304 – Find a confidential file

Team Information

Team Name : DogeCoin

Team Member : Dongbin Oh, Donghyun Kim, Donghyun Kim, Yeongwoong Kim

Email Address : dfc-dogecoin@naver.com

Instructions

Description The person, who is suspected of leaking a company's core technology, has a high level of computer knowledge, so he has safely stored the company's secret technical information in his PC. An investigator secured 1 hard disk and 1 USB storage of the suspect, but the USB was already formatted. There are some traces of browsing the company's confidential file, 'Small Modular Reactor.png', on the disk, but the original file does not exist. Find the confidential file hided by the suspect.

Target	Hash (MD5)
Atom_disk_0.dd	c34a205bbed1fc1c155ae00f64dce790
Atom_USB.dd	6f12b5dc4fceb4d1dfd87d8de4b06014

Questions Note, the suspect used Windows 10. You should analyze the following questions based on UTC+9 time zone.

1. What is the exact capacity of the encrypted volume? (40 points)
2. When was the volume (#1's encrypted volume) encrypted? (40 points)
3. Calculate SHA1 hash value of the confidential file (Small Modular Reactor.png). (220 points)

Teams must:

- Develop and document the step-by-step approach used to solve this problem to allow another examiner to replicate team actions and results.
- Specify all tools used in deriving the conclusion(s).

Tools used:

Name:	FTK Imager	Publisher:	Access Data
Version:	4.5.0.3		
URL:	https://accessdata.com/product-download/ftk-imager-version-4-5		

Name:	Autopsy	Publisher:	Basis Technology
Version:	4.18.0		
URL:	https://www.autopsy.com/download/		

Name:	HxD	Publisher:	Maël Hörz
Version:	2.5.0.0		
URL:	https://www.nirsoft.net/utils/recent_files_view.html		

Name:	PowerShell	Publisher:	Microsoft
Version:	5.1.19041.1023		
URL:	-		

Name:	HashMyFiles	Publisher:	NirSoft
Version:	2.38		
URL:	https://www.nirsoft.net/utils/hash_my_files.html		

Step-by-step methodology:

표현

Atom_disk_0.dd → **로컬 디스크 이미지**

Atom_USB.dd → **USB 이미지**

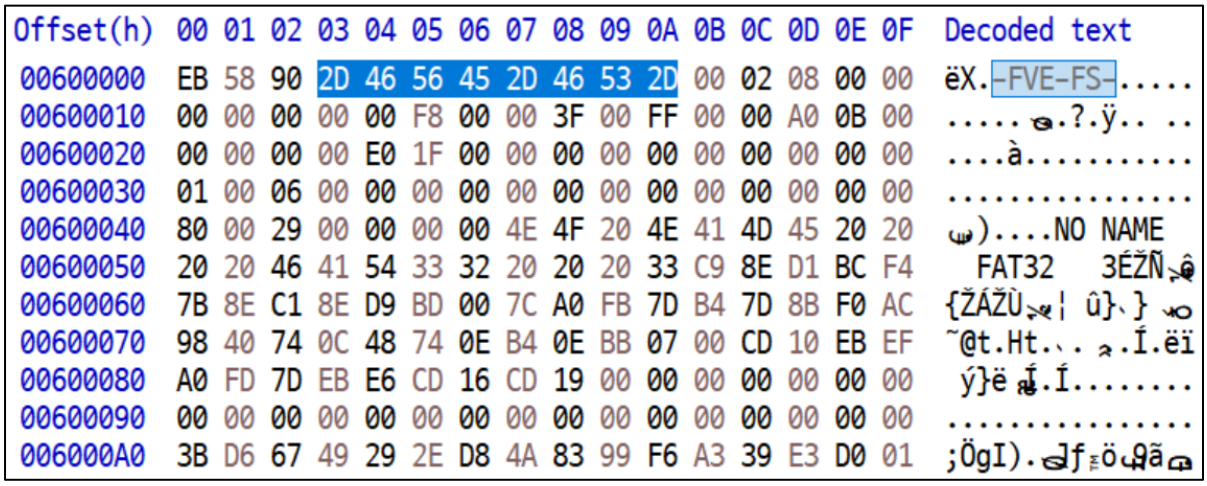
 # 데이터 구조를 설명할 때는 Reference가 되는 문서에서 표기한 영역명, 필드명 등을 따랐다.

1. 암호화된 볼륨의 크기

1.1. 암호화된 볼륨 찾기

암호화된 볼륨을 찾기 위해 Windows 환경에서 대표적으로 사용되는 볼륨 암호화 프로그램 (Bitlocker, TrueCrypt 등) 들의 특징을 기반으로 탐색을 수행하였다.

이에 로컬 디스크 이미지의 미할당 영역을 추출한 후, 시그니처 탐색을 수행한 결과, Bitlocker Drive Encryption (BDE) 구조를 식별할 수 있었다.



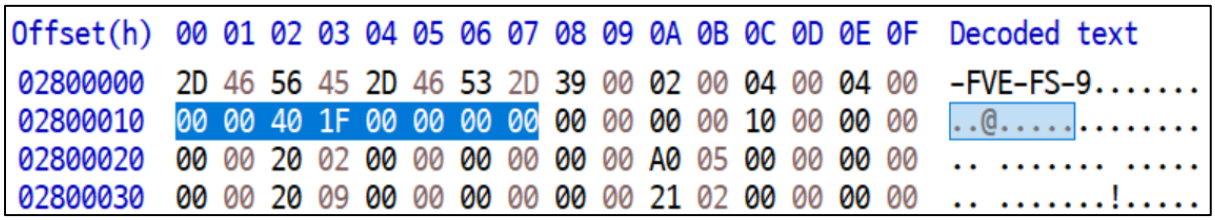
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00600000	EB	58	90	2D	46	56	45	2D	46	53	2D	00	02	08	00	00	ëX.-FVE-FS-.....
00600010	00	00	00	00	00	F8	00	00	3F	00	FF	00	00	A0	0B	00	.....ø..?.ÿ.. ..
00600020	00	00	00	00	E0	1F	00	00	00	00	00	00	00	00	00	00	....à.....
00600030	01	00	06	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00600040	80	00	29	00	00	00	00	4E	4F	20	4E	41	4D	45	20	20	€.)....NO NAME
00600050	20	20	46	41	54	33	32	20	20	20	33	C9	8E	D1	BC	F4	FAT32 3ÉŽÑ¼ô
00600060	7B	8E	C1	8E	D9	BD	00	7C	A0	FB	7D	B4	7D	8B	F0	AC	{ŽÁŽÙ½.¦ û}´}‹ð¬
00600070	98	40	74	0C	48	74	0E	B4	0E	BB	07	00	CD	10	EB	EF	˜@t.Ht.´.».Í.ëï
00600080	A0	FD	7D	EB	E6	CD	16	CD	19	00	00	00	00	00	00	00	ý}ëæÍ.Í.....
00600090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
006000A0	3B	D6	67	49	29	2E	D8	4A	83	99	F6	A3	39	E3	D0	01	;ÖgI).ØJƒ™ö£9ãÐ.

[그림 1] 미할당 영역에서 발견한 Bitlocker Drive Encryption 시그니처

1.2. Bitlocker Drive Encryption (BDE) 구조와 볼륨 크기

BDE 는 고유한 구조를 가지며 대부분의 필드는 알려졌다[1]. 발견한 BDE 데이터가 연속적으로 할당되어 있다면, 구조를 따라서 암호화된 볼륨 크기를 알아낼 수 있으며 해당 크기만큼 읽어내면 BDE 데이터를 추출할 수 있다. 본 시나리오에서 BDE 는 연속적으로 할당되어 있어서 온전하게 추출할 수 있었다.

본 시나리오에서는 분석할 대상이 Windows 10 운영체제라는 것을 명시했으므로, Windows 7 이후 BDE 구조를 따라서 분석하였다. 암호화된 볼륨의 크기는 FVE Metadata block header 에 명시되어 있다. FVE Metadata block 은 총 3 개가 있으며, 각 header 로부터 암호화된 볼륨 크기를 확인할 수 있다.



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
02800000	2D	46	56	45	2D	46	53	2D	39	00	02	00	04	00	04	00	-FVE-FS-9.....
02800010	00	00	40	1F	00	00	00	00	00	00	00	00	10	00	00	00	..@..... .....
02800020	00	00	20	02	00	00	00	00	00	00	A0	05	00	00	00	00	.. ..
02800030	00	00	20	09	00	00	00	00	00	00	21	02	00	00	00	00	.. ..!.....

[그림 2] FVE Metadata block의 header에 명시된 볼륨

[그림 2]를 통해 볼륨의 크기는 0x1F400000 (500 MB)이며, 해당 크기만큼 읽어서 추출한 후 Bitlocker 파티션으로 인식되는지 확인하였다.

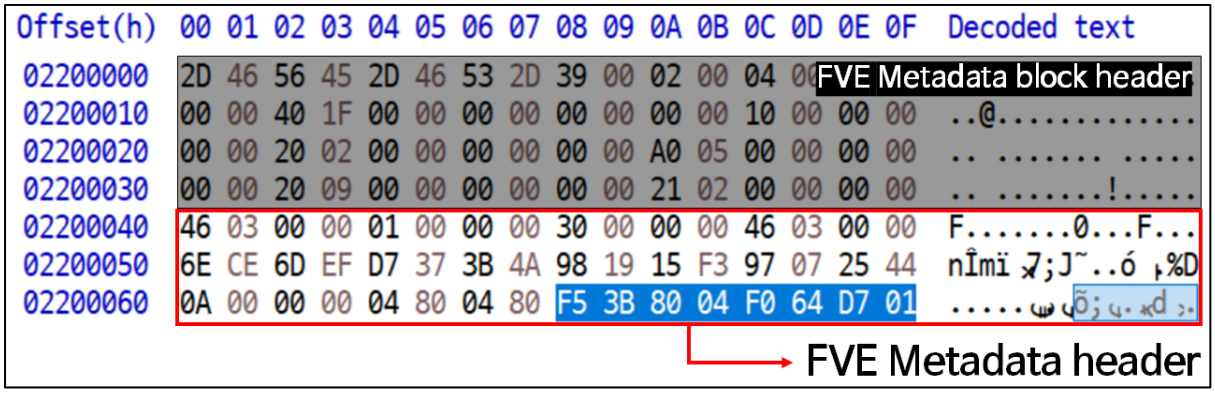
[그림 3] E 드라이브로 할당된 BDE 이미지

답: 0x1F400000 (500 MB)

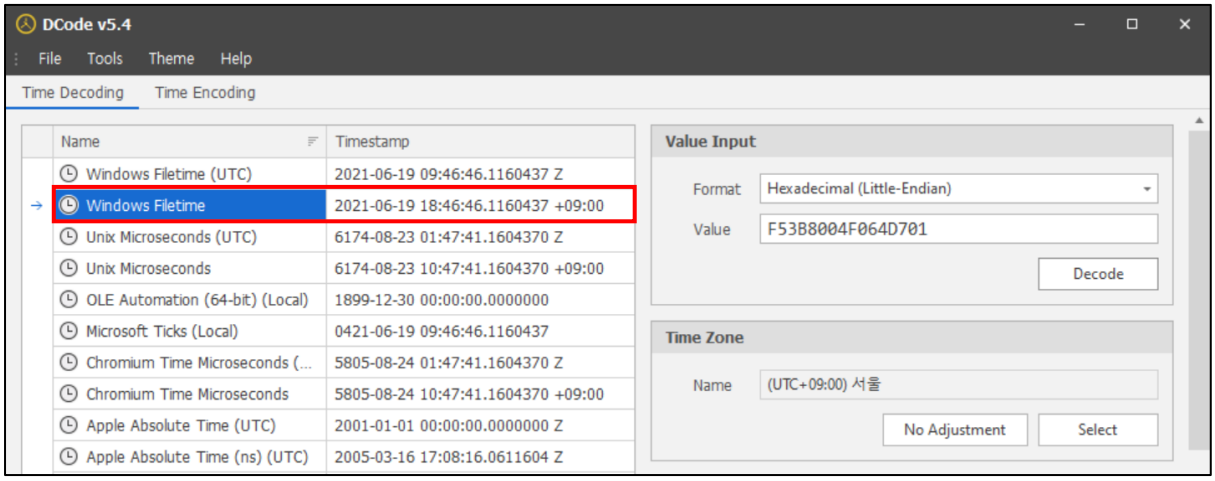
[1] BitLocker Drive Encryption (BDE) format specification, https://github.com/libyal/libbde/blob/main/documentation/BitLocker%20Drive%20Encryption%20(BDE)%20format.asciidoc

2. 볼륨이 암호화된 시각

볼륨이 암호화된 시각은 FVE Metadata block header 이후 나오는 FVE metadata header에서 알 수 있다.



[그림 4] 볼륨이 암호화된 시각 (Windows Filetime)



[그림 5] 볼륨이 암호화된 시각 해석

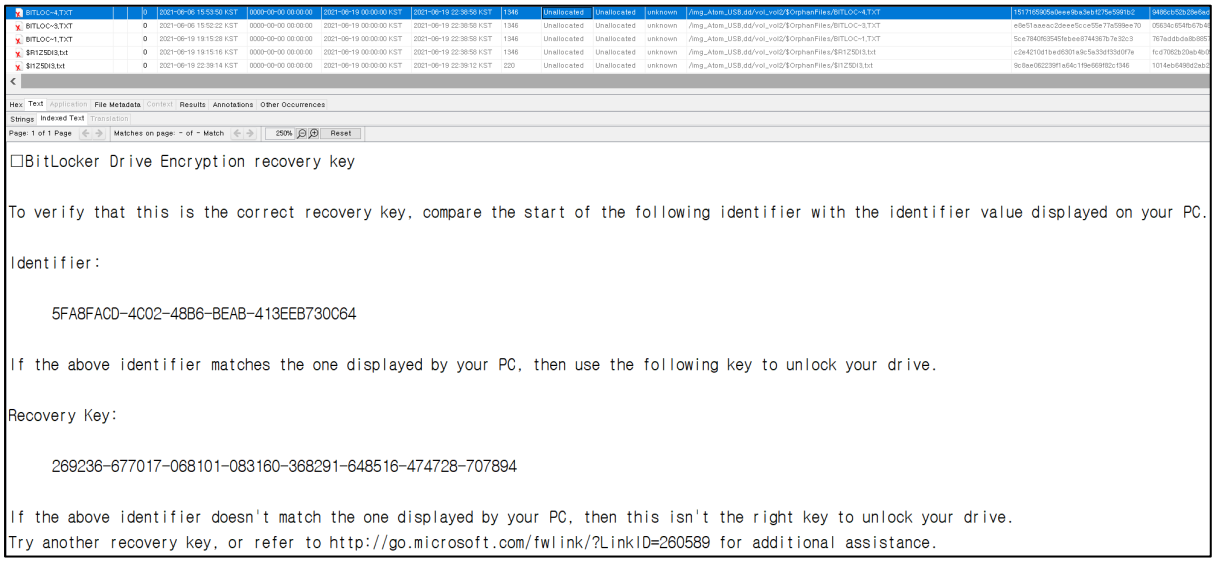
[그림 4]에서 확인한 볼륨이 암호화된 시각(0xF53B8004F064D701)을 [그림 5]와 같이 Dcode로 해석하였으며, 볼륨이 암호화된 시각은 2021-06-19 18:46:46.1160437 (UTC+9) 이다.

답: 2021-06-19 18:46:46.1160437 (UTC+9)

3. 기밀 파일의 SHA1 해시 값

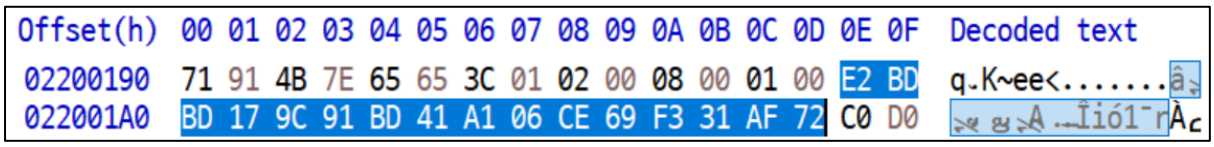
3.1. Bitlocker Recovery Key

BDE는 Recovery Key로도 복구할 수 있다. 주어진 USB 이미지에서 Bitlocker Recovery Key의 정보를 담은 파일들이 삭제된 흔적을 확인하였다.



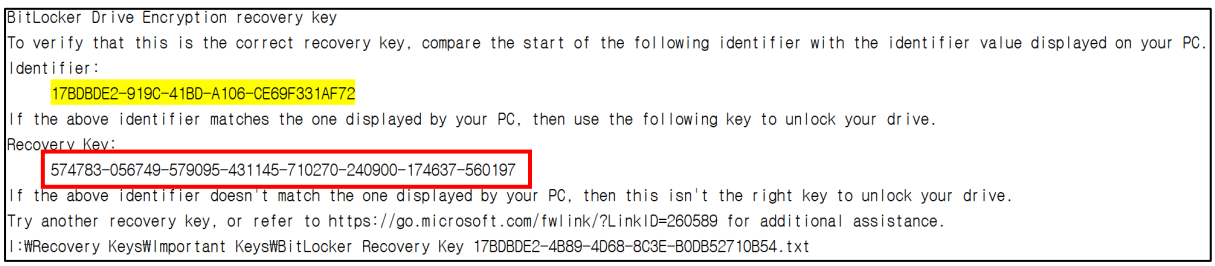
[그림 6] Bitlocker Recovery Key 파일의 삭제 흔적

Bitlocker Recovery Key 정보에는 Identifier와 Recovery Key가 있다. Identifier는 BDE의 FVE Volume Master Key에도 기록되어 있다. 복호화에 사용할 Recovery Key를 식별하기 위해서, 수집한 BDE 이미지의 Identifier를 확인하였다. 수집한 BDE 이미지의 Identifier는 {17BDBDE2-919C-41BD-A106-CE69F331AF72}이다.



[그림 7] 수집한 BDE 이미지의 Identifier

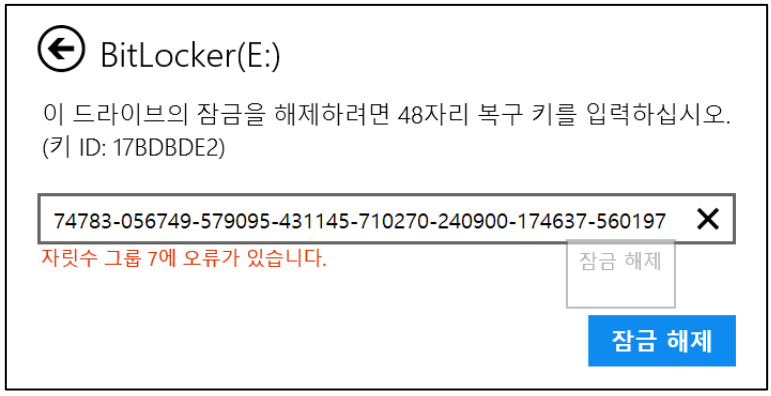
BDE 이미지의 Identifier에 맞는 Recovery Key를 식별하기 위해 USB 이미지에서 BDE 이미지의 Identifier를 문자열 탐색하였다.



[그림 8] 수집한 BDE 이미지에 대한 Recovery Key

3.2. Bitlocker 복호화

[그림 8]에서 확인한 Recovery 로 복호화를 시도하였지만 실패하였다. [그림 9]에서 "자릿수 그룹 7 에 오류가 있습니다." 문구를 통해서 숫자 6 자리 Brute Force 가 필요하다고 판단하였다.



[그림 9] 획득한 Recovery Key로 복호화 실패

[그림 10]과 같은 Brute Force 코드를 작성하여, [그림 11]과 같이 올바른 Recovery Key(574783-056749-579095-431145-710270-240900-174636-560197)를 얻었다.

```
$mount_drive_letter = "E:"

$recovery_key1 = "574783-056749-579095-431145-710270-240900-"
$recovery_key7 = ""
$recovery_key8 = "-560197"

for ($count = 0; $count -le 1000000; $count++)
{
    $recovery_key7 = $count.ToString()

    if ($recovery_key7.Length -lt 6)
    {
        $padd = "0" * (6 - $recovery_key7.Length)
        $recovery_key7 = $padd + $recovery_key7
    }
    $beauty_recovery_key = $recovery_key1 + $recovery_key7 + $recovery_key8
    $recovery_key = $beauty_recovery_key.replace("-", "")

    try
    {
        Unlock-BitLocker -MountPoint $mount_drive_letter -RecoveryPassword $recovery_key -
ErrorAction Stop
        Write-Output "[$count] Matched!!: $beauty_recovery_key"
        break
    }
    catch [System.Runtime.InteropServices.COMException]
    {
        Write-Output "[$count] Failed Recovery Key: $beauty_recovery_key"
    }
}
```

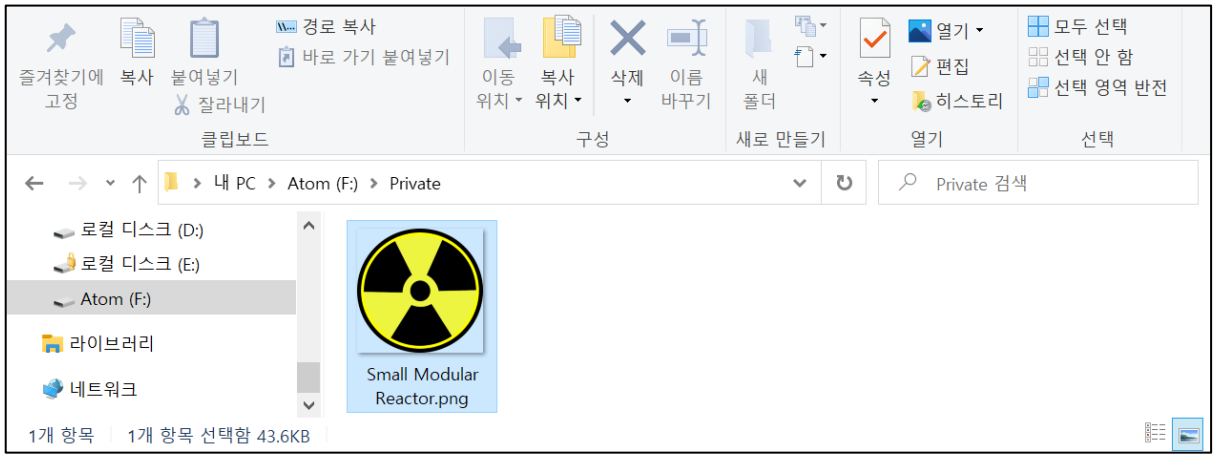
[그림 10] 올바른 Recovery Key 획득을 위한 Brute Force 코드

```
[174631] Failed Recovery Key: 574783-056749-579095-431145-710270-240900-174631-560197
[174632] Failed Recovery Key: 574783-056749-579095-431145-710270-240900-174632-560197
[174633] Failed Recovery Key: 574783-056749-579095-431145-710270-240900-174633-560197
[174634] Failed Recovery Key: 574783-056749-579095-431145-710270-240900-174634-560197
[174635] Failed Recovery Key: 574783-056749-579095-431145-710270-240900-174635-560197


   ComputerName: DESKTOP-SJP2L8M

VolumeType      Mount CapacityGB VolumeStatus           Encryption KeyProtector            AutoUnlock Protection
               Point                                    Percentage                         Enabled    Status
----------      ----- ---------- ------------           ---------- ------------            ---------- ----------
Data            E:          0.49 FullyEncrypted         100        {Password, RecoveryPas... False      On
[174636] Matched!! : 574783-056749-579095-431145-710270-240900-174636-560197
```

[그림 11] 올바른 Recovery Key 획득

3.3. 기밀 파일 확인

복호화 후, 볼륨 레이블이 “Atom”인 것을 확인하였으며, Private 폴더에 기밀 파일이 있었다.



[그림 12] 암호화된 볼륨에서 발견한 기밀 파일 (Small Modular Reactor.png)

해당 파일의 SHA1 해시 값을 계산하였다.

[그림 13] 기밀 파일의 SHA1 해시 값 계산

답: a095423c85ad951e210f6b82bbaa08a137c3d3de