フォレンジックツールの開発と実装

株式会社インターネットイニシアティブ
セキュリティ部　セキュリティ情報統括室
小林　稔

Ongoing Innovation

IIJ
Internet Initiative Japan

自己紹介

名前：小林 稔

所属：セキュリティ本部 セキュリティ情報統括室

- 2014年5月IIJ入社
- フォレンジック、インシデントレスポンス、IIJ-SECTメンバー
- 外部
  - Mauritius 2016 FIRST Technical Colloquium スピーカー／トレーナー
  - 2017年セキュリティキャンプ全国大会講師
  - Osaka 2018 FIRST Technical Colloquium トレーナー
  - Japan Security Analyst Conference 2018 スピーカー
  - Black Hat USA 2018 Briefings スピーカー
  - 2018年セキュリティキャンプ全国大会講師

アジェンダ

1. VSSの概要
2. ツール作成の方針
3. vss_carverの処理
4. libvshadowの改修
5. vss_catalog_manipulator
6. デモンストレーション
7. 今後の開発
8. まとめ

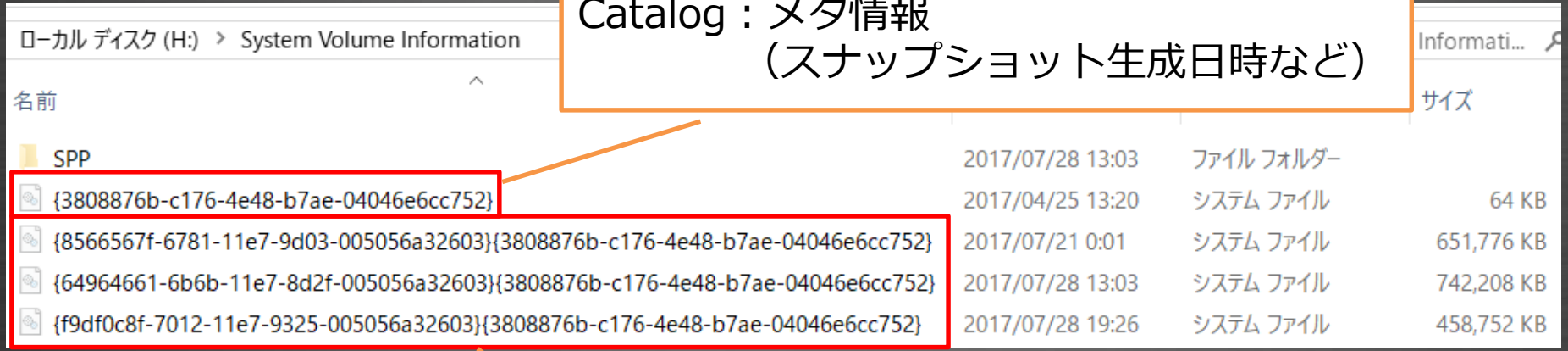
1. VSSの概要

Volume Shadow Copy Service (VSS)とは

- Windowsに標準搭載されているバックアップ関連機能で、NTFSボリュームのVSSスナップショット（以下、スナップショット）を作成することができる。
- スナップショットを参照すると、スナップショット作成時のストレージのデータにアクセスできる。削除された攻撃の痕跡を発見できる場合があるため、調査を行うときに非常に有用なデータとなる。
- スナップショットは差分バックアップであるため、スナップショット内のデータにアクセスするには、現在のボリュームのデータとマージする必要がある（スナップショットだけではデータにアクセスできない）。
- 容量の上限によって古いスナップショットが削除されたり、攻撃者やマルウェアによって削除されてしまう場合がある。また、削除されたスナップショットを復元する方法は存在しない。

⇒スナップショットが復元できれば非常に有用

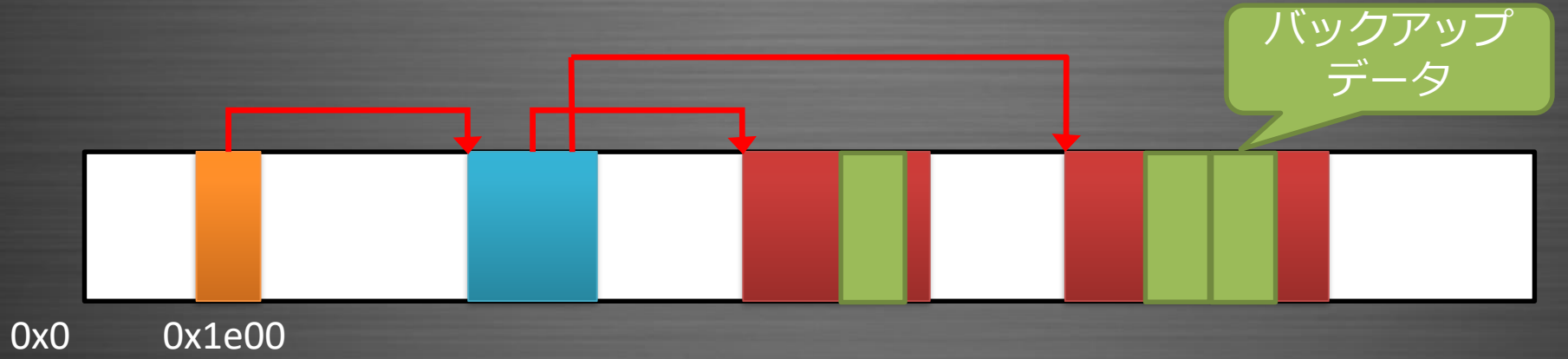
VSSデータファイル構成



Catalog：メタ情報
（スナップショット生成日時など）

Store：バックアップデータ

VSSデータ構造



ボリュームを16KBごとに分割した
データブロックと呼ばれる単位で管理する

2. ツール作成の方針

事前調査 1

スナップショットを削除した際の挙動の検証が行われている。

Deleted Shadow Copies

https://www.kazamiya.net/DeletedSC

以下の点に言及している。

1. カタログとストアのファイルは削除される。
2. カタログのデータがほとんど0x00で上書きされる。
3. ストア内のGUIDが書き換えられる。
4. ある商用ツールで削除されたスナップショットにアクセスできた。

事前調査 2

データを復元できたとして、どのようにスナップショットにアクセスするか。

⇒復元したカタログとストアを外部ファイルとして読み込ませる。

スナップショットのアクセスによく使われるオープンソースのツールを改修すればできそう。

- vshadowmount (libvshadow)
  - https://github.com/libyal/libvshadow
- 削除時にストア内の変化したGUIDについてもスナップショットのアクセスには使用していない。

この時点で検討したスナップショット復元手法

カタログの復元

- カタログの情報が失われているため、何らかの手段で再生成する必要がある。
- NTFSを含む多くのファイルシステムでは、ファイルを削除してもファイルエントリの削除フラグを変更するだけであるため、ストアファイルのファイルエントリからメタ情報（ファイル作成日時、オフセット等）を参照すればカタログを再生成できそう。

ストアの復元

- ストアファイルを復元すればよさそう。

VSS削除検証 1

VSSバックアップ領域設定

vssadmin.exe Resize ShadowStorage /For=C: /On=C: /MaxSize=10%

CドライブのVSSスナップショットの作成

wmic.exe shadowcopy call create Volume='C:¥'

VSSスナップショットの全削除

vssadmin.exe delete shadows /all

VSS削除検証 2

VSSスナップショット削除直後のMFTエントリの状態

VSS削除検証 3

VSSスナップショット削除後数分経過したMFTエントリの状態

```
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-b
                                                    -04046e6cc752}
```

カタログとストアのMFTエントリが削除されてしまう

ストアファイル名が見えるが、これは$I30に残っている情報でファイルではない

データ復元手法の再検討

ストアのMFTエントリが削除されてしまうため、ストアファイルのメタ情報を使う手法は使えない。つまり、ストアファイルの復元も簡単にはできない。

しかし、多くのファイルシステムでは、ファイルを削除してもデータのクリアは行わない。そのため、ディスクの未使用領域に削除前のデータが残っている可能性がある。

ディスクの未使用領域にあるデータを復元する手法として、カービング(Carving)が存在する。

カービングとは

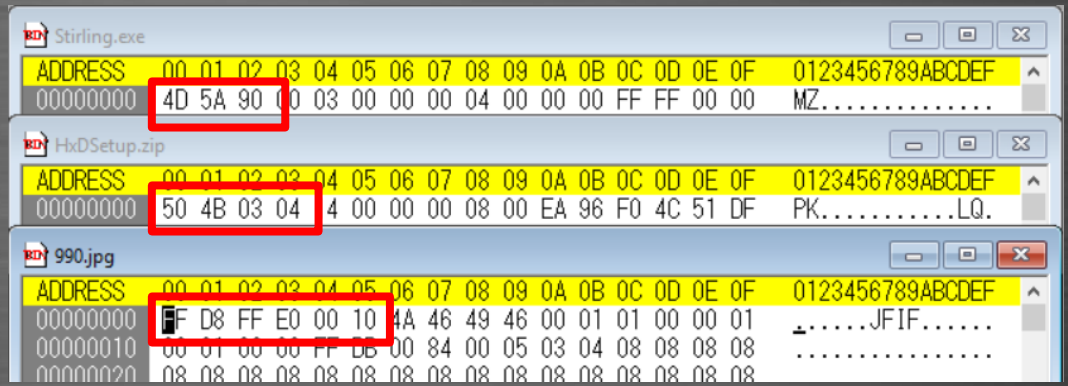
フォレンジックやインシデントレスポンスにおけるカービングとは、データフォーマットの特徴的なデータ列（ファイルヘッダ等）をキーにして、ディスクイメージファイルを検索する手法である。キーが現れたら、そこから連続するバイト列をファイルやデータレコードとして保存する。

例）

EXE：MZ¥x90

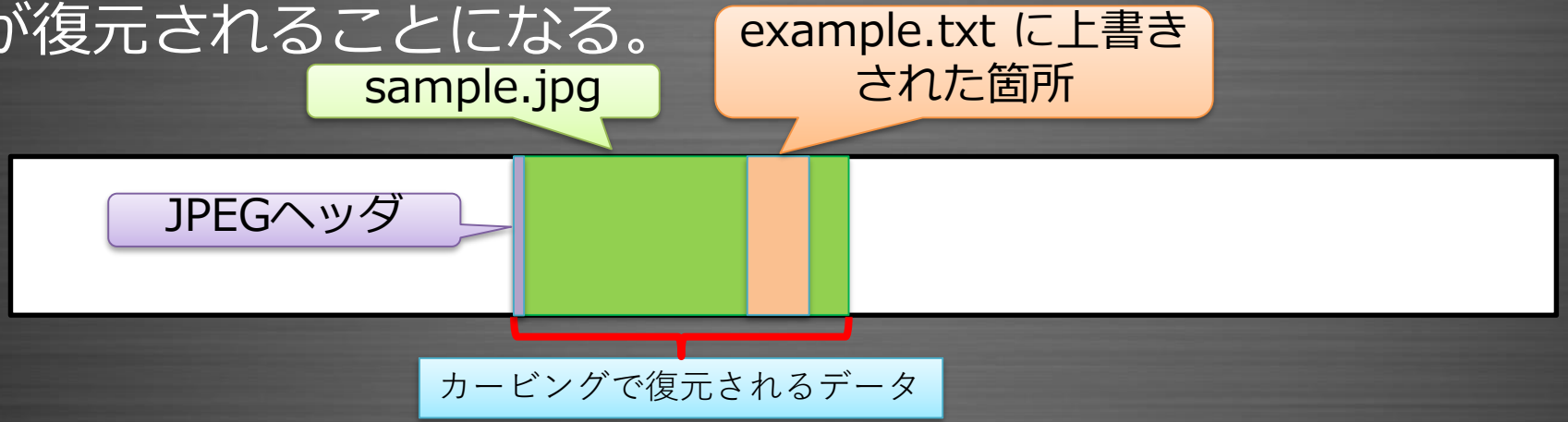
ZIP：PK¥x03¥x04

JPEG：¥xff¥xd8¥xff¥xe0¥x00¥x10



カービングの制限

復元するデータは未使用領域にあるため、他のファイルに上書きされる可能性がある。また、ディスクの使用状況によりファイルがフラグメント化される。このような場合、壊れたファイルが復元されることになる。



また、復元されたデータのメタデータ（ファイル名、ファイルサイズ、作成日時等）も知ることができない。

代表的なカービングツール

Foremost

- http://foremost.sourceforge.net/

Scalpel

- https://github.com/sleuthkit/scalpel

PhotoRec

- https://www.cgsecurity.org/wiki/PhotoRec

カービングに必要なこと

カービングを行うには、復元したいデータのフォーマットを知る必要がある。

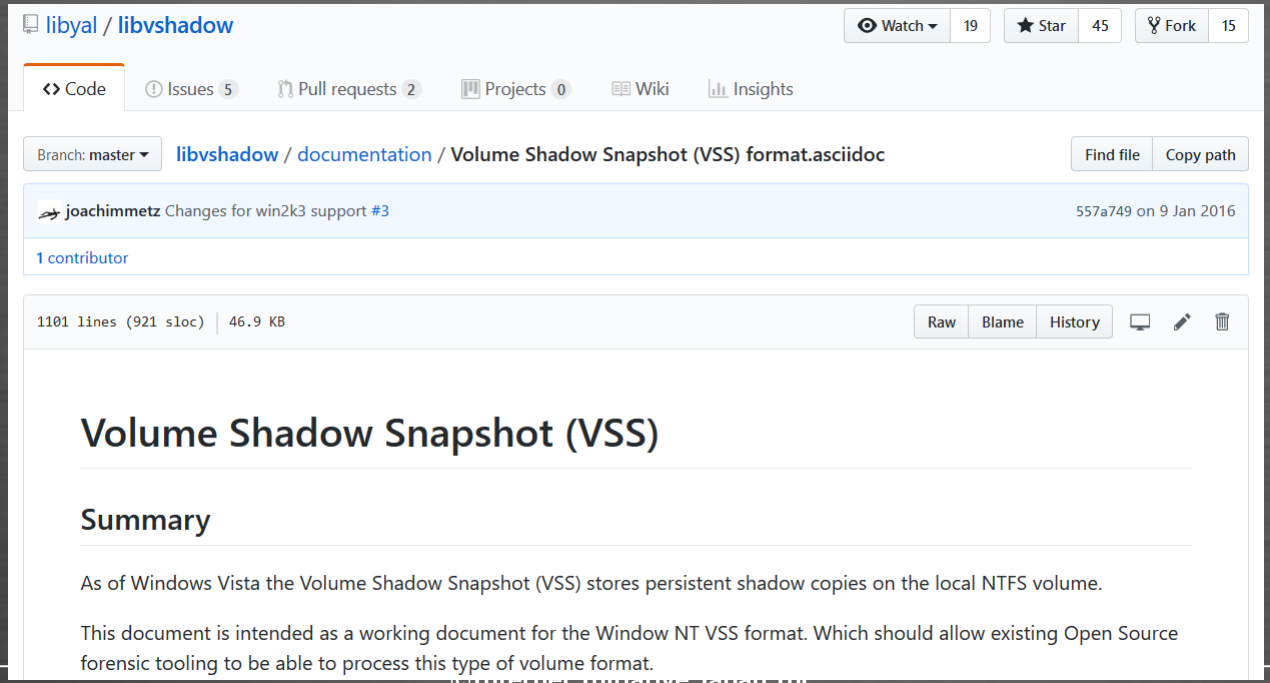
データに生成日時等のメタデータが含まれていれば、それらも解析対象とすることで単なるカービングよりも多くの情報を復元できる可能性がある。

しかし、マイクロソフトはカタログとストアのファイルフォーマットの詳細を公開していない。

VSSデータフォーマット

Volume Shadow Snapshot (VSS)

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



VSSボリュームヘッダ

2. Volume header

The VSS volume header is part of the NTFS volume header (or $Boot metadata file). The VSS volume header data is stored at offset 7680 (0x1e00) of the volume and is at least 100 bytes in size (**but probably 512 bytes, sector size**) and consists of:

Offset	Size	Value	Description
0	16		VSS identifier Contains a GUID
16	4		Version
20	4	0x01	Record type
24	8	0x1e00	Current offset The offset is relative to the start of the volume
32	8	**0x1e00**	**Unknown (Next offset?)** The offset is relative to the start of the volume
40	8		**Unknown (empty value)**
48	8		Catalog offset The offset is relative to the start of the volume 0 if no catalog

カタログブロックヘッダ

3.1. Catalog block header

The catalog block header is 128 bytes of size and consists of:

Offset	Size	Value	Description
0	16		VSS identifier Contains a GUID
16	4	0x01	Version
20	4	0x02	Record type
24	8		Relative (catalog block) offset The offset is relative to the start of t
32	8		Current (catalog block) offset The offset is relative to the start of t
40	8		Next (catalog block) offset The offset is relative to the start of t Contains 0 if this is the last block.

このヘッダの後にカタログエントリが記録される。

カタログエントリタイプ
0x01：未使用
0x02：生成日時など
0x03：ストアブロックのオフセット

0x02 と 0x03 で1セットのエントリとなる。

カタログエントリ(0x02)

3.2.2. Catalog entry type 0x02

A catalog entry type 0x02 is 128 bytes of size and consists of:

Offset	Size	Value	Description
0	8	0x02	Catalog entry type
8	8		Volume size
16	16		Store identifier Contains a GUID This GUID is used in the store filename
32	8		**Unknown (Sequence number)**
40	8		**Unknown (Flags?)** **0x40 ⇒ windows in vista and 7** **0x440 ⇒ in windows 8 (file backup?)**
48	8		Shadow copy creation time Contains a FILETIME
56	72		**Unknown (empty values)**

カタログエントリ(0x03)

3.2.3. Catalog entry type 0x03

A catalog entry type 0x03 is 128 bytes of size and consists of:

Offset	Size	Value	Description
0	8	0x03	Catalog entry type
8	8		Store block list offset The offset is relative to the start of the volume
16	16		Store identifier Contains a GUID This GUID is used in the store filename
32	8		Store header offset The offset is relative to the start of the volume
40	8		Store block range list offset The offset is relative to the start of the volume
48	8		Store (current) bitmap offset The offset is relative to the start of the volume
56	8		NTFS (metadata) file reference
64	8		**Allocated size**
72	8		Store previous bitmap offset The offset is relative to the start of the volume or 0 if not used
			Unknown

ストアブロックヘッダ

4.1. Store block header

The store block header is 128 bytes of size and consists of:

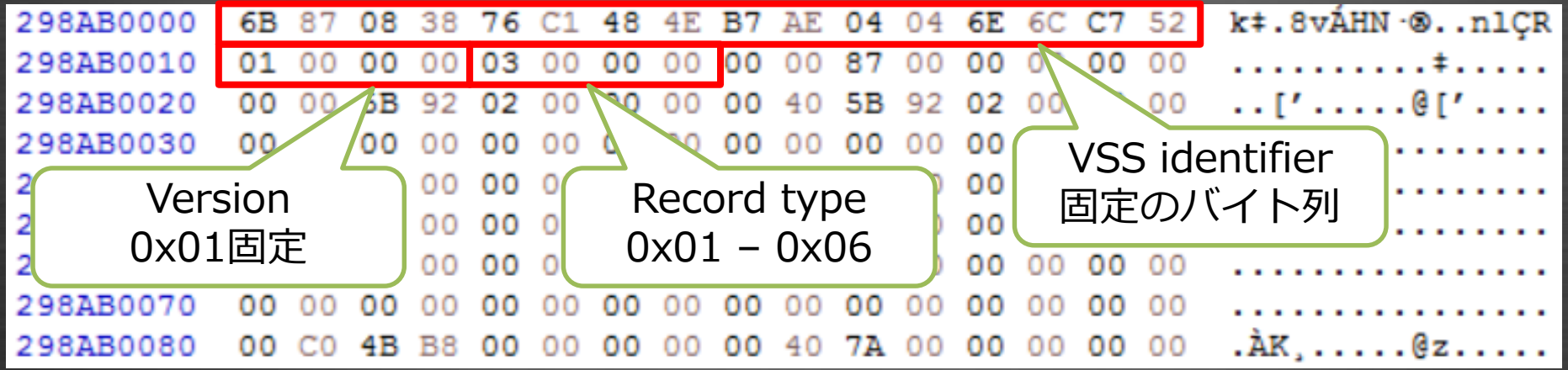
Offset	Size	Value	Description
0	16		VSS identifier Contains a GUID
16	4	0x01	Version
20	4		Record type
24	8		Relative (block) offset The offset is relative to the start of the store
32	8		Current (block) offset The offset is relative to the start of the volum
40	8		Next (block) offset The offset is relative to the start of the volum Contains 0 if this is the last block.
			Size of store information

4.1.1. Store block record types

Value	Identifier	Description
0x0000		**Unknown**
0x0001		Volume header
0x0002		Catalog block header
0x0003		Block descriptor list (Diff area table)
0x0004		Store header
0x0005		**Store block ranges list**
0x0006		Store bitmap

VSSデータカービングの条件

- VSSのデータフォーマットは必ず先頭に“VSS identifier”, “Version”, “Record type”というフィールドが含まれている。
- VSSは16KBごとのデータブロックで管理される。
  ⇒これらは必ずデータブロックの先頭に記録される。



ストアブロック(Store descriptor list)の例

ストアカービング後の問題点

VSS identifierをキーにして、カービングツールを使用すればストアを復元することはできる。しかし、以下の問題点が残る。

1. ストアは4種類5つのストアブロックから構成されるため、これらのグルーピングが必要となる。
2. カタログを再生成する必要がある（スナップショット削除時に0で上書きされているため）。

⇒カービングを行うだけでは不十分であるため、
専用のツールが必要になる。

カービングツールのプロトタイプを作成

VSS identifierとレコードタイプを基に、VSSデータブロックがどのように配置されているか確認するツールを作成した。

- スナップショットの順番とストアブロックの情報に何らかの相関関係などが見られないか確認する。

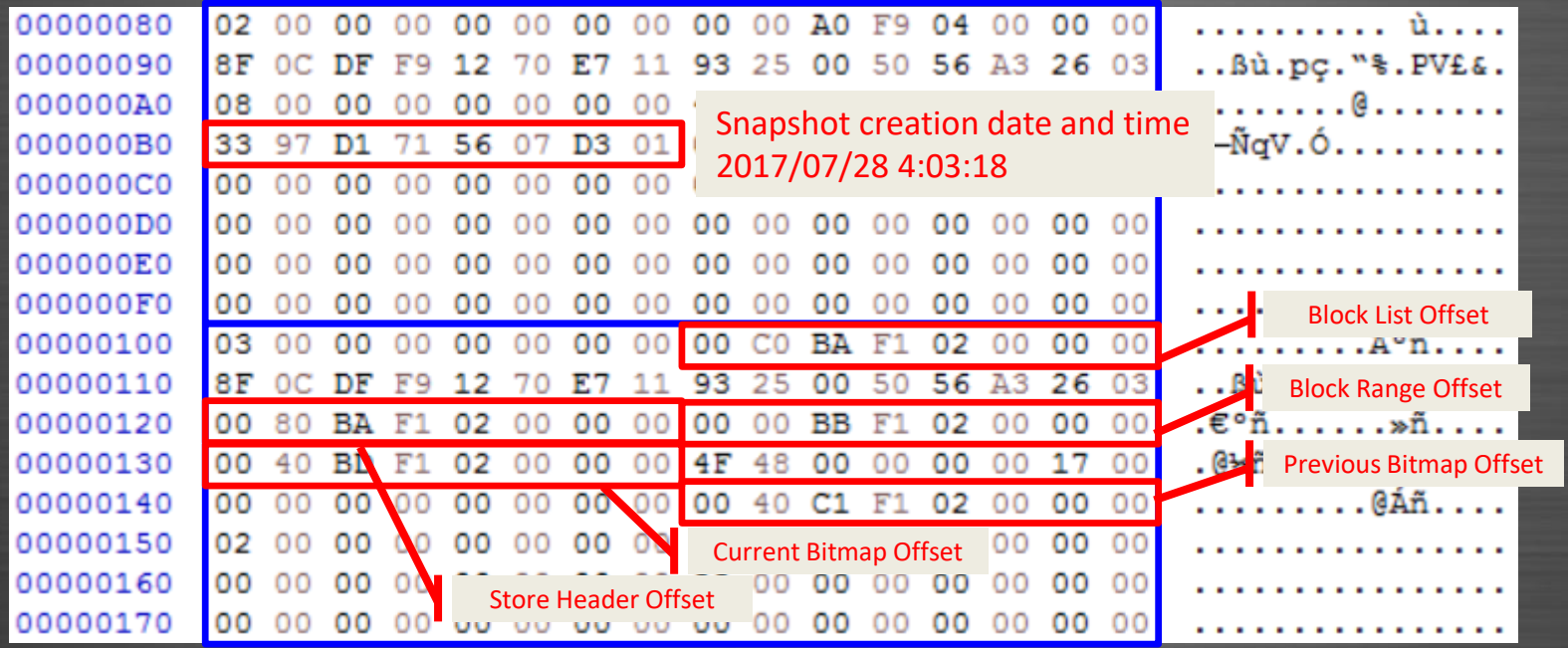
- ストアブロックがどのように配置されているのか確認する。スナップショット毎にストアブロックがある程度グループ分けできるのであれば、比較的簡単にストアを復元できると考えられる。

```
0x1e00 : 6b87083876c1484eb7ae04046e6cc752010000000100000001e000000000000001e000000000000000000000000000000
0x9c58000-0x9c64000(0x10000) : Ver:1 RType:2 Next:0x0
0x278bec000L-0x278ce8000L(0x100000L) : Ver:1 RType:3 Next:0x0
0x291d40000L-0x291d40000L(0x4000L) : Ver:1 RType:4 Next:0x0
0x291d44000L-0x291d44000L(0x4000L) : Ver:1 RType:3 Next:0x2925b0000L
0x291d48000L-0x291d48000L(0x4000L) : Ver:1 RType:5 Next:0x0
0x291d78000L-0x291da0000L(0x2c000L) : Ver:1 RType:6 Next:0x0
0x291de0000L-0x291e08000L(0x2c000L) : Ver:1 RType:6 Next:0x0
0x2925b0000L-0x2926ac000L(0x100000L) : Ver:1 RType:3 Next:0x2cbf88000L
0x2a2b88000L-0x2a2c84000L(0x100000L) : Ver:1 RType:3 Next:0x0
0x2ab5f4000L-0x2ab6f0000L(0x100000L) : Ver:1 RType:3 Next:0x0
0x2ae918000L-0x2ae918000L(0x4000L) : Ver:1 RType:4 Next:0x0
0x2ae91c000L-0x2ae91c000L(0x4000L) : Ver:1 RType:3 Next:0x2af1b0000L
0x2ae920000L-0x2ae920000L(0x4000L) : Ver:1 RType:5 Next:0x0
0x2ae93c000L-0x2ae964000L(0x2c000L) : Ver:1 RType:6 Next:0x0
0x2ae978000L-0x2ae9a0000L(0x2c000L) : Ver:1 RType:6 Next:0x0
0x2af1b0000L-0x2af2ac000L(0x100000L) : Ver:1 RType:3 Next:0x2b5610000L
0x2b5610000L-0x2b570c000L(0x100000L) : Ver:1 RType:3 Next:0x0
0x2cbf88000L-0x2cc084000L(0x100000L) : Ver:1 RType:3 Next:0x278bec000L
0x2e8044000L-0x2e8140000L(0x100000L) : Ver:1 RType:3 Next:0x2f81d8000L
0x2f1ba8000L-0x2f1ba8000L(0x4000L) : Ver:1 RType:4 Next:0x0
0x2f1bac000L-0x2f1bac000L(0x4000L) : Ver:1 RType:3 Next:0x2f2490000L
0x2f1bb0000L-0x2f1bb0000L(0x4000L) : Ver:1 RType:5 Next:0x0
0x2f1bd4000L-0x2f1bfc000L(0x2c000L) : Ver:1 RType:6 Next:0x0
0x2f1c14000L-0x2f1c3c000L(0x2c000L) : Ver:1 RType:6 Next:0x0
0x2f2490000L-0x2f258c000L(0x100000L) : Ver:1 RType:3 Next:0x0
0x3065c4000L-0x3065c4000L(0x4000L) : Ver:1 RType:4 Next:0x0
0x3065c8000L-0x3065c8000L(0x4000L) : Ver:1 RType:3 Next:0x306e84000L
0x3065cc000L-0x3065cc000L(0x4000L) : Ver:1 RType:5 Next:0x0
0x3065e4000L-0x30660c000L(0x2c000L) : Ver:1 RType:6 Next:0x0
0x30661c000L-0x3066440000L(0x2c000L) : Ver:1 RType:6 Next:0x0
0x306e84000L-0x306f80000L(0x100000L) : Ver:1 RType:3 Next:0x31820c000L
0x31820c000L-0x3183080000L(0x100000L) : Ver:1 RType:3 Next:0x2a2b88000L
```

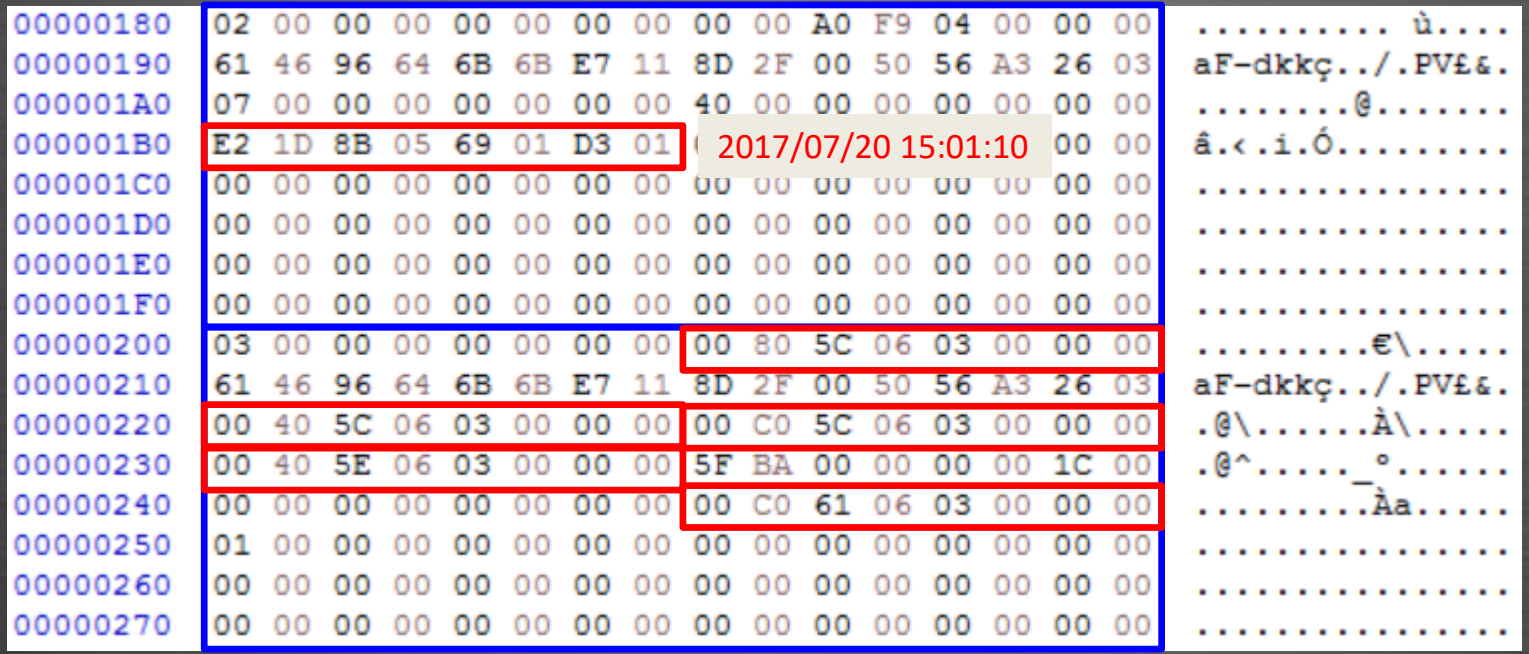
3つのスナップショットが保存されたディスクイメージを解析した結果。

カタログエントリ 1

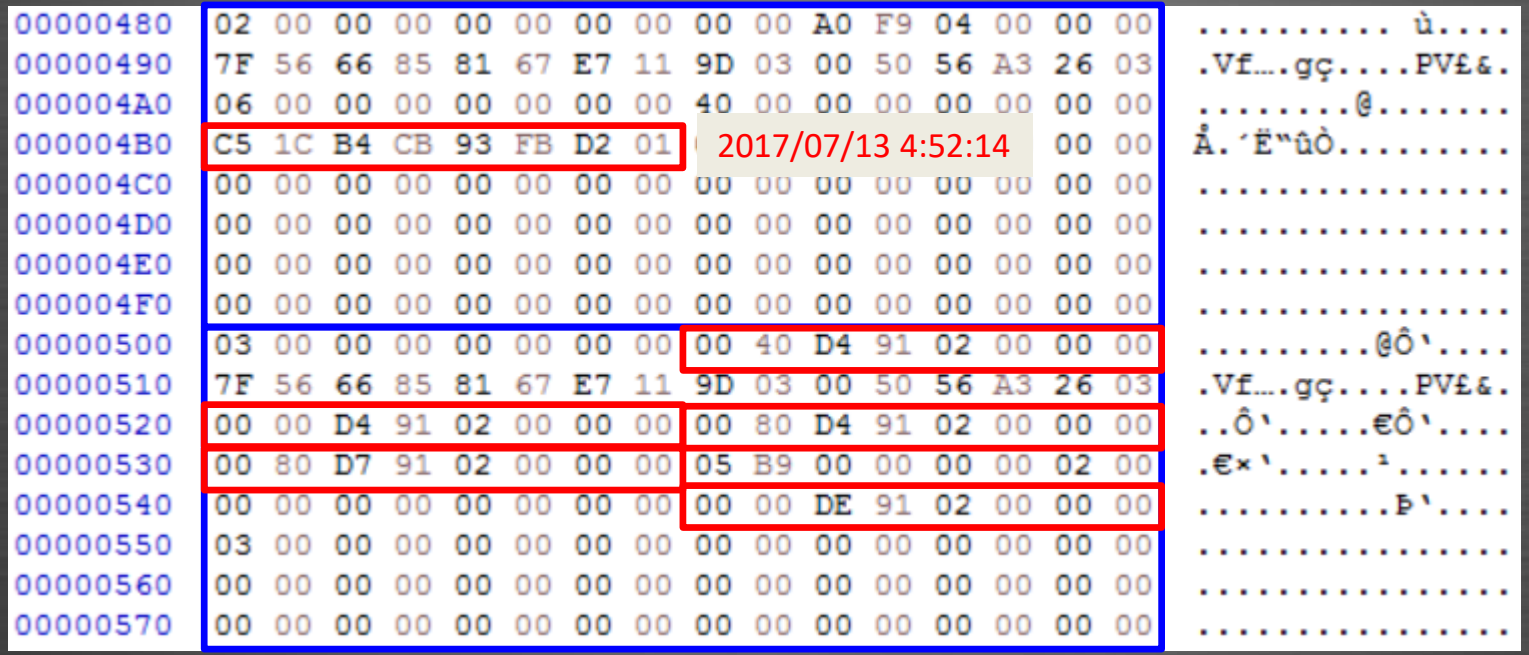
このディスクイメージのカタログを確認する。

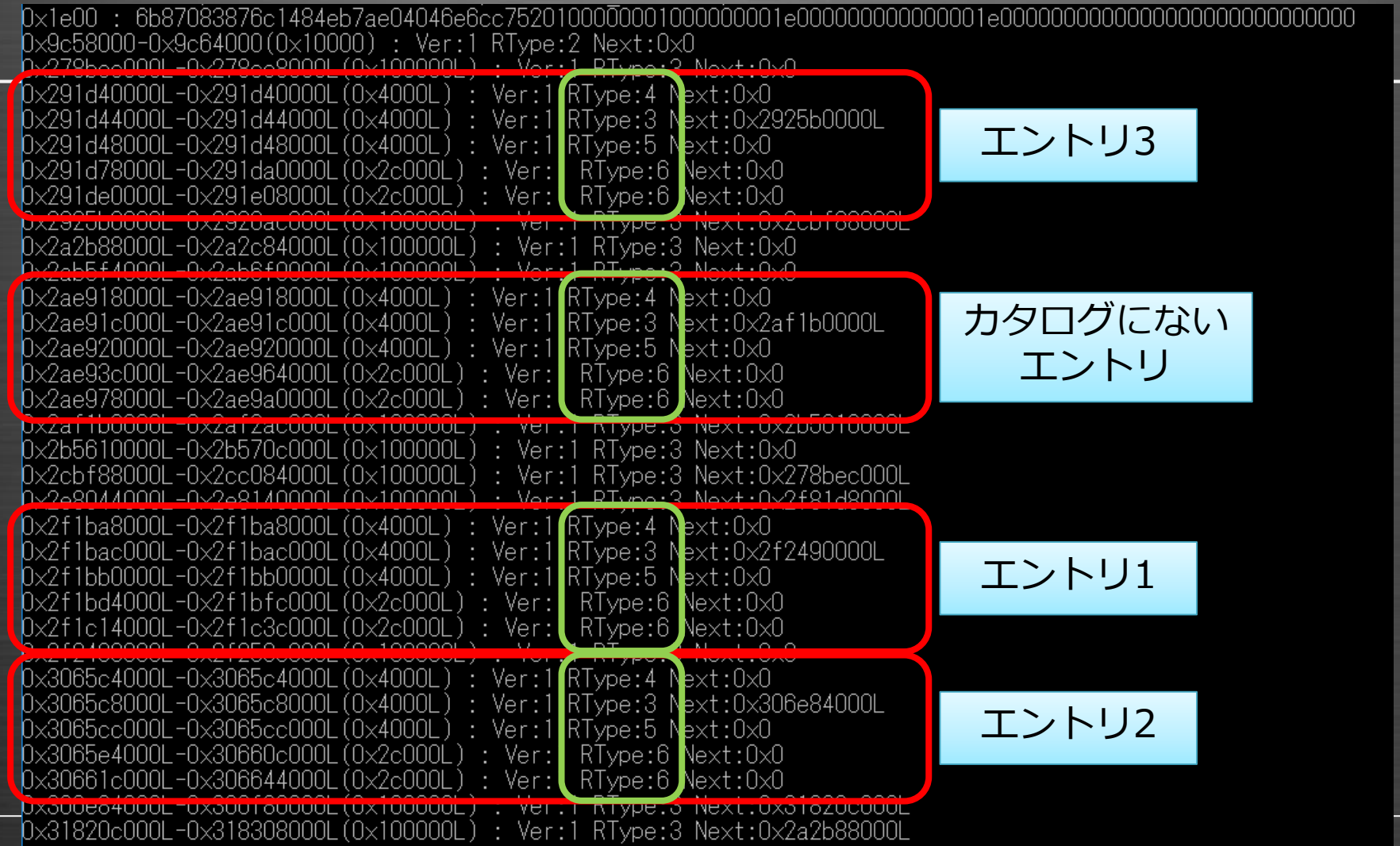


カタログエントリ 2



カタログエントリ 3




0x1e00 : 6b87083876c1484eb7ae04046e6cc752010000000100000001e000000000000001e000000000000000000000000000000
0x9c58000-0x9c64000(0x10000) : Ver:1 RType:2 Next:0x0
0x291d40000L-0x291d40000L(0x4000L) : Ver:1 RType:4 Next:0x0
0x291d44000L-0x291d44000L(0x4000L) : Ver:1 RType:3 Next:0x2925b0000L
0x291d48000L-0x291d48000L(0x4000L) : Ver:1 RType:5 Next:0x0
0x291d78000L-0x291da0000L(0x2c000L) : Ver: RType:6 Next:0x0
0x291de0000L-0x291e08000L(0x2c000L) : Ver: RType:6 Next:0x0
0x2a2b88000L-0x2a2c84000L(0x100000L) : Ver:1 RType:3 Next:0x0
0x2ae918000L-0x2ae918000L(0x4000L) : Ver:1 RType:4 Next:0x0
0x2ae91c000L-0x2ae91c000L(0x4000L) : Ver:1 RType:3 Next:0x2af1b0000L
0x2ae920000L-0x2ae920000L(0x4000L) : Ver:1 RType:5 Next:0x0
0x2ae93c000L-0x2ae964000L(0x2c000L) : Ver: RType:6 Next:0x0
0x2ae978000L-0x2ae9a0000L(0x2c000L) : Ver: RType:6 Next:0x0
0x2b5610000L-0x2b570c000L(0x100000L) : Ver:1 RType:3 Next:0x0
0x2cbf88000L-0x2cc084000L(0x100000L) : Ver:1 RType:3 Next:0x278bec000L
0x2f1ba8000L-0x2f1ba8000L(0x4000L) : Ver:1 RType:4 Next:0x0
0x2f1bac000L-0x2f1bac000L(0x4000L) : Ver:1 RType:3 Next:0x2f2490000L
0x2f1bb0000L-0x2f1bb0000L(0x4000L) : Ver:1 RType:5 Next:0x0
0x2f1bd4000L-0x2f1bfc000L(0x2c000L) : Ver: RType:6 Next:0x0
0x2f1c14000L-0x2f1c3c000L(0x2c000L) : Ver: RType:6 Next:0x0
0x3065c4000L-0x3065c4000L(0x4000L) : Ver:1 RType:4 Next:0x0
0x3065c8000L-0x3065c8000L(0x4000L) : Ver:1 RType:3 Next:0x306e84000L
0x3065cc000L-0x3065cc000L(0x4000L) : Ver:1 RType:5 Next:0x0
0x3065e4000L-0x30660c000L(0x2c000L) : Ver: RType:6 Next:0x0
0x30661c000L-0x306644000L(0x2c000L) : Ver: RType:6 Next:0x0
0x31820c000L-0x318308000L(0x100000L) : Ver:1 RType:3 Next:0x2a2b88000L
エントリ3
カタログにない
エントリ
エントリ1
エントリ2

ストアカービングの検討 1

ストアの容量のほとんどはストアデータブロック（バックアップデータ）が占めている。

ストアデータブロック自身は、ストアブロックリスト（レコードタイプ:3）によって、そのデータブロックが記録されていたオリジナルのオフセットとバックアップ先のオフセットがテーブルで管理されている。

ストアカービングの検討 2

4.3. Store block list

The store block list contains information about the data block ranges used by the snapshot.

The store block list is stored in blocks of 16384 (0x4000) bytes. Each store block list block consists of:

- a store block header of type 3
- an array of store block descriptors

4.3.1. Block descriptor

The block descriptor is 32 bytes of size and consists of:

Offset	Size	Value	Description
0	8		Original data block offset The offset is relative to the start of the volume
8	8		Relative store data block offset The offset is relative to the start of the store **lower bits used for different purpose?**
16	8		Store data block offset The offset is relative to the start of the volume
24	4		Flags
28	4		Allocation bitmap **Used if flag 0x02 is set, otherwise is should contain a value of 0**

オリジナルの
オフセット

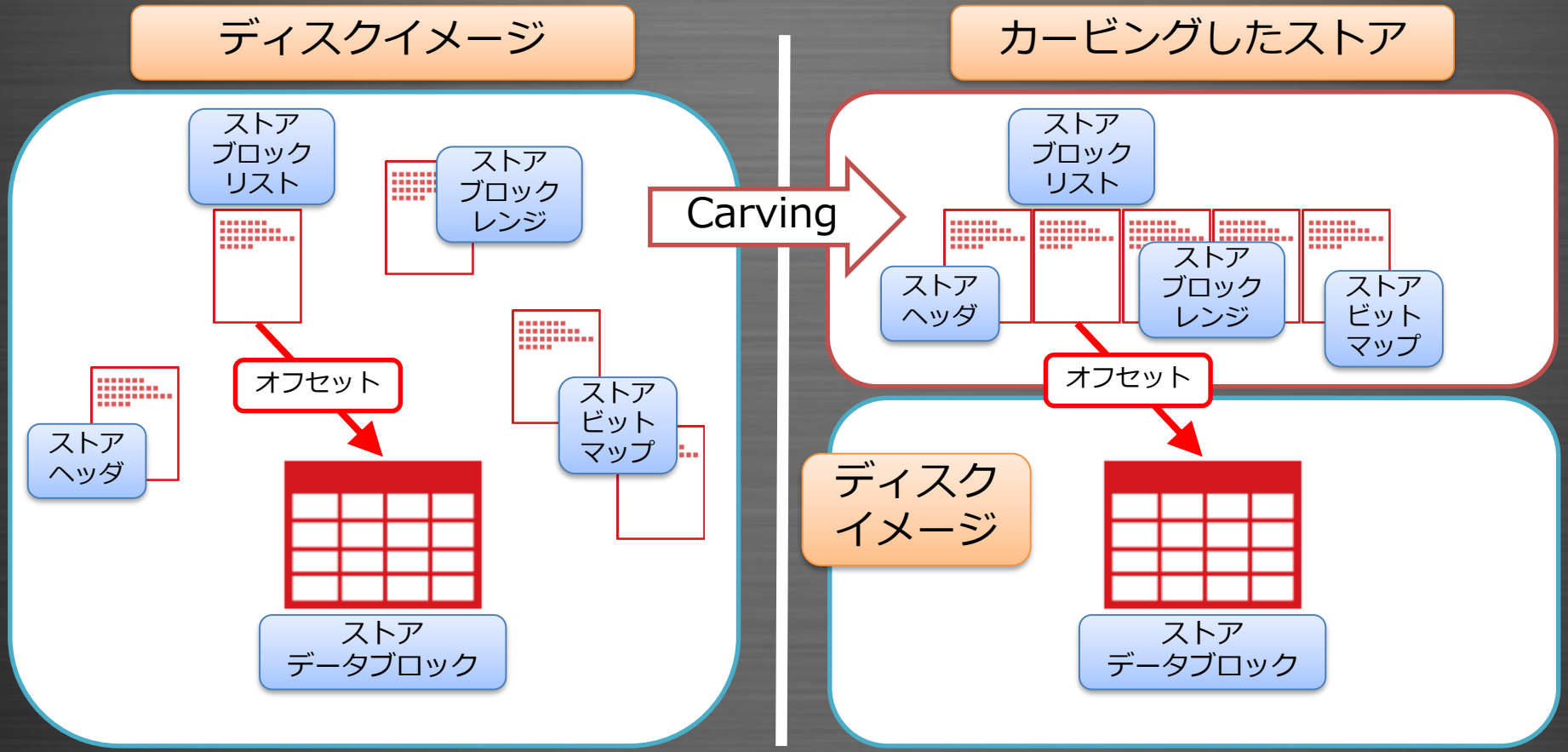
バックアップ先の
オフセット

ストアカービングの検討 3

ストアデータブロックをすべてカービングすると非常に大きなファイルとなってしまい扱いづらい。また、ストアデータブロックを別ファイルにする場合、ストアブロックリスト内のオフセットをすべて書き換えなければならないため、煩雑な処理が必要となる。

実用上、ディスクイメージ内のストアデータブロックを参照してもスナップショットのデータを読み込むことが可能であるため、ストアデータブロックを除いたストアブロックのみを別ファイルとして保存する。

ストアカービングの検討 4



カタログ再生成方法の検討 1

カービングでカタログも復元することができるが、データは”0”で上書きされているため、カービングによる復元ではなく再生成する必要がある。

カタログ内のストアブロックのオフセットを見ると、スナップショットの生成日時とストアブロックのオフセットには相関関係は見られない。

カタログ再生成方法の検討 2

そのため、大きなオフセットのスナップショットほど新しいと仮定して、カタログを生成する。

ただし、ディスクイメージ上にカタログがある場合は、そちらの情報を優先的に使用する。

また、カービングで発見したスナップショットのうち、カタログに無いものは、カタログ内の一番古いエントリよりも古いものとして扱う。
※ディスクイメージ上にカタログが存在するということは、カービングしたスナップショットは自動的に削除された古いものである可能性が高い。

検討結果

ここまでの結果から、削除済みスナップショットのデータを復元するツールと、それにアクセスするためのツールを作成できる見通しが立った。

3. vss_carverの処理

vss_carverの処理

1. VSS設定チェック
2. ディスク上のカタログ読み込み
3. ストアブロックカービング
4. カービングしたストアブロックをグルーピング
5. ストアブロックリストの精査
6. スナップショットエントリの重複チェック
7. ストア書き出し
8. カタログ書き出し

1. VSS設定チェック

ボリュームの先頭からのオフセット0x1e00にVSS identifierのバイト列が存在するか否かで、VSSが有効か無効か確認する。

VSSボリュームヘッダのカタログオフセットが”0”の場合、VSSは有効であるがスナップショットはすべて削除されている（または1つも作成されていない）状態である。

VSSが無効である場合はプログラムを終了する。

2. ディスク上のカタログ読み込み

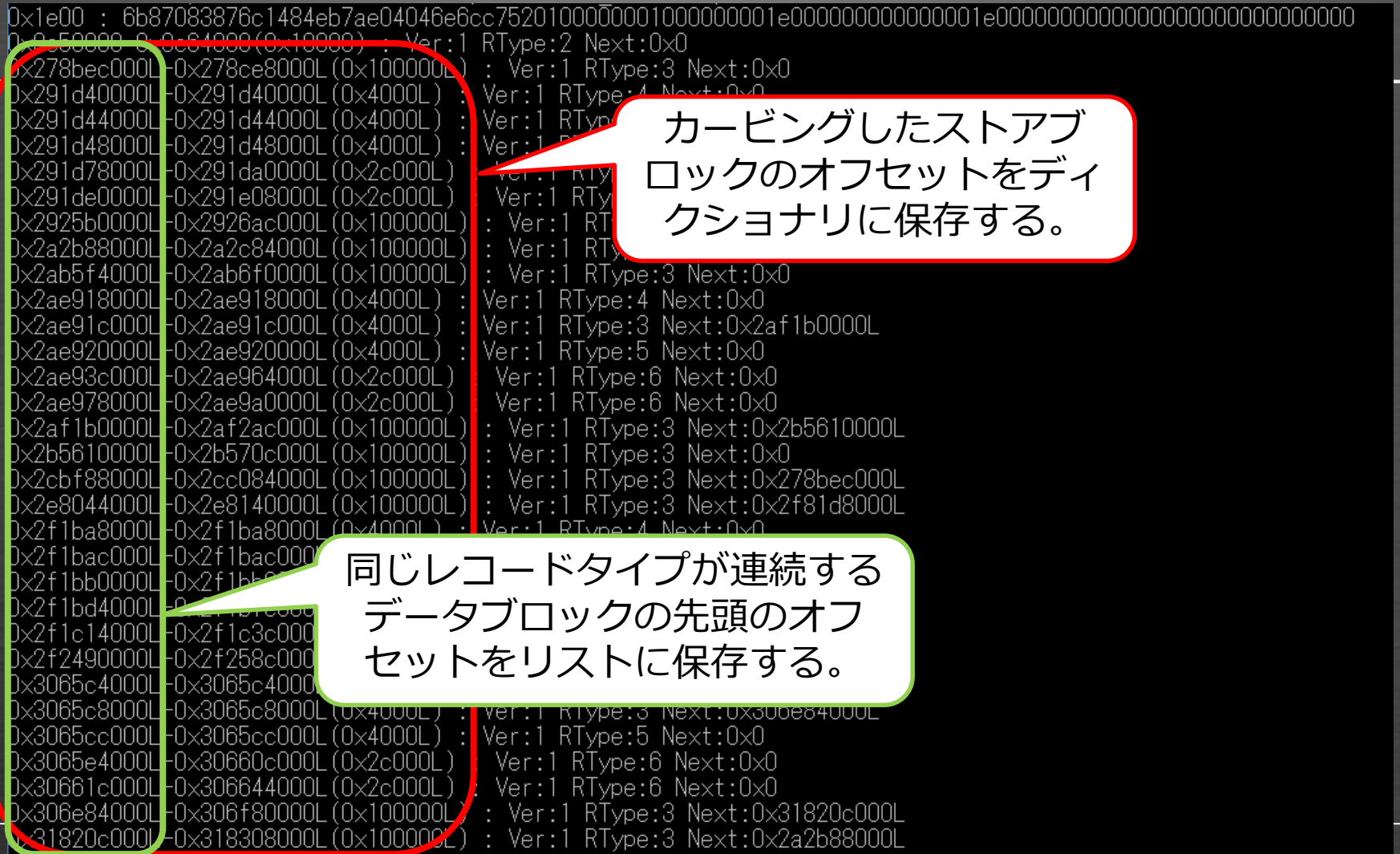
カービングによって得られたスナップショットの情報よりもWindowsに作成されたカタログの方が信頼性が高いため、ディスク上にカタログがあれば読み込む。

3. ストアブロックカービング

ボリュームの先頭から16KBごとに、VSS identifierとレコードタイプをキーにしてストアブロックをカービングする。

カービングで取得したストアブロックのオフセットをキーにして、ディクショナリにストアヘッダのデータを保存する（ストアブロックディクショナリ）。

また、同じレコードタイプが連続するストアブロックの先頭のオフセットをリストに保存する（ストアブロックリスト）。


0x1e00 : 6b87083876c1484eb7ae04046e6cc752010000000100000001e000000000000001e000000000000000000000000000000
Ver:1 RType:2 Next:0x0
0x278bec000L-0x278ce8000L(0x100000L) : Ver:1 RType:3 Next:0x0
0x291d40000L-0x291d40000L(0x4000L) : Ver:1 RType:4 Next:0x0
0x291d44000L-0x291d44000L(0x4000L) : Ver:1 RTyp
0x291d48000L-0x291d48000L(0x4000L) : Ver:1 R
0x291d78000L-0x291da0000L(0x2c000L) : Ver:1 RTy
0x291de0000L-0x291e08000L(0x2c000L) : Ver:1 RTy
0x2925b0000L-0x2926ac000L(0x100000L) : Ver:1 RT
0x2a2b88000L-0x2a2c84000L(0x100000L) : Ver:1 RTy
0x2ab5f4000L-0x2ab6f0000L(0x100000L) : Ver:1 RType:3 Next:0x0
0x2ae918000L-0x2ae918000L(0x4000L) : Ver:1 RType:4 Next:0x0
0x2ae91c000L-0x2ae91c000L(0x4000L) : Ver:1 RType:3 Next:0x2af1b0000L
0x2ae920000L-0x2ae920000L(0x4000L) : Ver:1 RType:5 Next:0x0
0x2ae93c000L-0x2ae964000L(0x2c000L) : Ver:1 RType:6 Next:0x0
0x2ae978000L-0x2ae9a0000L(0x2c000L) : Ver:1 RType:6 Next:0x0
0x2af1b0000L-0x2af2ac000L(0x100000L) : Ver:1 RType:3 Next:0x2b5610000L
0x2b5610000L-0x2b570c000L(0x100000L) : Ver:1 RType:3 Next:0x0
0x2cbf88000L-0x2cc084000L(0x100000L) : Ver:1 RType:3 Next:0x278bec000L
0x2e8044000L-0x2e8140000L(0x100000L) : Ver:1 RType:3 Next:0x2f81d8000L
0x2f1ba8000L-0x2f1ba8000L(0x4000L) : Ver:1 RType:4 Next:0x0
0x2f1bac000L-0x2f1bac000
0x2f1bb0000L-0x2f1bb
0x2f1bd4000L-0
0x2f1c14000L-0x2f1c3c000
0x2f24900000L-0x2f258c000
0x3065c4000L-0x3065c4000
0x3065c8000L-0x3065c8000L(0x4000L) : Ver:1 RType:3 Next:0x306e84000L
0x3065cc000L-0x3065cc000L(0x4000L) : Ver:1 RType:5 Next:0x0
0x3065e4000L-0x30660c000L(0x2c000L) : Ver:1 RType:6 Next:0x0
0x30661c000L-0x306644000L(0x2c000L) : Ver:1 RType:6 Next:0x0
0x306e84000L-0x306f80000L(0x100000L) : Ver:1 RType:3 Next:0x31820c000L
0x31820c000L-0x318308000L(0x100000L) : Ver:1 RType:3 Next:0x2a2b88000L
カービングしたストアブロックのオフセットをディクショナリに保存する。
同じレコードタイプが連続するデータブロックの先頭のオフセットをリストに保存する。

4. カービングしたストアブロックをグルーピング

カタログに記録されたストアブロックのオフセットおよびプロトタイプの実行結果より、ストアブロックのレコードタイプの並び順は、4, 3, 5, 6, 6であることが分かっている。

ストアブロックリストの中で、この順番で並んでいるストアブロック群を1つのスナップショットとしてグルーピングし、スナップショットリストに保存する。

5. ストアブロックリストの精査 1

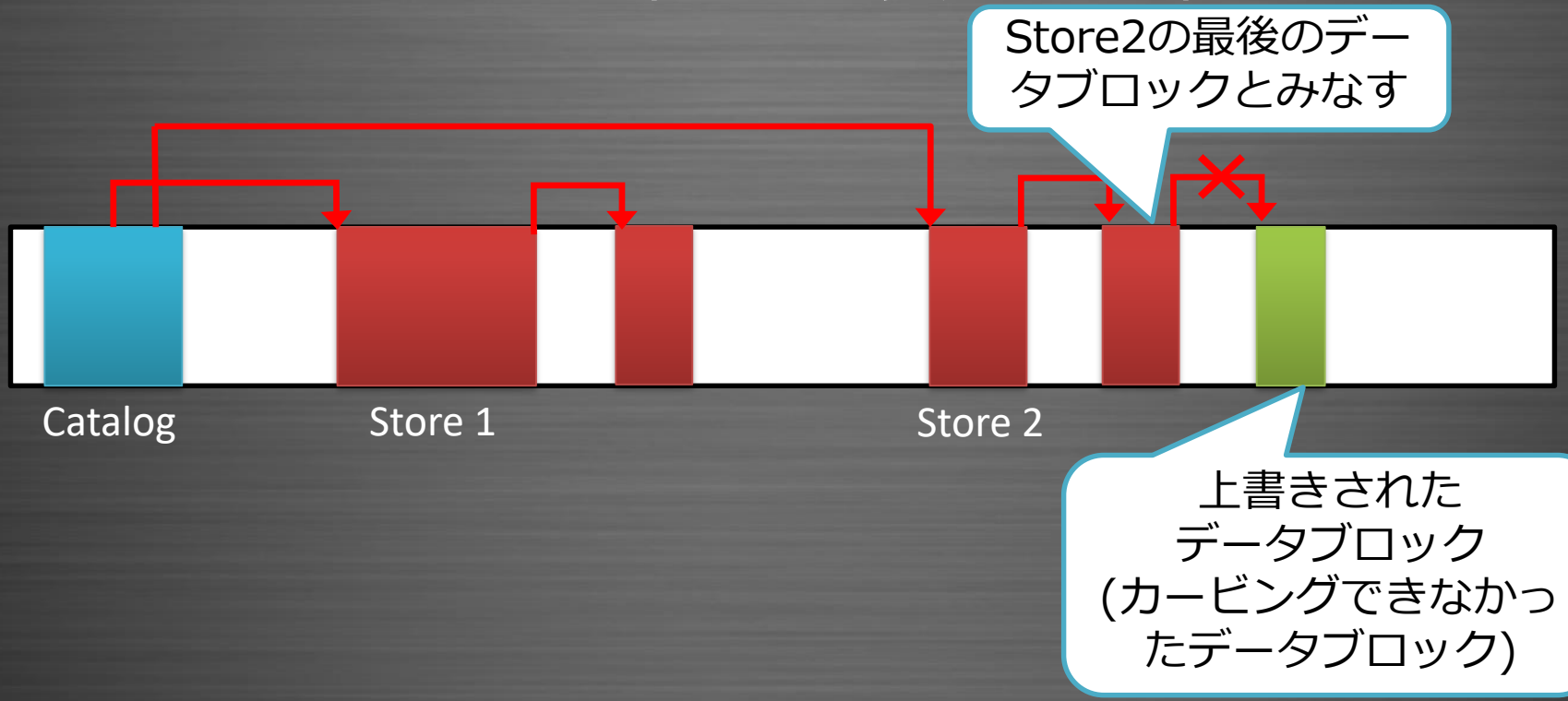
ストアブロックには、Next block offsetというフィールドが存在する。VSSによるバックアップ容量が確保しているバックアップ領域より大きくなる場合、新たにストアブロックを確保し、そのオフセットをNext block offsetに記録する。

ここでは、スナップショットリスト内のストアブロックのNext block offsetのリストを辿っていき、全てのNext block offsetがストアブロックディクショナリのキーとして存在するか確認する。

ストアブロックが他のデータで上書きされ、該当のオフセットがストアブロックとしてカービングできなかった場合、そこまでのリストとみなす。

5. ストアブロックリストの精査 2

Next block offsetによるストアブロックのリスト



6. スナップショットエントリの重複チェック

2. で読み込んだカタログのスナップショットエントリと、5. までに復元したスナップショットエントリが重複していないか確認する。

重複のチェックはストアヘッダのオフセットを比較することで行う。

重複している復元エントリがある場合、スナップショットリストから該当するエントリを削除する。

7. & 8. ファイル書き出し

ストアとカタログをファイルに保存する。

復元したスナップショットの生成日時の決め方

・ディスク上にカタログがある場合

　カタログ上の一番古いスナップショット生成日時よりも1時間前の日時を設定する。

・ディスク上にカタログがない場合

　vss_carverを実行した時間を基準に設定する。

4. libvshadowの改修

libvshadowの改修

1. 復元したカタログとストアを読み込むオプションの追加

2. カタログとストアをファイルから読み込む処理の追加

改修した点は非常に単純だが、libvshadowの作者は自作のライブラリを多用しており、ソースコードを読み解くのに時間がかかった。

5. vss_catalog_manipulator

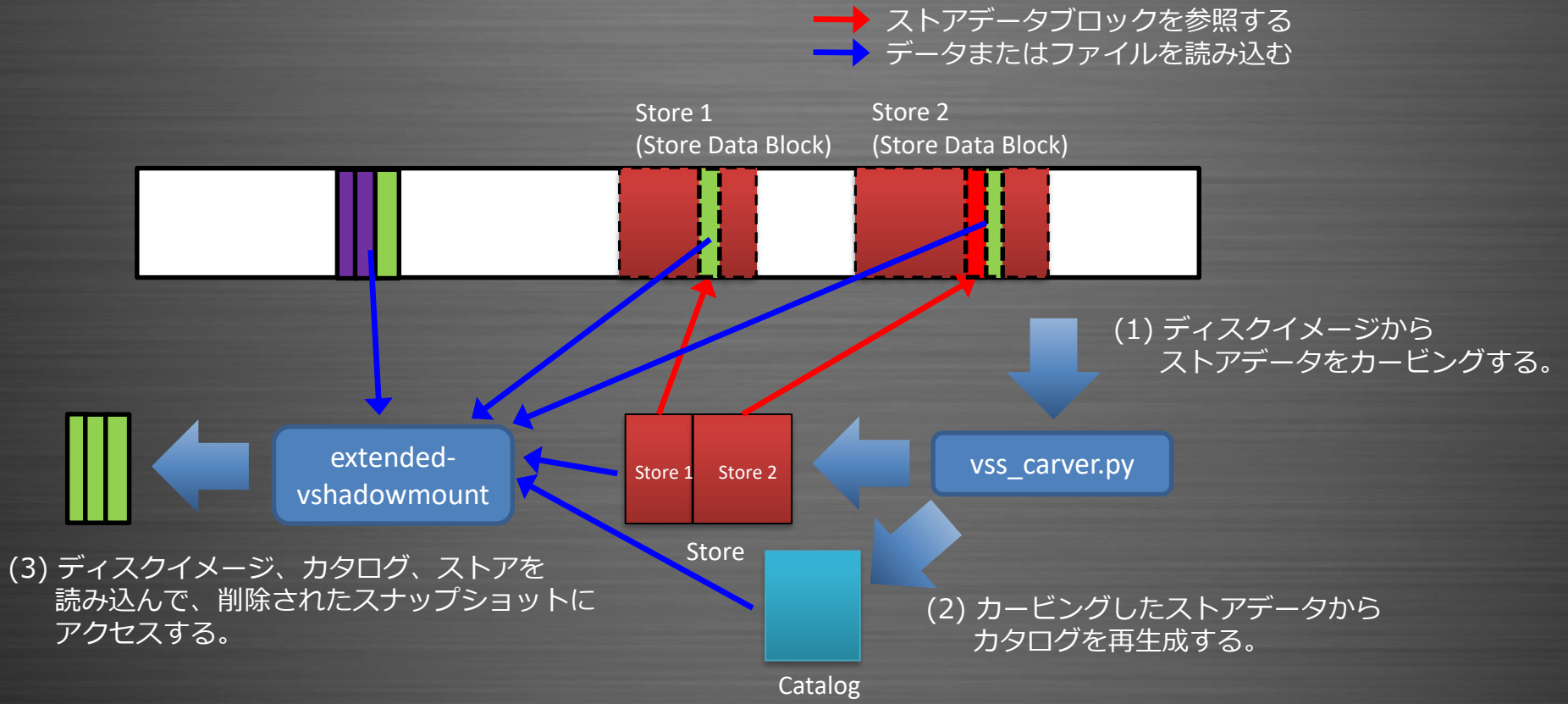
vss_catalog_manipulator

カタログエントリを操作するためのツール。

vss_carverはストアブロックのオフセットを基準にエントリの順番を決めているため、実際は正しくない場合がある。このような場合にエントリの入れ替えや削除等を行う。

独自の機能として、エントリを無効化／有効化する機能がある。これはカタログエントリタイプが0x01が未使用を表すタイプとして定義されていることを利用している（libvshadowと組み合わせることが前提）。

Tools overview



6. デモンストレーション

デモンストレーション

1. 自動的に削除されたスナップショットの復元

2. ランサムウェアに削除されたスナップショットの復元

7. 今後の開発

今後の開発予定

- libvshadowのパッチを本家にマージ
  - Pull Requestはsubmit済み
- 復元したスナップショットの生成日時の推定
- （上記を利用した）カタログエントリソートツールの作成

8. まとめ

まとめ 1

- 調査
  先人の知見を借りる。
  スナップショット削除時のVSSの挙動
  VSSデータフォーマット、ツールのソースコード
- 検証
  様々な環境から検証用データを取得する。
  仕様の確認、検証用ツールの作成
- 実装
  頑張る。

まとめ 2

- 作成したツールは以下のURLで公開しています。
  https://github.com/mnrkbys/vss_carver
  libvshadowのWindows版とLinux (SIFT)版のバイナリを配布

- ツール公開してからフィードバックをくれた人
  5人くらい

関連発表資料

- Internet Infrastructure Review（IIR）Vol.37
  VSSはユーザデータを守らない
  - https://www.iij.ad.jp/dev/report/iir/037.html
- Japan Security Analyst Conference 2018
  削除済みVSSスナップショットの復元
  - https://www.jpcert.or.jp/present/2018/JSAC2018_02_kobayashi.pdf
- Black Hat USA 2018 Briefings
  Reconstruct the World from Vanished Shadow: Recovering Deleted VSS Snapshots
  - https://www.blackhat.com/us-18/briefings/schedule/#reconstruct-the-world-from-vanished-shadow-recovering-deleted-vss-snapshots-9931

Ongoing Innovation

IIJ Internet Initiative Japan

ご清聴ありがとうございました