|  |  |
| --- | --- |
| Logo  Description automatically generated | Challenge Writeup  Digital Art Gallery  by Kouretas Panagiotis |

# Main Concept

# Unveil your artistic masterpieces at the Digital Art Gallery, showcasing your talent in a world of pixels and palettes. Note: For the gallery's security, we've curated the canvas by limiting certain file extensions.

# Exploitation

First of all, we deploy the challenge from CTFLib.

Then we are redirected at <http://localhost/index.php> which is the home page of the challenge as seen in the Figure 3.1. This is a “Digital Art Gallery” website.

A screenshot of a website

Description automatically generated

Figure 3 – Challenge’s Home Page / Digital Art Gallery.

When we click on the “Upload Your Art” button, we are transferred to <http://localhost/upload_art.php> as depicted in Figure 3.2. This page as indicated from the URL, appears to be a place we can upload our digital art pieces. Also, the URL unveils that this page is constructed using “PHP”.

A screenshot of a computer

Description automatically generated

Figure 3 - Upload Art page.

Now let's try testing the upload art functionality of this page by uploading a simple image. After our image “obiwan.jpeg” is uploaded successfully as seen in Figure 3.3, we notice that we can access it at <http://localhost/uploads/obiwan.jpeg>.

A screenshot of a computer

Description automatically generated

Figure 3 - Upload Success message.

However, if we try to access the <http://localhost/uploads/> directory we get the following error as it's depicted in Figure 3.4.

A screenshot of a computer

Description automatically generated

Figure 3 - Page Access error message.

Reading the description of the challenge one more time, I noticed that there is a note that indicates there are certain file extensions limited for the gallery's security.  
“Note: For the gallery's security, we've curated the canvas by limiting certain file extensions.”

After this, I tried to upload a PHP file including a simple “<?php phpinfo(); ?>” command as seen in Figure 3.5. This command is used to output a large amount of information about your PHP installation and can be used to identify installation and configuration problems.

A screen shot of a computer

Description automatically generated

Figure 3 - Simple PHP script.

The reason I tried to upload such a simple PHP file is to test how the application responds to the different file extensions.

As expected, I got the error “This file type is not allowed” as depicted in Figure 3.6.

A close up of a sign

Description automatically generated

Figure 3 - File type is not allowed error message.

After trying many different PHP web shells I found online, I still couldn’t find a way to bypass this file extension limit.

However, while I was trying to figure out how this application works and checking the network traffic from the browser developer tools (by pressing the “F12” button on the browser), I distinguished a clue that could help me with my problem. The server used for this website was “Apache”.

A screenshot of a computer

Description automatically generated

Figure 3 – Server used in browser developer tools.

Also, while I was searching about file upload vulnerabilities, I found this snippet of an article on “Portswigger” that was indicating a possible trick to bypass blacklisted extensions via file upload when Apache server is used as seen in Figure 3.8.

A screenshot of a computer

Description automatically generated

Figure 3 – Bypass blacklisted extension article on Portswigger.

This article is suggesting to upload an “.htaccess” file which is a configuration file used by Apache-based web servers. This file will contain the “AddType application/x-httpd-php .art” command that makes the application understand files with “.art” extension as files with “.php” extension and then upload a PHP web shell.

A black screen with white text

Description automatically generated

Figure 3 - Contents of .htaccess configuration file.

Since I uploaded this “.htaccess”, the Apache configuration must have been updated.

A close up of a sign

Description automatically generated

Figure 3 - .htaccess file successfully uploaded.

Right after that, I uploaded the new PHP web shell but this time as a “webshell.art” file.

A close up of a sign

Description automatically generated

Figure 3 - webshell.art successfully uploaded.

Following some online research about which PHP web shell should I use in my situation, I found this online GitHub repository <https://github.com/WhiteWinterWolf/wwwolf-php-webshell/blob/master/webshell.php> that seems to have the best one. This web shell allows an attacker to run commands on a server with web hosting capabilities. It's designed to provide unauthorized access to the server's filesystem and execute commands remotely.

So, we head to the <http://localhost/uploads/webshell.art> page. There we select “/var/www/html/uploads” as the “CWD” which is the current working directory that we want to execute commands. For the “Cmd” which is the command we want to execute we will choose “ls” which lists the files of the chosen directory.

As depicted in Figure 3.12, there are listed all the files we uploaded in the gallery.

A screenshot of a computer

Description automatically generated

Figure 3 – Uploaded gallery files listed.

Now let's list all the files that exist in the root directory. As seen in Figure 3.13, there is a “flag.txt” which is most possibly to be the file that contains the hidden flag.

A screenshot of a computer

Description automatically generated

Figure 3 – Root files listed.

We execute the “cat /flag.txt” command to view the contents of the file “flag.txt” as depicted in Figure 3.14.

A screenshot of a computer

Description automatically generated

Figure 3 - Viewing flag.txt content.

ALRIGHT! We successfully retrieved the hidden FLAG.

CTFLIB{Manipul8\_Ext3ns!0n\_M1ss10n\_Acc0mpl!sh3d}