CEG 4430/6430 Cyber Network Security

Project 3

(20 Points)

Through this project, students will learn **unrestricted file uploading vulnerability**, launching attacks against this vulnerability, and perform mitigation (graduate students).

Submission

1. Undergraduate:
   a. A report of your answers.
   b. Each team submits one report.
   c. Each team member needs to submit a list of all team members.

Questions

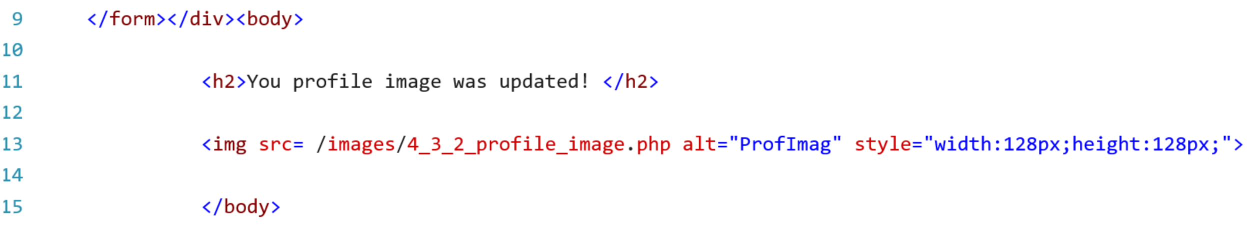
Assume that you are an attacker and you have access to the source code of the PHP file (you do not have access to the server).

1. What is the directory (or the path) in the server that stores the uploaded image? (1 points) and Justify your answer by analyzing the source code. (2 points)

Answer Question 1: The directory (the path) in the server that stores the uploaded image is under:

src= /images/4_3_2_profile_image.php

As shown in the snip of source code below.



```
</form></div><body>

          <h2>You profile image was updated! </h2>

          <img src= /images/4_3_2_profile_image.php alt="ProfImag" style="width:128px;height:128px;">

          </body>
```

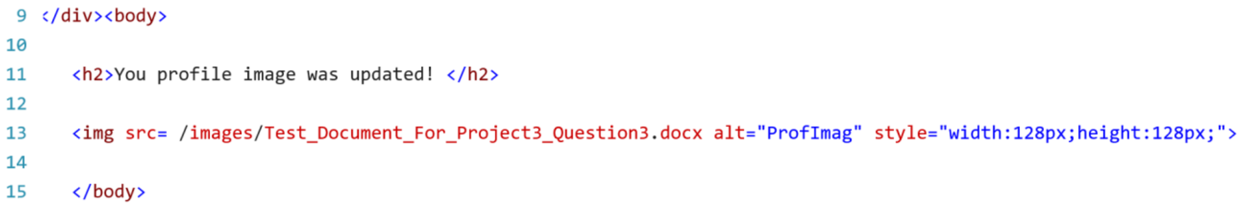
2. Suppose the uploaded image has a name called “myphoto.jpg”. How can you directly access this image in your browser? Show the path. (3 points)

Answer Question 2: file:///C:/xampp/htdocs/images/myphoto.jpg

3. Can you upload files that are not images to the server and directly get access to uploaded files? Justify your answer using both testing (2 points) and code analysis. (2 points)

Answer Question 3: You are able to upload and access other documents aside from images. I uploaded a document named “Test_Document_For_Project3_Question3” This document was able to be uploaded and the file path to it is as follows: “file:///C:?xampp/htdocs/images/Test_Document_For_Project3_Question3” Here is an image of the source code to see the path to the file as well.

Image of code for the file path:



```
</div><body>

    <h2>You profile image was updated! </h2>

    <img src= /images/Test_Document_For_Project3_Question3.docx alt="ProfImag" style="width:128px;height:128px;">

    </body>
```

4. Create a PHP file and upload it to the server. This PHP file will allow the attacker to remotely execute arbitrary commands in the server. (hint: see "system()" API in PHP). (5 points)

Answer Question 4:

This is the relevant PHP code: // Define the target location where the picture being

```
// uploaded is going to be saved.

$target = "pictures/" . basename($_FILES['uploadedfile']['name']);


// Move the uploaded file to the new location.

if(move_uploaded_file($_FILES['uploadedfile']['tmp_name'], $target))

{

echo "The picture has been successfully uploaded.";

}

else

{

echo "There was an error uploading the picture, please try again.";

}
```

To follow up on this question, a file created named "mynewfile.php" contains a system() call known as system($_GET['cmd']); This single line of code is what is important. Once the file is uploaded an attacker can simply go to the file that has been uploaded to the server and run any number of arbitrary commands by using a URL to connect to the file.

5. Mitigation (5 points)
    a. **Undergraduates**: Briefly but precisely explain how you can expand the sanitization function in the PHP script to mitigate this vulnerability.

Answer Question 5a: We can expand on the sanitation function of the PHP script by reducing the file size that can be uploaded as well as checking the file type. This will help to test that the file being uploaded is both within the set limited size range and is the type of file being asked for.

b. **Graduates**: Expand the sanitization function in the PHP script to mitigate this vulnerability. You need to submit the patched “profile_image.php”.

Answer Question 5b:

We will have an updated profile_image.php file uploaded with this document.

References

It may be helpful for the project to read the following articles.

1. https://cwe.mitre.org/data/definitions/434.html
2. PHP $_FILES: http://php.net/manual/en/reserved.variables.files.php