black hat USA 2018

AUGUST 4-9, 2018
MANDALAY BAY / LAS VEGAS

Reconstruct the World From Vanished Shadow: Recovering Deleted VSS Snapshots

Minoru Kobayashi / Hiroshi Suzuki
Internet Initiative Japan Inc.

Who are we?

- Minoru Kobayashi (@unkn0wnbit)
  - Forensic Investigator
  - Presenter & Hands-On Trainer
    - Mauritius 2016 FIRST TC, Osaka 2018 FIRST TC, domestic conferences in Japan
  - CISSP
- Hiroshi Suzuki (@herosi_t)
  - Malware Analyst & Forensic Investigator
  - Presenter & Hands-On Trainer
    - BlackHat USA/EU/ASIA, FIRST, domestic conferences in Japan

IIJ Internet Initiative Japan

wizSafe

Agenda

1. Introduction
2. The data structure of the VSS snapshots
3. The mechanism of the VSS snapshots
4. The support status of popular VSS snapshot parsers
5. The approach to accessing deleted VSS snapshots
6. Tools overview and restoration test
7. Demonstration
8. Conclusion

Introduction

Outline

- This presentation is our research on Volume Shadow Copy Service (VSS).
- VSS is a backup-related function that is a standard feature on Windows. It can create VSS snapshots (hereinafter referred to as snapshots) of NTFS volumes.
- We can access past data by referring to snapshots. Therefore, traces of attacks can be found. Thus, it will play an important role in incident response.
- However, if an amount of snapshots are over the upper limit of capacity, old ones are deleted by system. Besides, they can be deleted by attackers or malware. We cannot restore deleted snapshots but the data is still remaining.
- In this presentation, we will explain the mechanism of VSS, and discuss the approach of accessing deleted snapshot. In addition, we will also introduce test results of tools we implemented, and we will give demonstrations.

How to utilize VSS snapshots in an incident response

- We can analyze incidents more deeply by restoring traces of attackers and malware such as:
  - Tools used by attackers.
  - Archived files that are temporarily created by attackers.
  - Deleted Event logs.
  - Files that were encrypted by ransomware.
  - And other related artifacts.

Motivation (1)

- Snapshots are important artifacts, but there is no way to access deleted snapshots from Windows.
- Teru Yamazaki, who belongs to Cyber Defense Institute, Inc., confirmed a certain tool can access a deleted snapshot under certain conditions.
  - http://www.kazamiya.net/en/DeletedSC
- For the reasons above, if we could restore VSS related files, we should be able to access data, which is managed by VSS.

Motivation (2)

- Carving is very useful as a way of accessing files in deleted snapshots. However, this method has a fatal defect.
- Carving restores consecutive areas. However, a data chunk of snapshots is backed up in units of 16 KB data. Therefore, carving can only restore data up to 16 KB in that situation. In addition, meta information such as file creation date and time cannot be restored. Furthermore, it is necessary to correctly combine the current NTFS volume with backup data in snapshots when accessing them.
- For the reasons above, we needed a dedicated tool to access deleted snapshots, but there was no software that could be used freely. This is the second motivation.

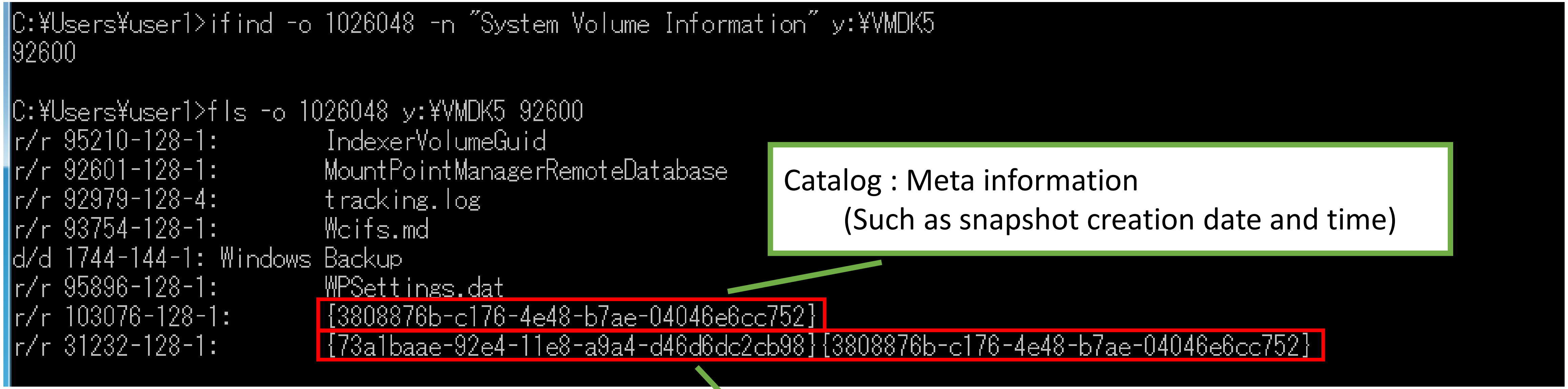
Goal

- Our goal is to create a tool to restore files from deleted snapshots in the following situations:
  - Snapshots that were automatically deleted due to lack of capacity.
  - Snapshots that were deleted by attackers, ransomware, and so on.

The data structure of VSS snapshots

VSS snapshot files

- VSS snapshot management data is saved in "System Volume Information" directly under the volume root.



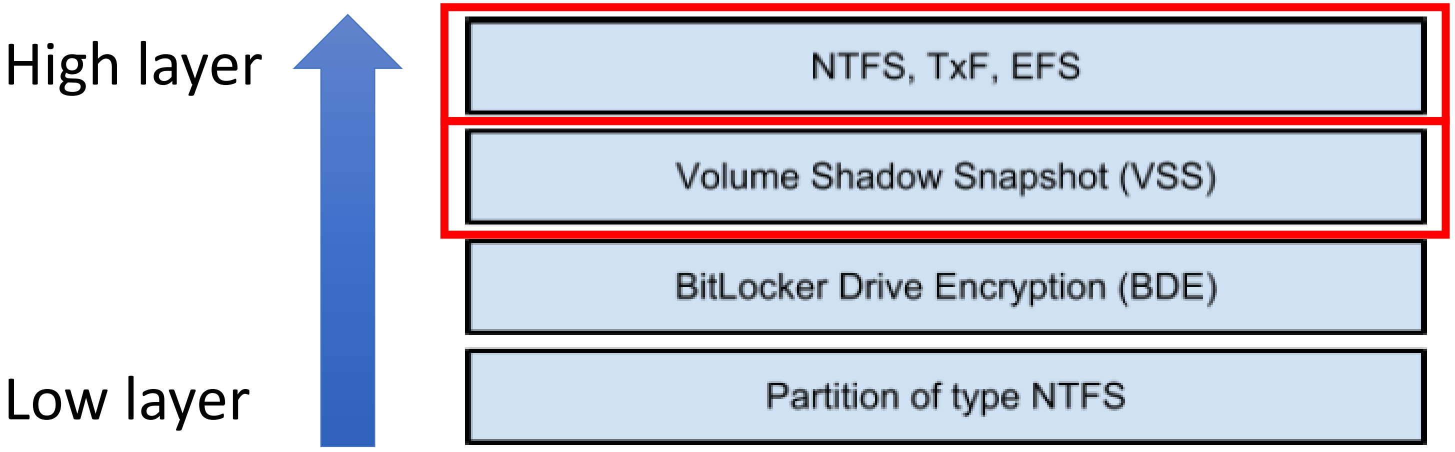
```
C:¥Users¥user1>ifind -o 1026048 -n "System Volume Information" y:¥VMDK5
92600

C:¥Users¥user1>fls -o 1026048 y:¥VMDK5 92600
r/r 95210-128-1:        IndexerVolumeGuid
r/r 92601-128-1:        MountPointManagerRemoteDatabase
r/r 92979-128-4:        tracking.log
r/r 93754-128-1:        Wcifs.md
d/d 1744-144-1: Windows Backup
r/r 95896-128-1:        WPSettings.dat
r/r 103076-128-1:       {3808876b-c176-4e48-b7ae-04046e6cc752}
r/r 31232-128-1:        {73a1baae-92e4-11e8-a9a4-d46d6dc2cb98}{3808876b-c176-4e48-b7ae-04046e6cc752}
```

Catalog : Meta information
(Such as snapshot creation date and time)

Store : Backed up data (Difference data)

VSS system layer

- The management data of VSS snapshots is existent as files, but the VSS operates on the layer lower than the NTFS. Therefore, when VSS refers to snapshots data, it follows offsets of each management data directly instead of parsing the NTFS file system.



High layer

Low layer

NTFS, TxF, EFS

Volume Shadow Snapshot (VSS)

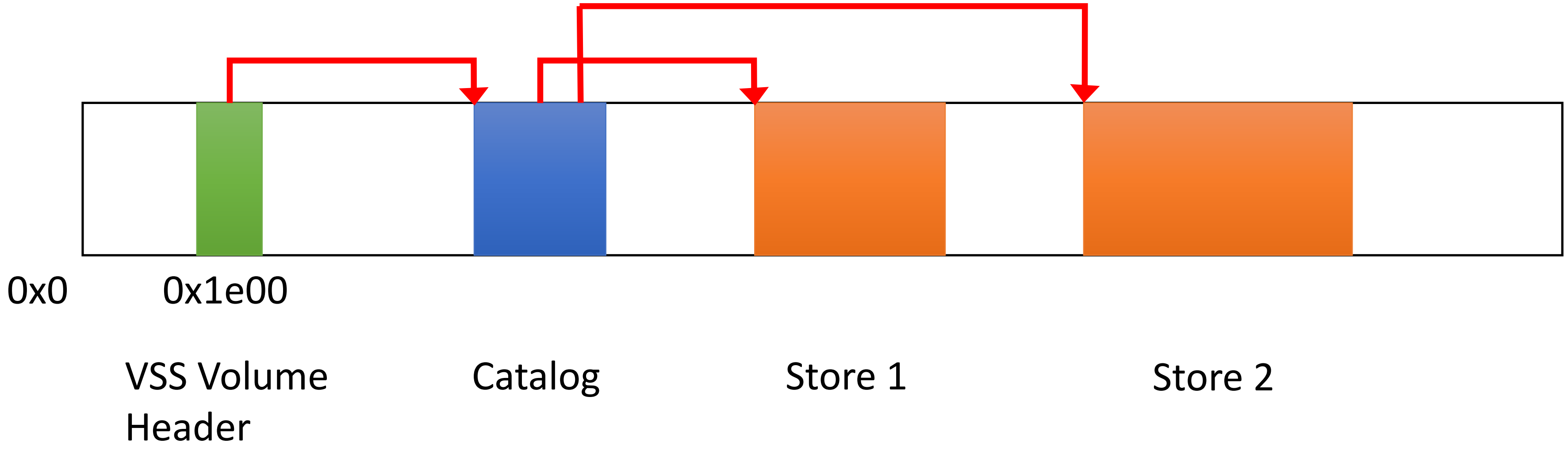
BitLocker Drive Encryption (BDE)

Partition of type NTFS

https://github.com/libyal/documentation/blob/master/Paper%20-%20Windowless%20Shadow%20Snapshots.pdf

Accessing the VSS snapshots

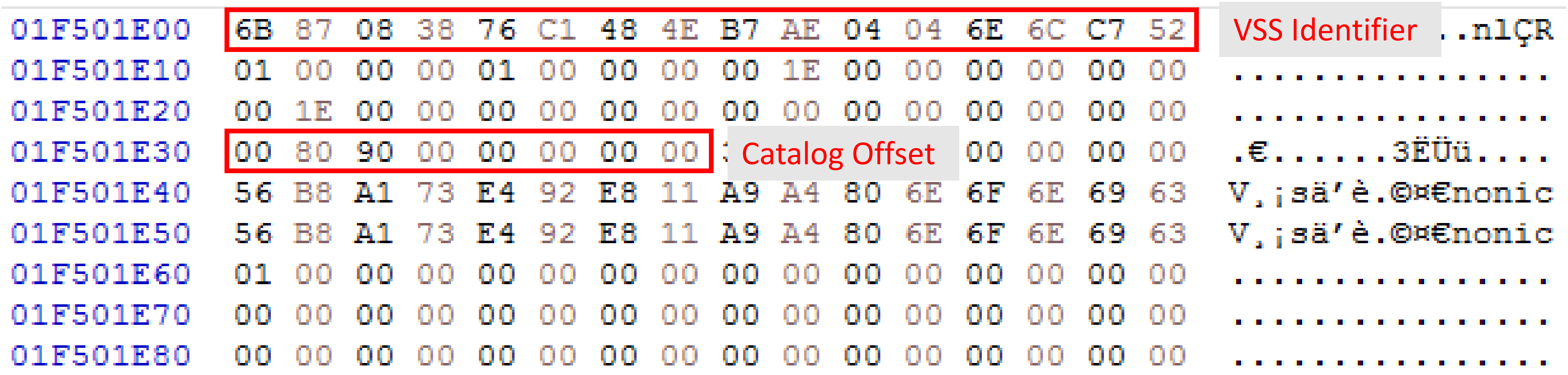
- Windows OS can access VSS snapshots by following the offset list from VSS volume header.



VSS Volume Header (1)

- The data is stored at 0x1e00 from the beginning of NTFS volume. It consists of:
- VSS Identifier
  - Specific 16-byte data is stored.
  - It is set if VSS is enabled on its NTFS volume.
- Catalog Offset
  - This is the Catalog offset from the beginning of NTFS volume.
  - If there is no snapshot, this is set to 0x0.

VSS Volume Header (2)

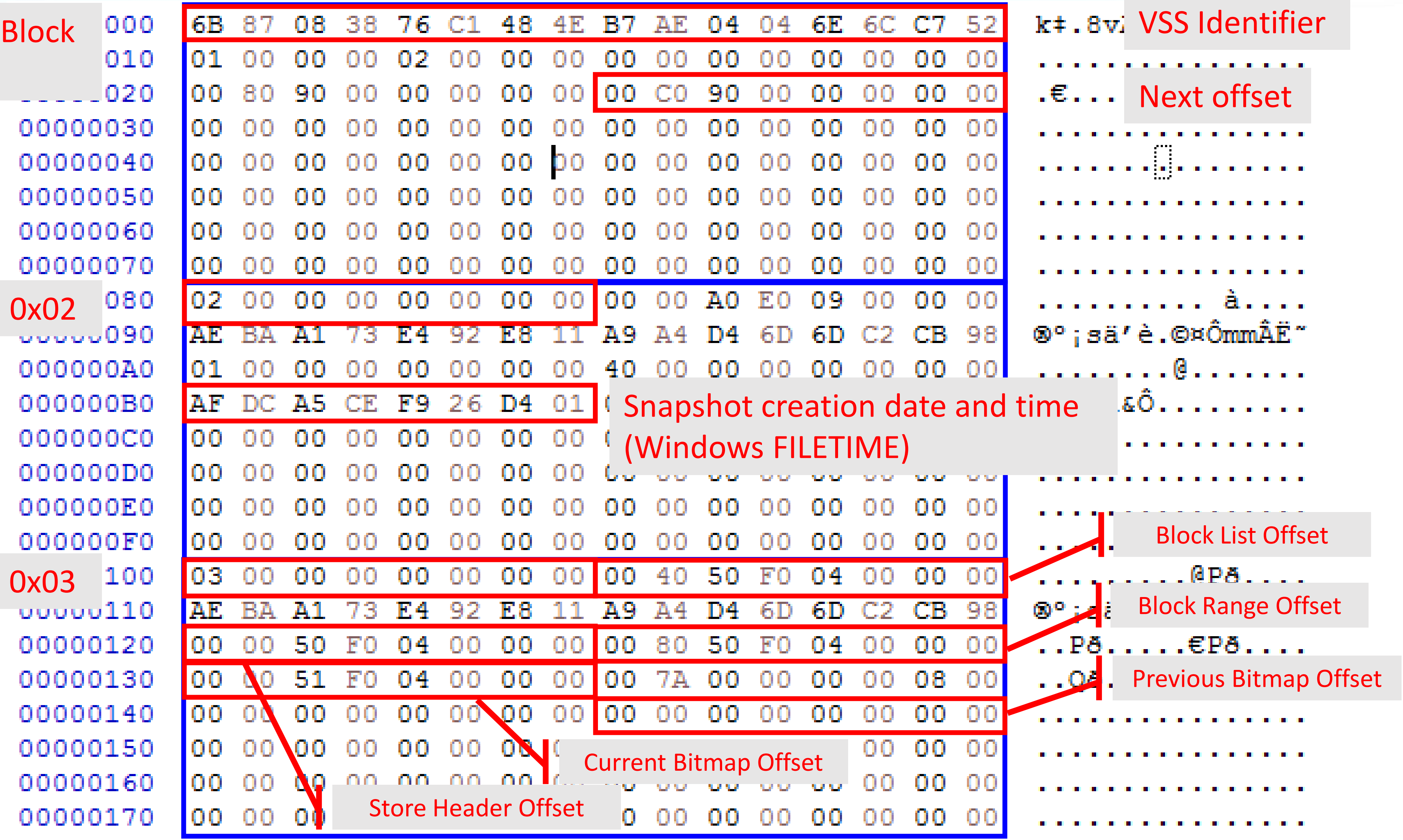


```
01F501E00  6B 87 08 38 76 C1 48 4E B7 AE 04 04 6E 6C C7 52  VSS Identifier ..nlÇR
01F501E10  01 00 00 00 01 00 00 00 00 1E 00 00 00 00 00 00  ................
01F501E20  00 1E 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01F501E30  00 80 90 00 00 00 00 00 3 Catalog Offset 00 00 00 00  .€......3ËÜü....
01F501E40  56 B8 A1 73 E4 92 E8 11 A9 A4 80 6E 6F 6E 69 63  V¸¡sä’è.©¤€nonic
01F501E50  56 B8 A1 73 E4 92 E8 11 A9 A4 80 6E 6F 6E 69 63  V¸¡sä’è.©¤€nonic
01F501E60  01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01F501E70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01F501E80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Catalog (1)

- Catalog Block Header
  - VSS Identifier
  - Next offset
    - It points to the next Catalog block.
- Catalog Entry
  - One snapshot consists of Catalog entry type 0x02 and 0x03.
  - Catalog Entry Type 0x02
    - It has information such as a snapshot creation date and time.
  - Catalog Entry Type 0x03
    - Store Header Offset, Store Block List Offset, Store Block Range Offset, Store Current Bitmap Offset, Store Previous Bitmap Offset, and so on

Catalog (2)



```
     000  6B 87 08 38 76 C1 48 4E B7 AE 04 04 6E 6C C7 52   k‡.8v
     010  01 00 00 00 02 00 00 00 00 00 00 00 00 00 00 00   ................
     020  00 80 90 00 00 00 00 00 00 C0 90 00 00 00 00 00   .€...
00000030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
     080  02 00 00 00 00 00 00 00 00 00 A0 E0 09 00 00 00   ........... à....
     090  AE BA A1 73 E4 92 E8 11 A9 A4 D4 6D 6D C2 CB 98   ®°¡sä’è.©¤ÔmmÂË˜
000000A0  01 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
000000B0  AF DC A5 CE F9 26 D4 01                              &Ô.........
000000C0  00 00 00 00 00 00 00 00                                 ........
000000D0  00 00 00 00 00 00 00 00                           ................
000000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000000F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ...
     100  03 00 00 00 00 00 00 00 00 40 50 F0 04 00 00 00   ..........@Pð....
     110  AE BA A1 73 E4 92 E8 11 A9 A4 D4 6D 6D C2 CB 98   ®°¡sä
00000120  00 00 50 F0 04 00 00 00 00 80 50 F0 04 00 00 00   ..Pð.....€Pð....
00000130  00 00 51 F0 04 00 00 00 00 7A 00 00 00 00 08 00   ..Qð.
00000140  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000150  00 00 00 00 00 00 00                   00 00 00   ................
00000160  00 00                                  00 00 00   ................
00000170  00 00 00                  0 00 00 00 00 00 00 00   ................
```

Catalog Block Header

VSS Identifier

Next offset

Entry Type 0x02

Snapshot creation date and time (Windows FILETIME)

Entry Type 0x03

Block List Offset

Block Range Offset

Previous Bitmap Offset

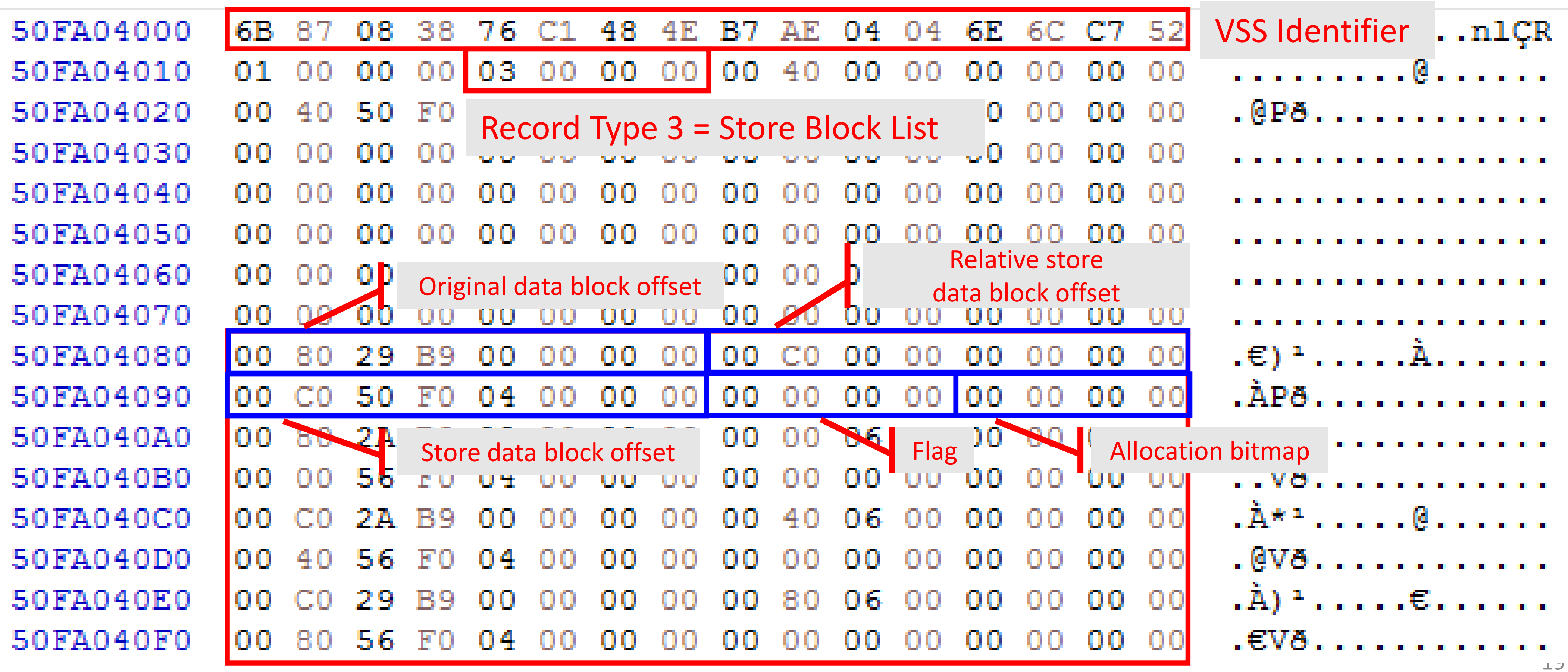
Current Bitmap Offset

Store Header Offset

Store (1)

- Store Block Header
  - One Store consists of 4 kinds of the Store block record types below.
- Store Header (Store Information) : Record Type 4
  - It contains information such as snapshot GUID, attribute flags, and a machine name.
- Store Block List : Record Type 3
  - It is an offsets table of original data blocks and backup data blocks.
- Store Block Range : Record Type 5
  - It is a list of offsets and range of a Store file itself.
- Store Current Bitmap / Store Previous Bitmap : Record Type 6
  - It is a bitmap indicating a usage status of data blocks on NTFS volume.
- Store Data Blocks
  - They are backed up data blocks.

Store (2) - Store Block List



```
50FA04000  6B 87 08 38 76 C1 48 4E B7 AE 04 04 6E 6C C7 52  ..nlÇR
50FA04010  01 00 00 00 03 00 00 00 00 40 00 00 00 00 00 00  .........@......
50FA04020  00 40 50 F0                               0 00 00 00  .@Pð............
50FA04030  00 00 00 00                               0 00 00 00  ................
50FA04040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
50FA04050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
50FA04060  00 00 00                      00 00              ................
50FA04070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
50FA04080  00 80 29 B9 00 00 00 00 00 C0 00 00 00 00 00 00  .€)¹.....À......
50FA04090  00 C0 50 F0 04 00 00 00 00 00 00 00 00 00 00 00  .ÀPð............
50FA040A0  00 80 2A                      00 00 06    00 00   ............
50FA040B0  00 00 56 F0 04 00 00 00 00 00 00 00 00 00 00 00  ..Vð............
50FA040C0  00 C0 2A B9 00 00 00 00 00 40 06 00 00 00 00 00  .À*¹.....@......
50FA040D0  00 40 56 F0 04 00 00 00 00 00 00 00 00 00 00 00  .@Vð............
50FA040E0  00 C0 29 B9 00 00 00 00 00 80 06 00 00 00 00 00  .À)¹.....€......
50FA040F0  00 80 56 F0 04 00 00 00 00 00 00 00 00 00 00 00  .€Vð............
```

VSS Identifier

Record Type 3 = Store Block List

Original data block offset

Relative store data block offset

Store data block offset

Flag

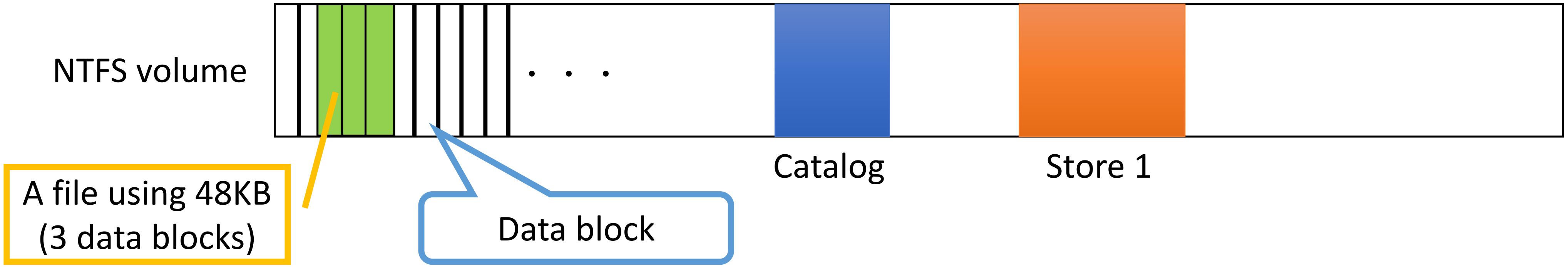
Allocation bitmap

The mechanism of VSS snapshots

The method of storing data of VSS snapshots

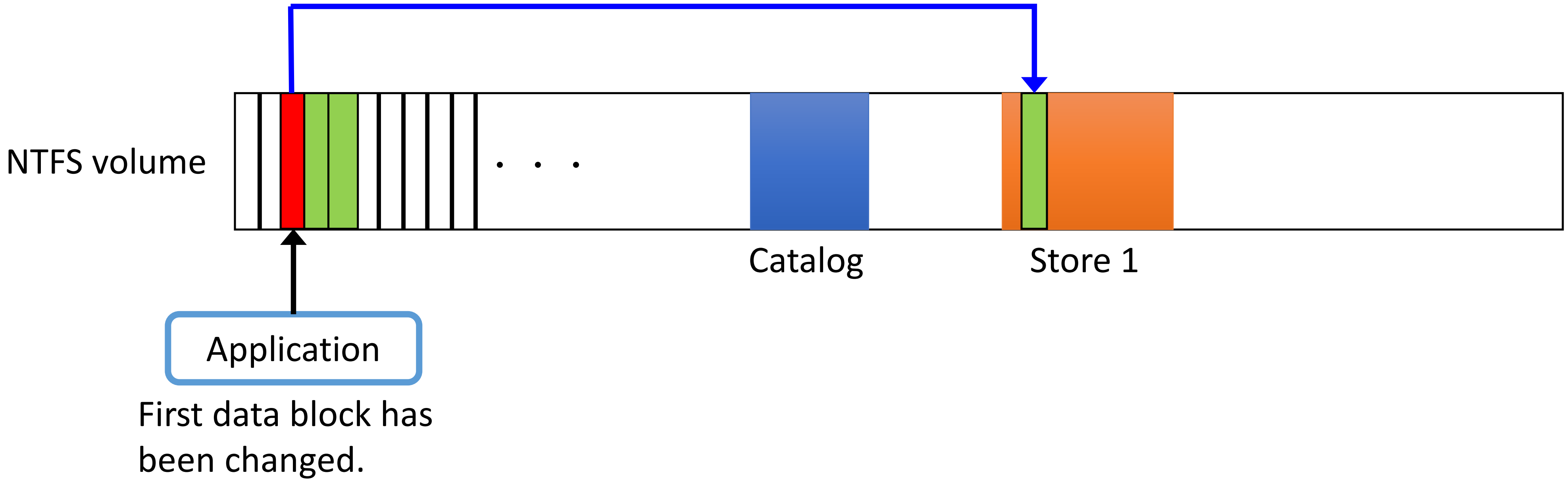
Storing VSS snapshots (1)

- When a snapshot is created, the Catalog and a Store are allocated.
- A data chunk to be backed up is managed in 16KB units called "data block".
- As an example, let's see how a file that uses three data blocks are backed up to snapshots.



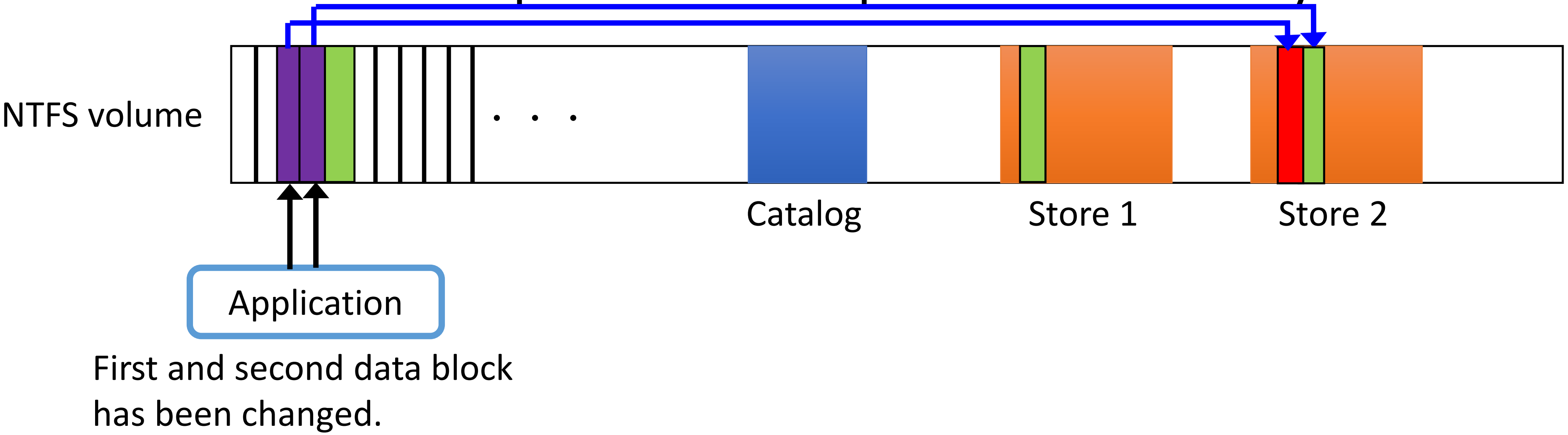
Storing VSS snapshots (2)

- When a data block in the file has been modified, the block is backed up to “Store 1” before it is overwritten.



Storing VSS snapshots (3)

- When a second snapshot is created, the VSS adds the second Catalog entry to the Catalog and allocates a second Store. After that, if the application modified the first and the second block, they are backed up to the second Store.
- The third or later snapshots will be processed similarly.

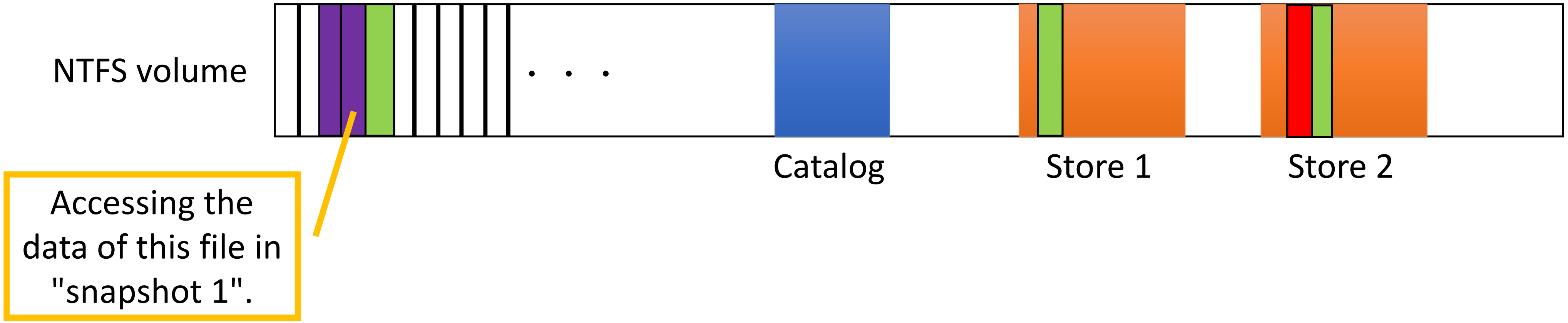


First and second data block has been changed.

The method of accessing data of VSS snapshots

Accessing VSS snapshots (1)

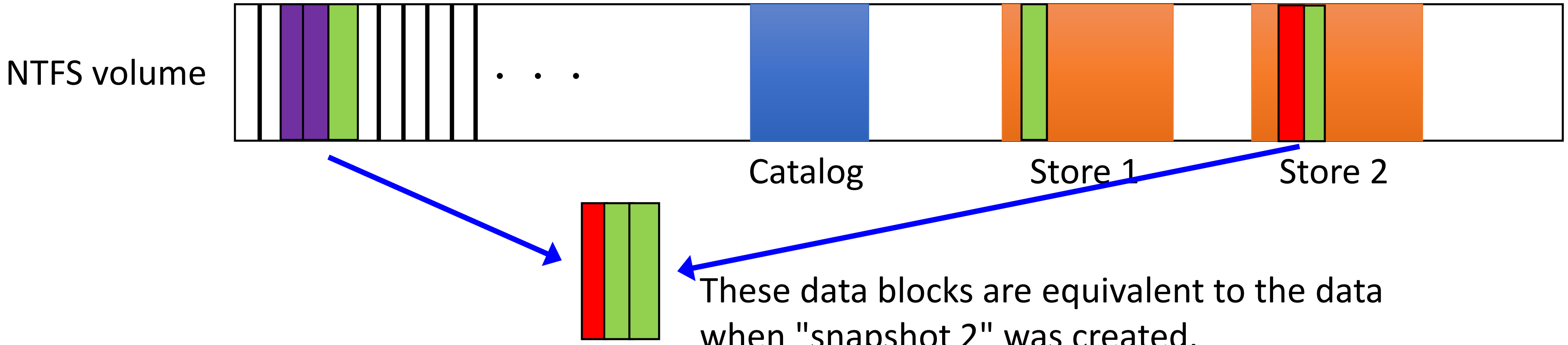
- When accessing backed up data of a snapshot, the VSS combines data blocks on the current NTFS volume with data blocks stored in multiple Store files to reproduce the data at the time of creating the snapshot.
- As an example, let’s consider a case of accessing a file in "snapshot 1".



Accessing the data of this file in "snapshot 1".

Accessing VSS snapshots (2)

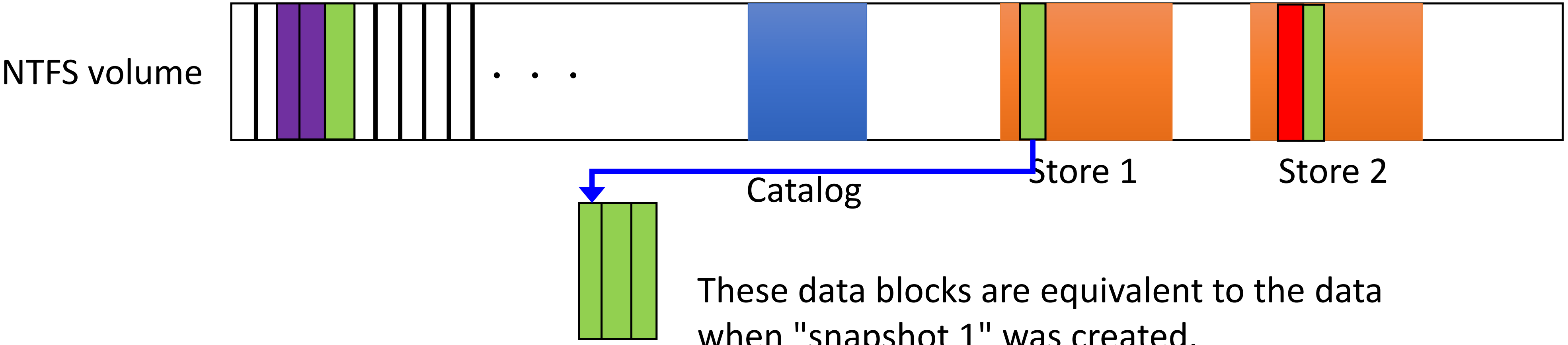
- First, the data blocks of the file on the current volume are combined with the data blocks stored in "Store 2".



These data blocks are equivalent to the data when "snapshot 2" was created.

Accessing VSS snapshots (3)

- Second, the data blocks, which are reconstructed at the previous step, are further combined with the data blocks stored in "Store 1" to recreate the data at the time of creating "Store 1".
- In this way, by combining data blocks on snapshots with data blocks on the current NTFS volume, we can access the data when a snapshot has been created.

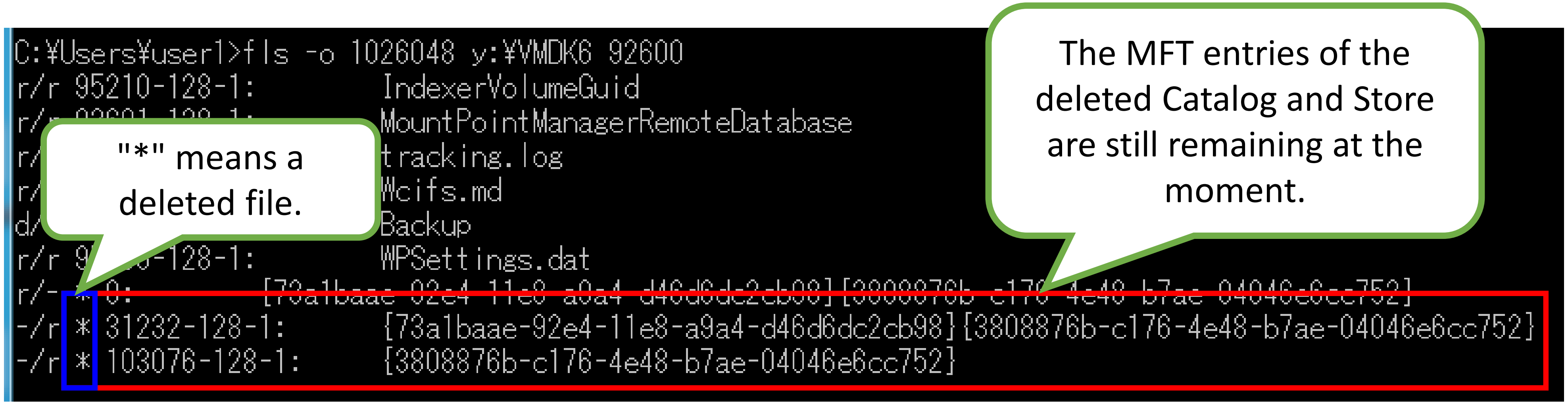


These data blocks are equivalent to the data when "snapshot 1" was created.

Deleting VSS snapshots

Deleting VSS snapshots (1)

- All snapshots are deleted with the following command.
  - vssadmin.exe delete shadows /all
- The state of the Catalog and the Store right after deleting the snapshot



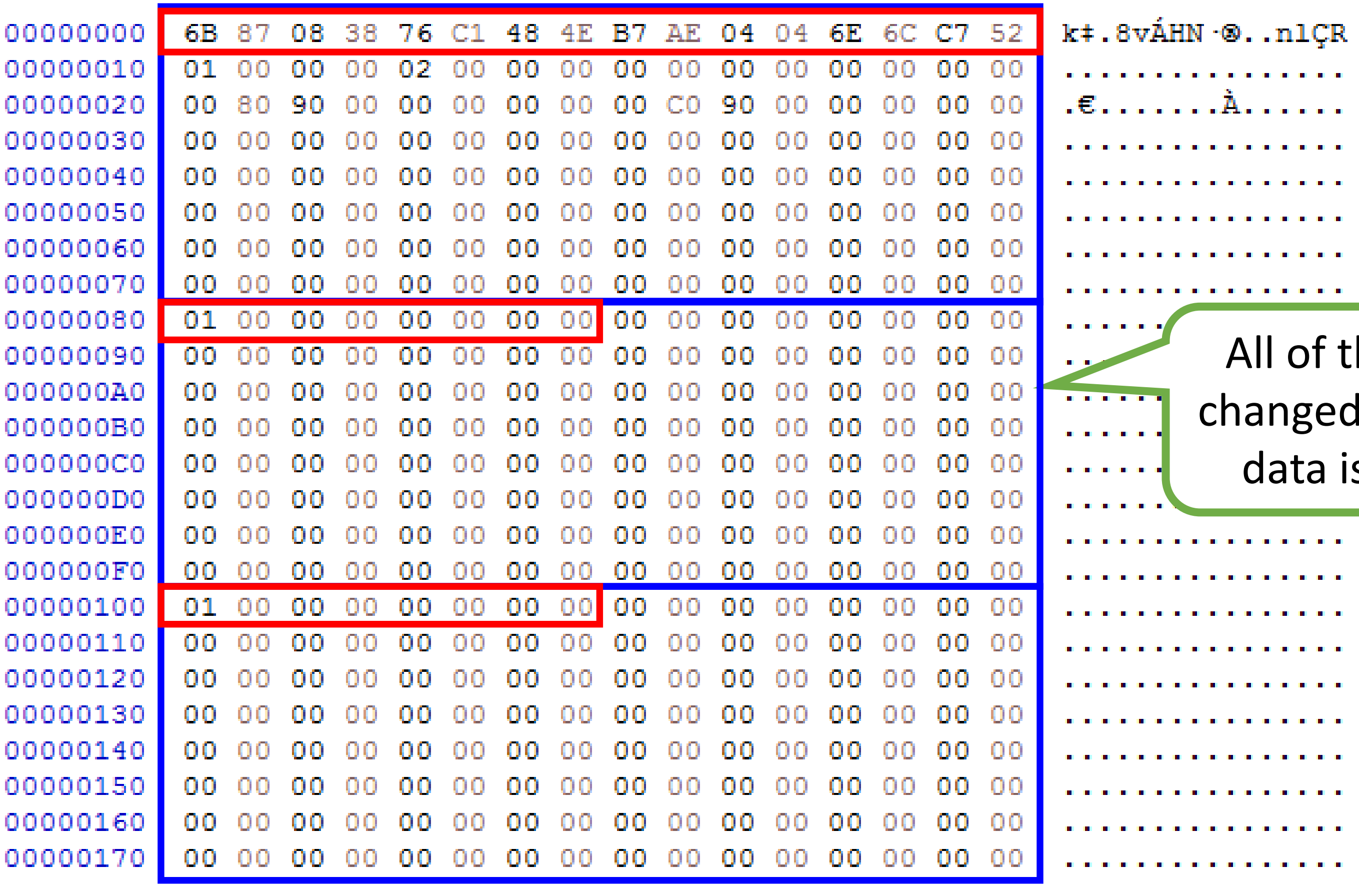
```
C:¥Users¥user1>fls -o 1026048 y:¥VMDK6 92600
r/r 95210-128-1:        IndexerVolumeGuid
r/r 92601-128-1:        MountPointManagerRemoteDatabase
r/                      tracking.log
r/                      Wcifs.md
d/                      Backup
r/r 9    -128-1:        WPSettings.dat
r/- * 0:            [73a1baae-92e4-11e8-a9a4-d46d6dc2cb98][3808876b-c176-4e48-b7ae-04046e6cc752]
-/r * 31232-128-1:      {73a1baae-92e4-11e8-a9a4-d46d6dc2cb98}{3808876b-c176-4e48-b7ae-04046e6cc752}
-/r * 103076-128-1:     {3808876b-c176-4e48-b7ae-04046e6cc752}
```

"*" means a deleted file.

The MFT entries of the deleted Catalog and Store are still remaining at the moment.

Deleting VSS snapshots (2)

- However, the Catalog data is almost completely gone as it was overwritten when the delete command was executed.

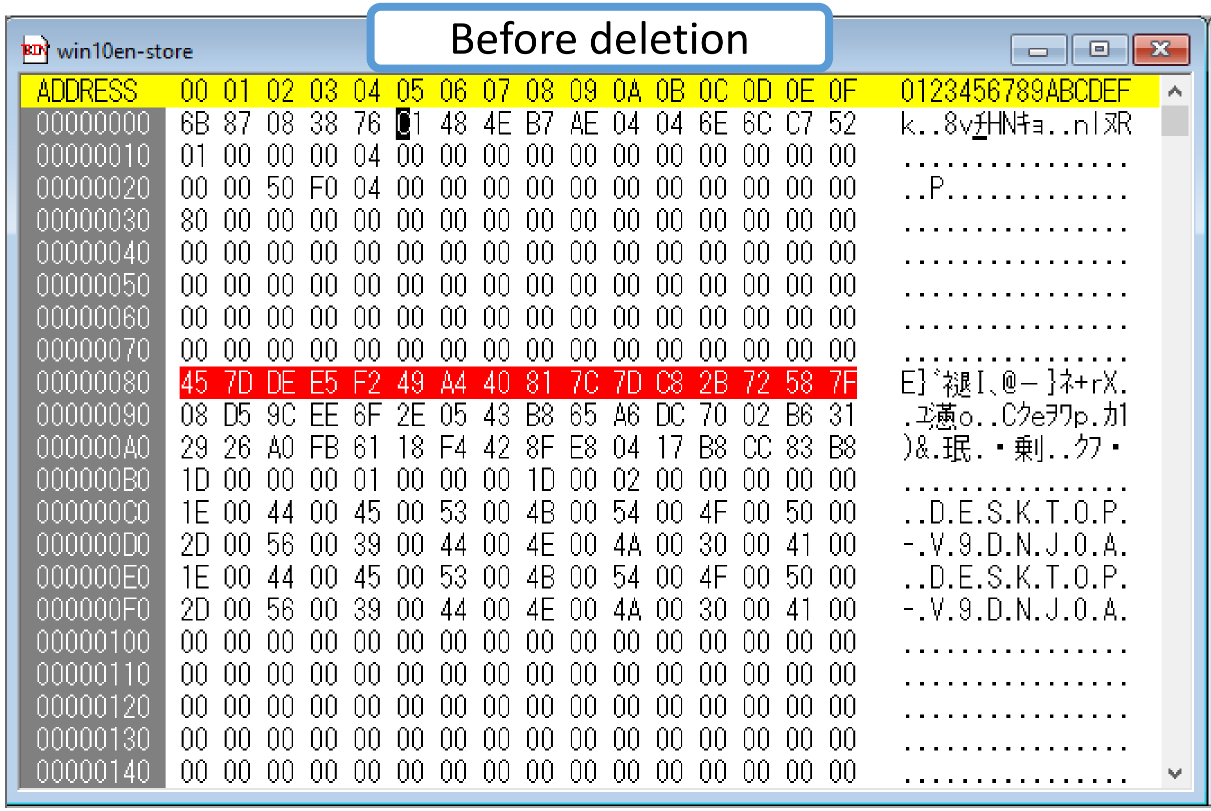


```
00000000  6B 87 08 38 76 C1 48 4E B7 AE 04 04 6E 6C C7 52  k‡.8vÁHN·®..nlÇR
00000010  01 00 00 00 02 00 00 00 00 00 00 00 00 00 00 00  ................
00000020  00 80 90 00 00 00 00 00 00 C0 90 00 00 00 00 00  .€.......À......
00000030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000080  01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000100  01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000140  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000160  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

All of the entry types are changed to 0x01, and other data is filled with 0x00.

Deleting VSS snapshots (3)

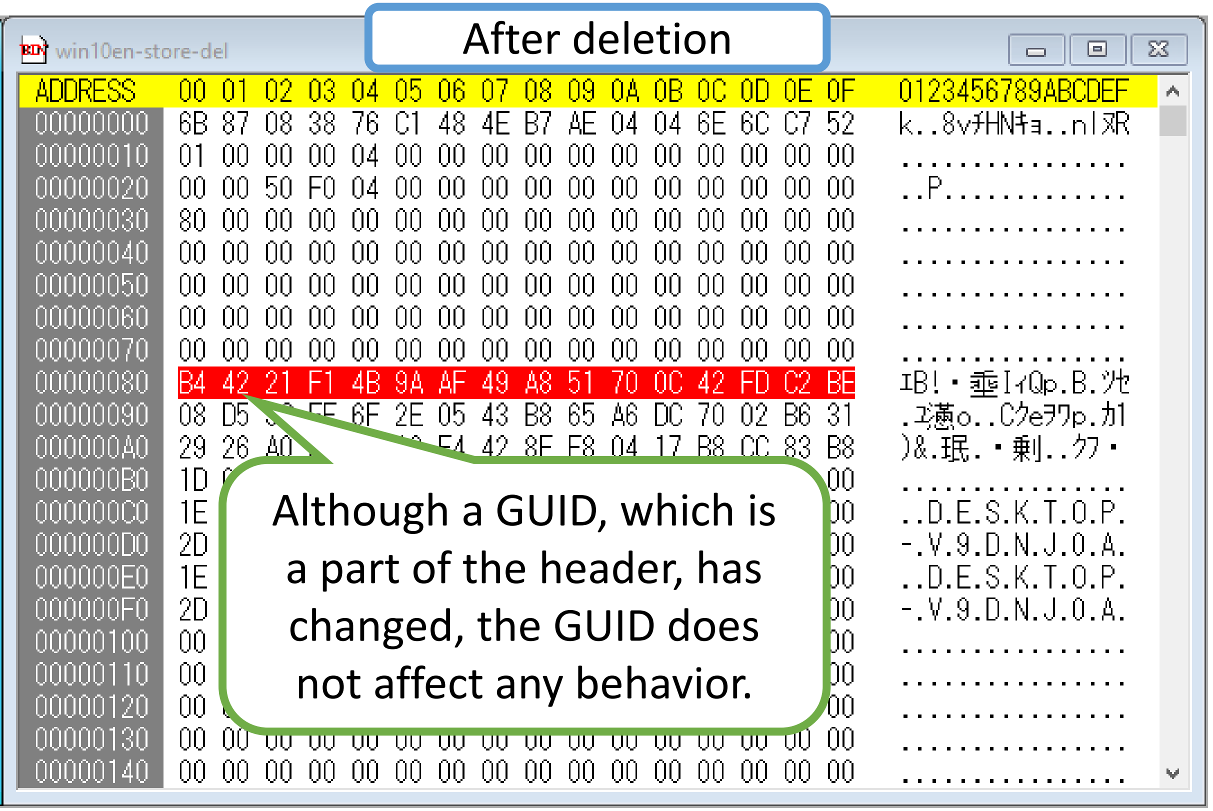
- In contrast, Store data is almost intact.



Before deletion

win10en-store

```
ADDRESS   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   0123456789ABCDEF
00000000  6B 87 08 38 76 C1 48 4E B7 AE 04 04 6E 6C C7 52   k..8vﾁHNｷｮ..nlﾇR
00000010  01 00 00 00 04 00 00 00 00 00 00 00 00 00 00 00   ................
00000020  00 00 50 F0 04 00 00 00 00 00 00 00 00 00 00 00   ..P.............
00000030  80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000080  45 7D DE E5 F2 49 A4 40 81 7C 7D C8 2B 72 58 7F   E}ﾞ褪I､@―}ﾈ+rX.
00000090  08 D5 9C EE 6F 2E 05 43 B8 65 A6 DC 70 02 B6 31   .ﾕ蜑o..Cｸeｦﾜp.ｶ1
000000A0  29 26 A0 FB 61 18 F4 42 8F E8 04 17 B8 CC 83 B8   )&.珉.・剩..ｸﾌ・
000000B0  1D 00 00 00 01 00 00 00 1D 00 02 00 00 00 00 00   ................
000000C0  1E 00 44 00 45 00 53 00 4B 00 54 00 4F 00 50 00   ..D.E.S.K.T.O.P.
000000D0  2D 00 56 00 39 00 44 00 4E 00 4A 00 30 00 41 00   -.V.9.D.N.J.0.A.
000000E0  1E 00 44 00 45 00 53 00 4B 00 54 00 4F 00 50 00   ..D.E.S.K.T.O.P.
000000F0  2D 00 56 00 39 00 44 00 4E 00 4A 00 30 00 41 00   -.V.9.D.N.J.0.A.
00000100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000140  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```



After deletion

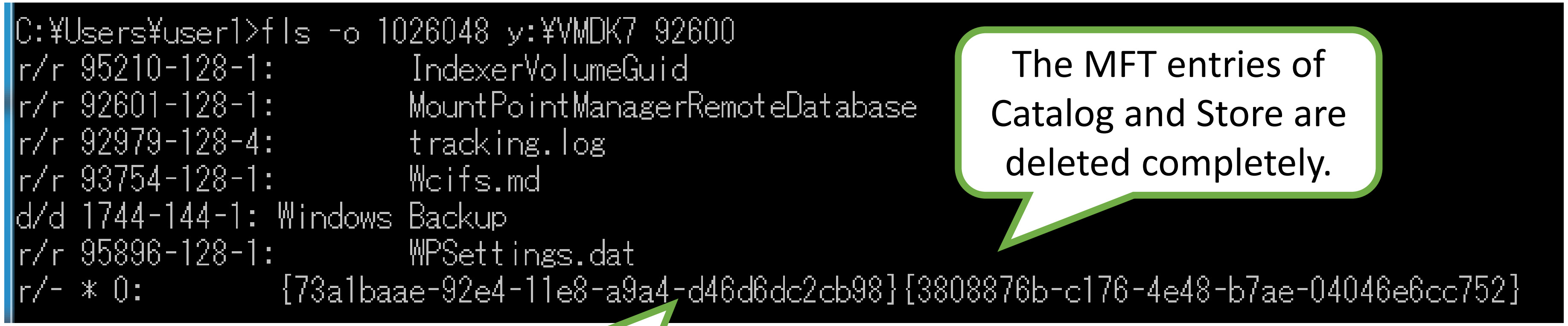
win10en-store-del

```
ADDRESS   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   0123456789ABCDEF
00000000  6B 87 08 38 76 C1 48 4E B7 AE 04 04 6E 6C C7 52   k..8vﾁHNｷｮ..nlﾇR
00000010  01 00 00 00 04 00 00 00 00 00 00 00 00 00 00 00   ................
00000020  00 00 50 F0 04 00 00 00 00 00 00 00 00 00 00 00   ..P.............
00000030  80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000080  B4 42 21 F1 4B 9A AF 49 A8 51 70 0C 42 FD C2 BE   ｴB!・垂IｨQp.B.ﾂｾ
00000090  08 D5       6F 2E 05 43 B8 65 A6 DC 70 02 B6 31   .ﾕ蜑o..Cｸeｦﾜp.ｶ1
000000A0  29 26 A0             42 8F F8 04 17 B8 CC 83 B8   )&.珉.・剩..ｸﾌ・
000000B0  1D                                           00   ................
000000C0  1E                                           00   ..D.E.S.K.T.O.P.
000000D0  2D                                           00   -.V.9.D.N.J.0.A.
000000E0  1E                                           00   ..D.E.S.K.T.O.P.
000000F0  2D                                           00   -.V.9.D.N.J.0.A.
00000100  00                                           00   ................
00000110  00                                           00   ................
00000120  00                                           00   ................
00000130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000140  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

Although a GUID, which is a part of the header, has changed, the GUID does not affect any behavior.

Deleting VSS snapshots (4)

- After several minutes of the snapshots deletion, the MFT entries are also removed.



```
C:¥Users¥user1>fls -o 1026048 y:¥VMDK7 92600
r/r 95210-128-1:        IndexerVolumeGuid
r/r 92601-128-1:        MountPointManagerRemoteDatabase
r/r 92979-128-4:        tracking.log
r/r 93754-128-1:        Wcifs.md
d/d 1744-144-1: Windows Backup
r/r 95896-128-1:        WPSettings.dat
r/- * 0:        {73a1baae-92e4-11e8-a9a4-d46d6dc2cb98}{3808876b-c176-4e48-b7ae-04046e6cc752}
```

The MFT entries of Catalog and Store are deleted completely.

The Store file name still exists, but it is only $I30 INDX entry.

The support status of popular VSS snapshot parsers

VSS snapshot parsers

- Commercial software : Forensic Tool Kit, X-ways Forensics, AXIOM, EnCase
- Free software : ShadowExplorer, ShadowKit
- Open source software : libvshadow
- Most tools cannot access deleted snapshots.
  - X-Ways can access snapshots if MFT entries of deleted Catalog and Store are still remaining.
  - However, these MFT entries are eventually deleted, it is not practical.
- As a result, we decided to adopt the libvshadow as the base of the tool we created.

Why libvshadow?

- In our experience, libvshadow could handle VSS snapshots, even when some commercial software could not handle them correctly.
- "vshadowmount" command, which reproduces snapshots as a raw disk image, is easy to use with other disk image processing tools.
- The VSS snapshot parser is implemented without any Windows file system related APIs unlike other software. In addition, it is open source software. Thus, it is easy to extend the functions.
- Anyone can download and use it for free, if they go to the link below:
  - https://github.com/libyal/libvshadow

The approach to accessing deleted VSS snapshots

Restoring deleted VSS snapshots (1)

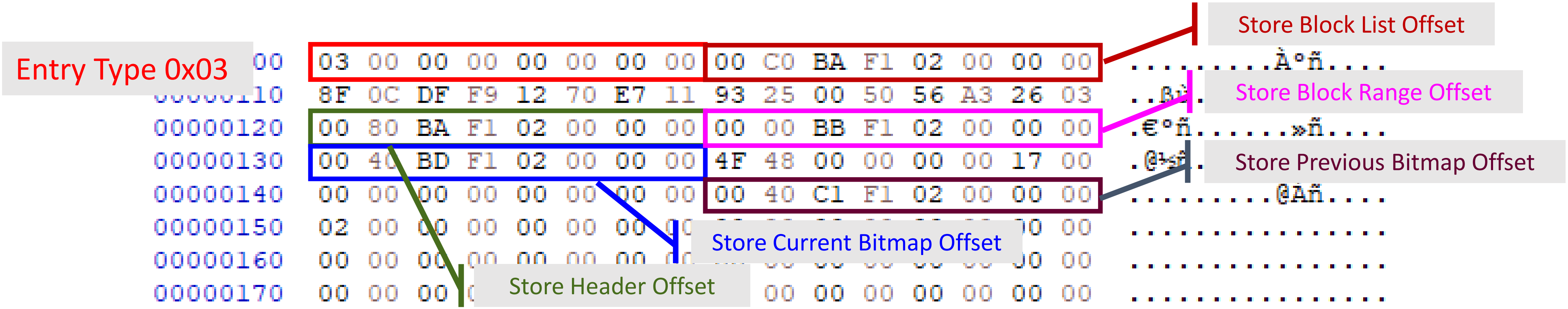
- We need to restore Store and regenerate Catalog to access the deleted snapshot.
- However, there are following problems for restoring Catalog and Store.
  1. Since the Store data is retained in the disk image after deletion, it can be carved from the disk image and the data could be restored. However, since the Store consists of 4 types of Store blocks, the carved Store blocks must be rebuilt into one Store.
  2. The data of the Catalog is completely lost after deletion. Therefore, it is necessary to regenerate it from the carved Store.
  3. When multiple Stores are carved, we cannot identify the order in which they were created.

Restoring deleted VSS snapshots (2)

- Problem 1
  - Since the Store data is retained in the disk image after deletion, it can be carved from the disk image and the data could be restored. However, since the Store consists of 4 types of Store blocks, the carved Store blocks must be rebuilt into one Store.
- Solution 1
  - We decided to check the positions of Store blocks on NTFS volume and consider how the Store can be rebuilt.

Restoring deleted VSS snapshots (3)

- The offsets of each Store block recorded in the Catalog entry type 0x03 reside within a relatively narrow address range (It varies depending on the size of NTFS volume).
  - Store Header Offset: 0x02F1BA8000
  - Store Block List Offset: 0x02F1BAC000
  - Store Block Range Offset: 0x02F1BB0000
  - Store Current Bitmap Offset: 0x02F1BD4000
  - Store Previous Bitmap Offset: 0x02F1C14000



```
        00   03 00 00 00 00 00 00 00  00 C0 BA F1 02 00 00 00   .........Àºñ....
     ...10   8F 0C DF F9 12 70 E7 11  93 25 00 50 56 A3 26 03   ..ßù
00000120   00 80 BA F1 02 00 00 00  00 00 BB F1 02 00 00 00   .€ºñ......»ñ....
00000130   00 4  BD F1 02 00 00 00  4F 48 00 00 00 00 17 00   .@½ñ.
00000140   00 00 00 00 00 00 00 00  00 40 C1 F1 02 00 00 00   .........@Áñ....
00000150   02 00 00 00 00 00 00 0                      0 00   ................
00000160   00 00 00 00 00 00 00 0                      0 00   ................
00000170   00 00 00                    00 00 00 00 00 00 00   ................
```

Entry Type 0x03

Store Block List Offset

Store Block Range Offset

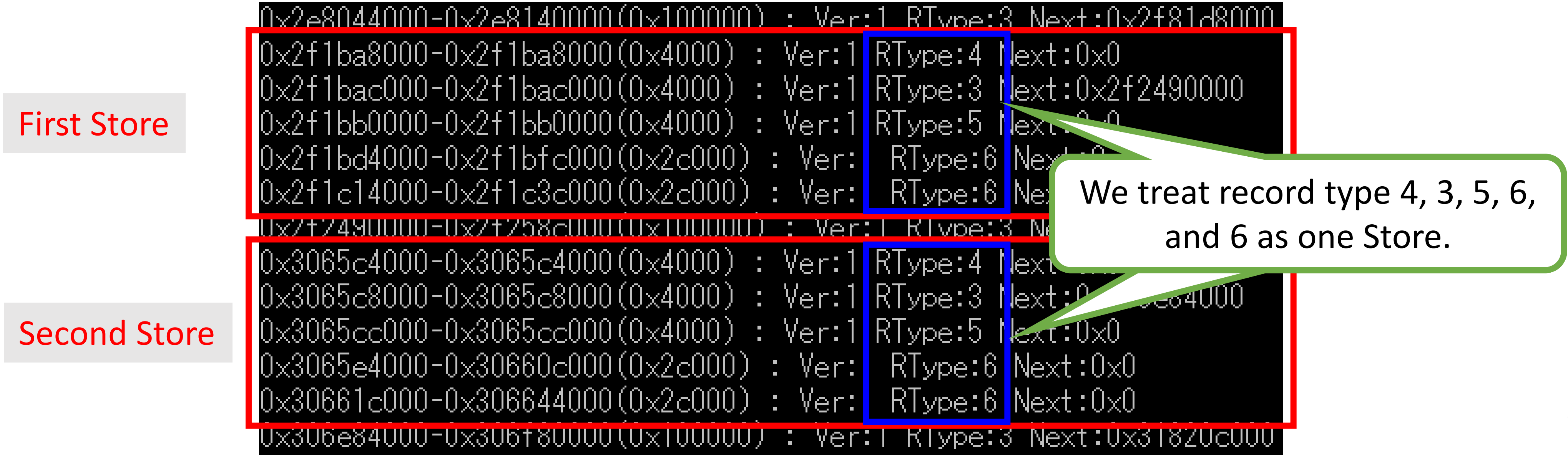
Store Previous Bitmap Offset

Store Current Bitmap Offset

Store Header Offset

Restoring deleted VSS snapshots (4)

- Next, we created a tool to search Store blocks in a disk image.
- The Store blocks that we found always appear in the order of record type 4, 3, 5, 6, 6. Therefore, we can consider that it is possible to carve them as a single Store.



```
0x2e8044000-0x2e8140000(0x100000) : Ver:1 RType:3 Next:0x2f81d8000
0x2f1ba8000-0x2f1ba8000(0x4000) : Ver:1 RType:4 Next:0x0
0x2f1bac000-0x2f1bac000(0x4000) : Ver:1 RType:3 Next:0x2f2490000
0x2f1bb0000-0x2f1bb0000(0x4000) : Ver:1 RType:5 Next:0x0
0x2f1bd4000-0x2f1bfc000(0x2c000) : Ver:  RType:6 Nex
0x2f1c14000-0x2f1c3c000(0x2c000) : Ver:  RType:6 Ne
0x2f2490000-0x2f258c000(0x100000) : Ver:1 RType:3 N
0x3065c4000-0x3065c4000(0x4000) : Ver:1 RType:4 Next
0x3065c8000-0x3065c8000(0x4000) : Ver:1 RType:3 Next:0      e34000
0x3065cc000-0x3065cc000(0x4000) : Ver:1 RType:5 Next:0x0
0x3065e4000-0x30660c000(0x2c000) : Ver:  RType:6 Next:0x0
0x30661c000-0x306644000(0x2c000) : Ver:  RType:6 Next:0x0
0x306e84000-0x306f80000(0x100000) : Ver:1 RType:3 Next:0x31820c000
```

We treat record type 4, 3, 5, 6, and 6 as one Store.

Restoring deleted VSS snapshots (5)

- Problem 2
  - The data of the Catalog is completely lost after deletion. Therefore, it is necessary to regenerate it from the carved Store.
- Solution 2
  - The main information of Catalog is as follows.
    - Snapshot creation date and time
    - Each of the Store offsets such as the Store Header Offset
  - The offsets can be obtained from the carved Store, but the snapshot creation date and time are completely lost.
  - We need to sort the snapshots by the creation date and time to access the data properly. In other words, if the order of the snapshots is correct, the creation date and time can be arbitrary value.
  - Therefore, we decided to set snapshot creation dates based on carved ones (This point is related to problem 3).

Restoring deleted VSS snapshots (6)

- Problem 3
  - When multiple Stores are carved, we cannot identify the order in which they were created.
- Solution 3
  - We assumed that if a new Store is allocated, a larger offset of an NTFS volume than the existing snapshots will be given.
  - When regenerating Catalog data, set the current date as the snapshot creation date for the Store with the largest offset.
  - Then, we set the timestamp of the snapshot to an hour before the creation date of the following snapshot.
  - However, in practice, it is possible that new stores are created with smaller offsets. Since we cannot determine the offsets automatically in the situation, we have created a tool to change the order of snapshots.

Tools overview and file restoration test

The tools created this time

- vss_carver.py
  - It can carve Store data from a disk image.
  - It can regenerate Catalog data from carved Store data.
  - If there is a Catalog in a disk image, that is merged with carved information (Catalog takes precedence).
- vss_catalog_manipulator.py
  - It can manipulate the Catalog entries (change the order of entries, delete entries, and so on.)
- extended-vshadowmount (based on libvshadow-20170902)
  - We added two new options for reading reconstructed Catalog and carved Store.

Example usage of vss_carver.py

- -o / --offset : The offset of NTFS volume from the beginning of disk image
- -i / --image : An input file path to disk image
- -c / --catalog : An output file path to a reconstructed Catalog file
- -s / --store : An output file path to a recovered Store file

```
vss_carver.py -o 123456 -i y:\image -c z:\catalog -s z:\store
```

Example usage of vss_catalog_manipulator.py

- list : print Catalog entries

```
vss_catalog_manipulator.py list z:\catalog
```

- move : move 5th Catalog entry to above 3rd entry

```
vss_catalog_manipulator.py move z:\catalog 5 3
```

- remove : remove 2nd Catalog entry

```
vss_catalog_manipulator.py remove z:\catalog 2
```

- enable : enable 4th Catalog entry

```
vss_catalog_manipulator.py enable z:\catalog 4
```

- disable : disable 7th Catalog entry

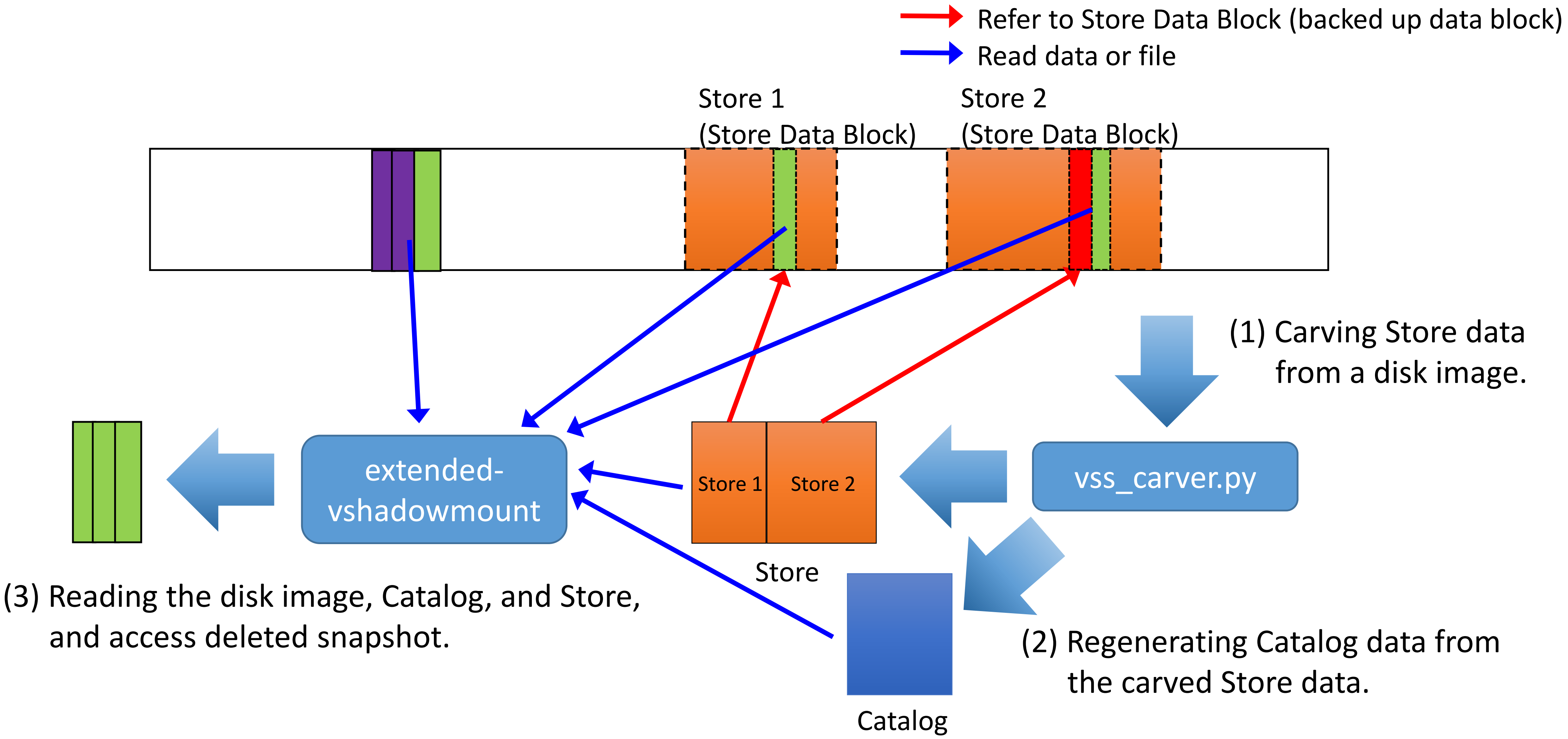
```
vss_catalog_manipulator.py disable z:\catalog 7
```

Example usage of extended-vshadowmount

- Added 2 new options
- -c : specify the Catalog file that is regenerated by vss_carver.py
- -s : specify the Store file that is carved by vss_carver.py

```
vshadowmount.exe -o 123456 -c z:\catalog -s z:\store y:\image x:
```

Tools overview



(1) Carving Store data from a disk image.

(2) Regenerating Catalog data from the carved Store data.

(3) Reading the disk image, Catalog, and Store, and access deleted snapshot.

File restoration test

- Preparation
  - We prepared files that are 3KB, 5MB and 15MB large. For each of them, we put 10 files on the disk. And, we created a snapshot.
  - After that, add 1 byte of data to the beginning of each file and save the file.
- Test 1
  - We deleted all snapshots (but the MFT entries still remain).
- Test 2
  - We deleted all snapshots and files. Then we copied another 10 files, whose size were 5MB, and deleted them. We repeated the operation five times.
- Test 3
  - We executed Teslacrypt to encrypt files.
  - Since we wanted to run it in a closed environment, we used Teslacrypt. It can run without the Internet.

The results of the file restoration test

Software	Test 1	Test 2	Test 3	Remarks
Commercial software A (Ver. X)	✔	✘	✘	It was able to restore when the entries of the deleted Catalog and the Store were in MFT.
Commercial software A (Ver. Z)	✘	✘	✘	Ver. Z is newer than X. It failed to recover data in test 1. It seems like a bug.
Commercial software B	✘	✘	✘	
Freeware C	✘	✘	✘	
vss_carver.py + libvshadow	✔	✔	✔	

✔ : All of files ware restored.

✘ : Any files ware NOT restored.

Demonstrations

Demo 1: Restoring VSS snapshots deleted by the system

- We prepared a Windows 7 disk image which was operated for a month.
- There are three snapshots in the disk image. However, we have been able to find one more snapshot when we used vss_carver.py.
- It means that we could recover data that is older than the data of the existing snapshots.

Demo 2: Restoring VSS snapshots deleted by ransomware

- Victim computer : Windows 10
  1. Creating a VSS snapshot.
  2. Modifying several existing files (MS Word and text file).
  3. Executing Teslacrypt.
  4. Created a snapshot of VM after the encryption.
- Analysis computer : Windows 7
  1. Mounting the disk image of the VM.
  2. Carving VSS snapshots with vss_carver.py.
  3. Mounting the image with extended-vshadowmount with the carved Catalog and the Store.
  4. Restoring data from the image after Teslacrypt execution.

ScopeSnapshots

- Since Windows 8, "ScopeSnapshots" is enabled by default.
- If the feature is enabled, only system files are backed up to VSS snapshot. Other data cannot be backed up.
- To disable this setting, change the following registry value and reboot the computer.
    - Key: HKLM\Software\Microsoft\Windows NT\CurrentVersion\SystemRestore
    - Value Name: ScopeSnapshots
    - Value Type: DWORD
    - Value Data: 0
- For details, check our report on ScopeSnapshots.
    - https://www.iij.ad.jp/en/dev/iir/pdf/iir_vol37_focused1_EN.pdf

Future Work

- Expanding the support of the extended-vshadowmount command into Linux (and macOS).
- Following the latest source code of libvshadow.
- Implementing identification of snapshot creation dates of the recovered Store.
- Implementing automatic sort by Store creation date and time (if we can realize the above).

Conclusion

- vss_carver.py can restore Catalog and Store data from a disk image. In addition, extended-vshadowmount offers the feature to access deleted VSS snapshots with the restored Catalog and Store data.

- We also confirmed that vss_carver.py is effective for snapshots that ware deleted by a system or ransomware.

- These tools are released already.
  - https://github.com/mnrkbys/vss_carver

References

- Deleted Shadow Copies
  - http://www.kazamiya.net/en/DeletedSC

- Volume Shadow Snapshot (VSS)
  - https://github.com/libyal/libvshadow/blob/master/documentation/Volume%20Shadow%20Snapshot%20(VSS)%20format.asciidoc

Questions?

- Thank you for your attention.