give_me_password

문제 분류	rev
poc 작성자	김재환
문제 개발자	김재환

1. 문제

`give_me_flag` 파일은 리눅스 실행 파일인 ELF 파일로, x86-64 환경에서 컴파일 된 것을 알 수 있습니다.

```
┌──(jalnik㉿jalnik)-[/mnt/c/Users/silmu/Desktop/자료/프로젝트/우석대학교 CTF/문제/rev/give_me_password(rev)/deploy]
└─$ file give_me_flag
give_me_flag: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, BuildID[sha1]=f00c71ac999427affce9d26cbec0d662a94fac60, for GNU/Linux 3.2.0, not stripped
```

문제를 실행하면 1번과 2번을 선택할 수 있고, 1번을 누르니 어떤 비밀번호가 존재하고 `약한 강도로 암호화` 되어 있다고 합니다.

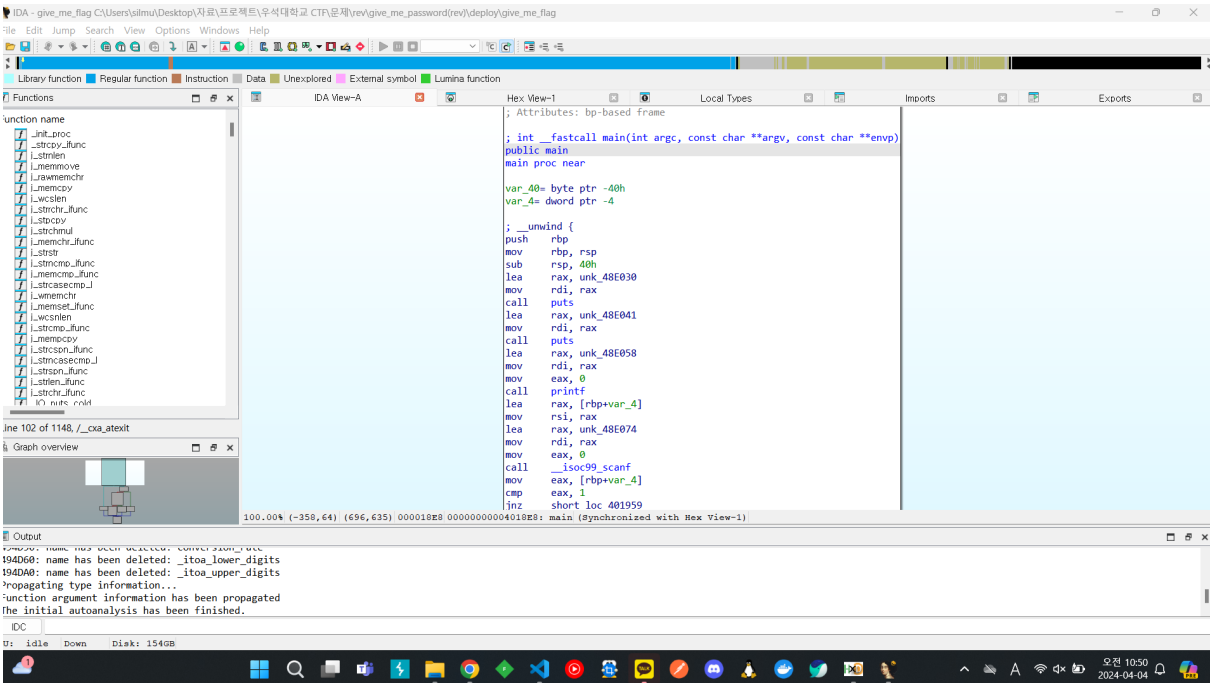
```
┌──(jalnik㉿jalnik)-[/mnt/c/Users/silmu/Desktop/자료/프로젝트/우석대학교 CTF/문제/rev/give_me_password(rev)/deploy]
└─$ ./give_me_flag
1. 문제 소개
2. 패스워드 입력
번호를 입력하세요: 1
SOTI만 공유하고 있는 비밀번호가 있다는 정보를 입수했다.
해당 정보는 동아리원들이 게으른 관계로 약한 강도로 암호화 되있다고 하는데... 비밀번호를 알아낼 수 있을까?

┌──(jalnik㉿jalnik)-[/mnt/c/Users/silmu/Desktop/자료/프로젝트/우석대학교 CTF/문제/rev/give_me_password(rev)/deploy]
└─$ ./give_me_flag
1. 문제 소개
2. 패스워드 입력
번호를 입력하세요: 2
패스워드를 입력하세요: 1123
ㅋ
```

- 혹시나 해서 아무 문자열이나 넣었더니 비웃습니다.

2. 풀이

비웃는 저 녀석을 혼내줘야 합니다. IDA라는 디컴파일러로 해당 파일을 분석해 보겠습니다.

시작하면 `ida`가 `main`함수를 자동으로 찾아줍니다.

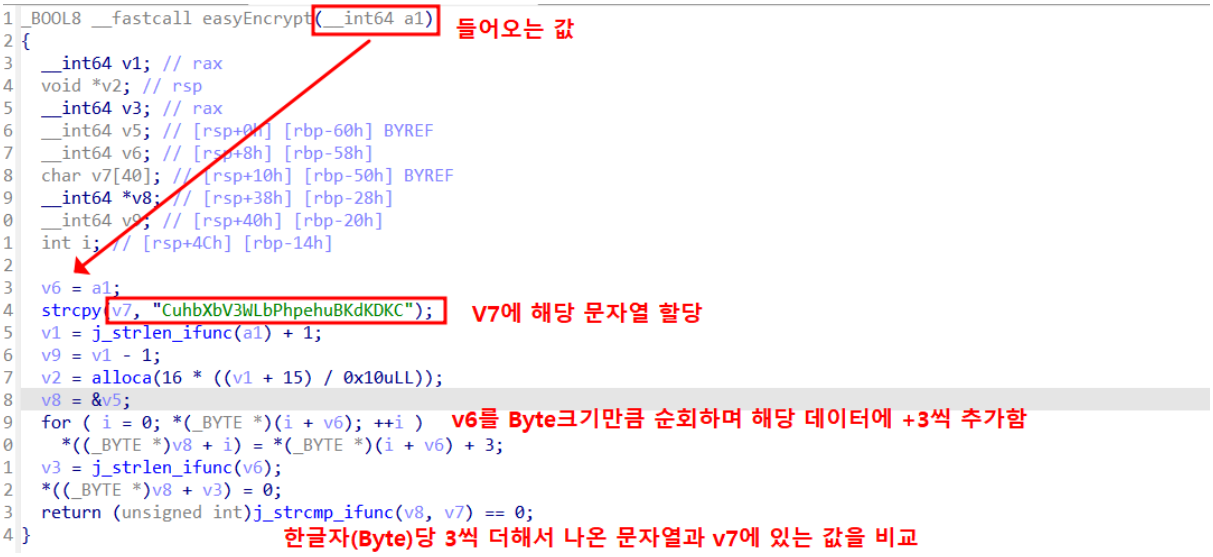


`main`함수 부분에서 `F5`키를 눌러 디컴파일을 진행합니다.

```
int v16; // edx
int v17; // ecx
int v18; // r8d
int v19; // r9d
__int64 v20; // rdx
int v21; // edx
int v22; // ecx
int v23; // r8d
int v24; // r9d
char v26[60]; // [rsp+0h] [rbp-40h] BYREF
int v27; // [rsp+3Ch] [rbp-4h] BYREF

puts(&unk_48E030, argv, envp);
puts(&unk_48E041, argv, v3);
printf((unsigned int)&unk_48E058, (_DWORD)argv, v4, v5, v6, v7, v26[0]);
_isoc99_scanf((unsigned int)&unk_48E074, (unsigned int)&v27, v8, v9, v10, v11, v26[0]);
if ( v27 == 1 )
{
  puts(&unk_48E078, &v27, v12);
}
else if ( v27 == 2 )
{
  printf((unsigned int)&unk_48E160, (unsigned int)&v27, v12, v13, v14, v15, v26[0]);
  _isoc99_scanf((unsigned int)"%50s", (unsigned int)v26, v16, v17, v18, v19, v26[0]);
  if ( (unsigned __int8)easyEncrypt(v26) )
  {
    puts(&unk_48E188, v26, v20);
    printf((unsigned int)&unk_48E1AE, (unsigned int)v26, v21, v22, v23, v24, v26[0]);
  }
  else
  {
    puts(&unk_48E1C2, v26, v20);
00001997 main:40 (401997)
```

- Main함수의 Scanf 함수 부분을 보면, 사용자의 입력값을 배열로 받아 해당 배열의 시작 주소를 넘겨줍니다.
- 넘겨주는 함수의 이름은 `easyEncrypt` 입니다.

무언가 암호화 하는 함수로 추정되는 `easyEncrypt` 를 확인해 보겠습니다.




```
_BOOL8 __fastcall easyEncrypt(__int64 a1)
{
  __int64 v1; // rax
  void *v2; // rsp
  __int64 v3; // rax
  __int64 v5; // [rsp+0h] [rbp-60h] BYREF
  __int64 v6; // [rsp+8h] [rbp-58h]
  char v7[40]; // [rsp+10h] [rbp-50h] BYREF
  __int64 *v8; // [rsp+38h] [rbp-28h]
  __int64 v9; // [rsp+40h] [rbp-20h]
  int i; // [rsp+4Ch] [rbp-14h]

  v6 = a1;
  strcpy(v7, "CuhbXbV3WLbPhpehuBKdKDKC");
  v1 = j_strlen_ifunc(a1) + 1;
  v9 = v1 - 1;
  v2 = alloca(16 * ((v1 + 15) / 0x10uLL));
  v8 = &v5;
  for ( i = 0; *(_BYTE *)(i + v6); ++i )
    *((_BYTE *)v8 + i) = *(_BYTE *)(i + v6) + 3;
  v3 = j_strlen_ifunc(v6);
  *((_BYTE *)v8 + v3) = 0;
  return (unsigned int)j_strcmp_ifunc(v8, v7) == 0;
}
```

- 이상한 문자열을 v7에 저장하고, 사용자의 입력값에 한글자(Byte)마다 +3을 해주고 비교합니다.
- 이때 사용자의 입력값에 +3한 값이 이상한 문자열과 같다면 True를 반환해 줍니다.

그렇다면 이상한 문자열을 1Byte씩 잘라서 3씩 빼주면 정답을 알 수 있지 않을까요? 이는 마치 `시저암호` 같습니다.

간단한 Python코드로 디코딩 하는 작업을 수행해 보겠습니다.

```python
encrypt = "CuhbXbV3WLbPhpehuBKdKDKC"

def decode_string(input_str):
    decode_str = ""
    for char in input_str:
        decode_str += chr(ord(char) - 3)

    return decode_str


if __name__=="__main__":
    flag = decode_string(encrypt)

    print(flag)
```

- 이상한 문자열을 한글자씩 가져와서 -3씩 빼줍니다.
- 해당 코드를 돌리면 정답을 얻을 수 있습니다.

이제 정답을 문제에 입력해 보겠습니다.

```
┌──(jalnik㉿jalnik)-[/mnt/c/Users/silmu/Desktop/자료/프로젝트/우석대학교 CTF/문제/rev/give_me_password(rev)/deploy]
└─$ ./give_me_flag
1. 문제 소개
2. 패스워드 입력
번호를 입력하세요: 2
패스워드를 입력하세요: @re_U_S0TI_Member?HaHAH@
```

정답입니다!

```
패스워드를 입력하세요: @re_U_S0TI_Member?HaHAH@
정답입니다! 너무 쉬웠나요?
정답은! SOTI{@re_U_S0TI_Member?HaHAH@}
```