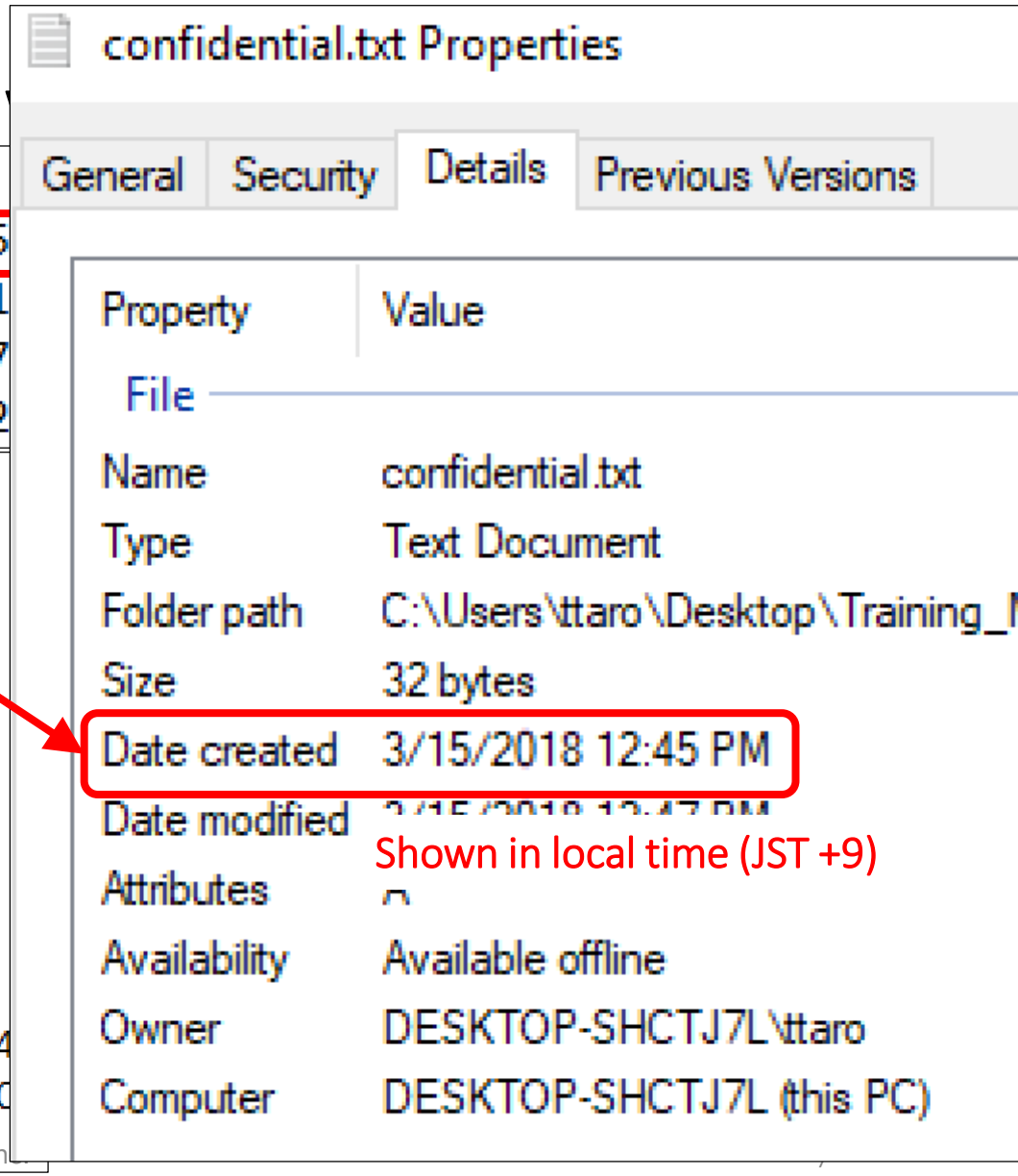
Extra Exercise: MFT Basics 1

Launching Program From ADS With Po

Recorded in UTC

```
$STANDARD_INFORMATION 1:
File Create Time (CTime): 2018-03-15 03:45:27:385
File Modified Time (ATime): 2018-03-15 03:47:04:1
MFT Entry modified Time (MTime): 2018-03-15 03:47
File Last Access Time (RTime): 2018-03-15 03:45:2
```

```
$DATA 2:
Non-resident flag: 01
Name length: 4
Flags:
Attribute Id: 0007
Non-Resident - Starting VCN: 0
Non-Resident - Last VCN: 367
Non-Resident - Offset to the Data Runs: 72
Non-Resident - Compression Unit Size: 0
Non-Resident - Allocated size of the attribute: 1507328
Non-Resident - Real size of the attribute: 1504240
Non-Resident - Initialized data size of the stream: 150424
Non-Resident - DataRuns: 31014078073101C98F02210280003104C
The Attribute's Name: mimi
```



Shown in local time (JST +9)

Extra Exercise: MFT Basics 1

Launching Program From ADS With PowerShell (6)

This operation requires PowerShell 3.0 or later.

- This PowerShell command line loads a script from the ADS of the target file.

```
powershell $m=Get-content -Path 'confidential.txt' -Stream 'mimi' -Raw;
Invoke-Expression $m; mimiWrapper
```

Enter this in a single line.

- We can see the Mimikatz's prompt. Then type "exit" to quit.

```
C:\Users\ttaro\Desktop\Training_Materials\TimelineAnalysis\example>powershell $m=Get-content -Path 'confid
ential.txt' -Stream 'mimi' -Raw; Invoke-Expression $m; mimiWrapper

  .#####.   mimikatz 2.1.1 (x64) built on Feb  5 2018 20:43:06
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # exit
Bye!

C:\Users\ttaro\Desktop\Training_Materials\TimelineAnalysis\example>
```

Extra Exercise: MFT Basics 2

Viewing MFT entries and examining a certain folder

Extra Exercise: MFT Basics 2

Viewing MFT entries and examining a certain folder (1)

- Conditions:
  - We parsed a $MFT of ttaro's disk image with analyzeMFT.py. The following is the result.
    - E:\Artifacts\other_timeline_analysis\Win7\analyzeMft-output.csv
- Goal:
  - By parsing $MFT, list up the files and folders that were placed on the Desktop of user "ttaro".

- We used the command below to execute analyzeMft.py in this case.

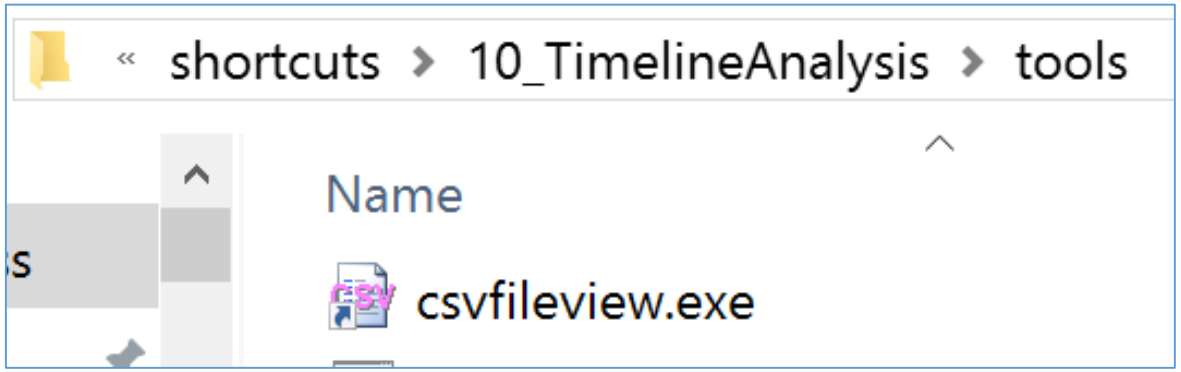
```
analyzeMFT.py -f E:\Artifacts\other_timeline_analysis\Win7\artifact\$MFT -a -e -o analyzeMft-output.csv
```

Extra Exercise: MFT Basics 2

Viewing MFT entries and examining a certain folder (2)

- Let's open it with CSVFileView.
  - "E:\Artifacts\other_timeline_analysis\Win7\analyzeMft-output.csv".

- In order to open it, drag the file above and drop it into csvfileview.exe icon in the shortcut folder.



Extra Exercise: MFT Basics 2

Viewing MFT entries and examining a certain folder (3)

- An output of analyzeMFT.py has 54 columns. Most of them are same as the one we viewed in demo 0. But the following useful columns were 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.

Extra Exercise: MFT Basics 2

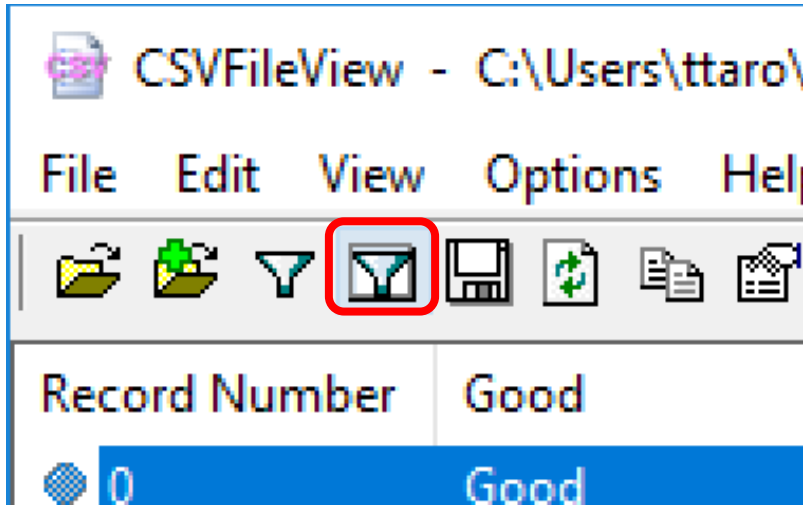
Viewing MFT entries and examining a certain folder (4)

- Notice:
  - Original ZIP format contains the creation time of files in the 10 millisecond scale. It fills the values under 10 millisecond with zero when extracting the files. Thus, old ZIP archives can trigger false positives by this uSec Zero detection.
  - In these days, ordinal ZIP archivers use extra fields to support higher-resolution timestamps. Thus, we hardly face those false positives.

Extra Exercise: MFT Basics 2

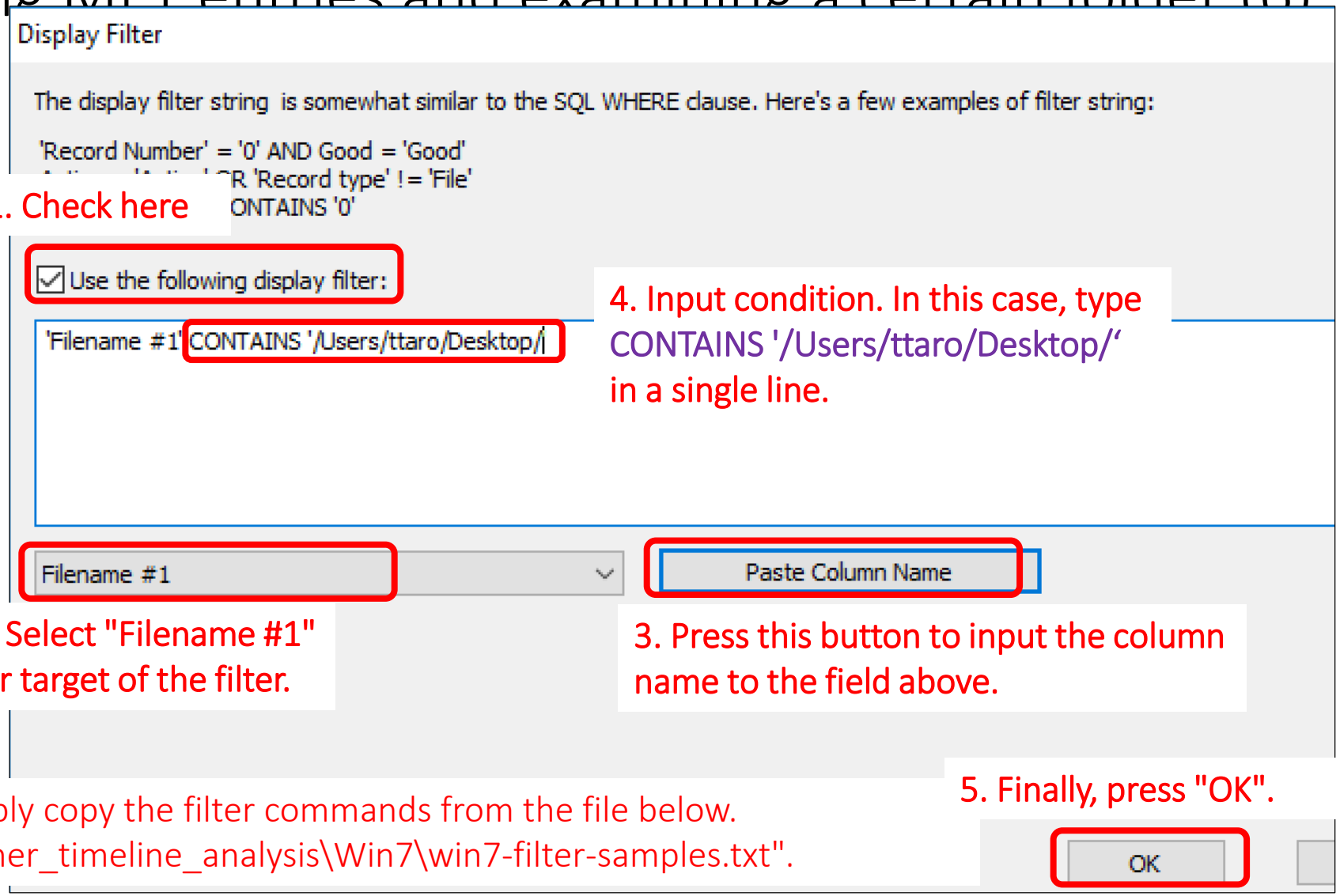
Viewing MFT entries and examining a certain folder (5)

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



Extra Exercise: MFT Basics 2

Viewing MFT entries and examining a certain folder (6)



1. Check here

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

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

4. Input condition. In this case, type CONTAINS '/Users/ttaro/Desktop/' in a single line.

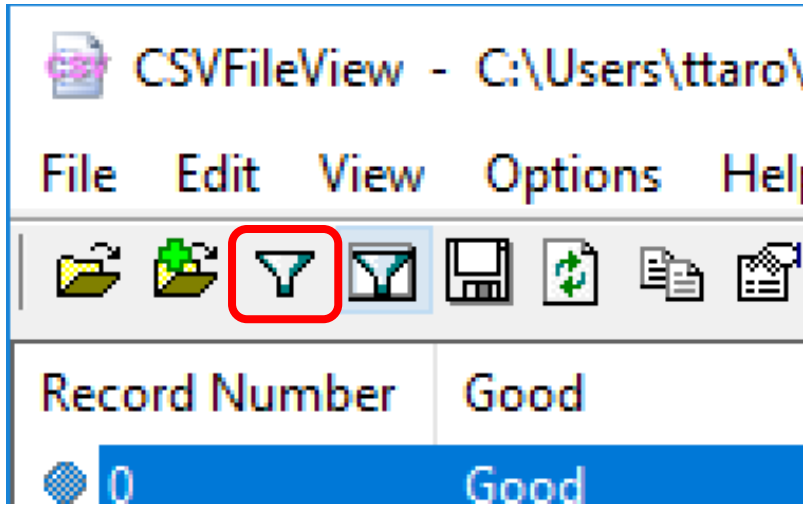
5. Finally, press "OK".

Or, you can simply copy the filter commands from the file below.
"E:\Artifacts\other_timeline_analysis\Win7\win7-filter-samples.txt".

Extra Exercise: MFT Basics 2

Viewing MFT entries and examining a certain folder (7)

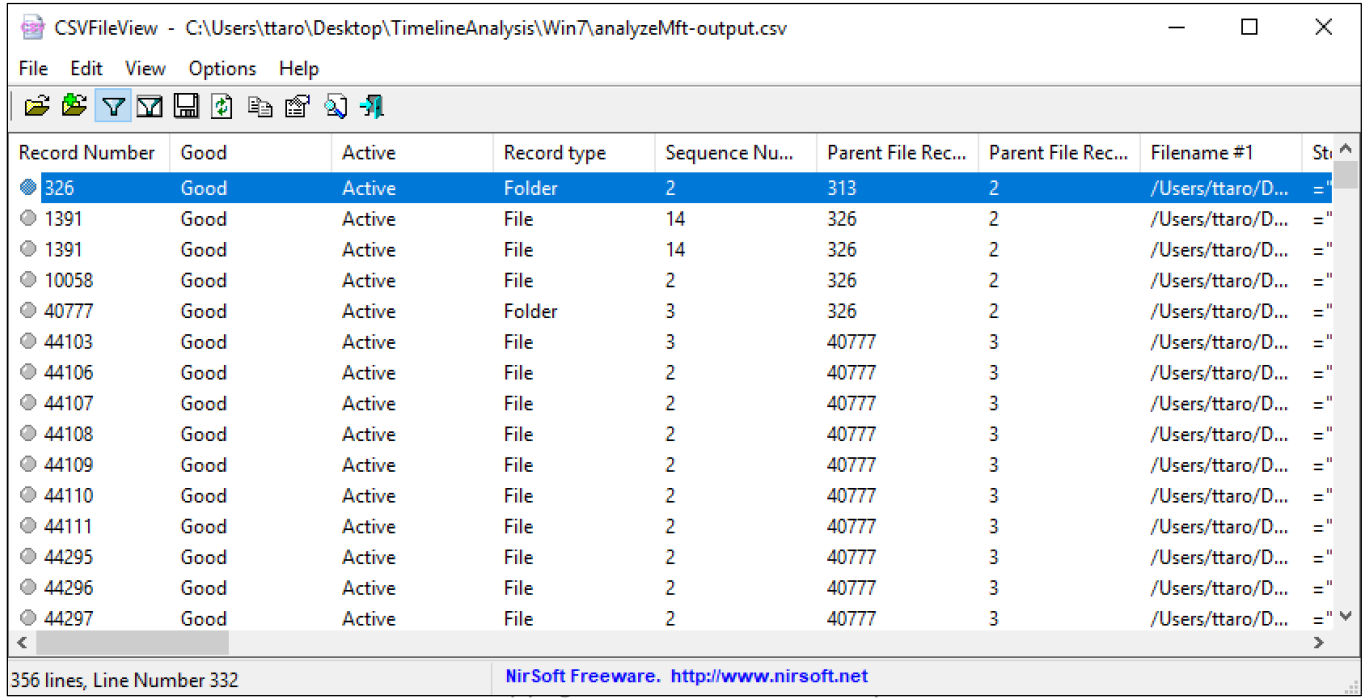
- To apply the filter, activate it with "Use Display Filter" button.



Extra Exercise: MFT Basics 2

Viewing MFT entries and examining a certain folder (8)

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

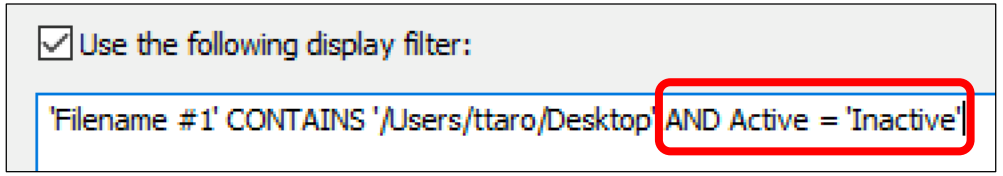


Record Number	Good	Active	Record type	Sequence Nu...	Parent File Rec...	Parent File Rec...	Filename #1	St
326	Good	Active	Folder	2	313	2	/Users/ttaro/D...	="
1391	Good	Active	File	14	326	2	/Users/ttaro/D...	="
1391	Good	Active	File	14	326	2	/Users/ttaro/D...	="
10058	Good	Active	File	2	326	2	/Users/ttaro/D...	="
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...	="

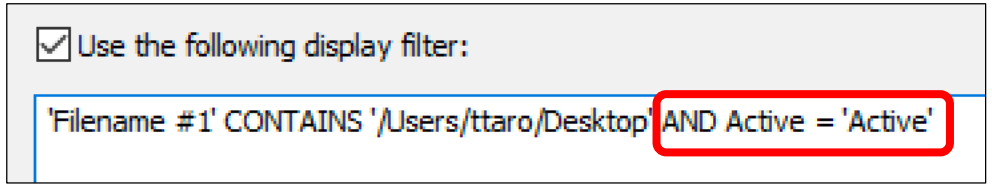
Extra Exercise: MFT Basics 2

Viewing MFT entries and examining a certain folder (9)

- "Active" column indicates whether the file exists 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 that exist in the folder by applying the filter to display rows with the column as "Active".



Extra Exercise: MFT Basics 3

Finding suspicious timestamps

Extra Exercise: MFT Basics 3

Finding suspicious timestamps (1)

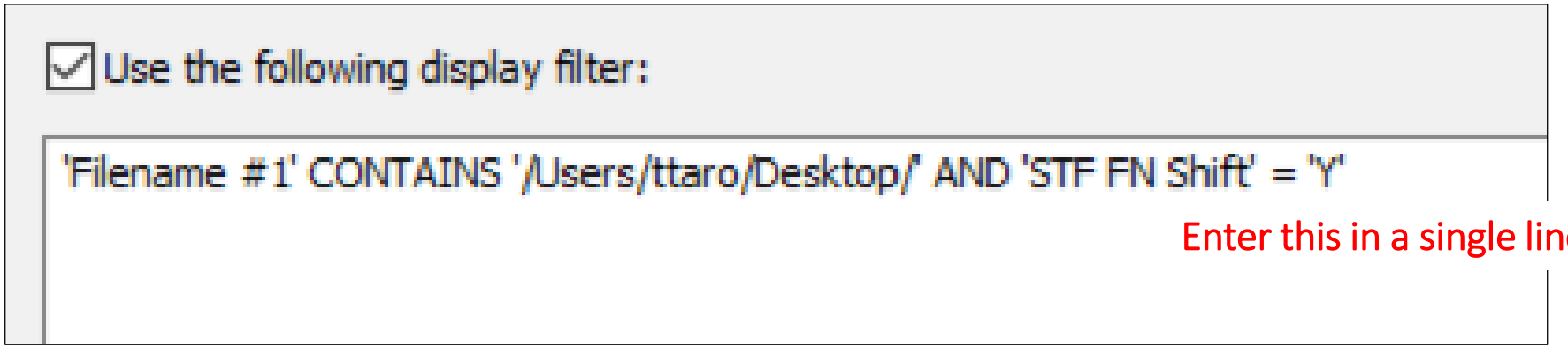
- Goal:
  - To find the files that are suspected of timestamp manipulation in ttaro's Desktop by checking "STF FN Shift" and "uSec Zero" columns.

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

Extra Exercise: MFT Basics 3

Finding suspicious timestamps (2)

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



Enter this in a single line.

You can simply copy the filter commands from the file "E:\Artifacts\other_timeline_analysis\Win7\win7-filter-samples.txt".

Extra Exercise: MFT Basics 3

Finding suspicious timestamps (3)

- You can confirm 148 entries when the filter to display rows with "STF FN Shift" column as 'Y‘ is applied. 147 of them were placed under the "SysinternalsSuite" folder.
- It indicates the possibility that the folder was extracted from an archive file such as zip. And SysinternalsSuite is a famous Windows utility package that is distributed as a zip archive file.

Extra Exercise: MFT Basics 3

Finding suspicious timestamps (4)

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

Click the field name "Filename #1" to sort by this field.

Rec...	Filename #1	std info Creati...
	/Users/ttaro/Desktop/GoodEveningForensic.txt	="2011-01-01 ...
	/Users/ttaro/Desktop/SysinternalsSuite/accesschk.exe	="2016-05-26 ...
	/Users/ttaro/Desktop/SysinternalsSuite/accesschk64	="2016-05-26

Extra Exercise: MFT Basics 3

Finding suspicious timestamps (5)

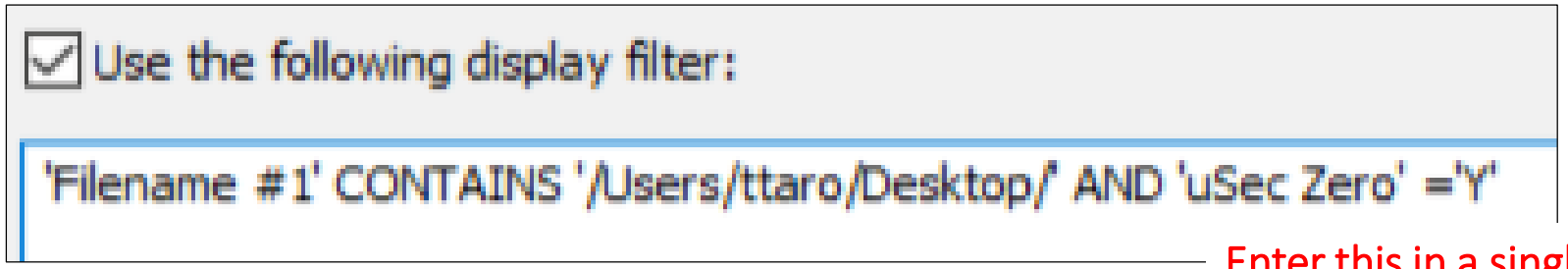
- You can see that the $SI timestamps are older than $FN timestamps. It implies that someone probably manipulated the file's timestamps after the file is created.

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"

Extra Exercise: MFT Basics 3

Finding suspicious timestamps (6)

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



Enter this in a single line.

You can simply copy the filter commands from the file "E:\Artifacts\other_timeline_analysis\Win7\win7-filter-samples.txt".

Extra Exercise: MFT Basics 3

Finding suspicious timestamps (7)

- You can confirm one entry by applying a 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 microsecond 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"

Extra Exercise: MFT Basics 3

Finding suspicious timestamps (8)

- 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 microsecond values, it becomes more difficult to detect.

Extra Exercise: MFT Basics 4

Recovering resident files from $MFT

Extra Exercise: MFT Basics 4

Recovering resident files from $MFT (1)

- Conditions:
  - There are two entries of deleted files that were located on ttaro's Desktop.
  - The names of the files are:
    - GoodMorningForensic.txt
    - GoodAfternoonForensic.txt
- Goal:
  - To recover those contents if it possible.
- Hints:
  - 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!

Extra Exercise: MFT Basics 4

Recovering resident files from $MFT (2)

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

☑ Use the following display filter:

```
Filepath CONTAINS '\Users\ttaro\Desktop' AND Filepath CONTAINS 'GoodMorningForensic.txt' OR Filepath CONTAINS 'GoodAfternoonForensic.txt'
```

Enter this in a single line.

You can simply copy the filter commands from the file "E:\Artifacts\other_timeline_analysis\Win7\win7-filter-samples.txt".

Extra Exercise: MFT Basics 4

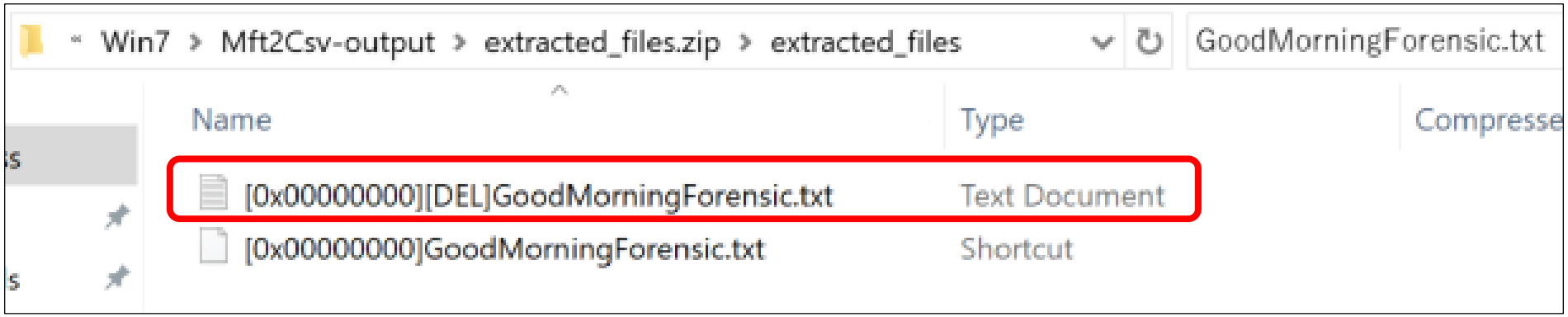
Recovering resident files from $MFT (3)

- A MFT entry list of Mft2Csv output has 124 columns. It is 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.

Extra Exercise: MFT Basics 4

Recovering resident files from $MFT (4)

- 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 all of those files (since it takes long time). If you can find the file you are looking for, extract only the target file.



Extra Exercise: Another $EA malware

Additional Exercise For Finding a Suspicious $EA

Extra Exercise: Another $EA malware

Additional Exercise For Finding a Suspicious $EA (1)

- Conditions:
  - There is a $MFT metadata file of another Windows 7 client. The host is suspected to be infected.
  - We already parsed its $MFT with Mft2Csv. Please check the results located in the folder below and find the suspicious file.
    - E:\Artifacts\other_timeline_analysis\Win7-2\Mft2Csv-output
- Goal:
  - To find files containing non-resident $EA attribute in the disk image.
- Hint:
  - You can find a suspicious file by the same method as $EA related exercise in File System Timeline Analysis section.

Extra Exercise: Another $EA malware

Additional Exercise For Finding a Suspicious $EA (2)

- By opening the file "Mft-Ea-Entries_2018-07-08_23-48-34_mod.csv", we can find that the following two files contain non-resident $EA attribute.

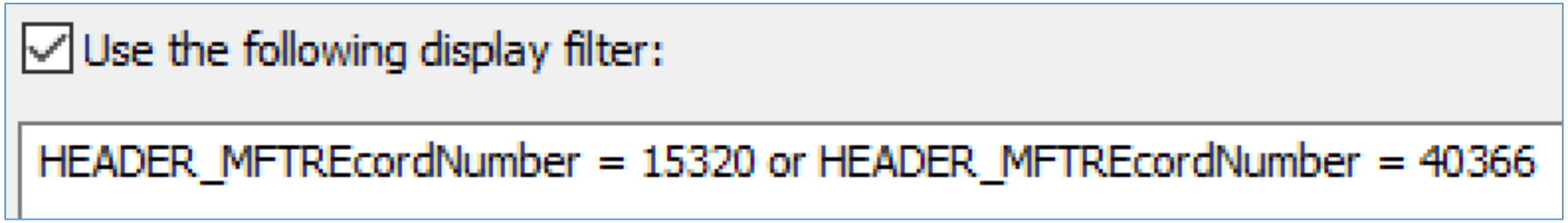
MftRef	MftRefSeqNo	Counter	EaFlags	EaName	EaValueLength	EaValue
15320	9	1	0x00		0	
40366	1	1	0x00		0	

Extra Exercise: Another $EA malware

Additional Exercise For Finding a Suspicious $EA (3)

- You can find the names of files that contain non-resident $EA attribute by matching "MftRef" column in the "Mft-Ea-Entries_2018-07-08_23-48-34_mod.csv" with "HEADER_MFTREcordNumber" column in the "Mft_2018-07-08_23-48-34.csv". You can do it with the following procedure.

1. "Mft_2018-07-08_23-48-34.csv" with csvfileview.exe.
2. Click "Edit Display Filter" button to open the "Display Filter" window.
3. Put the condition in the form like the below figure.
4. Apply the filter.



Extra Exercise: Another $EA malware

Additional Exercise For Finding a Suspicious $EA (4)

- Then, two files should be listed.
- Since we know that the second one is ignorable, the first one is suspicious.

HEADER_MFTREcordNumber	FilePath
15320	:\Users\ttaro\AppData\Local\Music.exe
40366	:\Windows\CSC\v2.0.6

- Actually, it's a banking malware called "ZeuS Panda". It saves important data in its $EA attribute.
- $MFT file parsed in this exercise is extracted from the disk image "E:\Artifacts\other_E01\infected_drive_a.E01". Therefore, you can also extract the malware from the image. Please check it if necessary.

Using "ifind" to get inode number of a certain file from a disk image

A command line sample of ifind.exe

- ifind is a part of The Sleuth Kit.
- This is a usage of ifind.exe for identifying the inode number of a target file in a disk image.

```
ifind.exe -o 206848 -n "Windows/System32/services.exe" E:\Artifacts\other_E01\infected_drive_b.E01
```

  - -o option is to set the offset of the target volume.
  - -n option is to set the file path to the target file.
  - The last argument is to set the target disk image.

- The command above would return inode value of the target file.

```
44477
```

Using "istat" to get information about a certain file from a disk image

A command line sample of istat.exe

- istat.exe is also a part of The Sleuth Kit.
- This is a usage of istat.exe to get information of a target file in a disk image.

```
istat.exe -o 206848 E:\Artifacts\other_E01\infected_drive_b.E01 44477
```

  - -o option is to set the offset of the target volume.
  - The second argument is to set the target disk image.
  - The last argument is to set the inode number of the target file.
- The command above would return several MFT related information about the target file.

```
MFT Entry Header Values:
Entry: 44477        Sequence: 2
>> snip <<
Attributes:
Type: $STANDARD_INFORMATION (16-0)   Name: N/A   Resident   size: 72
Type: $FILE_NAME (48-2)   Name: N/A   Resident   size: 90
>> snip <<
```

Practice Exercise: Malware in $EA 1

Finding suspicious $EA attribute

Practice Exercise: Malware in $EA 1

Finding suspicious $EA attribute (1)

- Conditions:
  - It is NOT related to the scenario 1.
  - The following disk image contains certain $EA related malware.
    - E:\Artifacts\other_E01\infected_drive_b.E01
- Goal:
  - To find out files that has non-resident $EA attributes in the disk image.
- Background:
  - $EA attributes are not used regularly.
  - When a $EA attribute is non-resident, it becomes more suspicious. It's because malware usually needs at least tens of kilobytes in size. Non-resident $EA attribute can take enough size to hide a malware.

Practice Exercise: Malware in $EA 1

Finding suspicious $EA attribute (2)

- In order to parse a MFT for checking $EA, we have to extract a file named $MFT from the disk image.
- Extracted $MFT is saved as the file below.
  - E:\Artifacts\other_timeline_analysis\Win7\artifact\$MFT

- Note: You can extract $MFT with the following command line. It's for your information only. You do not have to do this now.

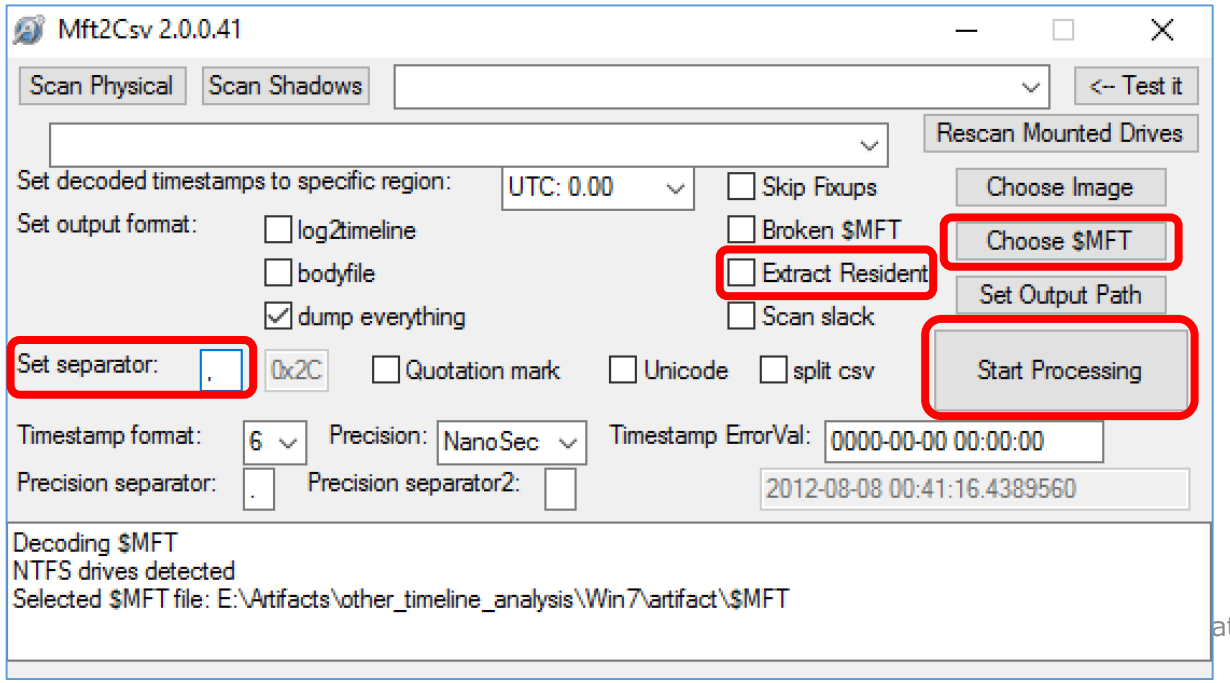
```
icat -o 206848 E:\Artifacts\other_E01\infected_drive_b.E01 0-128 > $MFT
```

- icat is a command to extract a certain file from disk images. It's a part of The Sleuth Kit.
- -o option is to specify the offset of the target volume. We will explain this later.
- 0-128 is an inode number and an attribute type identifier of the target file. Inode number 0 is reserved for $MFT. Type identifier 128 means $DATA attribute. Thus, 0-128 means a actual file body of $MFT.

Practice Exercise: Malware in $EA 1

Finding suspicious $EA attribute (3)

- Next, we should parse $MFT.
- Since it takes a time, we've already parsed it with Mft2Csv.
- Note: the following is an instruction to use Mft2Csv. It's for your information only. You do not have to do this now.



1. Press "Choose $MFT" and select target $MFT file. At this time, set the record size of $MFT. In this case, the value is 1024.
2. Change separator from “|” (pipe) to “,” (comma).
3. If you enable "Extract Resident", it would extract all resident data.
4. Press "Start Processing" and wait for the process to complete.

Practice Exercise: Malware in $EA 1

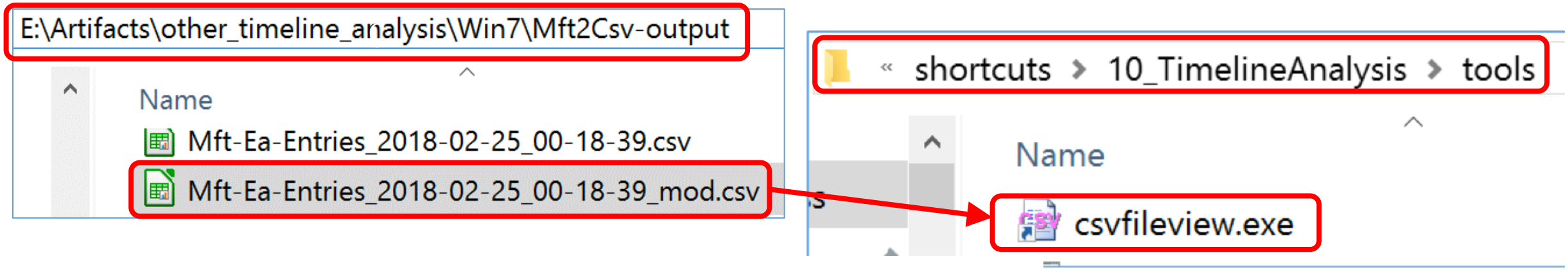
Finding suspicious $EA attribute (4)

- Outputs of Mft2Csv are located under the following folder.
  - E:\Artifacts\other_timeline_analysis\Win7\Mft2Csv-output
- We'll use the following two files in this exercise.
  - Mft_2018-02-25_00-18-39.csv
    - It's the main result. It lists all MFT entries.
  - Mft-Ea-Entries_2018-02-25_00-18-39_mod.csv
    - It contains detailed information of $EA attribute. Only entries containing $EA attribute are listed.
    - Since a separator of the header line in the original file is incorrect, we modified it. The original version is "Mft-Ea-Entries_2018-02-25_00-18-39.csv".

Practice Exercise: Malware in $EA 1

Finding suspicious $EA attribute (5)

- First, we should check "Mft-Ea-Entries_2018-02-25_00-18-39_mod.csv". Let's open this with csvfileview.exe.



Drag the csv file and drop to csvfileview.exe

Practice Exercise: Malware in $EA 1

Finding suspicious $EA attribute (6)

- The 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.
- Next, we should know the names and paths of the files that contain non-resident $EA attributes.

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	

Practice Exercise: Malware in $EA 1

Finding suspicious $EA attribute (7)

- We can find the names and paths of files that contain non-resident $EA attributes by matching "MftRef" column in the "Mft-Ea-Entries_2018-02-25_00-18-39_mod.csv" with "HEADER_MFTRecordNumber" column in the "**Mft_2018-02-25_00-18-39.csv**".

Mft-Ea-Entries_2018-02-25_00-18-39_mod.csv

MftRef	MftRefSeqNo	Counter	EaFlags	EaName	EaValueLength	EaValue
28152	3	1	0x00	001	64	CFF83699740200
40366	1	1	0x00		0	
44477	2	1	0x00	001	64	CFF83699740200
44477	2	1	0x00		0	

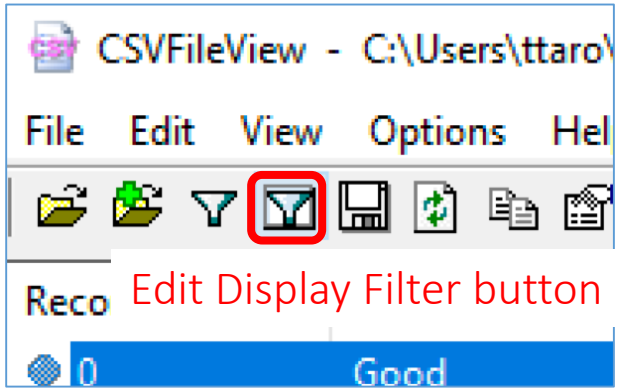
Mft_2018-02-25_00-18-39.csv

HEADER_MFTRecordNumber	H...	He...	F...	FN_...	FN_FileName	FilePath
40364	1	2	25...	1	MI36C5~1.EVT	:\Windows\System32\winevt\
40365	1	2	25...	1	MI4D4C~1.EVT	:\Windows\System32\winevt\
40366	1	2	11...	1	v2.0.6	:\Windows\CSC\v2.0.6
40367	1	1	40...	1	temp	:\Windows\CSC\v2.0.6\temp
40368	1	1	40...	1	pq	:\Windows\CSC\v2.0.6\pq

Practice Exercise: Malware in $EA 1

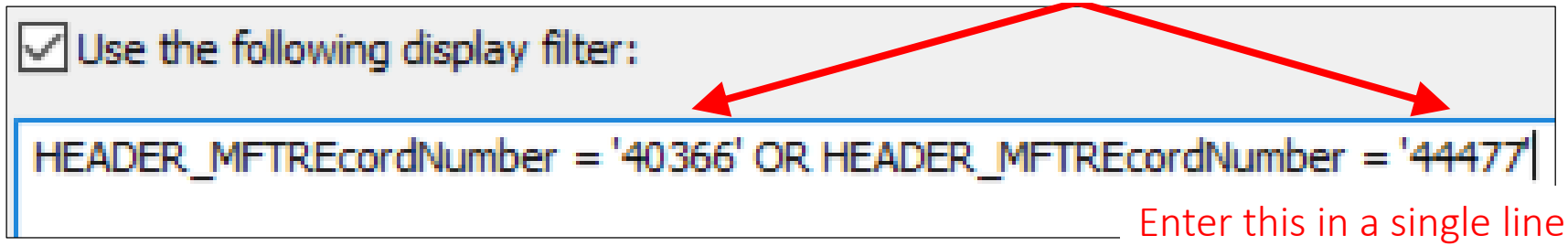
Finding suspicious $EA attribute (8)

- Let's do it with the following procedure.
  1. Open "**Mft_2018-02-25_00-18-39.csv"** with csvfileview.exe.
  2. Press the "Edit Display Filter" button to open the "Display Filter" window.
  3. Put the condition in the form like the figure below.
  4. Apply the filter.



Edit Display Filter button

We've got these numbers from MftRef column in Mft-Ea-Entries_2018-02-25_00-18-39_mod.csv



Enter this in a single line

You can simply copy the filter commands from the file "E:\Artifacts\other_timeline_analysis\Win7\win7-filter-samples.txt".

Practice Exercise: Malware in $EA 1

Finding suspicious $EA attribute (9)

- Finally, we've got the names and paths of files containing non-resident $EA attributes.

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

Practice Exercise: Malware in $EA 1

Finding suspicious $EA attribute (10)

- Some variants of a Trojan Zeroaccess are known for hiding a malicious payload in $EA attribute of "\Windows\System32\services.exe". We will extract the content of $EA with The Sleuth Kit in the next exercise.
- A folder named "\Windows\CSC\v2.0.6" also has non-resident $EA attribute. The folder is related to client-side caching feature. On this environment, we confirmed that the attribute was set on several freshly installed Windows environments. Therefore, we will regard it as benign.

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

Practice Exercise: Malware in $EA 1

Finding suspicious $EA attribute (11)

- Notice:
  - Microsoft uses $EA attributes of system binaries for Secure Booting. Therefore, thousands of system binaries have a $EA attribute on Windows 8 or later.
  - They also use $EA attributes for Windows Subsystem for Linux (WSL). Files and folders under Linux rootfs contain Linux timestamps as $EA attributes.
  - However, all $EA attributes of those system binaries and WSL files and folders are "resident". They contain a short data less than about 100 bytes.
  - Thus, files containing non-resident $EA attributes should be considered as suspicious.

Practice Exercise: Malware in $EA 2

Extracting Suspicious $EA attribute

Practice Exercise: Malware in $EA 2

Extracting Suspicious $EA attribute (1)

- Conditions:
  - It's NOT related to the scenario 1. However it’s a continuation of the previous exercise that is to find non-resident $EA attribute.
  - Thus, we are investigating the same disk image as we investigated in the previous exercise.
    - E:\Artifacts\other_E01\infected_drive_b.E01

- Goal:
  - To extract a content of $EA attribute that we found in the previous exercise.

- Note:
  - Details of $EA related Zeroaccess valiant and the procedure to extract the $EA attribute are described in the following web page. This is a good example for understanding $EA related malware.
    - http://journeyintoir.blogspot.com/2012/12/extracting-zeroaccess-from-ntfs.html

Practice Exercise: Malware in $EA 2

Extracting Suspicious $EA attribute (2)

- We use The Sleuth Kit to extract the data.
- First, specify the offset of the target partition in the disk image by the following command.

```
mmls.exe E:\Artifacts\other_E01\infected_drive_b.E01
```

```
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

      Slot      Start        End          Length       Description
000:  Meta      0000000000   0000000000   0000000001   Primary Table (#0)
001:  -------   0000000000   0000002047   0000002048   Unallocated
002:  000:000   0000002048   0000206847   0000204800   NTFS / exFAT (0x07)
003:  000:001   0000206848   0031455231   0031248384   NTFS / exFAT (0x07)
004:  -------   0031455232   0031457279   0000002048   Unallocated
```

There are two NTFS partitions in the disk. The first one (002) is very small (206847 sectors = about 100MB). It seems to be the "system reserved" partition, which is used to boot the system. The second one (003) has about 16GB in size. It is the target partition.

This is the offset of the 2nd NTFS volume.

Practice Exercise: Malware in $EA 2

Extracting Suspicious $EA attribute (3)

- Then, extract the contents by the following command.

```
icat.exe -o 206848 E:\Artifacts\other_E01\infected_drive_b.E01 44477-224 > ea_attr_services_exe.bin
```

Set the offset of the target partition.

44477-224 is the inode number and attribute type identifier of the target file. The inode number 44477 is HEADER_MFTREcordNumber of the target file that we found in the previous exercise. Type identifier 224 means $EA attribute type.

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

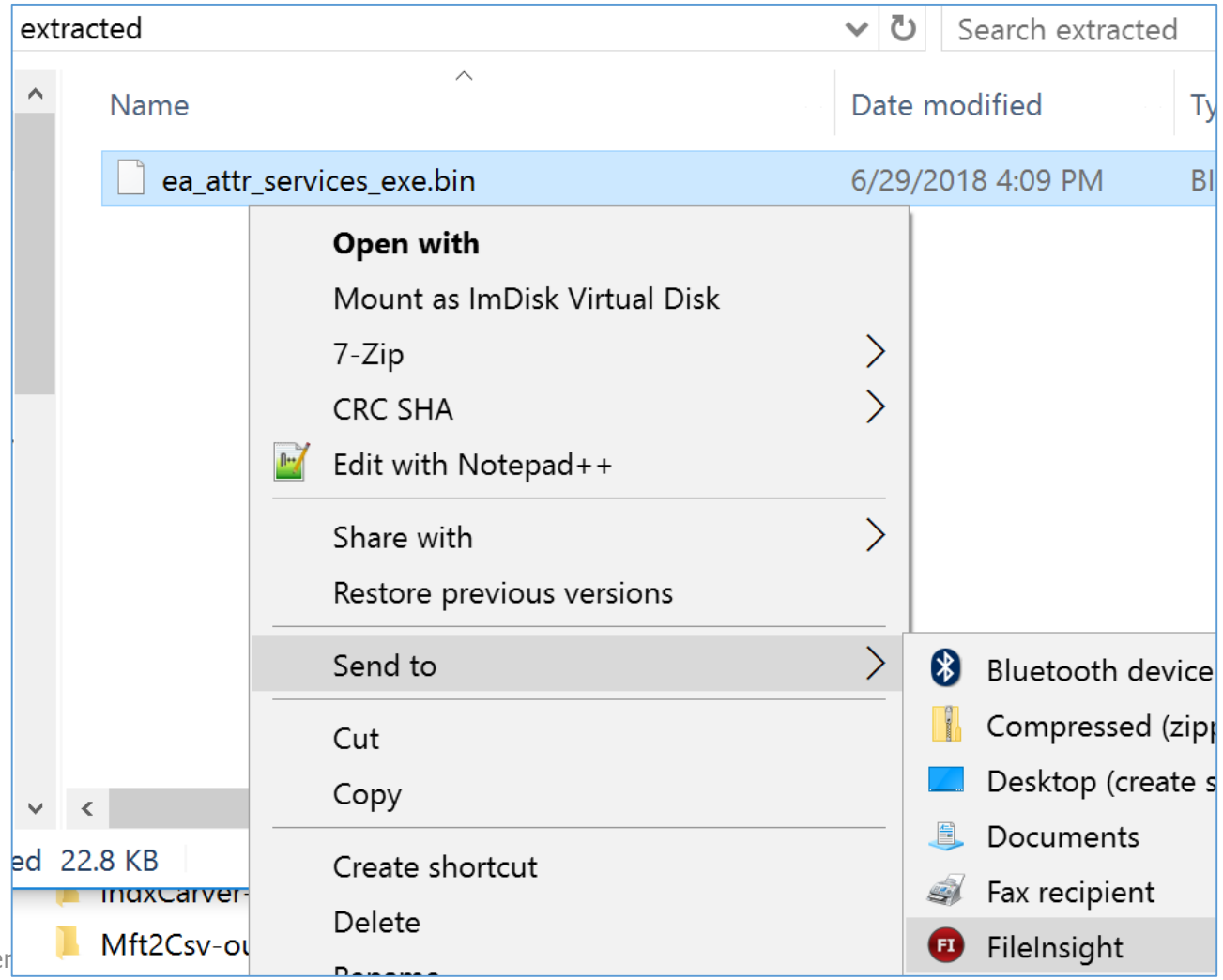
You can confirm attribute type identifier by checking the following web page.

- https://flatcap.org/linux-ntfs/ntfs/attributes/index.html

Practice Exercise: Malware in $EA 2

Extracting Suspicious $EA attribute (4)

- Then, we can view the content with hex editor such as FileInsight.



Practice Exercise: Malware in $EA 2

Extracting Suspicious $EA attribute (5)

- We can find an MZ header, a DOS stub, and a PE header around the offset 0x690 of the data. In other words, the extracted $EA data seems contains an executable image!

```
ea_attr_servic...
00000680  85 F6 75 F1 5B 5E C3 CC CC CC CC CC 90 90 90 E8  ..u.[^..........
00000695  00 52 00 00 4D 5A 90 00 03 00 00 00 04 00 00 00  .R..MZ..........
000006A0  FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00  ............@...
000006B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000006C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000006D0  E8 00 00 00 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C  ............!..L
000006E0  CD 21 54 68 69 73 20 70 72 6F 67 72 61 6D 20 63  .!This program c
000006F0  61 6E 6E 6F 74 20 62 65 20 72 75 6E 20 69 6E 20  annot be run in
00000700  44 4F 53 20 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00  DOS mode....$...
00000710  00 00 00 00 7D 27 E7 4D 39 46 89 1E 39 46 89 1E  ....}'.M9F..9F..
00000720  39 46 89 1E 39 46 88 1E 58 46 89 1E FA 49 D4 1E  9F..9F..XF...I..
00000730  32 46 89 1E FA 49 D6 1E 3B 46 89 1E FA 49 86 1E  2F...I..;F...I..
00000740  3A 46 89 1E 1E 80 F4 1E 3B 46 89 1E 27 14 1C 1E  :F......;F..'...
00000750  38 46 89 1E 30 3E 03 1E 35 46 89 1E 30 3E 18 1E  8F..0>..5F..0>..
00000760  38 46 89 1E 52 69 63 68 39 46 89 1E 00 00 00 00  8F..Rich9F.....
00000770  00 00 00 00 00 00 00 00 00 00 00 00 50 45 00 00  ............PE..
00000780  4C 01 04 00 E1 17 0C 50 00 00 00 00 00 00 00 00  L......P........
```

Practice Exercise: Malware in $EA 2

Extracting Suspicious $EA attribute (6)

- By deleting junk data placed before the MZ header, you can open it with a disassembler as an executable image.
- Then, you can analyze the image.

ea_attr_services_exe.bin*

```
00000000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..............
00000010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ........@.......
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00 00 00 00 00 E8 00 00 00  ................
00000040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ........!..L.!Th
00000050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
00000070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000080  7D 27 E7 4D 39 46 89 1E 39 46 89 1E 39 46 89 1E  }'.M9F..9F..9F..
00000090  39 46 88 1E 58 46 89 1E FA 49 D4 1E 32 46 89 1E  9F..XF...I..2F..
000000A0  FA 49 D6 1E 3B 46 89 1E FA 49 86 1E 3A 46 89 1E  .I..;F...I..:F..
000000B0  1E 80 F4 1E 3B 46 89 1E 27 14 1C 1E 38 46 89 1E  ....;F..'...8F..
000000C0  30 3E 03 1E 35 46 89 1E 30 3E 18 1E 38 46 89 1E  0>..5F..0>..8F..
000000D0  52 69 63 68 39 46 89 1E 00 00 00 00 00 00 00 00  Rich9F..........
000000E0  00 00 00 00 00 00 00 00 50 45 00 00 4C 01 04 00  ........PE..L...
000000F0  E1 17 0C 50 00 00 00 00 00 00 00 00 E0 00 02 21  ...P...........!
00000100  0B 01 09 00 00 34 00 00 00 34 00 00 00 00 00 00  .....4...4......
00000110  75 18 00 00 00 10 00 00 00 50 00 00 00 00 67 45  u........P....gE
00000120  00 10 00 00 00 02 00 00 05 00 00 00 00 00 00 00  ................
00000130  05 00 00 00 00 00 00 00 00 A0 00 00 00 04 00 00  ................
00000140  00 00 00 00 02 00 00 00 00 00 10 00 00 10 00 00  ................
```

IDA View-A | Hex View-1 | Structures | Enums

```
; BOOL __stdcall DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReas
public DllEntryPoint
DllEntryPoint proc near

hinstDLL= dword ptr  4
fdwReason= dword ptr  8
lpReserved= dword ptr  0Ch

cmp     [esp+fdwReason], 0FFFFFFFEh
jnz     short loc_456718E5
```

```
push    esi
push    edi
```

Practice Exercise: Malware in $EA 3

Revealing the infection process with a $UsnJrnl timeline

Practice Exercise: Malware in $EA 3

Revealing the infection process with a $UsnJrnl timeline (1)

- Conditions:
  - It is NOT related to the scenario 1. However it's a continuation of the previous exercise that is to extract non-resident $EA attribute
  - The system could be infected with malware.
  - A $EA attribute of \Windows\System32\services.exe could have been injected a malicious payload by the malware.
- Goal:
  - To find out which file caused the infection by examining the output of NTFS Log Tracker.
    - The output is located in the following folder.
    - E:\Artifacts\other_timeline_analysis\Win7\ntfs-log-tracker-output\

Practice Exercise: Malware in $EA 3

Revealing the infection process with a $UsnJrnl timeline (2)

- Open this file with CSVFileView.
  - "E:\Artifacts\other_timeline_analysis\Win7\ntfs-log-tracker-output\UsnJrnl.csv"
- This is the results of parsing $UsnJrnl:$J.
- In this case, there are few entries in "Logfile.csv" that is the results of parsing $Logfile. Thus, we check UsnJrnl.csv only.

- Note:
  - These CSV files are converted from a SQLite DB that are created by NTFS Log Tracker.
  - You can find a simple usage guide of NTFS Log Tracker in Appendix.

Practice Exercise: Malware in $EA 3

Revealing the infection process with a $UsnJrnl timeline (3)

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

Practice Exercise: Malware in $EA 3

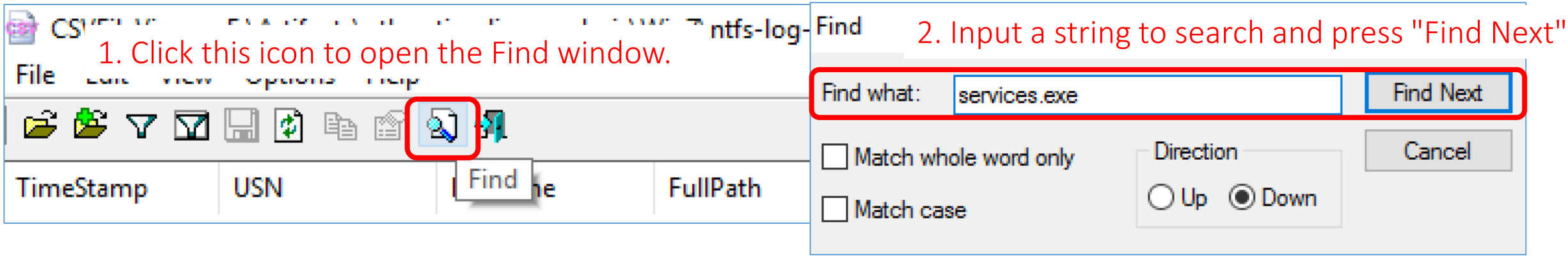
Revealing the infection process with a $UsnJrnl timeline (4)

- Let's find the file that caused the initial infection and reveal the sequence of events.
- Hints:
  - According to the result of the previous exercise, we are suspecting that the file "\Windows\System32\services.exe" is infected with malware.
  - In order to check program execution history around the infection, we should examine timestamps of prefetch files located in "\Windows\Prefetch". Creation and modification times of prefetch files show the first and last execution times of the related programs.

Practice Exercise: Malware in $EA 3

Revealing the infection process with a $UsnJrnl timeline (5)

- First, we should search entries showing $EA modification of the file "\Windows\System32\services.exe". Then, check other entries around them.



TimeStamp	USN	Filename	FullPath	EventIn
2018-02-25 00:01:40	22451080	services.exe	\Windows\System32\services.exe	File_Cr
2018-02-25 00:01:40	22451168	services.exe	\Windows\System32\services.exe	File_Cr
2018-02-25 00:01:40	22451256	services.exe	\Windows\System32\services.exe	File_Cr

These entries show $EA modification of the target file.

TimeStamp	Filename	FullPath	EventInfo
2018-02-25 00:01:40	services.exe	\Windows\System32\services.exe	File_Created/ Extended_Attr_Changed
2018-02-25 00:01:40	services.exe	\Windows\System32\services.exe	File_Created/ File_Added/ Extended_Attr_...
2018-02-25 00:01:40	services.exe	\Windows\System32\services.exe	File_Created/ Attr_Changed/ File_Added/ ...
2018-02-25 00:01:40	services.exe	\Windows\System32\services.exe	File_Created/ Attr_Changed/ File_Added/ ...
2018-02-25 00:01:40		\Windows\System32\	File_Renamed_New/ File_Closed
2018-02-25 00:01:41	@	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	Data_Overwritten
2018-02-25 00:01:42	35a0af39-a93f-48a4-a6eb-...	\Windows\System32\LogFiles\Scm\35a0af39-a93f-48a...	File_Truncated
2018-02-25 00:01:42	35a0af39-a93f-48a4-		_Truncated
2018-02-25 00:01:42	35a0af39-a93f-48a4-		_Truncated/ File_Closed
2018-02-25 00:01:44	DLLHOST.EXE-766398D2.pf	\Windows\Prefetch\DLLHOST.EXE-766398D2.pf	File_Truncated
2018-02-25 00:01:44	DLLHOST.EXE-766398D2.pf	\Windows\Prefetch\DLLHOST.EXE-766398D2.pf	File_Added/ File_Truncated
2018-02-25 00:01:44			File_Added/ File_Truncated/ File_Closed
2018-02-25 00:01:44			File_Truncated
2018-02-25 00:01:44			File_Added/ File_Truncated
2018-02-25 00:01:44	DLLHOST.EXE-766398D2.pf	\Windows\Prefetch\DLLHOST.EXE-766398D2.pf	File_Added/ File_Truncated/ File_Closed
2018-02-25 00:01:47	Macromed	\Windows\System32\Macromed	File_Created
2018-02-25 00:01:47	Macromed	\Windows\System32\Macromed	File_Created/ File_Closed
2018-02-25 00:01:47	Flash	\Windows\System32\Macromed\Flash	File_Created
2018-02-25 00:01:47	Flash	\Windows\System32\Macromed\Flash	File_Created/ File_Closed
2018-02-25 00:01:47	FlashInstall.log	\Windows\System32\Macromed\Flash\FlashInstall.log	File_Created
2018-02-25 00:01:47	FlashInstall.log	\Windows\System32\Macromed\Flash\FlashInstall.log	File_Created/ File_Added
2018-02-25 00:01:47	FlashInstall.log	\Windows\System32\Macromed\Flash\FlashInstall.log	File_Created/ File_Added/ File_Closed
2018-02-25 00:01:47	INSTALLFLASHPLAYER.EX...	\Windows\Prefetch\INSTALLFLASHPLAYER.EXE-931D6...	File_Created
2018-02-25 00:01:47	INSTALLFLASHPLAYER.EX...	\Windows\Prefetch\INSTALLFLASHPLAYER.EXE-931D6...	File_Created/ File_Added
2018-02-25 00:01:47	INSTALLFLASHPLAYER.EX...	\Windows\Prefetch\INSTALLFLASHPLAYER.EXE-931D6...	File_Created/ File_Added/ File_Closed

An suspicious entry containing a strange short file name.

Entries look like Flash installer activities. These are unnatural if the user didn't execute installation at that time.

TimeStamp	Filename	FullPath	EventInfo
2018-02-25 00:01:40	wfpdiag.etl	\Windows\System32\wfp\wfpdiag.etl	Data_Overwritten
2018-02-25 00:01:40	wfpdiag.etl	\Windows\System32\wfp\wfpdiag.etl	Data_Overwritten/ File_Closed
2018-02-25 00:01:40	{92d17ead-3bff-803f-f337...	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	File_Created
2018-02-25 00:01:40	U	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	File_Created/ Extended_Attr_Changed
2018-02-25 00:01:40	U	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	File_Created/ Attr_Changed/ Extended_Att...
2018-02-25 00:01:40	U	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	File_Created/ Attr_Changed/ Extended_Att...
2018-02-25 00:01:40	@	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	File_Created
2018-02-25 00:01:40	@	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	File_Created/ File_Added
2018-02-25 00:01:40	@	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	File_Created/ Attr_Changed/ File_Added
2018-02-25 00:01:40	@	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	File_Created/ Attr_Changed/ File_Added/ ...
2018-02-25 00:01:40	L	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	File_Created
2018-02-25 00:01:40	L	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	File_Created/ Attr_Changed
2018-02-25 00:01:40	L	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	File_Created/ Attr_Changed/ File_Closed
2018-02-25 00:01:40	n	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	File_Created
2018-02-25 00:01:40	n	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	File_Created/ File_Added
2018-02-25 00:01:40	n	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	File_Created/ Attr_Changed/ File_Added
2018-02-25 00:01:40	n	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	File_Created/ Attr_Changed/ File_Added/ ...
2018-02-25 00:01:40	{92d17ead-3bff-803f-f337...	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	File_Created/ Attr_Changed
2018-02-25 00:01:40	{92d17ead-3bff-803f-f337...	\Windows\Installer\{92d17ead-3bff-803f-f337-0d004dd...	File_Created/ Attr_Changed/ File_Closed
2018-02-25 00:01:40	services.e	[illegible]	[illegible]t_Changed
2018-02-25 00:01:40	services.e	[illegible]	[illegible]t_Changed/ File_Closed
2018-02-25 00:01:40	services.exe	\Windows\winsxs\x86_microsoft-windows-s..s-servicec...	Access_Right_Changed
2018-02-25 00:01:40	services.exe	\Windows\winsxs\x86_microsoft-windows-s..s-servicec...	Access_Right_Changed/ File_Closed
2018-02-25 00:01:40	services.exe	\Windows\System32\services.exe	File_Renamed_Old
2018-02-25 00:01:40		\Windows\System32\	File_Renamed_New
2018-02-25 00:01:40	services.exe	\Windows\System32\services.exe	File_Created/ Extended_Attr_Changed
2018-02-25 00:01:40	services.exe	\Windows\System32\services.exe	File_Created/ File_Added/ Extended_Attr_...
2018-02-25 00:01:40	services.exe	\Windows\System32\services.exe	File_Created/ Attr_Changed/ File_Added/ ...

These entries also contain strange short file names.

Practice Exercise: Malware in $EA 3

Revealing the infection process with a $UsnJrnl timeline (8)

At the same time, a file named ea.exe on user's Desktop was deleted.
Sometimes malware deletes itself after its infection.

TimeStamp	Filename	FullPath	EventInfo
2018-02-25 00:01:40	ea.exe	\Users\ttaro\Desktop\ea.exe	File_Closed/ File_Deleted
2018-02-25 00:01:40	CMD.EXE-4A81B364.pf	\Windows\Prefetch\CMD.EXE-4A81B364.pf	File_Truncated
2018-02-25 00:01:40	CMD.EXE-4A81B364.pf	\Windows\Prefetch\CMD.EXE-4A81B364.pf	File_Added/ File_Truncated
2018-02-25 00:01:40	CMD.EXE-4A81B364.pf	\Windows\Prefetch\CMD.EXE-4A81B364.pf	File_Added/ File_Truncated/ File_Closed
2018-02-25 00:01:40	CONHOST.EXE-1F3E9D7E....	\Windows\Prefetch\CONHOST.EXE-1F3E9D7E.pf	File_Truncated
2018-02-25 00:01:40	CONHOST.EXE-1F3E9D7E....	\Windows\Prefetch\CONHOST.EXE-1F3E9D7E.pf	File_Added/ File_Truncated
2018-02-25 00:01:40	CONHOST.EXE-1F3E9D7E....	\Windows\Prefetch\CONHOST.EXE-1F3E9D7E.pf	File_Added/ File_Truncated/ File_Closed
2018-02-25 00:01:40	wfpdiag.etl	\Windows\System32\wfp\wfpdiag.etl	Data_Overwritten
2018-02-25 00:01:40	wfpdiag.etl	\Windows\System32\wfp\wfpdiag.etl	Data_Overwritten/ File_Closed
2018-(			
2018-(			r_Changed

These entries imply execution of the command prompt.
Sometimes malware launch it to do something such as loading program, modifying files and so on.

Practice Exercise: Malware in $EA 3

Revealing the infection process with a $UsnJrnl timeline (9)

TimeStamp	Filename	FullPath	EventInfo
2018-02-25 00:01:39	CONSENT.EXE-531BD9EA....	\Windows\Prefetch\CONSENT.EXE-531BD9EA.pf	File_Truncated
2018-02-25 00:01:39	CONSENT.EXE-531BD9EA....	\Windows\Prefetch\CONSENT.EXE-531BD9EA.pf	File_Added/ File_Truncated
2018-02-25 00:01:39	CONSENT.EXE-531BD9EA....	\Windows\Prefetch\CONSENT.EXE-531BD9EA.pf	File_Added/ File_Truncated/ File_Closed
2018-02-25 00:01:39	EA.EXE-67BB4897.pf	\Windows\Prefetch\EA.EXE-67BB4897.pf	File_Created
2018-02-25 00:01:39	EA.EXE-67BB4897.pf	\Windows\Prefetch\EA.EXE-67BB4897.pf	File_Created/ File_Added
2018-02-25 00:01:39	EA.EXE-67BB4897.pf	\Windows\Prefetch\EA.EXE-67BB4897.pf	File_Created/ File_Added/ File_Closed

- The entries above imply execution of consent.exe. consent.exe is related to UAC. Thus, it seemed that some program required privileges at that time. We could guess the request was to modify files under system folder such as services.exe.
- The bottom entries imply execution of file named "ea.exe". It could be the file we mentioned before. It was located on the Desktop and was deleted after a second.

TimeStamp	Filename	FullPath	EventInfo
2018-02-25 00:01:37	U	\Users\ttaro\AppData\Local\{92d17ead-3bff-803f-f337-...	File_Created/ Extended_Attr_Changed
2018-02-25 00:01:37	U	\Users\ttaro\AppData\Local\{92d17ead-3bff-803f-f337-...	File_Created/ Attr_Changed/ Extended_Att...
2018-02-25 00:01:37	U	\Users\ttaro\AppData\Local\{92d17ead-3bff-803f-f337-...	File_Created/ Attr_Changed/ Extended_Att...
2018-02-25 00:01:37	@	\Users\ttaro\AppData\Local\{92d17ead-3bff-803f-f337-...	File_Created
2018-02-25 00:01:37	@	\Users\ttaro\AppData\Local\{92d17ead-3bff-803f-f337-...	File_Created/ File_Added
2018-02-25 00:01:37	@	\Users\ttaro\AppData\Local\{92d17ead-3bff-803f-f337-...	File_Created/ Attr_Changed/ File_Added/ ...
2018-02-25 00:01:37	@	\Users\ttaro\AppData\Local\{92d17ead-3bff-803f-f337-...	File_Created/ Attr_Changed/ File_Added/ ...
2018-02-25 00:01:37	L	\Users\ttaro\AppData\Local\{92d17ead-3bff-803f-f337-...	File_Created
2018-02-25 00:01:37	L	\Users\ttaro\AppData\Local\{92d17ead-3bff-803f-f337-...	File_Created/ Attr_Changed/ Content_Ind...
2018-02-25 00:01:37	L	\Users\ttaro\AppData\Local\{92d17ead-3bff-803f-f337-...	File_Created/ Attr_Changed/ Content_Ind...
2018-02-25 00:01:37	n	\Users\ttaro\AppData\Local\{92d17ead-3bff-803f-f337-...	File_Created
2018-02-25 00:01:37	n	\Users\ttaro\AppData\Local\{92d17ead-3bff-803f-f337-...	File_Created/ File_Added
2018-02-25 00:01:37	n	\Users\ttaro\AppData\Local\{92d17ead-3bff-803f-f337-...	File_Created/ Attr_Changed/ File_Added/ ...
2018-02-25 00:01:37	n	\Users\ttaro\AppData\Local\{92d17ead-3bff-803f-f337-...	File_Created/ Attr_Changed/ File_Added/ ...
2018-02-25 00:01:37	{92d17ead-3bff-803f-f337...	\Users\ttaro\AppData\Local\{92d17ead-3bff-803f-f337-...	File_Created/ Attr_Changed/ Content_Ind...
2018-02-25 00:01:37	{92d17ead-3bff-803f-f337...	\Users\ttaro\AppData\Local\{92d17ead-3bff-803f-f337-...	File_Created/ Attr_Changed/ Content_Ind...
2018-02-25 00:01:37	InstallFlashPlayer.exe	\Users\ttaro\AppData\Local\Temp\InstallFlashPlayer.exe	File_Created
2018-02-25 00:01:37	InstallFlashPlayer.exe	\Users\ttaro\AppData\Local\Temp\InstallFlashPlayer.exe	File_Created/ File_Added
2018-02-25 00:01:37	InstallFlashPlayer.exe	\Users\ttaro\AppData\Local\Temp\InstallFlashPlayer.exe	File_Created/ File_Added/ File_Closed

- There are other entries that look like Flash installer's activities. These entries logged 3 seconds before the similar ones we mentioned before.
- In addition, these entries are file creation events under a user folder. On the other hand, the similar ones are under the system folder. It could be because, the malicious program get privilege after these events.

TimeStamp	Filename	FullPath	EventInfo
2018-02-25 00:00:55	ea.zip	\Users\ttaro\Desktop\ea.zip	Object_ID_Changed
2018-02-25 00:00:55	ea.zip	\Users\ttaro\Desktop\ea.zip	Object_ID_Changed/ File_Closed
2018-02-25 00:00:55	1b4dd67f29cb1962.autom...	\Users\ttaro\AppData\Roaming\Microsoft\Windows\R...	File_Added
2018-02-25 00:00:55	1b4dd67f29cb1962.autom...	\Users\ttaro\AppData\Roaming\Microsoft\Windows\R...	Attr_Changed/ File_Added/ Data_Overwrit...
2018-02-25 00:00:55	ea.zip.lnk	\Users\ttaro\AppData\Roaming\Microsoft\Windows\R...	File_Created
2018-02-25 00:00:55	ea.zip.lnk	\Users\ttaro\AppData\Roaming\Microsoft\Windows\R...	File_Created/ File_Added
2018-02-25 00:00:55	ea.zip.lnk	\Users\ttaro\AppData\Roaming\Microsoft\Windows\R...	File_Created/ File_Added/ File_Closed
2018-02-25 00:00:57	1b4dd67f29cb1962.autom...	\Users\ttaro\AppData\Roaming\Microsoft\Windows\R...	Data_Overwritten
2018-02-25 00:00:57	1b4dd67f29cb1962.autom...	\Users\ttaro\AppData\Roaming\Microsoft\Windows\R...	Data_Overwritten/ File_Closed
2018-02-25 00:01:01	ea.exe	\Users\ttaro\Desktop\ea.exe	File_Created
2018-02-25 00:01:01	ea.exe	\Users\ttaro\Desktop\ea.exe	File_Created/ File_Added
2018-02-25 00:01:01	ea.exe	\Users\ttaro\Desktop\ea.exe	File_Created/ File_Added/ File_Closed
2018-02-25 00:01:01	ea.exe	\Users\ttaro\Desktop\ea.exe	Attr_Changed
2018-02-25 00:01:01	ea.exe	\Users\ttaro\Desktop\ea.exe	Attr_Changed/ File_Closed
2018-02-25 00:01:01	ea.exe	\Users\ttaro\Desktop\ea.exe	Attr_Changed
2018-02-25 00:01:01	ea.exe	\Users\ttaro\Desktop\ea.exe	Attr_Changed/ File_Closed

- The top and middle events imply that an archive file named "ea.zip" on the Desktop was actually opened.
- The bottom events are creation events of a file named "ea.exe" on the Desktop.

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.	The file seems to be extracted from zip file above as it has the same name with the zip file.
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 execute command lines using 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 from its execution time. Malware sometimes launch conhost.exe while executing cmd.exe.
2018-02-25 00:01:40	\Windows\System32\services.exe	$EA attr was changed.	$EA attribute of the target file was modified.

Practice Exercise: Malware in $EA 3
Revealing the infection process with a $UsnJrnl timeline (13)

- 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 that they were 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.

- In conclusion, we can assume that ea.exe is related to the infection of services.exe.

How to install and setup Elasticsearch/Kibana

How to install and setup Elasticsearch/Kibana

1. Install the latest Java Runtime Environment.
2. Download ElasticSearch from the following URL.
   - https://www.elastic.co/downloads/elasticsearch
3. Unzip ElasticSearch to any place on your computer.
   - You can launch ElasticSearch by running bat file "bin\elasticsearch.bat".
4. Download Kibana from following URL.
   - https://www.elastic.co/downloads/kibana
5. Unzip Kibana to any place on your computer.
   - You can launch Kibana by running bat file "bin\kibana.bat".
6. Install embulk by the following command.

```
PowerShell -Command "& {Invoke-WebRequest http://dl.embulk.org/embulk-latest.jar -OutFile embulk.bat}"
```

7. Install embulk plug-in for ElasticSearch by the following command.

```
embulk gem install embulk-output-elasticsearch
```

8. OK, you are ready to use ElasticSearch/Kibana and embulk.

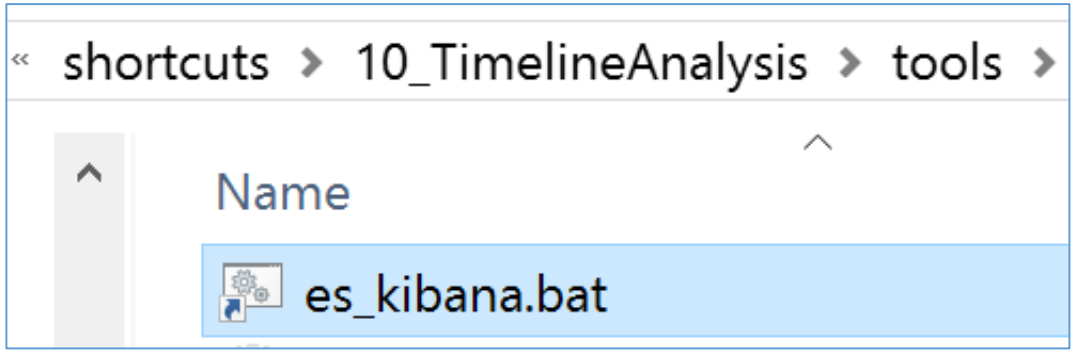
How to import csv data into Elasticsearch

How to import csv data into Elasticsearch (1)

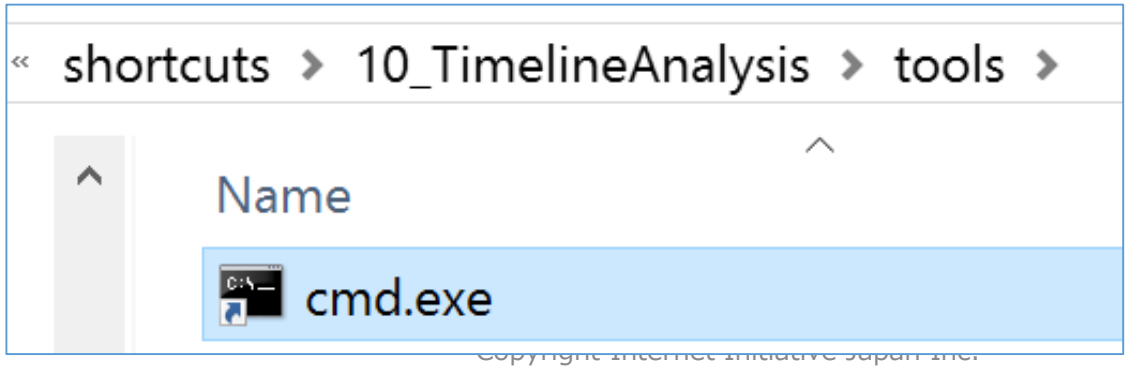
- These slides show the instruction to import csv data into Elasticsearch.
- We did the same method to prepare "Extra Exercise: Elasticsearch" in File System Timeline Analysis section
- All parameters used in this section are for preparing the exercises.
- When you use this instruction in your own case, please tweak the parameters to meet your purpose.

How to import csv data into Elasticsearch (2)

- Double-click the bat file below to launch ElasticSearch and Kibana.



- Launch the cmd.exe by double-clicking the icon.



How to import csv data into Elasticsearch (3)

- We'll use Embulk to import csv data into Elasticsearch. Embulk supports data transfer between various storages, databases, NoSQL and cloud services.
- Generate configuration file for loading the CSV file "UsnJrnl.csv" into ElasticSearch by executing the following command in the folder "elasticsearch".

```
embulk.bat guess seed-ntfs-log-tracker.yml -o config-ntfs-log-tracker.yml
```

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

This is the name of the file to generate.

```yaml
in:
  type: file
  path_prefix: "E:/Artifacts/other_timeline_analysis/Win10/ntfs-log-tracker-output/UsnJrnl.csv"
out:
  type: elasticsearch
  index: ntfslogtracker-win10
  index_type: ntfslogtracker
  nodes:
```

How to import csv data into

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

default_timezone: 'Asia/Tokyo'

```yaml
in:
  type: file
  path_prefix: E:/Artifacts/other_timeline_
  tracker-output/UsnJrnl.csv
  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: FullPath, type: string}
      - {name: EventInfo, type: string}
      - {name: SourceInfo, type: string}
      - {name: File Attribute, type: string}
```

Notepad++ automatically insert **tabs** as indents. However, the tabs must be replaced with **spaces**.

How to import csv data into Elasticsearch (5)

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

```
embulk.bat preview config-ntfs-log-tracker.yml
```

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

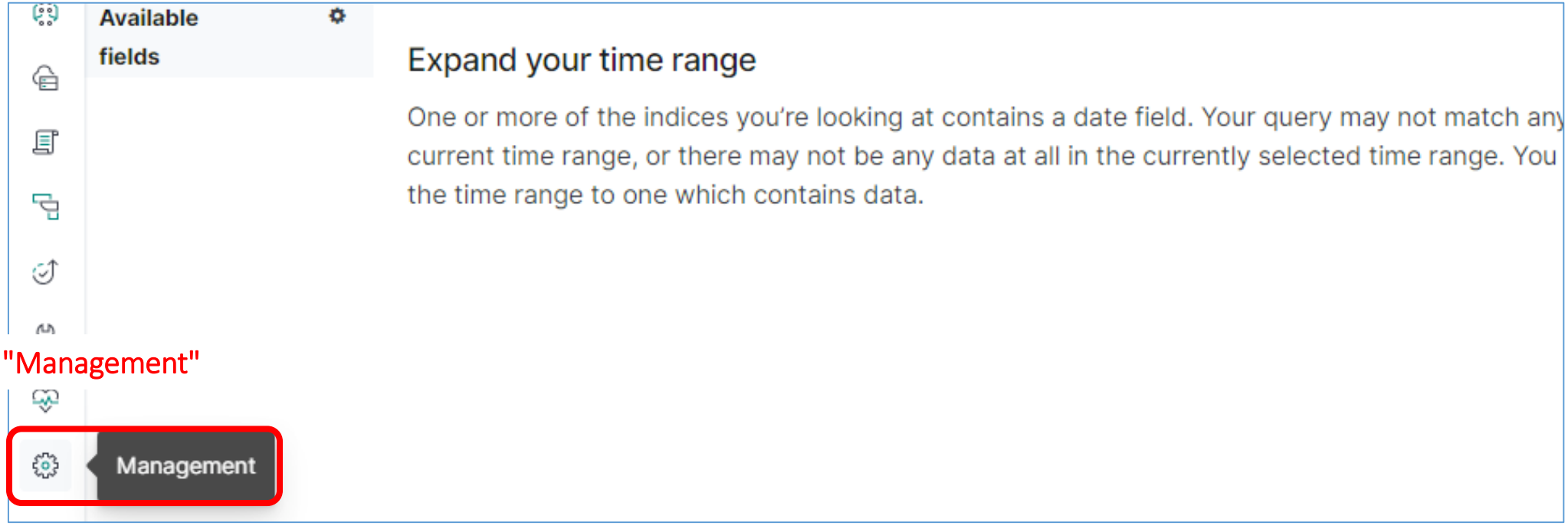
```
embulk.bat run config-ntfs-log-tracker.yml -c diff.yml
```

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

- Finally, open the following URL with a web browser (e.g. Chrome).
  - http://localhost:5601/

How to Import Proxy Log into Elasticsearch (6)

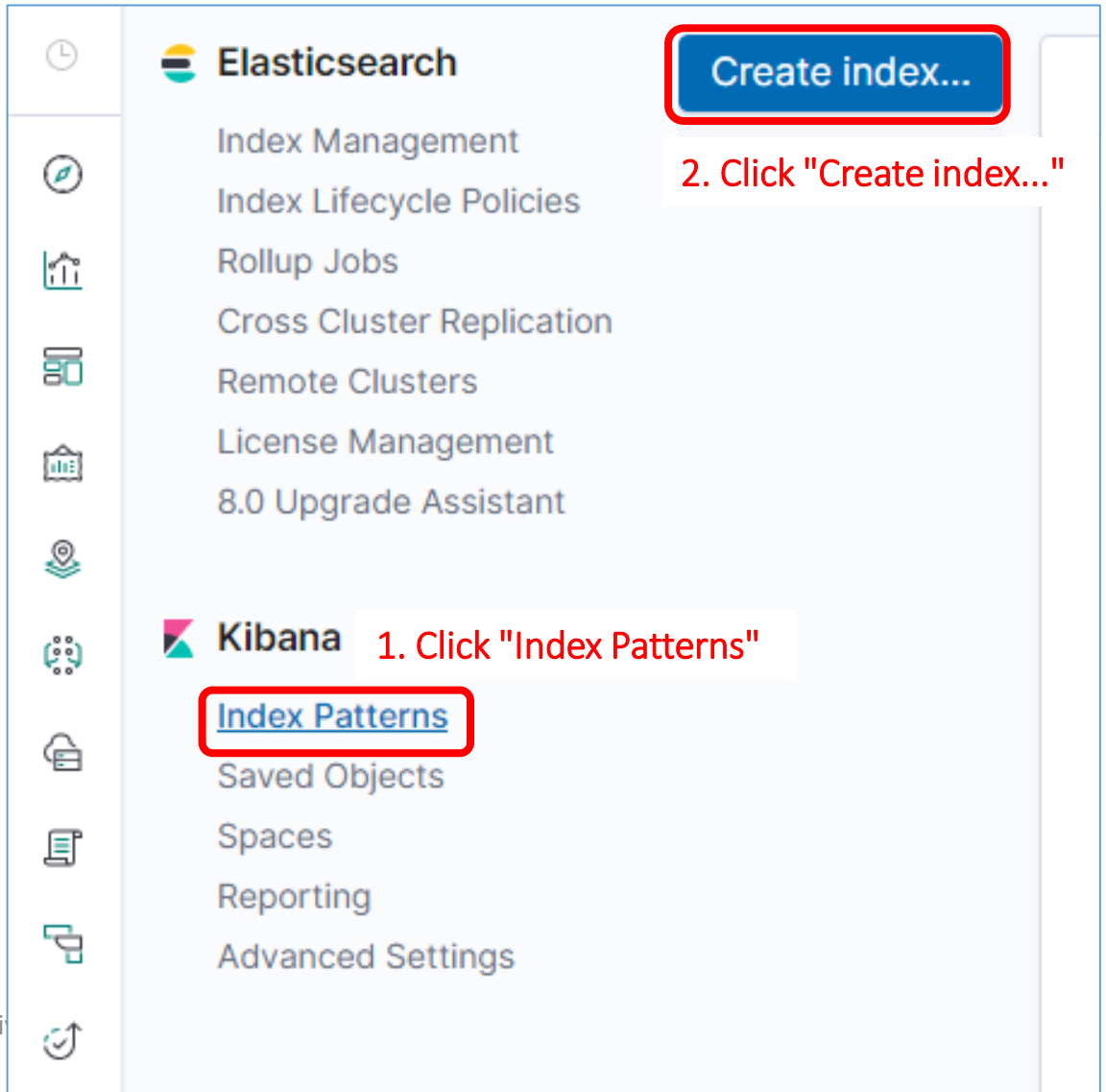
- Click "Management" in the left menu.



Click "Management"

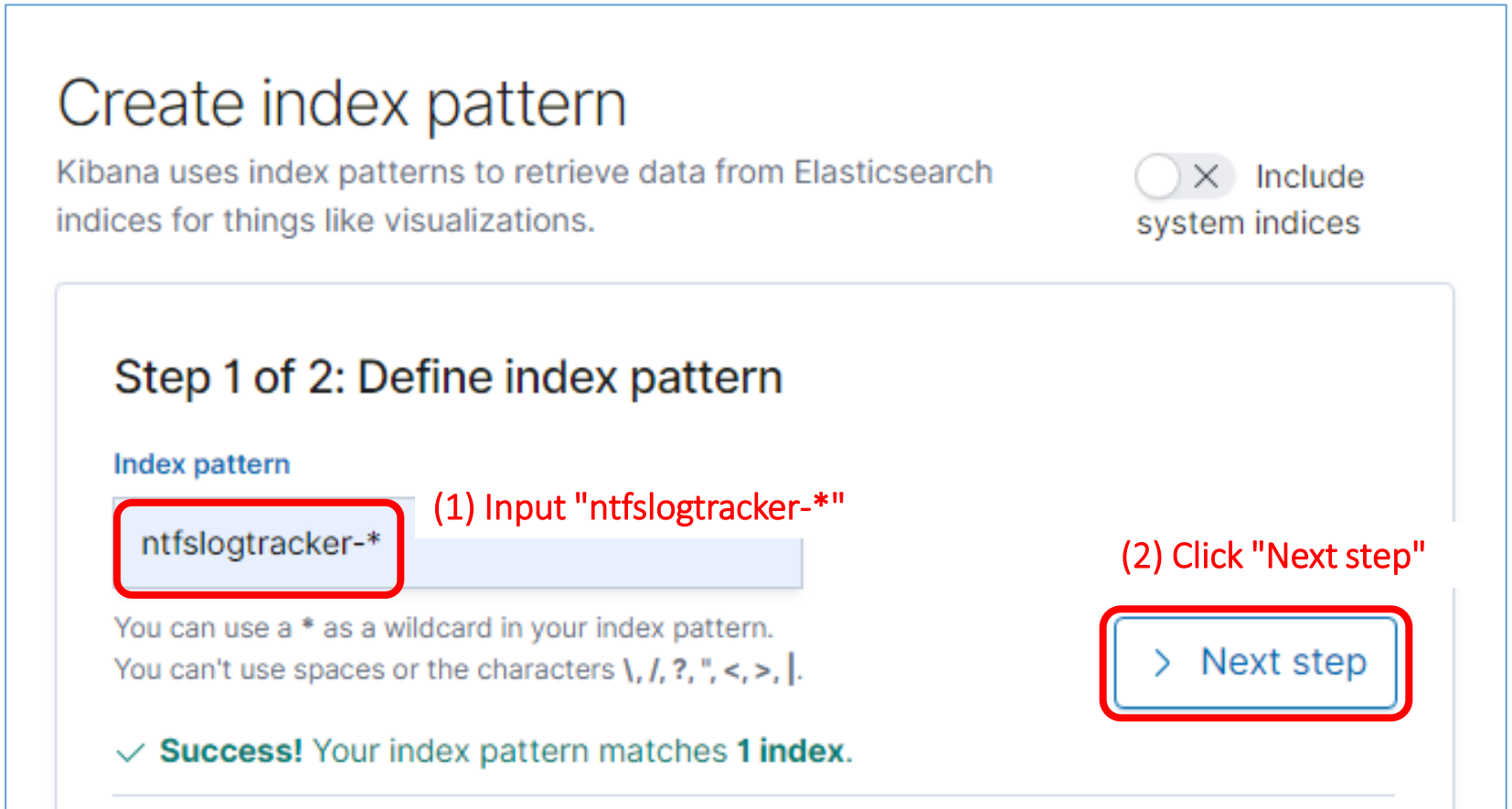
How to Import Proxy Log into Elasticsearch (7)

- Click "Index Patterns" and "Create index..."



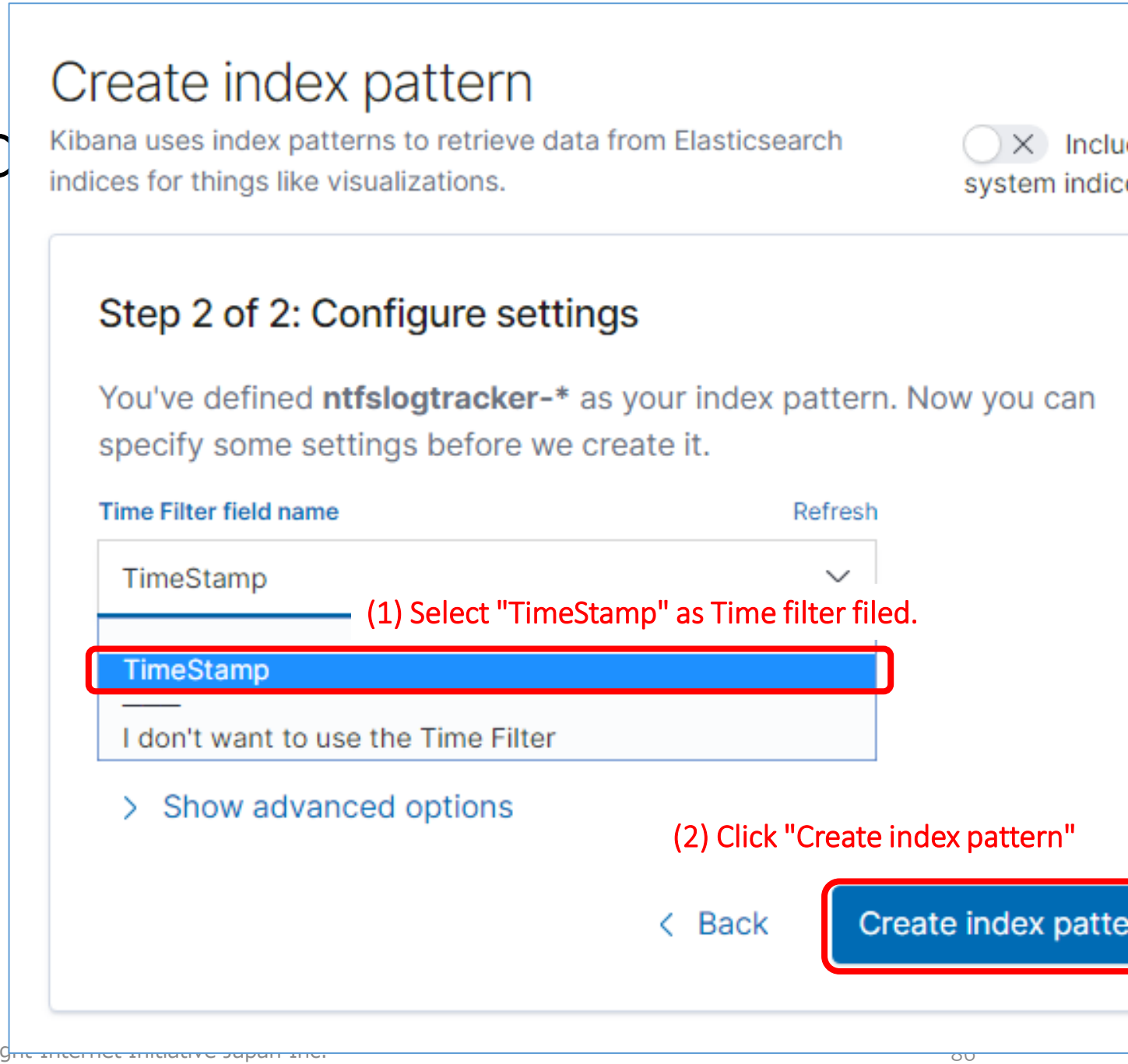
How to import csv data into Elasticsearch (8)

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



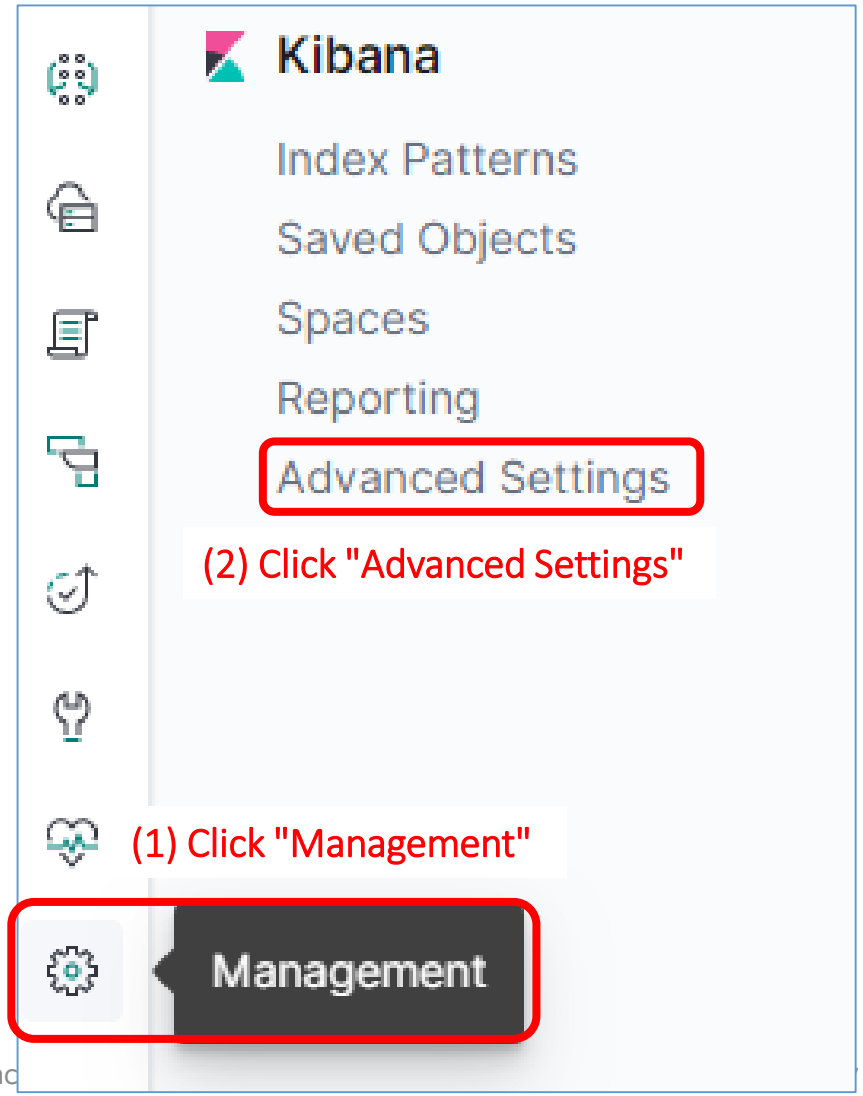
How to import csv d

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



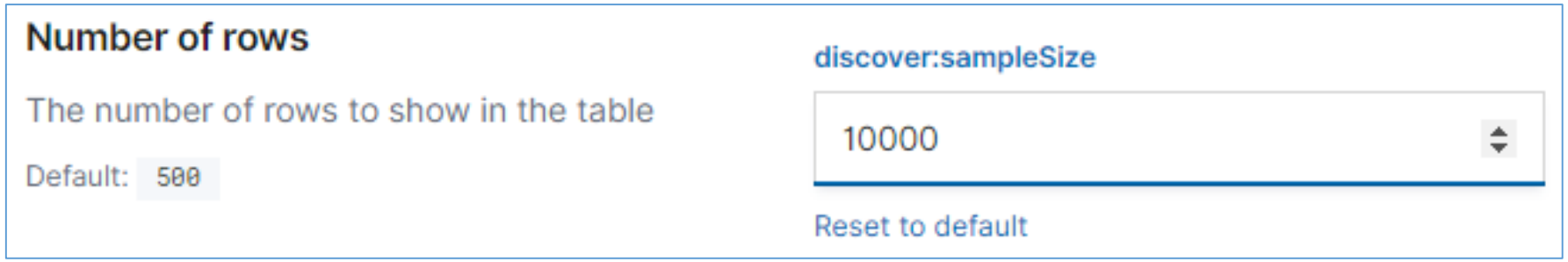
How to import csv data into Elasticsearch (10)

- Navigate to "Advanced Settings" page to set some options.

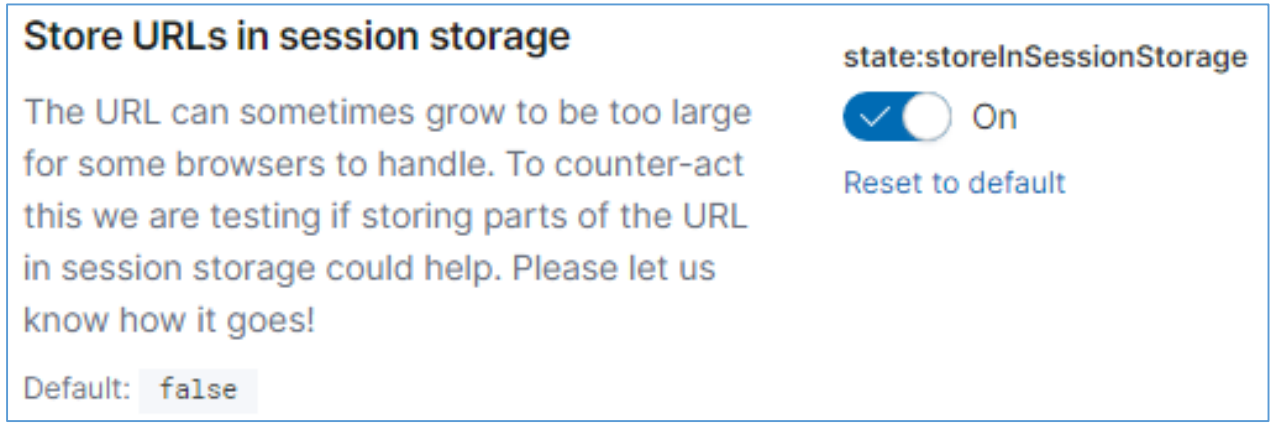


How to import csv data into Elasticsearch (11)

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



  - Change state:storeInSessionStorage from false to On.



Extra Exercise: usn-analytics

Viewing its summary report

Extra Exercise: usn-analytics

Viewing its summary report

- The result of USN Analytics is outputted to the following folder.
  - "E:\Artifacts\other_timeline_analysis\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.

- That's it for this exercise. This exercise is for just checking the report.

Extra Exercise: dealing with INDX

Searching for traces of zip files

Extra Exercise: dealing with INDX

Searching for traces of zip files (1)

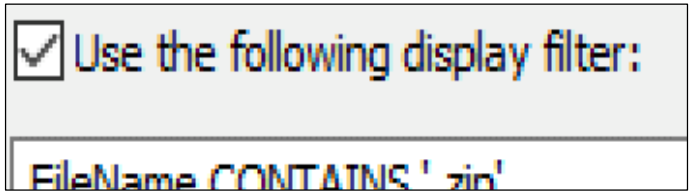
- Conditions:
  - You are investigating a compromised Windows client.
  - It is suspected that attackers stole confidential documents via the client.
  - You already know that the attackers created some zip files, and later, they deleted all of those.
  - There are no related entry in $MFT. There are some related entries in $UsnJrnl, but they do not contain the size of files. Therefore, you should examine INDX attributes.
  - The results of Indx2csv are placed in "E:\Artifacts\other_timeline_analysis\Win10\Indx2csv-output".

- Goals:
  - To determine the total size of the stolen data.

Extra Exercise: dealing with INDX

Searching for traces of zip files (2)

- Let's open the file named "Indx_I30_Entries_2018_03_03-01-19-11.csv" with CSVFileView.exe
- Then filter filename with a string ".zip".
- You can get information about files with .zip extension like the right figure. It could be an evidence of the fact that files had existed.



FileName	MFTReference	AllocSize	RealSize	CTime
d.zip	123625	126976	124858	2017-07-28 06
docs.zip	143284	114688	112374	2017-07-20 09
reports.zip	143410	249036800	249036244	2017-07-20 09
u.zip	147342	62693376	62691785	2017-07-21 09

Extra Exercise: Another solution for Elasticsearch Exercise

Parsing the log without Elasticsearch

Extra Exercise: Another solution for Elasticsearch Exercise Parsing the log without Elasticsearch (1)

- Conditions:
  - You are investigating a compromised Windows client.
  - It is NOT related the scenario. It's just an independent exercise.
  - You already know that attackers installed a RAT to the client. 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.)
  - The utility programs were installed to the folder "\ProgramData\s".
  - Related artifacts are placed in the following folder.
    - E:\Artifacts\other_timeline_analysis\Win10

- Goals:
  - To confirm whether the folder "\ProgramData\s" really existed.
  - To check file system related events logged around the deletion of the folder.

Extra Exercise: Another solution for Elasticsearch Exercise Parsing the log without Elasticsearch (2)

- Hints:
  - There are no entry related to the folder "\ProgramData\s" in $MFT and $Logfile, but $UsnJrnl contains it. You can find the deletion log in output of ntfs-log-tracker.
    - E:\Artifacts\other_timeline_analysis\Win10\ntfs-log-tracker-outpunt
  - In this case, UsnJrnl.csv is too large to be opened with CSVFileView and other CSV viewers. (Often CSV viewers cannot handle logs over about 320,000 lines well.)
  - To reduce the logs to open, we ll use some command line tools first.
  - In this case, we will not use Elasticsearch and Kibana.

Extra Exercise: Another solution for Elasticsearch Exercise Parsing the log without Elasticsearch (3)

- You can display the entries that are related to the folder by the following command.

```
FINDSTR "\ProgramData\s" UsnJrnl.csv
```

- Then you can get results like below.

```
C:\Users\ttaro\Desktop\Training_Materials\TimelineAnalysis\Win10\ntfs-log-tracker-output>FINDSTR "\ProgramData\s"
UsnJrnl.csv
"2017-07-28 17:23:50","441350304","s.zip","\ProgramData\s.zip","Data_Overwritten","Normal","Archive"
"2017-07-28 17:23:51","441350760","s.zip","\ProgramData\s.zip","Data_Overwritten/ File_Closed","Normal","Archive"
"2017-07-28 17:23:51","441350832","s.zip","\ProgramData\s.zip","File_Closed/ File_Deleted","Normal","Archive"
"2017-07-28 17:24:11","441544248","s","\ProgramData\s","File_Closed/ File_Deleted","Normal","Directory"
```

- The deletion entry of the folder exists in $UsnJrnl.
- It can be considered as an evidence of the folder existed.

Extra Exercise: Another solution for Elasticsearch Exercise Parsing the log without Elasticsearch (4)

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

```
"2017-07-28 17:24:11","441544248","s","\ProgramData\s","File_Closed/ File_Deleted","Norm
```

- We extract entries that was logged at the time below from the output of usn-analytics.
  - From 2017-07-28 17:20:00 to 2017-07-28 17:29:59.
- The extracted entries were saved to the file " E:\Artifacts\other_timeline_analysis\Win10\usn-analytics-output\usn_analytics_records_sub1.csv". Let's open it with CSVFileView.

- The command line sample for this filter is below.

```
powershell "Get-Content -Path 'usn_analytics_records-20170425T054422.csv' | Select-Object -first 1" > usn_analytics_records_sub1.csv
```

Enter this in a single line.

```
findstr /C:"2017/07/28 17:2" usn_analytics_records-20170425T054422.csv >> usn_analytics_records_sub1.csv
```

"FileName"	"Reason"	"FileAttr"	"FileID"	"ParentID"	"Path"
"w.vbs"	"OVERWRI...	"ARCHIVE"	"122856"	"122848"	"s\"
"w.vbs -> prograAAAAAAAAA.AAA"	"RENAME"	"ARCHIVE"	"122856"	"122848"	"s\"
"prograAAAAAAAAA.AAA -> prograBBB...	"RENAME"	"ARCHIVE"	"122856"	"5"	"\"
"prograBBBBBBBBB.BBB -> prograCCCC...	"RENAME"	"ARCHIVE"	"122856"	"5"	"\"
"prograCCCCCCCCC.(					
"prograDDDDDDDDD.I					
"prograEEEEEEEEE.EEE					
"prograFFFFFFFFF.FFF					
"prograGGGGGGGGG.(					
"prograHHHHHHHHH					
"prograIIIIIIIII.III -> prograJJJJJJJJJ.JJJ"	"RENAME"	"ARCHIVE"	"122856"	"5"	"\"
"prograJJJJJJJJJ.JJJ -> prograKKKKKKKKK....	"RENAME"	"ARCHIVE"	"122856"	"5"	"\"
"prograKKKKKKKKK.KKK -> prograLLLLL...	"RENAME"	"ARCHIVE"	"122856"	"5"	"\"
	<snip>				
"prograWWWWWWWWWW.WWW -> pro...	"RENAME"	"ARCHIVE"	"122856"	"5"	"\"
"prograXXXXXXXXX.XXX -> prograYYYY...	"RENAME"	"ARCHIVE"	"122856"	"5"	"\"
"prograYYYYYYYYY.YYY -> prograZZZZ...	"RENAME"	"ARCHIVE"	"122856"	"5"	"\"
"prograZZZZZZZZZ.ZZZ"	"DELETE|C...	"ARCHIVE"	"122856"	"5"	"\"
"s\"	"DELETE|C...	"FOLDER"	"122848"	"1587"	""

The logs show that files under the "s" folder were overwritten, then renamed many times, and deleted.

Extra Exercise: Elasticsearch

Checking The Deletion of Attacker's Working Folder (6)

- Consequently, we can say that many files were rewritten, then renamed many times and then 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 a data-erasing tool to delete their tools.

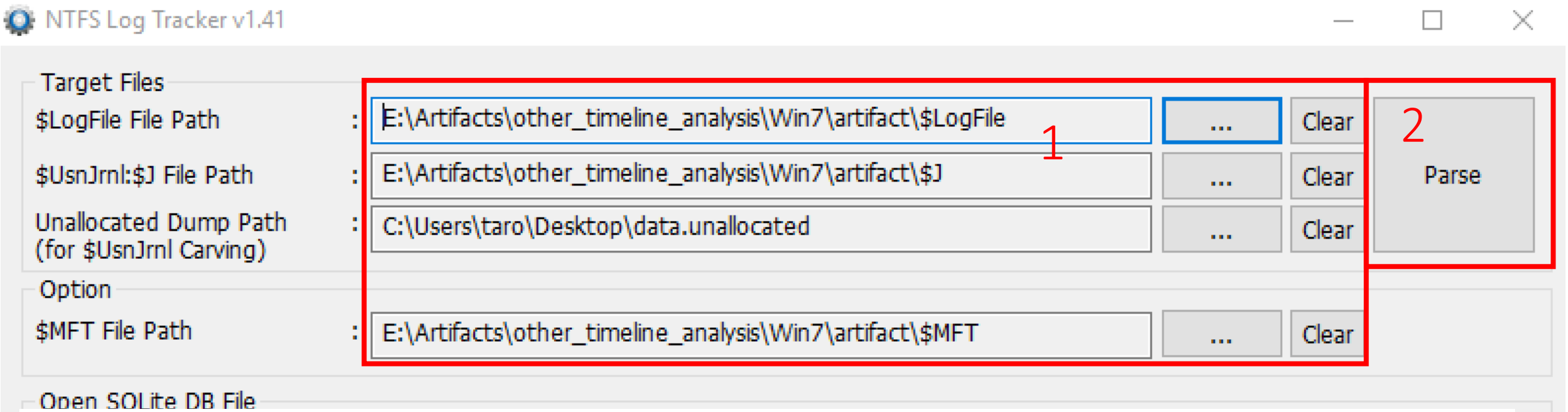
How to parse $Logfile and $UsnJrnl with NTFS Log Tracker

NTFS Log Tracker usages (1)

- First of all, we have to extract the following data from a target disk image.
  - $Logfile
    - It is located under the root of a NTFS volume. You can extract it with TSK and so on.
  - $UsnJrnl:$J
    - It is placed in the following path you can also extract it with TSK and so on.
      - $Extend\$UsnJrnl:$J
  - $MFT
    - It's located under the root of a NTFS volume. You can extract it with TSK and so on.
  - Unallocated Dump
    - You can dump unallocated data from the target disk image with the following command.

```
blkls.exe -A -o [offset of the target volume]  [path to the target disk image]  > data.unallocated
```

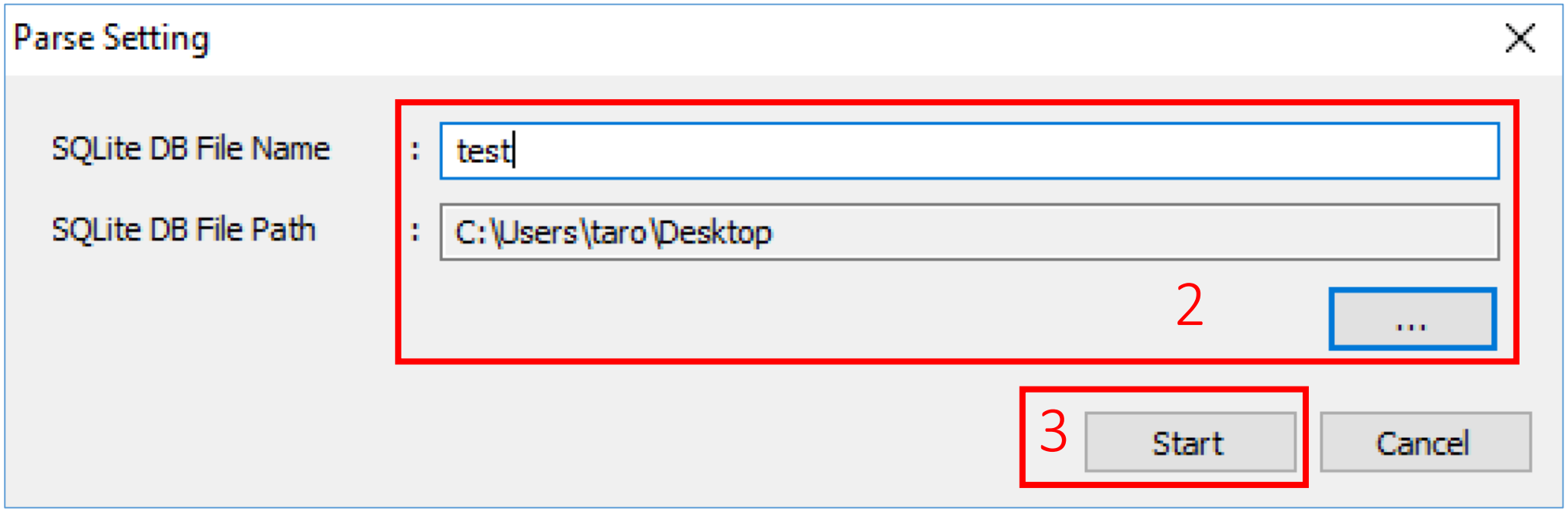
    - blkls is also a part of TSK.



1. First, set the files we prepared beforehand.
   - It requires at least $Logfile or $UsnJrnl:$J being set.
   - Unallocated Dump is option. If it was set, NTFS Log Tracker would carve remaining USN records in the unallocated spaces.
   - $MFT is also an option. NTFS Log Tracker parses $MFT to get file paths. If it's not set, the result would contain only file names and does not include file paths.
2. Then, press the "Parse" button.

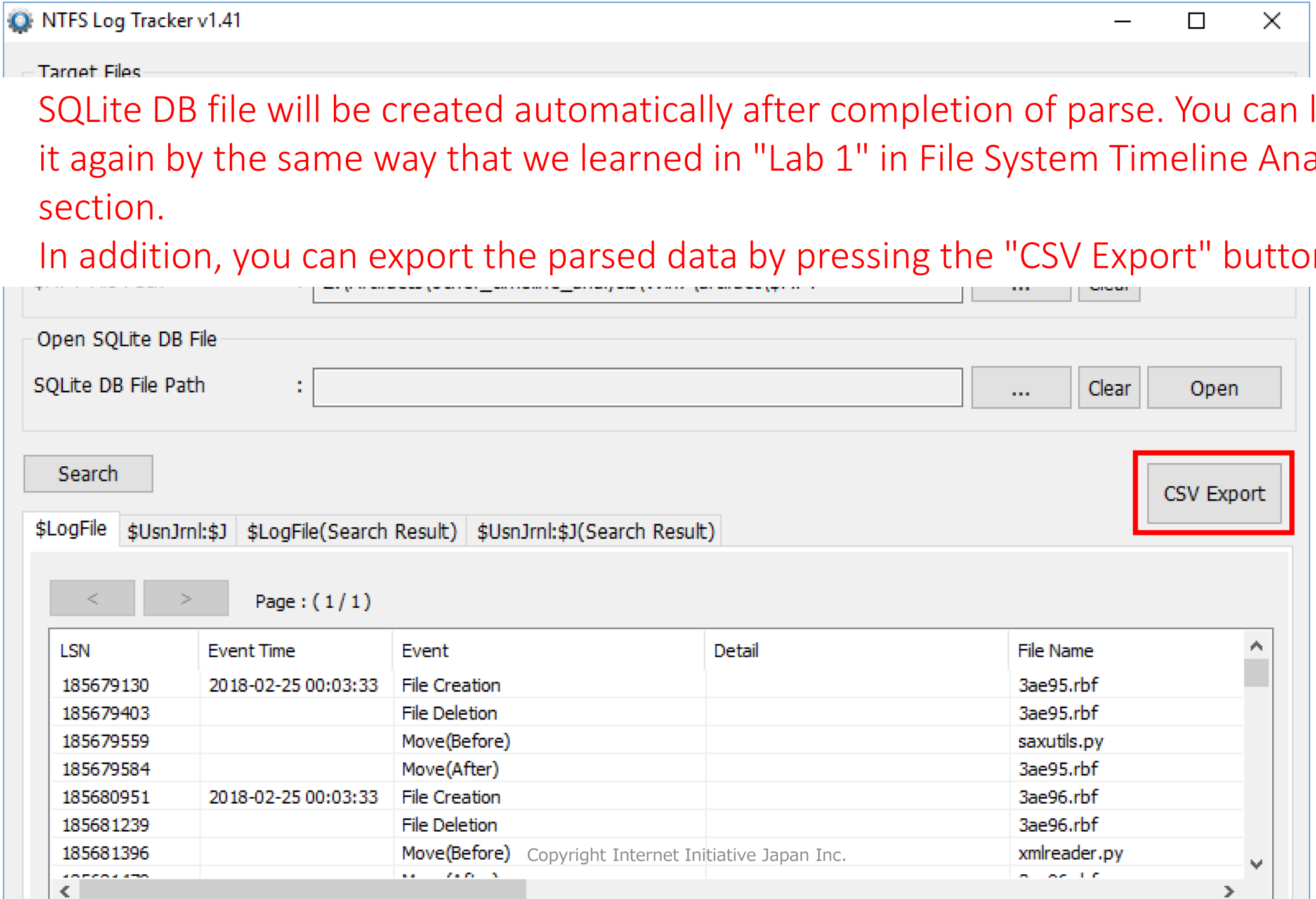
LSN	Event Time	Event	Detail	File Name

NTFS Log Tracker usages (3)



1. Next you will see this dialog.
2. Choose the DB file name and its path to the saved parsed data.
3. Finally, press the "Start" button to start parsing.

- SQLite DB file will be created automatically after completion of parse. You can load it again by the same way that we learned in "Lab 1" in File System Timeline Analysis section.
- In addition, you can export the parsed data by pressing the "CSV Export" button.



End of Document