1. Remote code execution via web shell upload:
   * Exploited this vulnerability by first uploading a valid picture on the server and intercepted that request in BurpSuite.
   * Then we modified the request’s file name and the content of the file to some PHP script file and then upload that file instead of the old file.
   * Now in BurpSuite’s Repeater you just need to change the path of the request to the desired file and when you send the request, it will get executed.
2. Web shell upload via Content-Type restriction bypass:
   * Now in this case we need to upload a PHP file again instead of the avatar image on the server but this time there is a content type restriction implemented so to bypass that we need to change the request and put the content type to **image/jpeg** which will allow the system to upload file of any type.
3. Web shell upload via path traversal:
   * In this scenario the application allows us to upload the PHP file but when checking the response, we got to know that the server is returning the PHP script as it is without executing it.
   * Now modify the HTTP request with the desired filename and content with the filename as **../xyz.php.** Now the file is uploaded to the server but the server is stripping the directory traversal sequence so we will obfuscate the directory traversal sequence by URL encoding the forward slash (/) character as **..%2.**
4. Web shell upload via extension blacklist bypass:
   * We will first try to upload a PHP file to the server’s backend which will throw and error saying that we are not allowed to upload PHP files, also it will tell us the information of the backend server.
   * Now replay the request with a filename as **.htaccess** and content type as **text/plain,** then we need to replace the content of the file to **AddType application/x-httpd-php .l33t.** Send the request and observe that the file was successfully uploaded, then use the back arrow in the Repeater to return to the original request and then upload the PHP file.
   * Then change the extension of .php file to .133t and the file will be successfully uploaded because our .htaccess file was forced the server to execute .133t file as .php file.
5. Web shell upload via obfuscated file extension:
   * In this case the application only allows us to upload JPEG and PNG files which means that the server only checks for the filenames to end with .jpeg or .png so we will change their name as .php%00.jpeg and we will see that the file was successfully uploaded.