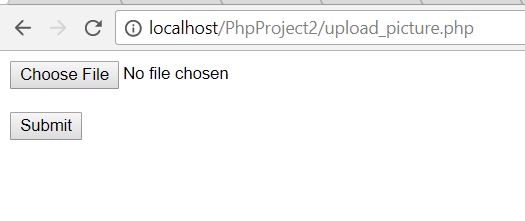
**Unrestricted Upload of file with Dangerous Type**

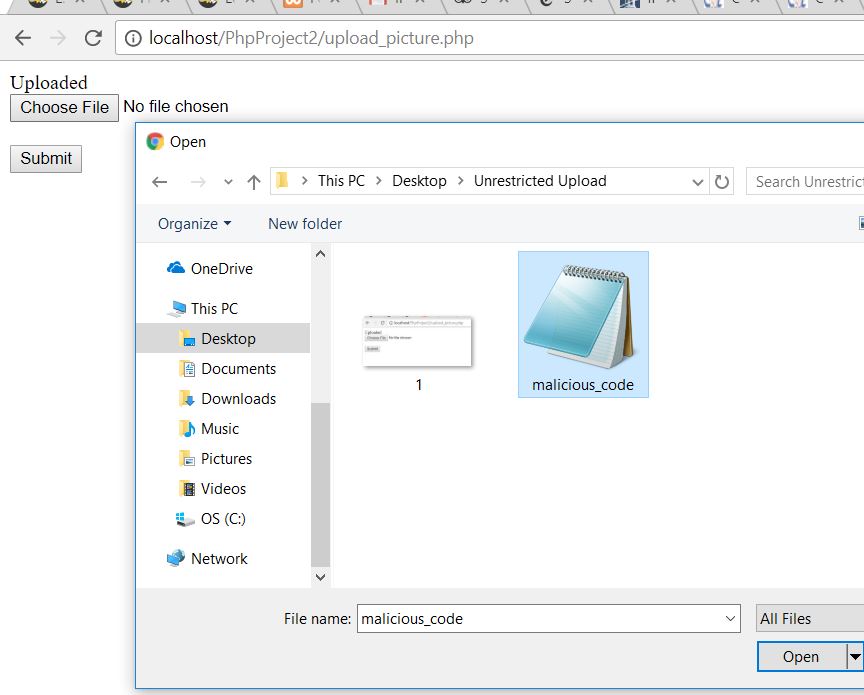
I will demonstrate a unique example of insecure interaction between components. I have chosen Unrestricted Upload of file with Dangerous Type. This is only part of what would be larger application and is for demonstration purposes.

We will see how an application can be vulnerable through unrestricted uploads of files. For example, without checking the file type, a user could upload a harmful ‘.php’ file that could be executed on the server. We will look at a part of a program that asks the user to upload an image. This program does not check the file type, so the user can upload any kind of file.

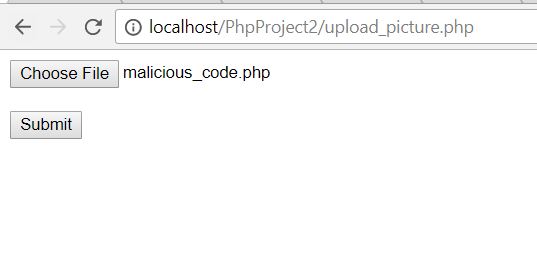
1. User is at page asking to upload a picture.



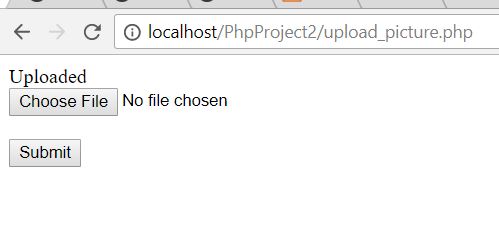
2. Select file to upload. Let’s pretend there is a malicious script in this file.



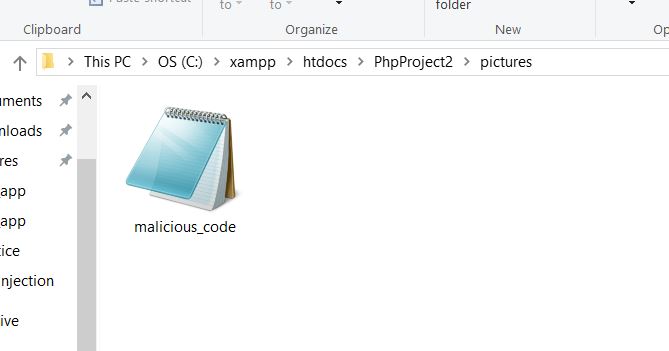
3. File selected.



4. Submit. Uploaded.



5. Confirm file upload successfully to web server directory.



As you can see there was no verification of file type, so we have a weak point in our application that we need to mitigate. Since the file ends in “.php “ it can be executed by the web server. Now let’s try to use the function getimagesize() to see if the file is an image or not. If it is not, it will return false. In the case that it is an image, we will allow for it to be uploaded to the web server. There are many different formats for images, but for this example we will only allow three: .jpg, .png, or .gif.

case "image/jpeg" ||"image/gif" || "image/png":

$location = 'pictures/';

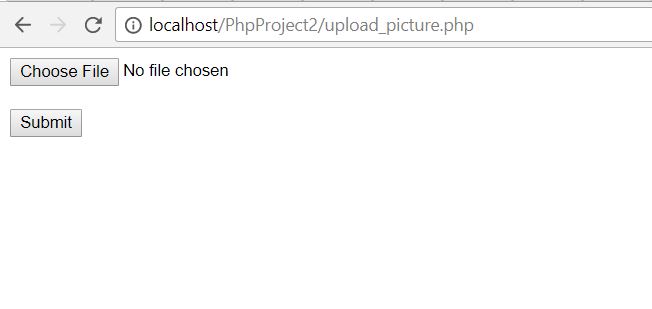
move\_uploaded\_file($temp\_file, $location.$name);

echo 'Uploaded successfully.';

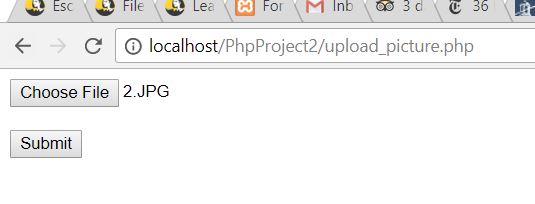
What our application is saying is that if the image is in .jpg OR .gif OR .png format, go ahead and upload it, if not, give the user an error message.

Let’s try to upload a .jpg image, with our new mitigated code in place, and then another file type that is not allowed.

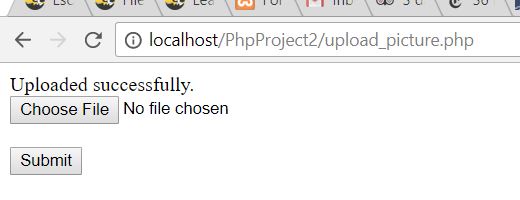
1. We will start back at the beginning.



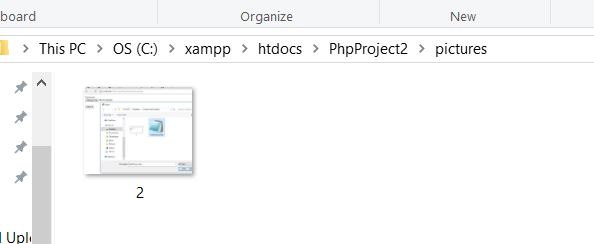
2. We have selected a .jpg image



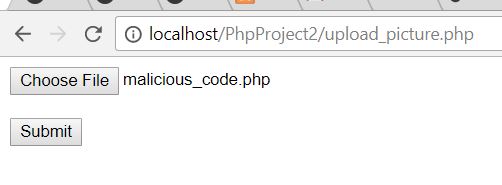
3. We have hit the submit button and shown that the file uploaded successfully.



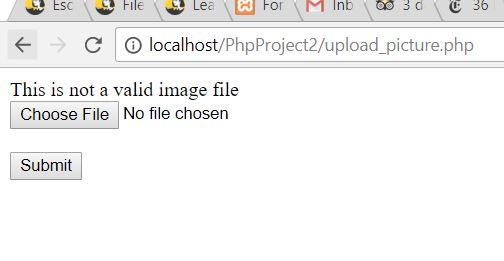
4. Shows that the image uploaded successfully in the correct directory.



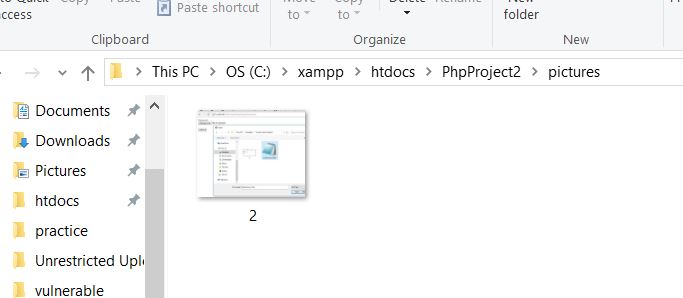
5. Now we have chosen to upload a file with a “.php” extension. The file is named malicious\_code.php.



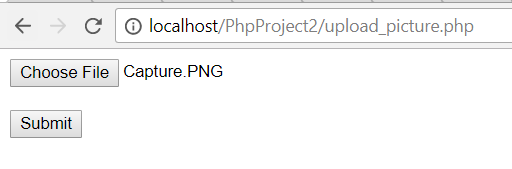
6. After pressing submit, the application have given us an error because the program will not allow a file that does not have a .jpg, .gif, or .png extension.



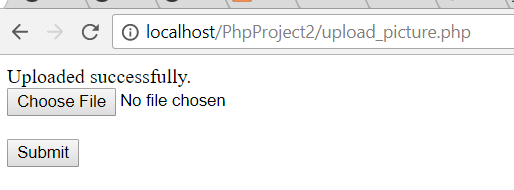
7. Just to be sure, we check and see that the file did not upload to the directory, just like we intended.

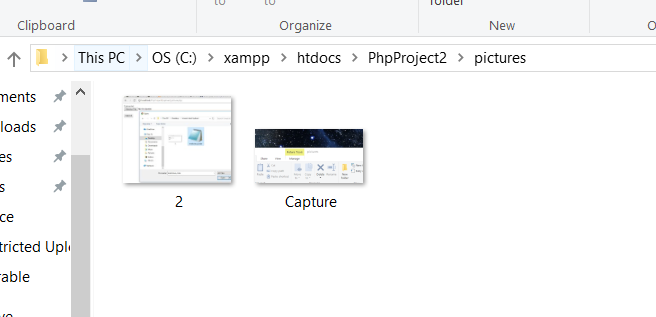


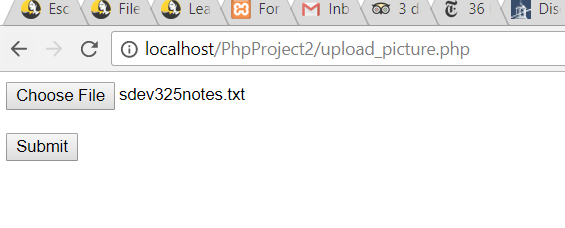
8. Now let’s try to upload a .png file just to make sure that we are allowing all of our chosen file types. We are now going to upload an image file called “capture.png.”

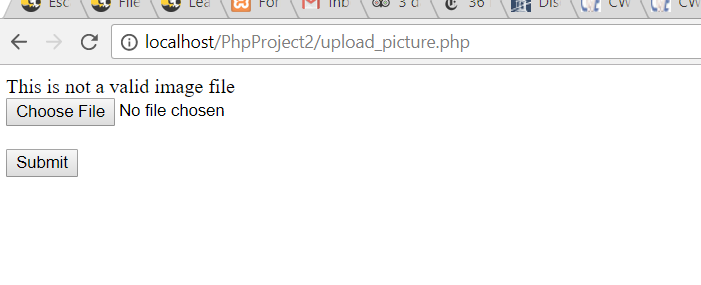


9. We get a message stating that it worked and we can see that it is in the directory.





10. Now let’s try one last thing. Let’s try to upload a text file with the extension .txt.

11. We press submit and get our error message. 

12. It did not upload to the directory because it is the wrong format. Our snippet application works as planned!!!

