FLACK 워게임 풀이

박찬암 (hkpco)

chanam.park@hkpco.kr

http://hkpco.kr/

2008. 6. 30

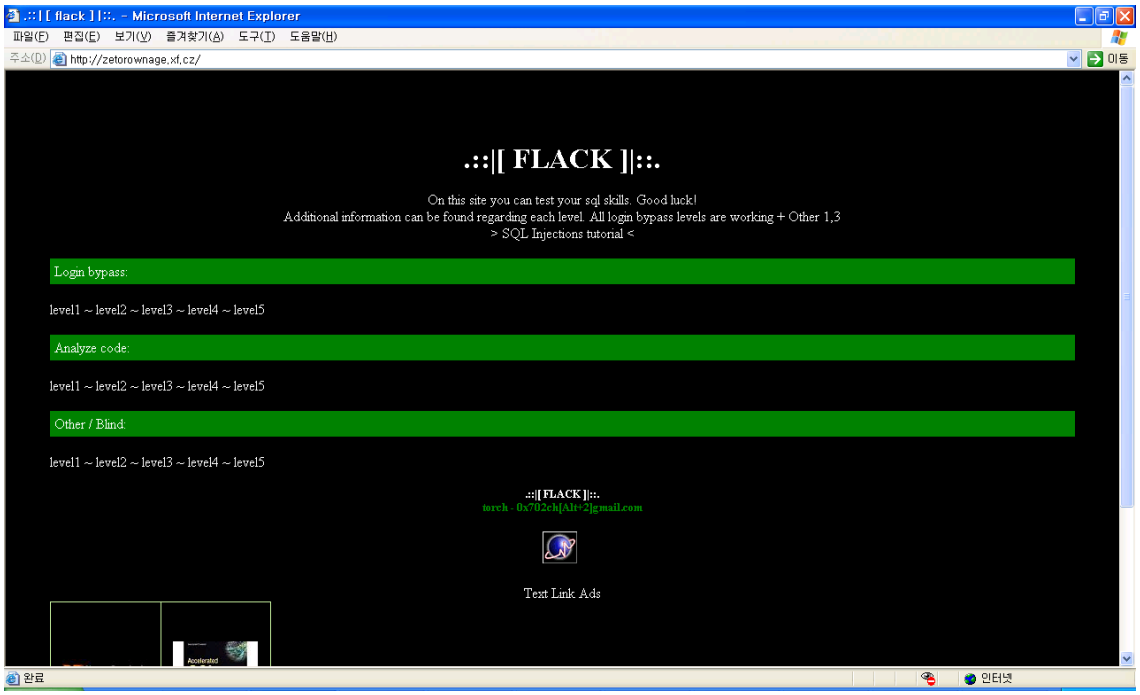
목차

0. 잡담

마침 심심하던 참에 우연히 흥미로운 워게임 하나를 발견하였다. 나의 무료함을 달래주리라 믿고 풀어보았는데, 전체적으로 어렵지 않고 무난하며 재미있었던 덕분에 잠깐이나마 즐겁게 시간을 보낼 수 있었다.

1. FLACK 워게임 소개

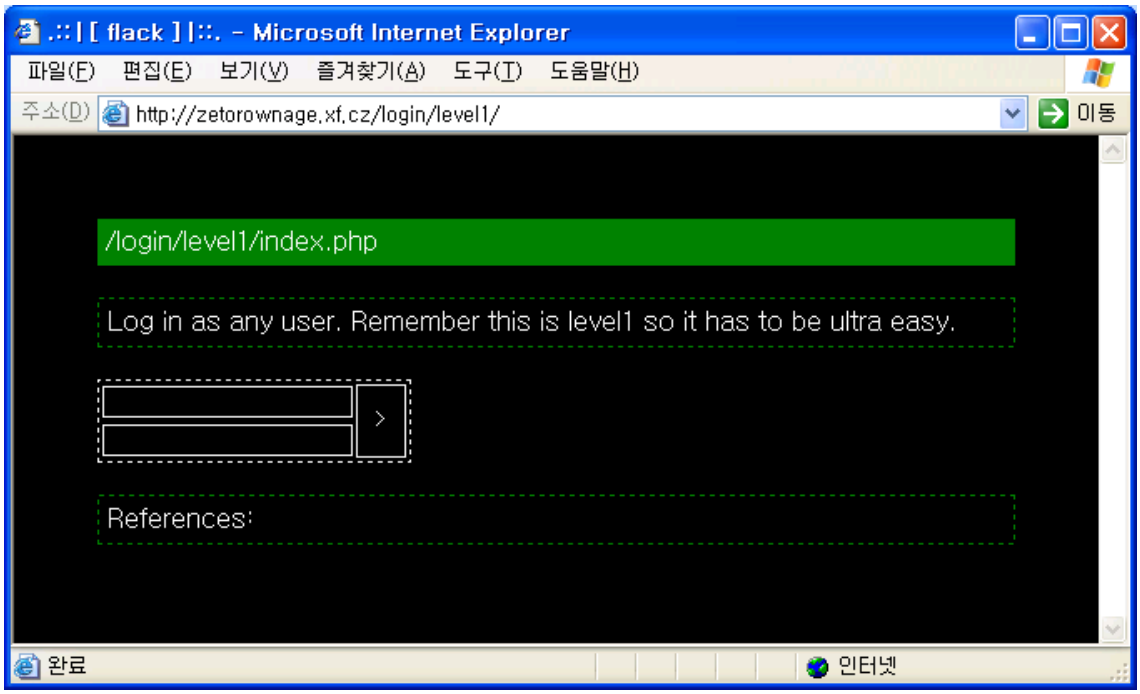
FLACK 워게임은 SQL Injection과 관련된 문제들로 이루어져 있으며 서버는 Linux, Mysql, Apache 기반이다. 모든 문제는 로그인에 성공하는 것이 문제를 푼 것으로 간주되는 다소 특이한 컨셉을 가지고 있으며, 전체 문제는 각각 Level1~Level5, Other1, Other3으로 구성되어 있다. 문제 유형은 기본적인 인증 우회부터 LOAD_FILE, Blind SQL Injection 등 어느 정도 대표적인 기법들을 포함하고 있으며, 워게임은 http://zetorownage.xf.cz/ 에서 풀 수 있다.



< FLACK 워게임의 시작 페이지 >

2. Level1

Level1 문제를 클릭하면 다음과 같은 로그인 페이지를 볼 수 있다.



테스트 결과 특수문자를 통하여 SQL 오류를 유도할 수 있었으며, 첫 번째 폼에 quote 문자를 적용했을 당시의 에러 메시지는 다음과 같다.

> The americans launched their rockets. Today we will all die.
>
> You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'aa' and pass='bb'' at line 1

입력한 문자열은 빨간색으로 표시하였으며, quote 문자에 의해서 오류가 발생하였다.

이를 통해 페이지의 로그인 처리 당시 SQL 쿼리는 다음과 같이 예상할 수 있다.

```
select * from member where user='$_POST['user']' and pass='$_POST['pass']';
```

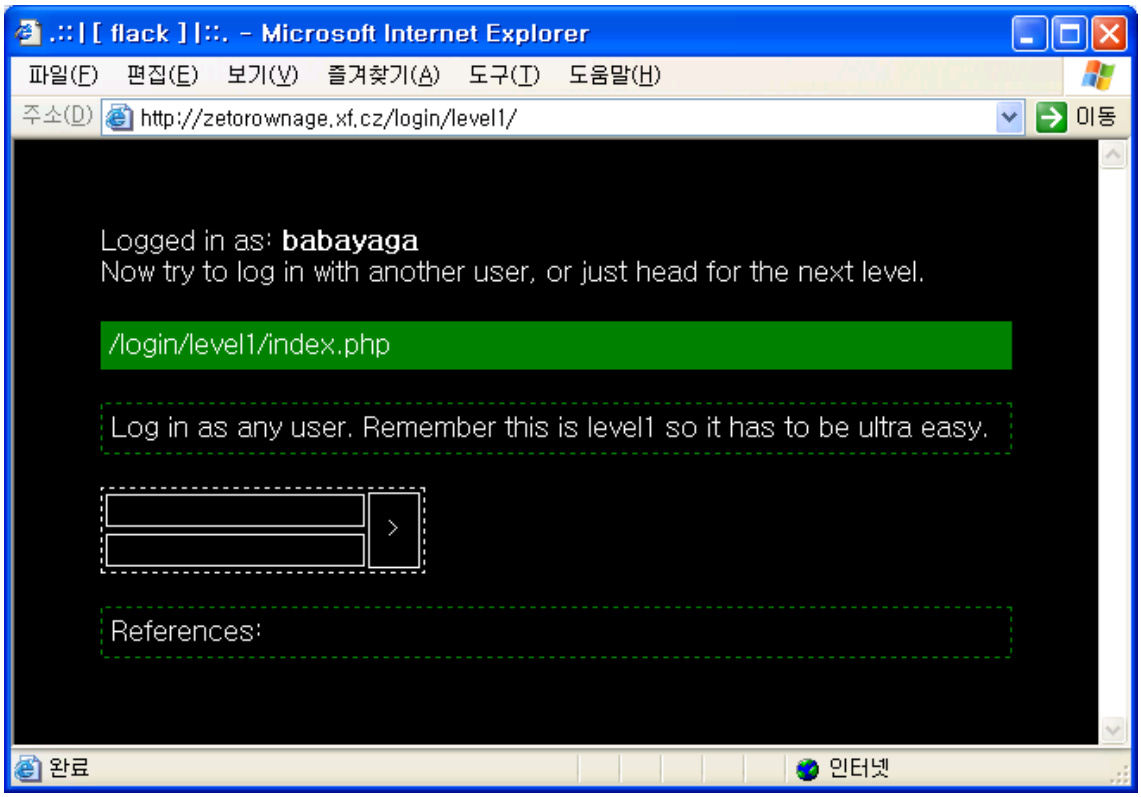
로그인을 우회하기 위하여 각각 다음과 같이 입력하였다.

위와 같은 공격 시도후의 SQL 쿼리는 다음과 같이 예상할 수 있다.

```
select * from member where user="or 1=1# and pass='whatever';
```

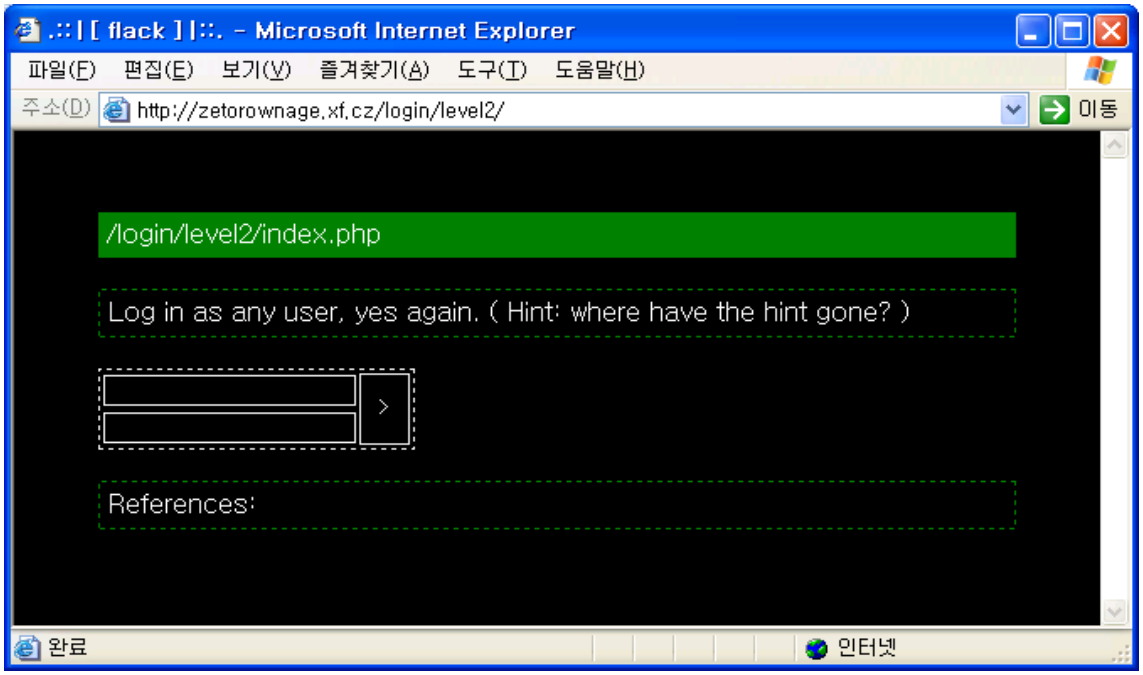
user는 무조건 참이 되고, 주석문 #에 의해서 그 이후의 쿼리는 무시 될 것이므로 예상대로라면 로그인이 이루어 질 것이다.

실제로 시도해 보면 babayaga라는 사용자로 로그인 되었다는 메시지를 볼 수 있다.



3. Level2

Level2도 역시 다음과 같은 로그인 페이지를 볼 수 있다.



우선 고의로 에러를 발생시켜 보기 위해 아래와 같이 입력하였다.

그에 대한 에러 메시지는 다음과 같았다.

> The americans launched their rockets. Today we will all die.
>
> You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'b' and pass='c'd')' at line 1

파란색으로 표시한 문자열은 기존의 로그인 처리를 위한 SQL 쿼리이고, 빨간색으로 된 문장은 우리가 폼에 입력한 문자열이다. 결과를 통하여 quote 문자를 이용한 쿼리 조작이 가능하게 보이지만, 에러 메시지를 살펴보면 로그인 처리를 위한 SQL 쿼리가 Level1과는 조금 다르다는 것을 알 수 있다. 로그인 처리 당시 SQL 쿼리는 다음과 같이 예상할 수 있다.

```
select * from member where (user='$_POST['user']' and pass='$_POST['pass']');
```

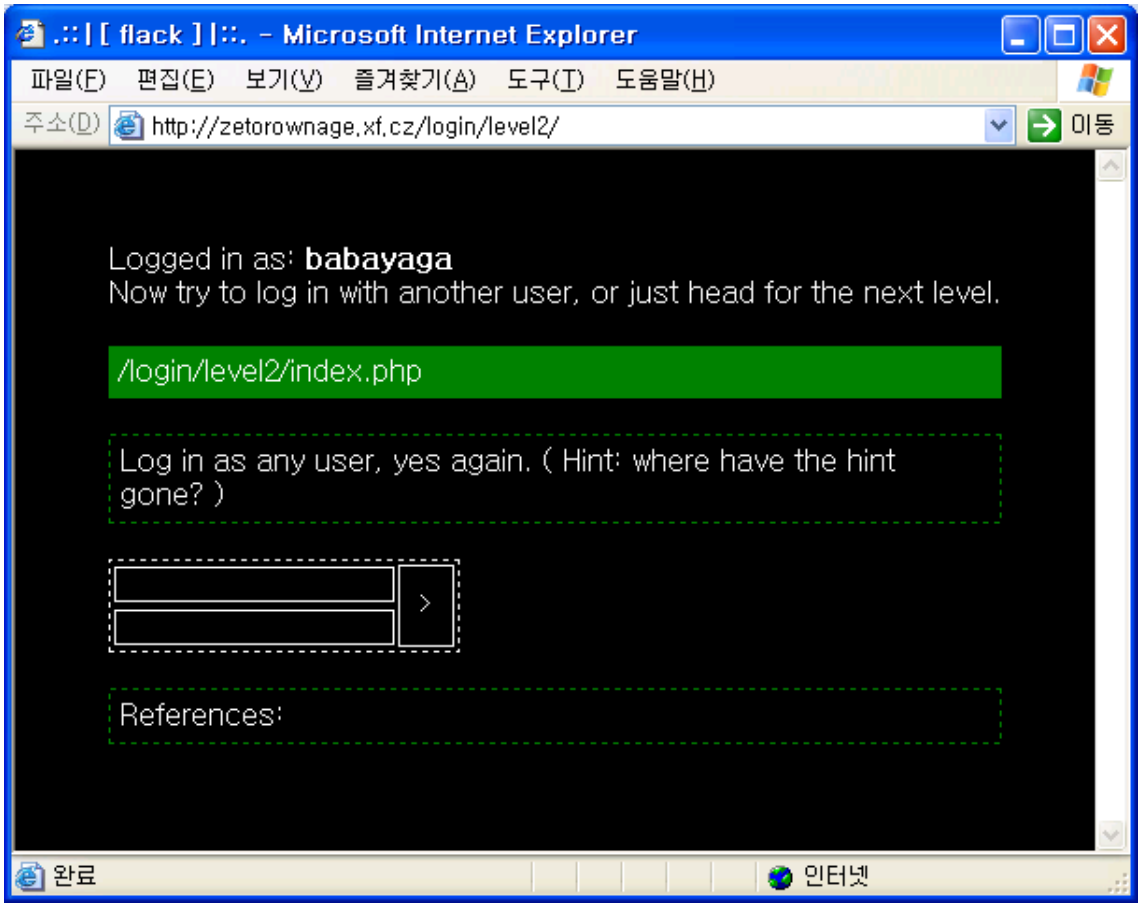
로그인 우회를 위해 다음과 같이 입력하였다.

위와 같은 공격 시도 후의 SQL 쿼리는 다음과 같이 예상할 수 있다.

```
SELECT * from member where (user='' or 1=1)#' and pass='whatever you want');
```

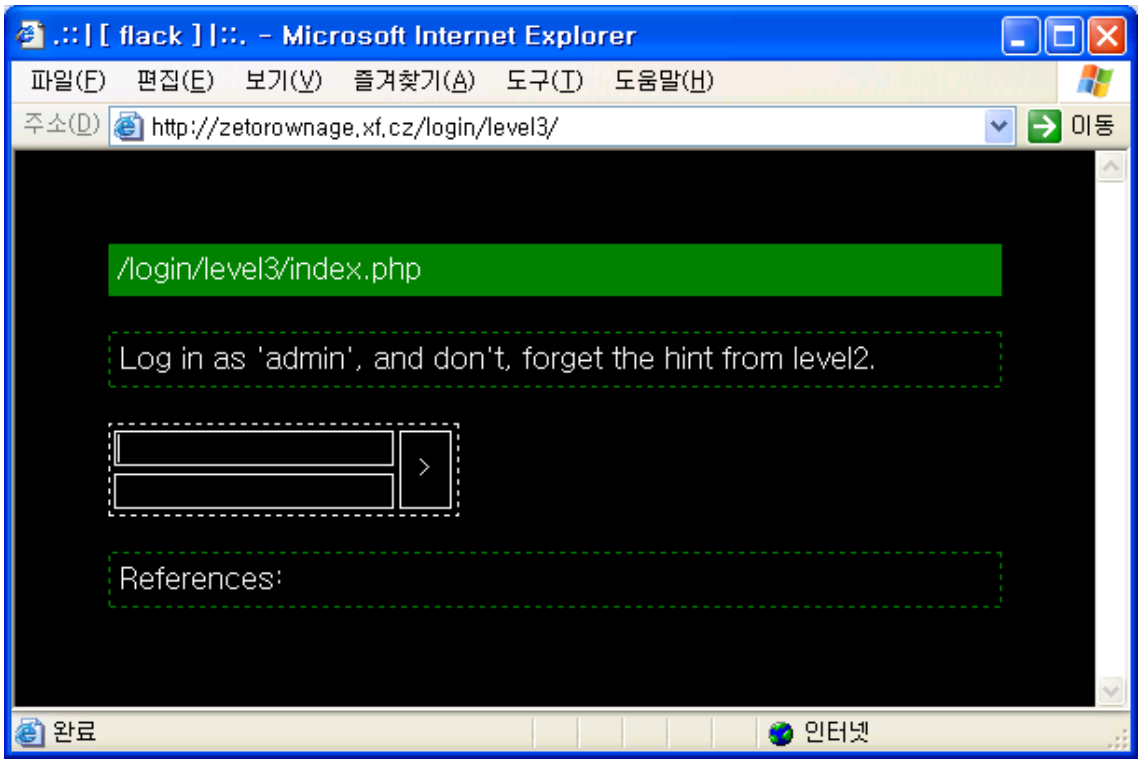
user의 조건은 무조건 참이 되고, ')' 문자를 통해 쿼리를 닫은 뒤에 주석문(#)을 추가하면 이후의 쿼리는 무시 될 것이므로 예상대로라면 전체 쿼리는 참이 되어 로그인에 성공 할 것이다.

실제로 공격 문자열을 입력해 보면 다음과 같이 babayaga라는 유저로 로그인이 되었음을 확인할 수 있다.



4. Level3

Level3의 로그인 페이지는 다음과 같다. admin이라는 주어진 사용자로 로그인을 하는 문제이며 Level2에서의 힌트를 잊지 말라고 적혀있다. 아마도 전 단계와 유사한 취약성을 가지고 있는 레벨인 것 같았다.



주어진 각 폼에 quote 문자 등을 통하여 고의적으로 SQL 오류를 유도해본 결과 아이디를 입력하는 폼, 즉 첫 번째 입력 박스에서만 이전 단계와 동일한 구성의 인젝션 취약성이 발생하였다. 아마도 로그인 처리를 위한 SQL 쿼리는 다음과 같이 Level2와 거의 동일하다고 보면 되겠지만, 두 번째 입력 폼은 addslashes(), mysql_real_escape_string()과 같은 함수를 통하여 quote 문자 등을 필터하고 있는 것처럼 보였다.

[$_POST['pass'] 필터링]

```
select * from member where (user='$_POST['user']' and pass='$_POST['pass']');
```

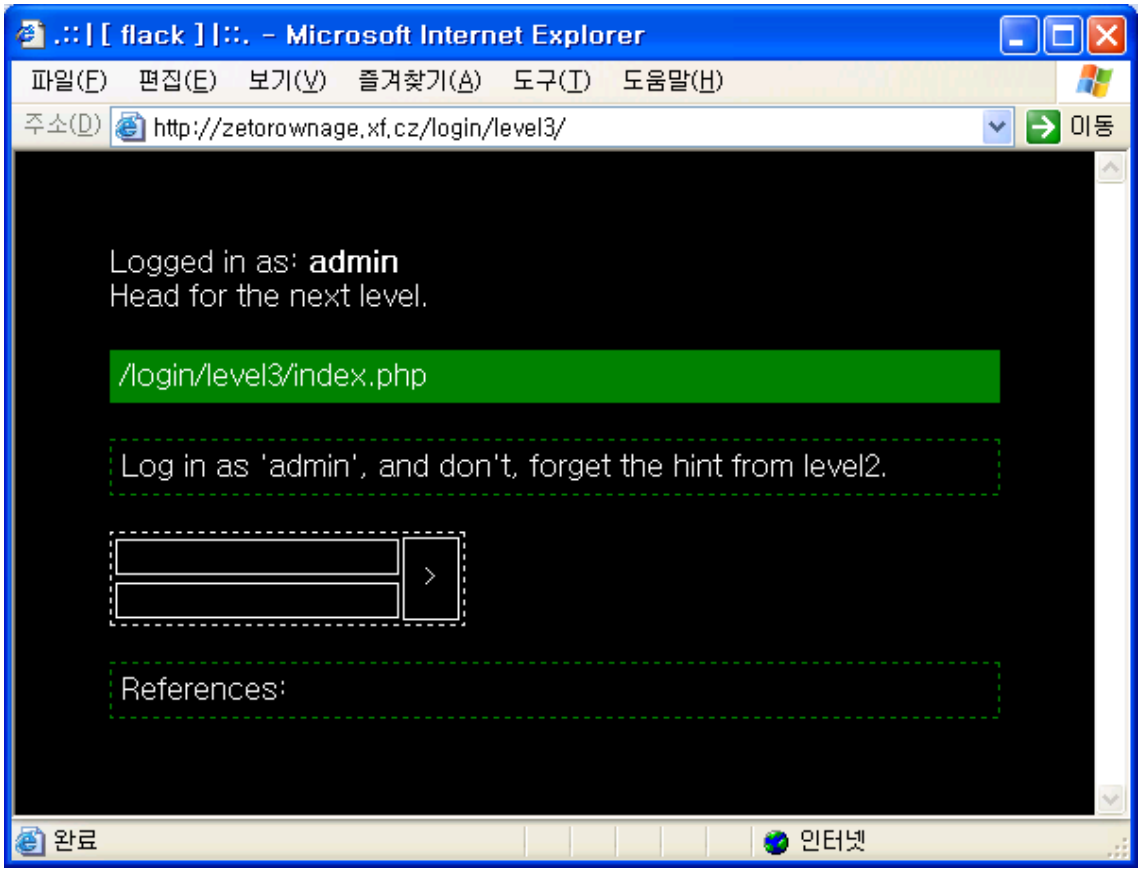
두 번째 폼의 필터링 때문에 다음과 같이 첫 번째 입력 폼을 대상으로 공격 코드를 구성하였다.

위와 같은 공격 시도 후의 SQL 쿼리는 다음과 같이 예상할 수 있다.

```
select * from member where (user='admin' and 1=1)#' and pass='hackko');
```

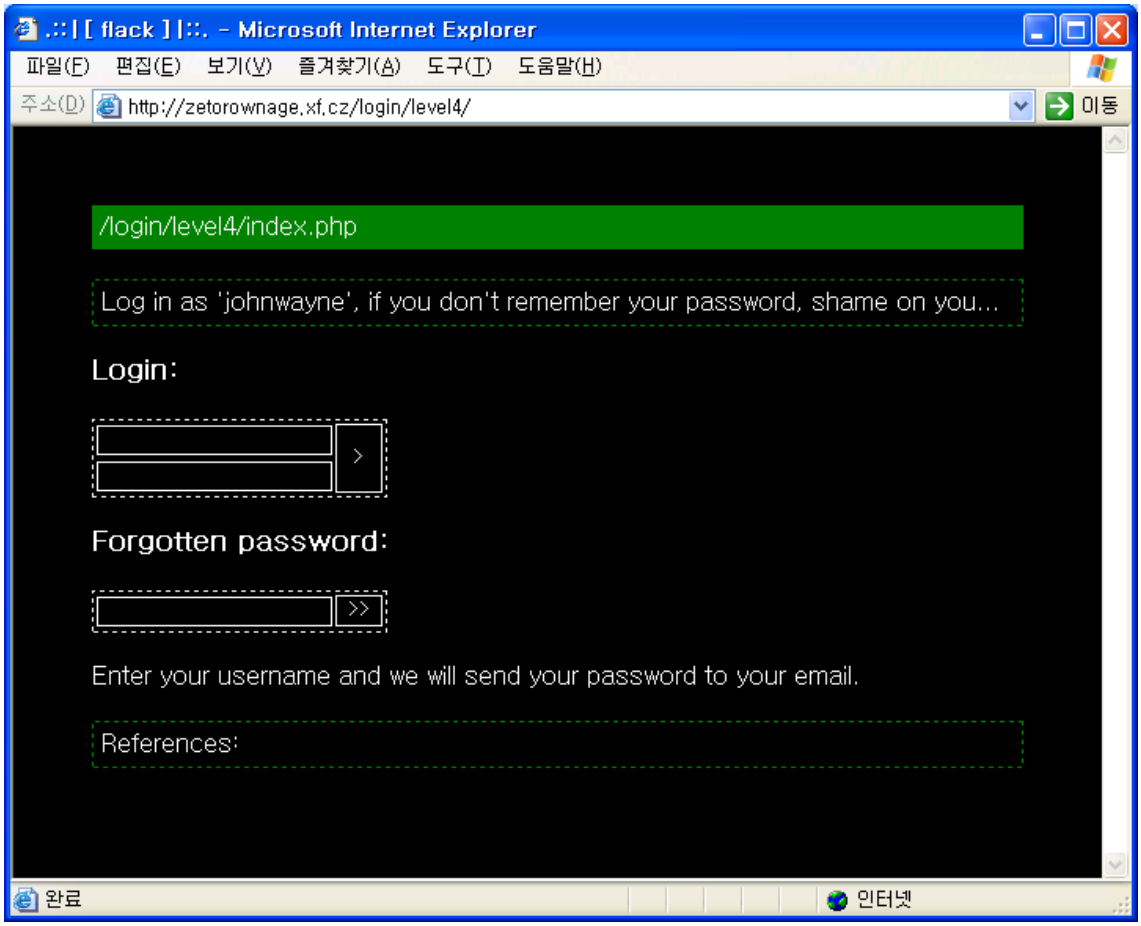
사용자 계정은 admin이며 1=1(무조건 참)이라는 조건 이후에 주석문(#)을 통하여 패스워드를 체크하는 쿼리를 무력화 시켜 로그인을 우회하는 원리이다.

실제로 공격을 시도하면 다음과 같이 admin 계정으로 로그인 되었음을 확인할 수 있다.



5. Level4

이전 레벨들과는 달리 로그인 폼 이외에 잃어버린 패스워드를 찾기 위한 폼이 추가되었다.



살펴본 결과 로그인 폼에서는 취약한 부분을 찾을 수 없었으며, 대신 Forgotten password: 아래 부분의 입력 폼에서 SQL 인젝션 취약성이 발생되었다. 주어진 폼에 hktest' 문자열을 입력하면 고의적으로 에러를 발생시킬 수 있으며, 해당 메시지는 다음과 같다.

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''hktest'' at line 1

폼에 사용자 이름을 입력 받아서 그에 해당하는 패스워드를 메일로 보내주도록 구현하는 과정에서 취약한 코드가 작성된 듯 했으며, SQL 쿼리는 다음과 같이 예상할 수 있다.

```
select username, password from login_table where username='_POST[$fuser]';
```

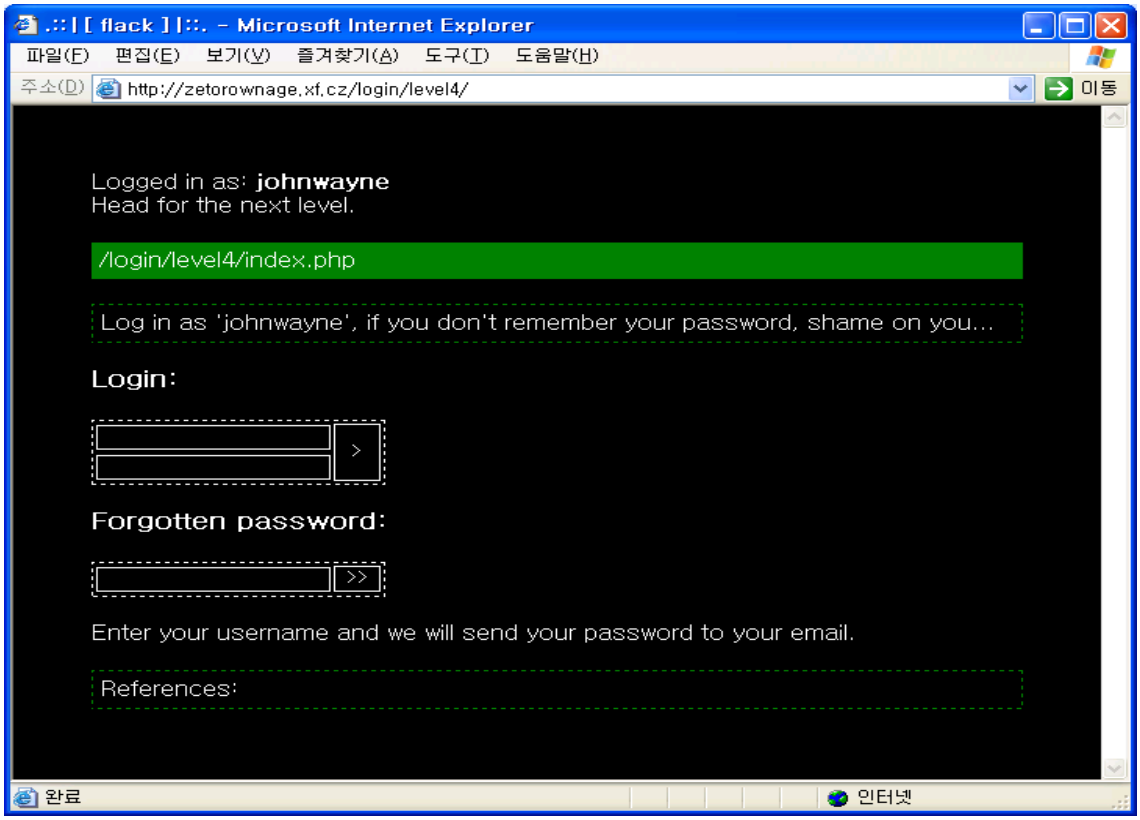
level4는 Blind SQL Injection으로 풀이가 가능한데, GET 또는 POST 방식이라는 것을 제외하면 Other1 문제와 전 풀이과정이 동일하기 때문에 Blind SQL Injection을 위한 Brute Force 코드와 함께 실제 적용 화면을 보여주고, 나머지 상세한 풀이 과정은 Other1로 대체하겠다.

Blind SQL Injection을 위한 Brute Force 소스 코드

http://hkpco.kr/code/blind_brute.c

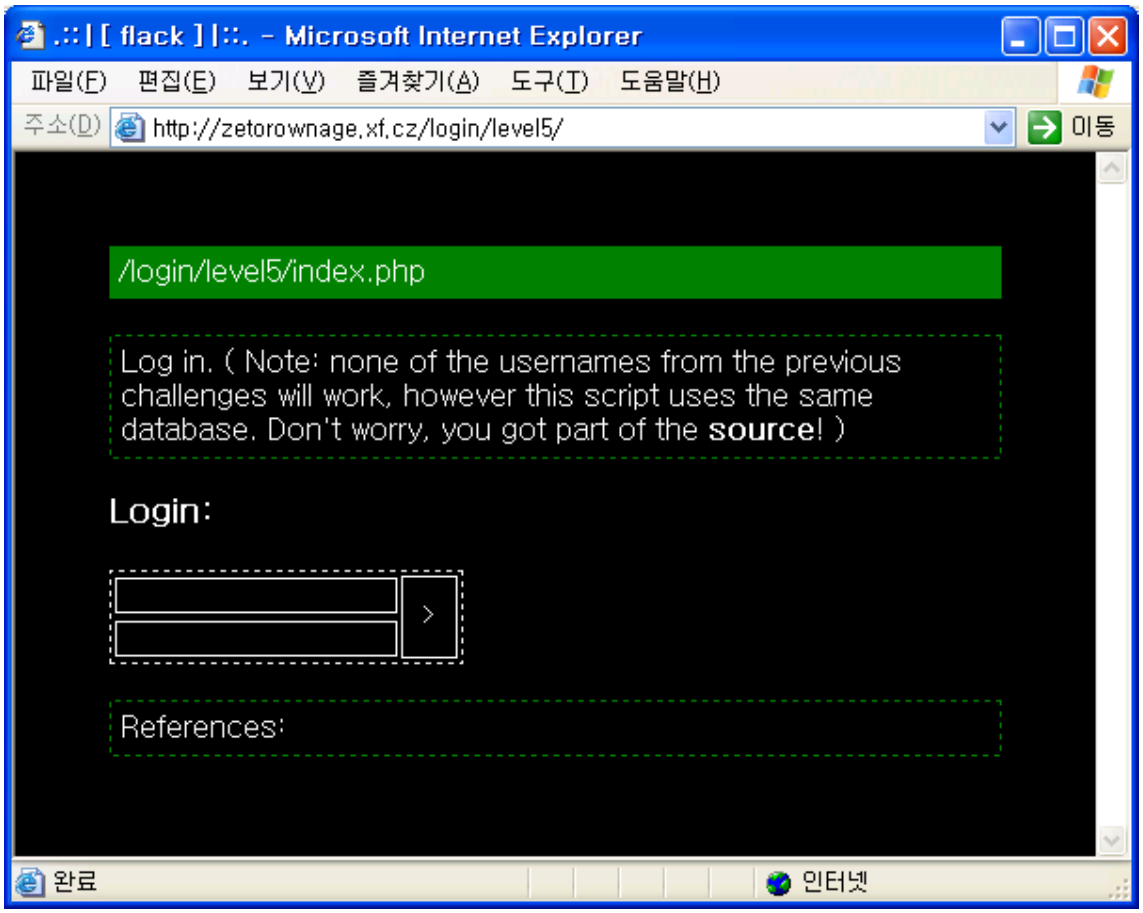
다음은 작성한 프로그램을 이용하여 패스워드를 알아내는 모습이다.

다음은 이렇게 알아낸 패스워드를 통하여 johnwayne 계정으로 로그인에 성공한 모습이다.

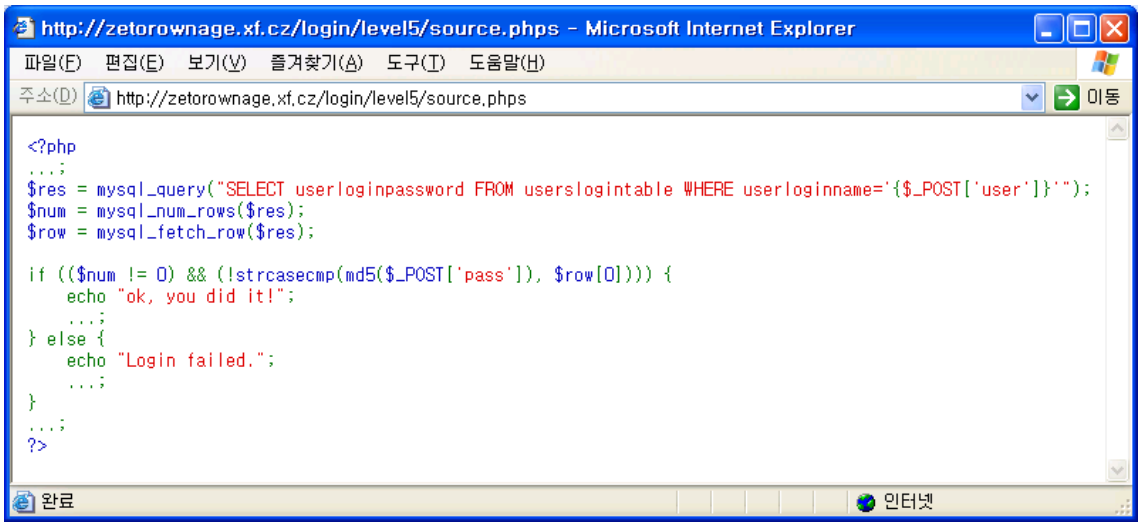


6. Level5

이번 레벨은 주어진 인증 부분의 소스 코드를 보여주고 이를 우회하는 작업을 필요로 한다.



위 문제에서 링크가 걸려있는 source 문자열을 클릭하면 다음과 같은 인증 소스 코드를 볼 수 있다.



다음은 주어진 전체 코드이다.

```
<?php
...;
$res = mysql_query("SELECT userloginpassword FROM userslogintable
WHERE userloginname='{$_POST['user']}'");
$num = mysql_num_rows($res);
$row = mysql_fetch_row($res);

if (($num != 0) && (!strcasecmp(md5($_POST['pass']), $row[0]))) {
    echo "ok, you did it!";
    ...;
} else {
    echo "Login failed.";
    ...;
}
...;
?>
```

간략히 분석해 보면, mysql_num_rows() 함수를 통해 SQL의 결과가 존재하는지를 체크한 뒤 strcasecmp() 함수를 이용하여 pass 입력 폼을 md5()로 암호화한 값과 mysql_fetch_row() 함수가 반환한 배열의 첫 번째 데이터(즉, userloginpassword 필드의 첫 번째 결과 값)와 비교하여 일치하면 로그인 성공을 나타내는 "ok, you did it!" 문자열을 출력해 주는 코드이다.

분석 결과만을 놓고 보면, 우리는 userloginname과 userloginpassword 필드의 값들을 알 수 없으므로 이를 무작정 유추해 내기에는 무리가 있다. 하지만, 첫 번째 필드에서 SQL 인젝션 취약성이 존재하기 때문에 각각의 폼에 다음과 같이 입력하면 우회가 가능하다.

' union select md5('hkpco')#
hkpco

위와 같이 입력하였을 경우 SQL 쿼리는 다음과 같이 예상할 수 있다.

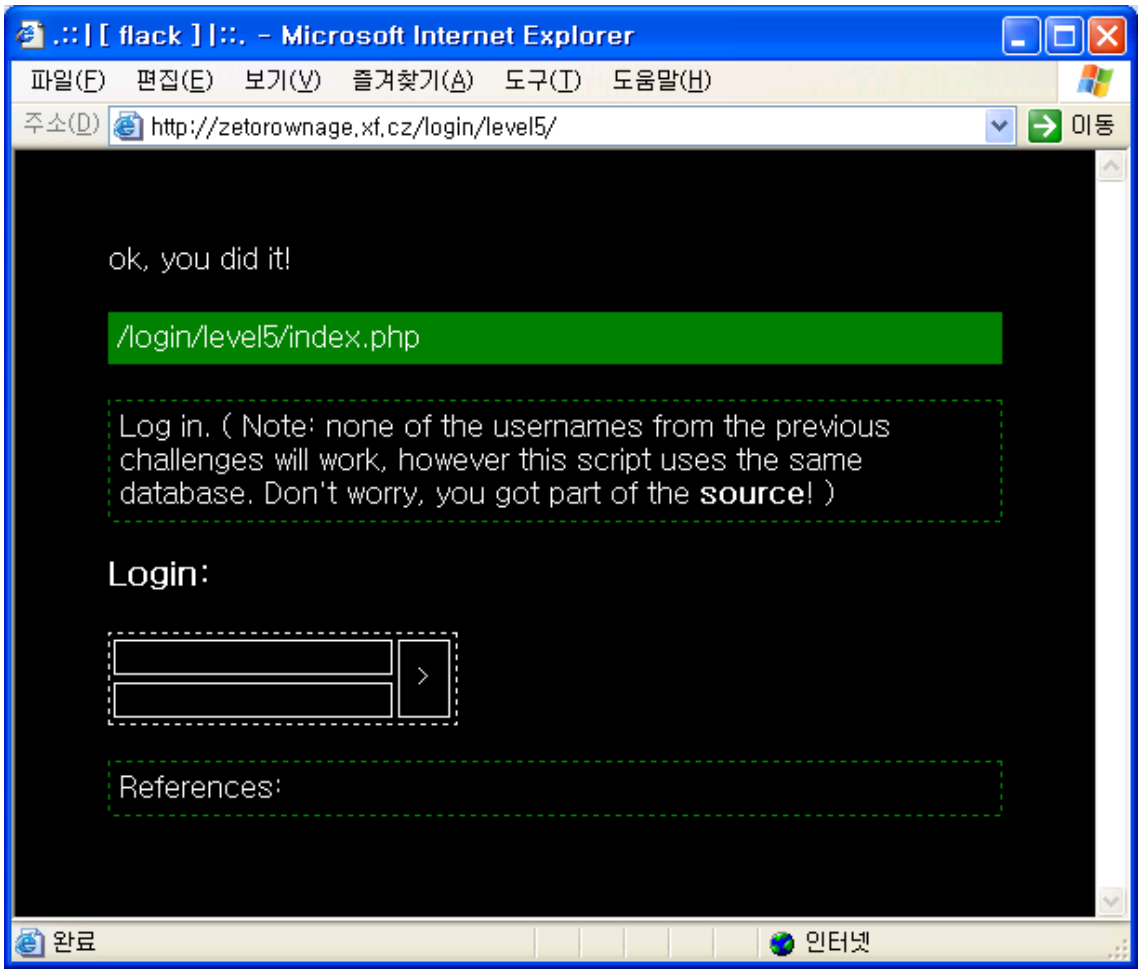
```
SELECT userloginpassword FROM userslogintable WHERE userloginname='' union select
md5('hkpco')#'
```

userloginname이 '', 즉 비어있으므로 원래 쿼리의 결과는 NULL이 되겠지만 union을 이용하여 select md5('hkpco'); 쿼리와 결과를 합친다. 그러면 전체 쿼리의 결과는 hkpco 문자열을

md5로 암호화한 값이 될 것이고 주석문(#)에 의해 이후의 쿼리는 무시될 것이다. 다음 php 코드를 보자.

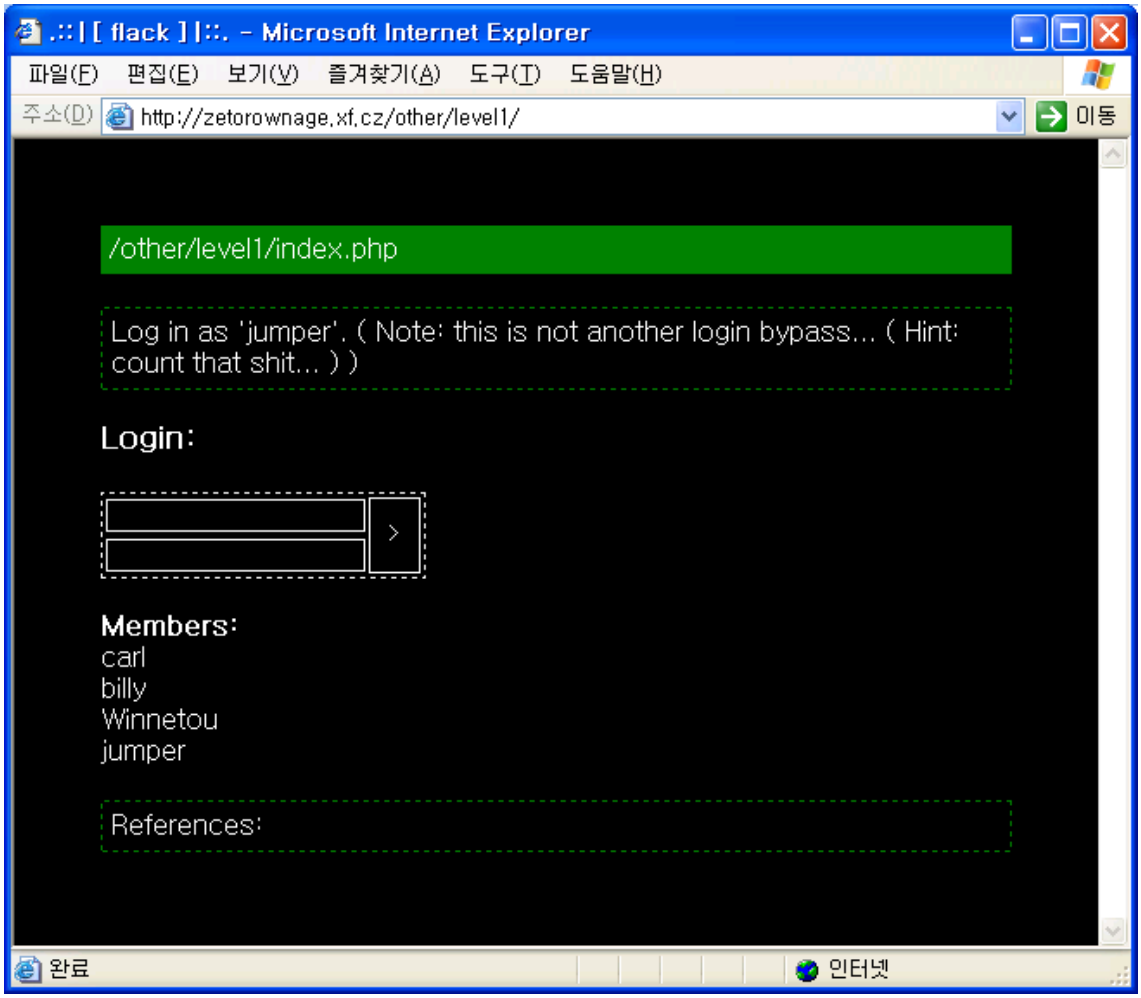
```
$num = mysql_num_rows($res);
$row = mysql_fetch_row($res);
if (($num != 0) && (!strcasecmp(md5($_POST['pass']), $row[0])))
```

첫 번째 폼에 입력한 union 쿼리 결과로 인하여 $num 변수는 1이 될 것이며, 해당 쿼리의 결과인 hkpco 문자열을 md5로 암호화한 값은 $row[0]에 저장되어 있을 것이다. 이 때 두 번째 폼에 입력 된 hkpco 문자열을 md5로 암호화한 값이 strcasecmp() 함수의 첫 번째 인자가 되고, $row[0]에도 역시 hkpco 문자열의 md5 값이 저장되어 있으므로 모든 체크를 우회하게 되어 로그인에 성공할 수 있게된다. 실제 공격코드들 입력한 결과는 다음과 같다.

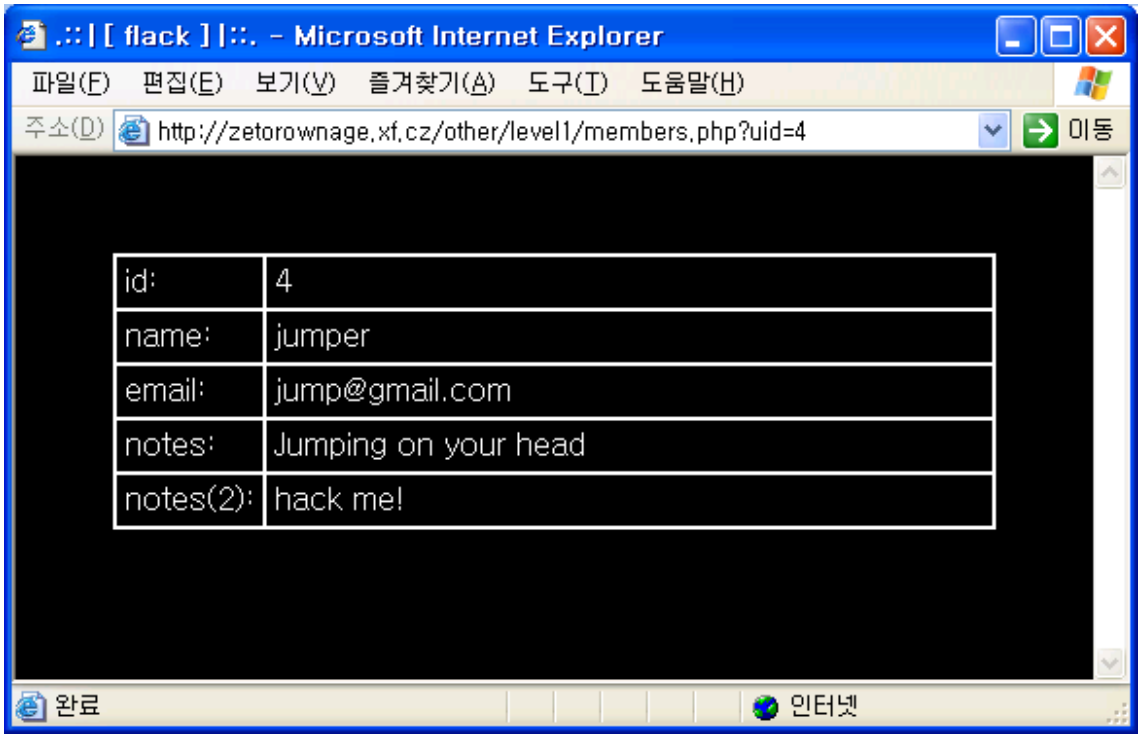


7. Other1

Other 섹션은 모든 문제가 로그인 이외에 다른 페이지 등이 주어진다는 것을 제외하면 Login 섹션과 유사하다. 다음과 같은 로그인 페이지를 볼 수 있다.



jumper 계정으로 로그인 하는 것이 이번 단계의 목표이며, Members: 아래에 사용자 이름으로 보이는 단어들을 클릭하면 다음과 같이 상세한 정보를 볼 수 있다. 우리가 로그인 해야 될 jumper역시 마찬가지이다.



로그인 페이지에서 quote 문자 등을 적용하여 SQL 에러를 유도해 보았지만 발생하지 않았다. 그런데 사용자 정보를 보여주는 members.php의 uid 변수에 quote 문자를 삽입해 보았더니 다음과 같은 에러 메시지를 볼 수 있었다.

URL: http://zetorownage.xf.cz/other/level1/members.php?uid=4'

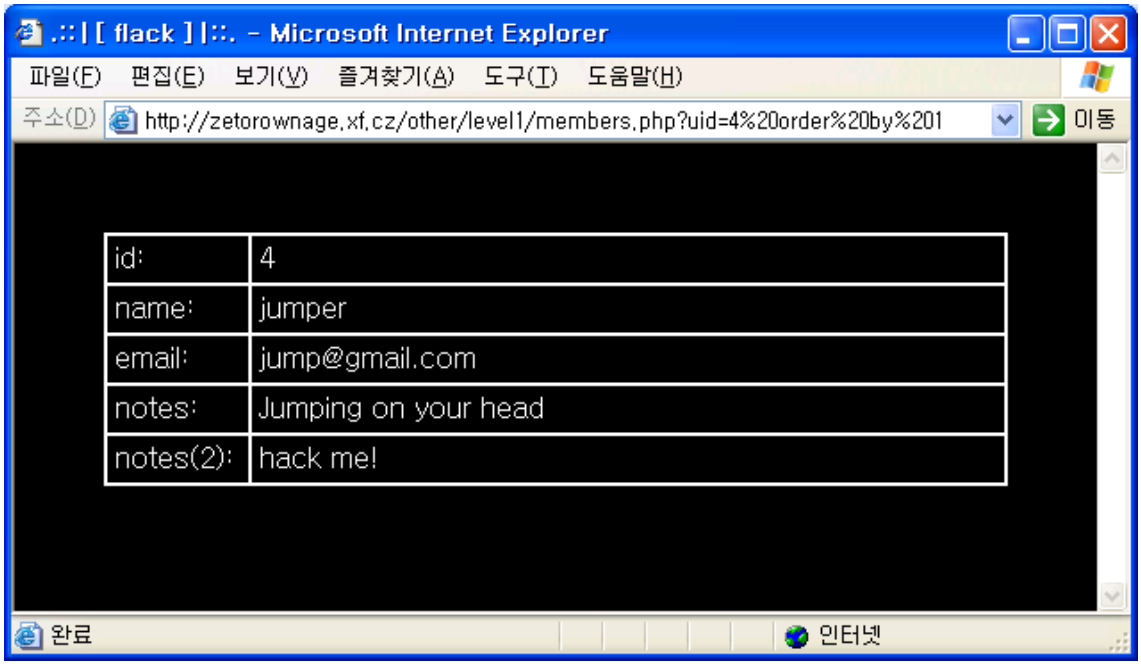
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '₩'' at line 1

우리가 입력한 quote 문자로 인해 SQL 에러가 발생하였지만 필터링에 의하여 앞에 Escape 문자(₩)가 추가되었다. 즉, SQL Injection은 적용되지만 quote 문자의 직접적인 입력을 통한 취약성 공격은 힘든 것처럼 보였다. 그래서 quote 문자를 제외하고 인젝션을 시도하였다.

우선 order by를 통하여 기존 select에서 사용되는 column의 개수를 추측하였다. order by에 대한 예를 잠깐 언급하자면, order by 1을 했을 경우에는 select된 첫 번째 필드 값을 기준으로 정렬하여 출력해주고, order by 2를 했을 경우에는 두 번째 필드 값을 기준으로 정렬하여 출력해준다. 여기서 필드 개수의 추측을 위하여 order by를 사용한 이유는, order by N 에서 N의 크기가 select 된 column의 개수보다 클 경우에는 에러를 발생시키기 때문에 에러 발생 바로 전 N값이 바로 select된 column의 개수가 될 것이기 때문이다.

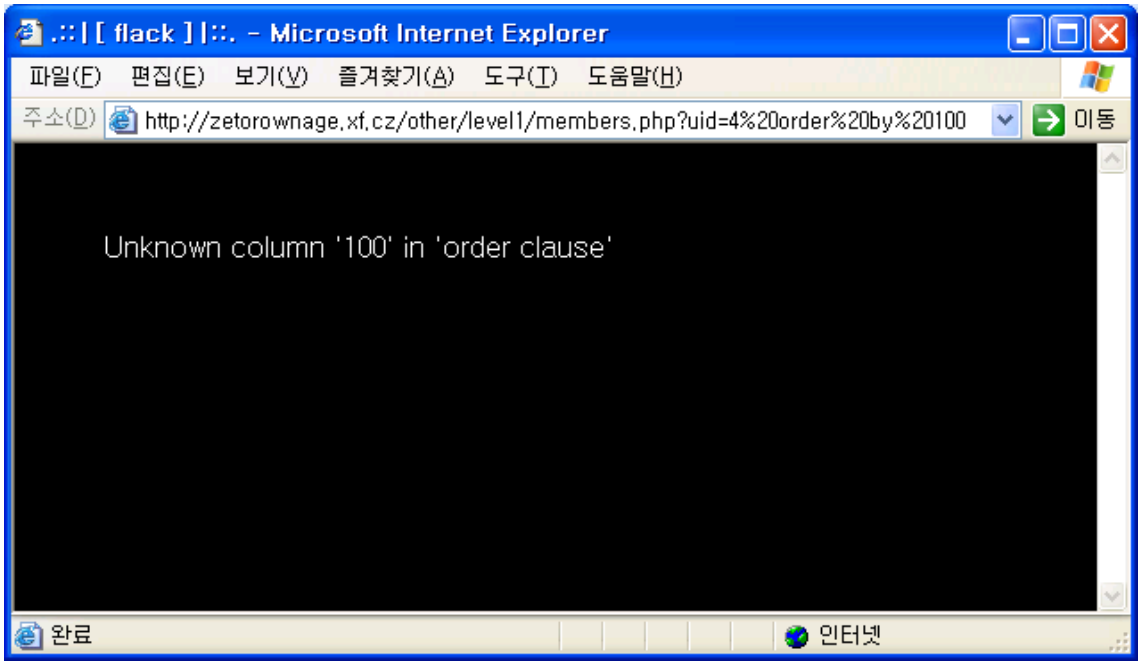
다음은 order by 1을 했을 경우의 결과이다.

URL: http://zetorownage.xf.cz/other/level1/members.php?uid=4 order by 1



다음은 order by 100을 했을 경우의 결과이다. select된 column의 개수가 100개 이하이므로 에러가 발생한다.

URL: http://zetorownage.xf.cz/other/level1/members.php?uid=4 order by 100



이러한 방법으로 구해낸 select된 column의 개수는 6개이다. 지금까지 column의 개수를 구해냈지만, Other1 문제에서 사용되는 테이블 이름이나 컬럼 이름 등을 알지 못하기 때문에 할 수 있는 일이 상당히 한정되어있다. 테이블과 컬럼 이름을 알아내기 위하여 union 구문과 함께 information_schema에 접근해 보았지만 사용할 수 없었다.

이전에 사용했던 order by를 통하여 선택 된 column의 개수뿐만 아니라 이름 또한 알아낼 수 있는데, 이는 간단한 코딩을 통한 무차별 대입 혹은 추측을 필요로 한다. 하지만 사용자 정보를 나타내는 표의 첫 번째 열의 단어인 id, name, email 등이 실제 column 이름과 동일하였으며, 앞서 order by로 알아 내었던 선택된 column의 개수(6개) 보다 출력되는 정보의 개수(5개)가 하나 더 적으므로 나머지 한 개의 column 이름만 찾아내면 되었다. 조금만 생각해보면 나머지 하나는 패스워드에 해당하는 column이라는 것을 추측할 수 있다.

members.php에서 사용중인 SQL 쿼리는 다음과 같이 예상 할 수 있다.

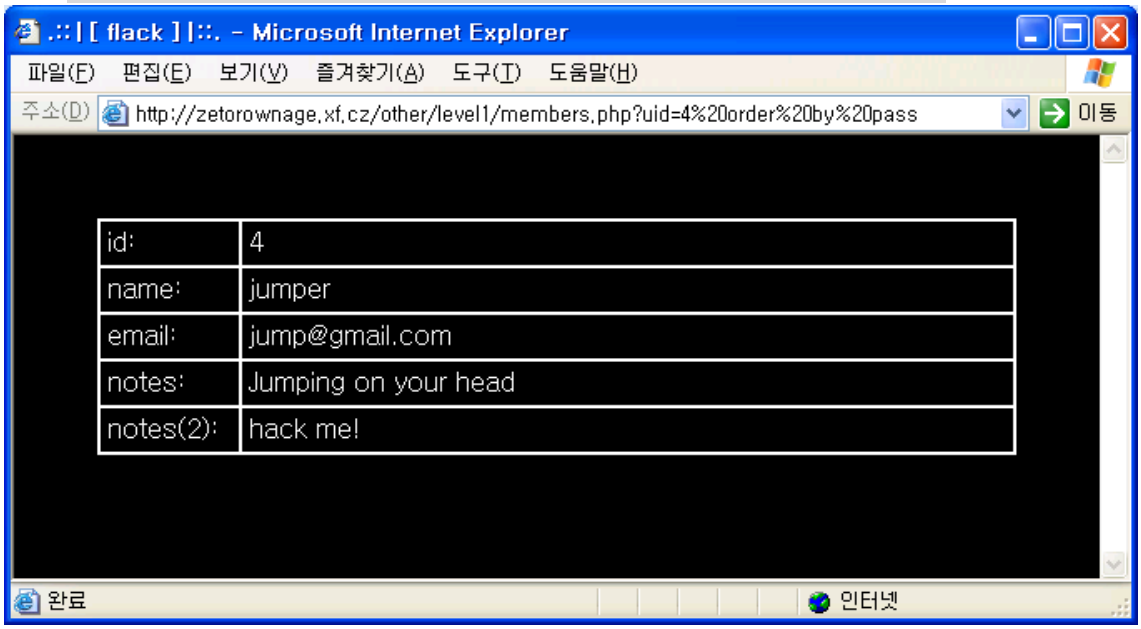
```
select id, name, email, notes, notes2, ??? from members where id='_GET[$uid]';
```

GET 형태의 변수인 $uid에 입력되는 값에 대하여 필터링 하고 있으므로 quote 문자 등을 제외하고 공격해야 하며, 그렇게 되면 굳이 주석문도 필요하지 않게 된다.

column 이름이 존재할 때와 그렇지 않은 경우의 order by 결과는 각각 다음과 같다.

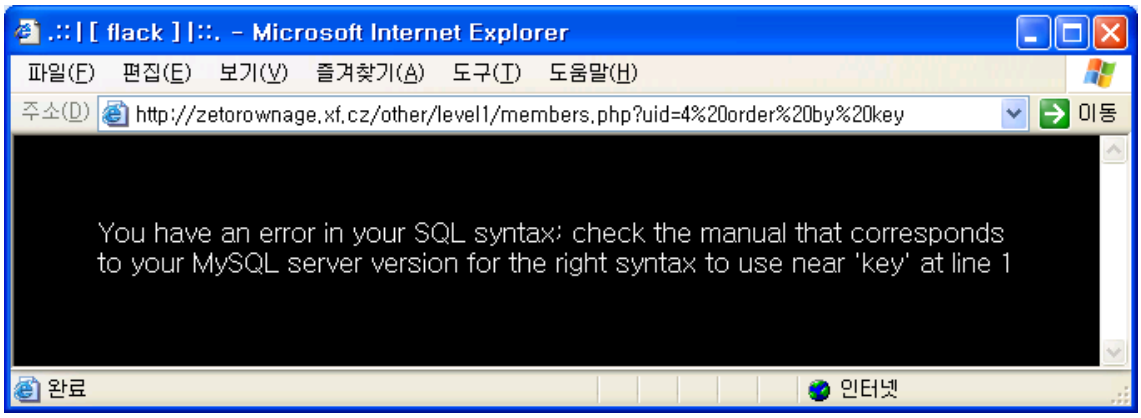
A> column이름이 존재할 경우

URL: http://zetorownage.xf.cz/other/level1/members.php?uid=4 order by pass



B> column 이름이 존재하지 않을 경우

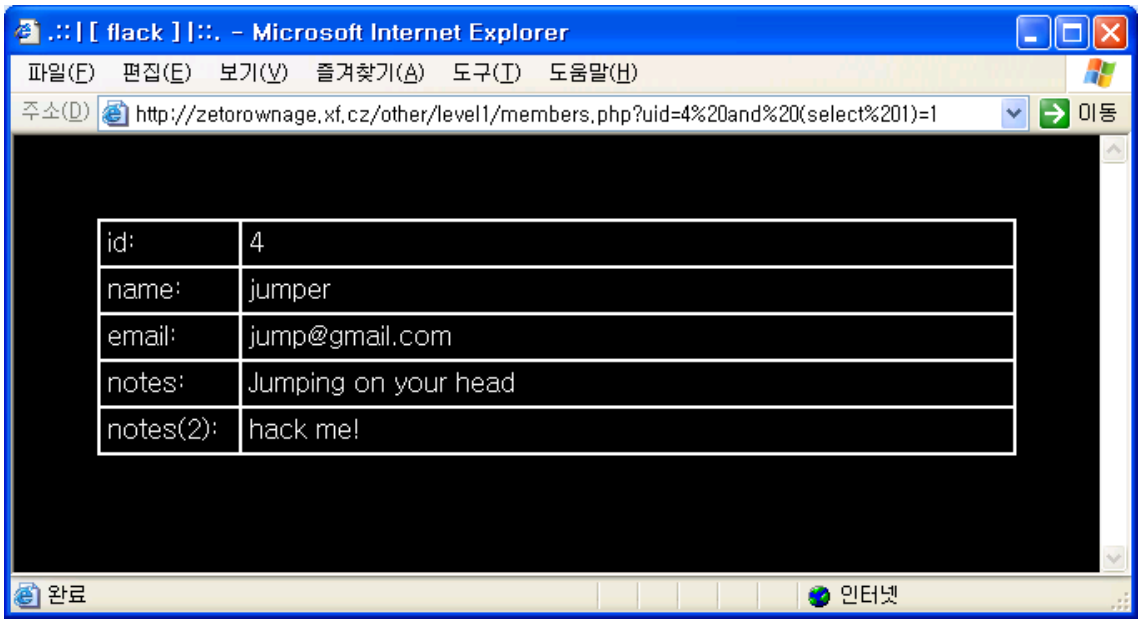
URL: http://zetorownage.xf.cz/other/level1/members.php?uid=4 order by key



패스워드에 대한 column은 위와 같은 방법을 통하여 찾을 수 있었고, 나머지는 사용자 정보가 출력되는 표의 첫 번째 행의 단어들과 모두 일치하기 때문에 모든 column 이름을 쉽게 알아낼 수 있었다. 이제 column 이름을 아는 상태에서 할 수 있는 일을 생각해 보면, pass 필드가 선택은 되었지만 다른 column 정보들과는 달리 출력 되지는 않았기 때문에 이를 알아내기 위한 방법으로 Blind SQL Injection을 생각해 볼 수 있다. 해당 공격이 가능한지에 대한 여부는 다음과 같이 테스트해 볼 수 있다.

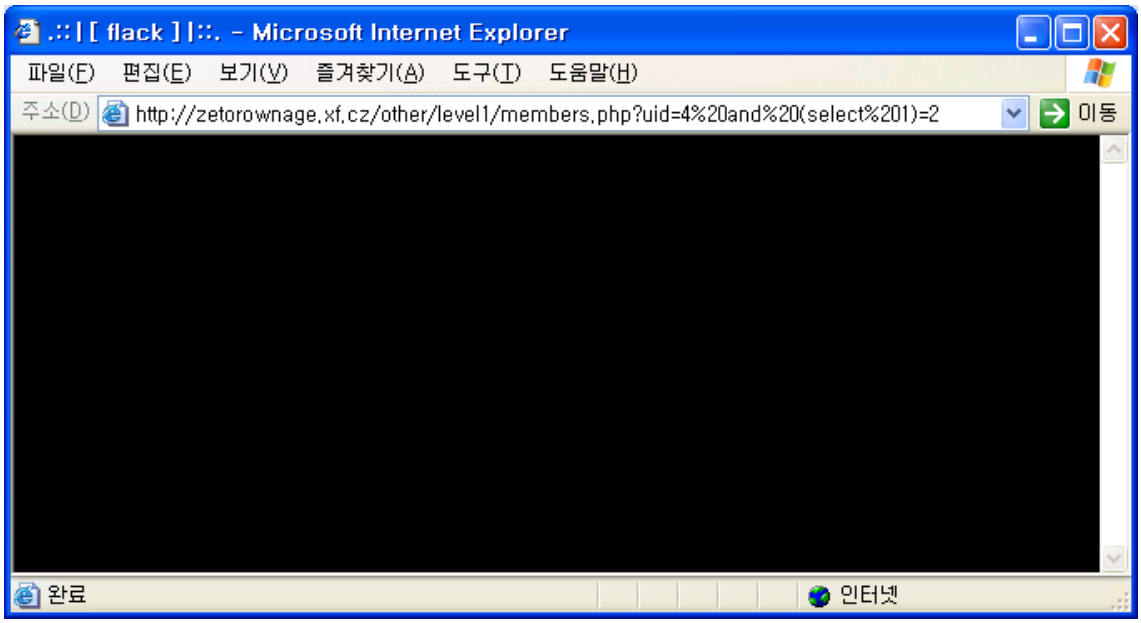
<쿼리가 참인 경우>

URL: http://zetorownage.xf.cz/other/level1/members.php?uid=4 and (select 1)=1



< 쿼리가 거짓인 경우 >

URL: http://zetorownage.xf.cz/other/level1/members.php?uid=4 and (select 1)=2



쿼리가 참인 경우와 거짓인 경우의 SQL 구문은 다음과 같이 유추해 볼 수 있다.

<참인 경우> - id가 4이고 select 1과 1은 같다. 즉, 1=1

```
select id, name, email, notes, notes2, pass from members where id=4 and (select 1)=1;
```

<거짓인 경우> - id가 4이고 select 1과 2는 같다. 즉, 1=2

```
select id, name, email, notes, notes2, pass from members where id=4 and (select 1)=2;
```

쿼리가 참인 경우는 정상적으로 id가 4인 사용자의 정보가 출력될 것이지만, 거짓일 경우에는 어떠한 결과값도 출력되지 않는다. 일반적으로 1=2인 경우는 존재하지 않기 때문이다. 이와 같은 방법을 통하여 pass 필드의 데이터를 알아낼 수 있다. 여기서 pass 필드의 값은 한 번에 알아내기가 힘들기 때문에 다음과 같이 한 문자씩 비교하는 방법을 사용해야 한다.

```
4 and ascii(substring((select pass),1,1))=[ascii 범위를 bruteforce]
4 and ascii(substring((select pass),2,1))=[ascii 범위를 bruteforce]
4 and ascii(substring((select pass),3,1))=[ascii 범위를 bruteforce]
.
.
.
```

계속해서 풀이를 진행하기 전에 ascii(substring((select pass),2,1))=N에 대하여 간단히 설명하자면, 우선 select pass를 통하여 pass필드의 데이터들을 출력해주는데 이 때, substring() 함수를 이용해서 **두 번째** 지점부터 1byte를 추출한다. 그리고 ascii() 함수를 통하여 반환된 아스키 값과 N을 비교하는 역할을 한다.

Blind SQL Injection의 수행 과정은 다음과 같이 나타낼 수 있다.

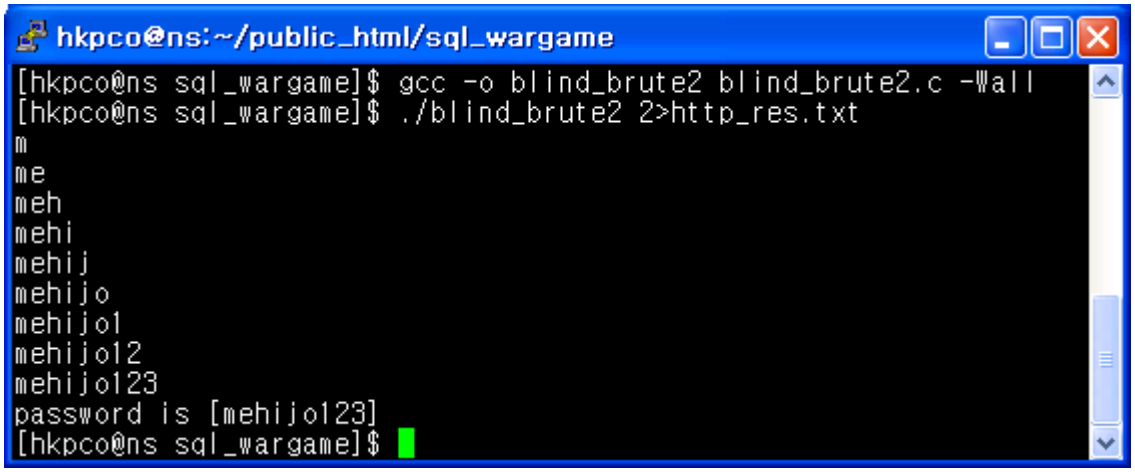
1. select pass 쿼리로 pass 필드(즉, 패스워드)를 반환
2. substring() 함수를 통해 반환된 문자열의 X 위치부터 1바이트를 반환
3. 해당 값을 ascii() 함수를 통하여 아스키 형태로 변환
4. 변환된 값이 Y 값과 일치한다면 X 값 증가 후 2번 과정부터 재 수행
5. 일치하지 않는다면 Y값 증가 후 4번 과정부터 재 수행

위 과정을 자동화 할 수 있도록 코딩하였으며 링크는 다음과 같다. select, thread 등의 다중 프로그래밍을 이용한다면 Brute Force를 훨씬 더 빠르게 만들 수 있을 것이다.

Other1 - Blind SQL Injection Brute Force http://hkpco.kr/code/blind_brute2.c

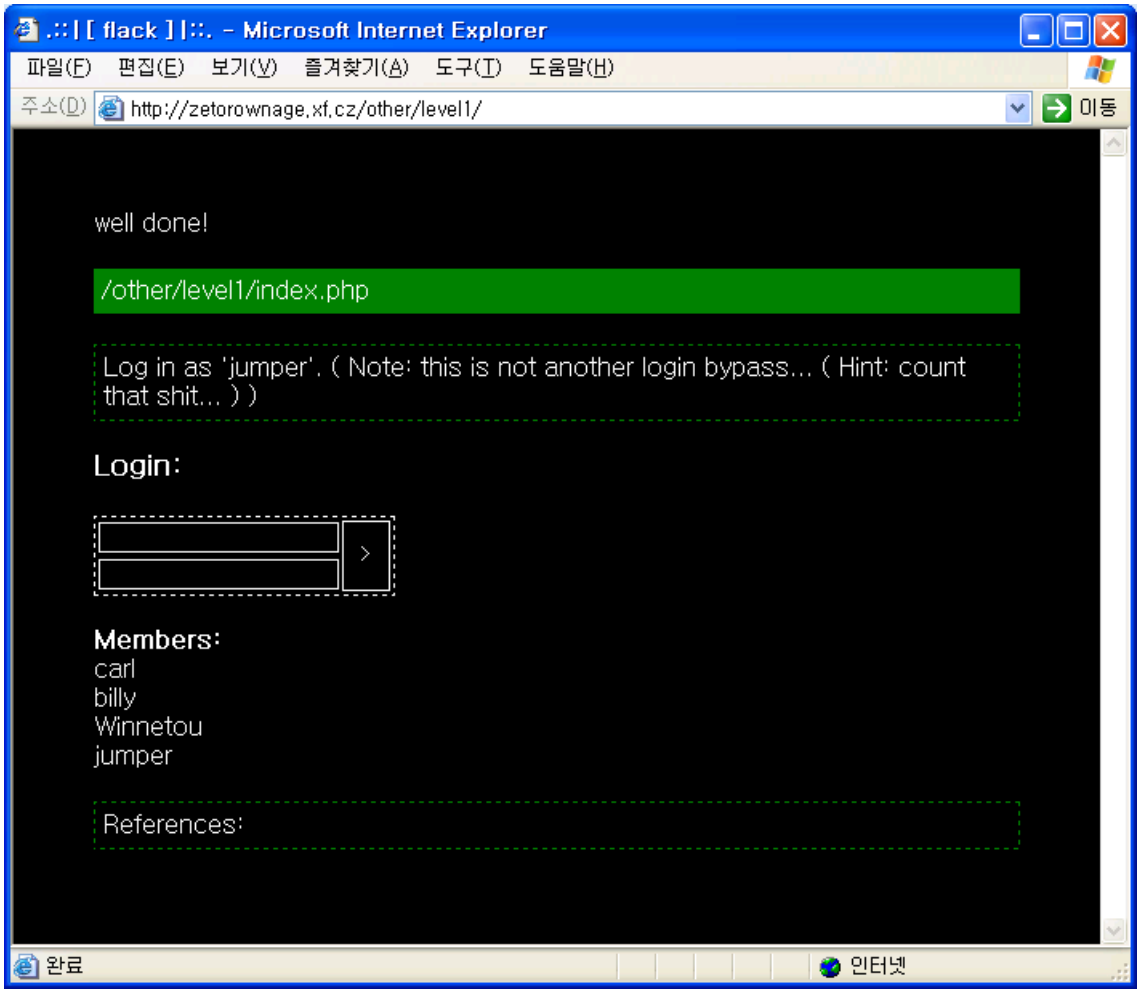
Brute Force는 아스키 전 범위가 아닌 알파벳과 숫자, 몇 가지 문자 등으로 제한하였고, 문자를 비교하는 SQL 쿼리 결과에 "jumper" 문자열이 존재하면 비교한 문자가 일치하는 것으로 가정하고 계속해서 패스워드의 다음 문자를 비교한다. 마지막으로 N+1 위치의 문자와 NULL 문자가 일치할 경우 최종 패스워드를 찾은 것으로 가정하고 프로그램을 종료한다.

다음은 컴파일 후 프로그램을 실행한 화면이다. 어느 정도 기다리면 다음과 같이 최종적인 패스워드를 구할 수 있다.



이렇게 찾은 패스워드를 이용하여 로그인을 하기 위해 각 폼에 다음과 같이 입력하였다.

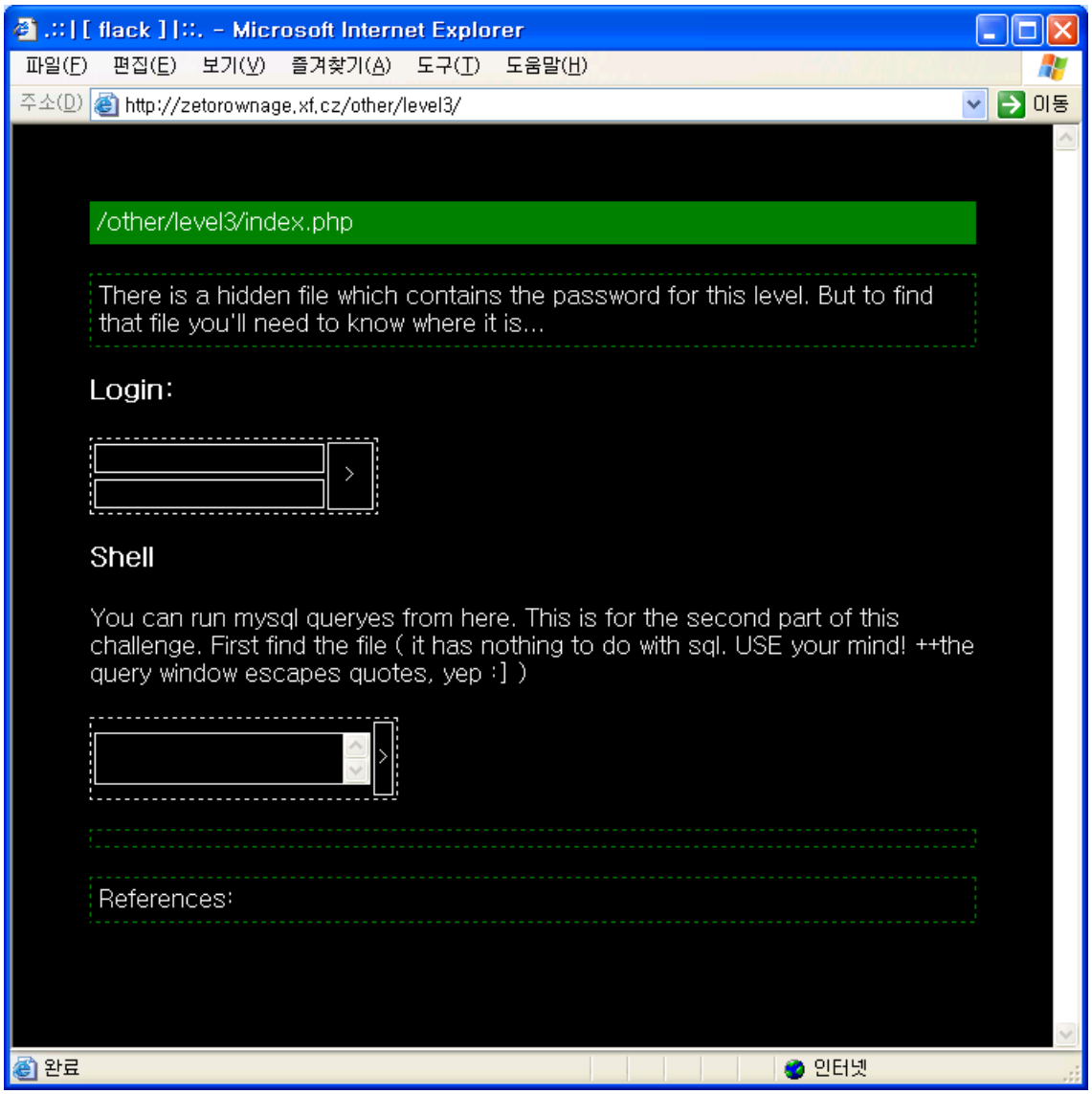
다음은 실제 로그인에 성공한 화면이다.



well done!

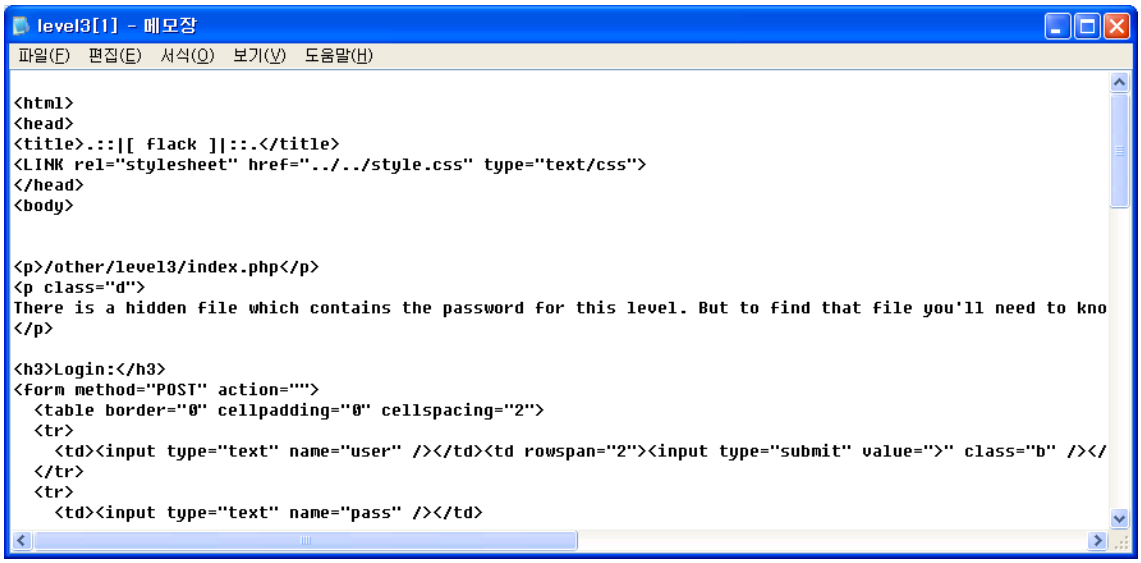
8. Other3

마지막 문제이다. 주어진 페이지는 다음과 같다.

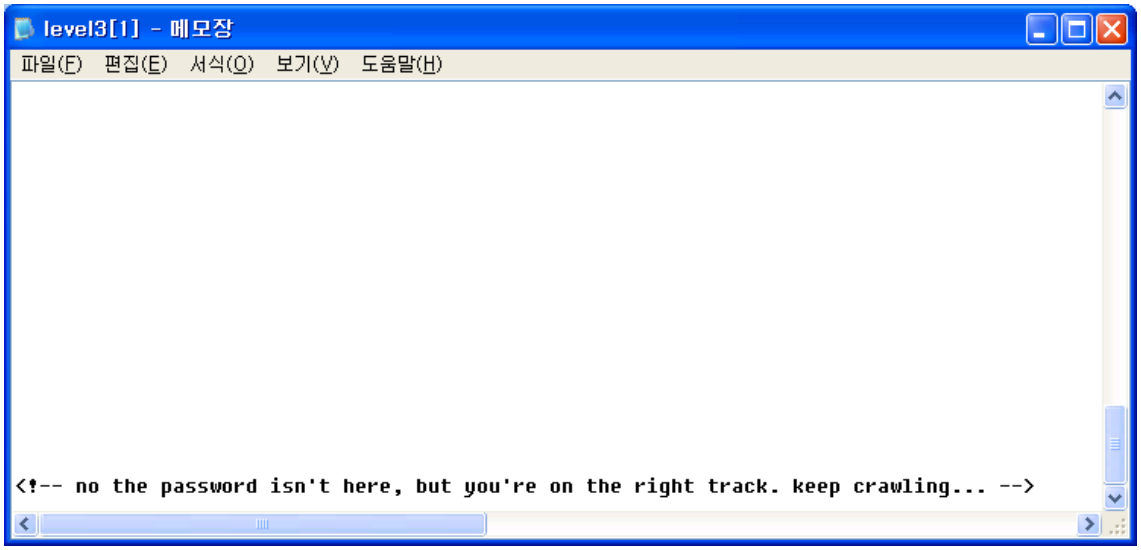


이번 문제를 풀기 위해서는 두 가지 작업을 해야 한다. 첫 번째는, 패스워드가 담긴 파일을 찾는 것이고 두 번째는, mysql 쿼리를 실행할 수 있는 입력 폼을 이용하는 것이다.

첫 번째 작업인 패스워드가 담긴 파일을 찾기 위해서 다음과 같이 문제 페이지를 소스 보기 기능을 통하여 살펴보았다.



소스를 살펴보던 중 가장 아래에 다음과 같은 문구가 보였다.



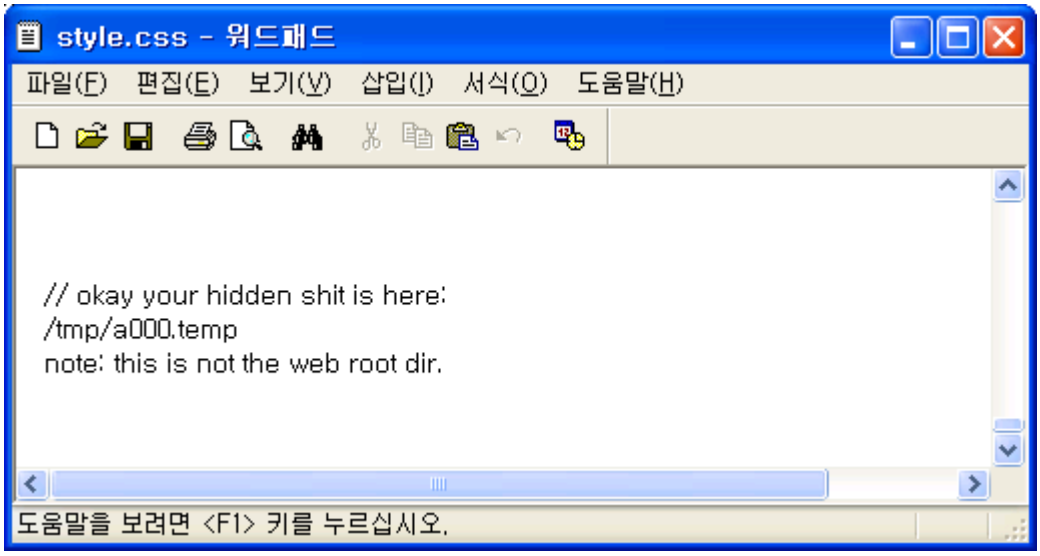
대충 해석해보면, 여기에 패스워드는 없지만 올바르게 접근하고 있다는 뜻이다. 그렇다면 여기서 또 다른 실마리를 찾을 수 있을 것이라 예상하고 살펴보던 도중 다음과 같은 코드가 눈에 띄었다.

```
<html>
<head>
<title>.::|[ flack ]|::.</title>
<LINK rel="stylesheet" href="../../style.css" type="text/css">
</head>
```

직접적으로 보이는 HTML 코드에는 특별한 것이 없었기 때문에 간접적으로 링크된 파일인 style.css를 의심하였으며, 아래 주소에서 받을 수 있었다.

http://zetorownage.xf.cz/other/level3/../../style.css

style.css를 살펴보면 최 하단에 다음과 같은 문구가 기록되어있는 것을 볼 수 있다.



읽어보면 패스워드는 /tmp/a000.temp에 저장되어 있는 것을 알 수 있으며, 지금까지를 첫 번째 단계로 가정하였다.

이제 두 번째 단계인 mysql 쿼리를 실행할 수 있는 폼을 이용하여 이 패스워드 파일을 열람해야 하는 것으로 보였다. 문제 페이지에도 언급되어 있다시피 쿼리에서 quote 문자를 필터링 하고 있으며, 살펴본 결과 사용할 수 있는 쿼리도 제한적이었다.

시험 삼아 쿼리를 요청할 수 있는 폼에 다음과 같이 입력해 보았다.

결과는 다음과 같았고, 이를 통해 쿼리가 정상적으로 수행되는 것을 알 수 있다.

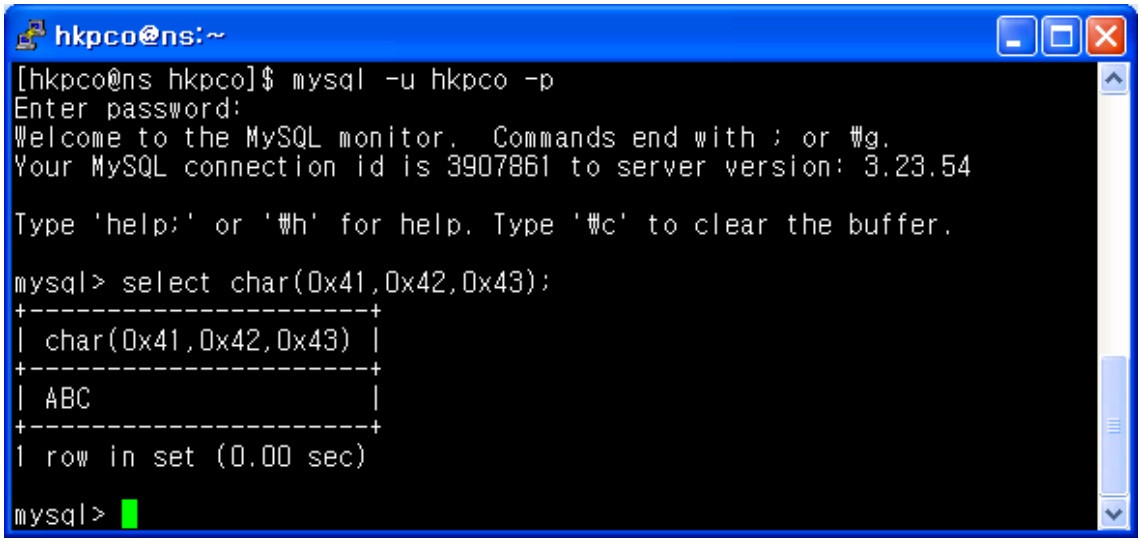
현재 주어진 것은 패스워드가 담긴 파일인 /tmp/a000.temp, 그리고 제한적이나마 SQL 쿼리를 수행할 수 있는 입력 폼이 있다. 위에서도 언급하였지만 이 입력 폼을 이용하여 파일 내용을 열람해야 하는 것으로 보이며, 우리는 여기서 LOAD_FILE() 함수를 생각해 볼 수 있다.

다음과 같이 사용하면 파일 내용의 열람이 가능할 것이다.

```
select LOAD_FILE( '/tmp/a000.temp' );
```

하지만 문제에서는 quote 문자를 필터링 하기 때문에 해당 쿼리는 정상적으로 실행되지 않는다. 이를 우회할 수 있는 방법을 찾아야 하는데, char() 함수를 이용하였다.

다음은 char() 함수를 사용한 예시이다. 아스키에 해당하는 숫자를 각각 문자로 바꾸어준다.



이제 char() 함수를 사용하기 위해서 "/tmp/a000.temp" 문자열을 아스키 숫자로 바꾸어 주어야 하는데, 작업이 번거롭다면 다음과 같이 간단한 코딩으로 해결할 수 있다. 변환을 수행해주는 코드는 다음과 같다.

conv.c

```
int main( int argc , char **argv )
{
        int i = 0;

        if( argc < 2 )
                return 0;
```

```
        for( ; i < strlen(argv[1]) ; i++ )
        {
                if( i == 0 )
                        printf( "0x%x" , argv[1][i] );

                else
                        printf( ",0x%x" , argv[1][i] );
        }

        printf("₩n");
}
```

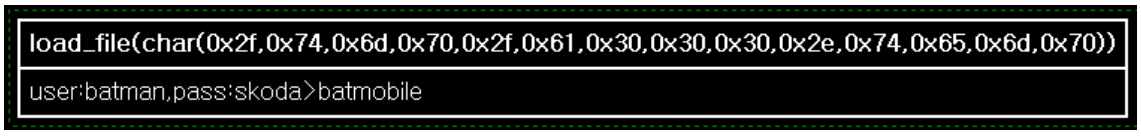
이제 간단히 /tmp/a000.temp 문자열을 char() 함수 인자형식에 맞게 변환할 수 있다.

```
[hkpco@ns sql_wargame]$ ./conv /tmp/a000.temp
0x2f,0x74,0x6d,0x70,0x2f,0x61,0x30,0x30,0x30,0x2e,0x74,0x65,0x6d,0x70
```

패스워드 파일을 열람하기 위한 최종적인 SQL 쿼리는 다음과 같다.

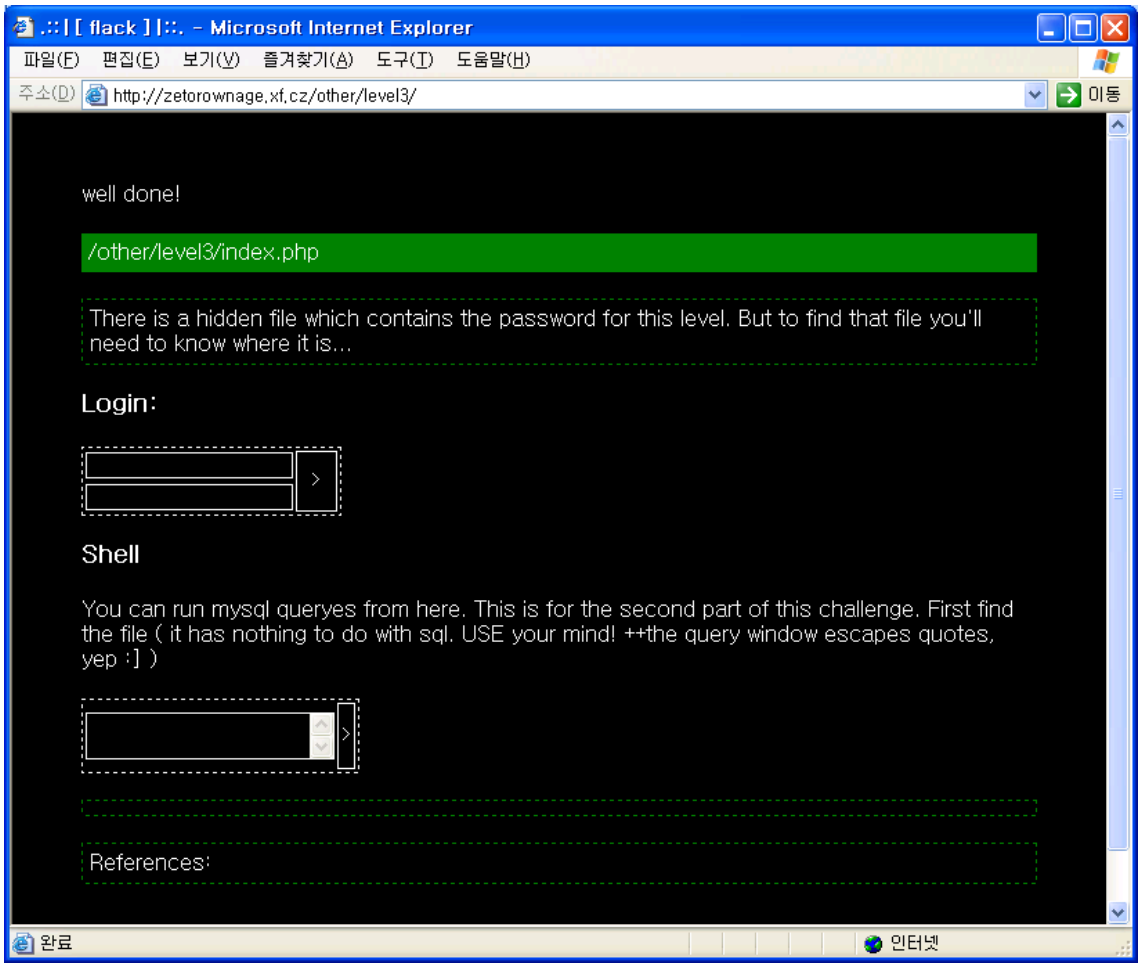
```
select load_file(char(0x2f,0x74,0x6d,0x70,0x2f,0x61,0x30,0x30,0x30,0x2e,0x74,0x65,0x6d,0x70));
```

해당 쿼리를 두 번째 폼에 입력하면 SQL 쿼리를 실행시킨 결과가 출력되며, 그 결과가 바로 /tmp/a000.temp 파일의 내용이다.



user는 **batman**, pass는 **skoda>batmobile**인 것을 알 수 있다. 최종 로그인을 위하여 각각 다음과 같이 입력하였다.

해당 계정으로 로그인을 시도하면 성공을 뜻하는 "well done!" 메시지를 볼 수 있다.



9. 후기

예전부터 국내외 많은 워게임을 접해보았지만 그에 대한 풀이를 작성하는 것은 이번이 처음이었고, 요즘은 워게임이라는 것 자체를 거의 접하지 않아서인지 감회가 새로웠다. 한때 국내에 수많은 워게임들이 생겨나고 사라지던 것이 반복되던 시기가 있었는데 현재는 대부분 없어지고 그 중 일부만이 유지되고 있다는 생각이 들어 안타깝기도 했다. 보다 더 많은 사람들이 직간접적으로 경험할 수 있는 유익하고 재미있는 워게임들이 계속 생겨나기를 바란다.

마지막으로 한마디 더 하자면, 웹 해킹은 대부분 결과만을 놓고 따지면 특별히 어렵거나 복잡한 테크닉들이 필요하지는 않지만 그 결과의 도출을 위한 발상이 힘들 때가 많다. 그런데 몇몇 사람들은 기존의 기술을 적용한 것뿐만 아니라 새로운 기술마저도 이러한 단편적인 생각 때문에 웹해킹은 쉽다라는 착각을 하기도 한다. 콜롬버스의 달걀이라고나 할까.