Jalnik(rev)

1. 문제

`Jalnik` 문제는 간단한 복호화 리버싱 문제입니다.

flag_en	2024-05-20 오후 8:16	PNG 파일	4KB
prob	2024-05-20 오후 8:16	응용 프로그램	103KB
sample	2024-05-20 오후 8:16	PNG 파일	11KB
sample_en	2024-05-20 오후 8:22	PNG 파일	11KB
solve	2024-05-20 오후 9:33	Python 원본 파일	1KB

파일이 4개가 주어졌습니다. `sample.png`, `sample_en.png` 는 암호화를 시켜놓은 샘플 같네요.

우리가 해독해야 하는 파일은 `flag_en.png` 파일입니다

```
HAHA!!! Jalnik, your file is Encryted!!!
```

`prob.exe`를 실행하면 `HAHA!!! Jalnik, your file is Encryted!!!` 라는 문자열을 출력하며 엔터를 누르면 파일을 암호화합니다.

이제 `IDA`로 열어서 분석을 해보겠습니다.

2. 풀이

> IDA로 처음 바이너리 파일을 연 화면 입니다.

```
; int __cdecl main(int argc, const char **argv, const char **envp)
public _main
_main proc near

argc= dword ptr  8
argv= dword ptr  0Ch
envp= dword ptr  10h

push    ebp
mov     ebp, esp
and     esp, 0FFFFFFF0h
sub     esp, 10h
call    ___main
mov     dword ptr [esp], offset aHahaJalnikYour ; "HAHA!!! Jalnik, your file is Encryted!!"...
call    _puts
call    __Z12FileEncodingv ; FileEncoding(void)
mov     dword ptr [esp], offset Format ; "Perfect!"
call    _printf
mov     eax, 0
leave
retn
_main endp
```

- 어셈블리어로 되어있으니 보기가 어렵습니다..
- IDA에서는 디컴파일이라는 기능을 지원하고 있습니다! `F5` 키를 눌러 디컴파일을 해보겠습니다.

> 💡디컴파일이란? 컴파일과는 반대로, 컴파일된 실행 파일을 소스코드로 되돌리는 작업이다.

> 디컴파일 된 화면 입니다.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __main();
  puts("HAHA!!! Jalnik, your file is Encryted!!!");
  FileEncoding();
  printf("Perfect!");
  return 0;
}
```

- 어셈블리어를 C언어 Level에서 보여줍니다. 훨씬 보기가 편하네요. 😊
- 코드를 분석 해보겠습니다.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __main();
  puts("HAHA!!! Jalnik, your file is Encryted!!!"); // 화면에 문자열을 출력 합니다.
  FileEncoding();                                   // FileEncoding()이라는 이름의 함수를 실행합니다.
  printf("Perfect!");                               // Perfect! 라는 문자열을 출력합니다.
  return 0;
}
```

- 아무래도 우리는 `FileEncoding()`을 해석해봐야 할 것 같습니다.
- FileEncoding() 함수를 더블클릭해서 들어가줍니다.

> FileEncoding()

```
int FileEncoding(void)
{
  char Str[31]; // [esp+15h] [ebp-43h] BYREF
  int Character; // [esp+34h] [ebp-24h]
  FILE *Stream; // [esp+38h] [ebp-20h]
  char *v4; // [esp+3Ch] [ebp-1Ch]
  char *FileName; // [esp+40h] [ebp-18h]
  int v6; // [esp+44h] [ebp-14h]
  FILE *v7; // [esp+48h] [ebp-10h]
  int i; // [esp+4Ch] [ebp-Ch]

  v7 = 0;
  strcpy(Str, "Encryted Jalnik's File KKKKKKK");
  v6 = strlen(Str);
  getchar();
  FileName = "sample.png";
  v7 = fopen("sample.png", "rb");
  if ( !v7 )
  {
    puts(Buffer);
    exit(0);
  }
  v4 = "sample_en.png";
  Stream = fopen("sample_en.png", "wb");
  for ( i = 0; ; ++i )
  {
    Character = fgetc(v7);
    if ( Character == -1 )
      break;
    Character ^= Str[i % v6];
    putchar(46);
    fputc(Character, Stream);
  }
  fclose(Stream);
  return fclose(v7);
}
```

- 이번엔 아까보다 훨씬 복잡해보입니다.
- 코드 영역 맨 윗쪽에 `char Str[31]` 과 같은 것들은 변수 선언입니다. 다들 잘 아실거라고 생각하고 넘어가겠습니다!
- 우리는 코드 해석을 합시다.

```
int FileEncoding(void)
{
  /*변수들은 코드에서 제외 했습니다.*/

  v7 = 0;

  // Str변수에 "Encryted Jalnik's File KKKKKKK"를 복사합니다.
  strcpy(Str, "Encryted Jalnik's File KKKKKKK");

  // Str변수의 길이를 구합니다.
  v6 = strlen(Str);

  // 단어를 하나 입력 받습니다. 아까 Enter를 눌렀어야 하는 부분네요!
  getchar();
  FileName = "sample.png";

  // 문제 폴더에 있는 sample.png를 read,binary 형태로 열어줍니다.
  v7 = fopen("sample.png", "rb");

  /* sample.png 가 없다면 프로그램 종료 */
  if ( !v7 )
  {
    puts(Buffer);
    exit(0);
  }

  v4 = "sample_en.png";

  // 문제 폴더에 sample_en.png라는 이름으로 파일을 하나 만들겁니다.
  Stream = fopen("sample_en.png", "wb");
  for ( i = 0; ; ++i )
  {
    // v7(sample.png)에서 한 글자씩 불러옵니다.
    Character = fgetc(v7);

    // 만약 불러온 글자가 없다면 반복문을 종료합니다.
    if ( Character == -1 )
      break;
    // 불러왔다면 불러온 글자와 Str[i%v6]와 ^(xor)연산을 해줍니다.
    Character ^= Str[i % v6];

    // 46은 AsciiCode에서 '.' 입니다. 실행시킬 때 나오던 '.' 이겠군요!
    putchar(46);

    // sample_en.png에 연산한 글자를 씁니다.
    fputc(Character, Stream);
  }
  fclose(Stream);

  // 반복이 다 종료됐다면(암호화가 끝났다면) 함수를 종료합니다.
  return fclose(v7);
}
```

```
#추가적으로 Str[i % v6] 연산은 이런식으로 작동합니다.

v6: "Encryted Jalnik's File KKKKKKK"의 길이 : 30
i : Str의 현재 인덱스 번호

[i % v6] : 나머지 연산
i가 0 일때, i % v6 (0 % 30)은 0 입니다.
i가 1 일때, i % v6 (1 % 30)은 1 입니다.
i가 2 일때, i % v6 (2 % 30)은 2 입니다.
i가 3 일때, i % v6 (3 % 30)은 3 입니다.
i가 4 일때, i % v6 (3 % 30)은 4 입니다.
                .
                .
                .
i가 30 일때, i % v6 (30 % 30)은 0 입니다.
i가 31 일때, i % v6 (31 % 30)은 1 입니다.

File의 길이가 얼마이던지 0 ~ 30 만큼만 암호화 시키는 중 이네요!
```

- 이제 POC 코드를 짜봅시다!

> POC

- 우선 우리는 `sample_en.png`가 아니라 `flag_en.png`를 해독해야 합니다.
- Python3로 짜봅시다!

```python
# flag_en.png를 불러옵니다.
with open('flag_en.png', 'rb') as file:
    # flag_en.png를 해독하고 저장할 파일을 생성해줍니다.
    with open('flag_de.png', 'wb') as sol:
        # flag_en.png를 읽어옵니다.
        content = file.read()
```

- 우선 파일을 읽어옵니다.

```python
KEY = "Encryted Jalnik's File KKKKKKK"
KEY_LEN = len(KEY)
```

- 다음 우리 우리는 KEY를 이용하여 암호화 하고 있다는 것을 알았습니다.
- KEY 값을 가져옵시다.

```
import struct

for i in range(len(content)):
    s = content[i] ^ ord(KEY[i % KEY_LEN])       # 복호화 코드!
    print(".")
    sol.write(struct.pack('B', s))              # Byte 형태로 파일에 쓰기!
```

- `XOR` 연산은 우리가 암호학 시간에 배웠던 대로, 같은 값과 다시 연산해주면 본래의 값이 나옵니다.
- 다음 `flag_de.png` 에 복호화 된 내용을 써줍니다.

```
💡 struct document
https://docs.python.org/ko/3/library/struct.html
```

```
import struct

KEY = "Encryted Jalnik's File KKKKKKK"
KEY_LEN = len(KEY)

with open('flag_en.png', 'rb') as file:
    with open('flag_de.png', 'wb') as sol:
        content = file.read()

        for i in range(len(content)):
            s = content[i] ^ ord(KEY[i % KEY_LEN])
            print(".")
            sol.write(struct.pack('B', s))
```

- 전체 코드입니다.
- 실행시켜 Flag를 얻어봅시다!

Flag!!! 😊

SOTI{Ple@se_D0n't_tell_J@LN2K}