[pwnable.kr WriteUp]

이태규(ltg2030)

[Toddler's Bottle] flag 7PT

문제 설명
Papa brought me a packed present! let's open it. Download : http://pwnable.kr/bin/flag This is reversing task. all you need is binary

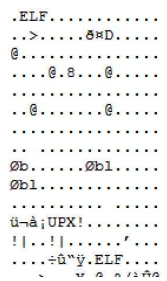
Packing : 파일의 크기를 줄이고, 내부 코드와 리소스, 엔트리 포인트등을 숨기기위한 작업. 실행 압축 방식으로 파일 내부에 압축해체 코드를 포함하고 있어 실행과 동시에 메모리에서 압축을 해제하고 바로 실행된다.

gdb : linux에서 기본적으로 제공되는 debugging tool
- 디스어셈블 : disassemble 주소/이름
- BreakPoint : b *주소/이름
- 실행 : run
- 메모리 or 레지스터 보기 : x/[x, i, wx, bx] 주소/이름/레지스터

등의 다양한 명령어로 debugging을 할 수 있다.

문제 풀이(이론)

1. 우선 Hex Editor로 flag파일을 열어보면



이런 글이 나오는데 결국 이 파일은 ELF 구조이고 UPX 패킹 되어있음을 알 수 있다.

2. 따라서 upx-391w 프로그램을 통해 flag를 언패킹하고 debugging 해야한다.

3. 그러나 flag를 언패킹한 결과 64bit 실행 파일로 쉽게 debugging 할 수 없어 64 bit linux 환경에서 gdb를 통해 debugging 하였다.

4.

```
(gdb) disass main
Dump of assembler code for function main:
   0x0000000000401164 <+0>:    push   %rbp
   0x0000000000401165 <+1>:    mov    %rsp,%rbp
   0x0000000000401168 <+4>:    sub    $0x10,%rsp
   0x000000000040116c <+8>:    mov    $0x496658,%edi
   0x0000000000401171 <+13>:   callq  0x402080 <puts>
   0x0000000000401176 <+18>:   mov    $0x64,%edi
   0x000000000040117b <+23>:   callq  0x4099d0 <malloc>
   0x0000000000401180 <+28>:   mov    %rax,-0x8(%rbp)
   0x0000000000401184 <+32>:   mov    0x2c0ee5(%rip),%rdx        # 0x6c2070
<flag>
   0x000000000040118b <+39>:   mov    -0x8(%rbp),%rax
   0x000000000040118f <+43>:   mov    %rdx,%rsi
   0x0000000000401192 <+46>:   mov    %rax,%rdi
   0x0000000000401195 <+49>:   callq  0x400320
   0x000000000040119a <+54>:   mov    $0x0,%eax
   0x000000000040119f <+59>:   leaveq
   0x00000000004011a0 <+60>:   retq
End of assembler dump.
```

gdb로 flag를 disassemble 하면 위의 결과가 나온다. flag를 실행시키면 I will malloc() and strcpy the flag there. take it. 라는 힌트를 준다. callq 0x4099d0이 malloc이므로 callq 0x400320 이 부분이 strcpy 일 것이라고 예상할 수 있다. 따라서 다음 구문의 주소인 0x40119a에 break point를 두고 [rdi], [rdi+1], [rdi+2], ... 의 값을 보며 0이 나올 때 까지가 flag 내용임을 알 수 있다.

결과가 좀 길지만 써보자면

```
(gdb) x/bx $rdi
0x6c96b0:       0x55
(gdb)
0x6c96b1:       0x50
(gdb)
0x6c96b2:       0x58
(gdb)
0x6c96b3:       0x2e
(gdb)
0x6c96b4:       0x2e
```

```
(gdb)
0x6c96b5:       0x2e
(gdb)
0x6c96b6:       0x3f
(gdb)
0x6c96b7:       0x20
(gdb)
0x6c96b8:       0x73
(gdb)
0x6c96b9:       0x6f
(gdb)
0x6c96ba:       0x75
(gdb)
0x6c96bb:       0x6e
(gdb)
0x6c96bc:       0x64
(gdb)
0x6c96bd:       0x73
(gdb)
0x6c96be:       0x20
(gdb)
0x6c96bf:       0x6c
(gdb)
0x6c96c0:       0x69
(gdb)
0x6c96c1:       0x6b
(gdb)
0x6c96c2:       0x65
(gdb)
0x6c96c3:       0x20
(gdb)
0x6c96c4:       0x61
(gdb)
0x6c96c5:       0x20
(gdb)
0x6c96c6:       0x64
(gdb)
0x6c96c7:       0x65
(gdb)
0x6c96c8:       0x6c
```

```
(gdb)
0x6c96c9:       0x69
(gdb)
0x6c96ca:       0x76
(gdb)
0x6c96cb:       0x65
(gdb)
0x6c96cc:       0x72
(gdb)
0x6c96cd:       0x79
(gdb)
0x6c96ce:       0x20
(gdb)
0x6c96cf:       0x73
(gdb)
0x6c96d0:       0x65
(gdb)
0x6c96d1:       0x72
(gdb)
0x6c96d2:       0x76
(gdb)
0x6c96d3:       0x69
(gdb)
0x6c96d4:       0x63
(gdb)
0x6c96d5:       0x65
(gdb)
0x6c96d6:       0x20
(gdb)
0x6c96d7:       0x3a
(gdb)
0x6c96d8:       0x29
(gdb)
0x6c96d9:       0x00
(gdb)
```

이렇게 나오는데 이것을 하나하나 분석할 순 없으므로 여기서 0x(XX) 부분만 뽑아내어 글자로 바꾸어서 출력해주는 Python Script를 만들었다.

문제 풀이(실습)

위의 값을 flag.txt에 넣어준 뒤

```
f=open("flag.txt",'r')
while True:
    line = f.readline()
    line = f.readline()
    if not line:
        break
    x=ord(line[12]);
    y=ord(line[13]);
    if x>=ord('0') and x<=ord('9'):
        x-=ord('0');
    else:
        x-=ord('a')-10;
    if y>=ord('0') and y<=ord('9'):
        y-=ord('0');
    else:
        y-=ord('a')-10;
    z=chr(x*16+y);
    print(z,end="");
```

라는 Python Script를 실행하면

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:44:40) [MSC v.1600 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
UPX...? sounds like a delivery service :)
>>>
```

UPX...? sounds like a delivery service :) 라는 flag값을 얻을 수 있다.

실행화면