Dreamhack find-the-spy

당신은 보안 담당자로서 내부 기밀을 유출한 혐의를 받고 있는 직원 A에 대해 내부 감사를 진행하고 있습니다. A의 PC에서 획득한 메모리 덤프로부터 A의 혐의를 입증하여 플래그를 찾으세요.

A의 회사 동료 B의 증언은 다음과 같습니다.

> "그날 따라 행동이 수상하고 무언가 불안해 보였어요. 특히 어떤 압축 파일을 유심히 보는 것처럼 보였는데... 자세한 내용은 잘 모르겠습니다"

Info

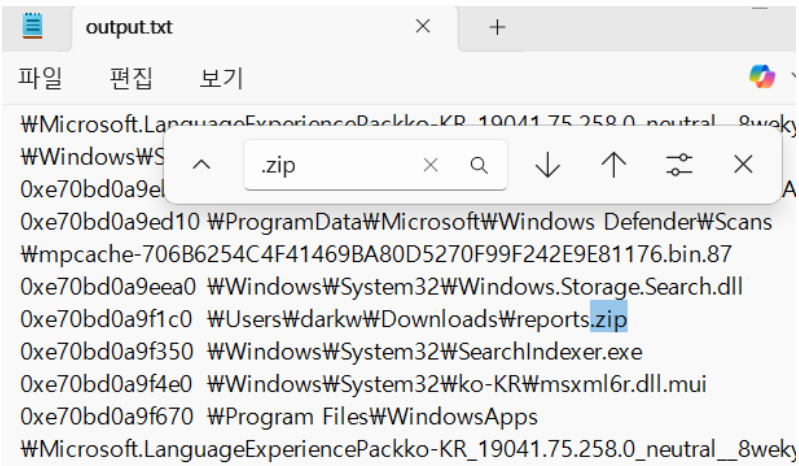
- FLAG = `DH{A_B}`
  - A: 시간 (`yyyyMMddhhmmss` 형식)
  - B: 장소
- 예를 들어 A가 `20240315180312` 이고, B가 `Seoul` 이라면, 플래그는 `DH{20240315180312_Seoul}` 입니다.

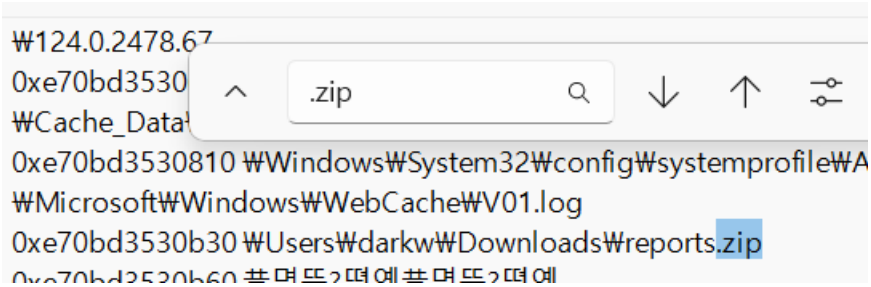
문제는 다음과 같다.

먼저 주어진 메모리 덤프에서 압축 파일 형태, 즉 zip을 찾아야 할 것 같다.

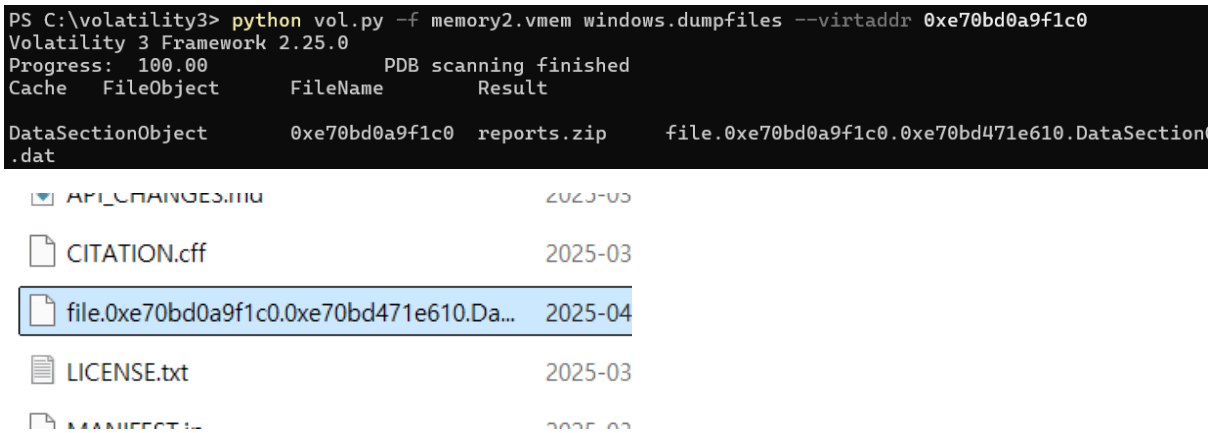
```
PS C:\volatility3> $env:PYTHONUTF8=1
PS C:\volatility3> python vol.py -f memory2.vmem windows.filescan.FileScan > output.txt
PS C:\volatility3>              PDB scanning finished
```

Volatility의 windows.filescan 플러그인을 이용하여 windows.filescan 플러그인의 결과를 텍스트 파일로 저장한다. 이렇게 하면 더 쉽게 zip 파일을 찾아낼 수 있다.





zip으로 찾아보면 두 개의 주소가 존재한다. 가상 메모리 주소를 얻었으므로 각각 값을 덤프해서 데이터를 살펴보려고 했다.



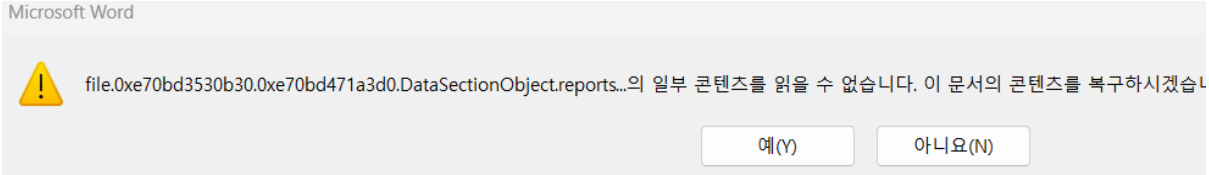
Dat 파일이 생성되었다.

이 파일을 열어보면 다음과 같다. Gmail의 첨부 파일인 것 같은데 딱히 유의미한 결과를 찾기 어려워 일단 두 번째 주소의 값을 먼저 보기로 했다.

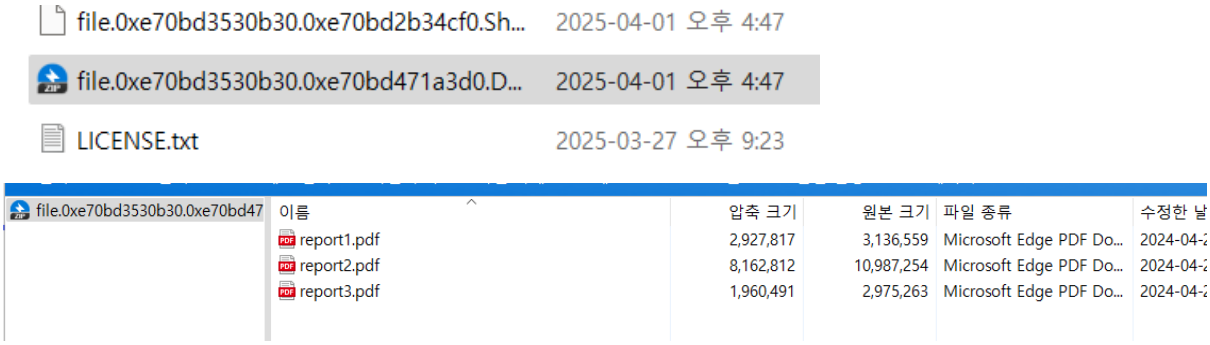
```
PS C:\volatility3> python vol.py -f memory2.vmem windows.dumpfiles --virtaddr 0xe70bd3530b30
Volatility 3 Framework 2.25.0
Progress:  100.00               PDB scanning finished
Cache   FileObject      FileName        Result

DataSectionObject       0xe70bd3530b30  reports.zip     file.0xe70bd3530b30.0xe70bd471a3d0.DataSectionObjec
.dat
SharedCacheMap  0xe70bd3530b30  reports.zip     file.0xe70bd3530b30.0xe70bd2b34cf0.SharedCacheMap.reports.z
```

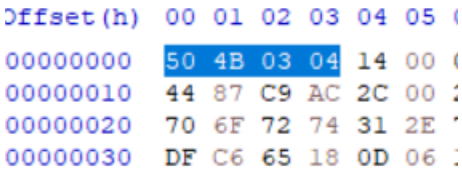
이제 두 번째 주소의 값을 덤프 한다. 첫번째 주소에서 했던 것과 동일한 방법으로 파일을 생성해준다.



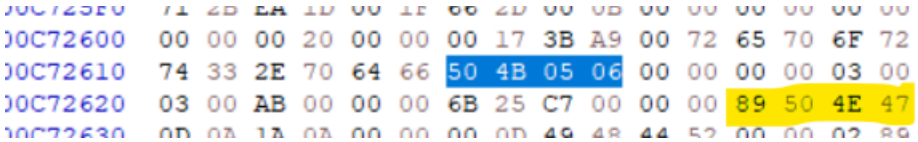
그런데 이렇게 에러가 생긴다. 확장자를 zip으로 바꿔주고 다시 열어보려고 했다.



안에는 이렇게 pdf파일들이 들어있다. 그런데 유의미한 결과는 없는 거 같아서 zip파일을 헥스 에디터에서 열어보았다.



헤더 시그니처는 정상적으로 들어가 있다. 그 뒤에는 푸터 시그니처를 찾아보았다.



푸터 시그니처도 확인할 수 있었다. 그런데 그 줄 좀 아래에 png 파일의 헤더 시그니처가 보였다. 즉 이 문제는 zip파일 뒷부분에 png 파일을 숨겨둔 문제라 할 수 있겠다.

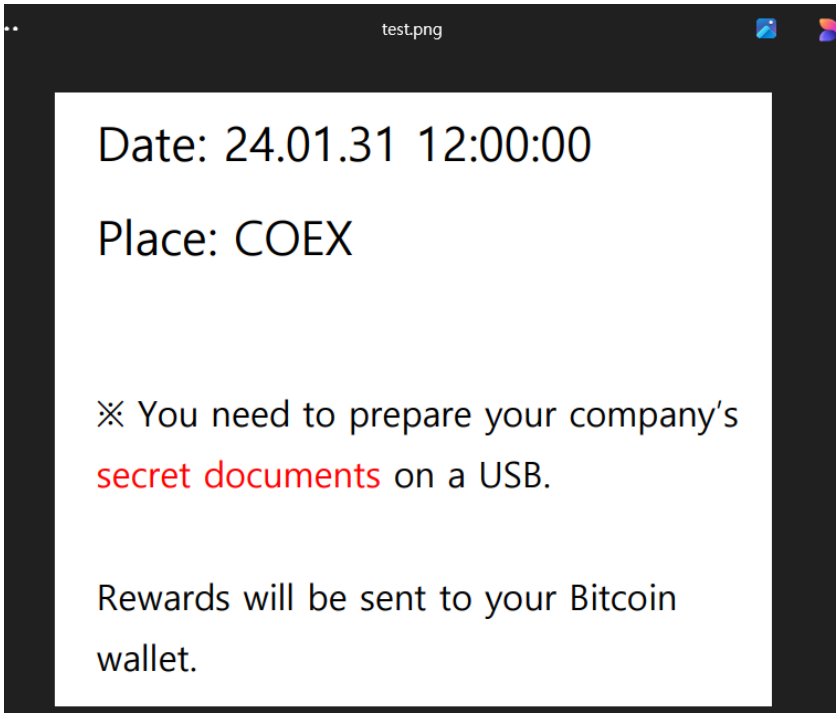
Png 파일의 헤더 시그니처부터 파일 끝까지 복사해서 새로운 파일에 붙여넣는다.

```
E0  40 36 03 00 00 00 B2 19 00 00 00 D9 0C 00 00 00   @6....
F0  C8 66 00 00 00 64 33 00 00 00 20 9B 01 00 00 90   Èf...d
00  CD 00 00 00 80 6C 06 00 00 40 36 03 00 00 C0 33   Í...€l
10  E0 FF 01 10 BE EA 6B 68 C7 34 A1 00 00 00 00 49   àÿ..¾êhÇ4¡
20  45 4E 44 AE 42 60 82 00 00 00 00 00 00 00 00 00   END®B`
30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ......
40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ......
50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ......
```

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Deco

00000000   89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52  %PNG
00000010   00 00 02 89 00 00 02 2B 08 02 00 00 00 0C 61 94  ...%
00000020   93 00 00 00 09 70 48 59 73 00 00 0E C4 00 00 0E  "...
00000030   C4 01 95 2B 0E 1B 00 00 20 00 49 44 41 54 78 DA  Ä.•+
00000040   EC DD 77 5C 13 E7 1F 07 F0 EF 73 97 B0 F7 90 25  ìÝw\
00000050   4E C4 3D 2A EE BD F7 C2 59 EB 16 6B 1D 3F B7 75  NÄ=*
00000060   E1 6A 9D 6D D5 3A EA AC 7B 6B AD 7B B4 CE BA 07  áj.m
```

확장자를 png로 바꿔주고 열면 이렇게 시간과 장소 값을 얻을 수 있다.



따라서 플래그는 **DH{20240131120000_COEX}** 이다.