




Zero (rev)

1. 문제

간단한 암호화 문제예요. 근데 뭐가 문제인지 잘 해독이 안되네요!
암호화를 풀고 Flag 값을 알아내주세요!

flag.txt를 해독하시면 됩니다.

처음 폴더를 열면 3개의 파일이 있습니다.

 flag	2024-05-21 오전 1:15	텍스트 문서	1KB
 sample	2024-05-21 오전 1:15	텍스트 문서	1KB
 Zero	2024-05-21 오전 1:15	응용 프로그램	130KB

flag.txt와 sample.txt는 zero.exe로 암호화 된 샘플 파일입니다.

우리는 flag.txt를 복호화해야 합니다.

2. 분석 및 POC 작성

Main() 함수

```
__int64 v7; // [rsp+28h] [rbp-58h]
int v8; // [rsp+30h] [rbp-50h]
char v9; // [rsp+34h] [rbp-4Ch]
char v10[32]; // [rsp+40h] [rbp-40h] BYREF
char Str[32]; // [rsp+60h] [rbp-20h] BYREF
FILE *Stream; // [rsp+68h] [rbp-0h]
char *FileName; // [rsp+88h] [rbp+8h]
int v14; // [rsp+94h] [rbp+14h]
int v15; // [rsp+98h] [rbp+18h]
int v16; // [rsp+9Ch] [rbp+1Ch]
int m; // [rsp+A0h] [rbp+20h]
int k; // [rsp+A4h] [rbp+24h]
int j; // [rsp+A8h] [rbp+28h]
int i; // [rsp+ACH] [rbp+2Ch]

__main()
{
    strcpy(Str, "ue-u08,u)Fju~Huah0u");
    strcpy(v10, "_this_is_sample_flag_");
    *(_QWORD *)&v8 = 0LL;
    v7 = 0LL;
    v8 = 0;
    v9 = 0;
    v16 = strlen(Str);
    puts("Key Gen....");
    for ( i = 0; i < v16; ++i )
    {
        v3 = genKey(Str[i]);
        v6[i] = v3;
    }
    v15 = strlen(v6);
    v14 = strlen(v10);
    puts("Encrypt....");
    for ( j = 0; j < v15; ++j )
    {
        for ( k = 0; k < v14; ++k )
        {
            v4 = encrypt(v6[j] % v15, v10[k]);
            v10[k] = v4;
        }
    }
    puts("Write File....");
    FileName = "sample.txt";
    Stream = fopen("sample.txt", "wb");
    for ( m = 0; m < strlen(v10); ++m )
        fprintf(Stream, "%x", (unsigned __int8)v10[m]);
    fclose(Stream);
    return 0;
}
```

- zero.exe를 IDA로 연 후, 디컴파일을 한 화면입니다.
- 코드를 한줄씩 분석해봅시다!

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
    /*
    변수를 선언하는 부분은 너무 길어서 삭제했습니다!
    */
```

```

_main();

// Str과 v10에 각각 변수를 문자열을 할당 해줍니다.
strcpy(Str, "ue~uo&,u}Fju+~HuaH0u");
strcpy(v10, "_this_is_sample_flag_");
*(_QWORD *)v6 = 0LL;
v7 = 0LL;
v8 = 0;
v9 = 0;

// Str 변수의 길이를 구해줍니다.
v16 = strlen(Str);
puts("Key Gen...");
for ( i = 0; i < v16; ++i )
{
    // genKey()함수에 Str[i]를 넣고 호출합니다.
    // 아마도 Str 변수를 복호화하는 과정 같습니다.
    v3 = genKey(Str[i]);
    // 복호화된 key값을 v6 배열에 넣어줍니다.
    v6[i] = v3;
}

// 복호화된 key값의 길이와 v10:flag의 길이를 구해줍니다.
v15 = strlen(v6);
v14 = strlen(v10);
puts("Encrypt...");

// 복호화된 key 길이만큼 반복합니다.
for ( j = 0; j < v15; ++j )
{
    for ( k = 0; k < v14; ++k )
    {
        // encrypt 함수에 v6[j % v15]와 v10:flag를 차례대로 보냅니다.
        // v6[j % v15]와 v10:flag 이 부분은 python으로 코드를 짤 때,
        // 자세히 설명하겠습니다.
        v4 = encrypt(v6[j % v15], v10[k]);

        // encrypt() 함수의 결과 값을 v10 배열에 저장합니다.
        v10[k] = v4;
    }
}
puts("Write File...");
FileName = "sample.txt";
Stream = fopen("sample.txt", "wb");
// sample.txt에 결과 값들을 16진수로 저장합니다.
for ( m = 0; m < strlen(v10); ++m )
    fprintf(Stream, "%x", (unsigned __int8)v10[m]);
fclose(Stream);
return 0;
}

```

- `main()` 함수의 코드를 분석한 결과, `genkey()` 함수로 `key` 값을 복호화 하고, `encrypt()` 함수로 `flag`를 암호화 한 후, 파일에 쓰고 있습니다.
- 다음으로 `genKey()` 함수를 분석해 보겠습니다.

genKey()

```
__int64 __fastcall genKey(char a1)
{
    if ( (unsigned int)check(a1) )
        return (unsigned int)(a1 ^ 0x46);
    else
        return (unsigned int)((((char)(a1 - 32) + 73) % 95 + 32));
}
```

- `genKey()` 함수입니다. 굉장히 간단하네요! 😊

```
__int64 __fastcall genKey(char a1)
{
    // check() 함수를 호출합니다.
    if ( (unsigned int)check(a1) )
        // check() 함수의 결과 값이 1(참)이라면 a1 ^ 0x46을 return 합니다.
        return (unsigned int)(a1 ^ 0x46);
    else
        // 0(거짓)일 경우 ((a1 - 32) + 73) % 95 + 32를 한 값을 return 합니다.
        return (unsigned int)((((char)(a1 - 32) + 73) % 95 + 32));
}
```

- `check()` 함수까지 확인하고 분석해보겠습니다.

```
__bool8 __fastcall check(char a1)
{
    return a1 == 0x46;
}
```

- `check()` 함수도 굉장히 간단하군요!
- 그냥 `a1 == 0x46` 이면 `1(참)` 을 return 합니다.
- 즉 `main()` 함수에서 넘어온 값이 `0x46` 이면 `0x46` 과 XOR 연산을 하고 아니면 `else` 구문에 연산을 하고 있습니다.
- Python으로 복호화 코드를 짜봅시다! 😊

```
def check(c):
    return c == 0x46

def genKey(c):
    if check(c):
        return c ^ 0x46
    else:
        return (((c - 32) + 73) % 95 + 32)

KEY = 'ue~uo&,u}Fju+~HuaH0u'

v3 = []

for i in range(20):
    v3.append(genKey(ord(KEY[i])))

for i in v3:
    print(chr(i), end='')
```

- 한번 실행시켜 볼까요?

```
_Oh_You_gT_th2_K2y_
```

- `key` 값을 얻었습니다! 하지만 아직 갈길이 멉니다... 😓
- 이제 `encrypt()` 함수를 분석해 봅시다!

encrypt()

```
_int64 __fastcall encrypt(unsigned __int8 a1, unsigned __int8 a2)
{
    return a1 + (unsigned int)a2;
}
```

- `encrypt()` 함수도 굉장히 간단하네요..?
- 넘어온 인자 2개를 더한 다음 return 합니다!
- 그러면 `encrypt()` 도 코드로 한번 짜봅시다!
- 복호화 연산을 하는 함수이기 때문에 이름은 `decrypt()` 로 바꿔주겠습니다.

```
def decrypt(a, b):
    return (a - b) % 256
```

- 여기서 주의해야 할 점은 이번엔 `-` 연산이라는 점 입니다.
- XOR 의 경우 복호화 연산을 할때, 동일한 값과 다시 XOR 을 해주면 되지만 `+` 연산은 그렇지 않습니다. `-` 연산을 해야합니다!
- 코드 상에서 `% 256` 을 하는 이유는 여러분이 찾아보시면 더 도움이 될 것 같습니다! 밑에 있는 친구가 잘 알려줄거라고 생각합니다 😊

퀴즈: <https://wrtm.ai/>

🔊 HINT: 확장 ASCII

- 그럼 이제 `main()` 함수에서 파일을 읽어오는 부분부터 짚 코드를 작성해 보겠습니다.

```
def check(c):
    return c == 0x46

def genKey(c):
    if check(c):
        return c ^ 0x46
    else:
        return (((c - 32) + 73) % 95 + 32)

def decrypt(a, b):
    return (a - b) % 256

KEY = 'ue~uo&,u}Fju+~HuaH0u'

v3 = []

for i in range(20):
    v3.append(genKey(ord(KEY[i])))

len_key = len(v3)
with open('flag.txt', 'r') as f:
```

```

line = f.readline()

# flag.txt에서 읽어온 데이터를 1byte씩 리스트에 저장합니다.
# 즉 [cb, c7, cc, c1 ...] 이런식으로 16진수로 저장합니다.
hex_list = [int(line[i:i+2], 16) for i in range(0, len(line), 2)]
len_flag = len(hex_list)

# key의 길이 만큼 반복합니다.
for i in range(len_key):

    # hex_list의 길이만큼 반복합니다.
    for j in range(len_flag):
        # hex_list의 값과 v3[i % len(v3)]의 값을 뺍니다.
        # v3[i % len(v3)]의 연산 부분은 밑에서 설명드리겠습니다.
        hex_list[j] = decrypt(hex_list[j], v3[i % len_key])

for i in hex_list:
    print(chr(i), end='')

```

c 코드에서 건너뛴 `v6[j % v15]` 부분에 대한 설명을
우리에게 친숙한 python코드 `v3[i % len_key]`로 설명하겠습니다.

`len(v3)`의 값은 임의의 값(20)으로 대체 하겠습니다.

```

i가 0이고 len_key가 20일때, 1 % 20 = 0
i가 1이고 len_key가 20일때, 1 % 20 = 1
i가 2이고 len_key가 20일때, 1 % 20 = 2
i가 2이고 len_key가 20일때, 1 % 20 = 3
.
.
.
i가 20이고 len_key가 20일때, 20 % 20 = 0
i가 21이고 len_key가 20일때, 20 % 20 = 1

```

즉 `v3[0]`, `v3[1]`, `v3[2]` ... 순서로 반복됩니다.

- 그렇다면 이제 `decrypt(hex_list[j], v3[i % len_key])` 이 부분을 자세히 살펴보겠습니다.

```

decrypt(hex_list[j], v3[i % len_key])

# 이 부분의 흐름은 이렇게 됩니다.
# for j in range(len_flag))의 첫번째 반복
hex_list[0] - v3[0 % len(v3)] -> '_'
hex_list[1] - v3[0 % len(v3)] -> '_'
hex_list[2] - v3[0 % len(v3)] -> '_'
.
.
.
# for j in range(len_flag))의 두번째 반복
hex_list[0] - v3[1 % len(v3)] -> 'o'
hex_list[1] - v3[1 % len(v3)] -> 'o'
hex_list[2] - v3[1 % len(v3)] -> 'o'
.
.
.

```

```
# for j in range(len_flag)의 세번째 반복
hex_list[0] - v3[1 % len(v3)] -> '_'
hex_list[1] - v3[1 % len(v3)] -> '_'
hex_list[2] - v3[1 % len(v3)] -> '_'
```

- key 값인 `_oh_You_gT_th2_k2y_` 를 `hex_list` 각 인덱스에서 반복마다 모두 빼주고 있습니다.
- 이렇게 분석과 복호화 코드까지 작성했습니다!
- 실행시켜봅시다 😊



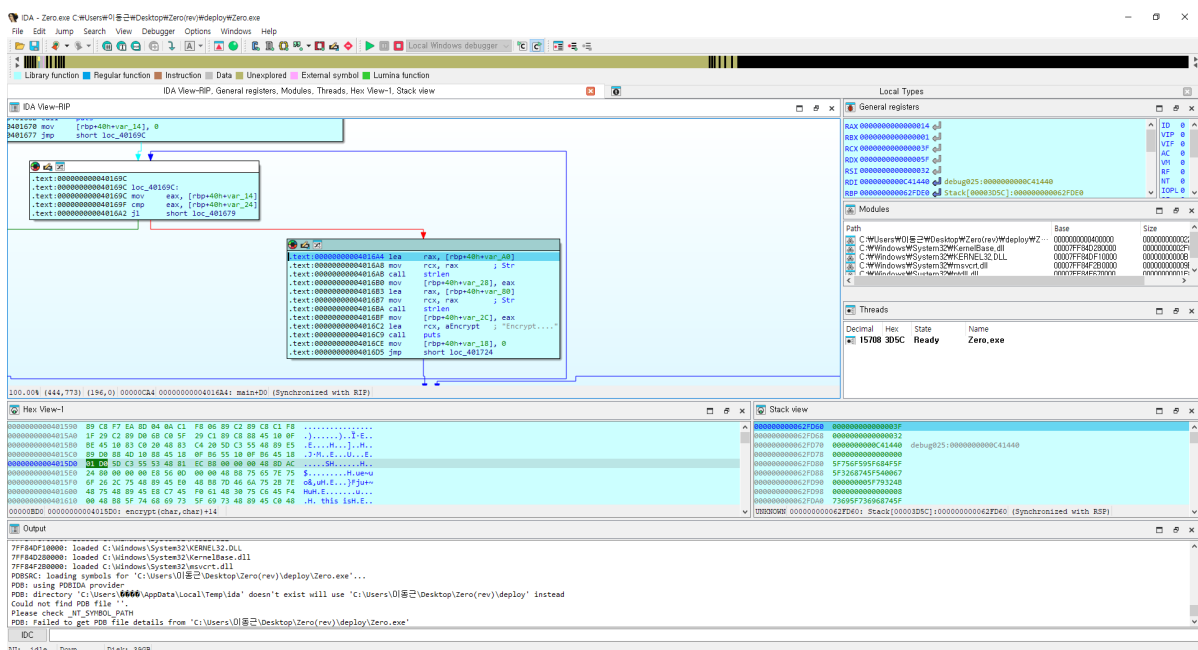
- 어... 이거 결과 값이.. 이상합니다...
- 우리가 놓친게 있는 것 같습니다. 자세히 보니 `_oh_You_gT_th2_k2y_` 부분에 뭔가 빠진 것 같습니다. 😊
- `genKey()` 가 동작하는 부분을 한번 살펴봐야겠습니다.

genKey() 동적 분석

- 먼저 IDA를 열고 `breakpoint` 를 설정해주겠습니다.



- `genKey()` 가 끝난 시점인 `strlen(v6)` 부분에 가서 `v6` 가 어떤 값을 가지고 있는지 보겠습니다.
- `v15 = strlen(v6)` 부분에 F2를 눌러 `breakpoint` 를 설정해주세요!
- 다음 F9를 눌러 동적 디버깅을 시작하겠습니다!



- 동적 디버깅이 시작되었습니다. 어셈블리어네요 .. 우리는 이 친구랑은 친하지가 않습니다. **F5**를 눌러 디컴파일을 해주세요!

```

20  _main();
21  strcpy(Str, "ue-u08,uFju+~HuaH8u");
22  strcpy(v10, "_this_is_sample_flag_");
23  *(_DWORD *)v6 = 0LL;
24  v7 = 0LL;
25  v8 = 0;
26  v9 = 0;
27  v10 = strlen(Str);
28  puts("Key Gen...");
29  for ( i = 0; i < v10; ++i )
30  {
31      v3 = genKey(Str[i]);
32      v6[i] = v3;
33  }
34  v15 = strlen(v6);
35  v14 = strlen(v10);
36  puts("Encrypt...");
37  for ( j = 0; j < v15; ++j )
38  {
39      for ( k = 0; k < v14; ++k )
40      {
41          v4 = encrypt(v6[j] % v15, v10[k]);
42          v10[k] = v4;
43      }
44  }
45  puts("Write File...");
46  FileName = "sample.txt";
47  Stream = fopen("sample.txt", "wb");
48  for ( m = 0; m < strlen(v10); ++m )
49      fprintf(Stream, "%x", (unsigned __int8)v10[m]);
50  00000C44 main:35 (4016A4)
  
```

- 한결 보기 편하네요 🍷
- 이제 **v6**를 눌러서 **genKey()**로 만들어진 **key** 값을 보겠습니다.

Stack[00003D5C]:0000000000062FD80	db	5Fh	; _
Stack[00003D5C]:0000000000062FD81	db	4Fh	; 0
Stack[00003D5C]:0000000000062FD82	db	68h	; h
Stack[00003D5C]:0000000000062FD83	db	5Fh	; _
Stack[00003D5C]:0000000000062FD84	db	59h	; Y
Stack[00003D5C]:0000000000062FD85	db	6Fh	; o
Stack[00003D5C]:0000000000062FD86	db	75h	; u
Stack[00003D5C]:0000000000062FD87	db	5Fh	; _
Stack[00003D5C]:0000000000062FD88	db	67h	; g
Stack[00003D5C]:0000000000062FD89	db	0	
Stack[00003D5C]:0000000000062FD8A	db	54h	; T
Stack[00003D5C]:0000000000062FD8B	db	5Fh	; _
Stack[00003D5C]:0000000000062FD8C	db	74h	; t
Stack[00003D5C]:0000000000062FD8D	db	68h	; h
Stack[00003D5C]:0000000000062FD8E	db	32h	; 2
Stack[00003D5C]:0000000000062FD8F	db	5Fh	; _
Stack[00003D5C]:0000000000062FD90	db	48h	; K
Stack[00003D5C]:0000000000062FD91	db	32h	; 2
Stack[00003D5C]:0000000000062FD92	db	79h	; y
Stack[00003D5C]:0000000000062FD93	db	5Fh	; _
Stack[00003D5C]:0000000000062FD94	db	0	

- 우리가 구했던 **key** 값이 있습니다.
- 근데 좀 다릅니다..? **_Oh_You_gT_th2_k2y_**의 **g**와 **T** 사이에 **0**이 들어가 있습니다!
- 맞습니다.. **NULL** 값이네요.. 저것 때문이었군요...
- **C**언어에서는 문자열의 마지막을 **NULL** 값으로 판단합니다.
- 즉 **strlen()** 함수는 **_Oh_You_g**까지만을 **key** 값이라고 판단하고 있는겁니다.
- 하지만 **python**에서의 문자열은 **Nu1l** 문자를 특별한 종료 신호로 사용하지 않습니다.

```

'''
Python의 문자열은 내부적으로 길이 정보를 포함하고 있으며, 이 길이 정보를 사용하여 문자열의
끝을 알 수 있습니다.

즉, Python 문자열은 시작 인덱스와 함께 문자열의 길이를 저장하고 있어서, 문자열의 끝을 정확히
알 수 있습니다.

이러한 방식 덕분에 Python에서는 문자열의 처리가 매우 유연하며,
널 문자와 같은 특정 문자에 의존하지 않고도 문자열의 시작부터 끝까지 다룰 수 있습니다.
'''

# python 간단한 예시
s = "hello\0world"
print(s)
  
```

- strlen() 설명
- ✓ strlen document: <https://www.ibm.com/docs/ko/i/7.3?topic=functions-strlen-determine-string-length>
- 설명을 참고하세요 :)

- 마지막으로 코드를 작성해 보겠습니다!

```
def check(c):
    return c == 0x46

def genKey(c):
    if check(c):
        return c ^ 0x46
    else:
        return (((c - 32) + 73) % 95 + 32)

def decrypt(a, b):
    return (a - b) % 256

KEY = 'ue~uo&,u}Fju+~HuaH0u'

v3 = []

for i in range(20):
    v3.append(genKey(ord(KEY[i])))

# len_key 부분만 변경하겠습니다 !
len_key = len('_Oh_You_g')

with open('flag.txt', 'r') as f:
    line = f.readline()
    hex_list = [int(line[i:i+2], 16) for i in range(0, len(line), 2)]
    len_flag = len(hex_list)

    for i in range(len_key):
        for j in range(len_flag):
            hex_list[j] = decrypt(hex_list[j], v3[i % len_key])

    for i in hex_list:
        print(chr(i), end='')
```

FLAG 

```
SOTI{Yeah..It's..Null..00!}
```

```
SOTI{Yeah..It's..Null..00!}
```

- Flag를 획득했습니다!