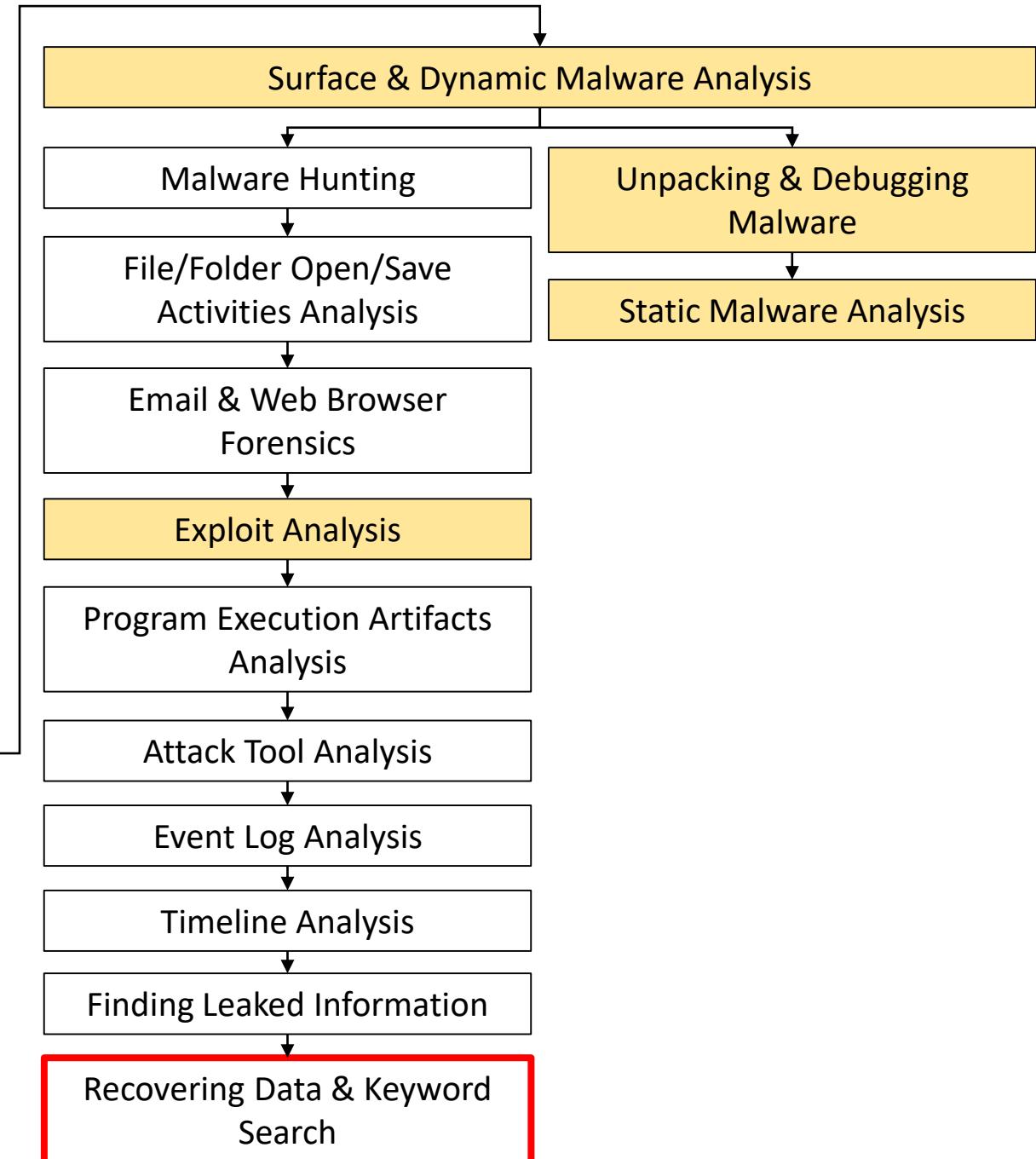
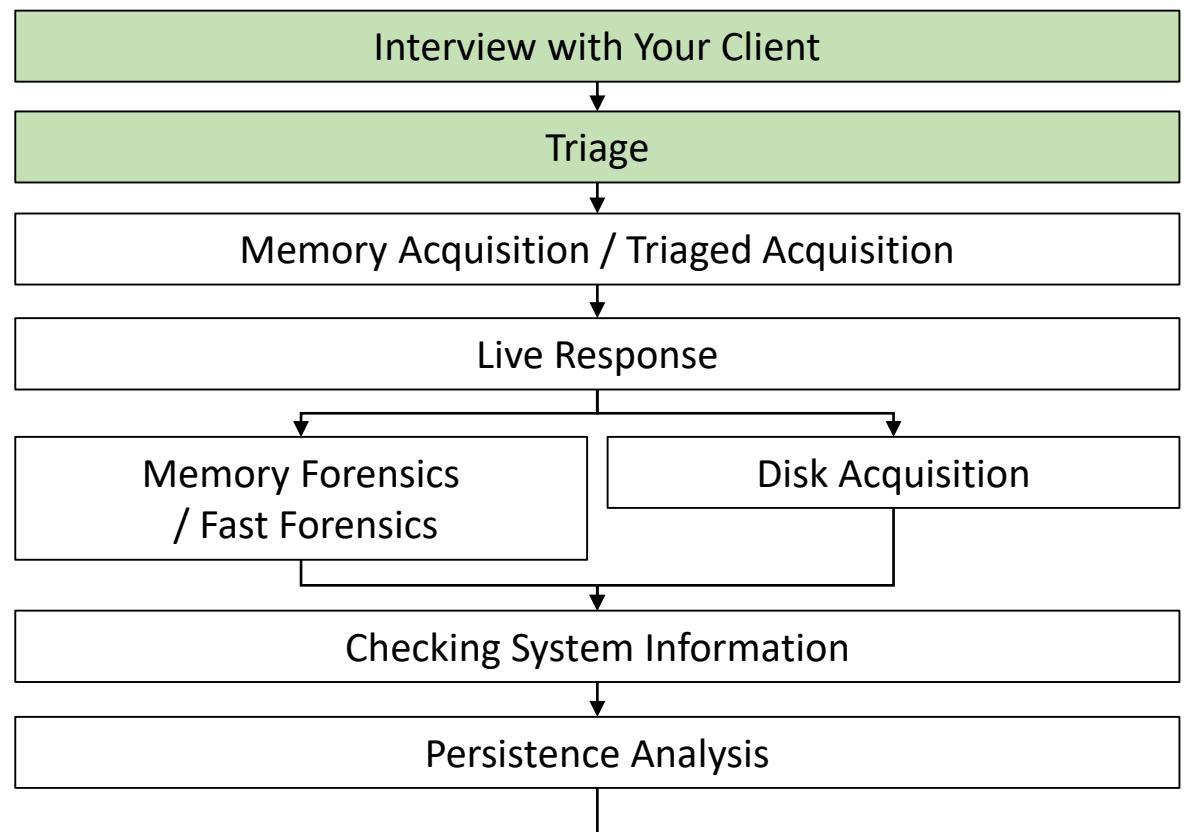


# Recovering Deleted Data



# Data Recovering 101

- What is Data Recovering?
  - It is to recover deleted data. Usually, it means extracting deleted files from a disk image.
  - When data that we need to recover is not a form of the complete file, but the data contains a record or some records such as event logs or journal logs, we sometimes try to recover data on per record basis, not on per file basis.
- Why Data Recovering?
  - Attackers often delete files such as attacking tools and collected archives. If we recover and analyze them, we can understand their purpose and what they actually did.
  - We sometimes get important evidences from old log records that were recovered from unallocated spaces. It is very useful when the logs have already been rotated.

# Various Data Recovering Features/Methods

- Recycle Bin
  - We can recover files that were dropped into Recycle Bin. It is easy to extract them from disk images.
- VSS (Volume Shadow Snapshots)
  - Modern Windows systems automatically perform periodic backup with Volume Shadow Copy Service. We sometimes find deleted files from those snapshots.
- Resident data
  - As we mentioned before, Windows does not delete data immediately when files are deleted. It just flags the file entries as "deleted" in Master File Table. Therefore, we can extract deleted files if the entries still exist.
- Carving & keyword search
  - Carving is the process to search large data chunk such as pagefile, unallocated space, file slack, and even the whole disk image, for a specific data. Typically, we use magic or hard coded values in file formats, keywords, regex patterns and so on to identify particular data formats. It takes a long time.

# Recovering Files From Recycle Bin

# Recovering Files From Recycle Bin (1)

- When you drop some files into the Recycle Bin, those files are moved to the following directory.
  - \\$Recycle.Bin\<USER SID>\

```
C:\$Recycle.Bin>dir /a
Volume in drive C has no label.
Volume Serial Number is 1CD6-DDDD

Directory of C:\$Recycle.Bin

06/25/2018  02:12 PM    <DIR>      .
06/25/2018  02:12 PM    <DIR>      ..
06/25/2018  02:12 PM    <DIR>      S-1-5-18
06/17/2018  10:43 PM    <DIR>      S-1-5-21-1070966257-3988512674-567233159-1000
07/04/2018  07:56 PM    <DIR>      S-1-5-21-1070966257-3988512674-567233159-1001
```

# Recovering Files From Recycle Bin (2)

- When a file is dropped into the Recycle Bin, it is renamed to a meaningless strings that starts with "\$R", like "\$R6IABX7.zip". In this case, its metadata, including original name, path, and timestamps, is also saved as another file named like "\$I6IABX7.zip". Names of metadata files start with "\$I". Their file extension is the same as the original one.
- There are some tools to parse these metadata files. We will introduce them later.

```
C:\$Recycle.Bin\S-1-5-21-1070966257-3988512674-567233159-1001>dir /a
Volume in drive C has no label.
Volume Serial Number is 1CD6-DDDD

Directory of C:\$Recycle.Bin\S-1-5-21-1070966257-3988512674-567233159-1001

07/04/2018  07:56 PM    <DIR>   .
07/04/2018  07:56 PM    <DIR>   ..
07/04/2018  07:56 PM                  96 $I6IABX7.zip
07/04/2018  07:56 PM                  98 $IJMIURV.png
05/05/2018  04:08 PM                228,659 $R6IABX7.zip
07/04/2018  07:40 PM                97,187 $RJMIURV.png
06/17/2018  10:45 PM                  129 desktop.ini
                                         5 File(s)      326,169 bytes
                                         2 Dir(s)  17,983,774,720 bytes free
```

These files contain metadata

Renamed files contain original data

# Recycle Bin Analysis Tools (1)

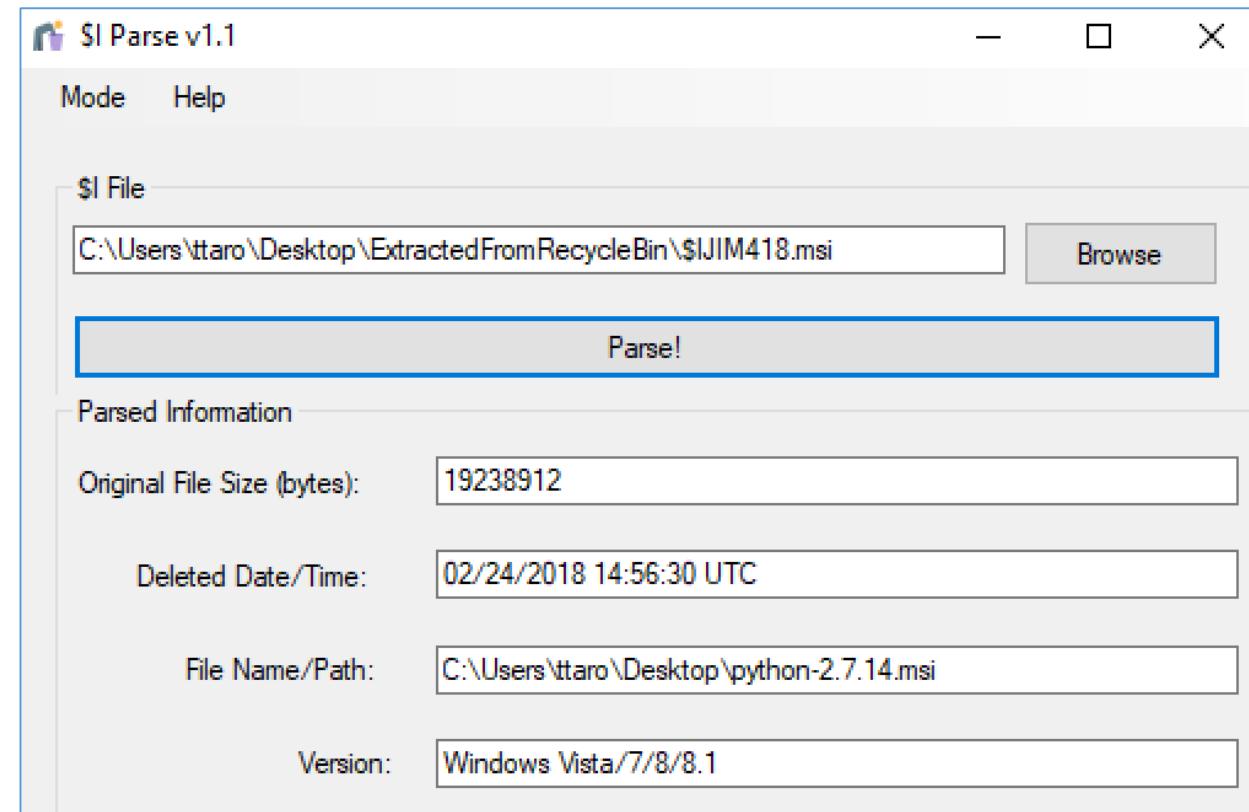
- Rifiuti2
  - It is the command line tool to parse \$I files.
  - It can parse files under the given directory.

```
Administrator: Command Prompt
C:\WINDOWS\system32>C:\Users\ttaro\Desktop\rifiuti2-0.6.1-win\x64\rifiuti-vista.exe C:\$Recycle.Bin\S-1-5-21-775692482-2859662954-4177908799-1000
Recycle bin path: 'C:\$Recycle.Bin\S-1-5-21-775692482-2859662954-4177908799-1000'
Version: 2

Index Deleted Time      Size     Path
$IXQWGLO.zip    2018-04-10 09:41:22    32665216      C:\Users\ttaro\Desktop\sig.s.zip
$IV6R2WQ.dat     2018-04-24 06:35:20    262144       C:\Users\ttaro\Desktop\UsrClass.dat
$I8P7P96.exe     2018-05-24 09:36:25    1464109      C:\Users\ttaro\Desktop\cl64_328.exe
```

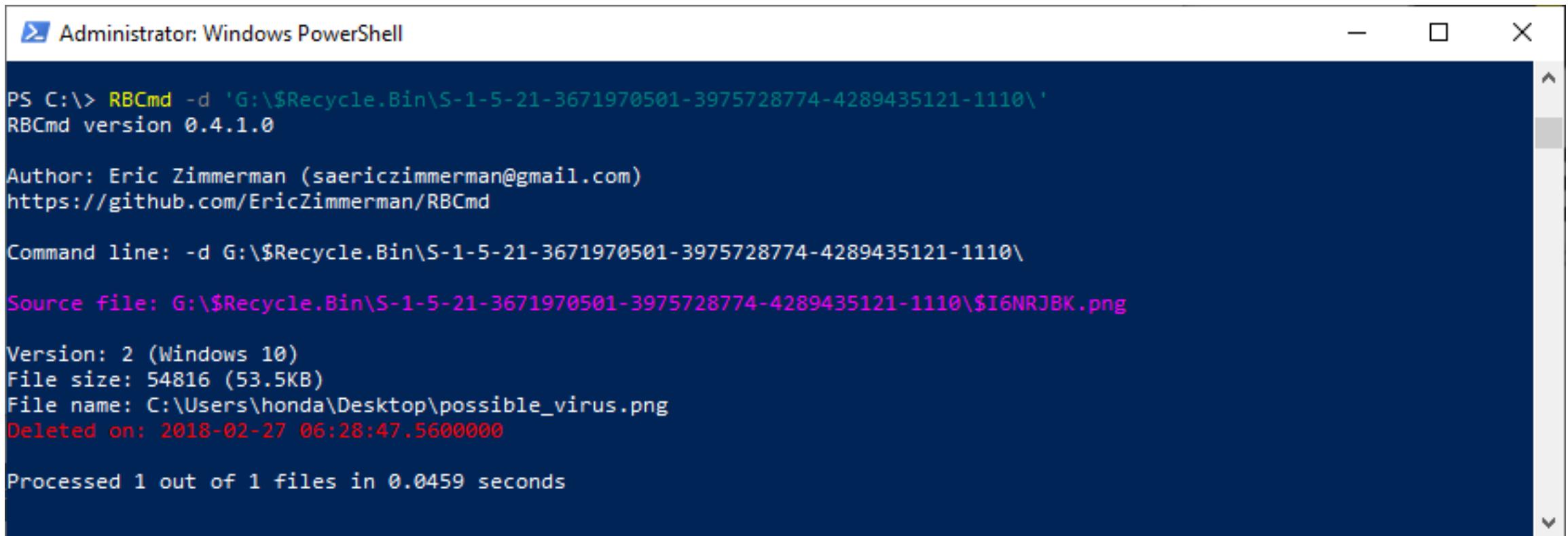
# Recycle Bin Analysis Tools (2)

- \$I Parse
  - It parses files that start with \$I.
  - This tool has two operation modes. File mode is to parse just one file and directory mode is to parse files under the target folder.
  - You have to extract target files from disk images first.



# Recycle Bin Analysis Tools (3)

- RBCmd
  - This is yet another command line tool to parse INFO2/\$I files.



```
Administrator: Windows PowerShell
PS C:\> RBCmd -d 'G:\$Recycle.Bin\S-1-5-21-3671970501-3975728774-4289435121-1110\'  
RBCmd version 0.4.1.0  
  
Author: Eric Zimmerman (saericzimmerman@gmail.com)  
https://github.com/EricZimmerman/RBCmd  
  
Command line: -d G:\$Recycle.Bin\S-1-5-21-3671970501-3975728774-4289435121-1110\  
  
Source file: G:\$Recycle.Bin\S-1-5-21-3671970501-3975728774-4289435121-1110\$I6NRJBK.png  
  
Version: 2 (Windows 10)  
File size: 54816 (53.5KB)  
File name: C:\Users\honda\Desktop\possible_virus.png  
Deleted on: 2018-02-27 06:28:47.5600000  
  
Processed 1 out of 1 files in 0.0459 seconds
```

# Practice Exercise 1:

Check the files located in the Recycle Bin with Rifiuti2

# Practice Exercise 1:

Check the files located in the Recycle Bin with Rifiuti2 (1)

- Conditions:

- It is NOT related to the scenario 1. It is just an independent exercise.
- This is an exercise to deal with Recycle Bin.
- The target disk image is the following.
  - E:\Artifacts\scenario1\_E01\Client-Win10-2\_honda.E01
- The target user honda's SID is the following.
  - S-1-5-21-3671970501-3975728774-4289435121-1110

- Goal:

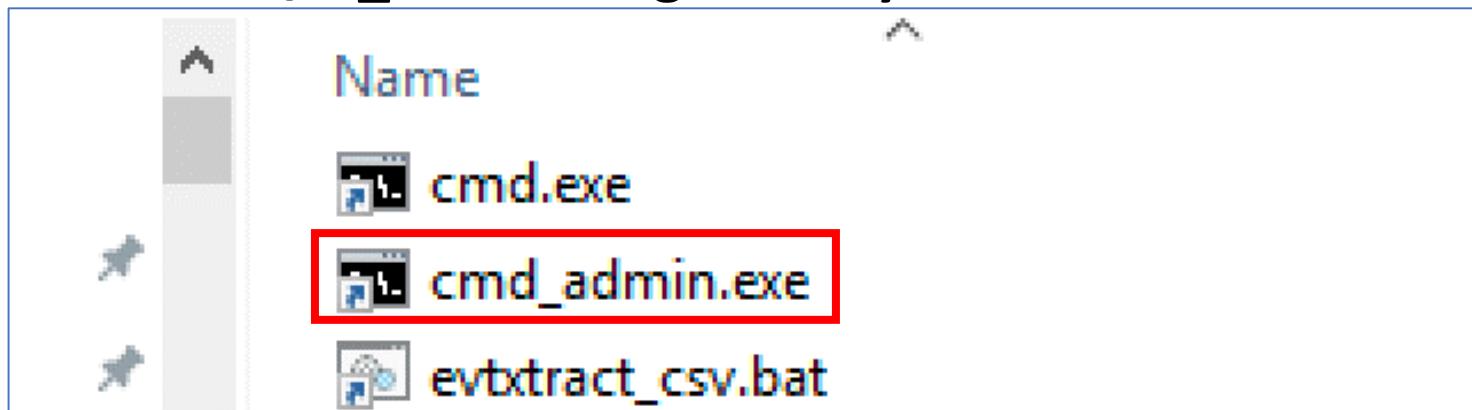
- To confirm files under user honda's Recycle Bin folder.

# Practice Exercise 1:

Check the files located in the Recycle Bin with Rifiuti2 (2)

- First of all, mount the disk image below with Arsenal Image Mounter.
  - E:\Artifacts\scenario1\_E01\Client-Win10-2\_honda.E01
- Launch cmd.exe as administrator by double-clicking the following icon.

**Shortcuts\09\_RecoveringDataKeywordSearch**



# Practice Exercise 1:

## Check the files located in the Recycle Bin with Rifiuti2 (3)

- Check the Recycle.Bin folder for user "Honda" with the following command.
  - Note: Honda's SID is S-1-5-21-3671970501-3975728774-4289435121-1110

```
dir G:\$Recycle.Bin\S-1-5-21-3671970501-3975728774-4289435121-1110
```

Honda's SID

```
Volume in drive G has no label.  
Volume Serial Number is B81C-324B
```

```
Directory of G:\$Recycle.Bin\S-1-5-21-3671970501-3975728774-4289435121-1110
```

02/27/2018 03:28 PM	112	\$I6NRJBK.png
02/27/2018 03:24 PM	54,816	\$R6NRJBK.png
2 File(s)	54,928 bytes	
0 Dir(s)	27,537,383,424 bytes free	

Metadata file

Actual content file

# Practice Exercise 1:

Check the files located in the Recycle Bin with Rifiuti2 (4)

- Run rifiuti command to parse the metadata file under the target location.

```
rifiuti-vista.exe -z G:\$Recycle.Bin\S-1-5-21-3671970501-3975728774-4289435121-1110
```

```
Recycle bin path: 'G:\$Recycle.Bin\S-1-5-21-3671970501-3975728774-4289435121-1110'
```

```
Version: 2
```

```
OS Guess: Windows 10 or above
```

```
Time zone: Tokyo Standard Time [+0900]
```

Index	Deleted	Time	Size	Path	The original file name and path
\$I6NRJBK.png		2018-02-27 15:28:47	54816	C:\Users\honda\Desktop\possible_virus.png	Size

- If you are interested to check the actual content of the file, you can open the content file that have a name starting with "\$R".

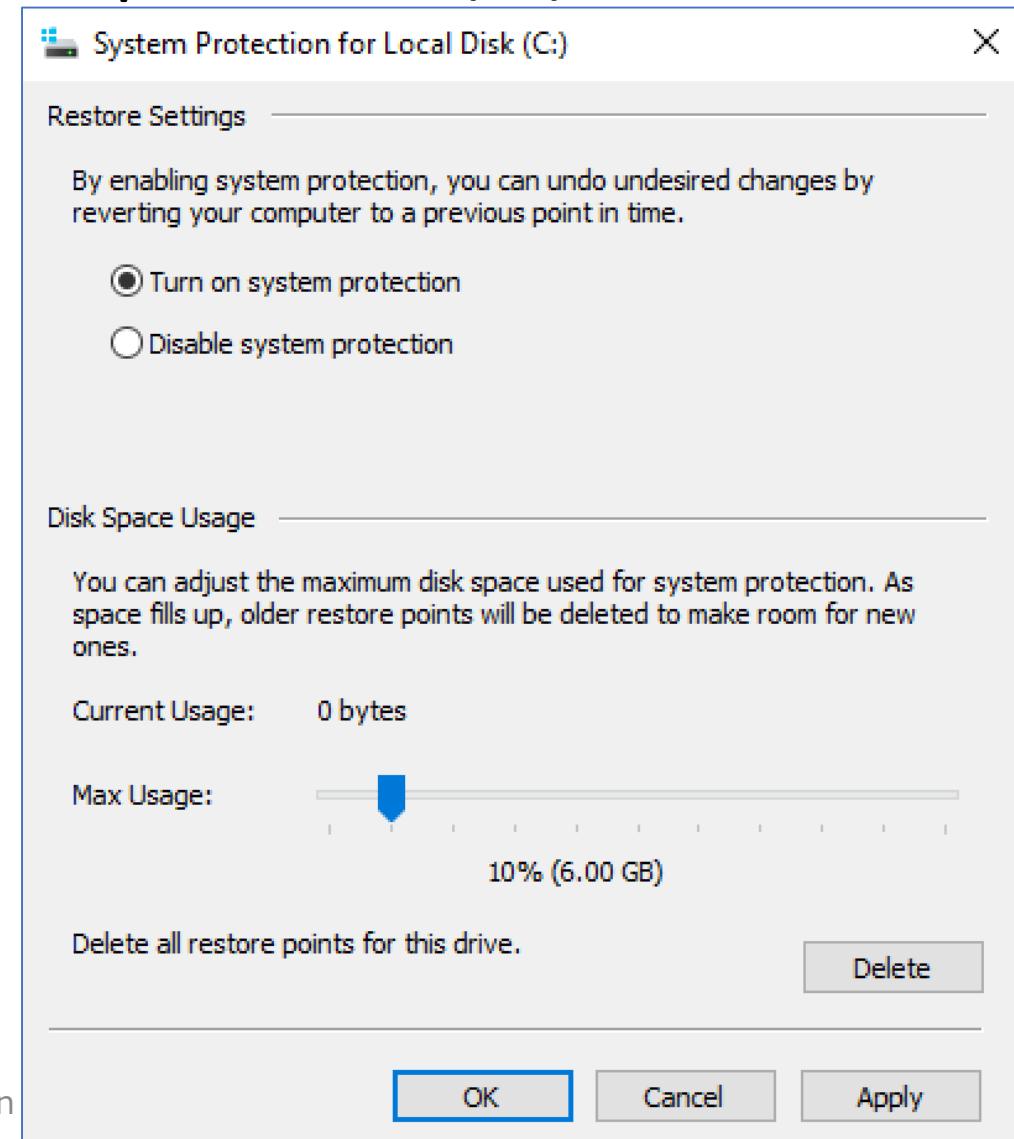
# Preparation for the Next Exercise

- Unmount the disk image.
- Quit all applications.

# Data Recovering With VSS Snapshots

# Data Recovering With VSS Snapshots (1)

- Windows OS takes backup automatically with Volume Shadow Copy Service (VSS). It is for "System Protection for Local Disk" function.
- We can restore a system with VSS snapshots taken by this function.
- We can also extract files from VSS snapshots contained in disk images.



# Data Recovering With VSS Snapshots (2)

- The default backup targets and triggers to take snapshots depend on Windows version.

	Files to backup	Backup interval / trigger to take backup
Windows 7	All files except for some files*1.	Once a week.
Windows 8 and above	Typically, system binaries and file system metadata files are taken*2.	Triggered by execution of several specific functions such as Windows Update and each installer.
Windows Servers	All files except for some files*1.	Twice (7am and 12pm) on every weekday.

- The following registry key contains settings of exclusion files.
  - HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Control\BackupRestore\FilesNotToSnapshot
- Microsoft recommends to use "Previous versions" function for user files to backup. However, we can change this behavior by configuring ScopeSnapshots feature.

For further information:

[https://www.ij.ad.jp/en/dev/iir/pdf/iir\\_vol37\\_focused1\\_EN.pdf](https://www.ij.ad.jp/en/dev/iir/pdf/iir_vol37_focused1_EN.pdf)

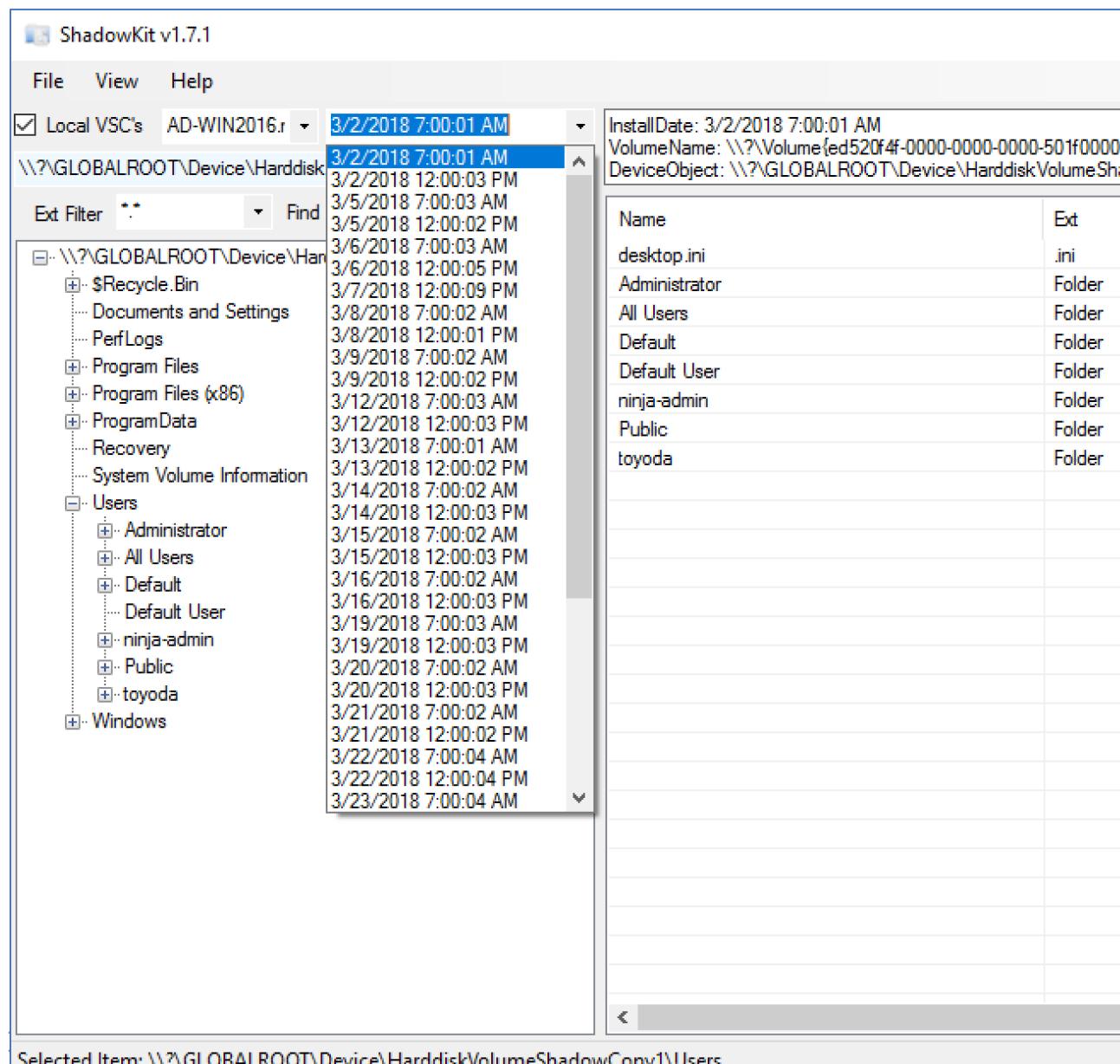
# Data Recovering With VSS Snapshots (3)

- How VSS works?
  - When a VSS snapshot is created, "Catalog" and "Store" are created and saved under System Volume Information folder located at the NTFS root.
  - Catalog is used to store metadata such as creation date of the snapshot.
  - Store backs up the actual data. It keeps only the differential data.
  - Therefore, when a certain snapshot is restored, it applies those stored differentials to the current files.
  - If a snapshot is deleted, its Catalog and Store are also deleted. However we can recover deleted snapshots with vss\_carver if the Store data is not overwritten.
- For further information, check our briefing session at Black Hat USA 2018:
  - RECONSTRUCT THE WORLD FROM VANISHED SHADOW: RECOVERING DELETED VSS SNAPSHOTS
    - <https://i.blackhat.com/us-18/Thu-August-9/us-18-Kobayashi-Reconstruct-The-World-From-Vanished-Shadow-Recovering-Deleted-VSS-Snapshots.pdf>

# VSS Exploring Tools (1)

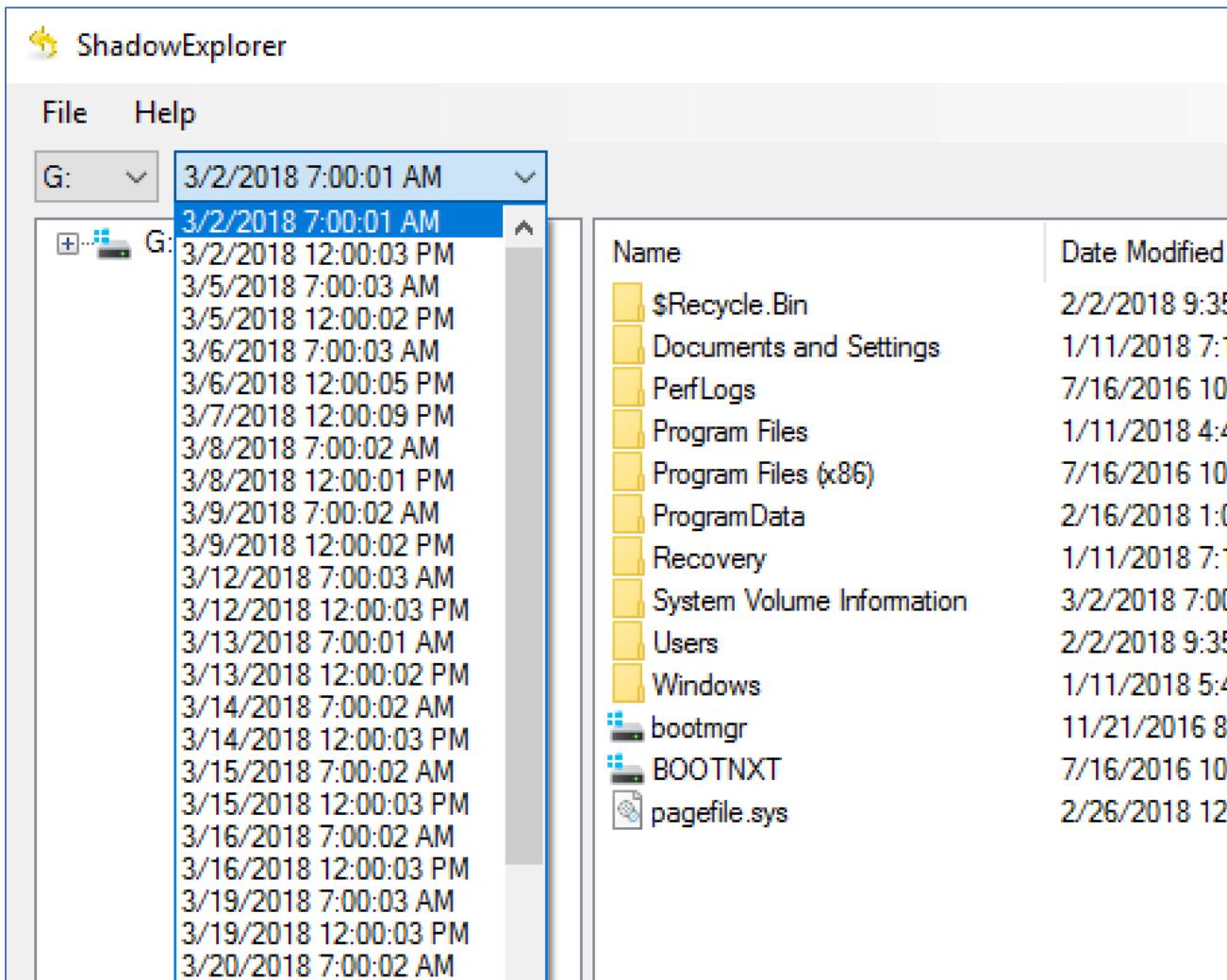
- ShadowKit

- It can be used for exploring VSS snapshots with GUI.
- We can recover snapshots data, including files in the Recycle Bin folder.
- However, it cannot treat file system metadata such as \$MFT.



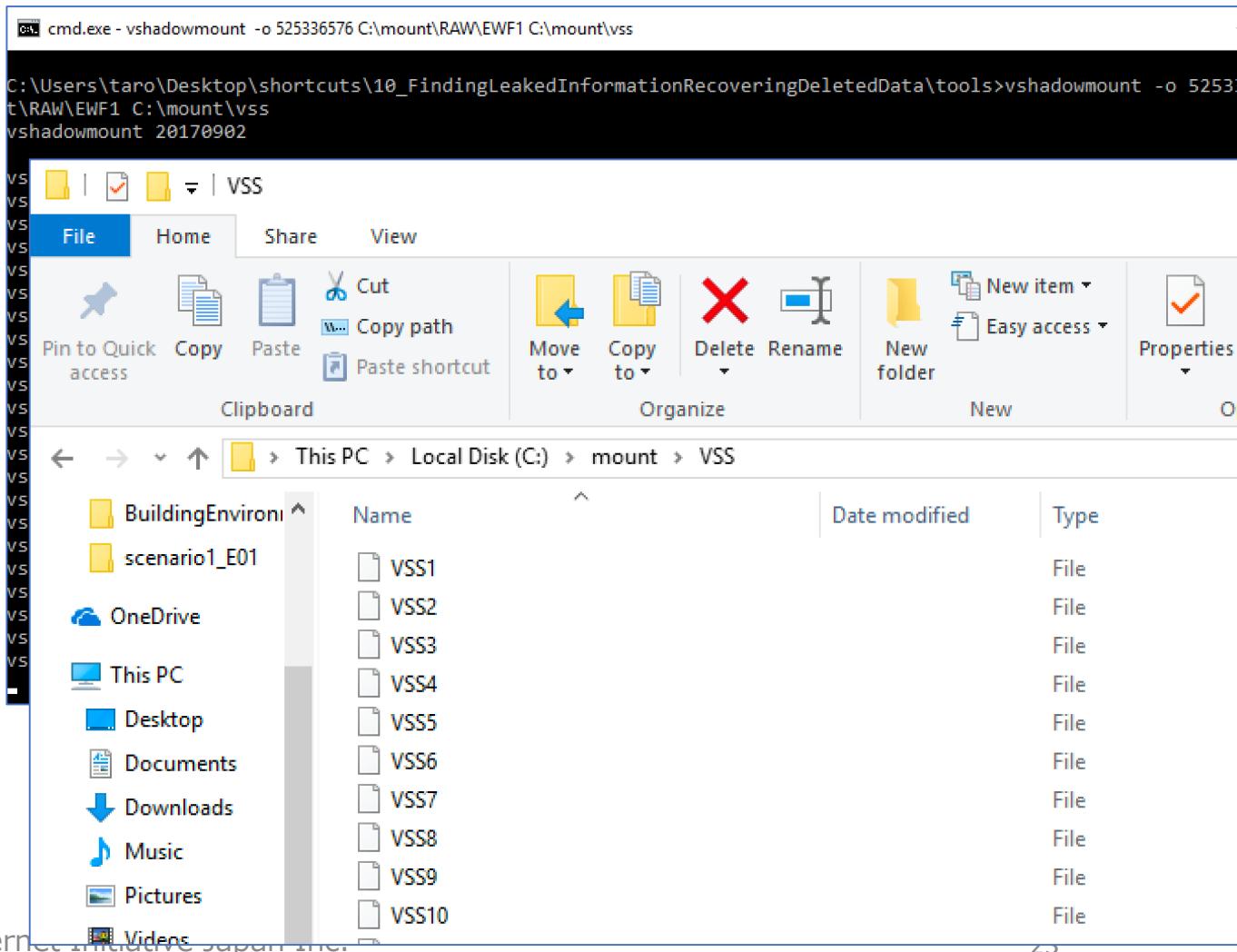
# VSS Exploring Tools (2)

- ShadowExplorer
  - It is also a GUI based VSS snapshot explorer and very similar to ShadowKit.
  - It cannot treat file system metadata, either.



# VSS Exploring Tools (3)

- **vshadowmount**
  - It presents each VSS snapshot as a raw disk image.
  - This tool is a part of libvshadow library, and the library is open source.



# VSS Exploring Tools (4)

## QUESTION

### vssadmin & mklink

- We can also explore VSS snapshots using basic Windows commands.
- First, check the location of each snapshot with vssadmin command. Then, create a symbolic link with mklink command. In that way, we can view the VSS snapshots.

Administrator: Command Prompt

```
C:\Windows\system32>vssadmin list shadows /for=G:  
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool  
(C) Copyright 2001-2013 Microsoft Corp.  
  
Contents of shadow copy set ID: {c558eee1-b75e-45e6-a6dc-eb7a9f94403a}  
Contained 1 shadow copies at creation time: 3/2/2018 7:00:01 AM  
Shadow Copy ID: {3d943be1-705f-455e-90dd-316b7870d95a}  
Original Volume: (G):\?\Volume{ed520f4f-0000-0000-0000-501f00000000}\  
Shadow Copy Volume: \?\GLOBALROOT\Device\HarddiskVolumeShadowCopy44  
Originating Machine: AD-WIN2016.ninja-motors.net  
Service Machine: AD-WIN2016.ninja-motors.net  
Provider: 'Microsoft Software Shadow Copy provider 1.0'  
Type: ClientAccessible  
Attributes: Persistent, Client-accessible, No auto release, No writers, Differentiation
```

Administrator: Command Prompt

```
C:\>mklink /d C:\mount\vss\shadow44 \?\GLOBALROOT\Device\HarddiskVolumeShadowCopy44\  
symbolic link created for C:\mount\vss\shadow44 <<====> \?\GLOBALROOT\Device\HarddiskVolumeShadowCopy44
```

```
C:\>cd C:\mount\vss\shadow44
```

```
C:\mount\VSS\shadow44>dir  
Volume in drive C has no label.  
Volume Serial Number is 8274-CA45
```

```
Directory of C:\mount\VSS\shadow44
```

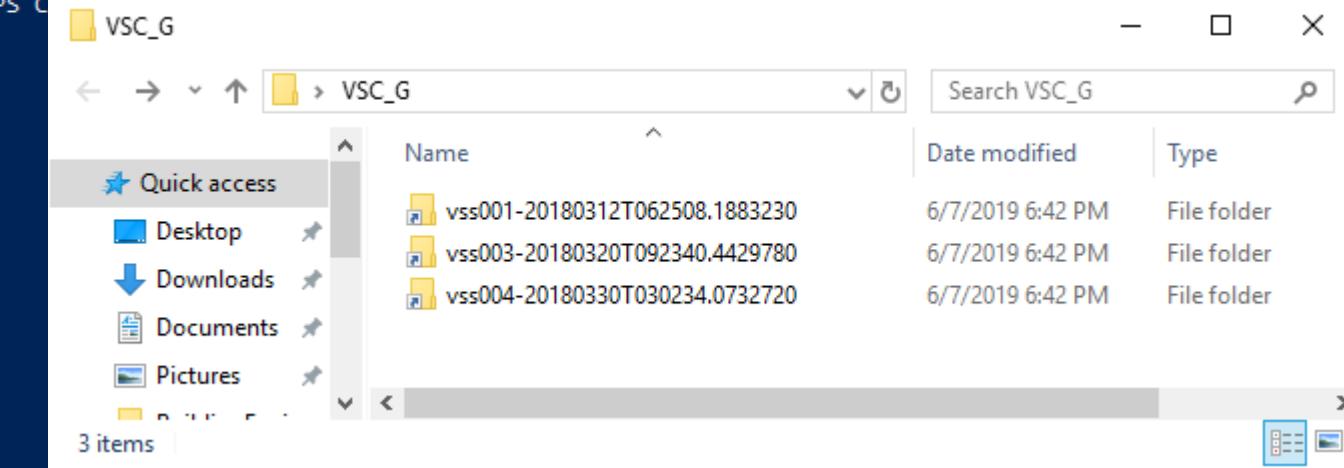
```
07/16/2016 10:23 PM <DIR> PerfLogs  
01/11/2018 04:40 PM <DIR> Program Files  
07/16/2016 10:23 PM <DIR> Program Files (x86)  
02/02/2018 09:35 PM <DIR> Users  
01/11/2018 05:46 PM <DIR> Windows  
0 File(s) 0 bytes  
5 Dir(s) 37,912,285,184 bytes free
```

```
C:\mount\VSS\shadow44>  
C:\mount\VSS\shadow44>  
C:\mount\VSS\shadow44>
```

# VSS Exploring Tools (5)

- **VSCMount**

- You can access VSS via symbolic links and files that are managed by the snapshots with Command Prompt or Explorer.
- <https://binaryforay.blogspot.com/2018/09/introducing-vscmount.html>



A screenshot of a Windows File Explorer window titled "VSC\_G". The window shows three items: "Desktop", "Downloads", and "Documents". Below these, there are three folder icons labeled "vss001-20180312T062508.1883230", "vss003-20180320T092340.4429780", and "vss004-20180330T030234.0732720". The file names, dates modified, and types are listed to the right of the folder icons.

Name	Date modified	Type
vss001-20180312T062508.1883230	6/7/2019 6:42 PM	File folder
vss003-20180320T092340.4429780	6/7/2019 6:42 PM	File folder
vss004-20180330T030234.0732720	6/7/2019 6:42 PM	File folder

```
Administrator: Windows PowerShell
PS C:\Windows\system32> vscmount --dl G: --mp C:\Users\taro\Desktop\VSC --ud
VSCMount version 0.5.3.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/VSCMount

Command line: --dl G: --mp C:\Users\taro\Desktop\VSC --ud

Creating directory 'C:\Users\taro\Desktop\VSC_G'
Mounting VSCs to 'C:\Users\taro\Desktop\VSC_G'

VSCs found on volume G: 3. Mounting...
    VSS 1      (Id {56e9d26b-0f72-4b9b-bf54-77f55c88b1e0}, Created on: 2018-03-12 06:25:08.1883230)
    VSS 3      (Id {28fe2faf-78ca-4fd8-9c76-da645e01a2bb}, Created on: 2018-03-20 09:23:40.4429780)
    VSS 4      (Id {90cd587f-692a-4b84-a9fe-d40d8b9889d4}, Created on: 2018-03-30 03:02:34.0732720)

Mounting complete. Navigate VSCs via symbolic links in 'C:\Users\taro\Desktop\VSC_G'

To remove VSC access, delete individual VSC directories or the main mountpoint directory
PS C:\Windows\system32>
```



**Normal Explorer process**

You don't currently have permission to access this folder.  
Click Continue to permanently get access to this folder.

**Explorer++ with administrative privilege**

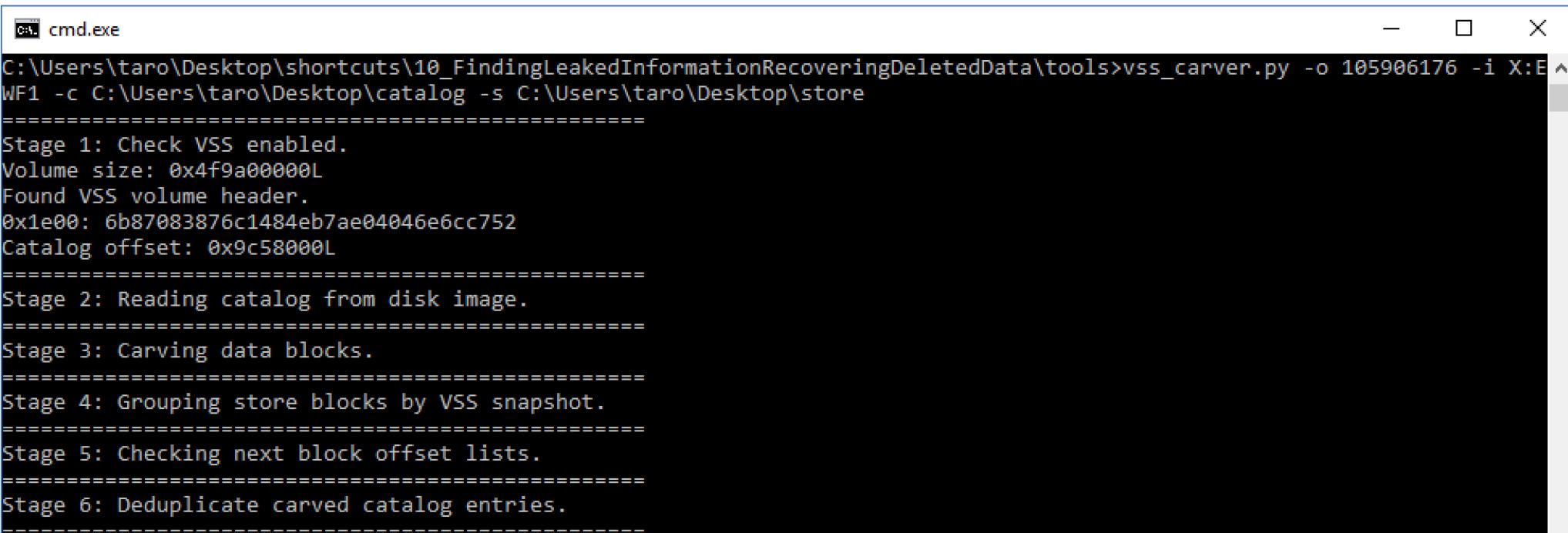
C:\Users\taro\Desktop\vsc\_G\vss008-20180312T062508.1883230\Users\toyoda

**Note that the Explorer process cannot gain administrator privilege. Therefore, you cannot access contents that are owned by other accounts. Instead, you can access them with third-party alternative explorer application such as Explorer++; FreeCommander or WinFile, or Command Prompt, with administrator privilege.**

Name	Type	Size	Date Modified	Attributes
atom	File folder		2/19/2018 3:37 PM	----D--
AppData	File folder		1/30/2018 11:41 AM	-H-D--
Application Data	File folder		1/30/2018 11:41 AM	-H-SD--
Contacts	File folder		1/30/2018 11:42 AM	--R-D--
Cookies	File folder		1/30/2018 11:41 AM	-L-SD--
Desktop	File folder			-D--
Documents	File folder			-D--
Downloads	File folder			-D--
Evernote	File folder			D--
Favorites	File folder			--R-D--
Links	File folder			-H-SD--
Local Settings	File folder			1/30/2018 11:42 AM
Music	File folder		1/30/2018 11:42 AM	--R-D--
My Documents	File folder		1/30/2018 11:41 AM	-H-SD--
Nethood	File folder		1/30/2018 11:41 AM	-H-SD--
OneDrive	File folder		1/30/2018 11:44 AM	--R-D--
Pictures	File folder		1/30/2018 11:43 AM	--R-D--
PrintHood	File folder		1/30/2018 11:41 AM	-H-SD--
Recent	File folder		1/30/2018 11:41 AM	-H-SD--
Saved Games	File folder		1/30/2018 11:42 AM	--R-D--
Searches	File folder		1/30/2018 11:42 AM	--R-D--
SendTo	File folder		1/30/2018 11:41 AM	-H-SD--
Templates	File folder		1/30/2018 11:41 AM	-H-SD--
Videos	File folder		1/30/2018 11:42 AM	--R-D--
スタートメニュー	File folder		1/30/2018 11:41 AM	-H-SD--
NTUSER.DAT	DAT File	1.50 MB	3/6/2018 9:37 PM	AH-----
ntuser.dat.LOG1	LOG1 File	192 KB	1/30/2018 11:41 AM	AH-S---
ntuser.dat.LOG2	LOG2 File	416 KB	1/30/2018 11:41 AM	AH-S---
NTUSER.DAT(2d9fccb...)	BLF File	64.0 KB	2/2/2018 9:03 PM	AH-S---
NTUSER.DAT(2d9fccb...)	REGTRANS-MS File	512 KB	2/2/2018 9:03 PM	AH-S---
NTUSER.DAT(2d9fccb...)	REGTRANS-MS File	512 KB	1/30/2018 3:33 PM	AH-S---
ntuser.ini	Configuration settings	20 bytes	1/30/2018 11:41 AM	-H-S---
ntuser.pol	POL File	456 bytes	2/6/2018 5:30 PM	AHRS---

# VSS Exploring Tools (6)

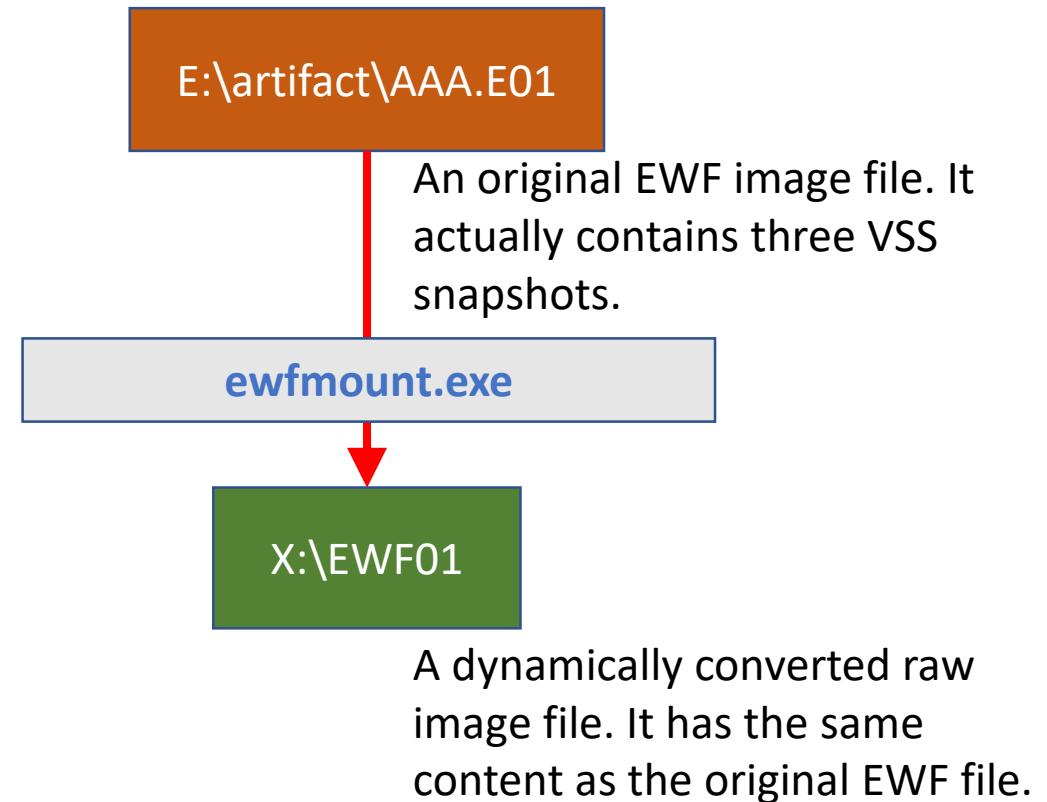
- **vss\_carver**
  - It is a tool to carve deleted VSS snapshots.
  - It requires enhanced version of vshadowmount to mount carved snapshots.  
The enhanced version is released by the same author of this tool.



```
cmd.exe
C:\Users\taro\Desktop\shortcuts\10_FindingLeakedInformationRecoveringDeletedData\tools>vss_carver.py -o 105906176 -i X:E\WF1 -c C:\Users\taro\Desktop\catalog -s C:\Users\taro\Desktop\store
=====
Stage 1: Check VSS enabled.
Volume size: 0x4f9a0000L
Found VSS volume header.
0x1e00: 6b87083876c1484eb7ae04046e6cc752
Catalog offset: 0x9c58000L
=====
Stage 2: Reading catalog from disk image.
=====
Stage 3: Carving data blocks.
=====
Stage 4: Grouping store blocks by VSS snapshot.
=====
Stage 5: Checking next block offset lists.
=====
Stage 6: Deduplicate carved catalog entries.
=====
```

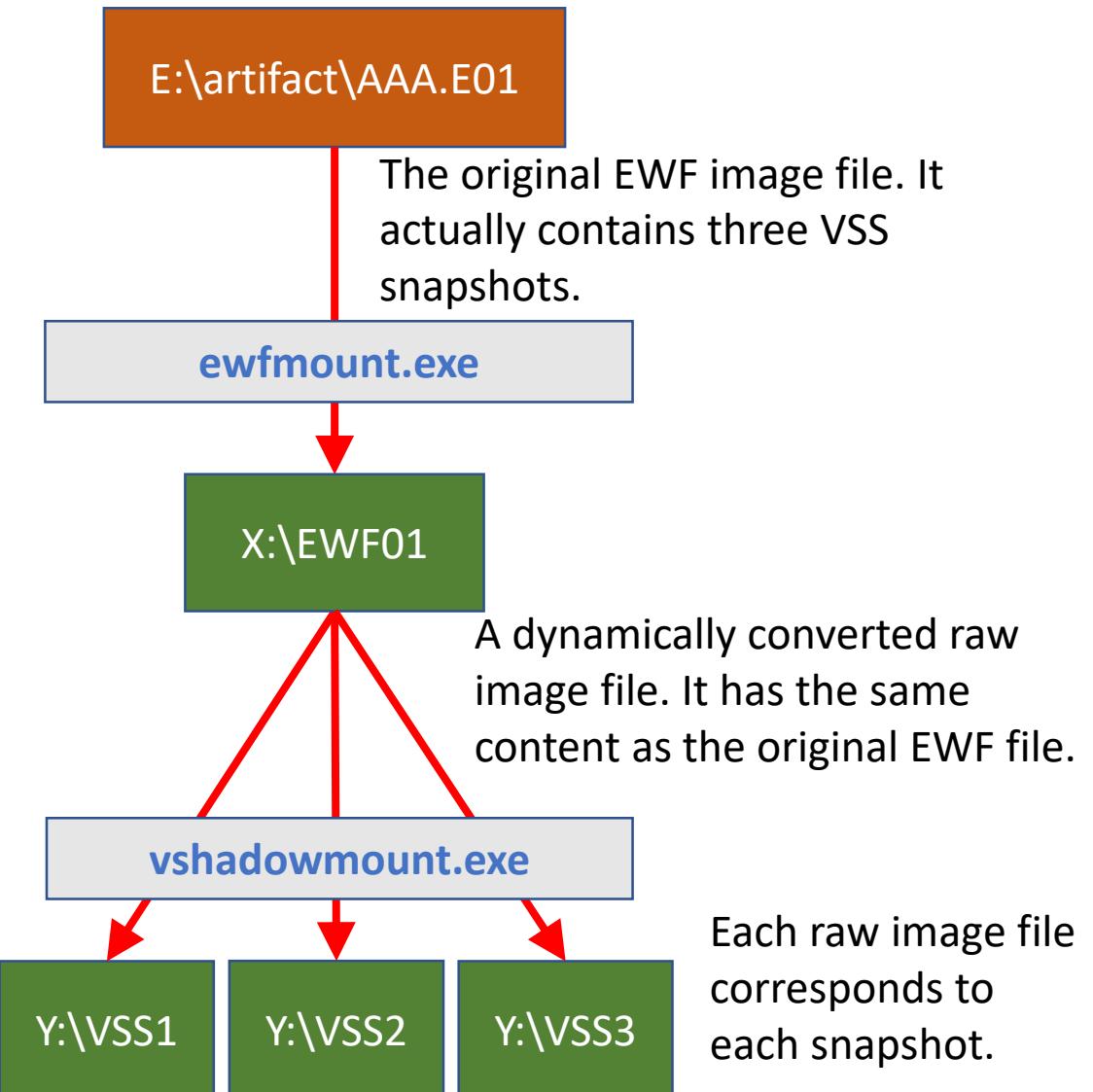
# How We Access VSS Snapshots (1)

- We will use `ewfmount.exe` and `vshadowmount.exe` to access VSS snapshots contained in a certain EWF (.E01) disk image.
- `ewfmount` command dynamically converts from an original EWF image into a raw image file named EWF01. In this case, the EWF file AAA.E01 is presented as a raw image file under drive X.
- Then, we can mount and parse the raw image file with image mounting and parsing tools.
- Note that, Arsenal Image Mounter supports E01 format. It can dynamically mount E01 files. However, some image mounting and parsing tools support raw image format only.



# How We Access VSS Snapshots (2)

- vshadowmount command dynamically converts VSS snapshots within a certain raw image file into individual raw image files. In this case, there are three VSS snapshots in the original disk image, and those snapshots are presented as individual raw image files under drive Y.
- Then, we can mount and parse the raw image files VSS1, VSS2, and VSS3 with image mounting and parsing tools.



# Practice Exercise 2:

## Recovering Deleted VSS Snapshots

# Practice Exercise 2: Recovering Deleted VSS Snapshots (1)

IMPORTANT

- Condition:

- The following disk image contains a Windows volume compromised by TFlower ransomware.
  - E:\Artifacts\other\_E01\infected\_drive\_d.E01
- A huge text file "sample.txt" is placed on the user Taro's Desktop, and it is encrypted by the ransomware.
- VSS snapshots are also deleted by the ransomware.

- Goal:

- To recover the encrypted text file.

- Hint:

- You might carve deleted VSS snapshots with vss\_carver.

Dear Sir/Ma,

Sorry to inform you but many files have been encrypted.  
This simply means that you will not be able to access them.

To get the DECRYPT TOOL for your files.

You may upload 1 of your encrypted files.  
But, the file should not contain any sensitive information.

E-MAIL Address:>>

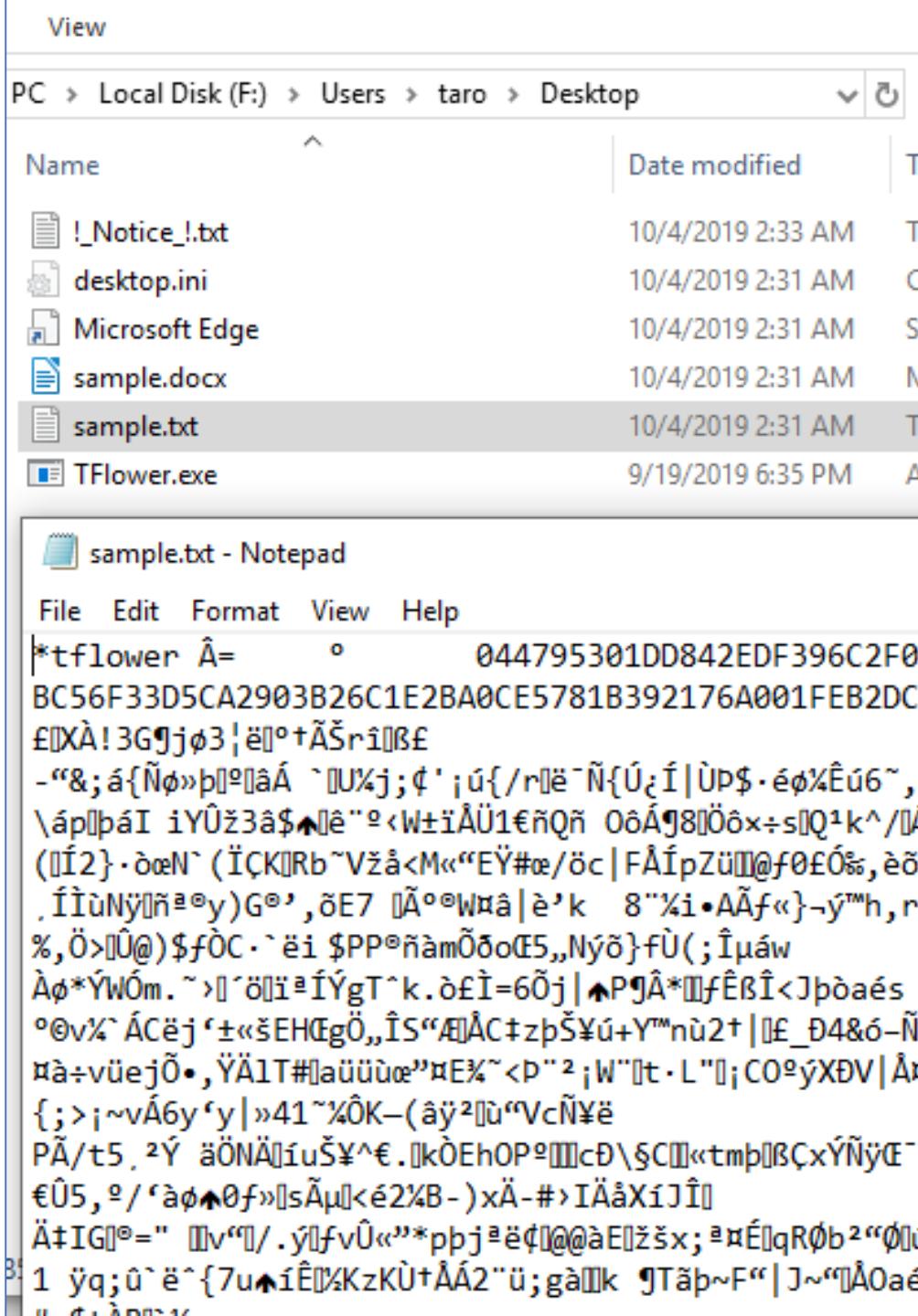
[flower.harris@protonmail.com](mailto:flower.harris@protonmail.com)

[flower.harris@tutanota.com](mailto:flower.harris@tutanota.com)

# Practice Exercise 2:

## Recovering Deleted VSS Snapshots (2)

- “sample.txt” and “sample.docx” are placed on the user Taro’s Desktop. And these files are encrypted by the ransomware.

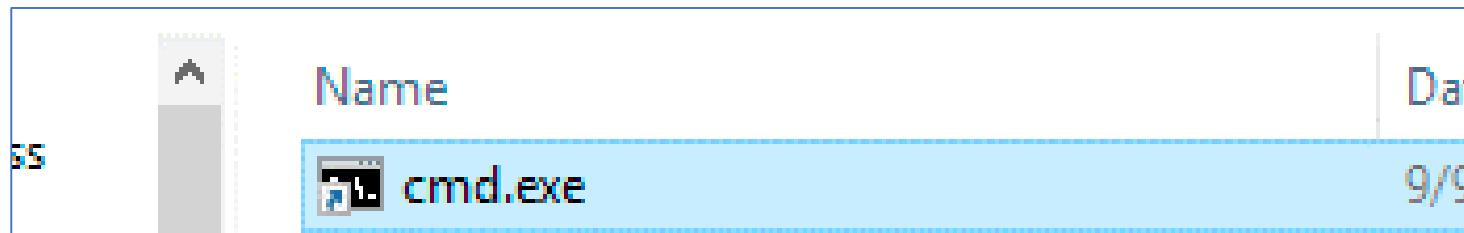


# Practice Exercise 2:

## Recovering Deleted VSS Snapshots (3)

- In order to examine the compromised volume, we should use ewfmount first.
  - Launch cmd.exe from the path below.

### Shortcuts\09\_RecoveringDataKeywordSearch



- Convert from an EWF image file to the raw image dynamically with the following command.

```
ewfmount.exe E:\Artifacts\other_E01\infected_drive_d.E01 X:
```

Path to the target disk image

Where dynamically converted  
raw image files are represented

```
cmd.exe - ewfmount.exe E:\Artifacts\other_E01\infected_drive_d.E01 X:
```

```
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.
```

```
C:\shortcuts\09_RecoveringDataKeywordSearch\tools>ewfmount.exe E:\Artifacts\other_E01\infected_drive_d.E01 X:
ewfmount 20190317
```

# Practice Exercise 2: Recovering Deleted VSS Snapshots (4)

- Check the partition table

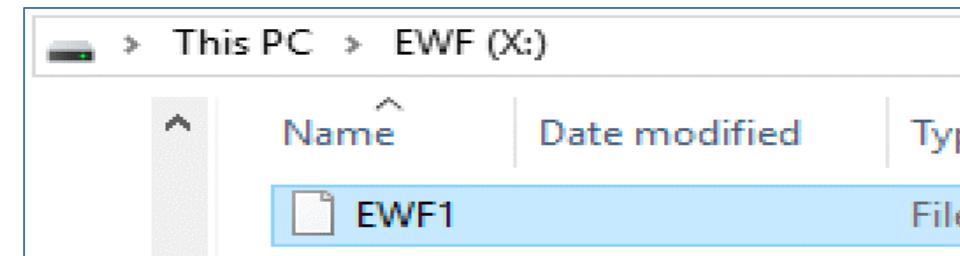
- The E01 image is presented as a raw disk image on drive X.
- Next, let's confirm volume offsets of the image with the following command.

```
mmls.exe X:\EWF1
```

```
GUID Partition Table (EFI)
Offset Sector: 0
Units are in 512-byte sectors
```

The result shows the partition table of the image. The 6th partition is recognized as Basic data partition, which seems to be a regular system volume.

	Slot	Start	End	Length	Description
000:	Meta	0000000000	0000000000	0000000001	Safety Table
001:	-----	0000000000	0000002047	0000002048	Unallocated
002:	Meta	0000000001	0000000001	0000000001	GPT Header
003:	Meta	0000000002	0000000033	0000000032	Partition Table
004:	000	0000002048	0000411647	0000409600	EFI system partition
005:	001	0000411648	0000673791	0000262144	Microsoft reserved partition
006:	002	0000673792	0062912511	0062238720	Basic data partition
007:	-----	0062912512	0062914559	0000002048	Unallocated



# Practice Exercise 2: Recovering Deleted VSS Snapshots (5)

- Get the target volume's VSS information
  - The following command shows the VSS snapshots stored in the volume.

```
vshadowinfo.exe -o 344981504 X:\EWF1
```

Path to the target disk image. It must be a file in raw image format.

The offset of the volume in bytes.

We can calculate it by multiplying the sector size (512 bytes) by offset of the volume (sector 673,792).

cmd.exe

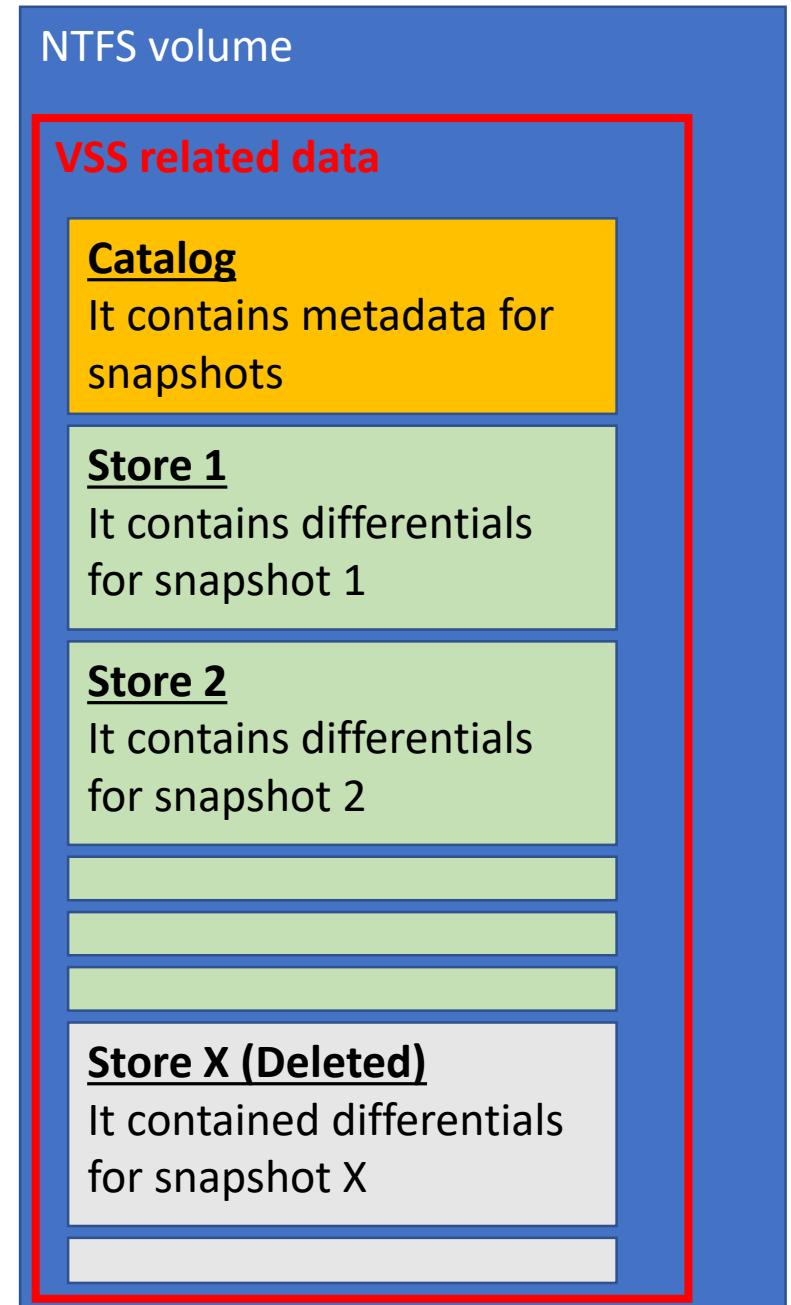
```
C:\shortcuts\09_RecoveringDataKeywordSearch\tools>vshadowinfo.exe -o 344981504 X:\EWF1
vshadowinfo 20180403
```

Volume Shadow Snapshot information:  
Number of stores: 0

There are no VSS snapshots. It must have been deleted by the ransomware.

# How We Carve The Deleted VSS

- We will access deleted VSS snapshots by the following procedures.
  1. We will run vss\_carver to recover deleted Store data. In addition, vss\_carver also generates a catalog entry for recovered Store data at the same time.
  2. We might use vss\_catalog\_manipulator to tweak the generated Catalog data if necessary.
  3. We would execute enhanced vshadowmount to present all VSS snapshots including recovered ones as raw disk images.

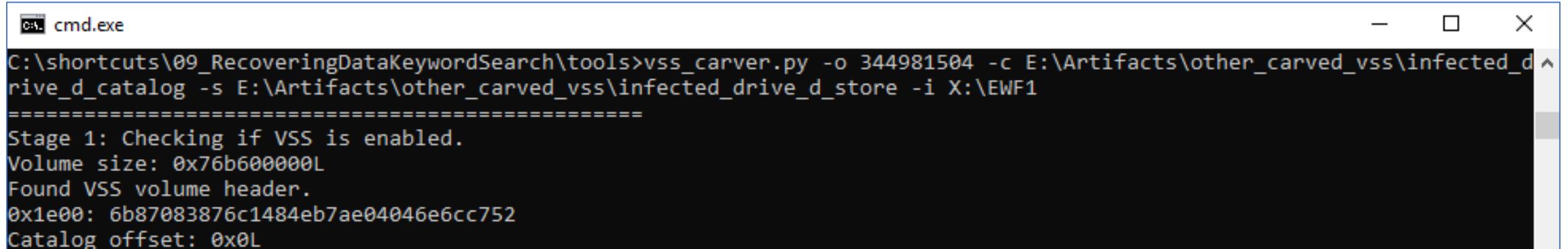


# Practice Exercise 2: Recovering Deleted VSS Snapshots (6)

- Find deleted VSS snapshots in order to restore the encrypted file (1)
  - The following command carves deleted VSS snapshots. **We prepared the result in advance** because it takes a long time. So **you do not need to execute it now**.

```
vss_carver.py -o 344981504 -c infected_drive_d_catalog -s infected_drive_d_store -i  
X:\EWF1
```

- The options means:
  - -i: Path to the target disk image file.
  - -o: Offset of the target volume to carve VSS snapshots.
  - -s: Path to the carved store data.
  - -c: Path to the generated catalog data.



```
C:\shortcuts\09_RecoveringDataKeywordSearch\tools>vss_carver.py -o 344981504 -c E:\Artifacts\other_carved_vss\infected_drive_d_catalog -s E:\Artifacts\other_carved_vss\infected_drive_d_store -i X:\EWF1  
=====  
Stage 1: Checking if VSS is enabled.  
Volume size: 0x76b600000L  
Found VSS volume header.  
0x1e00: 6b87083876c1484eb7ae04046e6cc752  
Catalog offset: 0x0L
```

# Practice Exercise 2:

## Recovering Deleted VSS Snapshots (7)

- Find deleted VSS snapshots in order to restore the encrypted file (2)
  - We were able to get the carved Store and updated Catalog by executing `vss_carver`. They are saved in the following folder.

Name	
	<code>infected_drive_d_catalog</code>
	<code>infected_drive_d_store</code>

# Practice Exercise 2: Recovering Deleted VSS Snapshots (8)

- Find deleted VSS snapshots in order to restore the encrypted file (3)
  - We are able to confirm carved VSS snapshots with following command.

Prepared "Catalog" that was generated by vss\_carver.

```
py -3 vss_catalog_manipulator.py list E:\Artifacts\other_carved_vss\infected_drive_d_catalog
```

```
[0] Enable, Date: 2019-10-04 02:06:23, GUID: 21608282-08e6-e911-8f9c-705681abad74
```

This is the carved VSS snapshot!

- This command lists both carved snapshots and existing snapshots at the same time.
- vss\_carver can recover only "Store" data of VSS snapshots, and it generates a new "Catalog" for carved "Store" with certain timestamps and GUIDs based on values of the current ones. Generated timestamps are set an hour earlier than the earliest one or the current time, and are set when they are found. Sometimes you need to modify these timestamps in order to correct the order of carved snapshots. vss\_catalog\_manipulator is also capable of modifying them.
- We can estimate creation time of carved snapshots from timestamps of files contained in them.

# Practice Exercise 2: Recovering Deleted VSS Snapshots (9)

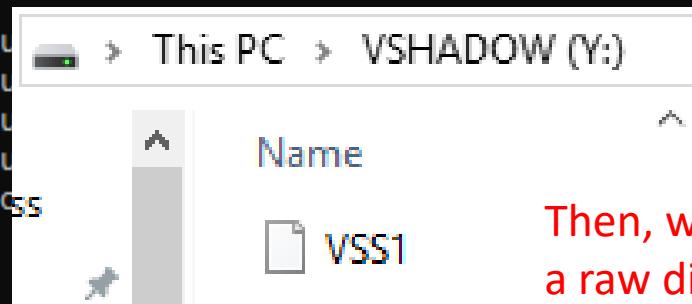
- Mount the carved VSS snapshot (1)
  - Convert the VSS snapshot into a raw disk image dynamically by the following command.  
Enhanced version of vhadoopmount use carved "Store" and generated "Catalog" to do that.

```
vshadowmount -o 344981504 -c E:\Artifacts\other_carved_vss\infected_drive_d_catalog  
-s E:\Artifacts\other_carved_vss\infected_drive_d_store X:\EWF1 y: Enter this in a single line.
```

↓ Prepared "Store" file      ↑ Prepared "Catalog" file

```
cmd.exe - vshadowmount -o 344981504 -c E:\Artifacts\other_carved_vss\infected_drive_d_catalog -s E:\Artifacts\other_carved_vss\infected_drive_d_sto...  
C:\shortcuts\09_RecoveringDataKeywordSearch\tools>vshadowmount -o 344981504 -c E:\Artifacts\other_carved_vss\infected_drive_d_catalog -s E:\Artifacts\other_carved_vss\infected_drive_d_store X:\EWF1 y:  
vshadowmount 20180403
```

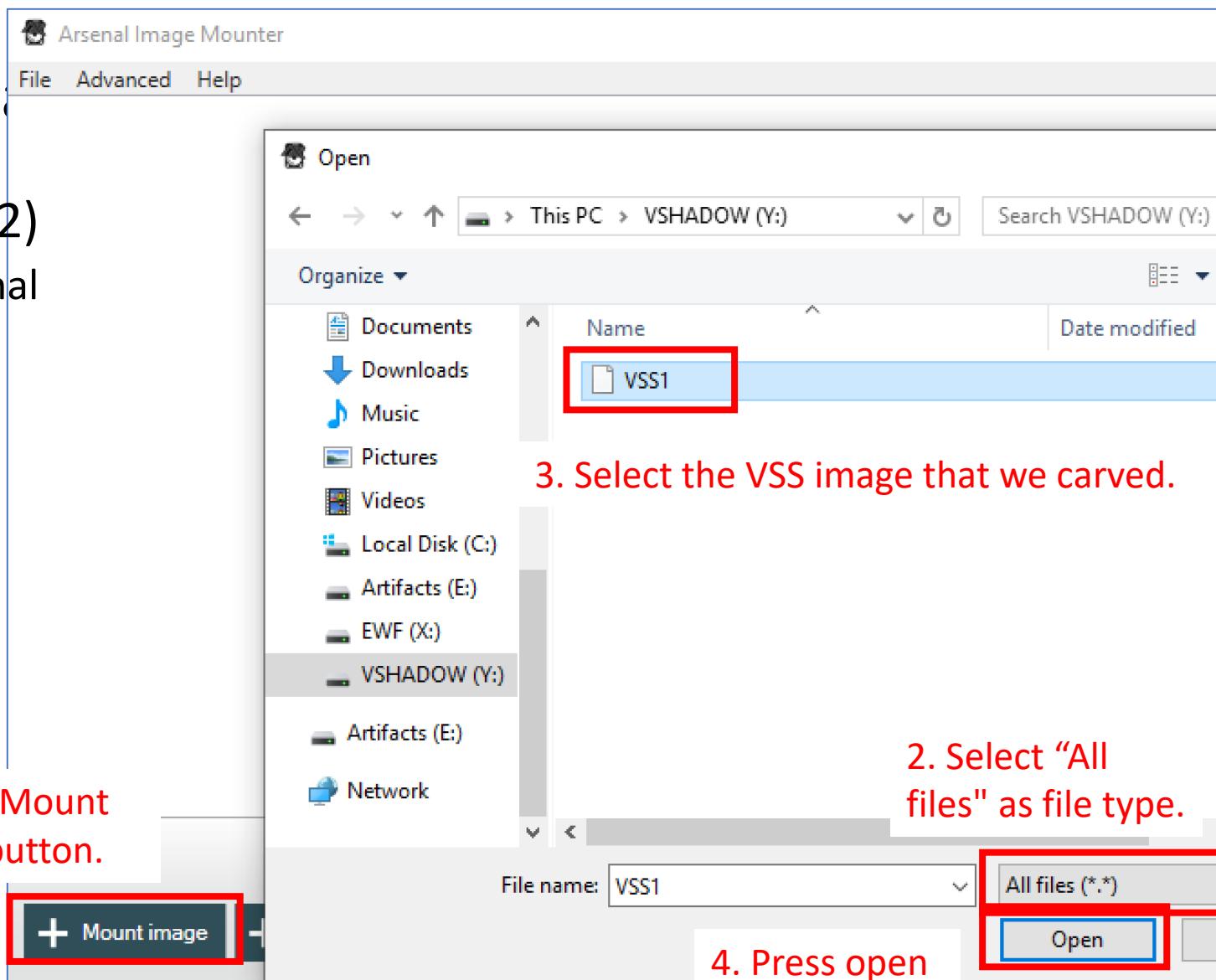
```
vshadowmount_dokan_CreateFile: unsupported path: \autoru  
vshadowmount_dokan_CreateFile: unsupported path: \autoru  
vshadowmount_dokan_CreateFile: unsupported path: \autoru  
vshadowmount_dokan_CreateFile: unsupported path: \AutoRu  
vshadowmount_dokan_CreateFile: unsupported path: \desktop  
vshadowmount_dokan_CreateFile: unsupported path: \*.  
vshadowmount_dokan_CreateFile: unsupported path: \*.  
vshadowmount_dokan_CreateFile: unsupported path: \*.  
vshadowmount_dokan_CreateFile: unsupported path: \*.  
vshadowmount_dokan_CreateFile: unsupported path: \a*.
```



Then, we can see the snapshot as a raw disk image "VSS1".

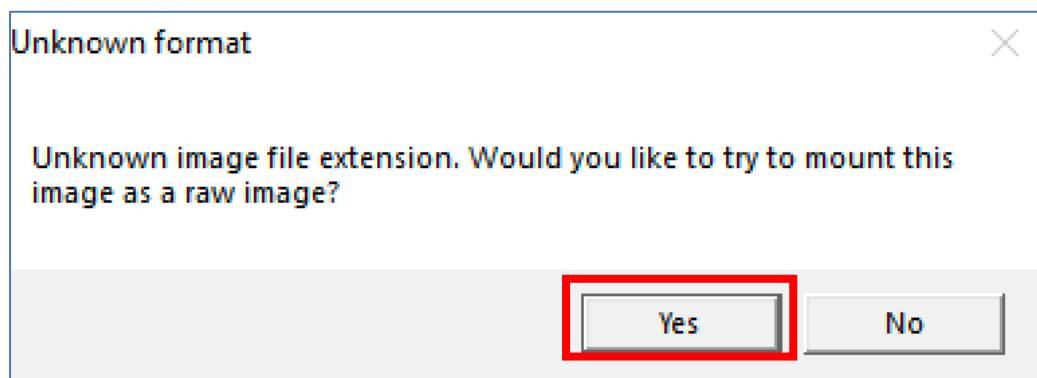
# Practice Exercise 2: Recovering Deleted VSS Sn

- Mount the carved VSS snapshot (2)
  - Mount the carved image with Arsenal Image Mounter.

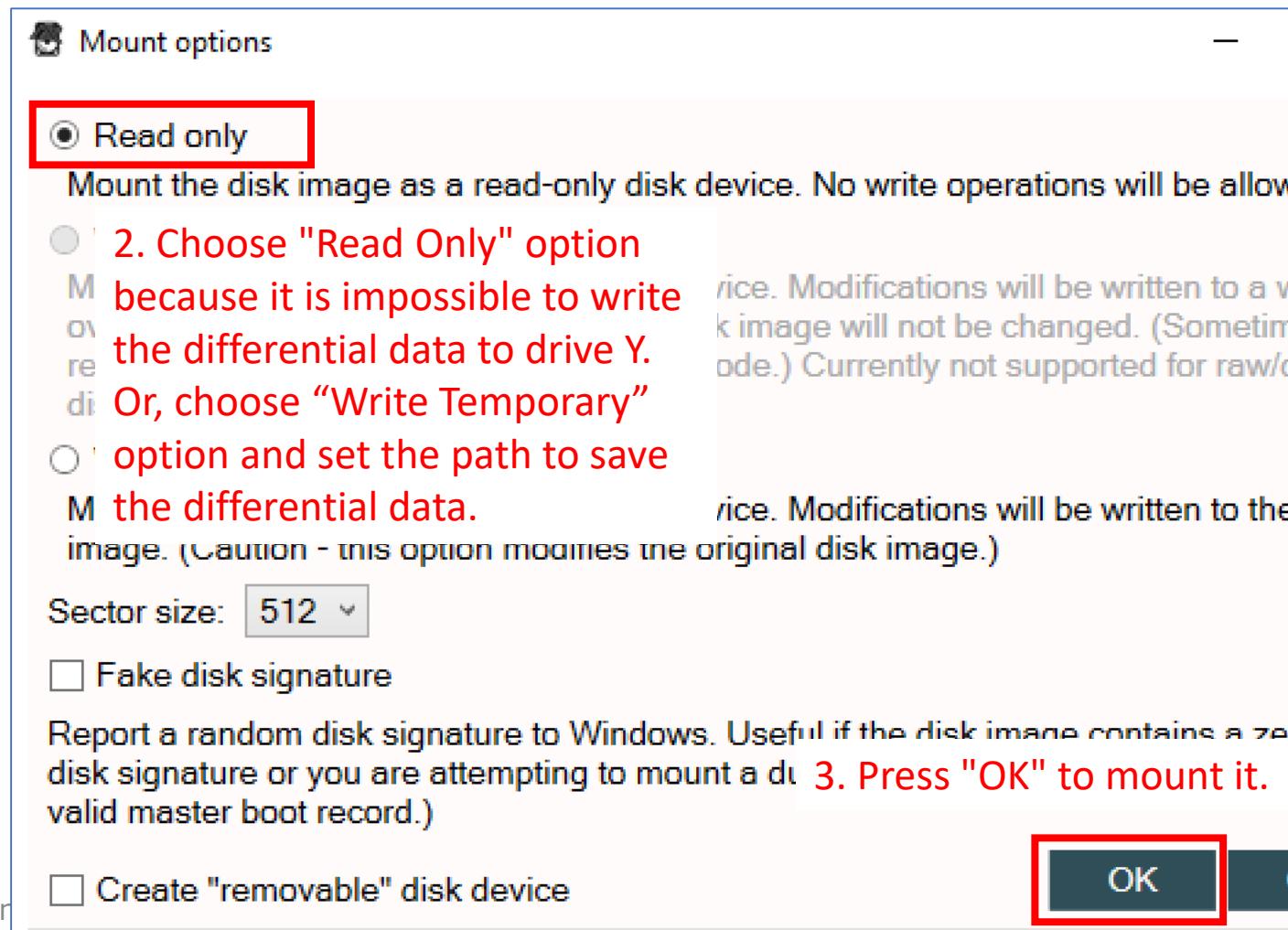


# Practice Exercise 2: Recovering Deleted VSS Snapshots (11)

- Mount the carved VSS snapshot (3)
  - Mount the carved image with Arsenal Image Mounter. (Cont.)

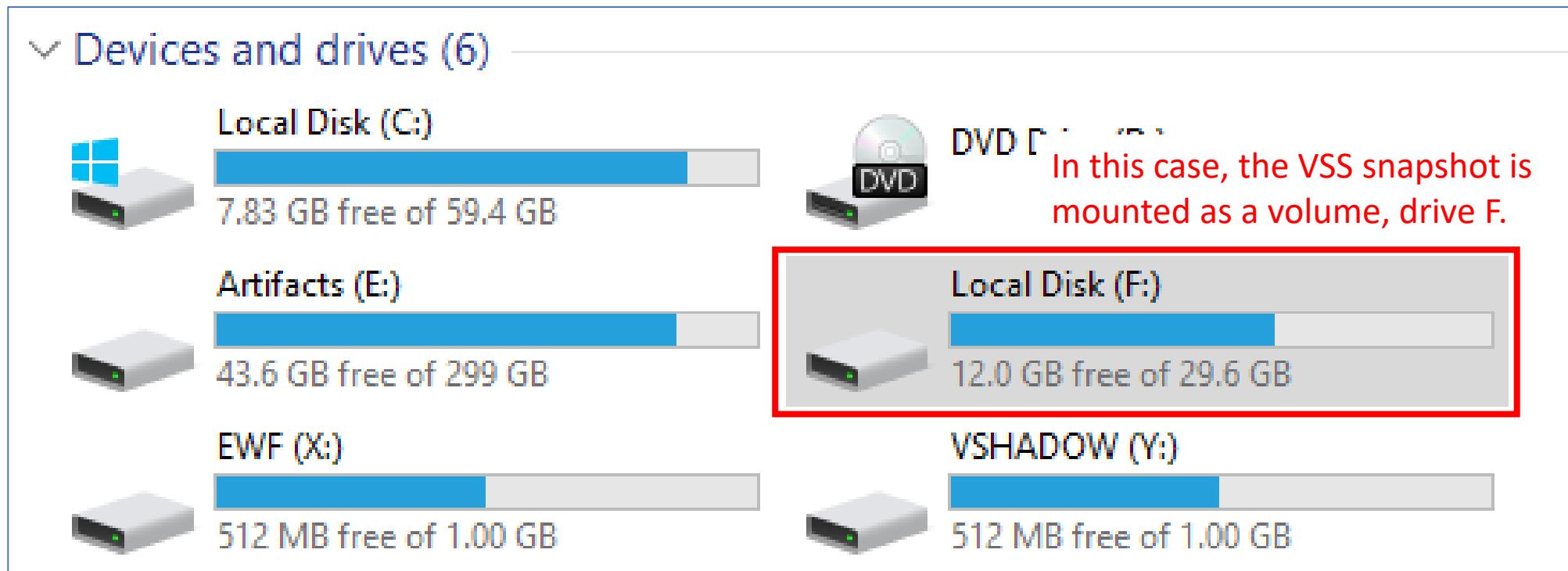


1. Press Yes on  
this dialog.



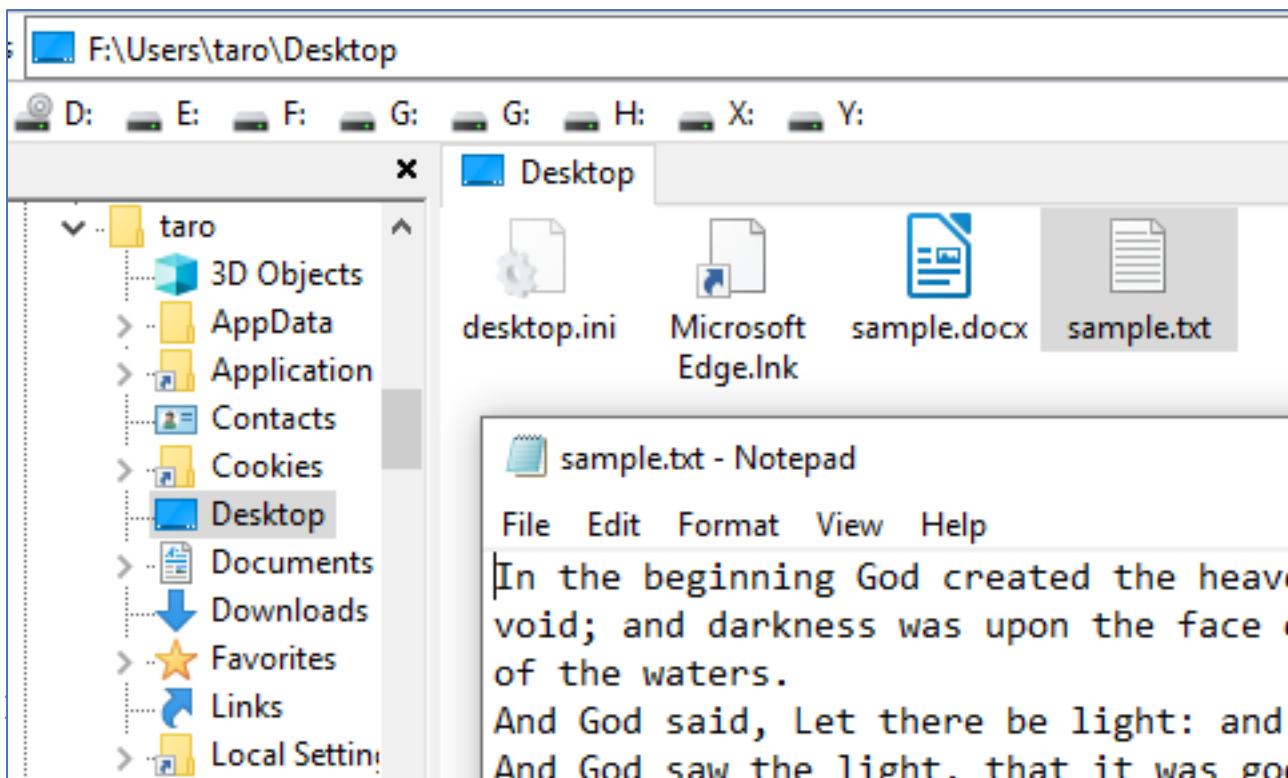
# Practice Exercise 2: Recovering Deleted VSS Snapshots (12)

- Mount the carved VSS snapshot (4)
  - Mount the carved image with Arsenal Image Mounter. (Cont.)



# Practice Exercise 2: Recovering Deleted VSS Snapshots (13)

- Finally, you can get the original files in the carved VSS snapshot.
- You can confirm that the txt file is restored. However, the docx file is not restored properly. Carving data is not always successful.
- Note: In this case, you can use Explorer++ to open Taro's %USERPROFILE% folder without modifying the volume.  
Explorer++ is useful to explore volumes mounted as “read-only”.



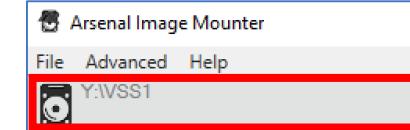
# Practice Exercise 2:

## Recovering Deleted VSS Snapshots (14)

- Note:
  - The data contained in deleted VSS snapshots might be overwritten with encrypted contents and other general files. Therefore, carving deleted VSS snapshots is not always success.
  - When a file is updated/deleted, VSS saves the differentials of the original file and updated one. Thus, files that have never updated are not stored in VSS snapshots.

# Practice Exercise 2: Recovering Deleted VSS Snapshots (15)

- Unmount the carved VSS snapshot
  - Quit Event Log Explorer and close the Explorer window.
  - Then, unmount the drive F on Arsenal Image Mounter



1. Select the image.



2. Press Remove Button.

# Practice Exercise 2: Recovering Deleted VSS Snapshots (16)

- Unmount all
  - Press "Ctrl+c" on the vshadowmount's prompt.

```
cmd.exe - vshadowmount -o 344981504 -c E:\Artifacts\other_carved_vss\infected_drive_d_catalog -s E:\Artifacts\other_carved_vss\infected_drive_d_sto...
C:\shortcuts\09_RecoveringDataKeywordSearch\tools>vshadowmount -o 344981504 -c E:\Artifacts\other_carved_vss\infected_drive_d_catalog -s E:\Artifacts\other_carved_vss\infected_drive_d_store X:\EWF1 y:
vshadowmount 20180403

vshadowmount_dokan_CreateFile: unsupported path: \autorun.inf.
vshadowmount_dokan_CreateFile: unsupported path: \autorun.inf.
vshadowmount_dokan_CreateFile: unsupported path: \autorun.inf.
vshadowmount_dokan_CreateFile: unsupported path: \AutoRun.inf.
vshadowmount_dokan_CreateFile: unsupported path: \desktop.ini.
```

Press "Ctrl+c" keys and quit vshadowmount.exe.

- Finally, press "Ctrl+c" on the ewfmount's prompt.

```
cmd.exe - ewfmount.exe E:\Artifacts\other_E01\infected_drive_d.E01 X:
C:\shortcuts\09_RecoveringDataKeywordSearch\tools>ewfmount.exe E:\Artifacts\other_E01\infected_drive_d.E01 X:
ewfmount 20190317

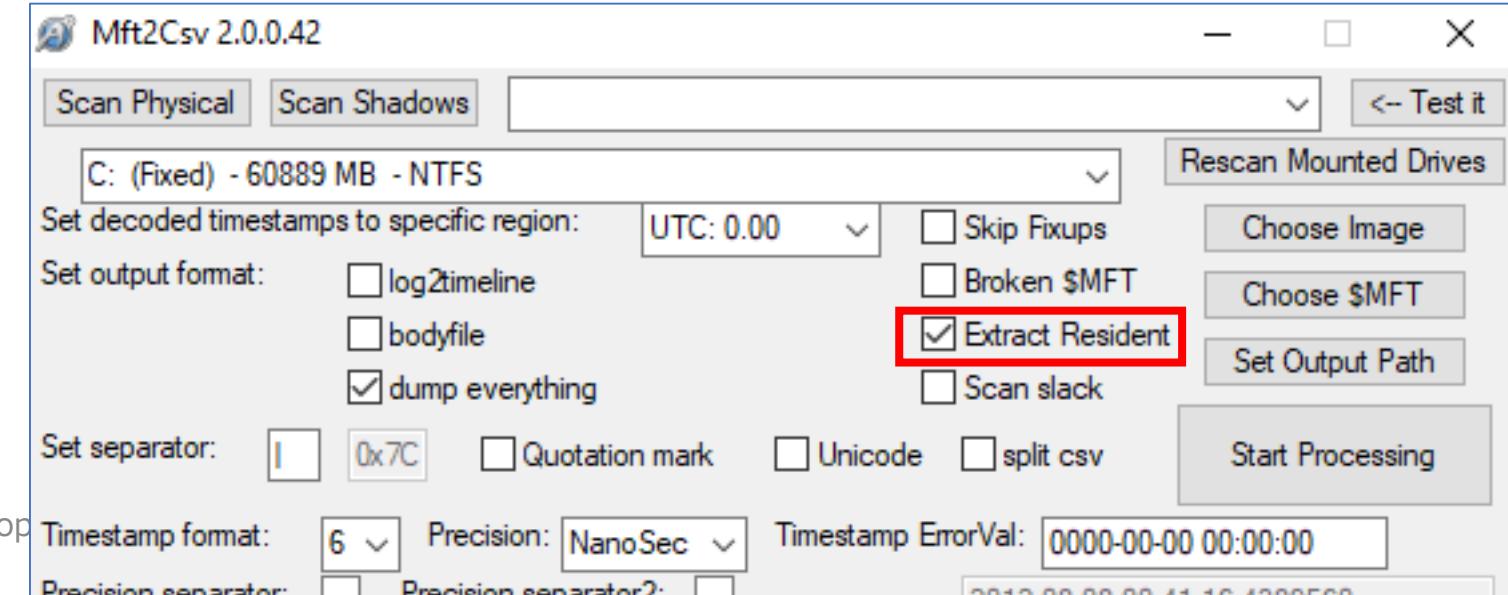
mount_dokan_CreateFile: unable to retrieve file entry for path: \a
mount_dokan_CreateFile: unable to retrieve file entry for path: \autorun.inf...
mount_dokan_CreateFile: unable to retrieve file entry for path: \autorun.inf.
mount_dokan_CreateFile: unable to retrieve file entry for path: \autorun.inf.
```

Press "Ctrl+c" keys and quit ewfmount.exe.

# Resident Data in MFT Records

# Resident Data in MFT Records

- As we already mentioned in Timeline Analysis chapter, Windows does not delete data immediately when files are deleted. Windows just flags the files' entries as "deleted" in Master File Table.
- Therefore, we might be able to extract deleted files if the entries exist.
  - Sometimes this is categorized as one of the data carving techniques.
  - For example, you can extract resident files from \$MFT by executing Mft2Csv with “Extract Resident” option.



# Data Carving and Keyword Search

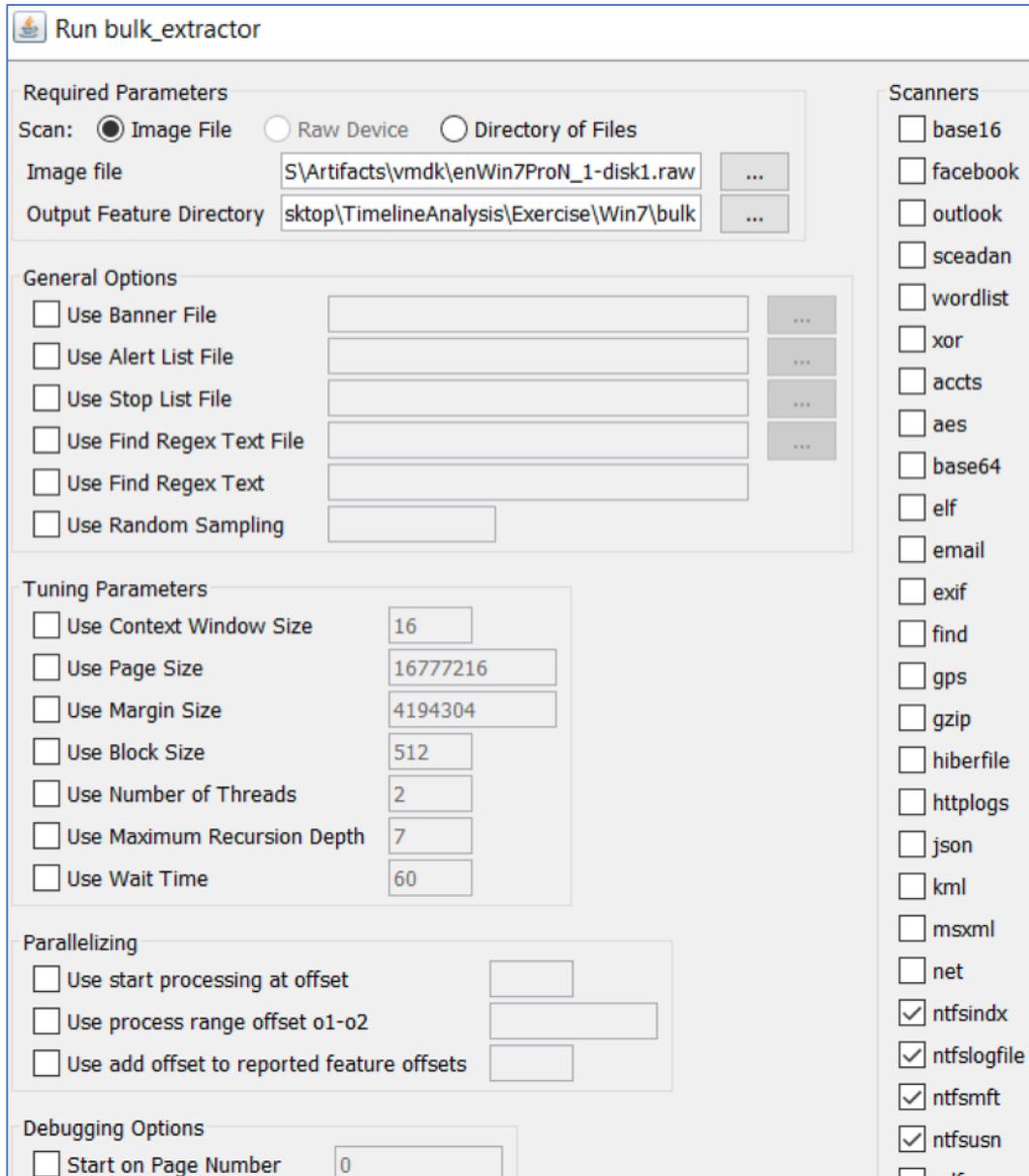
# Data Carving

# Data Carving

- Carving is to find and extract a specific data structure from large data chunk such as disk images, memory images, swap files, unallocated spaces, file slack, and so on.
- We often use this method to recover deleted files from unallocated space of acquired disk images.
- In order to find a specific data structure, we typically use file magic, signatures, and other characteristics of their format.

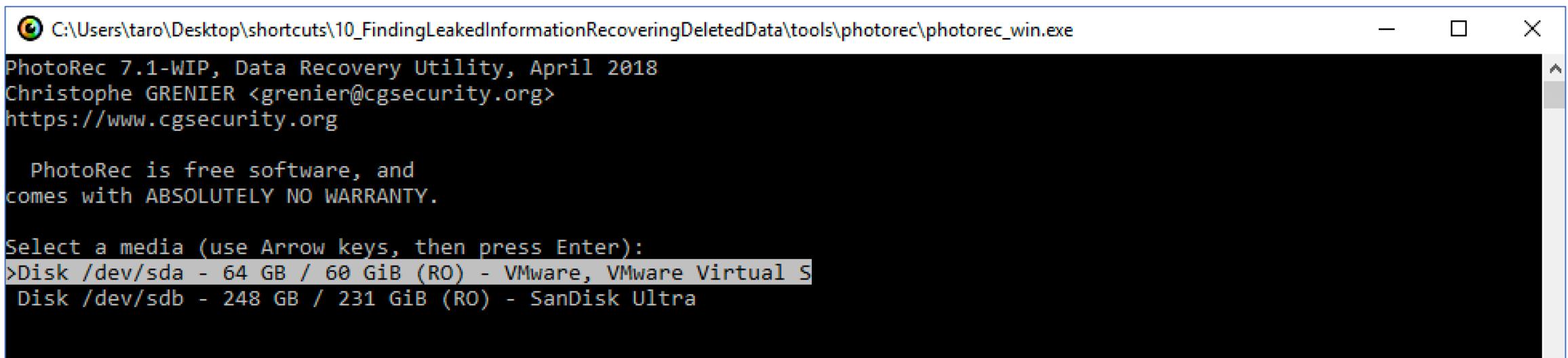
# Data Carving Tools (1)

- Bulk Extractor with Record Carving
  - It is an enhanced version from the 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.



# Data Carving Tools (2)

- PhotoRec
  - It is a file recovering tool.
  - It supports some popular file systems such as NTFS, ext2/3/4, HFS+, and so on.  
It can carve files with awareness for file system format.
  - It is capable of making custom signature.
  - It requires the target volume to be mounted.



The screenshot shows a Windows command-line window titled 'C:\Users\taro\Desktop\shortcuts\10\_FindingLeakedInformationRecoveringDeletedData\tools\photorec\photorec\_win.exe'. The window displays the PhotoRec 7.1-WIP Data Recovery Utility, version April 2018, developed by Christophe GRENIER. It includes the URL <https://www.cgsecurity.org>. A note states that PhotoRec is free software and comes with ABSOLUTELY NO WARRANTY. The user is prompted to 'Select a media (use Arrow keys, then press Enter)'. Two disk options are listed: 'Disk /dev/sda - 64 GB / 60 GiB (RO) - VMware, VMware Virtual S' and 'Disk /dev/sdb - 248 GB / 231 GiB (RO) - SanDisk Ultra'. The 'Disk /dev/sda' option is highlighted with a blue border.

```
C:\Users\taro\Desktop\shortcuts\10_FindingLeakedInformationRecoveringDeletedData\tools\photorec\photorec_win.exe

PhotoRec 7.1-WIP, Data Recovery Utility, April 2018
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

PhotoRec is free software, and
comes with ABSOLUTELY NO WARRANTY.

Select a media (use Arrow keys, then press Enter):
>Disk /dev/sda - 64 GB / 60 GiB (RO) - VMware, VMware Virtual S
Disk /dev/sdb - 248 GB / 231 GiB (RO) - SanDisk Ultra
```

# Data Carving Tools (3)

- **Foremost**

- It is an old simple file carving tool.
- It is capable of using regex custom signatures.
- It can carve from raw images. It can also carve from memory images and swap files.

```
cmd.exe
C:\shortcuts\12_RecoveringDataKeywordSearch\tools\foremost>foremost.exe -h
foremost version 1.5.7 by Jesse Kornblum, Kris Kendall, and Nick Mikus.
$ foremost [-v|-V|-h|-T|-Q|-q|-a|-w|-d] [-t <type>] [-s <blocks>] [-k <size>]
      [-b <size>] [-c <file>] [-o <dir>] [-i <file>]

-V  - display copyright information and exit
-t  - specify file type. (-t jpeg,pdf ...)
-d  - turn on indirect block detection (for UNIX file-systems)
-i  - specify input file (default is stdin)
-a  - Write all headers, perform no error detection (corrupted files)
-w  - Only write the audit file, do not write any detected files to the disk
-o  - set output directory (defaults to output)
-c  - set configuration file to use (defaults to foremost.conf)
-q  - enables quick mode. Search are performed on 512 byte boundaries.
```

# Data Carving Tools Comparison Table

- Each tool has both strong and weak points.

	Bulk Extractor with Record Carving	PhotoRec	Foremost
Performance	Y		
File system awareness		Y	
Carving with memory images	Y		Y
Per record carving	Y		
Custom ascii search	Y	Y	Y
Custom ascii search with regex	Y		Y
Custom binary search		Y	Y
Custom binary search with regex			Y
Plugin extendable	Y		

# Scenario 1 Labs: Lab 1

To search pagefiles for the golden tickets with foremost

# Scenario 1 Labs: Lab 1

To search pagefiles for the golden tickets with foremost (1)

- Condition:
  - We have already known that the attacker could use the golden ticket on client-win10-1.
- Goal:
  - To search the pagefile for the golden ticket structure. The pagefile contains swap-outed data. If we could find it, it could be a clear evidence of golden ticket being used.

ex.kirbi	
00000000	76 84 00 00 05 6B 30 84 00 00 05 65 A0 84 00 00 v.....k0.....e....

This is a typical structure of the golden ticket.

Roughly speaking, "krbtgt" and the domain name are placed around the first part. The account name and timestamps are placed near the last. Then, the structure ends with the string "krbtgt" and the victim domain name.

Ref:

<http://blog.gentilkiwi.com/securite/mimikatz/golden-ticket-kerberos>

000004A0	65 74 A2 84 00 00 00 24	30 84 00 00 00 1E A0 84	et.....\$0.....
000004B0	00 00 00 03 02 01 01 A1	84 00 00 00 0F 30 84 00	.....0.....
000004C0	00 00 09 1B 07 61 54	0 00	aToyoda.....
000004D0	00 07 03 05 00 40 E0	1 18	@.....
000004E0	0F 32 30 31 38 30 32 32	5 31 51	20180228030351Z
000004F0	A6 84 00 00 00 11 18 0F	32 30 32 38 30 32 32 36	.....20280226
00000500	30 33 30 33 35 31 5A A7	84 00 00 00 11 18 0F 32	030351Z.....2
00000510	30 32 38 30 32 32 36 30	33 30 33 35 31 5A A8 84	0280226030351Z..
00000520	00 00 00 12 1B 10 6E 69	€ 74 6F	.....ninja-moto
00000530	72 73 2E 6E 65 74 A9 84	0u uu uu ss su 84 00 00	rs.net.....50...
00000540	00 2F A0 84 00 00 00 03	02 01 02 F1 C4 C0 00 00	./.....
00000550	20 30 84 00 00 00 1A 1B	06 6B 72 € 74 1B	"krbtgt"
00000560	10 6E 69 6E 6A 61 2D 6D	6F 74 6F 72 73 2E 6E 65	ninja-motors.ne
00000570	74	t	

Target account name

Creation time

Expiration time

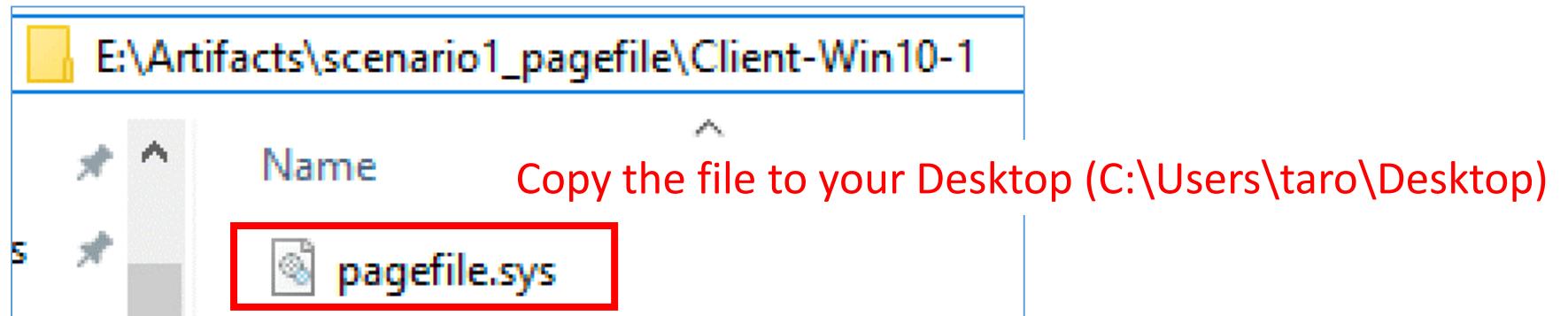
"krbtgt"

Victim domain name

# Scenario 1 Labs: Lab 1

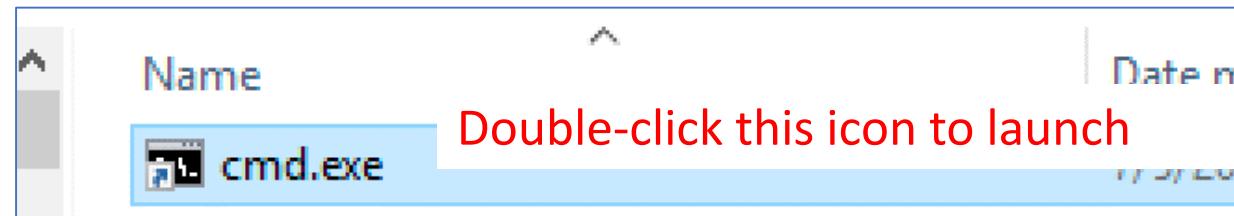
To search pagefiles for the golden tickets with foremost (3)

- First, copy the following pagefile to your Desktop. It is because foremost fails to process images on an exFAT formatted removable drives.



- Then, launch cmd.exe by double-clicking the shortcut icon.

Shortcuts\09\_RecoveringDataKeywordSearch



# Scenario 1 Labs: Lab 1

To search pagefiles for the golden tickets with foremost (4)

- Execute the following commands to carve with foremost!

```
cd foremost  
mkdir carved  
foremost.exe -c foremost-asn1.conf -o carved C:\Users\taro\Desktop\pagefile.sys
```

-c option is to set a configuration file

-o option is to set an output folder

The last argument is the target image

1	asn1	y	4096	\x76\x82??\x30\x82??\xa0\x03\x02\x01\x05\xa1\x03\x02\x01\x16
2	asn1_84	y	4096	\x76\x84????\x30\x84????\xa0\x84\x00\x00\x00\x03\x02\x01\x05\xa1\x84\x00\x00\x00\x03\x02\x01\x16

- This is the content of the configuration file. These are the signatures for the golden tickets' ASN.1 structure.
- We referenced the blog below to make these signatures.
  - <https://blog.didierstevens.com/2016/08/12/mimikatz-golden-ticket-dcsync/>

# Scenario 1 Labs: Lab 1

To search pagefiles for the golden tickets with foremost (5)

- You might get the following result.

The screenshot shows the output of the foremost tool. On the left, a terminal window displays the report for a file named `pagefile.sys`. The report includes details like the file path (`C:\Users\taro\Desktop\pagefile.sys`), start time (`Sun Jul 29 12:08:34 2018`), length (`Unknown`), and a table of extracted files. The table lists three files: `07044862.asn1`, `07044864.asn1`, and `07044867.asn1`, along with their sizes (4 KB) and offsets (3606969356, 3606970828, 3606972300). The number 3 is also present in the table. A red arrow labeled "The report" points from the terminal window to the top of the foremost interface. Another red arrow labeled "Carved files" points from the `audit.txt` entry in the report to the `asn1` folder in the file browser.

**foremost\carved**

Num	Name (bs=512)	Size	File Offset
0:	07044862.asn1	4 KB	3606969356
1:	07044864.asn1	4 KB	3606970828
2:	07044867.asn1	4 KB	3606972300

**foremost\carved\asn1**

Name
07044862.asn1
07044864.asn1
07044867.asn1

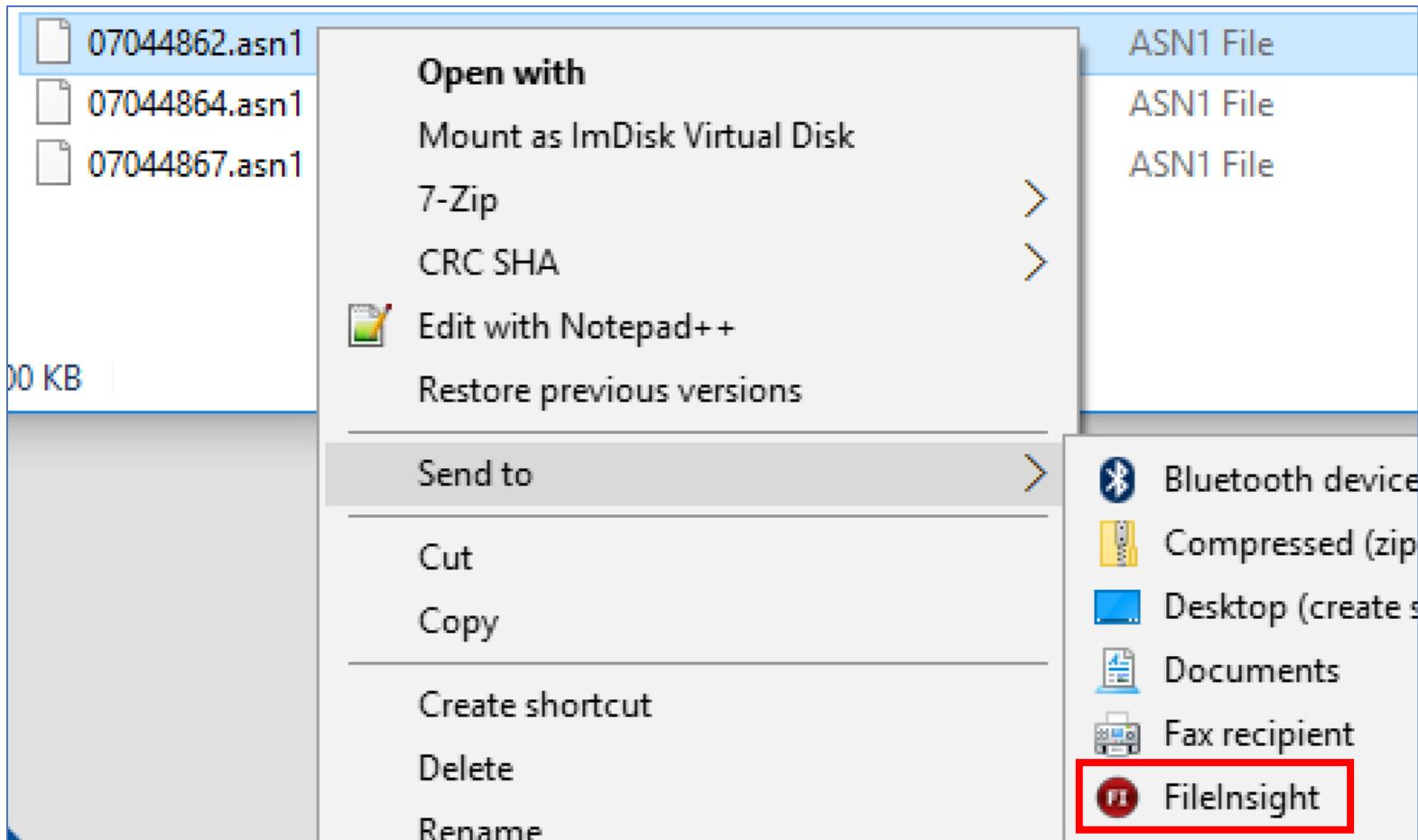
9 File: C:\Users\taro\Desktop\pagefile.sys  
10 Start: Sun Jul 29 12:08:34 2018  
11 Length: Unknown  
12  
13 Num Name (bs=512) Size File Offset  
14  
15 0: 07044862.asn1 4 KB 3606969356  
16 1: 07044864.asn1 4 KB 3606970828  
17 2: 07044867.asn1 4 KB 3606972300  
18 Finish: Sun Jul 29 12:08:47 2018  
19  
20 3 FILES EXTRACTED  
21  
22 asn1:= 3

iative Japan Inc.

# Scenario 1 Labs: Lab 1

To search pagefiles for the golden tickets with foremost (6)

- Let's open the carved files with a hex editor such as FileInsight.



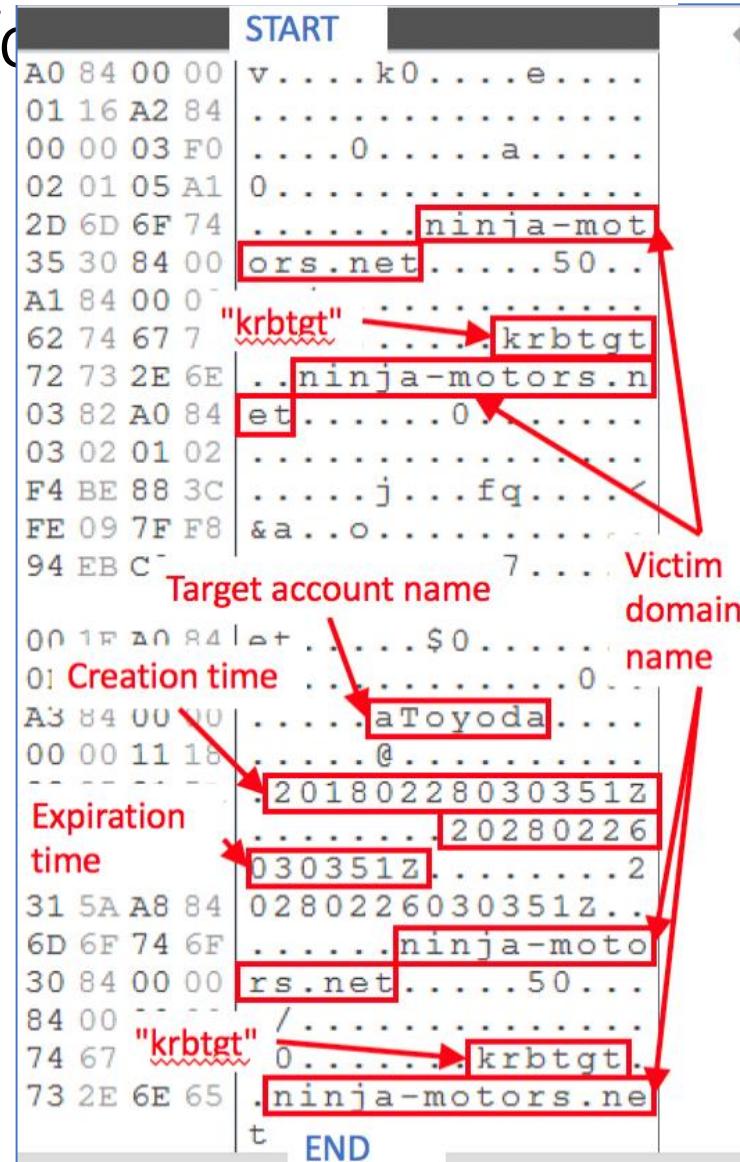
# Scenario 1 Labs: Lab 1

To search pagefiles for the golden tickets with forensic tools

- The first part looks very similar to the typical golden ticket structure we checked before. However, we cannot find the last part that should contain the target user name and timestamps. In addition, we cannot find the end of the structure that should contain strings "krbtgt" and its domain name placed after the target username and timestamps.
- Moreover, the domain name is written in all capital letters. Therefore, we cannot consider these carved files as the golden ticket.
- They might be parts of other credential data structures.

07044862 asn1	
00000000	76 82 05 A0 30 82 05 9C
00000010	01 16 A2 82 04 72 30 82
00000020	04 6E 61 82 04 6A 30 82
00000030	04 66 A0 03 02 01 05 A1
00000040	12 1B 10 4E 49 4E 4A 41
00000050	2D 4D 4F 54 4F 52 53 2E
00000060	4E 45 54 A2 25 30 23 A0
00000070	03 02 01 02 A1 1C 30 1A
00000080	1B 06 6B 72 62 74 67 74
00000090	1B 10 4E 49 4E 4A 41 2D
000000A0	4D 4F 54 4F 52 53 2E 4E
000000B0	45 54 A3 82 04 22 30 82
000000C0	04 1E A0 03 02 01 12 A1
000000D0	03 02 01 02 A2 82 04 10
000000E0	04 82 04 0C F8 F8 6E D5
000000F0	AB BE 0E D7 C5 A8 75 29
00000100	DE 17 8E 82 B2 CE 0F B0
00000110	3F 46 C3 0D E4 EC B6 D2
00000120	A3 C2 70 A3 56 62 AA 61
00000130	F0 89 DD 0C 0B 82 C1 F1
00000140	BE 65 C6 45 BC D4 04 1F

Typical golden ticket structure



# Scenario 1 Labs: Lab 1

To search pagefiles for the golden tickets with foremost (8)

- Unfortunately, we failed to find the structure of the golden tickets in this exercise.
- This kind of task does not always success. Also it takes a lot of time.
- However, by performing carving methods, we might recover deleted important files such as forged credentials, rotated program execution artifacts, stolen archives, and so on.

Per Record Carving

# Per Record Carving

- If it is impossible to restore the whole file body, we can use per record carving method.
  - It recovers data per record basis instead of per file basis. We typically use this method to recover the following records.
    - \$MFT records
    - \$INDEX records
    - \$USNJrnl:\$J records
    - Event logs
    - SQLite records
    - Web browser histories
- 
- We have already mentioned them in File System Timeline Analysis chapter.
- We have already mentioned them in Web Browser Forensics chapter.

# Per Record Carving Tools

- We have already leaned about these per record carving tools in Timeline Analysis Chapter.
  - Bulk Extractor with Record Carving
    - It supports recovering data of INDX, \$LOGFILE, \$MFT, and \$UsnJrnl:\$J per record basis.
    - We usually use USN Analytics to parse \$J records carved by this tool.
  - MftCarver
    - It supports recovering data of \$MFT per record basis.
    - We usually use MFT2Csv to parse \$MFT records carved by this tool.
  - IndxCarver
    - It supports recovering data of INDX per record basis.
    - We usually use Indx2Csv to parse INDX records carved by this tool.

# Per-Record Carving of Event Logs

# Event Log Record Carving 101 (1)

- What is Event Log Record Carving?
  - It is a method to recover event records from unallocated spaces or memory images by carving when you lost event logs for some reasons, such as:
    - Deleted by attackers
    - Time past
    - Misconfiguration
    - ...
- Why should we do this?
  - Carbon Black said, “*It’s fitting, given the destructive nature of contemporary attacks, that 72% of IR professionals saw counter IR in the form of destruction of logs*”, in their report. So, this task is very important for IR.
    - <https://www.carbonblack.com/global-incident-response-threat-report/november-2018/>
- Tools
  - EvtXtract
    - <https://github.com/williballenthin/EVTXtract>
  - Evtx Explorer/EvtxECmd + Bulk Extractor with Record Carving
    - <https://ericzimmerman.github.io/#!index.md>
    - [https://www.kazamiya.net/en/bulk\\_extractor-rec](https://www.kazamiya.net/en/bulk_extractor-rec)

# Event Log Record Carving 101 (2)

- We will use a similar technique we applied in Event Log Analysis section. In that section, we applied the per record carving methods to each file.
- In contrast, we will apply the method to a whole disk image, an unallocated space of a disk, or a pagefile.

# Algorithm of EvtXtract

---

1. Scan for chunk signatures ("ElfChnk")
  - check header for sane values ( $0x80 \leq \text{size} \leq 0x200$ )
  - verify checksums (header, data)
2. Extract records from valid chunks found in (1)
3. Extract templates from valid chunks found in (1)
4. Scan for record signatures
  - check header for sane values
  - extract timestamp
  - attempt to parse substitutions
  - attempt to decode substitutions into EID, other fields
5. Reconstruct records by reusing old templates with recovered substitutions

# Using EvtXtract

# Using EvtXtract (1)

- Open the folder below on Desktop of the analysis machine, then open command prompt.
  - Shortcuts\10\_FindingLeakedInformationRecoveringDeletedData\tools
- Convert a disk image to a raw disk image dynamically with ewfmount.

```
ewfmount E:\artifacts\scenario1_E01\Client-Win10-1_toyoda.E01 Y:
```

- Extract event logs with record carving using EvtXtract from the raw disk image

```
evtextract Y:\EWF1 > E:\artifacts\scenario1_eventlog\record_carving_with_evtextract\Client-Win10-1-evtextract.xml
```

Note that this command takes long time! We have already prepared the command result. The whole command needs to be written in a single line.

# Using EvtXtract (2)

- Next, fix two bugs in the result of EvtXtract XML file

```
py.exe fix_evtextract_xml.py  
E:\Artifacts\scenario1_eventlog\record_carving_with_evtextract\Client-Win10-1-evtextract.xml
```

Note that this command takes long time! We have already prepared the command result.

- Remove “\0”      The whole command needs to be written in a single line.
- Move Xml header into the top of the file
- Convert XML to CSV (strictly speaking, it is TSV though...) with our script.

```
py.exe xml_evtx_parse.py  
E:\artifacts\scenario1_eventlog\record_carving_with_evtextract\Client-Win10-1-  
evtextract.xml.fixed.xml >  
E:\artifacts\scenario1_eventlog\record_carving_with_evtextract\Client-Win10-1-  
evtextract.xml.fixed.xml.csv
```

Note that this command takes long time! We have already prepared the command result.  
The whole command needs to be written in a single line.

# Using EvtXtract (3)

- Then add a CSV header and sort lines.

```
add_header_and_sort.bat E:\Artifacts\scenario1_eventlog\record_carving_with_evtextract\Client-Win10-1-evtextract.xml.fixed.xml.csv
```

Note that this command takes long time! We have already prepared the command result.  
The whole command needs to be written in a single line.

- Now we are ready to check the results.
- Actually, if you execute “evtextract\_csv.bat”, you can perform the same process related to EvtXtract like the following.

```
evtextract_csv.bat Y:\EWF1 "" E:\Artifacts\scenario1_eventlog\record_carving_with_evtextract Client-Win10-1-evtextract
```

Note that this command takes long time! We have already prepared the command result.  
The whole command needs to be written in a single line.

# Scenario 1 Labs: Lab 2

Checking carved events

# Scenario 1 Labs: Lab 2

## Checking carved events (1)

- Condition:
  - We already know that the task "SyS" was registered by the attackers on Client-win10-1 at Mar 15 6:53PM from Event Log analysis.
  - However, we could not find the task on the current disk image of Client-win10-1.
- Goal:
  - To find out when the task was deleted by checking carved event logs.

# Scenario 1 Labs: Lab 2

## Checking carved events (2)

- After executing EvtXtract and the bat files on client-win10-1, we got many carved event logs. The bat would output parsed csv files for each Event ID.

Artifacts (E:) > Artifacts > scenario1_eventlog > record_carving_with_evtextract	
	Name
	Client-Win10-1-evtxtract.xml.fixed.xml.100_Microsoft-Windows-TaskScheduler_Operational.csv
	Client-Win10-1-evtxtract.xml.fixed.xml.110_Microsoft-Windows-TaskScheduler_Operational.csv
	Client-Win10-1-evtxtract.xml.fixed.xml.140_Microsoft-Windows-TaskScheduler_Operational.csv
	Client-Win10-1-evtxtract.xml.fixed.xml.141_Microsoft-Windows-TaskScheduler_Operational.csv
	Client-Win10-1-evtxtract.xml.fixed.xml.400_Windows PowerShell.csv
	Client-Win10-1-evtxtract.xml.fixed.xml.403_Windows PowerShell.csv
	Client-Win10-1-evtxtract.xml.fixed.xml.1014_System.csv
	Client-Win10-1-evtxtract.xml.fixed.xml.1102_Microsoft-Windows-TerminalServices-RDPClient_O...

# Scenario 1 Labs: Lab 2

## Checking carved events (3)

- Open the result with LibreOffice.
  - ID 141 is a deletion event of a scheduled task.
  - Let's open “Client-Win10-1-evtxtract.xml.fixed.xml.141\_Microsoft-Windows-TaskScheduler\_Operational.csv” on “E:\artifacts\scenario1\_eventlog\record\_carving\_with\_evtxtract”.

local_time	EID	Category	User	Computer	Data_Name
2018-03-13 13:46:35.673298	141	Microsoft-W	S-1-5-18	client-win10-1.ninja-motors.net	\Adobe Acrob
2018-03-23 13:26:30.447702	141	Microsoft-W	S-1-5-18	client-win10-1.ninja-motors.net	\SyS
2018-03-14 23:03:18.748373	141	Microsoft-W	S-1-5-18	client-win10-1.ninja-motors.net	\Microsoft\Wi

You can get the deletion time of the task “\SyS” that had dumped credentials in memory of Client-Win10-1.

# Scenario 1 Labs: Lab 2

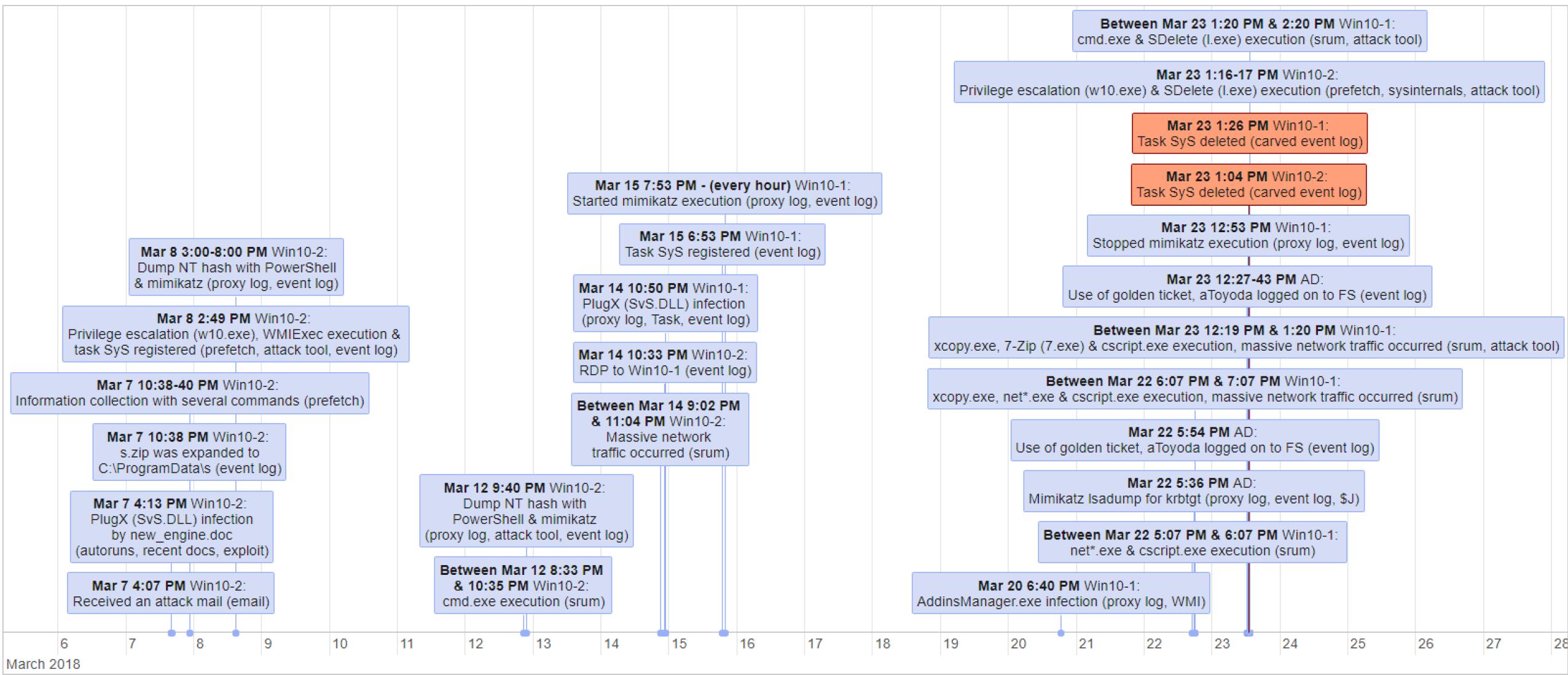
## Checking carved events (4)

- How about Client-Win10-2?
  - Let's open “Client-Win10-2-evtxtract.xml.fixed.xml.141\_Microsoft-Windows-TaskScheduler\_Operational.csv” on “E:\artifacts\scenario1\_eventlog\record\_carving\_with\_evtxtract”.
  - You can also find out the similar log.

local_time	EID	Category	User	Computer	Data	Name_TaskName
2018-03-23 13:04:31.450266	141	Microsoft-Win	S-1-5-18	client-win10-2.ninja-motors.net	\SyS	
2018-03-19 19:48:50.753693	141	Microsoft-Win	S-1-5-18	client-win10-2.ninja-motors.net	\Microsoft Office 15 Sync M	

You can get the deletion time of the task “\SyS” that had dumped credentials in memory of Client-Win10-2.

# Scenario 1 Labs: Lab 2



# Scenario 1 Labs: Lab 3

Golden ticket detection in carved events

# Scenario 1 Labs: Lab 3

## Golden ticket detection in carved events (1)

- Open cmd.exe in “shortcuts\09\_RecoveringDataKeywordSearch”.
- Type the command below.

```
py.exe golden_tickets.py  
E:\artifacts\scenario1_eventlog\record_carving_with_evtextract\AD-Win2016-  
evtextract.xml.fixed.xml.csv
```

Enter this line in a single line.

# Mimikatz Golden/Silver Tickets Detection Method

- We could detect Golden/Silver Tickets by enumerating user names and SIDs in all security logs and checking duplicate user names corresponding to each SID.

SID	User
S-1-2-21-XXXXX-500	Administrator
S-1-2-21-XXXXX-1000	Alice
S-1-2-21-XXXXX-1001	Bob
S-1-2-21-XXXXX-1002	Carol
S-1-2-21-XXXXX-1003	Dave, Mallory
S-1-2-21-XXXXX-1004	Ellen
S-1-2-21-XXXXX-1005	Frank
...	...



Golden/Silver Ticket

```
C:\Users\taro\Desktop\shortcuts\10_FindingLeakedInformationRecoveringDeletedData\tools>py.exe golden_tickets.py ..\artif  
acts\scenario1_eventlog\record_carving_with_evtextract\AD-Win2016-evtxtract.xml.fixed.xml.csv  
found possible golden/silver tickets used (small chars in domain): 2018-03-23 20:07:35.207716 aToyoda ninja-motors.net 4  
624  
found possible golden/silver tickets used (small chars in domain): 2018-03-23 20:07:35.509317 aToyoda ninja-motors.net 4  
624  
found possible golden/silver tickets used (small chars in domain): 2018-03-23 21:52:08.118158 aToyoda ninja-motors.net 4  
624  
found possible golden/silver tickets used (small chars in domain): 2018-03-23 12:27:49.056072 aToyoda@ninja-motors.net n  
inj... 4769  
found possible golden/silver tickets used (small chars in domain): 2018-03-23 12:35:35.521643 aToyoda@ninja-motors.net n  
inj... 4769  
found possible golden/silver tickets used (small chars in domain): 2018-03-23 12:35:35.523720 aToyoda ninja-motors.net 4  
624  
found possible golden/silver tickets used (small chars in domain): 2018-03-23 12:35:35.709713 aToyoda ninja-motors.net 4  
624  
found possible golden/silver tickets used (small chars in domain): 2018-03-23 12:35:35.717196 aToyoda@ninja-motors.net n  
inj... 4769  
found possible golden/silver tickets used (small chars in domain): 2018-03-23 12:42:09.258982 aToyoda@ninja-motors.net n  
inj... 4769  
found possible golden/silver tickets used (small chars in domain): 2018-03-23 12:42:09.260756 aToyoda@ninja-motors.net n  
inj... 4769  
found possible golden/silver tickets used (small chars in domain): 2018-03-23 12:42:09.262184 aToyoda ninja-motors.net 4  
624  
found possible golden/silver tickets used (small chars in domain): 2018-03-23 12:43:27.659119 aToyoda ninja-motors.net 4  
624
```

We got many lower letters in the domain name  
in log entries, which implies golden tickets being  
used on AD-Win2016.

```
found possible golden/silver tickets used (small chars in domain): 2018-03-23 16:22:07.936146 aToyoda ninja-motors.net 4
624
found possible golden/silver tickets used (small chars in domain): 2018-03-23 16:22:09.397907 aToyoda ninja-motors.net 4
624
found possible golden/silver tickets used (small chars in domain): 2018-03-23 18:12:08.513659 aToyoda ninja-motors.net 4
624
found possible golden/silver tickets used (small chars in domain): 2018-03-23 18:12:09.386068 aToyoda ninja-motors.net 4
624
found possible golden/silver tickets used (small chars in domain): 2018-03-23 18:14:35.684368 aToyoda ninja-motors.net 4
624
found possible golden/silver tickets used (small chars in domain): 2018-03-23 18:14:35.983265 aToyoda ninja-motors.net 4
624
found possible golden/silver tickets used (small chars in domain): 2018-03-23 18:40:13.766752 aToyoda ninja-motors.net 4
624
found possible golden/silver tickets used (small chars in domain): 2018-03-23 18:40:13.946373 aToyoda ninja-motors.net 4
624
found possible golden/silver tickets used (small chars in domain): 2018-03-23 20:02:08.315937 aToyoda ninja-motors.net 4
624
found possible golden/silver tickets used (small chars in domain): 2018-03-23 20:02:09.169973 aToyoda ninja-motors.net 4
624
found possible golden/silver tickets used (small chars in domain): 2018-03-23 20:02:09.169973 aToyoda ninja-motors.net 4
624
found possible golden/silver tickets used (small chars in domain): 2018-03-23 20:02:09.169973 aToyoda ninja-motors.net 4
624
found possible golden/silver tickets used (duplicated sids): S-1-5-21-3671970501-3975728774-4289435121-1106 {'TOYODA',
ATOYODA'}
```

We also found a duplicated SID for user "toyoda".  
This also implies golden tickets being used.

# Another Record Carving Tool for Event Logs

Checking carved events with BulkExtractor with Record Carving + EvtxECmd (1)

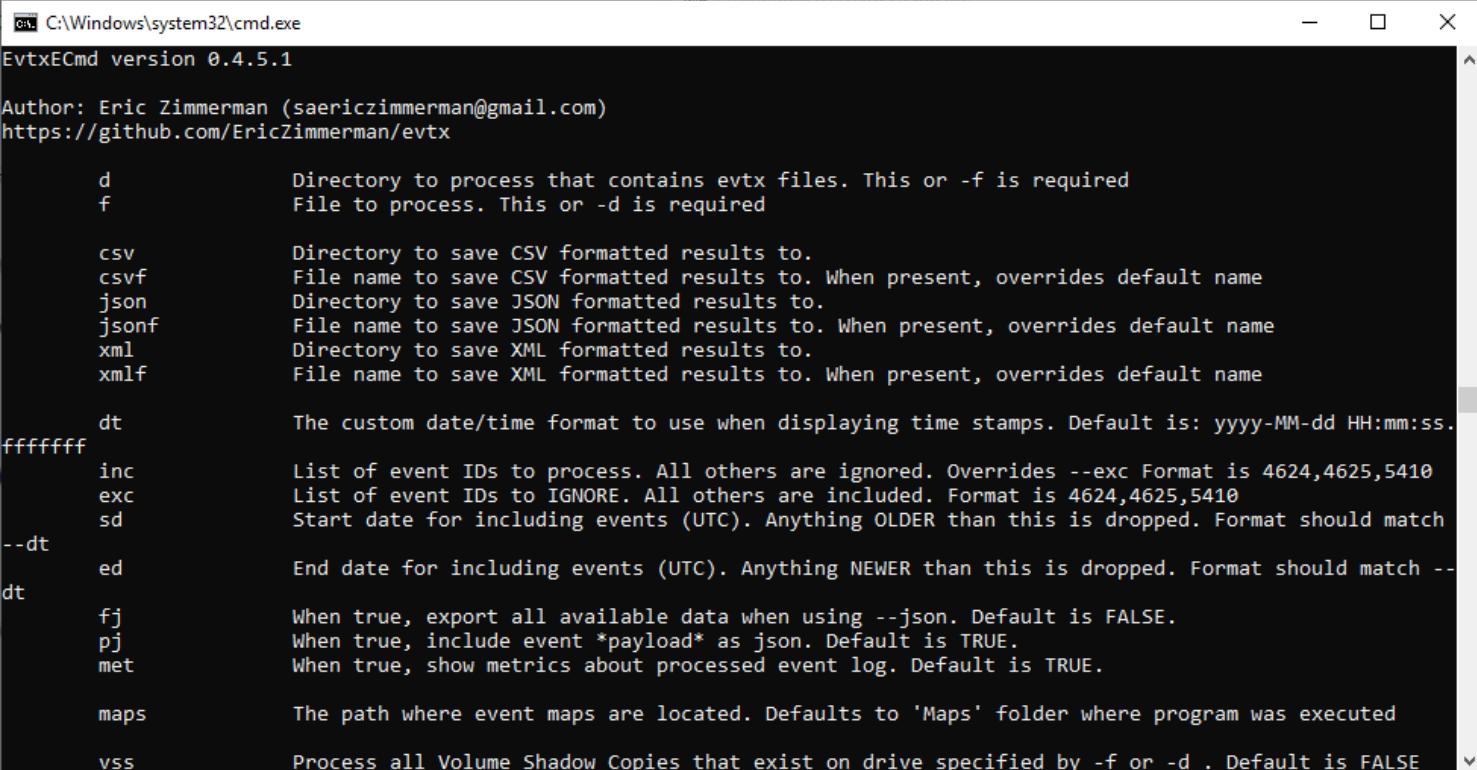
- You can carve event log records with “Bulk Extractor with Record Carving”. And you can analyze the carved records with “EvtxECmd”.
- This is another method of event log per-record carving.
- This is neither Lab nor Exercise.

# Another Record Carving Tool for Event Logs

Checking carved events with BulkExtractor with Record Carving + EvtxECmd (2)

- **EvtxECmd**

- This is a command line tool for parsing event logs, by Eric Zimmerman.
- It can parse Evtx format and save as CSV, JSON or XML format.
- You need to create a “map” file if you would like to see parsed CSV format, or you should see payload column that is written in the JSON format.



```
C:\Windows\system32\cmd.exe
EvtxECmd version 0.4.5.1

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/evt

d          Directory to process that contains evt files. This or -f is required
f          File to process. This or -d is required

csv        Directory to save CSV formatted results to.
csvf       File name to save CSV formatted results to. When present, overrides default name
json       Directory to save JSON formatted results to.
jsonf      File name to save JSON formatted results to. When present, overrides default name
xml        Directory to save XML formatted results to.
xmlf      File name to save XML formatted results to. When present, overrides default name

dt         The custom date/time format to use when displaying time stamps. Default is: yyyy-MM-dd HH:mm:ss.
ff        List of event IDs to process. All others are ignored. Overrides --exc Format is 4624,4625,5410
inc       List of event IDs to IGNORE. All others are included. Format is 4624,4625,5410
exc       Start date for including events (UTC). Anything OLDER than this is dropped. Format should match
sd        --dt
ed        End date for including events (UTC). Anything NEWER than this is dropped. Format should match --
dt
fj        When true, export all available data when using --json. Default is FALSE.
pj        When true, include event *payload* as json. Default is TRUE.
met      When true, show metrics about processed event log. Default is TRUE.

maps     The path where event maps are located. Defaults to 'Maps' folder where program was executed

vss      Process all Volume Shadow Copies that exist on drive specified by -f or -d . Default is FALSE
```

# Another Record Carving Tool for Event Logs

Checking carved events with BulkExtractor with Record Carving + EvtxECmd (3)

- First, in order to get the offset of the Windows partition, execute mmls command.

```
mmls E:\Artifacts\scenario1_E01\Client-Win10-2_honda.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	0001026047	0001024000	NTFS / exFAT (0x07)	
003: 000:001	0001026048	0104855551	0103829504	NTFS / exFAT (0x07)	
004: -----	0104855552	0104857599	0000002048	Unallocated	

Offset

Choose the big NTFS partition

# Another Record Carving Tool for Event Logs

Checking carved events with BulkExtractor with Record Carving + EvtxECmd (4)

- Second, execute blkls command to get unallocated space.

```
blkls -A -o 1026048 E:\Artifacts\scenario1_E01\Client-Win10-2_honda.E01 >  
E:\Artifacts\scenario1_eventlog\record_carving_with_BE\Client-Win10-2\unalloc.dat
```

You do not need to execute this command because the command takes long time and we have already prepared the image of unallocated space in the path above. When executing this command, enter the whole line in a single line.

# Another Record Carving Tool for Event Logs

Checking carved events with BulkExtractor with Record Carving + EvtxECmd (5)

- Third, execute `bulk_extractor` command with `evtx` scanner to carve event log records in the extracted unallocated space.

```
bulk_extractor -E evtx -o  
E:\Artifacts\scenario1_eventlog\record_carving_with_BE\Client-Win10-2\Carved  
E:\Artifacts\scenario1_eventlog\record_carving_with_BE\Client-Win10-2\unalloc.dat
```

You do not need to execute this command because the command takes long time and we have already prepared the image of unallocated space in the path above. When executing this command, enter the whole line in a single line.

# Another Record Carving Tool for Event Logs

Checking carved events with BulkExtractor with Record Carving + EvtxECmd (6)

- Lastly, in order to analyze the carved records, execute EvtxECmd command.

```
EvtxECmd.exe -d E:\Artifacts\scenario1_eventlog\record_carving_with_BE\Client-Win10-2\Carved --csv E:\Artifacts\scenario1_eventlog\record_carving_with_BE\Client-Win10-2
```

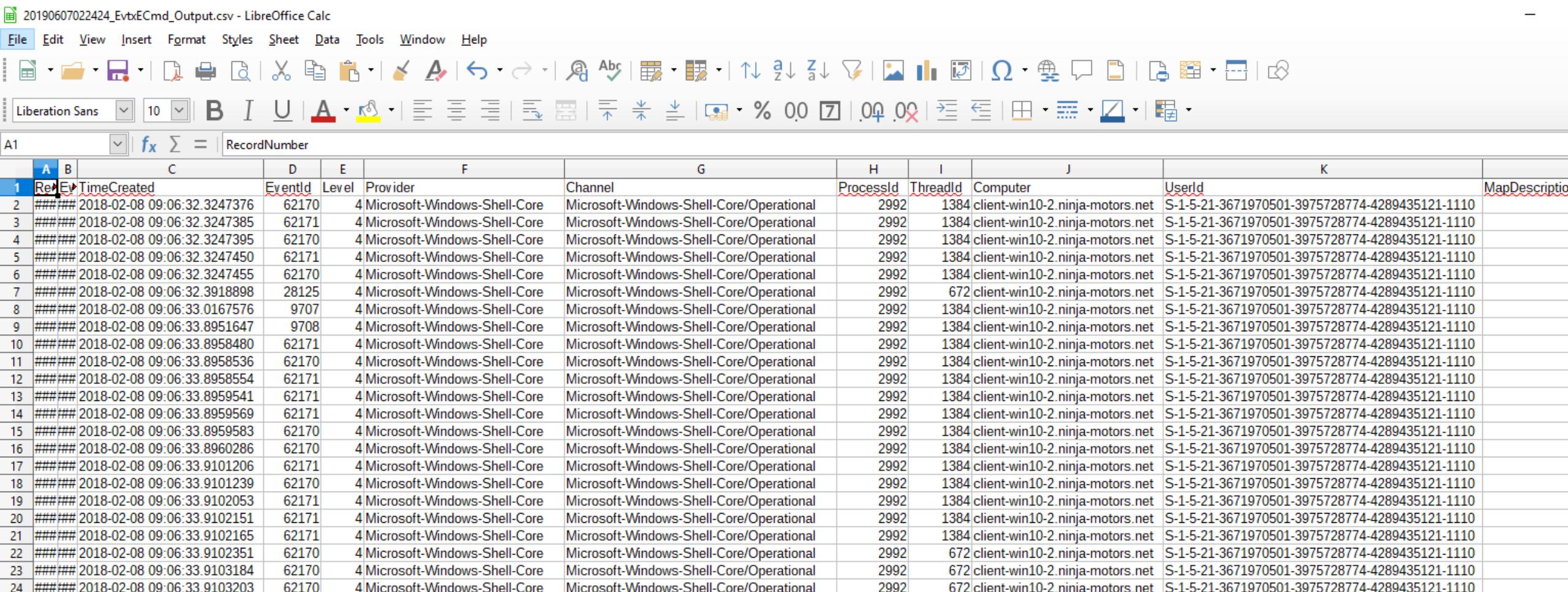
You do not need to execute this command because we have already prepared the result in the directory above.

When executing this command, enter the whole line in a single line.

# Another Record Carving Tool for Event Logs

Checking carved events with BulkExtractor with Record Carving + EvtxECmd (7)

- You can see the carved events.



The screenshot shows a LibreOffice Calc spreadsheet titled "20190607022424\_EvtxECmd\_Output.csv - LibreOffice Calc". The spreadsheet has a header row labeled "RecordNumber" and contains approximately 24 rows of event log data. The columns are labeled A through K. Column A is "RecordNumber", B is "EventId", C is "TimeCreated", D is "EventId", E is "Level", F is "Provider", G is "Channel", H is "ProcessId", I is "ThreadId", J is "Computer", K is "UserId", and L is "MapDescription". The data consists of Microsoft Windows Shell Core Operational events. The "TimeCreated" column shows dates from February 2018. The "Computer" column consistently lists "client-win10-2.ninja-motors.net". The "UserId" column shows S-1-5-21-3671970501-3975728774-4289435121-1110. The "MapDescription" column is empty. The "ThreadId" column contains values 1384 and 672. The "ProcessId" column contains values 2992 and 1384.

	A	B	C	D	E	F	G	H	I	J	K	L
1	RecordNumber	EventId	TimeCreated	EventId	Level	Provider	Channel	ProcessId	ThreadId	Computer	UserId	MapDescription
2	####	2018-02-08 09:06:32.3247376	62170	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
3	####	2018-02-08 09:06:32.3247385	62171	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
4	####	2018-02-08 09:06:32.3247395	62170	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
5	####	2018-02-08 09:06:32.3247450	62171	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
6	####	2018-02-08 09:06:32.3247455	62170	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
7	####	2018-02-08 09:06:32.3918898	28125	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	672	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
8	####	2018-02-08 09:06:33.0167576	9707	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
9	####	2018-02-08 09:06:33.8951647	9708	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
10	####	2018-02-08 09:06:33.8958480	62171	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
11	####	2018-02-08 09:06:33.8958536	62170	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
12	####	2018-02-08 09:06:33.8958554	62171	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
13	####	2018-02-08 09:06:33.8959541	62171	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
14	####	2018-02-08 09:06:33.8959569	62171	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
15	####	2018-02-08 09:06:33.8959583	62170	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
16	####	2018-02-08 09:06:33.8960286	62170	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
17	####	2018-02-08 09:06:33.9101206	62171	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
18	####	2018-02-08 09:06:33.9101239	62170	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
19	####	2018-02-08 09:06:33.9102053	62171	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
20	####	2018-02-08 09:06:33.9102151	62171	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
21	####	2018-02-08 09:06:33.9102165	62171	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	1384	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
22	####	2018-02-08 09:06:33.9102351	62170	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	672	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
23	####	2018-02-08 09:06:33.9103184	62170	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	672	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		
24	####	2018-02-08 09:06:33.9103203	62170	4	Microsoft-Windows-Shell-Core	Microsoft-Windows-Shell-Core/Operational	2992	672	client-win10-2.ninja-motors.net	S-1-5-21-3671970501-3975728774-4289435121-1110		

EvtXtract		BulkExtractor + EvtxeCmd	
count	Event ID	count	Event ID
21	100	11	100
1	1000		
5	10000		
1	10002		
15	1001		
10	1002		
13	1004	13	1004
		2	1009
66	101	57	101
14	1010		
12	1012		
1	1014		
2	1015	1	1015
2	1016	1	1016
26	102	10	102
1	1020	1	1020
2	1026	2	1026
8	1027		
4	103		

11	104	4	104
		107	
6	1089	3	1089
1	110	3	1104
4	1104	3	1105
6	1105	2	1108
4	1108	3	1109
6	1109	2	111
4	111	1	1116
1	1116	7	116
2	12	1	129
13	129	1	140
		1	145
1	145	2	153
		1	1531
27	1500	29	1501
		1	1531

By the way, these are comparative results of record-carving between EvtXtract and BulkExtractor

+ EvtE Cmd on unallocated data of Client-Win10-2.E01.

As you can see, in most cases, EvtXtract is superior to BulkExtractor + EvtxE Cmd.

However, the latter one could sometimes carve several events that EvtXtract was not able to do.

As a result, we recommend you should use both.

# Keyword Search

# Keyword Search

- We perform it in order to get certain strings such as command names and file names from unallocated spaces, memory images, paging files and so on.
- We usually apply this method with keywords such as names of well-known attacking tools, archiving tools, deletion tools, and extensions of archive files.
- Then, we could get the arguments of the tools, and metadata of those files and folders. Metadata might provide their attributes such as size and timestamps. In other words, we might get the names of tools the attacker used, the names of files they accessed, size of the files, and timestamps of them.

# Practice Exercise 3:

Gathering attributes of the archives that attackers created

# Practice Exercise 3:

## Gathering attributes of the archives that attackers created (1)

- Conditions:
  - It is NOT related to the scenario 1. It's just an independent exercise.
  - We are investigating a certain compromised system.
  - We already know that the attacker used 7-Zip to archive stolen files. In addition, the attacker renamed "7z.exe" to "7.exe".
  - The stolen files are "docs.zip" and "secret.zip".
  - In this case, we will search the following pagefile.sys for traces of 7-Zip execution.
    - E:\Artifacts\scenario2\_extracted\_pagefile\win10\pagefile.sys
- Goal:
  - To confirm any evidence related to the archive files.

# Practice Exercise 3:

## Gathering attributes of the archives that attackers created (2)

- Launch the command prompt from the shortcut folder.
- Start text search with the following command.

```
bulk_extractor -E find -f 7\.exe -o  
E:\Artifacts\scenario2_carved_command\BE_output_win10_pagefile  
E:\Artifacts\scenario2_pagefile\win10\pagefile.sys
```

Enter this in a single line.

- -E option is to set the scanner plugin to use.
- -f option is to set the search string for find plugin.
- -o option is to set the output folder.
- The last argument is the target image file.
- In this case, we prepared the result in advance. Please do not execute it now. It takes a while.

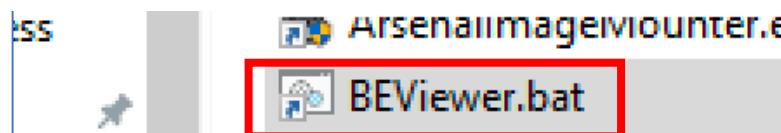
# Practice Exercise 3:

## Gathering attributes of the archives that attackers created (3)

- Then, open the result with BEViewer.

### Shortcuts\09\_RecoveringDataKeywordSearch

1. Launch it from the shortcuts folder.

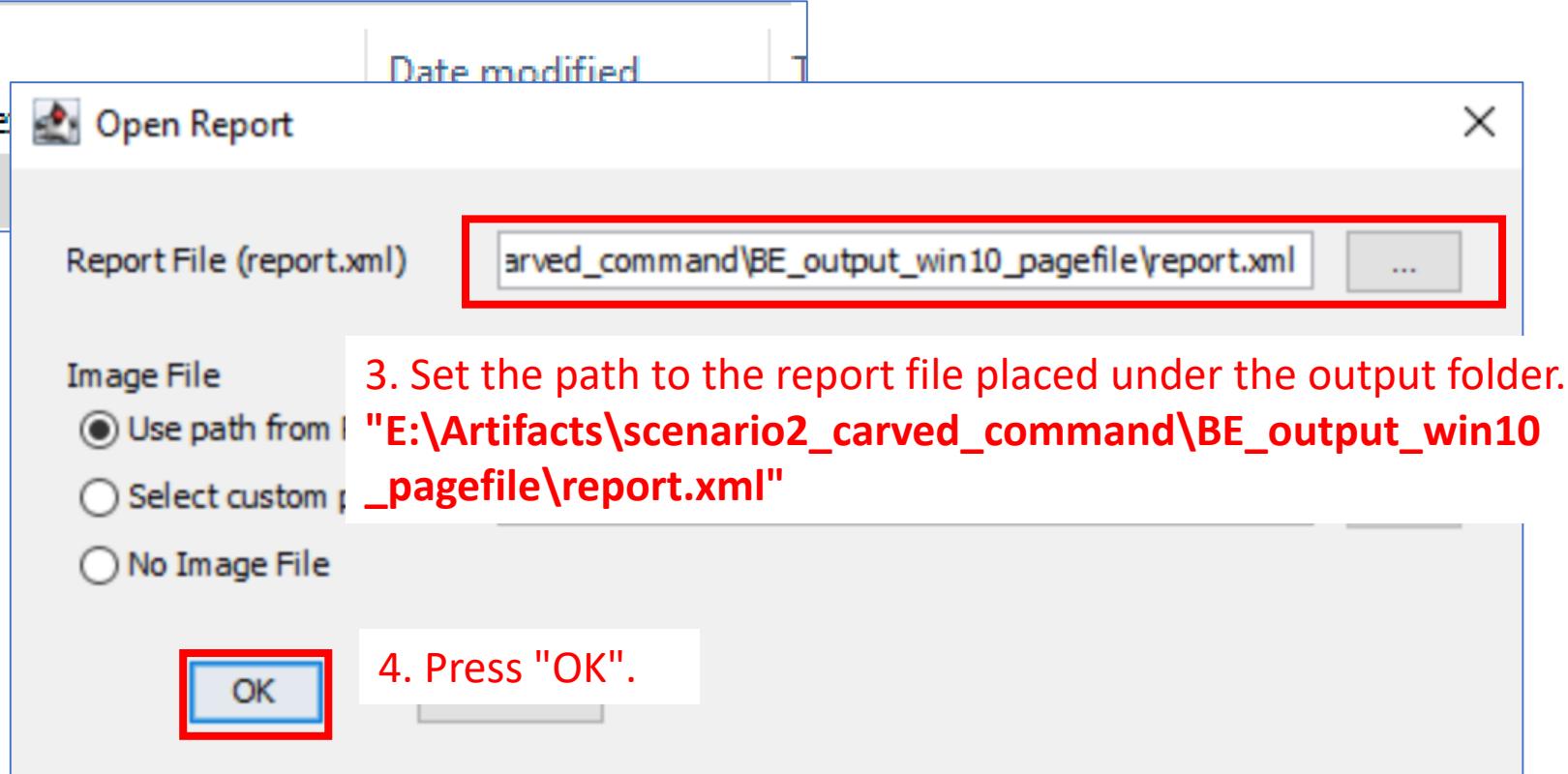


Bulk Extractor Viewer

File Edit View Bookmarks Tools

Open Report... Ctrl+O

2. Choose "Open Report..." from File menu.



# Practice Exercise 3:

## Gathering attributes of the archives that attackers created (4)

- You can get the result like following. It contains several important information.
  1. Select "find.txt" in the Reports window
  2. Click a certain line in the Feature window.
  3. You can see the raw data around detected strings on the target disk image.

The screenshot shows a forensic analysis interface. On the left, under 'Reports', there is a tree view with 'Be output s2-win10' expanded, showing 'find.txt' (highlighted with a red box, labeled 1) and 'find\_histogram.txt'. In the center, the 'Feature Filter' section has 'Match case' unchecked and 'find.txt' entered in the 'Feature File' field. Below it, the 'Feature' table lists several entries: '753689 7.exe' (highlighted with a blue box, labeled 2), '318791535 7.exe', and '459594237 7.exe'. On the right, the 'Image File' section displays file details for 'pagefile.sys': 'Feature File: find.txt', 'Forensic Path: 753689', and 'Feature: 7.exe'. The bottom pane shows the raw disk image data with some lines highlighted in yellow and red boxes, corresponding to the feature hits. A red box labeled 3 points to the raw data area.

- The result looks like a command line to create an archive file. We can guess the original command line as below.

7.exe a -tzip -p1234qwer d z:\public\docs\\*

It might be the password for the archive.

Cop

It might be a part of the archive file name.

It might be the source files of the archive

# Scenario 1 Labs: Lab 4

Keyword search with Mimikatz commands on AD-win2016

# Scenario 1 Labs: Lab 4

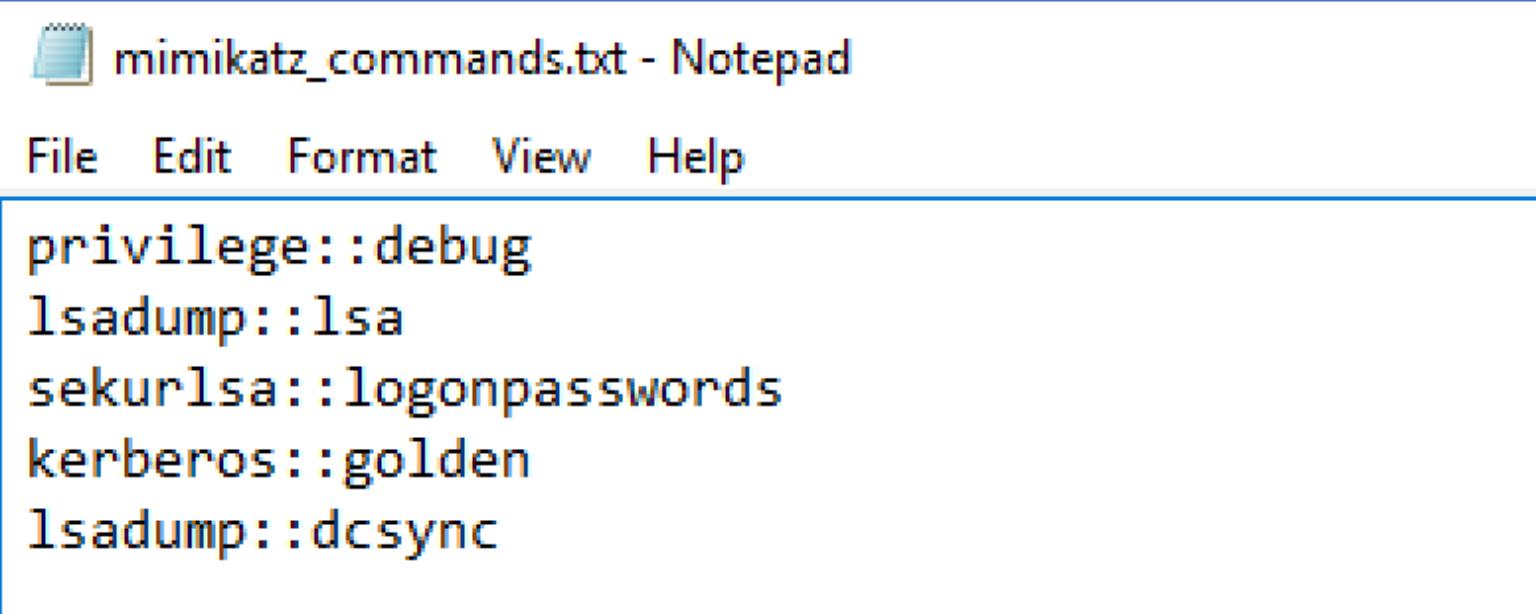
## Keyword search with Mimikatz commands on AD-win2016 (1)

- Conditions:
  - This is an investigation for scenario 1.
  - We also know that the credential information was stolen from the AD server and the attacker could have used Mimikatz for it.
- Goal:
  - To find traces of Mimikatz execution on AD-win2016 with Bulk Extractor's text search function.

# Scenario 1 Labs: Lab 4

## Keyword search with Mimikatz commands on AD-win2016 (2)

- We already prepared Mimikatz command list as the following txt file. Mimikatz usually output these commands with its result. So if the attacker redirected the result to a file, we might be able to carve the files.
  - C:\Tools\mimikatz\_commands.txt



A screenshot of a Windows Notepad window titled "mimikatz\_commands.txt - Notepad". The window contains a menu bar with File, Edit, Format, View, and Help. Below the menu is a list of Mimikatz commands:

```
privilege::debug
lsadump::lsa
sekurlsa::logonpasswords
kerberos::golden
lsadump::dcsync
```

# Scenario 1 Labs: Lab 4

## Keyword search with Mimikatz commands on AD-win2016 (3)

- Launch the Command Prompt from the shortcuts folder.
- Start text search with the following command.

```
bulk_extractor -E find -F C:\Tools\mimikatz_commands.txt -o  
E:\Artifacts\scenario1_carved_command\BE_output_ad E:\Artifacts\scenario1_E01\AD-  
Win2016.E01
```

Enter this in a single line.

- -E option is to set the scanner plugin to use.
- -F option is to set the text file containing search strings for find plugin.
- -o option is to set the output folder.
- The last argument is the target image file.
- In this case, we prepared the result in advance. Please do not execute it now. It takes several hours.

# Scenario 1 Labs: Lab 4

## Keyword search with Mimikatz commands on AD-win2016 (4)

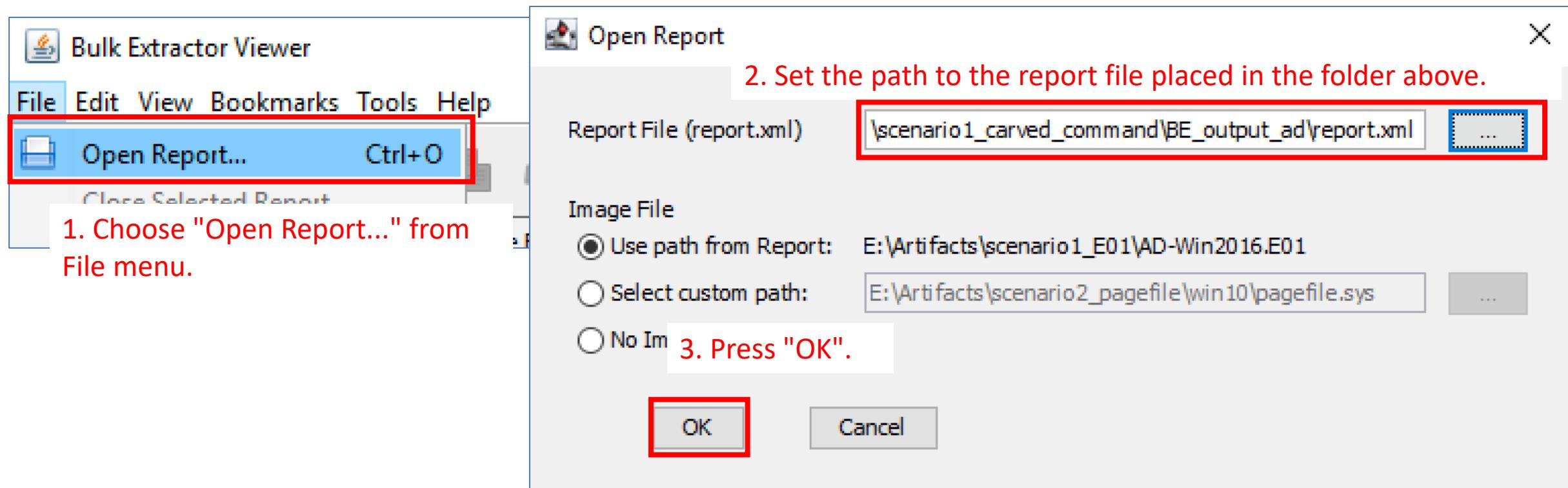
- It will take several hours, so we prepared the result in advance.
- Input Ctrl+c to cancel it and view the report that we prepared.

```
C:\shortcuts\09_RecoveringDataKeywordSearch>bulk_extractor -E find -F C:\Tools\mimikatz\privilege\privilege.py -o E:\Artifacts\scenario1_E01\AD-Win2016.E01
bulk_extractor version: 1.6.0-dev-rec03
Input file: E:\Artifacts\scenario1_E01\AD-Win2016.E01
Output directory: E:\Artifacts\scenario1_carved_command\BE_output_ad
Disk Size: 53687091200
Threads: 2
20:04:17 Offset 67MB (0.12%) Done in 0:47:38 at 20:51:55
20:04:18 Offset 150MB (0.28%) Done in 0:35:10 at 20:39:29
20:04:27 Offset 234MB (0.44%) Done in 0:54:34 at 20:59:02
20:04:31 Offset 318MB (0.59%) Done in 0:51:24 at 20:55:56
20:04:34 Offset 402MB (0.75%) Done in 0:45:35 at 20:50:09
^C
C:\shortcuts\09_RecoveringDataKeywordSearch>
```

# Scenario 1 Labs: Lab 4

## Keyword search with Mimikatz commands on AD-win2016 (5)

- Let's open the reports located in following folder.
  - E:\Artifacts\scenario1\_carved\_command\BE\_output\_ad



# Scenario 1 Labs: Lab 4

## Keyword search with Mimikatz commands on AD-win2016 (6)

- The report shows traces of Mimikatz lsadump command. It could be an evidence to prove that the attacker stole NTLM hash of the user krbtgt on the AD host.
- Typically, attackers generate Golden Tickets with the krbtgt's hash.

The screenshot shows a forensic analysis interface with the following details:

- Reports**: BE\_output\_ad, find.txt, find\_histogram.txt
- Feature Filter**: Match case
- Feature File**: find.txt
- Image File**: AD-Win2016.E01
- Feature File**: find.txt
- Forensic Path**: 27603791959
- Feature**: lsadump::lsa
- Image**: 27603791959
- Path to the output file**: ..... (.....0.....0.....)
- Using**: 'C:\ProgramData\A8Lmsa3o.log' for logfile : OK....
- mimikatz(commandline) # privilege::debug..Privilege 'Z0' OK....**
- lsadump::lsa /inject /name:krbtgt .Domain**
- ID : 000001f6 (502)..User : krbtgt.... \* Pr.....**
- 758065da5688b77436ae381d844ba6.. LM : .. Hash NTLM: 267580**
- 65da5688b77436ae381d844ba6.. ntlm- 0: 26758065da5688b77436ae3**
- Dumped HASH :bec8.... \* W**
- Digest.. 01 c1b5ab6ca664352be669a79c99241529.. 02 a2fa..**
- 8830ala5906b675662ff50c1b6.. 03 89985b70b3b9733ec0c777083f45**

Red boxes highlight the following:

- Path to the output file
- Using 'C:\ProgramData\A8Lmsa3o.log' for logfile : OK....
- Executed command: # lsadump::lsa /inject /name:krbtgt .Domain
- Target username: krbtgt.... \* Pr.....
- Dumped HASH :bec8.... \* W

# Scenario 1 Labs: Lab 5

Keyword search with Mimikatz commands on client-win10-1

# Scenario 1 Labs: Lab 5

Keyword search with Mimikatz commands on client-win10-1 (1)

- Conditions:

- This is an investigation for scenario 1.
- In the scenario, we have determined that Mimikatz was used to steal and forge credentials.

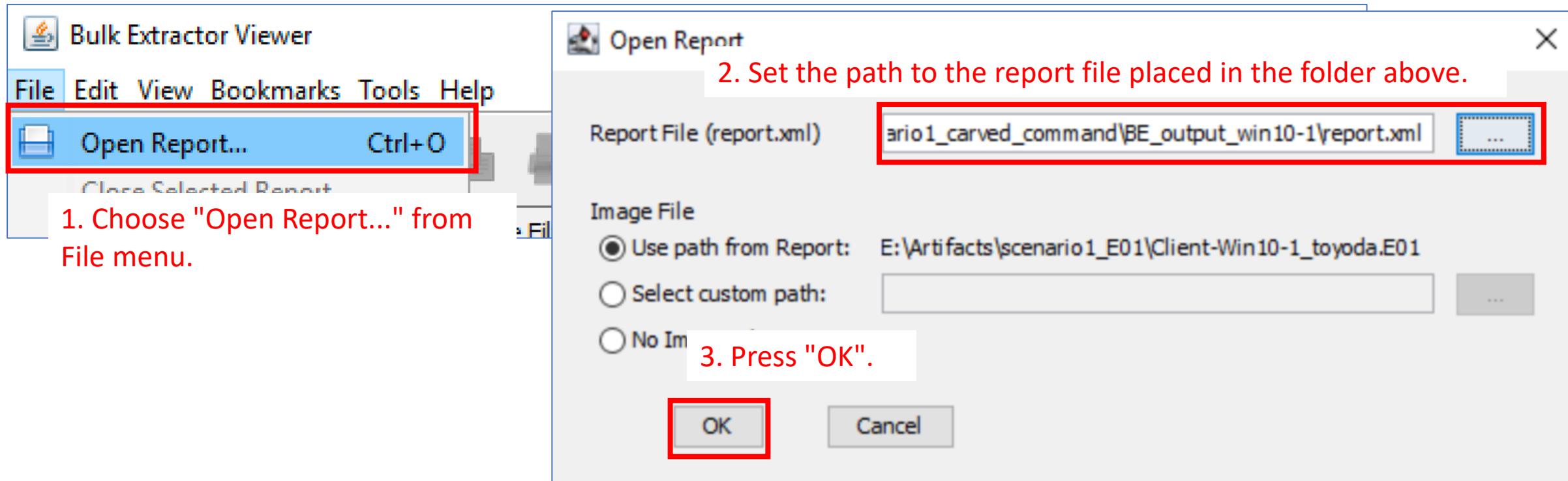
- Goal:

- To find traces of Mimikatz execution on client-win10-1 with Bulk Extractor's text search function.

# Scenario 1 Labs: Lab 5

## Keyword search with Mimikatz commands on client-win10-1 (2)

- After canceling the execution, let's view the prepared report saved in the following folder.
  - E:\Artifacts\scenario1\_carved\_command\BE\_output\_win10-1



# Scenario 1 Labs: Lab 5

## Keyword search with Mimikatz commands on client-win10-1 (3)

The screenshot shows the EnCase Forensic software interface. On the left, the 'Reports' pane displays a folder 'BE\_output\_win10-1' containing files: 'find.txt' (selected and highlighted with a red border) and 'find\_histogram.txt'. The 'Feature Filter' checkbox is checked, and the filter text box contains 'Match case'. In the center, the 'Feature File find.txt' pane shows a list of log entries. A specific entry is selected and highlighted with a red border: '1683927570 privilege::debug'. To the right, the 'Image File Client-Win10-1\_toyoda.E01' pane shows the raw disk image data. A portion of the log entries from the image is displayed, starting with '1683927488 ..Using 'C:\ProgramData\A8Lmsa3o.log' for logfile : OK....mimika'. The line containing the selected feature value ('privilege::debug') is also highlighted with a red border. Red annotations are overlaid on the interface:

- Select "find.txt" in the Reports pane and click a certain line in the Feature pane.
- You can see the raw data around detected strings on the target disk image.
- This part looks like a standard output of the logonpasswords command included in Mimikatz.

Image File	Client-Win10-1_toyoda.E01
Feature File	find.txt
Forensic Path	1683927570
Feature	privilege::debug
Image	
1683927488	..Using 'C:\ProgramData\A8Lmsa3o.log' for logfile : OK....mimika
1683927552	tz(commandline) # privilege::debug..Privilege '20' OK....mimikat
1683927616	z(commandline) # sekurlsa::logonpasswords....Authentication Id :
1683927680	0 ; 294431 (00000000:00047elf)..Session : Interactive
1683927744	from l..User Name : toyoda..Domain : NINJA-M
1683927808	OTORS..Logon Server : AD-WIN2016..Logon Time : 2018/-21-3671970501-39757287
	Primary... * Username
	* NTLM : a68d3722cf
1683928064	9db8ba058ef0273bala2cd... * SHA1 : 33efc45367940654143493460
1683928128	104e7a565dffel... * DPAPI : 0d13f7d38627f6a09830191e407fb2bc
1683928192	...tspkg :....wdigest :.... * Username : toyoda... * Domain :
1683928256	NINJA-MOTORS... * Password : (null)...kerberos :.... * Username
1683928320	: toyoda... * Domain : NINJA-MOTORS.NET... * Password : (null)
1683928384	...ssp :....credman :.... [00000000]... * Username : NINJA-MOTOR
1683928448	S\toyoda... * Domain : MS.Outlook.15:toyoda@EXCHNG-WIN2016.nin

# Scenario 1 Labs: Lab 6

Keyword search with Mimikatz commands on client-win10-2

# Scenario 1 Labs: Lab 6

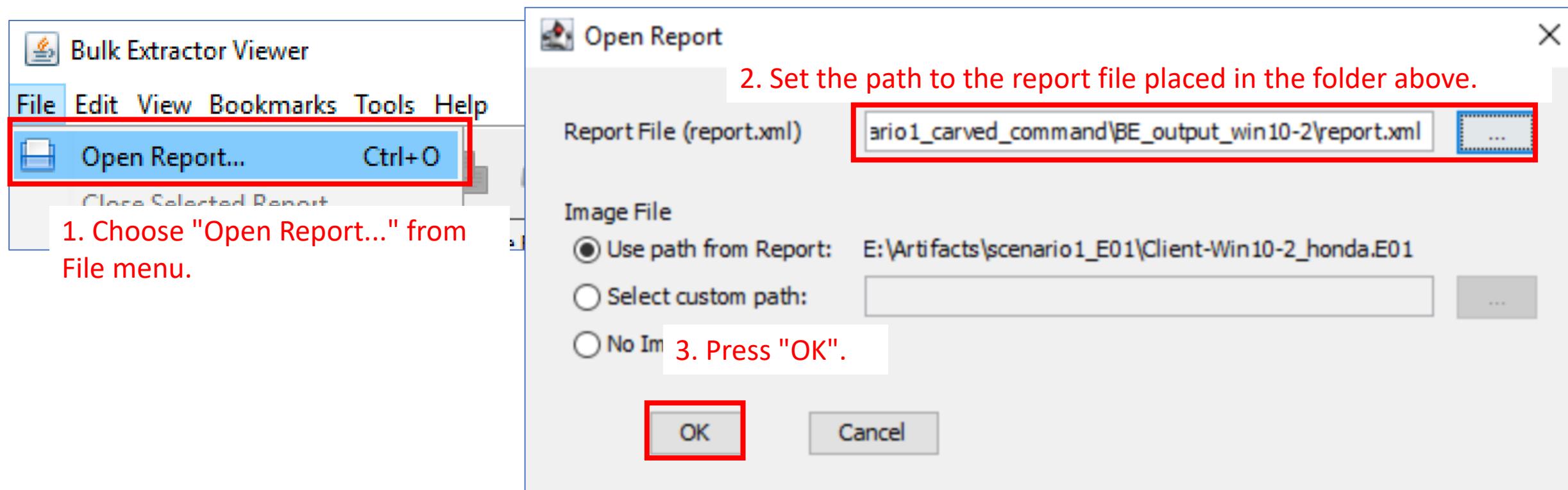
Keyword search with Mimikatz commands on client-win10-2 (1)

- Conditions:
  - This is an investigation for scenario 1.
  - In the scenario, we have already determined that Mimikatz was used to steal and forge credentials.
- Goal:
  - To find traces of Mimikatz execution on client-win10-2 with Bulk Extractor's text search function.

# Scenario 1 Labs: Lab 6

## Keyword search with Mimikatz commands on client-win10-2 (2)

- Let's open the reports located in the following folder.
  - E:\Artifacts\scenario1\_carved\_command\BE\_output\_win10-2



# Scenario 1 Labs: Lab 6

## Keyword search with Mimikatz commands on client-win10-2 (3)

- We can also confirm that Mimikatz's logonpasswords command was executed on this host like the previous one.

The screenshot shows the Volatility Forensics interface. On the left, under 'Reports', there is a folder 'BE output win10-2' containing files: 'find.txt' (highlighted with a red box) and 'find\_histogram.txt'. The main pane displays a table titled 'Feature File find.txt' with several rows of data. The 'Feature' column contains entries like 'privilege::debug' and 'sekurlsa::logonpasswords', with the latter appearing multiple times. To the right, detailed information for the file 'find.txt' is shown, including its 'Image File' (Client-Win10-2\_honda.E01), 'Feature File' (find.txt), 'Forensic Path' (5945725521), and 'Feature' (sekurlsa::logonpasswords). The 'Image' pane below shows raw memory dump data. A red box highlights a specific entry: '5945725888 \* Username : ninja-rdp... \* Domain : NINJA-MOTORS... \* NTLM : 0fab10218d1904124795128ca7cd8202... \* SHA1 : 4a6adcc2e93'. A red arrow points from this entry to a block of text at the bottom of the image pane. This text, also highlighted with a red box, reads: 'This part seems to be NTLM hash of ninja-rdp account. This might be the same file as we found in attack tool analysis.'

Feature	Value
Image File	Client-Win10-2_honda.E01
Feature File	find.txt
Forensic Path	5945725521
Feature	sekurlsa::logonpasswords

Image

```
5945725440 tz(commandline) # privilege::debug..Privilege '20' OK....mimikat
5945725504 z(commandline) # sekurlsa::logonpasswords....Authentication Id :
5945725568 0 : 28151027 (00000000:0lad8cf3)..Session : RemoteInt
5945725632 eractive from 2..User Name : ninja-rdp..Domain
5945725696 : NINJA-MOTORS..Logon Server : AD-WIN2016..Logon Time
5945725760 : 2018/03/09 16:08:08..SID : S-1-5-21-36719705
5945725824 01-3975728774-4289435121-3102...msv :.... [00000003] Primary...
5945725888 * Username : ninja-rdp... * Domain : NINJA-MOTORS... * NTLM : 0fab10218d1904124795128ca7cd8202... * SHA1 : 4a6adcc2e93
5945725952 : d95c6439474b5ff9d9485364f2c21... * DPAPI : 81968b084ddeadf24
5945726016 d55706694d6ff9...tspkg :.... * Digest :.... * Username : ninja-rdp
5945726080 5 This part seems to be NTLM hash of ninja-rdp account. This
5 might be the same file as we found in attack tool analysis.
5
5945726336 : 0 ; 28120254 (00000000:0iad14be)..Session : interact
5945726400 5ive from 2..User Name : DWM-2..Domain : Windo
5945726464 w Manager..Logon Server : (null)..Logon Time : 2018/
```

# Results of the Keyword Search Labs

- So far, we can get the evidences of the following facts:
  - The attacker explicitly used lsadump command of Mimikatz on AD-Win2016 to steal krbtgt's NTLM hash. It also means that the attacker had privilege on the host.
  - The attacker explicitly used logonpasswords command of Mimikatz on client-win10-1 and client-win10-2. They could have stolen the credentials of the users who were logging onto the hosts. It means that the attacker had privilege on each host.
- Although we already know these facts, the new evidences could reinforce the evidence that we found before. Confirming each fact with multiple evidences is very important.

# Wrap Up

# Conclusion (1)

- We often attempt to restore the following data.
  - Tools that the attacker used
  - Archive files that the attacker made to bring the contents out
  - Traces of commands that the attacker executed in attacks
  - Evidence files such as browser cache, prefetch and so on
  - Event log and other logs that are already rotated
- Carving methods take a long time and sometimes generate noisy output. Therefore, we have to be clear about what data to restore. In addition, we also have to be sure about how to restore them.

# Conclusion (2)

- We recommend to use these carving methods and test your custom signatures in advance.
- If you cannot find something by these recovering methods, it does not prove that something did not exist. It just means the method was not efficient in that case. We cannot know whether it was overwritten or it did not actually exist.

# Tools

- Rifiuti2 <https://abelcheung.github.io/rifiuti2/>
- RBCmd <https://ericzimmerman.github.io/#!index.md>
- \$I Parse <https://df-stream.com/recycle-bin-i-parser/>
- ShadowKit <http://www.easymetadata.com/shadowkit/>
- ShadowExplorer <https://www.shadowexplorer.com/>
- Vshadowmount <https://github.com/libyal/libvshadow/wiki/Mounting>
- vssadmin & mklink [https://www.forensicswiki.org/wiki/Mount\\_shadow\\_volumes\\_on\\_disk\\_images](https://www.forensicswiki.org/wiki/Mount_shadow_volumes_on_disk_images)
- VSCMount <https://binaryforay.blogspot.com/2018/09/introducing-vscmount.html>
- vss\_carver [https://github.com/mnrbkby/vss\\_carver](https://github.com/mnrbkby/vss_carver)
- Bulk Extractor with Record Carving [https://www.kazamiya.net/bulk\\_extractor-rec](https://www.kazamiya.net/bulk_extractor-rec)
- Photorec <https://www.cgsecurity.org/wiki/PhotoRec>
- Foremost <http://foremost.sourceforge.net/>
- MftCarver <https://github.com/jschicht/MftCarver>
- IndxCarver <https://github.com/jschicht/IndxCarver>
- EvtXtract <https://github.com/williballenthin/EVTXtract>
- Evtx Explorer/EvtxECmd <https://ericzimmerman.github.io/#!index.md>

# Signature Samples for Photorec and Foremost

# Signature Samples (1)

## Additional Signatures for Photorec

- Following samples are additional signature samples that we often use in actual incidents. You can check and try it if necessary.

```
# Prefetch files
pf 0 0x11, 0x00, 0x00, 0x00, 0x53, 0x43, 0x43, 0x41
pf 0 0x17, 0x00, 0x00, 0x00, 0x53, 0x43, 0x43, 0x41
pf 0 0x1A, 0x00, 0x00, 0x00, 0x53, 0x43, 0x43, 0x41
pf 0 0x1E, 0x00, 0x00, 0x00, 0x53, 0x43, 0x43, 0x41
pfcomp(10) 0 "MAM"
```

These are saved in the file below.  
C:\Tools\photorec\photorec-samples.sig

# Signature Samples (2)

## Additional Signatures for Foremost

```
# Prefetch files.
pf      y      1000000  \x11\x00\x00\x00\x53\x43\x43\x41
pf      y      1000000  \x17\x00\x00\x00\x53\x43\x43\x41
pf      y      1000000  \x1A\x00\x00\x00\x53\x43\x43\x41
pf      y      1000000  \x1E\x00\x00\x00\x53\x43\x43\x41
pfcomp  y      1000000  MAM

# RAR archives
rar     y      10000000  \x52\x45\x7E\x5E
rar     y      10000000  \x52\x61\x72\x21\x1A\x07\x00

# Scheduled tasks
task    y      100000  /\xff\xfe<.\?.x.m.l.+?<.T.a.s.k.+?<.U.R.I.>.\\".+?<.C.o.m.m.a.n.d.>./ /<./.T.a.s.k.>./

# ASN1 structures such as extracted credentials by Mimikatz.
asn1    y      4096   \x76\x82??\x30\x82??\xa0\x03\x02\x01\x05\xa1\x03\x02\x01\x16
asn1_84 y      4096   \x76\x84????\x30\x84????\xa0\x84\x00\x00\x00\x03\x02\x01\x05\xa1\x84\x00\x00\x03\x02\x01\x16

# EDB databases such as Windows Desktop Search's Index, Exchange databases, and AD databases(NTDS.DIT).
edb    y      100000000  ????\'xef\xcd\xab\x89\x20\x06\x00\x00\x00\x00\x00
edb    y      100000000  ????\'xef\xcd\xab\x89\x20\x06\x00\x00\x01\x00\x00\x00
```