Lab 19	
Name	Dang Hoang Nguyen
Student ID	SE171946

What are File Upload vulnerabilities, and how do they present a risk to web applications and servers? Discuss the potential consequences of improperly handled file uploads, including the execution of malicious code, system compromise, and data breaches. Explain how these vulnerabilities differ from other input-related security issues.

Vulnerabilities related to file uploads are security flaws in web applications that let users submit files to the server. The web application and the server architecture are both seriously at danger from these vulnerabilities. They can have serious repercussions, including the execution of malicious code, system penetration, and data breaches, if they are not handled correctly.

1. 1. Malicious Code Execution: The potential for attackers to upload files containing malware or scripts is one of the biggest hazards connected with file upload vulnerabilities. These files have the potential to be run on the server once they are uploaded, which could result in a number of security problems, from full system compromise to unauthorized access to private information.

   2. System Compromise: An attacker may be able to access the system as a whole through improperly managed file uploads. Attackers may upload files that take advantage of holes in the operating system or server-side software, giving them the ability to run arbitrary commands, escalate privileges, or obtain unauthorized access.

   3. Data Breaches: If attackers are able to upload and access private information kept on the server, file upload vulnerabilities may also lead to data breaches. Attackers might, for instance, upload malicious files intended to retrieve or alter private information, including user credentials, financial records, or personal data.

Unlike other input-related security flaws like SQL injection or cross-site scripting (XSS), file upload vulnerabilities do not directly manipulate the data that the application stores or displays; instead, they include the transfer of potentially executable content to the server. File upload vulnerabilities give attackers the ability to directly interact with the server environment, providing a wider range of dangers, compared to other input-related vulnerabilities that can only allow attackers to change data or execute scripts within the context of the online application.

Describe the process of exploiting a File Upload vulnerability in a web application. What types of files and content might an attacker use to exploit such vulnerabilities, and how can they bypass common security checks?

Exploiting a File Upload vulnerability typically involves the following steps:

1. **Identifying Vulnerable Functionality**: Attackers first identify web applications that allow file uploads and assess whether there are any weaknesses in how files are processed and stored on the server.
2. **Uploading Malicious Files**: Once a vulnerable upload functionality is identified, attackers upload files containing malicious payloads. These payloads may include scripts, malware, or other executable content designed to exploit vulnerabilities in server-side software or the underlying operating system.
3. **Bypassing Security Checks**: Attackers may attempt to bypass common security checks implemented by the web application, such as file type validation or size restrictions, to upload malicious files

successfully. This can involve techniques such as renaming file extensions, disguising file types, or using techniques like steganography to hide malicious content within seemingly harmless files.

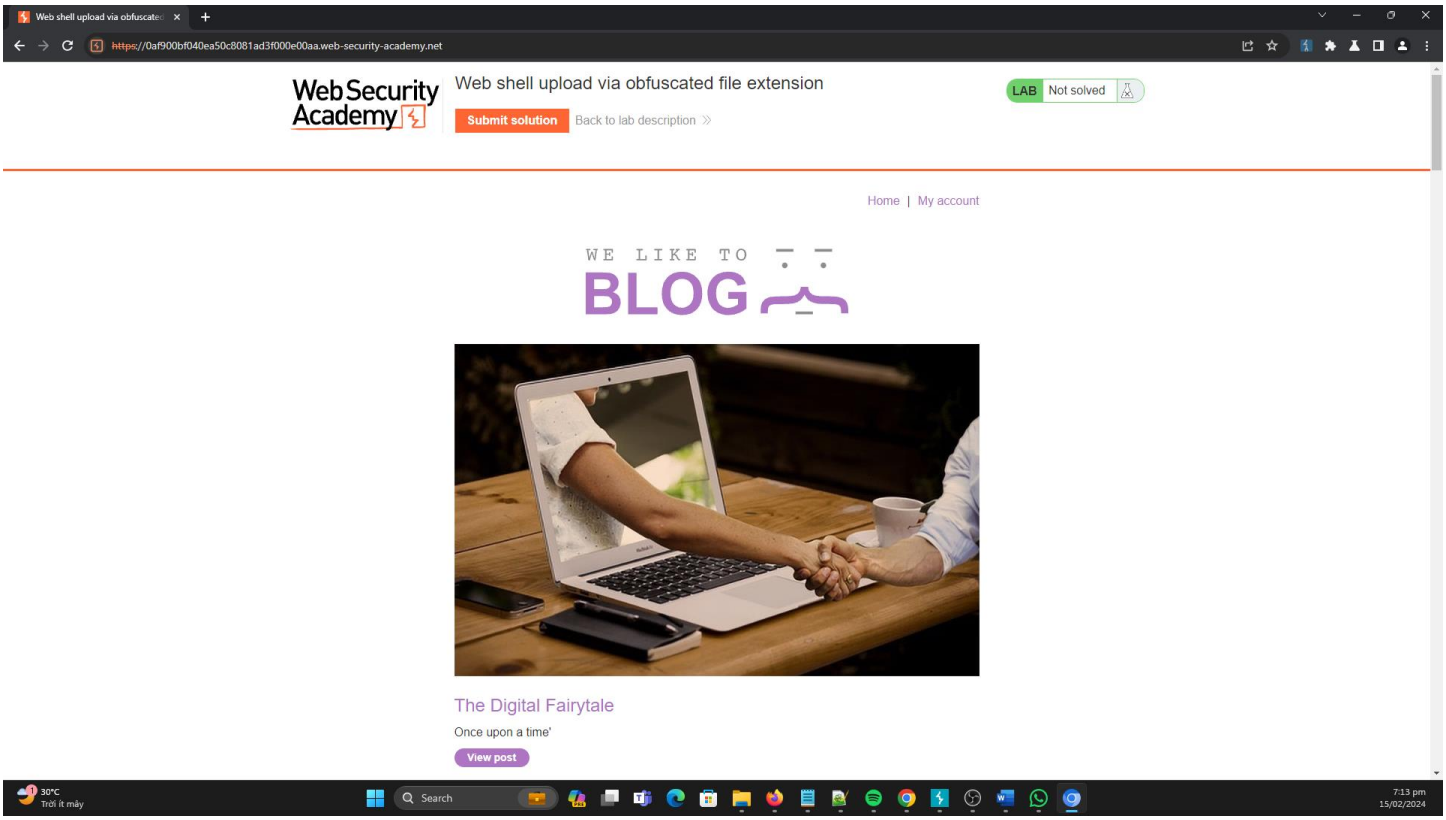
4. **Executing Malicious Code**: Once the malicious file is successfully uploaded to the server, attackers may trigger its execution by accessing it through the web application or exploiting other vulnerabilities to execute the uploaded file.

Types of files and content attackers might use to exploit file upload vulnerabilities include:

- **Malicious Scripts**: Attackers may upload scripts such as PHP, JavaScript, or ASP files containing code designed to execute arbitrary commands, access sensitive information, or perform malicious actions on the server.
- **Executable Files**: Attackers may upload executable files such as .exe or .dll files containing malware or backdoors that can compromise the server's security or provide remote access to attackers.
- **Web Shell Scripts**: Attackers may upload web shell scripts, which are small scripts that provide attackers with a web-based interface to execute commands and interact with the server remotely.

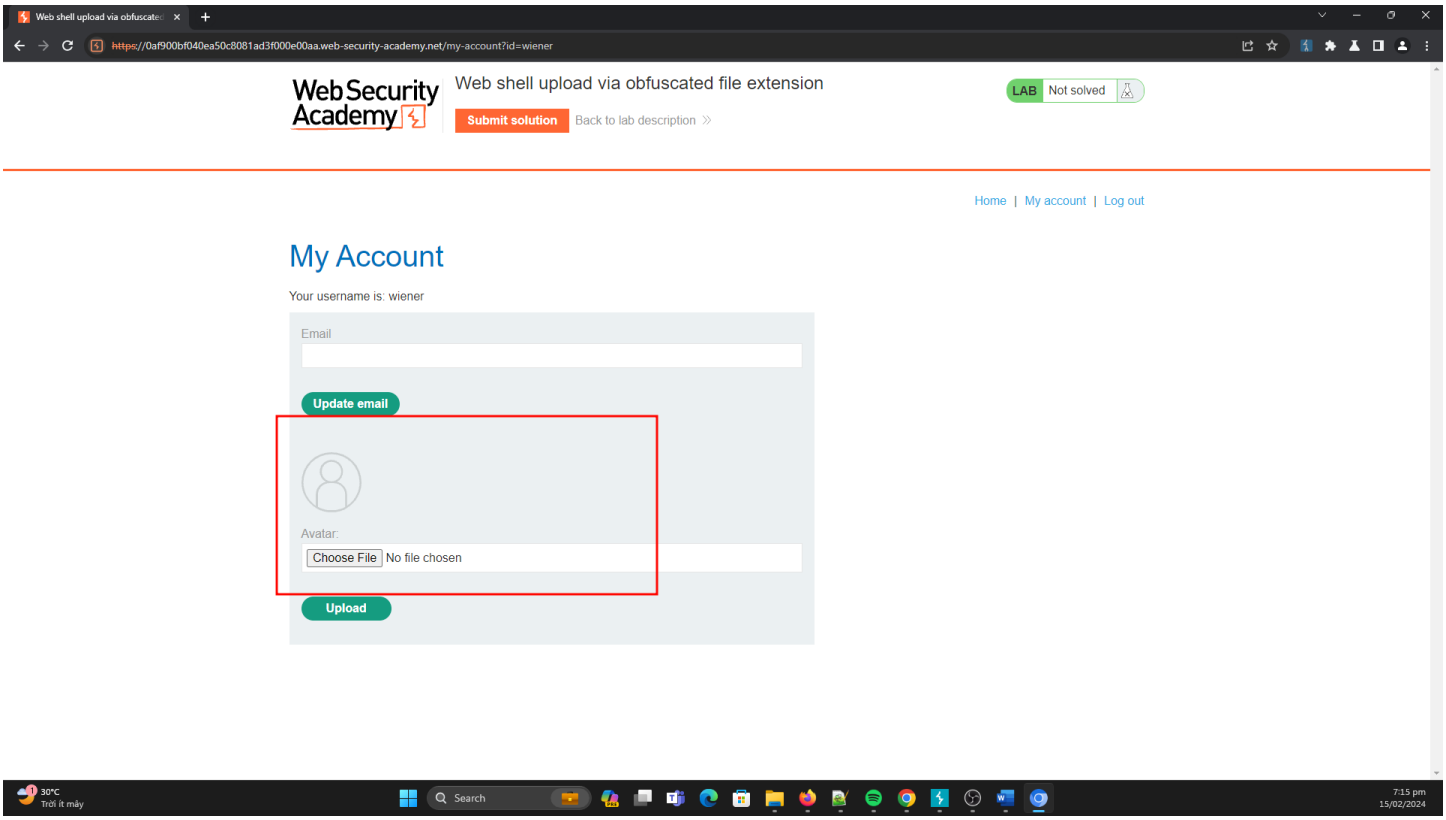
Lab: Web shell upload via obfuscated file extension

We will start by gaining access to the lab and exploring the "my account" section.

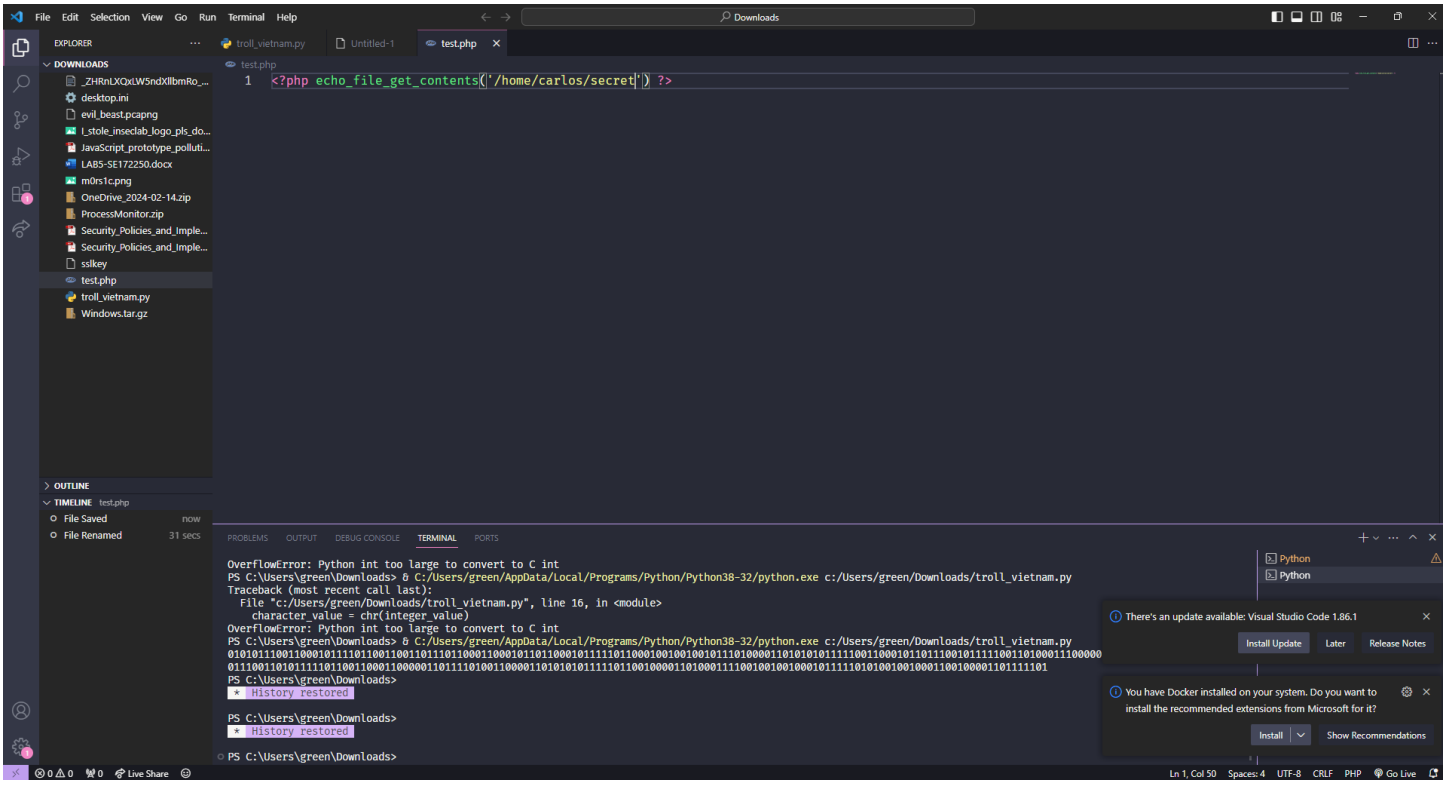


Next, in order to log in, we'll use the following credentials: Password: Peter; Username: weiner.

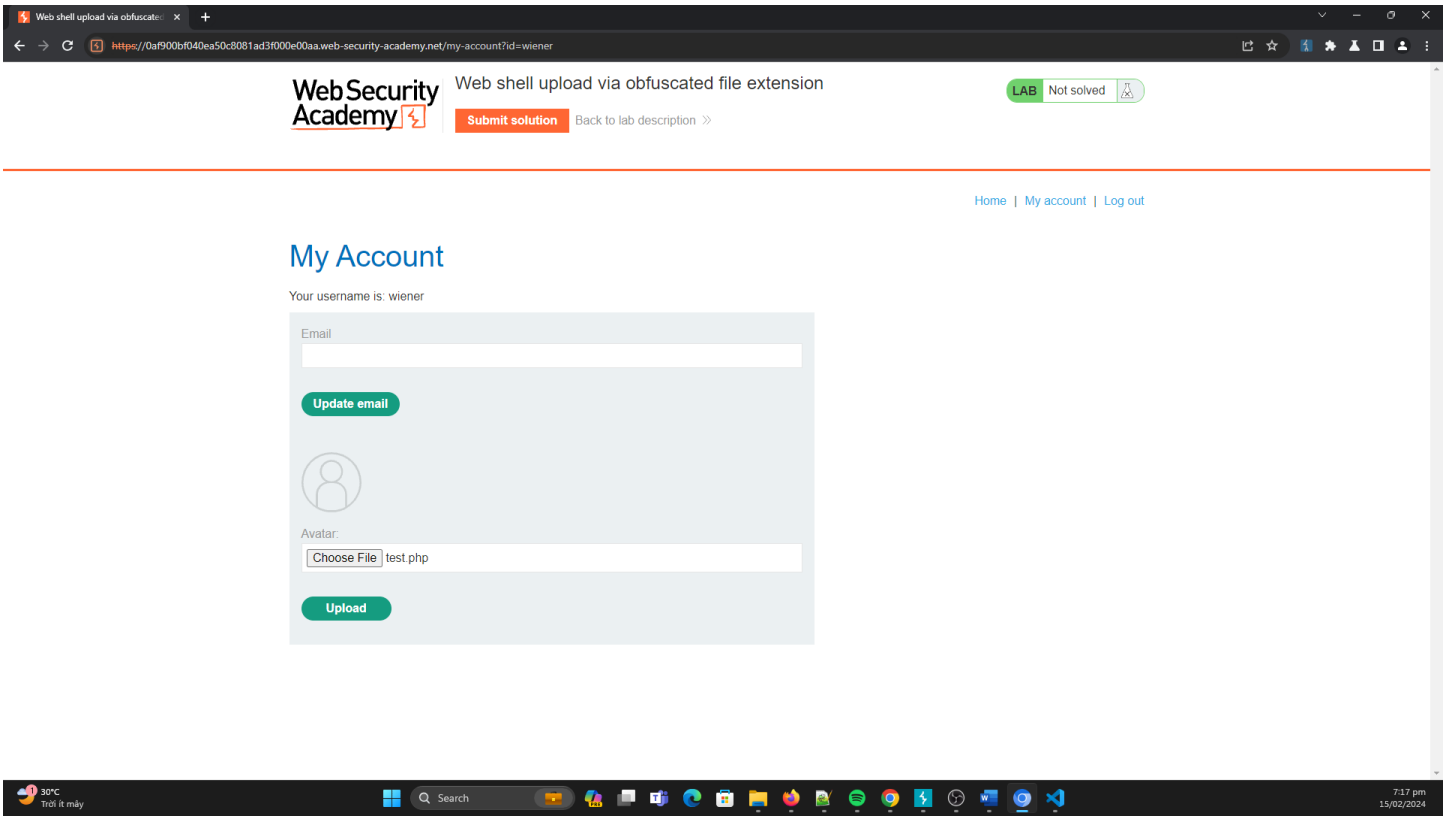
We can see a file upload section; let's check if we can use this to upload a PHP file.

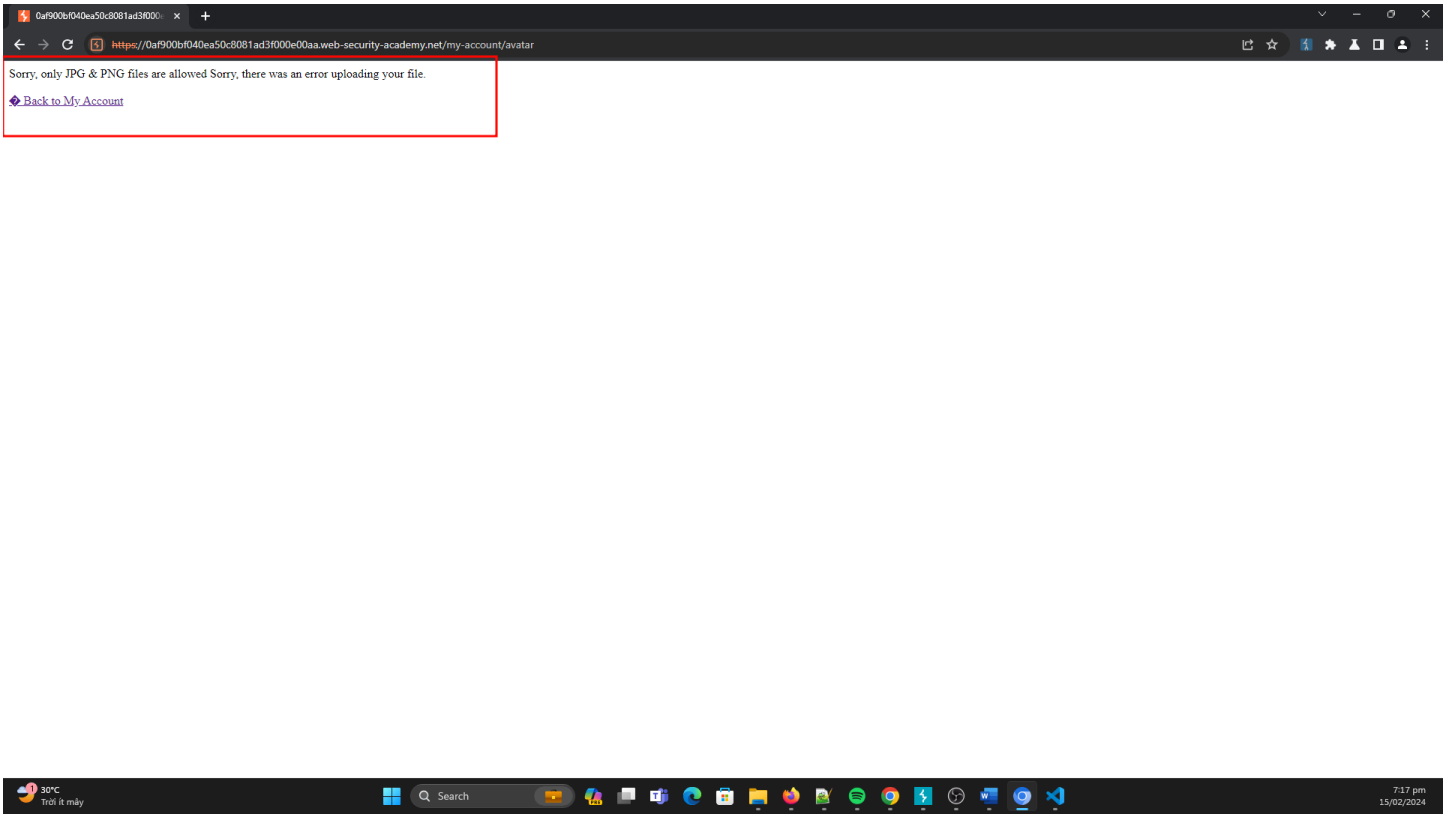


We have to extract the contents of the file from /home/carlos/secret in accordance with the lab's instructions.Next we'll create a PHP file called test.php in order to exfiltrate the file.



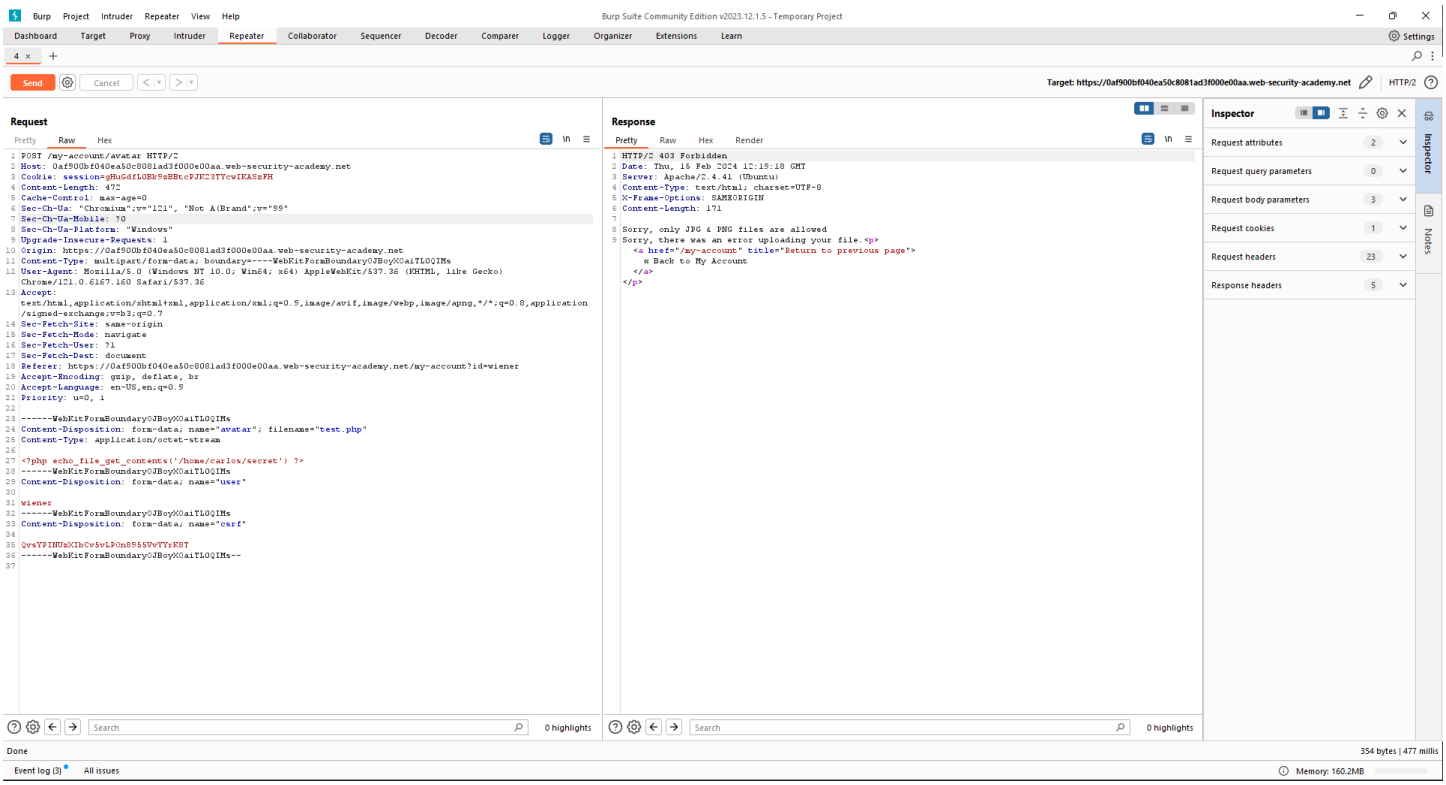
We receive an error message stating that only JPG and PNG files are permitted when we attempt to upload the PHP file.



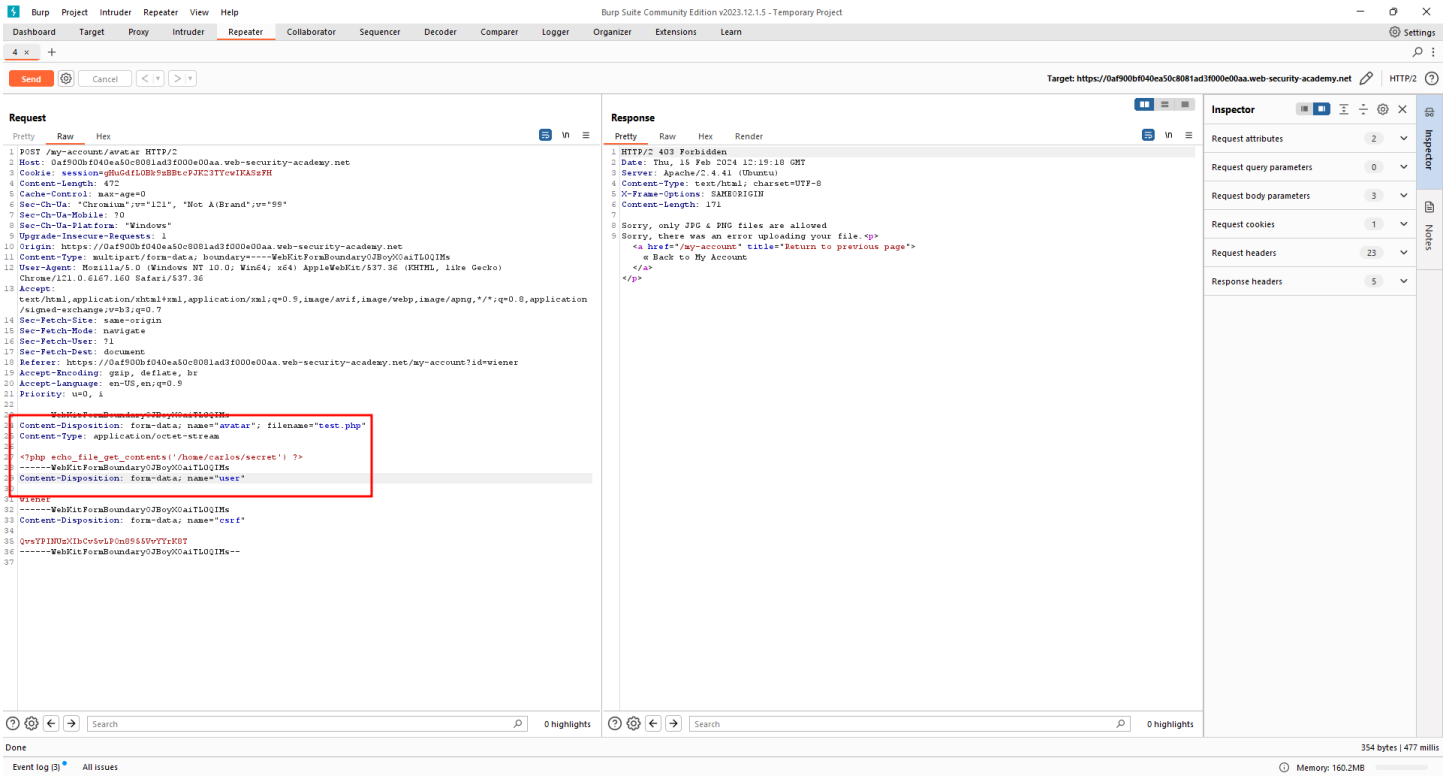


We can now make the PHP file appear to be a PNG or JPG file because we know that it can only handle these two file types.

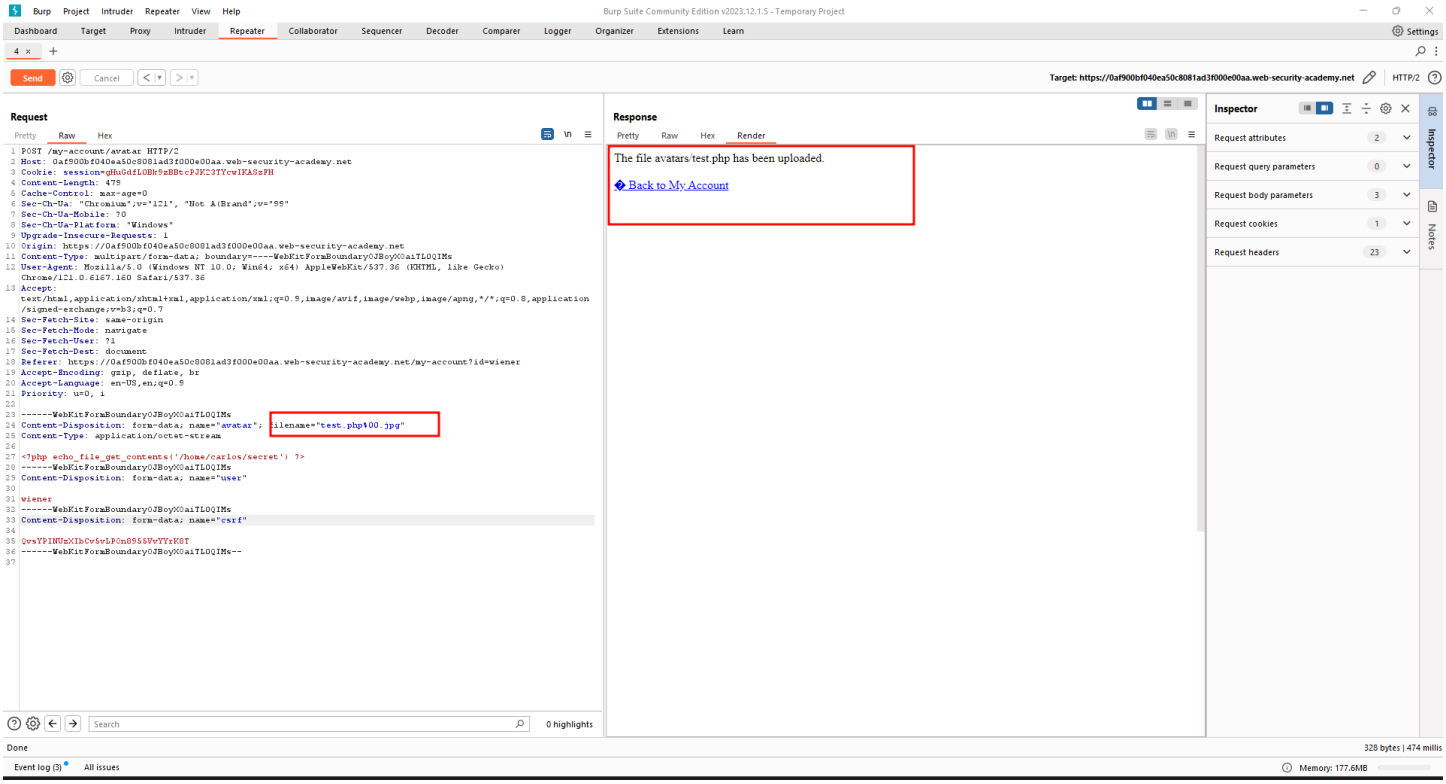
Because of that, Using Burp Suite, we will intercept the post request.



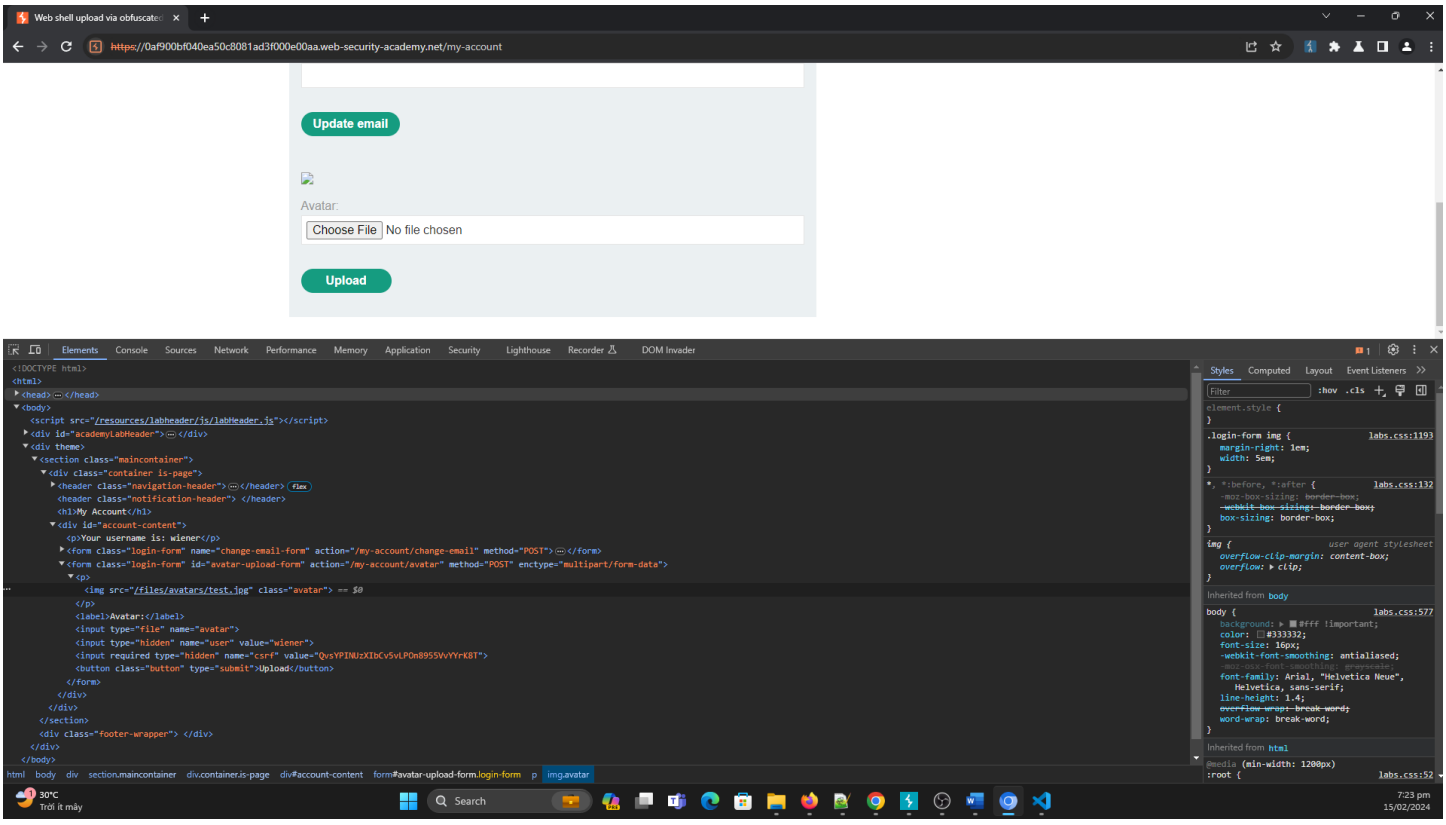
Then we can see that there is a section with the file that we are uploading, i.e., test.php



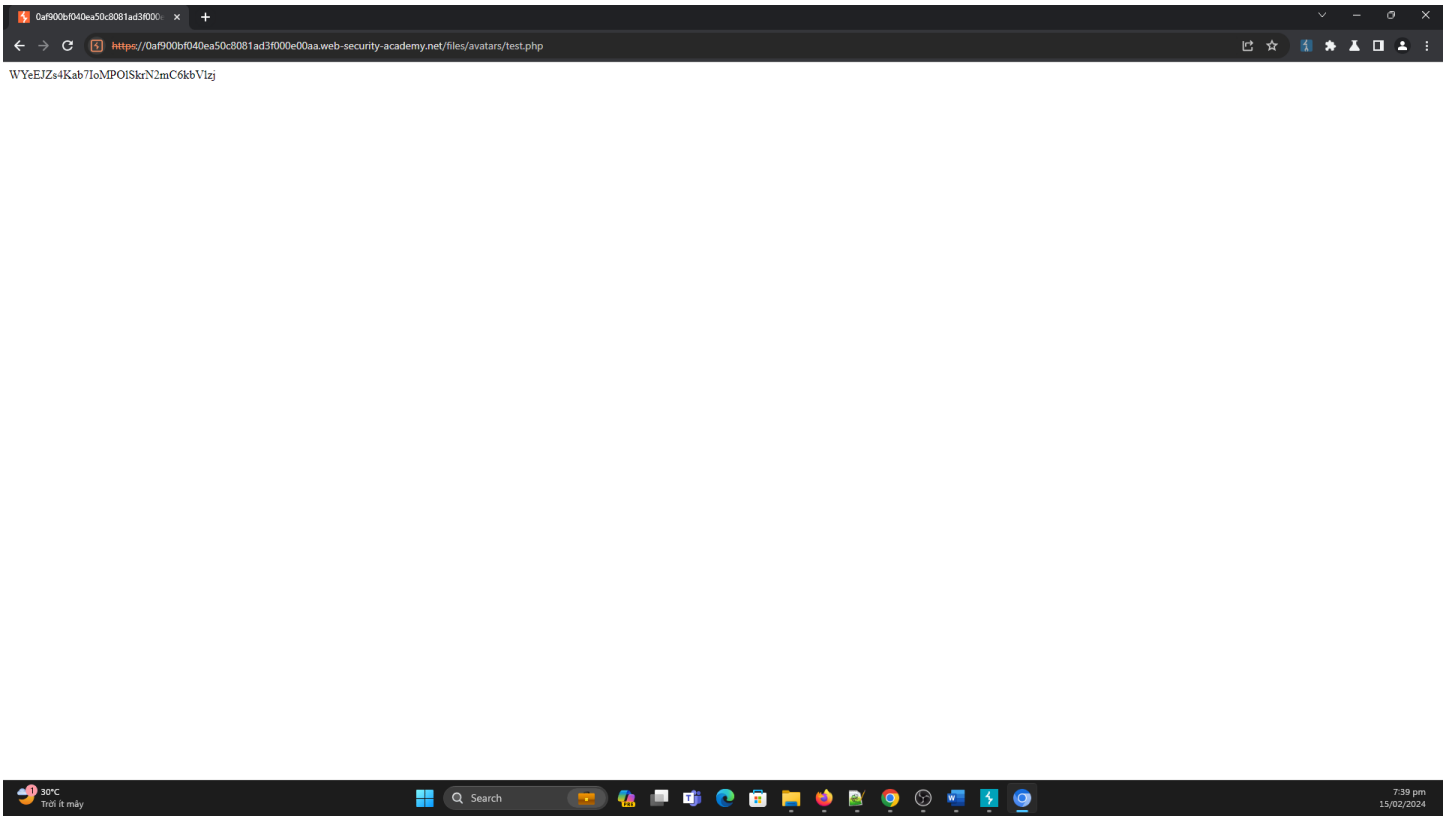
The file will now have its extension changed from test.php to test.php%00.jpg in an attempt to obscure it. Programmers use the letter %00, which stands for null byte, to trick file systems that only utilize the file extension to determine the kind of file. This character is frequently used to indicate the end of a string. It looks just like a safe.jpg image, but it contains a PHP script.

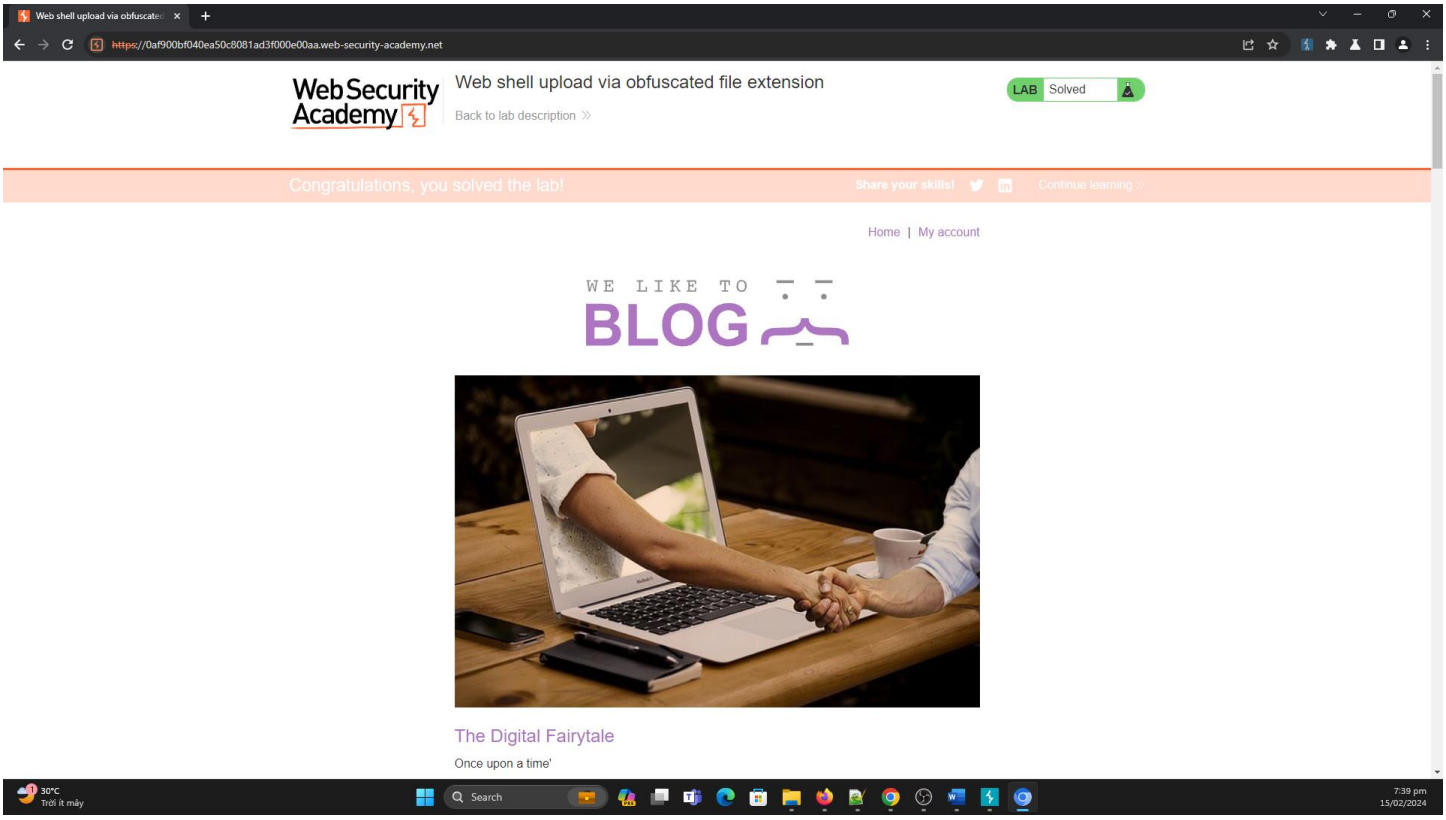


After examining the GET request in the Burp Suite repeater, we will locate and retrieve the uploaded file's directory.



In order to access the data we are attempting to gain, we will need to change the directory to /files/avatars/test.php.


https://0af900bf040ea50c8081ad3f000e00aa.web-security-academy.net/files/avatars/test.php
WYeEJZs4Kab7IoMPOlSkrN2mC6kbVlzj


https://0af900bf040ea50c8081ad3f000e00aa.web-security-academy.net
Web Security Academy
Web shell upload via obfuscated file extension
Back to lab description
LAB Solved
Congratulations, you solved the lab!
Share your skills!
Continue learning
Home | My account
WE LIKE TO BLOG
The Digital Fairytale
Once upon a time'