Encoder writup

1. 이 문제는 base64 과정의 테이블과 알고리즘을 커스텀한 문제입니다.

2. 프로그램을 실행하면 문자열을 입력 받고, 입력된 문자열에 따라 다른 문자열이 출력됨.

```
user@LCWsambook:~$ ./test
Hello
8dOWrxhI
```

3. IDA에서 열어보면 main 함수에서 아래와 같은 코드를 볼 수 있다.

```
while ( v2 < v8 )
{
    v9 = (((((((__int64)*((char *)v11 + v2) << 8) | *((char *)v11 + v2
+ 1)) << 8) | *((char *)v11 + v2 + 2)) << 8) | *((char *)v11 + v2 + 3))
<< 8) | *((char *)v11 + v2 + 4);
    v10[v3] = byte_4020[(v9 >> 35) & 0x1F];
    v10[v3 + 1] = byte_4020[(v9 >> 30) & 0x1F];
    v10[v3 + 2] = byte_4020[(v9 >> 25) & 0x1F];
    v10[v3 + 3] = byte_4020[(v9 >> 20) & 0x1F];
    v10[v3 + 4] = byte_4020[(v9 >> 15) & 0x1F];
    v10[v3 + 5] = byte_4020[(v9 >> 10) & 0x1F];
    v10[v3 + 6] = byte_4020[(v9 >> 5) & 0x1F];
    v10[v3 + 7] = byte_4020[v9 & 0x1F];
    v2 += 5;
    v3 += 8;
}
if ( v8 % 5 == 1 || v8 == 1 )
{
  for ( i = 1; i <= 6; ++i )
    v10[v3 - i] = 61;
}
else if ( v8 % 5 == 2 || v8 == 2 )
{
  for ( j = 1; j <= 4; ++j )
    v10[v3 - j] = 61;
}
else if ( v8 % 5 == 3 || v8 == 3 )
{
  for ( k = 1; k <= 3; ++k )
    v10[v3 - k] = 61;
}
else if ( v8 % 5 == 4 || v8 == 4 )
{
  v10[v3 - 1] = 61;
}
```

4. While문 안에서 8비트씩 5번 왼쪽으로 시프트를 하고 있고, 그 값을 0x1F와의 연산을 통해 5비트씩 짤라서 특정 테이블을 참조해 배열에 저장하고 있는 것을 확인할 수 있다.

5. 테이블(byte_4020)을 확인해보면 32개의 문자열이 들어가 있다.

```
; _BYTE byte_4020[32]
byte_4020       db 62h, 64h, 66h, 68h, 6Ah, 6Ch, 6Eh, 70h, 36h, 38h, 2Bh
                                        ; DATA XREF: sub_11C8+27A↑o
                                        ; sub_11C8+2AE↑o ...
                db 41h, 43h, 45h, 47h, 49h, 4Bh, 4Dh, 4Fh, 51h, 53h, 55h
                db 57h, 59h, 72h, 74h, 76h, 78h, 7Ah, 5Ah, 32h, 34h
_data           ends
```

6. 입력된 문자열의 변환되었을 때 뒤에 있는 "="을 넣어주는 코드이다.

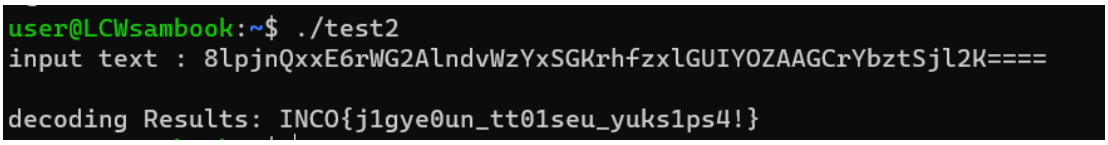
```
if ( v8 % 5 == 1 || v8 == 1 )
{
  for ( i = 1; i <= 6; ++i )
    v10[v3 - i] = 61;
}
else if ( v8 % 5 == 2 || v8 == 2 )
{
  for ( j = 1; j <= 4; ++j )
    v10[v3 - j] = 61;
}
else if ( v8 % 5 == 3 || v8 == 3 )
{
  for ( k = 1; k <= 3; ++k )
    v10[v3 - k] = 61;
}
else if ( v8 % 5 == 4 || v8 == 4 )
{
  v10[v3 - 1] = 61;
}
```

7. 위 과정들을 분석했다면 encrypt 코드를 바탕으로 decrypt 코드를 작성하면 된다. (예시)

```
#include<stdio.h>
#include<string.h>
#include<stdint.h>

int main()
{
 uint64_t table[256] = {
         -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,  /* 00-0F */
         -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,  /* 10-1F */
         -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,10,-1,-1,-1,-1,  /* 20-2F */
```

```
        -1,-1,30,-1,31,-1, 8,-1, 9,-1,-1,-1,-1, 0,-1,-1,  /* 30-3F */
        -1,11,-1,12,-1,13,-1,14,-1,15,-1,16,-1,17,-1,18,  /* 40-4F */
        -1,19,-1,20,-1,21,-1,22,-1,23,29,-1,-1,-1,-1,-1,  /* 50-5F */
        -1,-1, 0,-1, 1,-1, 2,-1, 3,-1, 4,-1, 5,-1, 6,-1,  /* 60-6F */
         7,-1,24,-1,25,-1,26,-1,27,-1,28,-1,-1,-1,-1,-1,  /* 70-7F */
        -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,  /* 80-8F */
        -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,  /* 90-9F */
        -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,  /* A0-AF */
        -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,  /* B0-BF */
        -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,  /* C0-CF */
        -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,  /* D0-DF */
        -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,  /* E0-EF */
        -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1   /* F0-FF */
};
char text[500000] = { 0, };
char decoding[100000] = { 0, };
int tLength = 0, tIndex = 0, eIndex = 0;
uint64_t step = 0;

printf("input text : ");
fgets(text, 500000, stdin);
text[strlen(text) - 1] = '\0';

tLength = strlen(text);

while (tIndex < tLength)
{
        step = 0;
        step = step << 0 | table[text[tIndex + 0]];
        step = step << 5 | table[text[tIndex + 1]];
        step = step << 5 | table[text[tIndex + 2]];
        step = step << 5 | table[text[tIndex + 3]];
        step = step << 5 | table[text[tIndex + 4]];
        step = step << 5 | table[text[tIndex + 5]];
        step = step << 5 | table[text[tIndex + 6]];
        step = step << 5 | table[text[tIndex + 7]];

        decoding[eIndex + 0] = step >> 32;
        decoding[eIndex + 1] = (step >> 24) & 0xFF;
        decoding[eIndex + 2] = (step >> 16) & 0xFF;
        decoding[eIndex + 3] = (step >> 8) & 0xFF;
        decoding[eIndex + 4] = (step >> 0) & 0xFF;

        eIndex += 5;
        tIndex += 8;
}

printf("\ndecoding Results: ");
for (int i = 0; i < eIndex; i++)
{
        printf("%c", decoding[i]);
}
printf("\n");
}
```

8. 플래그 획득


user@LCWsambook:~$ ./test2
input text : 8lpjnQxxE6rWG2AlndvWzYxSGKrhfzxlGUIYOZAAGCrYbztSjl2K====
decoding Results: INCO{j1gye0un_tt01seu_yuks1ps4!}