structure-based carving

Description

> 주어진 바이너리 파일에서 플래그를 찾아보세요!
> 힌트는 압축 패스워드는 ZIP 구조 어딘가에... 입니다.

사용한 도구

HxD

Background

zip 파일 구조

- 크게 3개의 파일 구조 (**Local File Header, Central Directory, End of Central directory record**)로 이루어짐
- **Local File Header**
  - ZIP 파일 구조에서 압축된 각 파일에 대한 정보를 담는 부분
  - 압축 전후 파일 크기, 파일 수정 시간, CRC-32 체크섬, 파일이름 지역 포인터, 압축 해제시 필요한 아카이브 버전 등 포함
- **File Name**
  - 압축된 파일 이름 형식에 대한 임의의 길이와 바이트 순서 표시
  - 파일 이름의 길이는 65536 문자 초과 불가
- **File Data**
  - 임의의 길이로 구성된 바이트 배열 형태로 압축된 파일 컨텐츠
  - 파일이 비거나 디렉토리 포함시 이 배열 사용 X
  - 하지만 그 다음 Local File Header 제목은 해당 파일이나 디렉토리 설명
- **Central Directory**
  - Local File Header의 확장 데이터 뷰 제공

    - Local File Header의 포함된 데이터를 더하여 파일 속성, 구조에 대한 로컬 기준 가짐
- **End of central directory record**
    - 모든 아카이브의 싱글 템플릿으로 제공하며 아카이브의 종료 작성
    - 포함된 데이터에서 가장 중요한 데이터는 Central Directory 블록의 시작과 로컬 참조의 시작, 아카이브 레코드들의 숫자임

zip 파일 동작 원리

1. ZIP파일을 실행 하면 먼저 End of Central Directory 로 들어감
2. 그리고 End of Central Directory 에 존재 하는 값인 Central Header Offset 에 존재 하는 Offset으로 이동
3. Central Header Offset에 들어있는 값을 따라 이동을 해보면 Central Directory 시작 Offset으로 감
4. Central Directory의 맨위 부터 아래로 내려오면서 내부에 존재하는 Central Header의 개수를 구별하면서 Central Directory에 존재하는 Local Header Offset에 들어있는 값을 확인
5. 확인한 Local Header Offset의 값을 바탕으로 Local Header 시작 Offset으로 감
6. Local Header Offset 으로 간 뒤 해당 File Data로 이동함으로서 zip파일 사용자가 zip파일 내부 압축 데이터를 확인

Local File Header 구조

https://velog.io/@sooboon/ZIP-%ED%8C%8C%EC%9D%BC-%EA%B5%AC%EC%A1%B0

필드	크기	설명
Signature	4 바이트	시그니처 (0x04034b50, 리틀엔 - 50 4B 03 04)
Version	2 바이트	압축 해제 시 필요한 버전
Flags	2 바이트	바이트 식별자
		비트 00: 암호화된 파일
		비트 01: 압축 옵션
		비트 02: 압축 옵션
		비트 03: 데이터 설명자

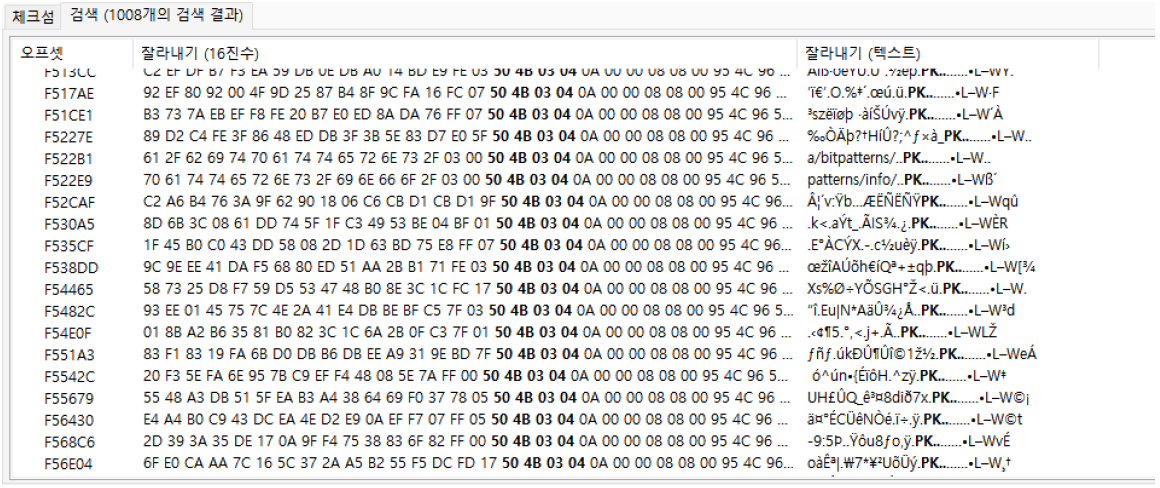
필드	크기	설명
		비트 04: 향상된 디플레이션
		비트 05: 패치된 압축 데이터
		비트 06: 강력한 암호화
		비트 07-10: 사용되지 않음
		비트 11: 언어 인코딩
		비트 12: 예약됨
		비트 13: 마스크 헤더 값
		비트 14-15: 예약됨
Compression method	2 바이트	압축 유형 선택 (보통 0x08 Deflated 사용)
		00: 압축 없음
		01: 수축
		02: 압축 계수 1로 감소
		03: 압축 계수 2로 감소
		04: 압축 계수 3으로 감소
		05: 압축 계수 4로 감소
		06: 내파됨
		07: 예약 됨
		08: 수축됨
		09: 향상 수축됨
		10: PKWare DCL 내파됨
		11: 예약 됨
		12: BZIP2를 사용하여 압축됨
		13: 예약됨
		14: LZMA
		15-17: 예약됨
		18: IBM TERSE를 사용하여 압축됨
		19: IBM LZ77 z
		98: PPMd 버전 I, Rev 1

필드	크기	설명
File modification time	2 바이트	마지막 파일 수정 시간
		비트 00-04: 초를 2로 나눈 값
		비트 05-10: 분
		비트 11-15: 시간
File modification date	2 바이트	마지막 파일 수정 날짜
		비트 00-04: 일
		비트 05-08: 월
		비트 09-15: 1980년부터 연도
Crc-32 checksum	4 바이트	파일 내용의 오류 체크
		이 필드가 작성되지 않을 경우 압축 프로그램은 손상된 파일로 간주 압축해제 거부
Compressed size	4 바이트	압축된 데이터의 바이트 크기
Uncompressed size	4 바이트	원본 데이터의 바이트 크기
File name length	2 바이트	파일 이름의 길이
Extra field length	2 바이트	추가 예약 필드로 사용 X
File name	가변 크기	상대 경로를 포함하는 파일의 이름
Extra field	가변 크기	추가 정보 저장하는데 사용
		이 필드는 헤더와 데이터 쌍의 순서로 구성
		여기서 헤더는 2 바이트 식별자와 2 바이트 데이터로 사이즈 필드를 가짐

1. HxD로 binary 파일 분석

문제를 다운받으면 carving_target.bin 이라는 바이너리 파일이 존재한다.

Zip 구조 어딘가에 패스워드가 존재하니 zip의 시그니처인 50 4B 03 04를 찾아봤다.



1008개의 검색 결과가 나왔다 → 검색 범위를 좁힐 필요성 존재 → zip 파일 구조를 알아야 함

zip 시그니처 뒤에 이어지는 숫자를 알기 위해선 압축 생성 버전과 Flags, Compression method 정도를 알아야한다.

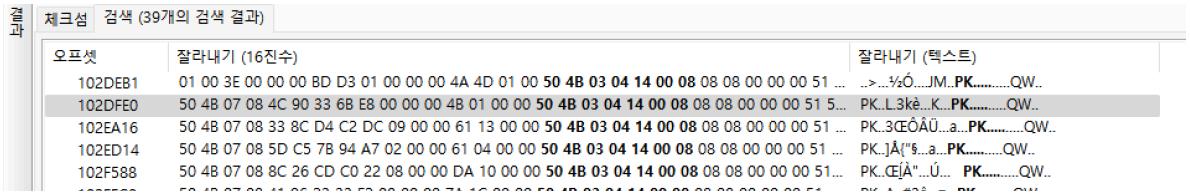
압축 생성 버전 : 14 (일반적으로 쓰이는 버전)

Flags : 00 (암호화 된 파일)

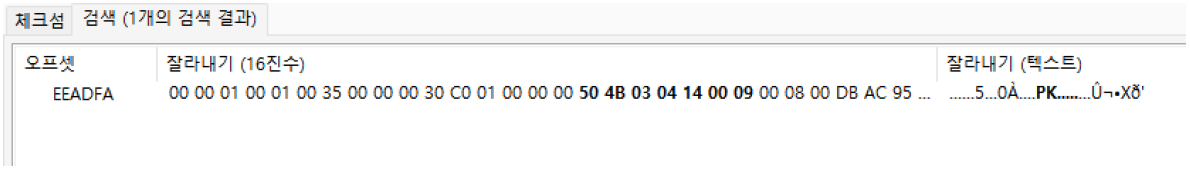
압축 유형 : 08 or 09 (일반적으로 쓰이는 유형)

2. 50 4B 03 04 14 00 08 / 50 4B 03 04 14 00 09 검색

50 4B 03 04 14 00 08 검색 - 39개의 결과



50 4B 03 04 14 00 09 검색 - 1개의 결과 → 20240421로 시작하는 png 파일 존재 → 추출

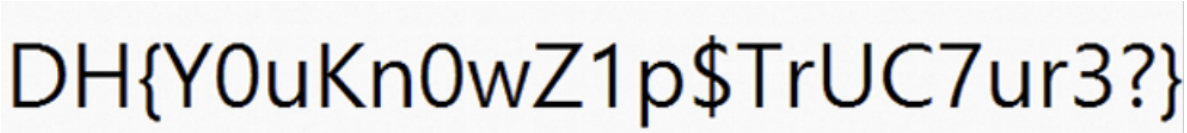


아래로 내리다보면 password 비밀번호를 발견할 수 있다.

```
00EEBCD0  01 02 14 00 14 00 09 00 08 00 DB AC 95 58 F0 27  ..........Û¬•Xð'
00EEBCE0  25 E5 A3 0E 00 00 DB 0E 00 00 13 00 24 00 00 00  %å£...Û.....$...
00EEBCF0  00 00 00 00 20 00 00 00 00 00 00 00 00 32 30 32 34  .... ........2024
00EEBD00  30 34 32 31 5F 32 31 33 38 30 32 2E 70 6E 67 7A  0421_213802.pngz
00EEBD10  31 70 5F 70 34 73 35 77 30 33 64 5F 31 73 5F 61  1p_p4s5w03d_1s_a
00EEBD20  31 62 32 63 33 64 34 65 35 66 36 00 00 00 00 00  1b2c3d4e5f6.....
00EEBD30  00 00 00 50 4B 05 06 00 00 00 00 01 00 01 00 65  ...PK..........e
00EEBD40  00 00 00 D4 0E 00 00 00 00 50 4B 03 04 14 00 00  ...Ô.....PK.....
00EEBD50  00 08 00 00 00 21 50 CA 12 DF 97 AA B6 01 00 77  .....!PÊ.ß—ª¶..w
00EEBD60  28 06 00 07 00 00 00 32 32 30 2E 73 64 62 BC 9D  (.......220.sdb¼.
```

3. 50 4B 03 04 14 00 09로 시작하는 부분 추출

추출 후 a1b2c3d4e5f6을 입력하여 압축을 해제하면 flag 이미지 파일이 나온다.



FLAG

DH{Y0uKn0wZ1p$TrUC7ur3?}