Bài tập luyện tập:

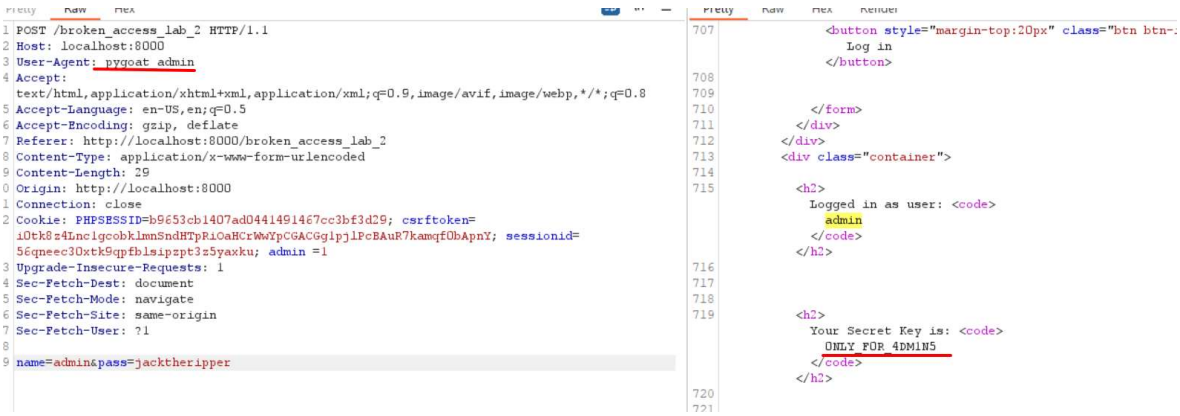
1) http://localhost:8000/broken_access_lab_2

nhờ vào gợi ý của bài

Ta thấy trường lưu trữ thông tin browser là “User-Agent”

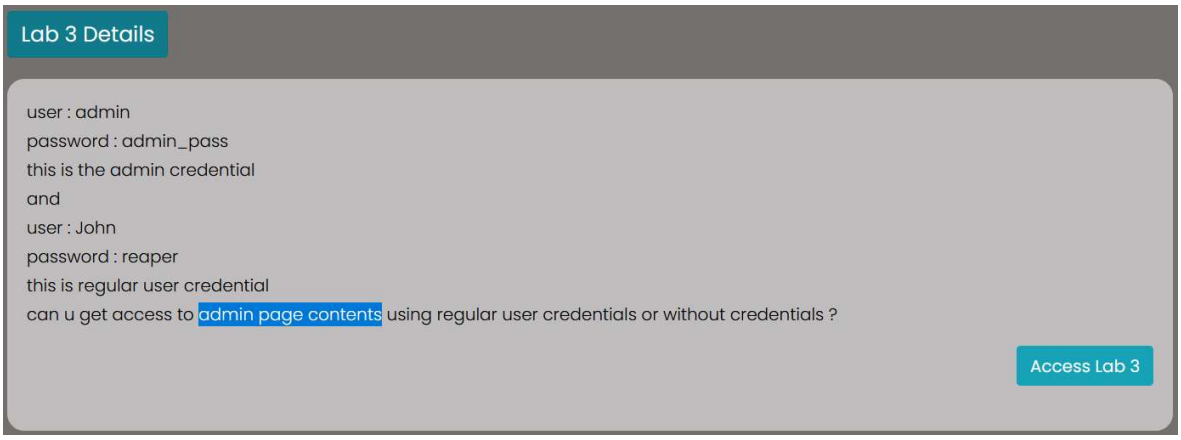
Ta đổi User-Agent mặc định khi gửi thành : pygoat_admin

Và ta lấy được secret-key



2) http://localhost:8000/broken_access_lab_3

theo đề bài yêu cầu ta lấy được nội dung của trang

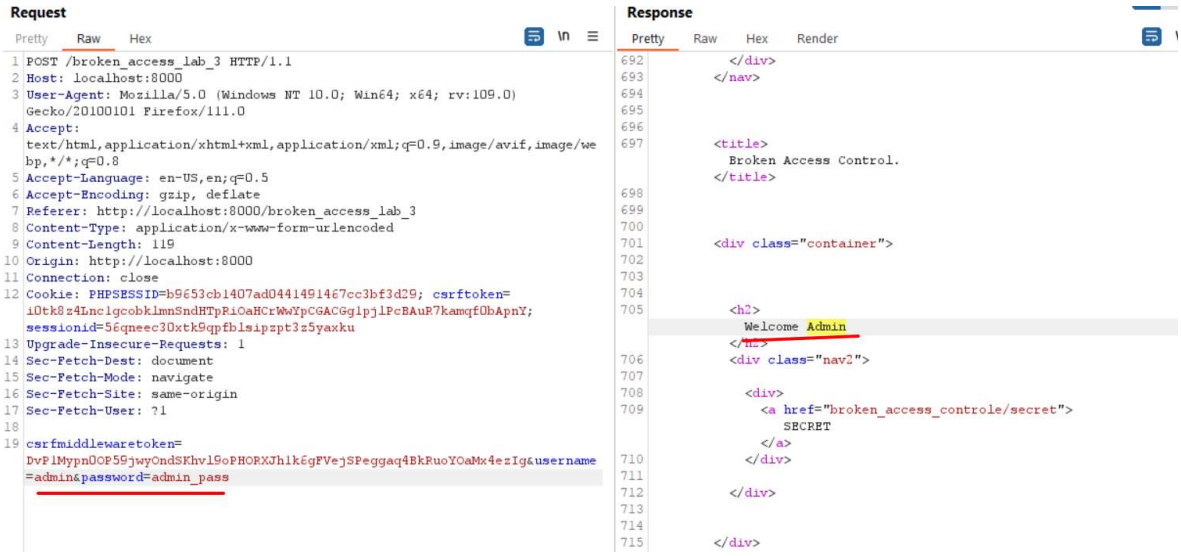


Ta đăng nhập bình thường với 2 tài khoản John – reaper

Và tài khoản admin – admin_pass

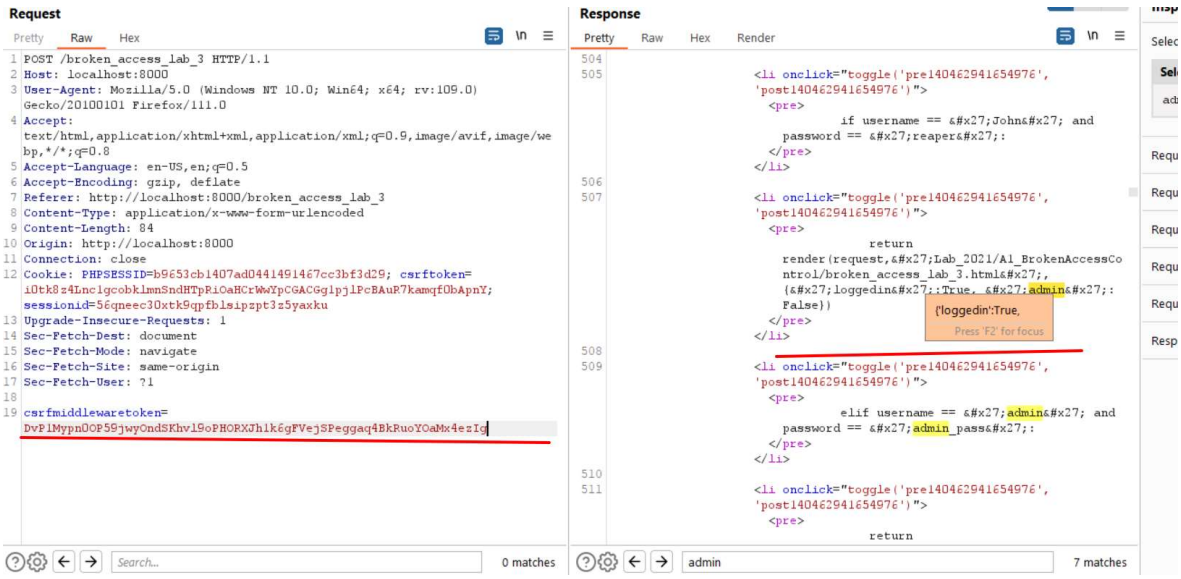
Tài khoản John không có gì khác thay đổi

Với tài khoản admin cũng không có gì thay đổi



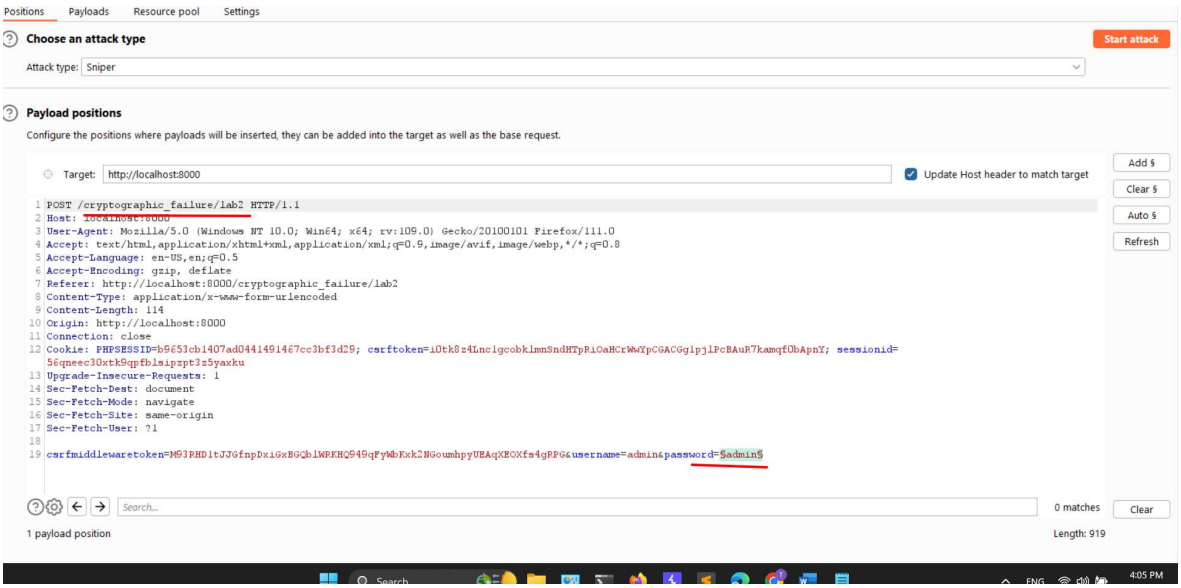
Ta thử thay đổi theo đề bài gợi ý là có thể đăng nhập hoặc không đăng nhập để có thể lấy được nội dung của trang

Nội dung đăng nhập user và password đã được xóa đi và gửi, ta nhận về kết quả như hình bên dưới

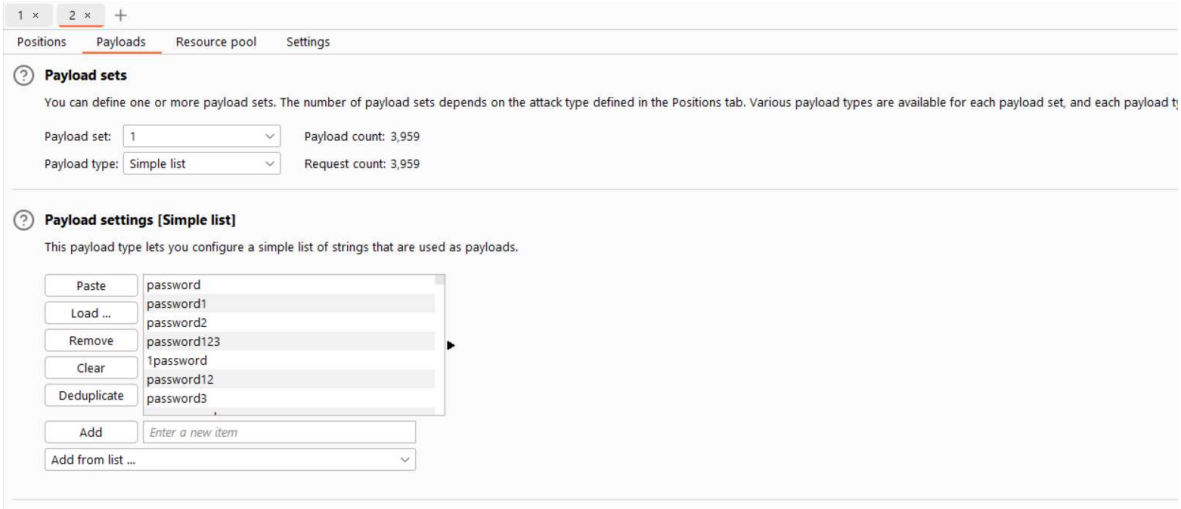


3) http://localhost:8000/cryptographic_failure/lab2

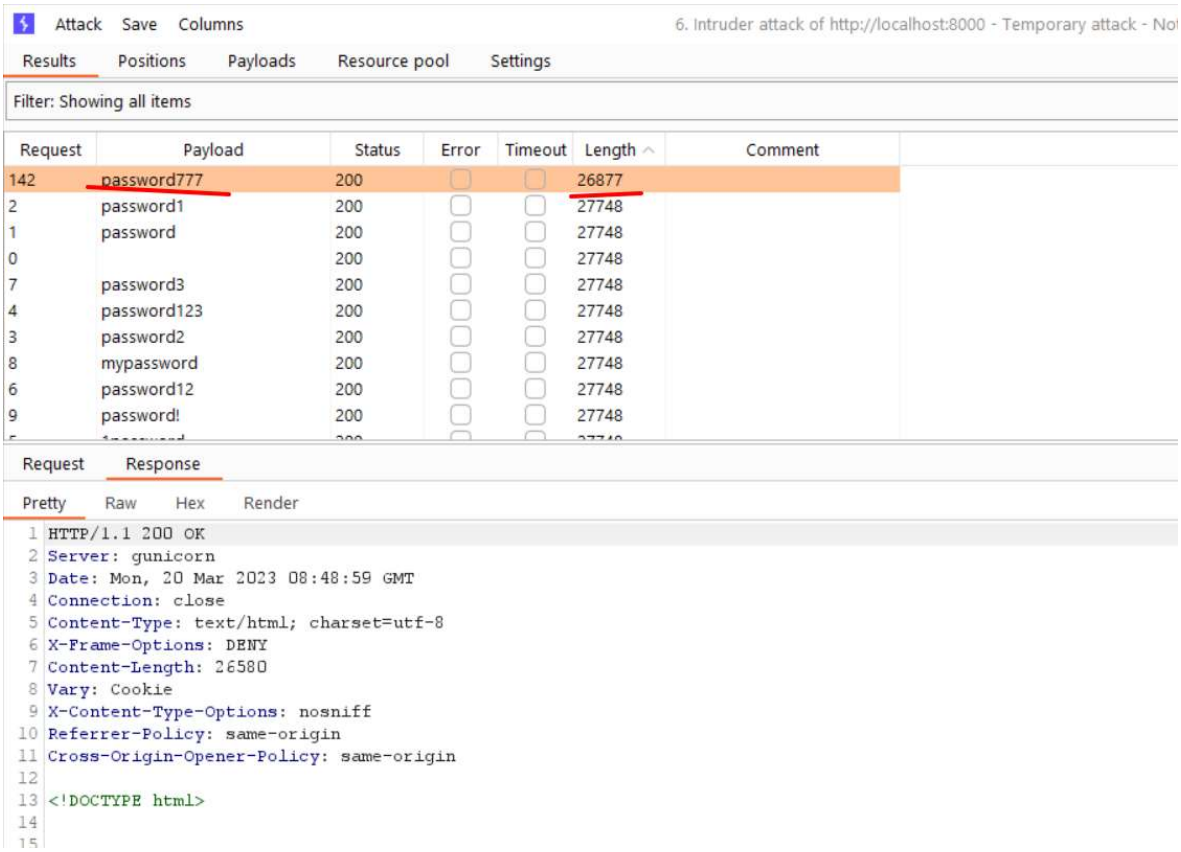
Dựa trên cách làm của bài 1 ta đi tìm tool để crack password nhưng khi làm vậy ta phát hiện ra cách tool crack hash trên mạng thì đều dùng phương pháp brute force nên ta dùng chức năng intruder của burp suite và dùng worklist password phổ biến trên github



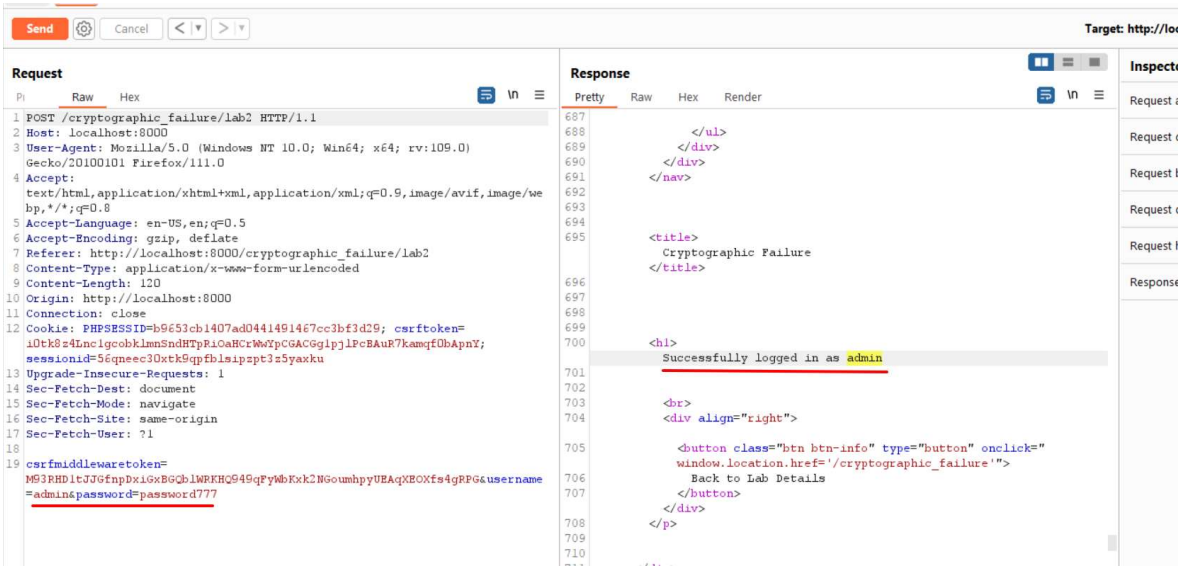
Payload:



Và ta thấy

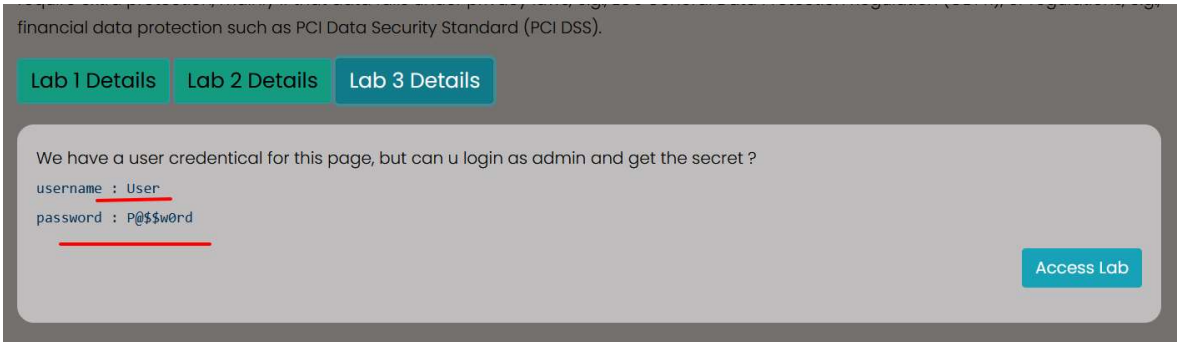


Password: password777 có thay đổi ở độ dài trả về

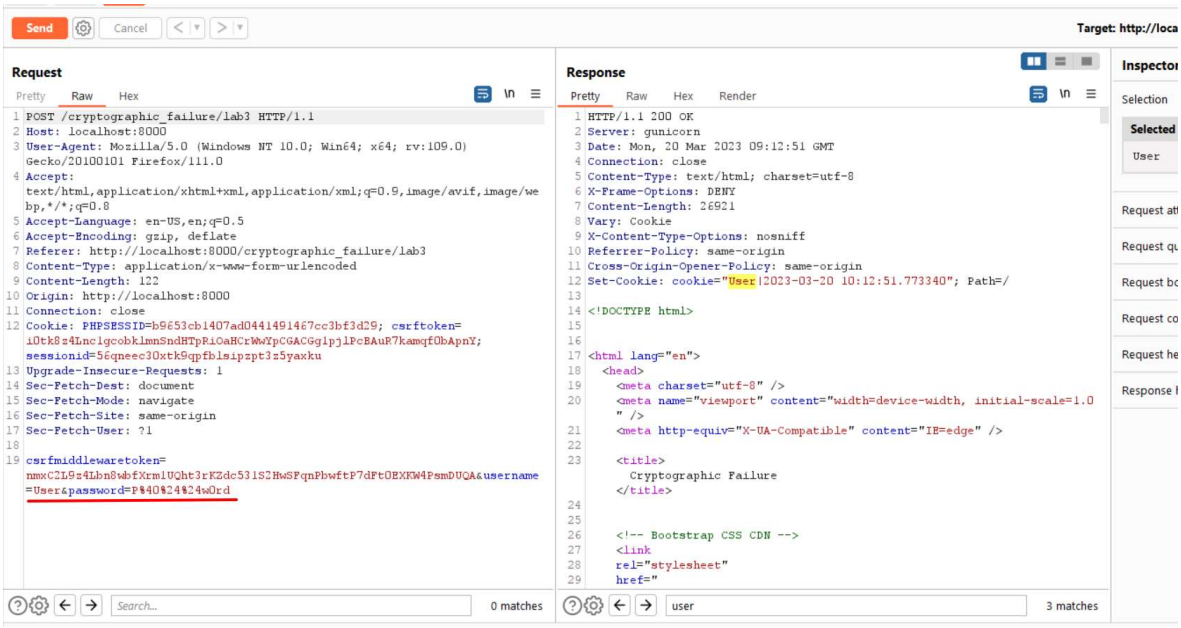


4) http://localhost:8000/cryptographic_failure/lab3

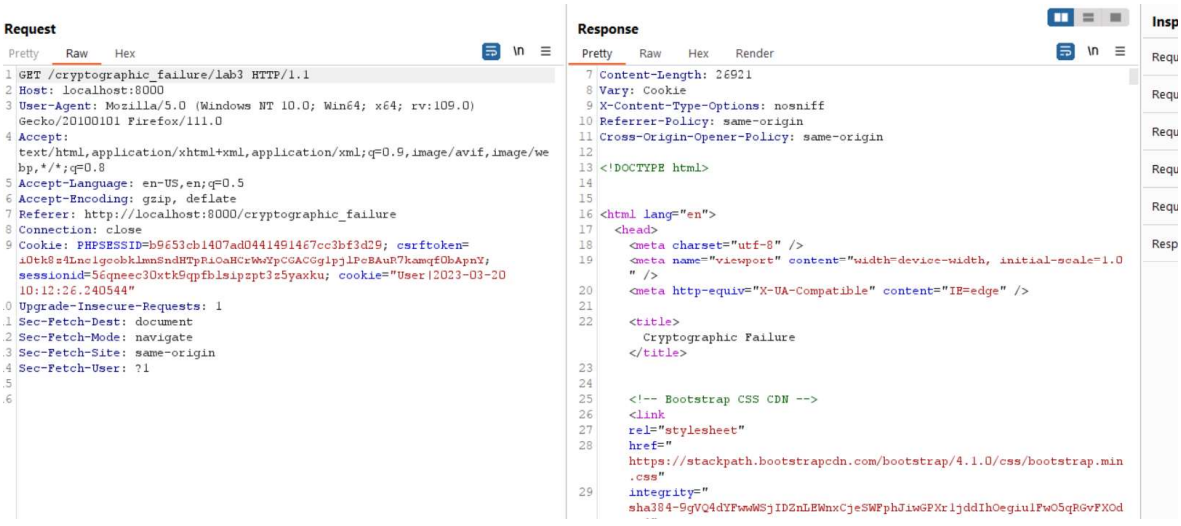
đề bài cho tài khoản đăng nhập



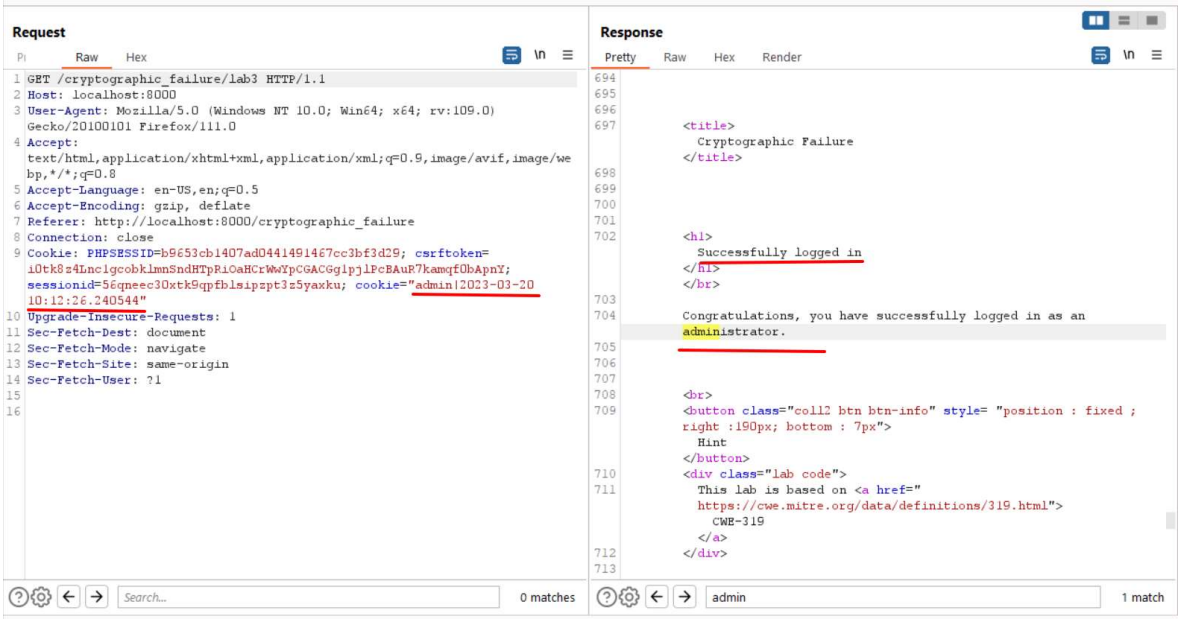
Ta đăng nhập với tài khoản đã cho



Và thấy có thêm trường cookie mới với tên đăng nhập và thời gian đăng nhập



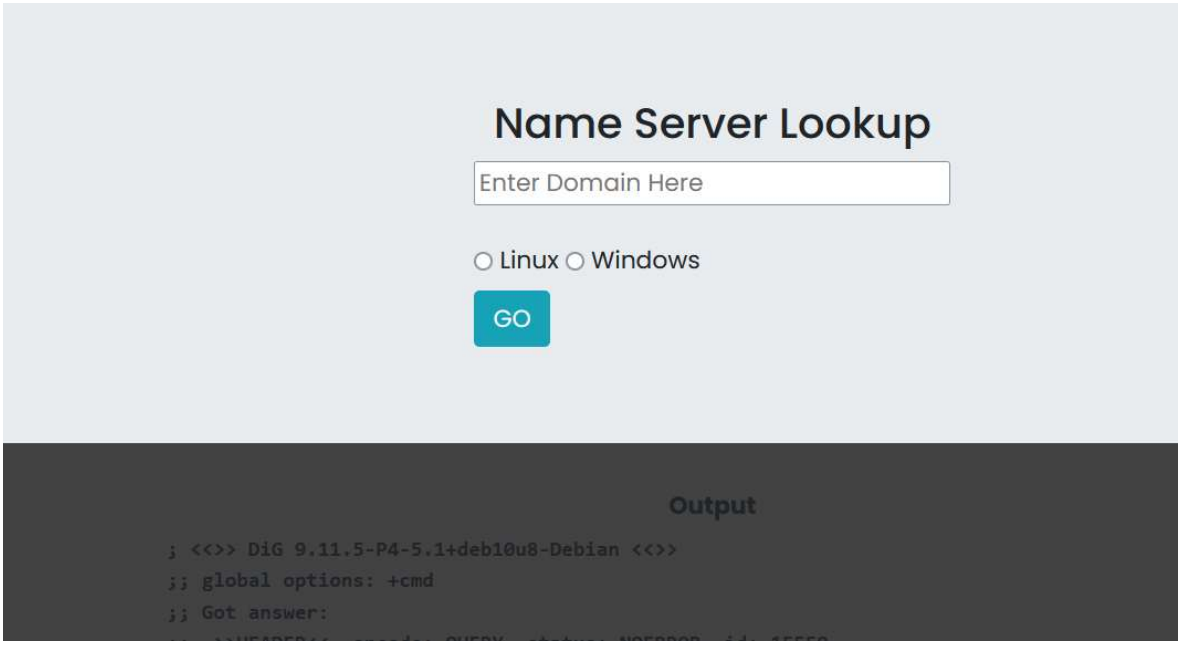
Thấy vậy ta sửa từ User thành Admin và xem kết quả



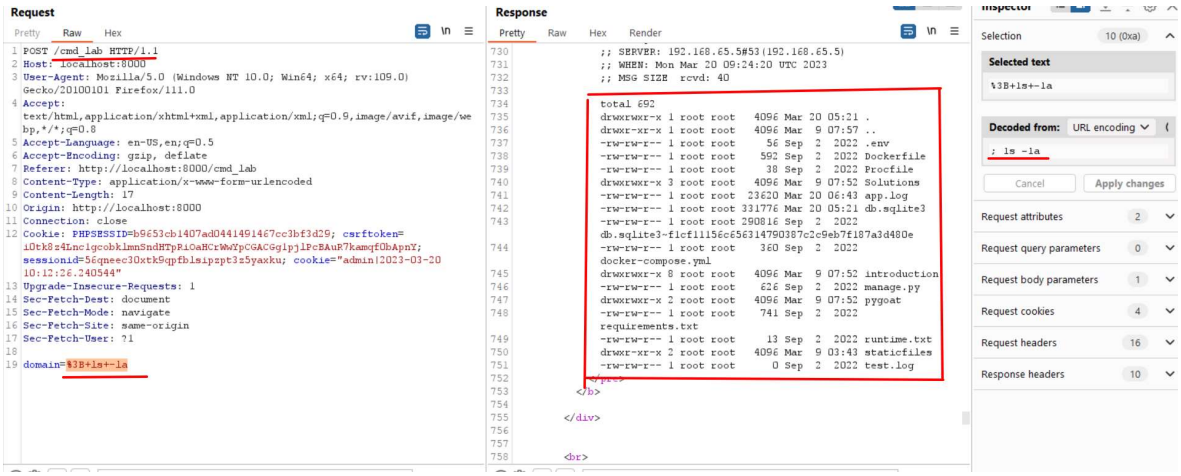
Đăng nhập thành công với từ cách admin

5) http://localhost:8000/cmd_lab

command injection



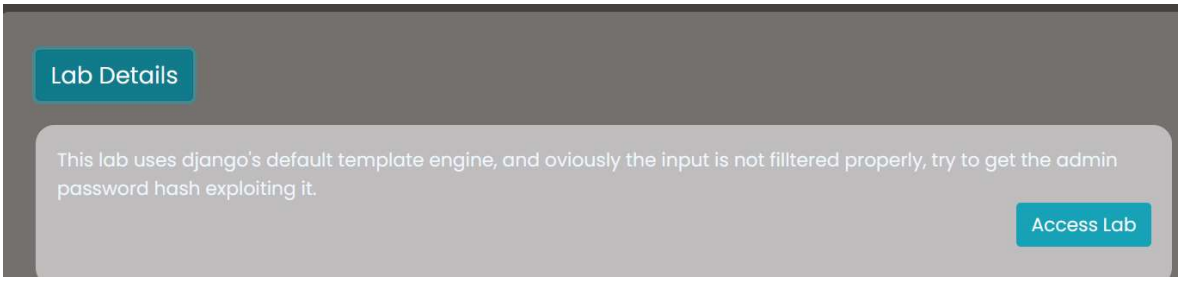
Server thiết kế để thực thi lệnh lookup, ta thấy vậy ta liền chèn thử theo syntax của lệnh và kèm theo code



Kết quả thu được

6) http://localhost:8000/ssti/lab

Đề bài yêu cầu tìm thông tin của admin



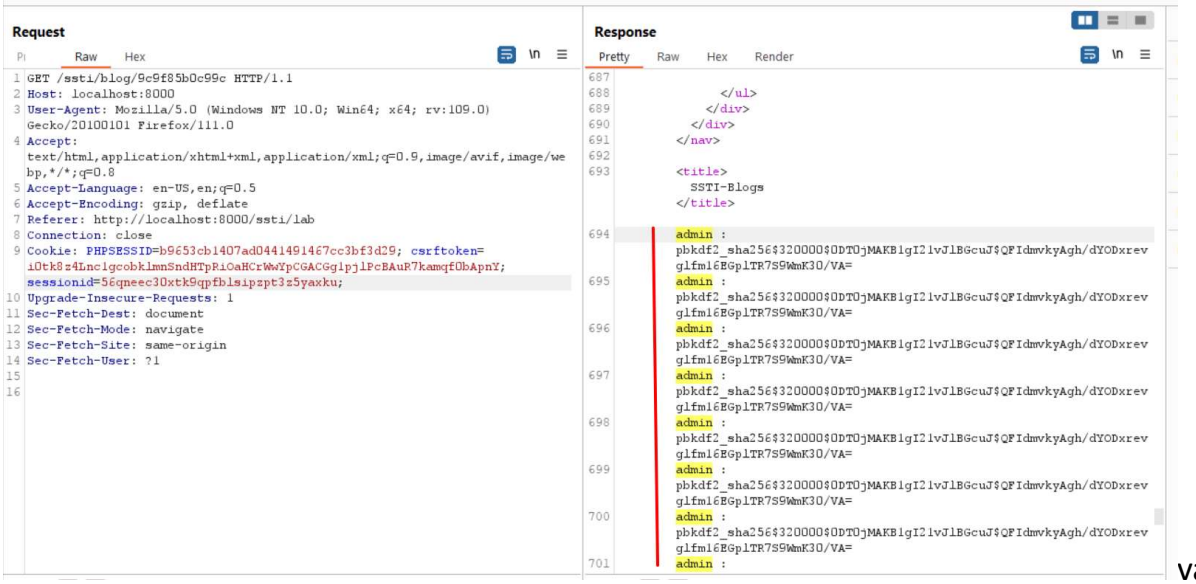
Sau khi tìm hiểu thì ta thấy bài này chạy Django frameworks

Ta tìm cách khai thác template injection của Django framework

Và đây là payload

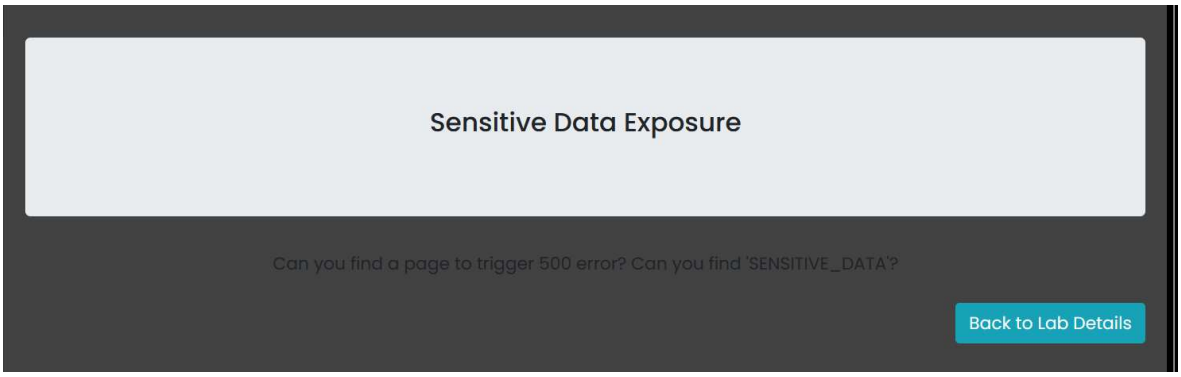
{% load log %}{% get_admin_log 10 as log %}{% for e in log %}

{{e.user.get_username}} : {{e.user.password}}{% endfor %}



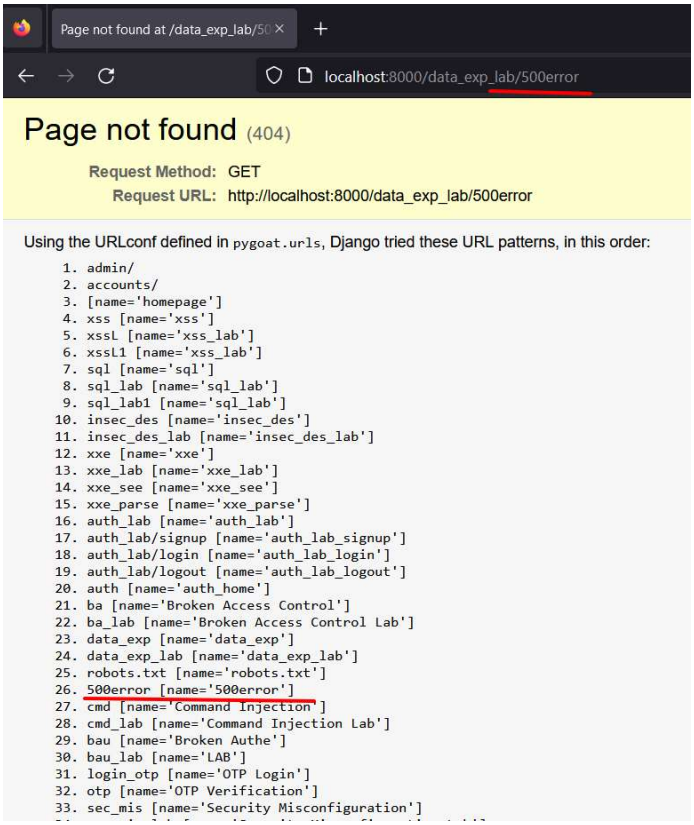
và đây là kết quả nhận được.

7) http://localhost:8000/data_exp_lab



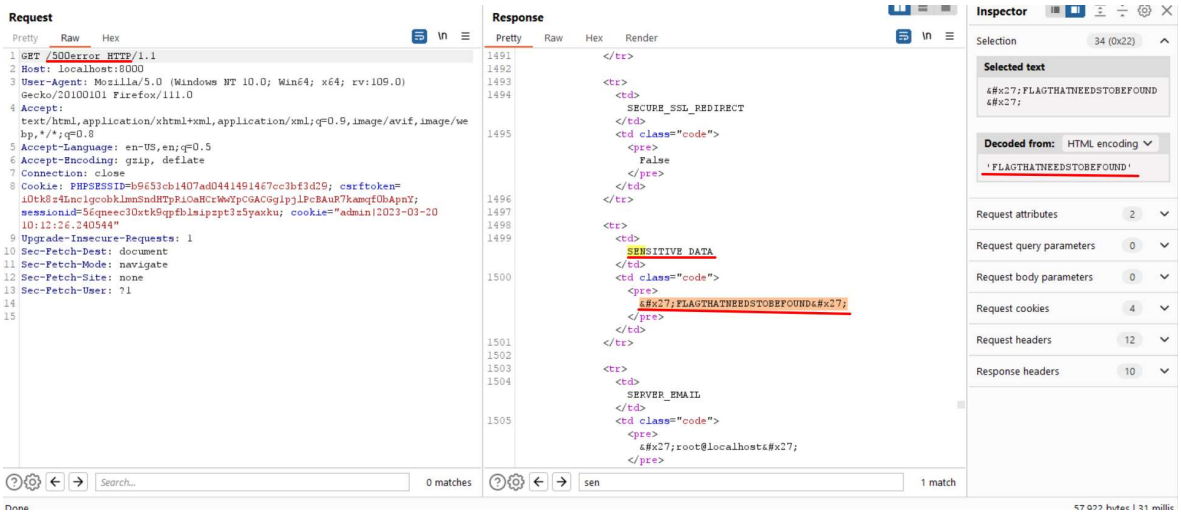
Dựa vào đề bài ta thử tìm 500error trong page

Không xuất ra gì cả nhưng nó lại đưa ta một list path của trang web và ta thấy được path 500error nằm ở đó



Ta thử truy cập với path là “http://localhost:8000/500error”

Và tìm được Sensitive_data như đề bài yêu cầu



8) http://localhost:8000/sec_mis_lab3

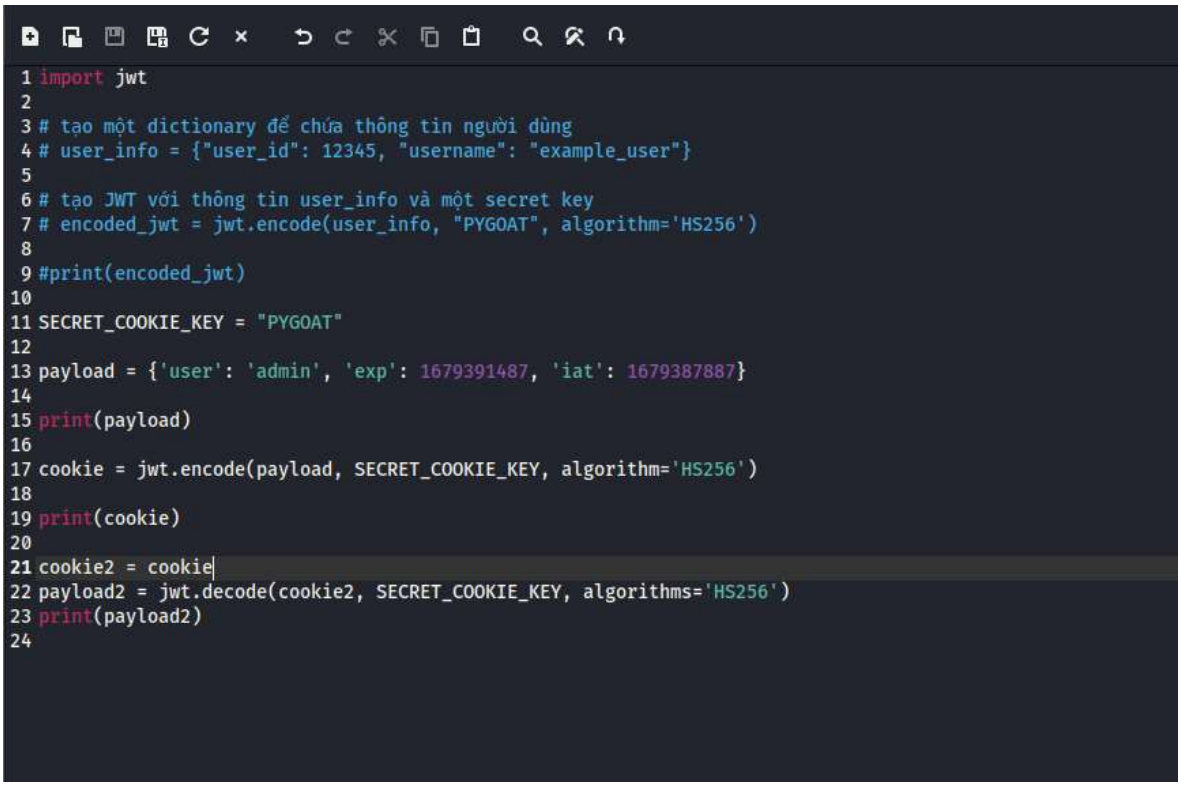
sau khi khai thác bài lab 2 ở trên và đọc đề bài thì đề bài cho code

ta có SECRET_COOKIE_KEY = “PYGOAT”

Theo code ta thấy server tạo 1 cookie để xác nhận danh tính của user : auth_cookie

Nếu user == admin thì đúng thì đăng nhập thành công

Và ta đã tạo được shellcode như sau

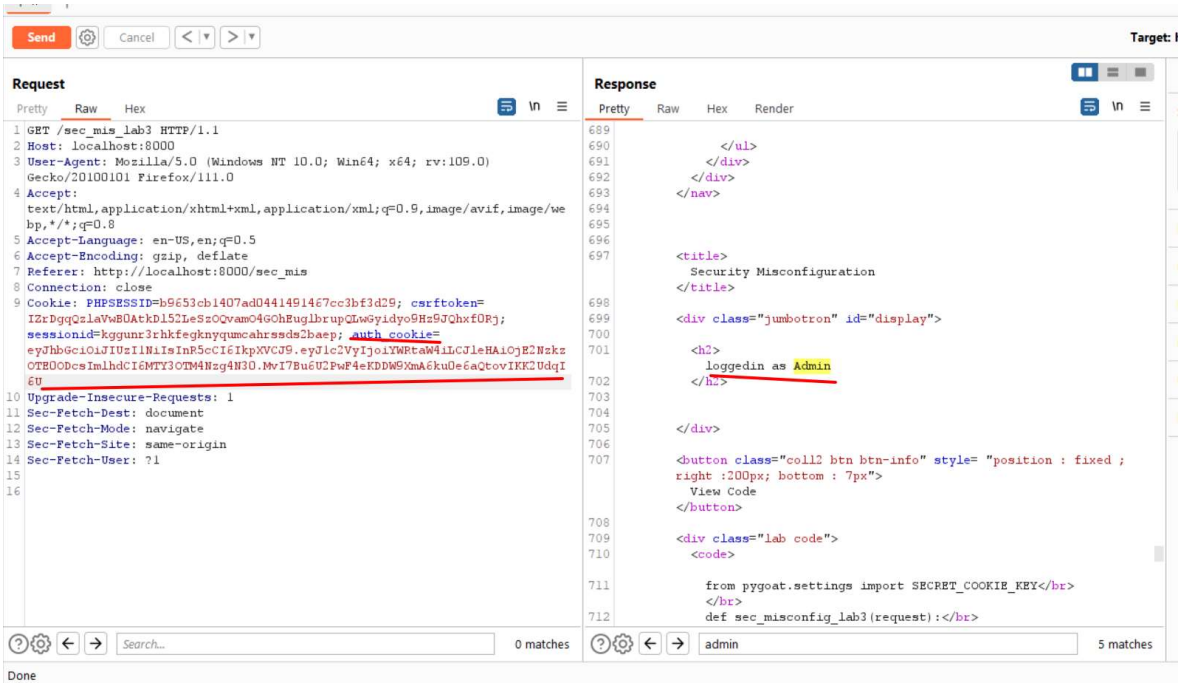


```python
import jwt

# tạo một dictionary để chứa thông tin người dùng
# user_info = {"user_id": 12345, "username": "example_user"}

# tạo JWT với thông tin user_info và một secret key
# encoded_jwt = jwt.encode(user_info, "PYGOAT", algorithm='HS256')

#print(encoded_jwt)

SECRET_COOKIE_KEY = "PYGOAT"

payload = {'user': 'admin', 'exp': 1679391487, 'iat': 1679387887}

print(payload)

cookie = jwt.encode(payload, SECRET_COOKIE_KEY, algorithm='HS256')

print(cookie)

cookie2 = cookie
payload2 = jwt.decode(cookie2, SECRET_COOKIE_KEY, algorithms='HS256')
print(payload2)
```

Kết quả thực thi

```
┌──(ductoan㉿kali)-[~/Desktop/Lab 5]
└─$ python3 shell.py
{'user': 'admin', 'exp': 1679391487, 'iat': 1679387887}
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoiYWRtaW4iLCJleHAiOjE2NzkzOTE0ODcsImlhdCI6MTY3OTM4Nzg4N30.MvI7Bu6U2PwF4eKDDW9XmA6ku0e6aQtovIKK2UdqI6U
{'user': 'admin', 'exp': 1679391487, 'iat': 1679387887}

┌──(ductoan㉿kali)-[~/Desktop/Lab 5]
└─$
```

Và ta đã đăng nhập thành công



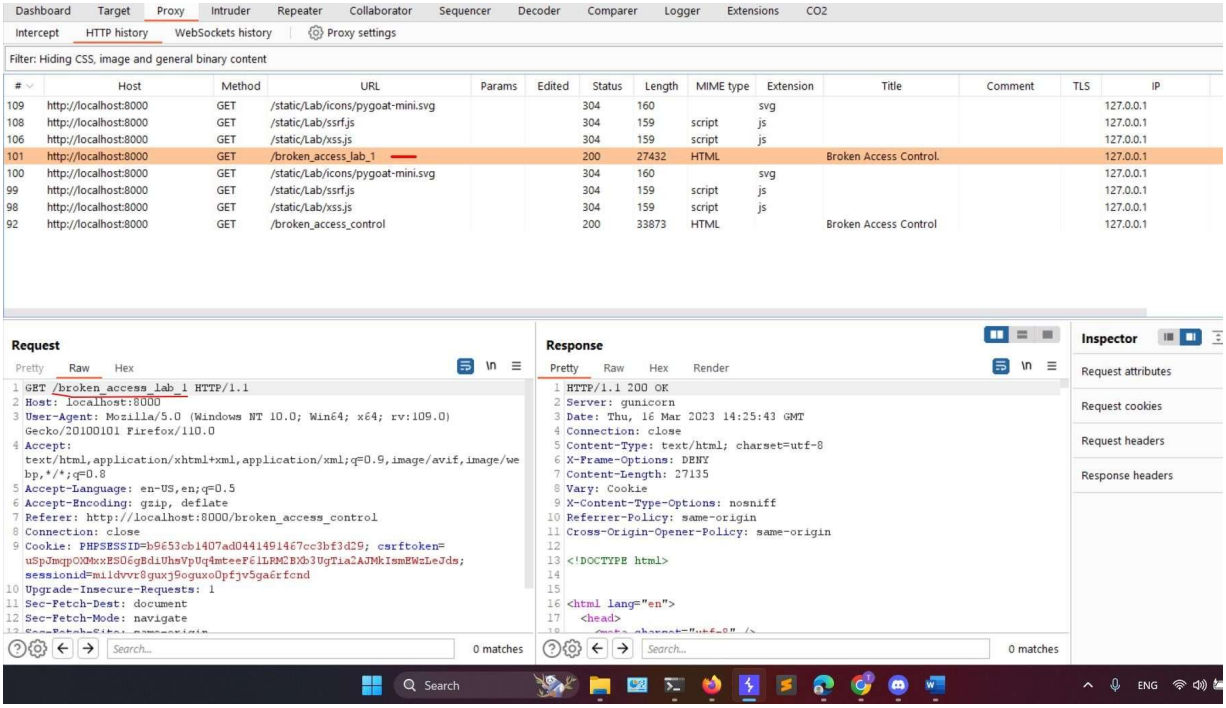
Bài tập 1 và Bài tập 2:

Tiêu đề: broken Access control thực thi hành động ngoài quyền hạn cho phép.

Mô tả: Thay đổi cookie Admin để có thể login với quyền Admin.

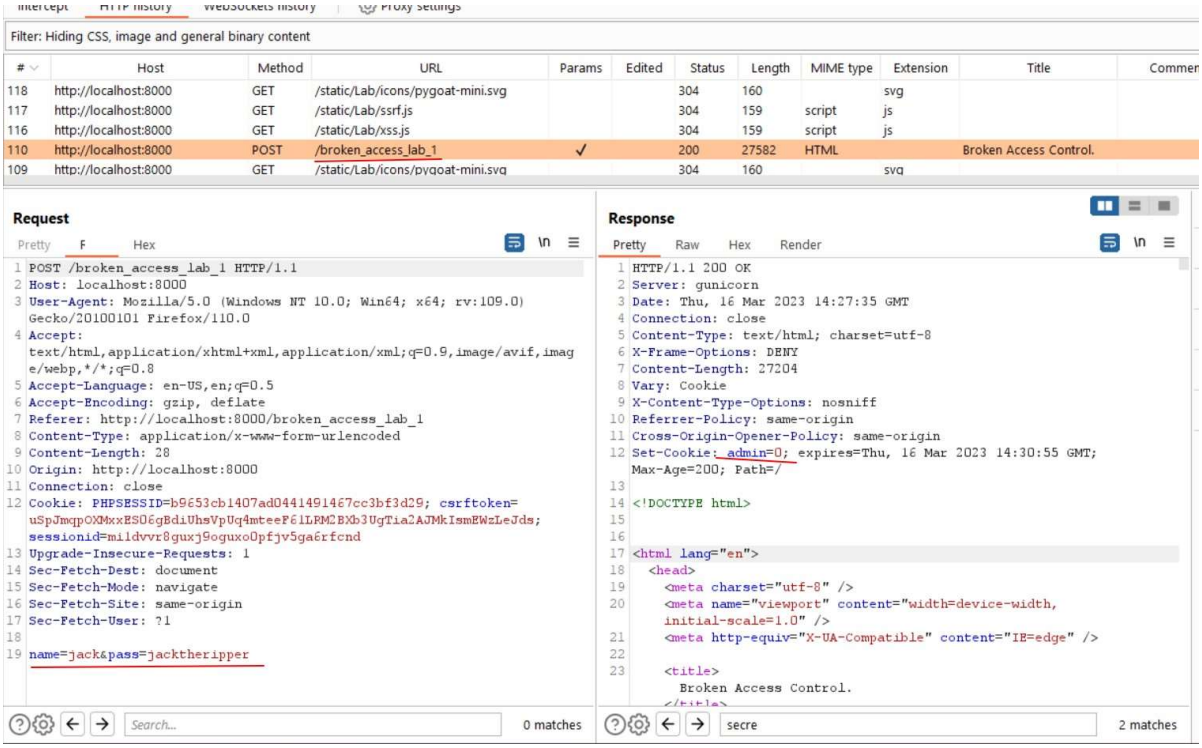
DEMO:

Truy cập vào trang mở broken_access_lab1

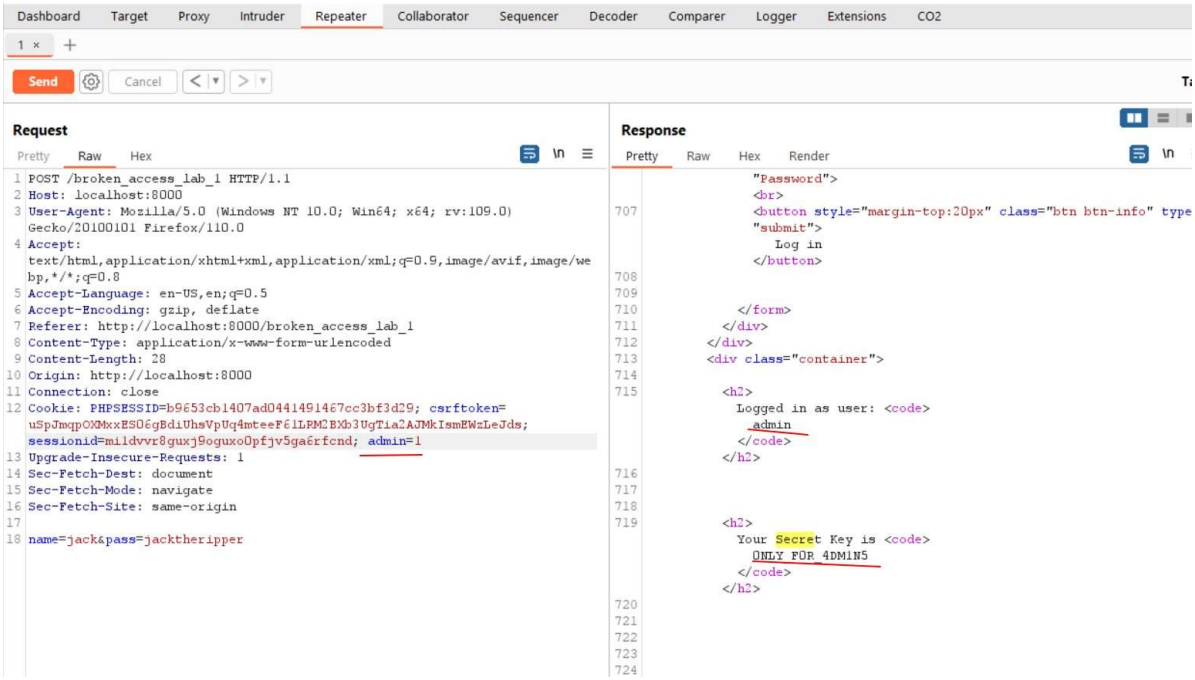


Đăng nhập với user: jack, password: jacktheripper

Response của server trả có thêm phần Set-cookie admin=0



Ta sửa thêm phần đó vào cookie (admin=1) Và lấy được secret key



Mức độ ảnh hưởng: nghiêm trọng, leo thang đặc quyền người dùng thông thường có thể thực hiện các hành động mà chỉ có Admin mới có thể thực hiện.

Đề xuất cách ngăn chặn:

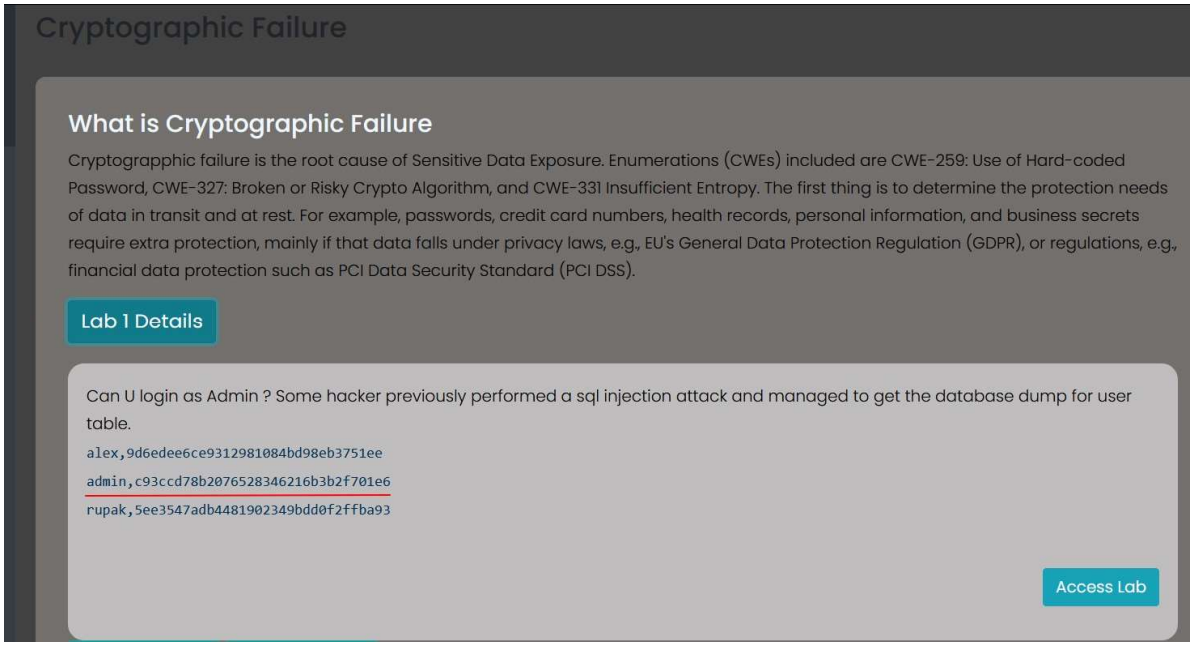
- Ngoại trừ các tài nguyên công khai, từ chối theo mặc định.
- Kiểm soát truy cập mô hình nên thực thi quyền sở hữu hồ sơ thay vì chấp nhận rằng người dùng có thể tạo, đọc, cập nhật hoặc xóa bất kỳ bản ghi nào.

Bài tập 3:

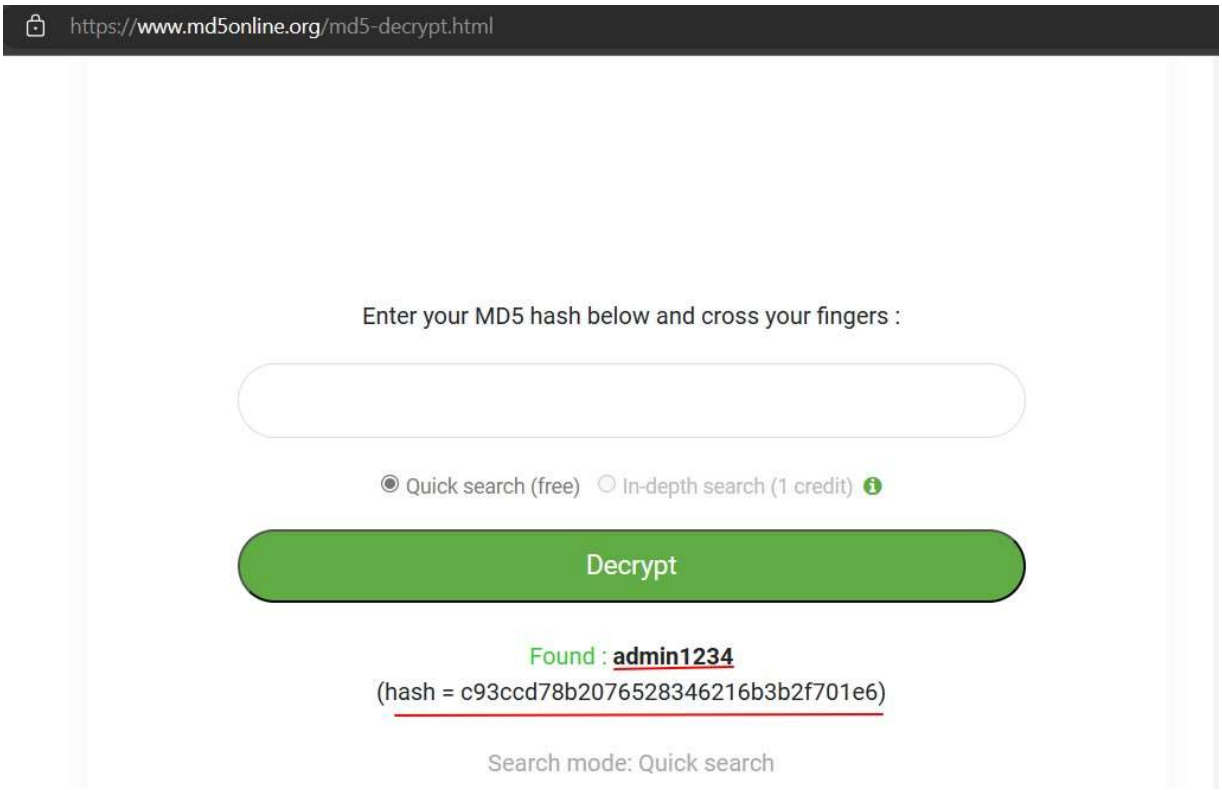
Tiêu đề: Mã hóa thất bại làm rò rỉ thông tin nhạy cảm của người dùng như mật khẩu, thông tin giao dịch…

Mô tả lỗ hổng: Sử dụng các thuật toán mã hóa kém an toàn để mã hóa, lỗi thời các thông tin quan trọng của người dùng. Đây là một trong những nguyên nhân của việc lộ các dữ liệu nhạy cảm quan trọng hoặc xâm phạm hệ thống.

DEMO:

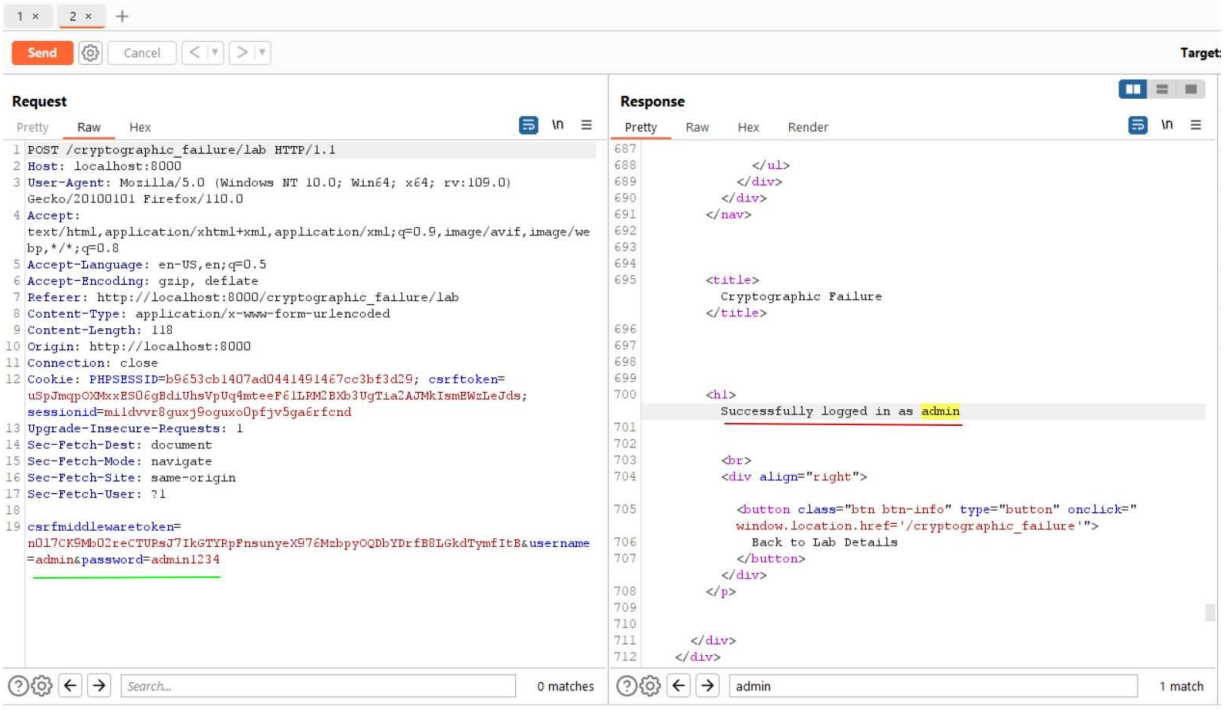


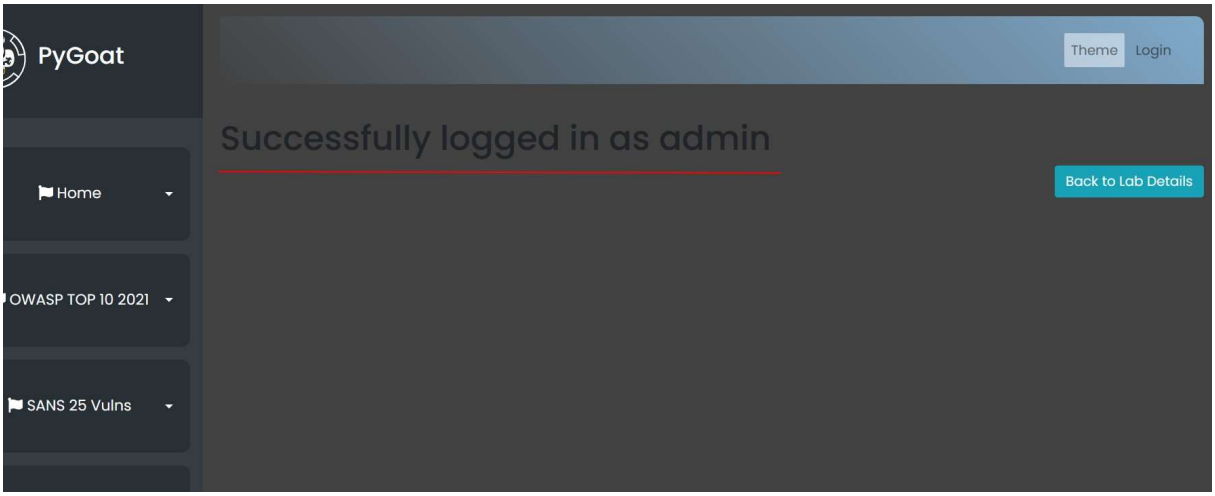
Ta đem phần password đã bị mã hóa đi giải mã



Kết quả thu được “admin1234”

Và ta dùng kết quả đó để đăng nhập





Mức độ ảnh hưởng: nghiêm trọng chiếm quyền điều khiển cảu các user, có được các thông tin nhạy cảm của người dùng.

Đề xuất giải pháp :

- Phân loại dữ liệu được xử lý, lưu trữ hoặc truyền bởi một ứng dụng.
- Xác định dữ liệu nào nhạy cảm theo luật riêng tư, yêu cầu quy định hoặc nhu cầu kinh doanh.
- Mã hóa tất cả dữ liệu trong quá trình vận chuyển với các giao thức an toàn như

TLS với mật mã Secrecy (FS) chuyển tiếp, ưu tiên mật mã của máy chủ và các tham số bảo

mật. Thực thi mã hóa bằng cách sử dụng các chỉ thị như HTTP Strict Transport Security (HSTS).

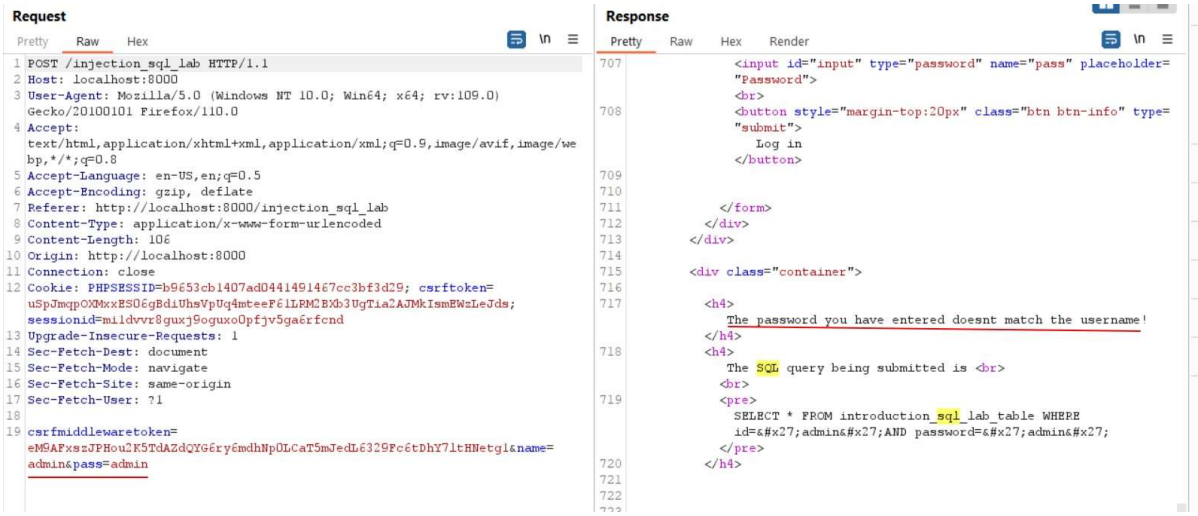
Bài tập 4:

Tiêu đề: SQL injection. Những kẻ tấn công có thể truy xuất và thay đổi dữ liệu, có nguy cơ hiển thị dữ liệu nhạy cảm được lưu trữ trên máy chủ SQL.

Mô tả lỗ hổng: Thay vì đăng nhập như người dùng bình thường attacker sẽ sử dụng các truy xuất Sql. SQL Injection thường xảy ra khi bạn yêu cầu người dùng đầu vào, như tên người dùng/userid của họ và thay vì tên/ID, người dùng cung cấp cho bạn câu lệnh SQL mà bạn sẽ vô tình chạy trên cơ sở dữ liệu của mình.

DEMO:

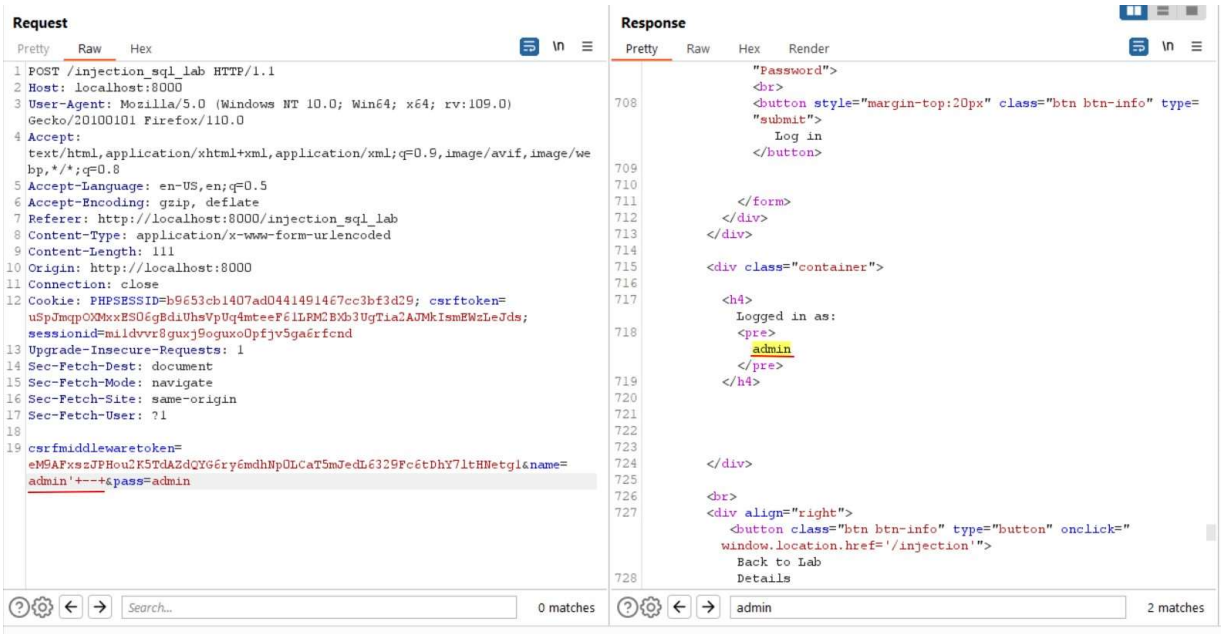
Đăng nhập như người dùng bình thường



Đăng nhập admin với giá trị nhập trường username= admin'+--+, còn trường password đã bị bỏ qua

Đánh giá mức độ ảnh hưởng: nghiêm trọng tác động đến dữ liêu trên sql Server, làm lộ hông tin của người dùng. Ăn cắp hoặc sao chép dữ liệu của trang web hoặc hệ thống. Người dùng có thể đăng nhập vào ứng dụng với tư cách người dùng khác, ngay cả với tư cách quản trị viên.

Khuyên cáo khắc phục: Luôn kiểm tra kỹ các trường nhập dữ liệu và các bạn cần ràng buộc thật kỹ dữ liệu người dùng nhập vào.


Request
Pretty Raw Hex
1 POST /injection_sql_lab HTTP/1.1
2 Host: localhost:8000
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/110.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://localhost:8000/injection_sql_lab
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 111
10 Origin: http://localhost:8000
11 Connection: close
12 Cookie: PHPSESSID=b9653cb1407ad04414914e67cc3bf3d29; csrftoken=uSpJmqpOXMxxESOégBdiUhsVpUq4mteeF61LRM2BXb3UgTia2AJMkIsmEWzLeJds; sessionid=mildvvr8guxj9oguxoOpfjv5ga6rfcnd
13 Upgrade-Insecure-Requests: 1
14 Sec-Fetch-Dest: document
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-User: ?1
18
19 csrfmiddlewaretoken=eM9AFxszJPHou2K5TdAZdQYGéryémdhNpOLCaT5mJedLé329Fc6tDhY7ltHNetg1&name=admin'+--+&pass=admin
Search...
0 matches
Response
Pretty Raw Hex Render
"Password">

<button style="margin-top:20px" class="btn btn-info" type="submit">
Log in
</button>
</form>

<h4>
Logged in as:
<pre>
admin
</pre>
</h4>

<button class="btn btn-info" type="button" onclick="window.location.href='/injection'">
Back to Lab
Details
admin
2 matches

Bài tập 5:

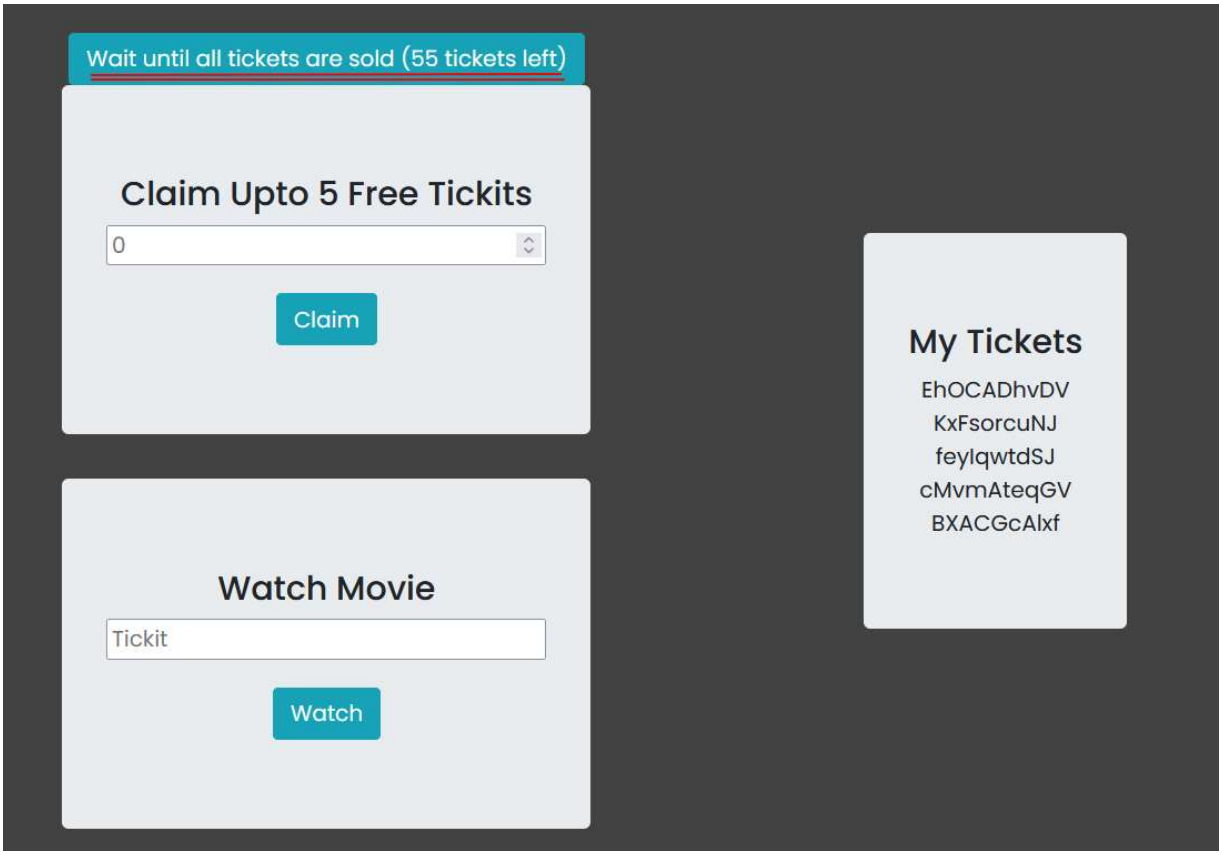
Tiêu đề: Insecure design gây ra các lỗ hổng bảo mật ảnh hưởng đến tài sản của các các cá nhân tổ chức.

Mô tả: Thiết kế không an toàn là một thể loại rộng đại diện cho các điểm yếu khác nhau, được thể hiện là thiết kế kiểm soát thiếu hoặc không hiệu quả dẫn đến các lỗ hổng khác nhau.

DEMO

Vì do không xác thực một người dùng chỉ được tạo một account vì vậy attacker đã tạo nhiều tài khoản để lấy hết các vé xem phim

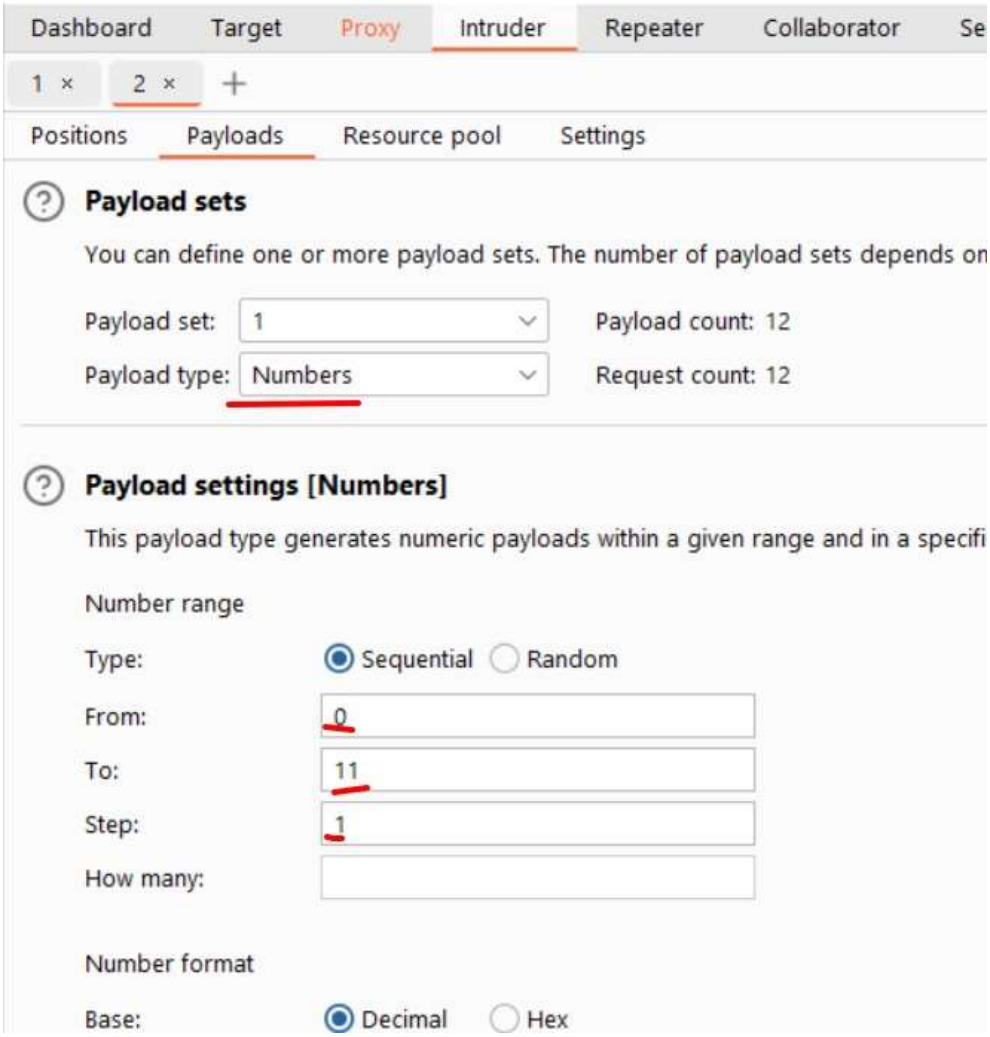
Một account lấy được 5 vé

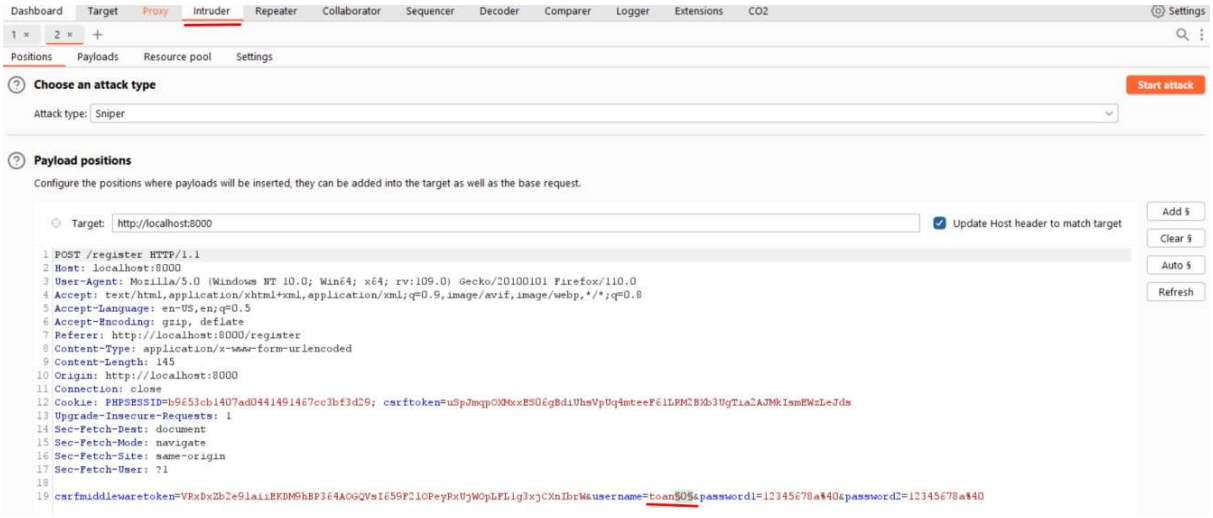


Còn lại 55 vé. Ta tạo thêm 11 tài khoản để lấy hết số vé còn lại. Ta bắt gói tin đăng ký tài khoản lại

```
POST /register HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/110.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8000/register
Content-Type: application/x-www-form-urlencoded
Content-Length: 145
Origin: http://localhost:8000
Connection: close
Cookie: PHPSESSID=b9653cb1407ad0441491467cc3bf3d29; csrftoken=uSpJmqpOXMxxES06gBdiUhsVpUq4mteeF61LRM2BXb3UgTia2AJMkIsmEWzLeJds
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

csrfmiddlewaretoken=VRxDxZb2e9laiiEKDM9hBP364AOGQVsI659F21OPeyRxUjWOpLFL1g3xjCXnIbrW&username=toan0&password1=12345678a%40&password2=12345678a%40
```

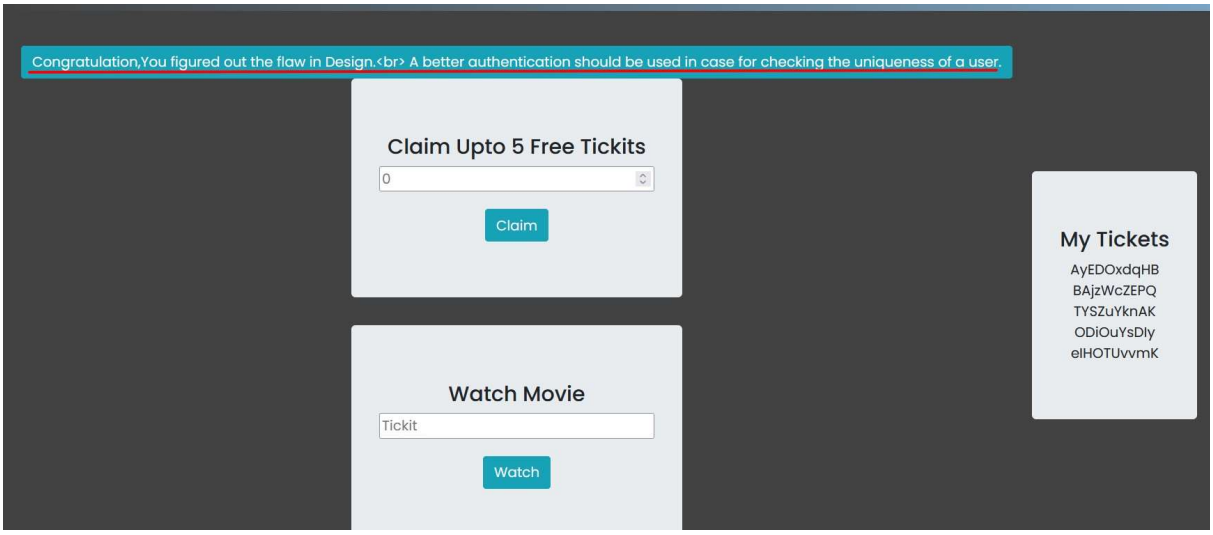
Đưa vào intruder để tạo nhiều tài khoản cho nhanh Với các tài khoản khác nhau cùng mật khẩu





Ta dùng 11 tài khoản mới tạo đó và lấy hết 55 vé còn

lại Sau khi lấy hết được vé



Ta thực hiện chức năng watch movie và nhận được dòng thông báo trên

Đánh giá: nghiêm trọng,phụ thuộc vào ngữ cảnh và code của ứng dụng attacker có thể đánh cắp thông tin nhạy cảm của người dùng, truy cập trái phép thực hiện các hành vi trái phép dưới quyền Admin.

Đề xuất giải pháp

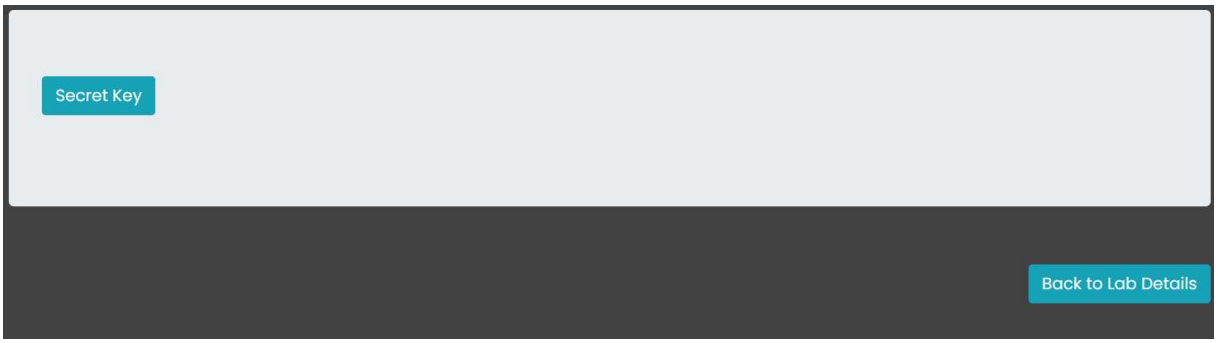
- Sử dụng các thư viện an toàn các thiết kế an toàn
- sử dụng các nguyên tắc thiết kế phần mềm SSDLC với các phần mềm đánh giá bảo mật và liên quan đến luồn thực thi

Bài tập 6:

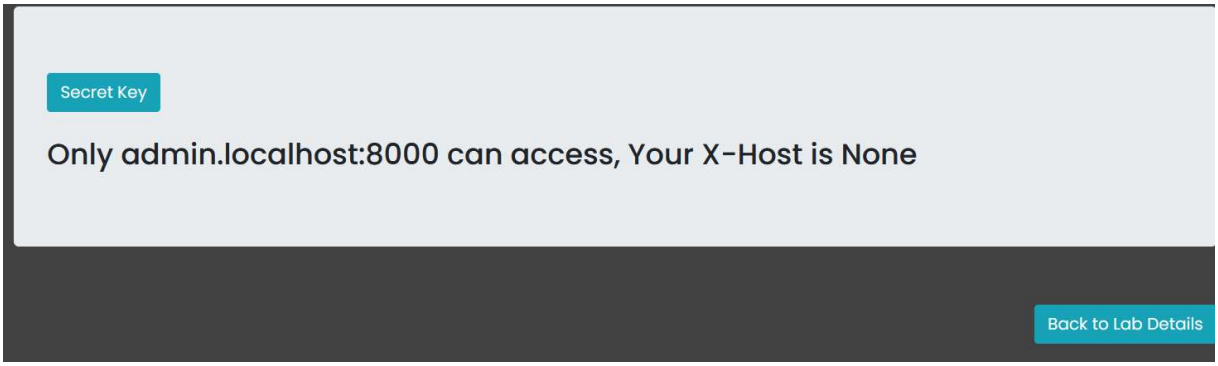
Tiêu đề: Leo quyền admin thông qua lỗ hổng ở X-Host trong Header

Mô tả: : Security Misconfiguration là lỗi bảo mật khi các tính năng không cần thiết được cài đặt, tài khoản mặc định và mật khẩu vẫn được bật và không thay đổi.

DEMO: ta muốn lấy secret key nhưng



Chỉ có admin.localhost:8000 mới có thể nhận được secret key

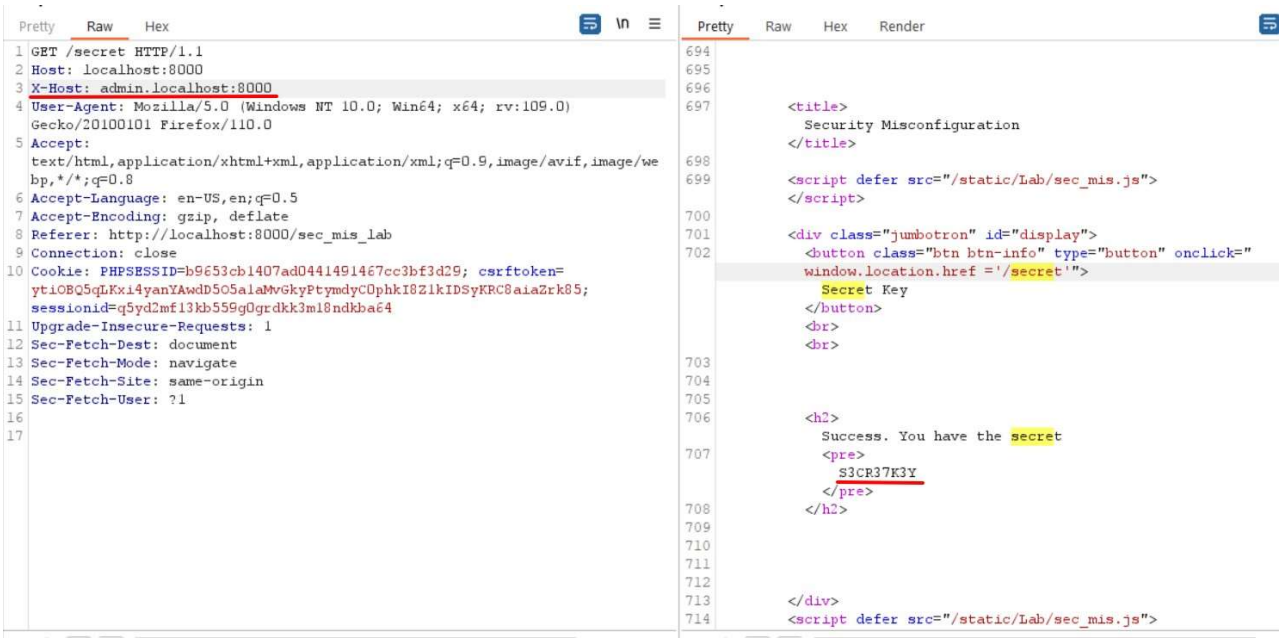


Va server trả lời là thiếu trường X-Host sau đó tìm hiểu ra thì X-Host nằm ở header gói tin gửi đi

```
GET /secret HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
Gecko/20100101 Firefox/110.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,imag
e/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8000/sec_mis_lab
Connection: close
Cookie: PHPSESSID=b9653cb1407ad044149146 7cc3bf3d29; csrftoken=
ytiOBQ5qLKxi4yanYAwdD5O5alaMvGkyPtymdyC0phkI8Z1kIDSyKRC8aiaZrk85;
sessionid=q5yd2mf13kb559g0grdkk3ml8ndkba64
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
```

```
<div class="jumbotron" id="display">
  <button class="btn btn-info" type="button" onclick="
  window.location.href ='/secret'">
    Secret Key
  </button>
  <br>
  <br>

  <h3>
    Only admin.localhost:8000 can access, Your X-Host is
    None
  </h3>

</div>
<script defer src="/static/Lab/sec_mis.js">
</script>
```

Ta thêm trường header X-Host: admin.localhost:8000 và chạy



Kết quả thu được secret key

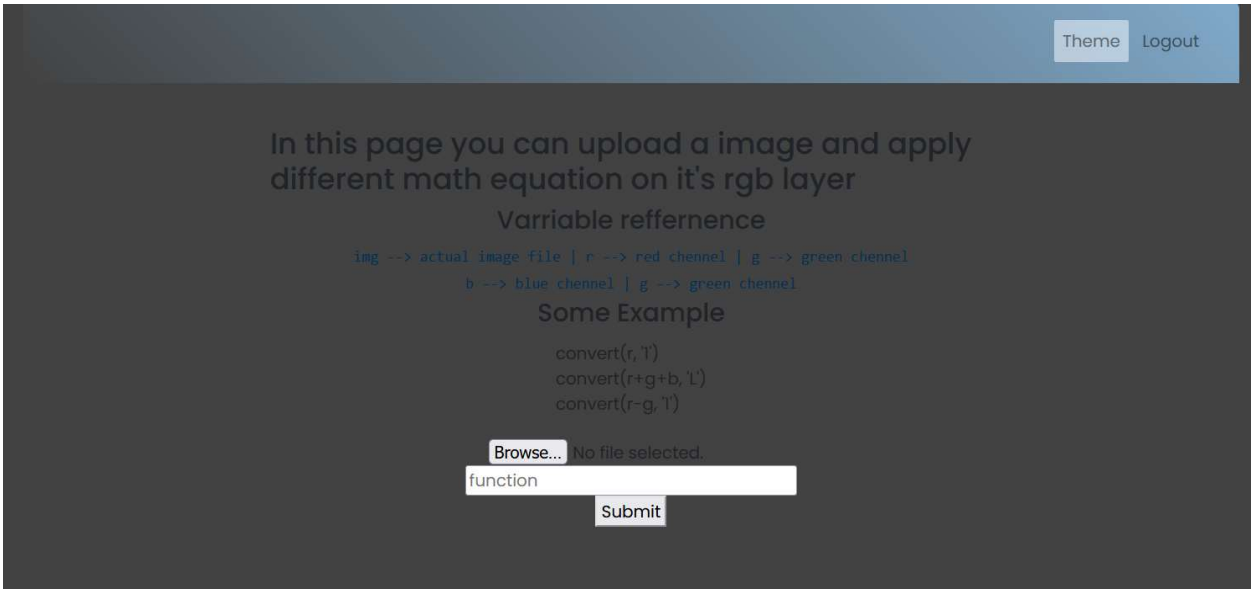
Đánh giá: Nghiêm trọng, attacker có thể tấn công dựa vào các lỗ hổng khi các tính năng không cần thiết được cài đặt.

Đề xuất giải pháp:

- Xóa hoặc không cài đặt các tính năng và khung không sử dụng.
- Một nhiệm vụ để xem xét và cập nhật các cấu hình phù hợp với tất cả các ghi chú bảo mật, cập nhật và bản vá như một phần của quy trình quản lý bản vá.
- Gửi chỉ thị bảo mật cho khách hàng, ví dụ: tiêu đề bảo mật.

1. http://localhost:8000/a9_lab2

ban đầu



Đọc code bên dưới máy chủ là

```
def a9_lab2(request):
if not request.user.is_authenticated:
 return redirect('login')

if request.method == "GET":
 return render (request,"Lab/A9/a9_lab2.html")
elif request.method == "POST":
 try :
  file=request.FILES["file"]
  function_str = request.POST.get("function")
  img = Image.open(file)
  img = img.convert("RGB")
  r,g,b = img.split()
  output = ImageMath.eval(function_str,img = img, b=b, r=r, g=g)
  # saving the image
  buffered = BytesIO()
  output.save(buffered, format="JPEG")
  img_str = base64.b64encode(buffered.getvalue()).decode("utf-8")
  bufferd_ref = BytesIO()
  img.save(bufferd_ref, format="JPEG")
  img_str_ref = base64.b64encode(bufferd_ref.getvalue()).decode("utf-8")
  try :
   return render(request,"Lab/A9/a9_lab2.html",{"img_str": img_str,"img_str_ref":img_str_ref, "success": True})
  except Exception as e:
   print(e)
   return render(request, "Lab/A9/a9_lab2.html", {"data": "Error", "error": True})
 except Exception as e:
  print(e)
  return render(request, "Lab/A9/a9_lab2.html", {"data":"Please Upload a file", "error":True})
```

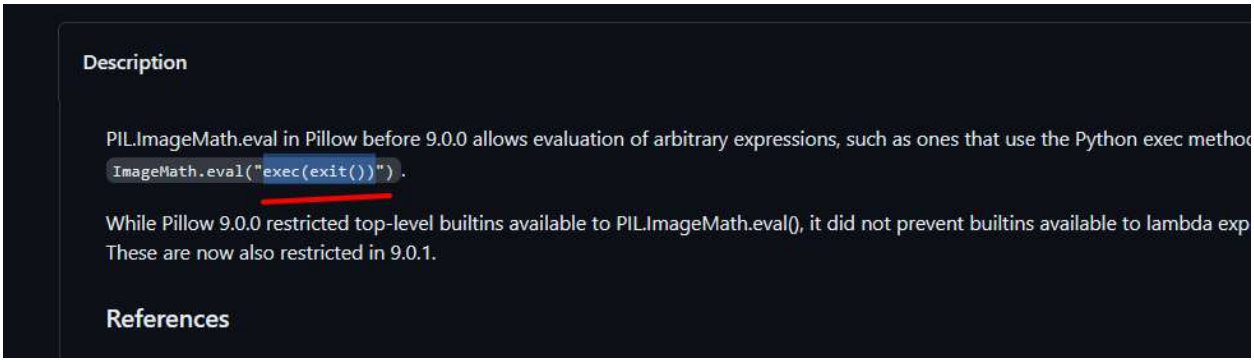
Hint View Code

Ta nhìn thấy đầu ra output

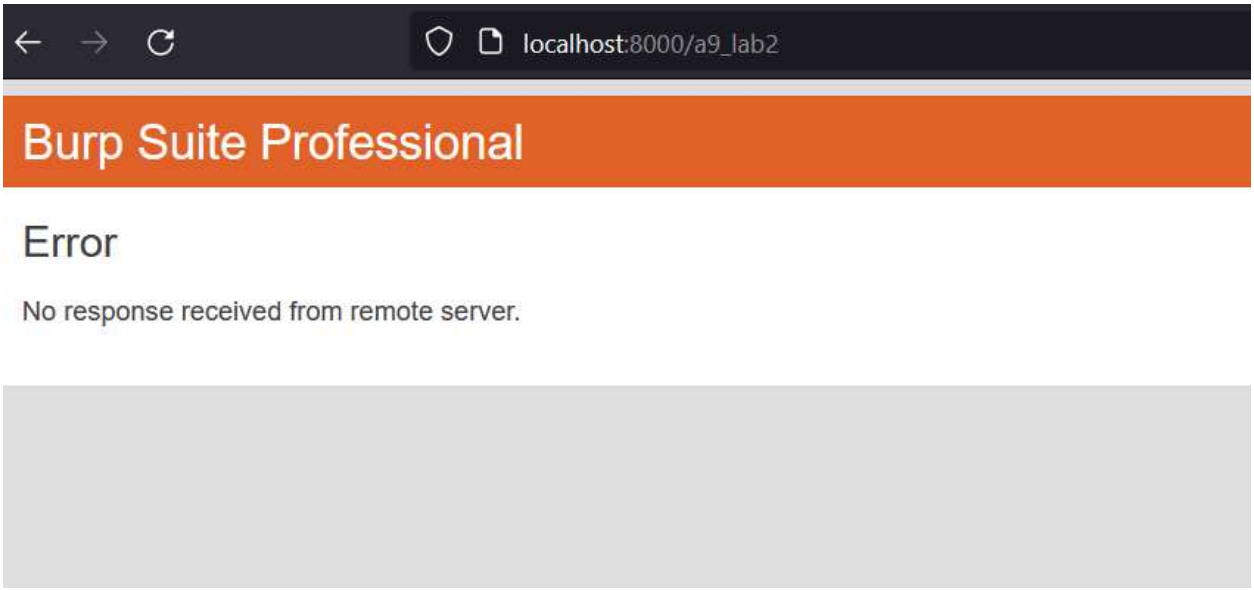
```
try :
 file=request.FILES["file"]
 function_str = request.POST.get("function")
 img = Image.open(file)
 img = img.convert("RGB")
 r,g,b = img.split()
 output = ImageMath.eval(function_str,img = img, b=b, r=r, g=g)
 # saving the image
 buffered = BytesIO()
 output.save(buffered, format="JPEG")
 img_str = base64.b64encode(buffered.getvalue()).decode("utf-8")
 bufferd_ref = BytesIO()
 img.save(bufferd_ref, format="JPEG")
 img_str_ref = base64.b64encode(bufferd_ref.getvalue()).decode("utf-8")
```

Tìm kiếm ta thấy một CVE 2022-22817
Hướng dẫn về cách khai thác lỗi ở hàm này

Ta chạy lệnh này



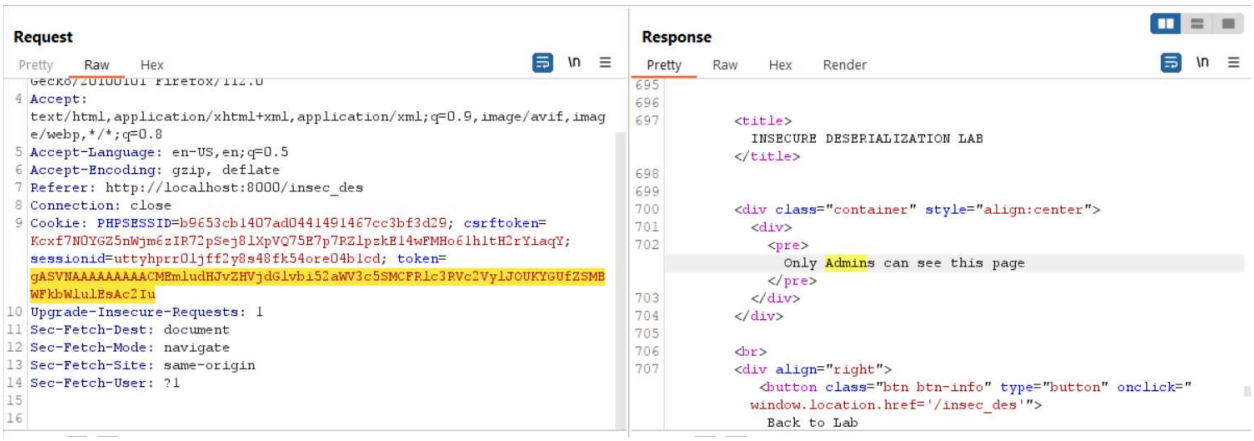
Nó đã thoát khỏi server và kết quả trả về như sau



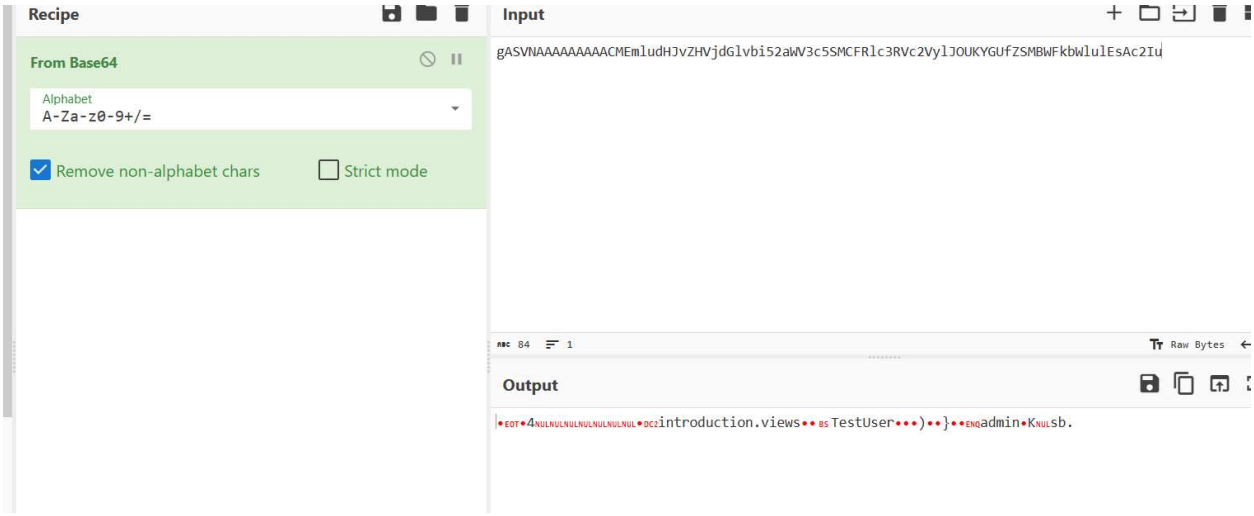
2. http://localhost:8000/insec_des_lab

ban đầu

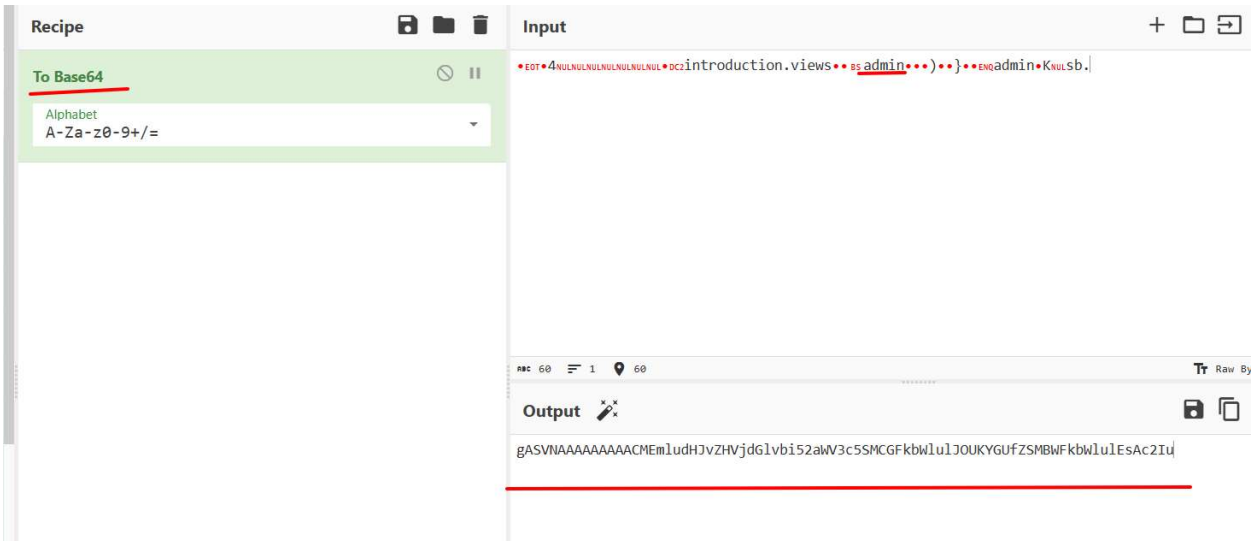
Ta xem trong trang web có gì để ta khai thác
Thì ta thấy có cookie có phần token có gì đó khác biệt thay đổi
Vì vậy



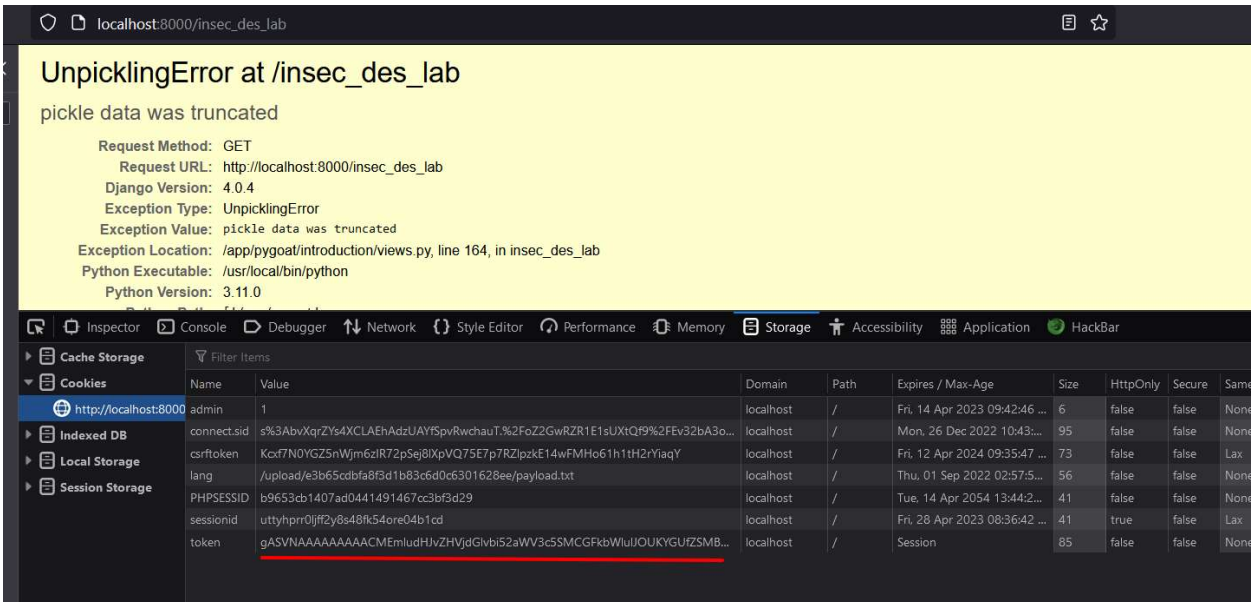
Ta đem phần đó để decode base64



Đây là kết quả ta thử sửa đoạn TestUser thành admin xem có kết quả như thế nào khác không



Ta thay thế vào trường token trong cookie

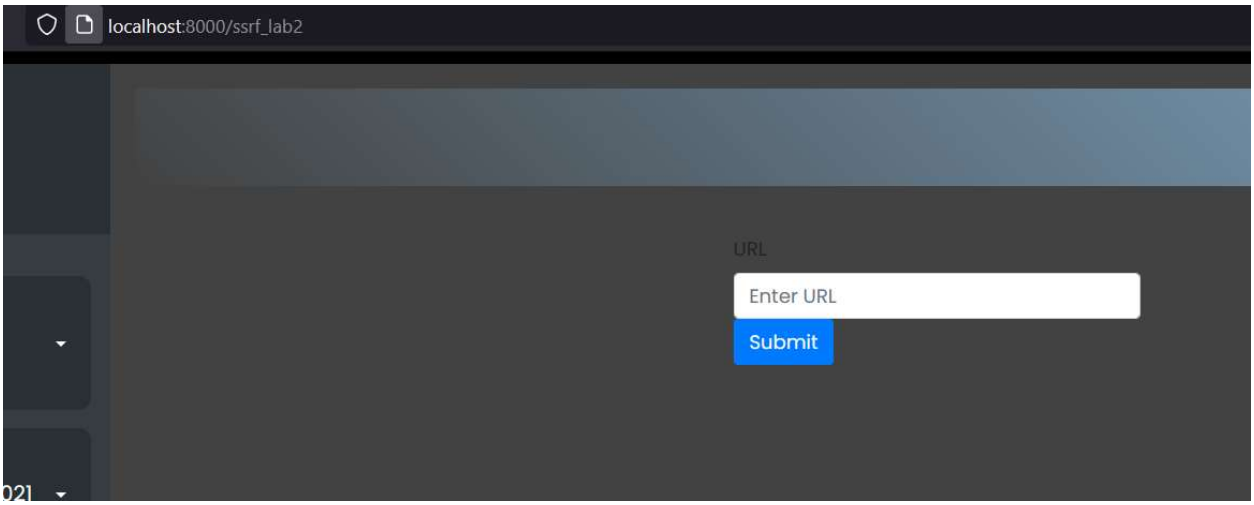


và đây là kết quả thu được

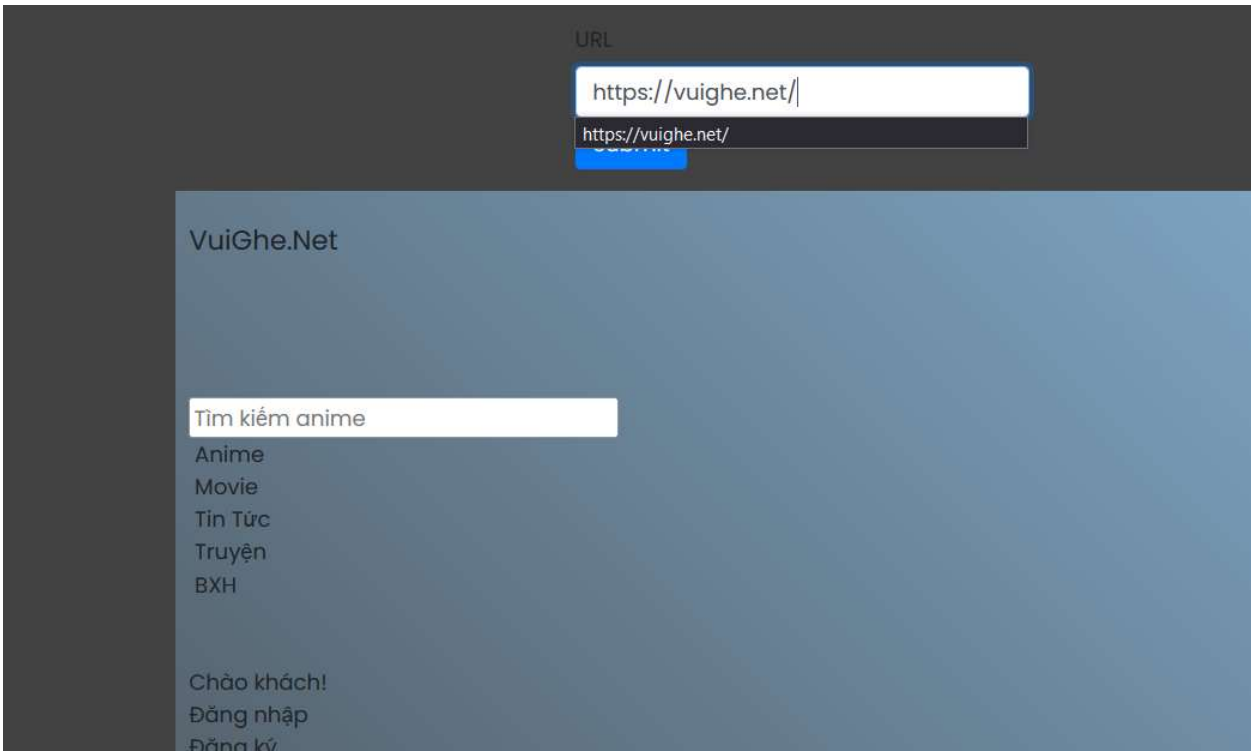
Tìm tòi một thời gian thì ta đọc được SECRETKEY:ADMIN123

3. http://localhost:8000/ssrf_lab2

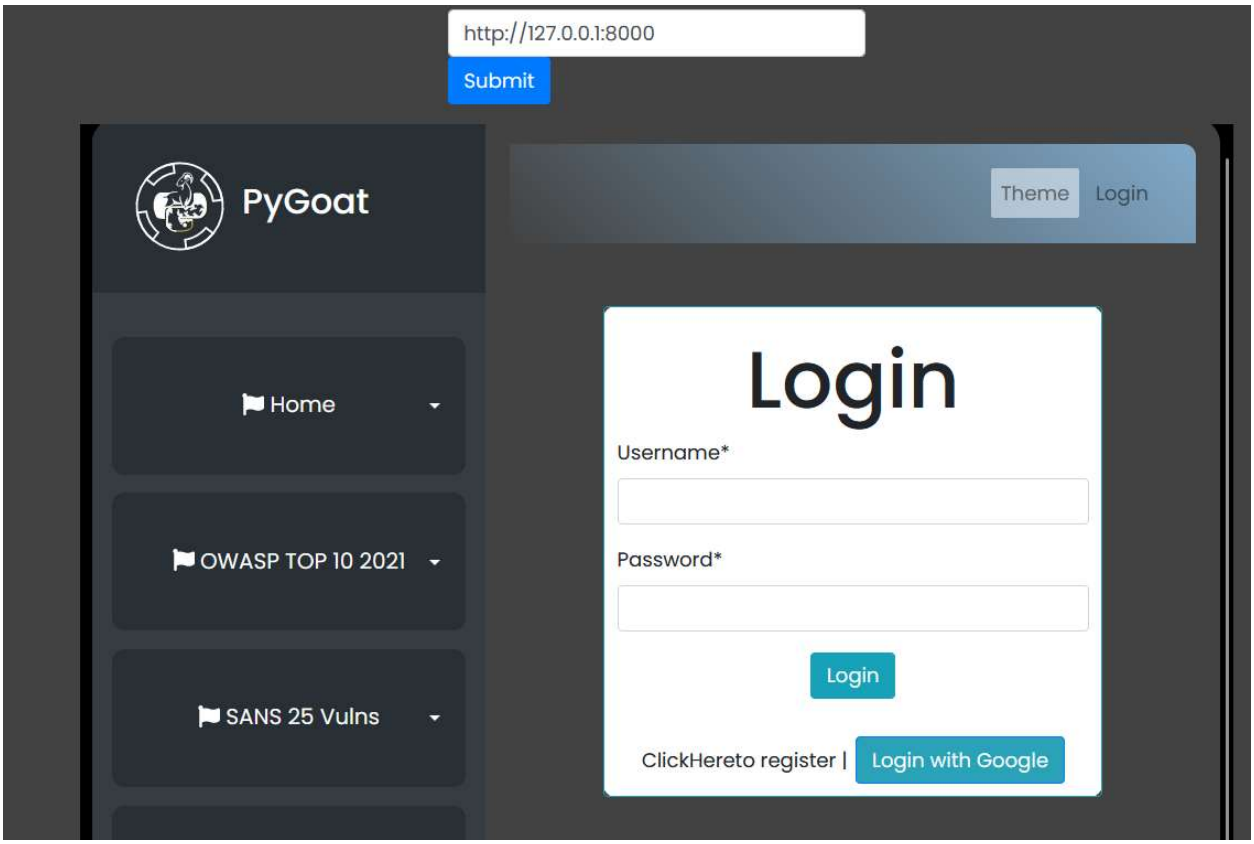
Ban đầu



Ta nhập các đường dẫn vào

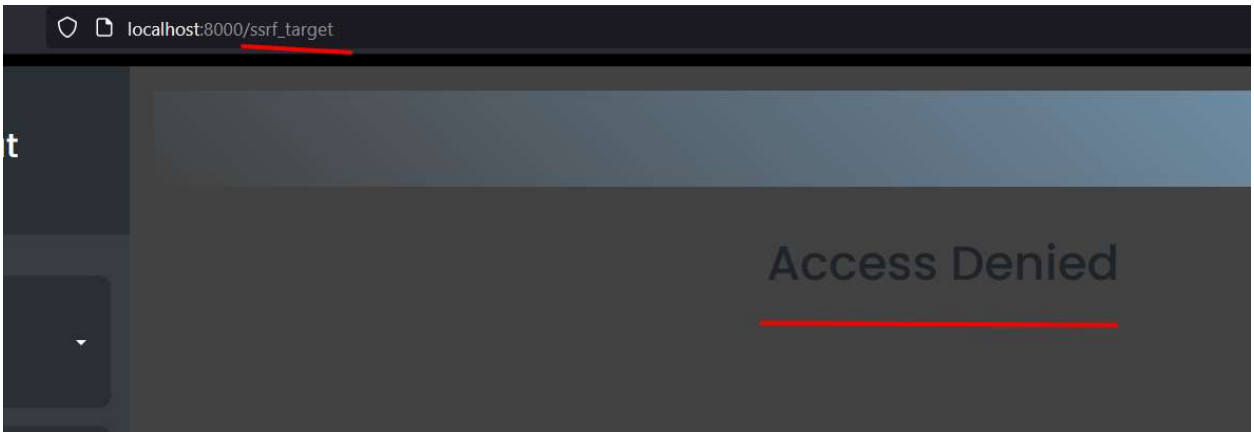


Sau một thời gian suy nghĩ sao ta không thử nhập url của máy chủ vào

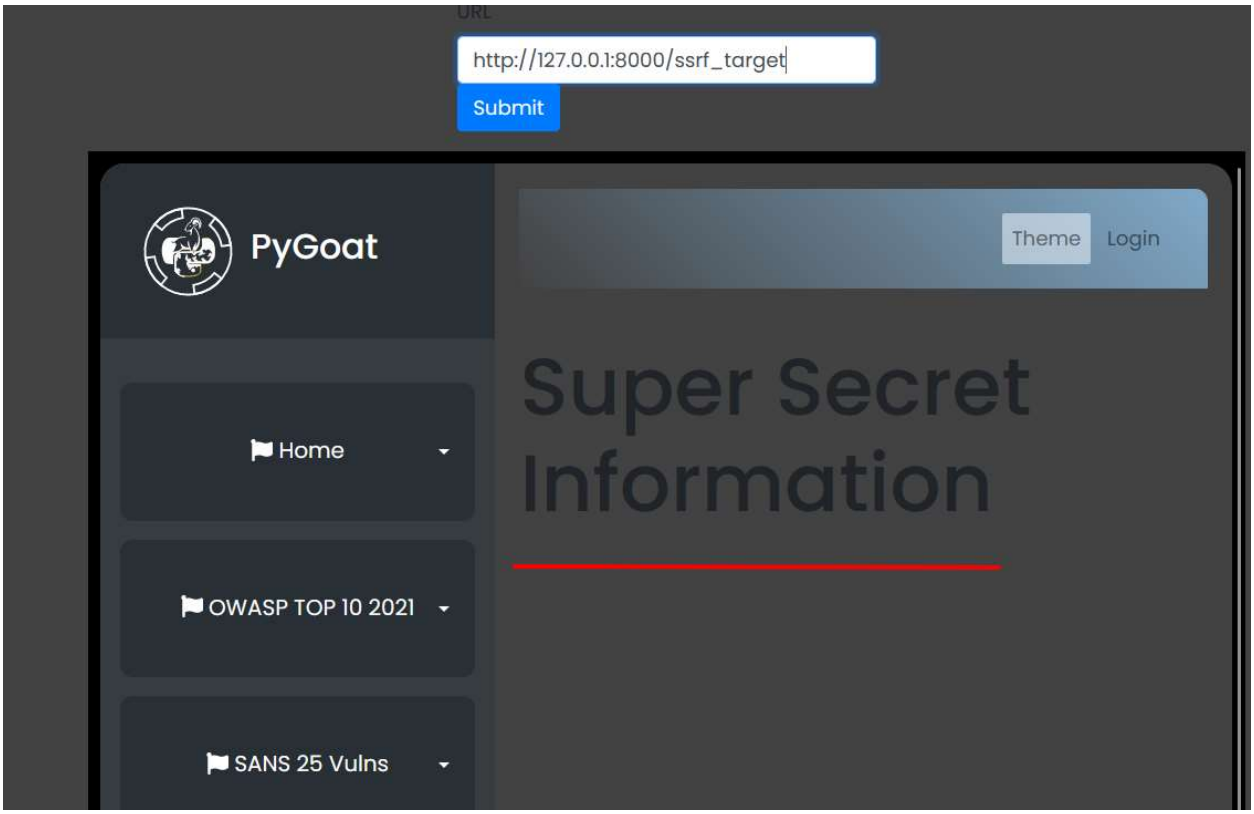


Và nó thực thi được này

Giờ ta vào thử đường dẫn ssrf_target



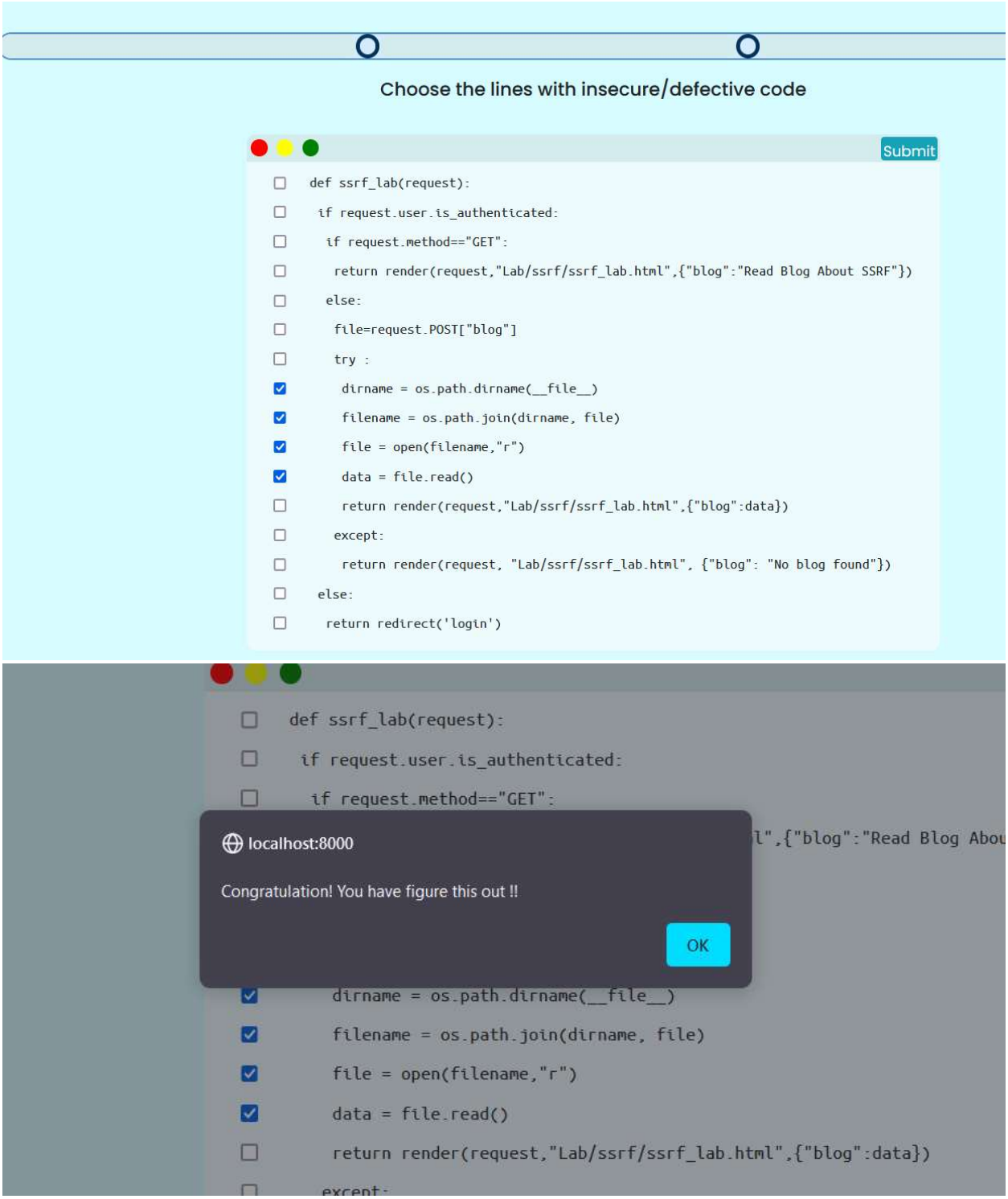
Nhưng nó đã bị chặn, giờ ta lợi dụng hướng suy nghĩ vừa rồi để truy cập đường dẫn đó xem sao



Nó đã thành công

4. http://localhost:8000/ssrf_discussion

Lỗi xảy ra ở các dòng sau



Và ở phía front-end có các dòng code không an toàn sau

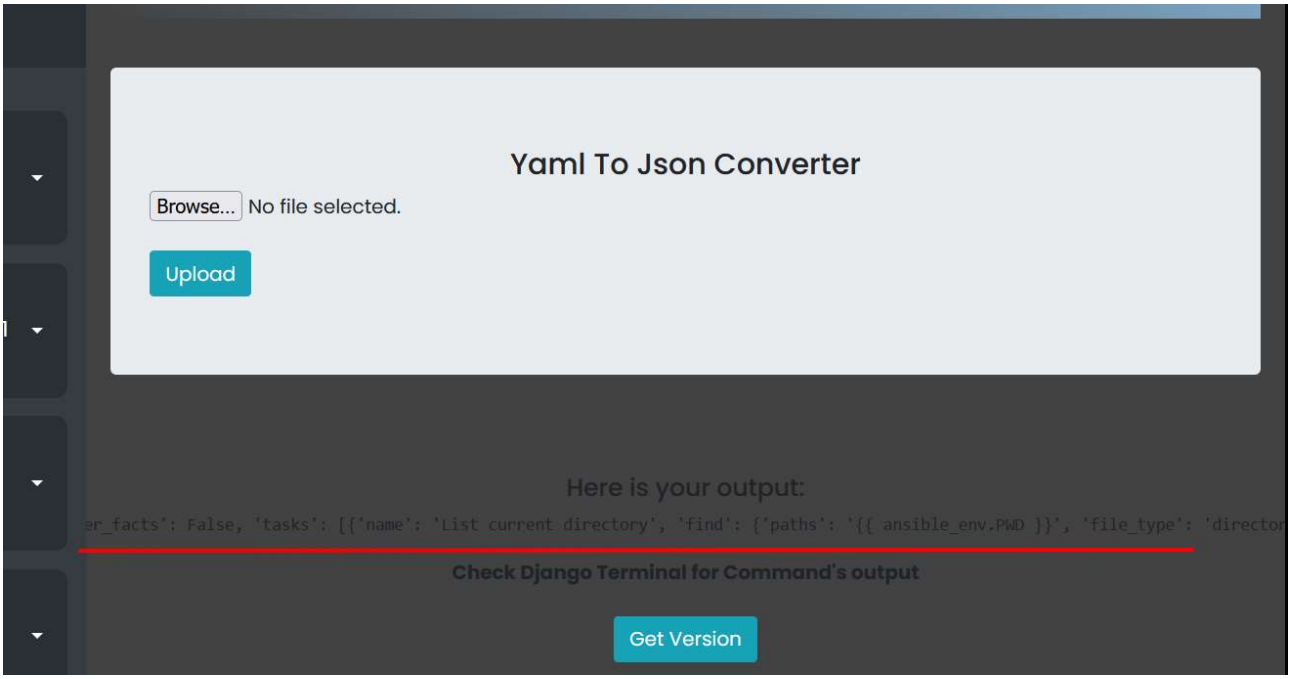
Some insecure codes in frontend side also ...

Submit

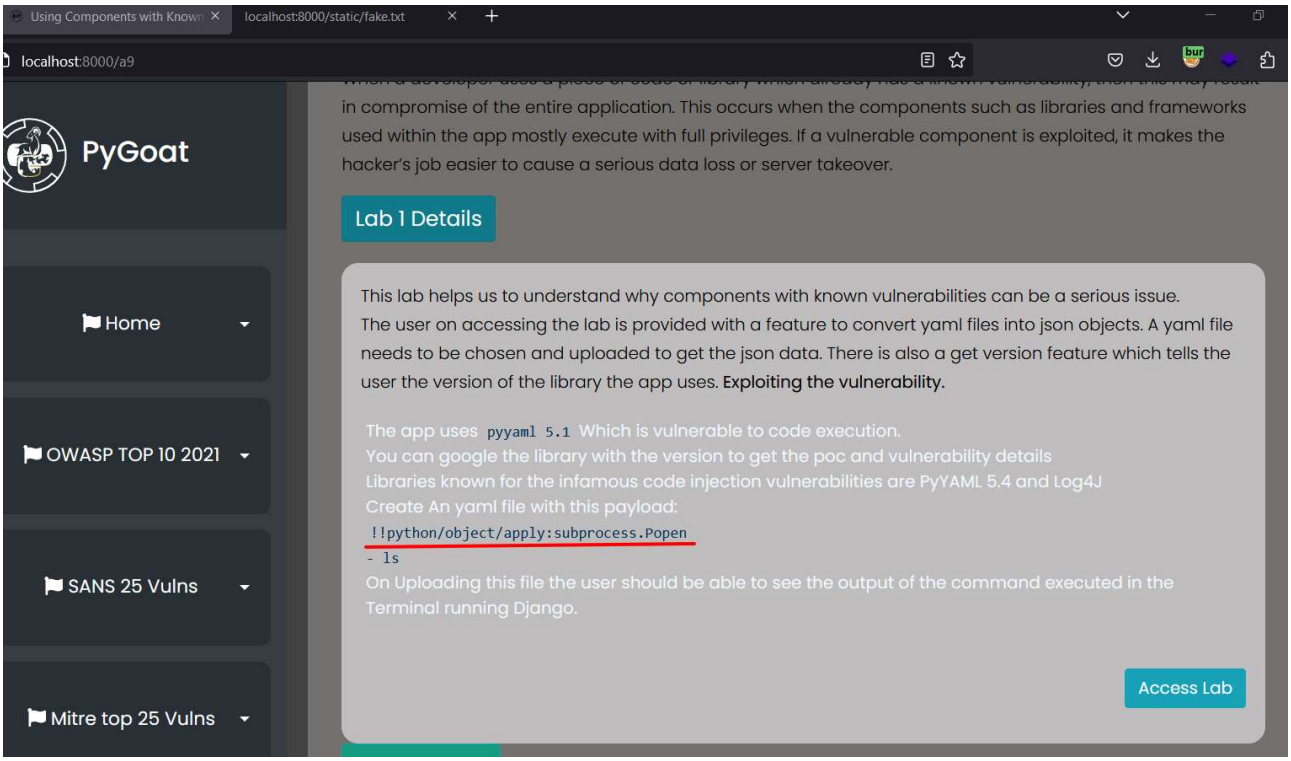
```
<div>
 <form method="post" action="/ssrf_lab">
  <input type="hidden" name="blog" value="templates/Lab/ssrf/blogs/blog1.txt">label>
  <button type="submit" class="btn btn-info"> Blog1 </button>
 </form>
 <form method="post" action="/ssrf_lab">
  <input type="hidden" name="blog" value="templates/Lab/ssrf/blogs/blog2.txt">label>
  <button type="submit" class="btn btn-info"> Blog1 </button>
 </form>
 <form method="post" action="/ssrf_lab">
  <input type="hidden" name="blog" value="templates/Lab/ssrf/blogs/blog3.txt">label>
  <button type="submit" class="btn btn-info"> Blog1 </button>
 </form>
 <form method="post" action="/ssrf_lab">
  <input type="hidden" name="blog" value="templates/Lab/ssrf/blogs/blog4.txt">label>
  <button type="submit" class="btn btn-info"> Blog1 </button>
 </form>
</div>
```

<form method="post" action="/ssrf_lab">
<input type="hidden" name="blog" value="templates/Lab/ssrf/blogs/blog1.txt">label>
<button type="submit" class="btn btn-info"> Blog1 </button>
</form>
localhost:8000
Congratulation! you have detected defective codes in html
OK
tes/Lab/ssrf/blogs/blog2.txt">label>
og1 </button>
<form method="post" action="/ssrf_lab">
<input type="hidden" name="blog" value="templates/Lab/ssrf/blogs/blog3.txt">label>
<button type="submit" class="btn btn-info"> Blog1 </button>
</form>
<form method="post" action="/ssrf_lab">
<input type="hidden" name="blog" value="templates/Lab/ssrf/blogs/blog4.txt">label>
<button type="submit" class="btn btn-info"> Blog1 </button>
</form>

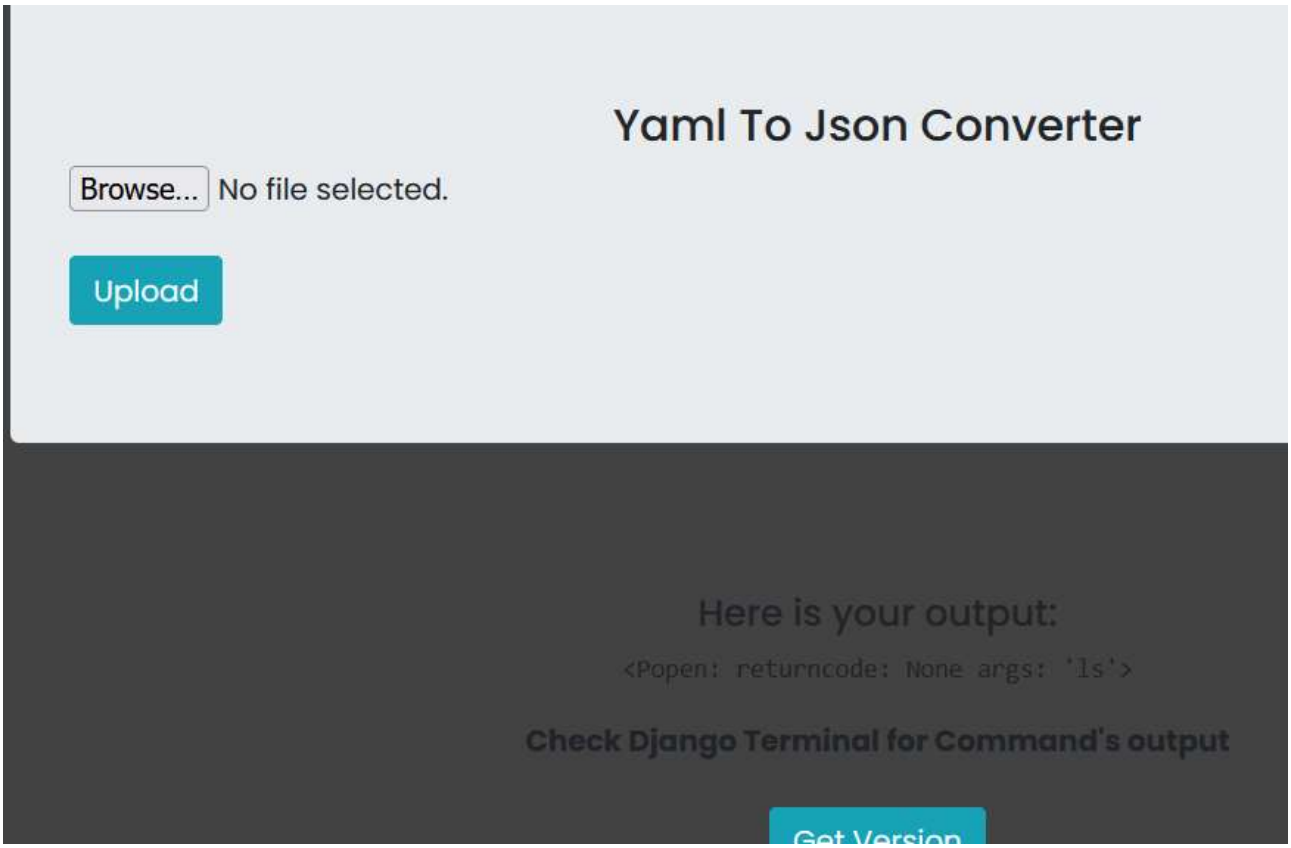
A6:2021–Vulnerable and Outdated Components



Khi ta upload một file bình thường thì kết quả sẽ như thế này

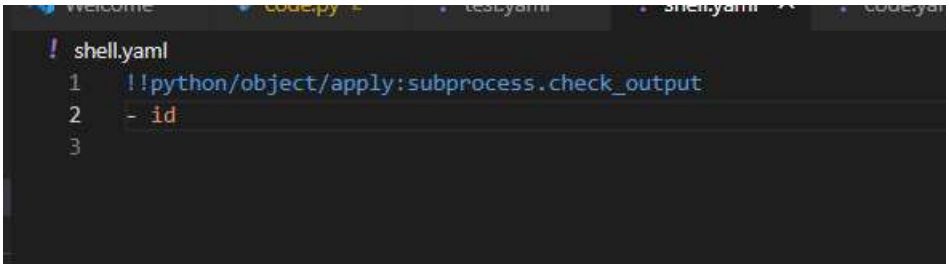


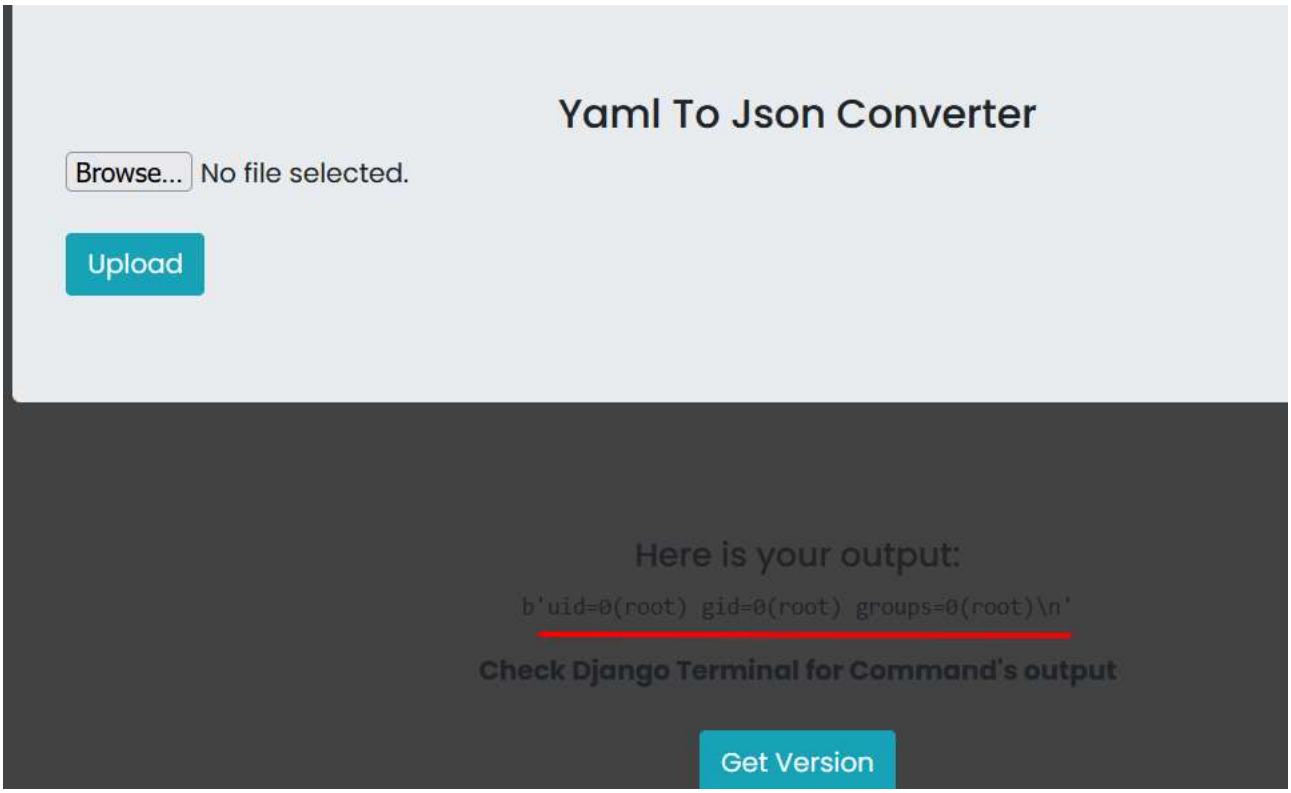
Có hướng dẫn ta làm theo



Nhưng có gì đó sai ta thử với id xem

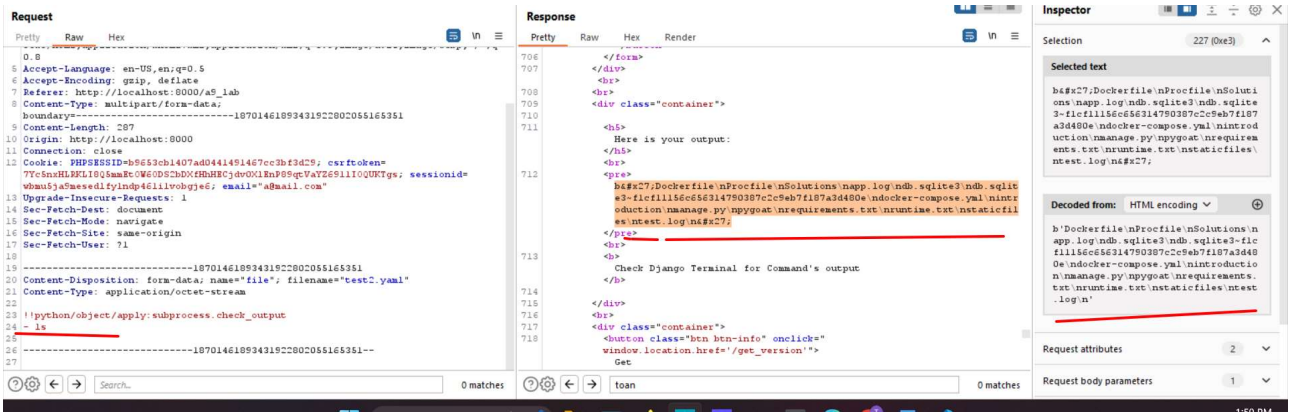
Khi kia là Open ta sửa lại thành check_output





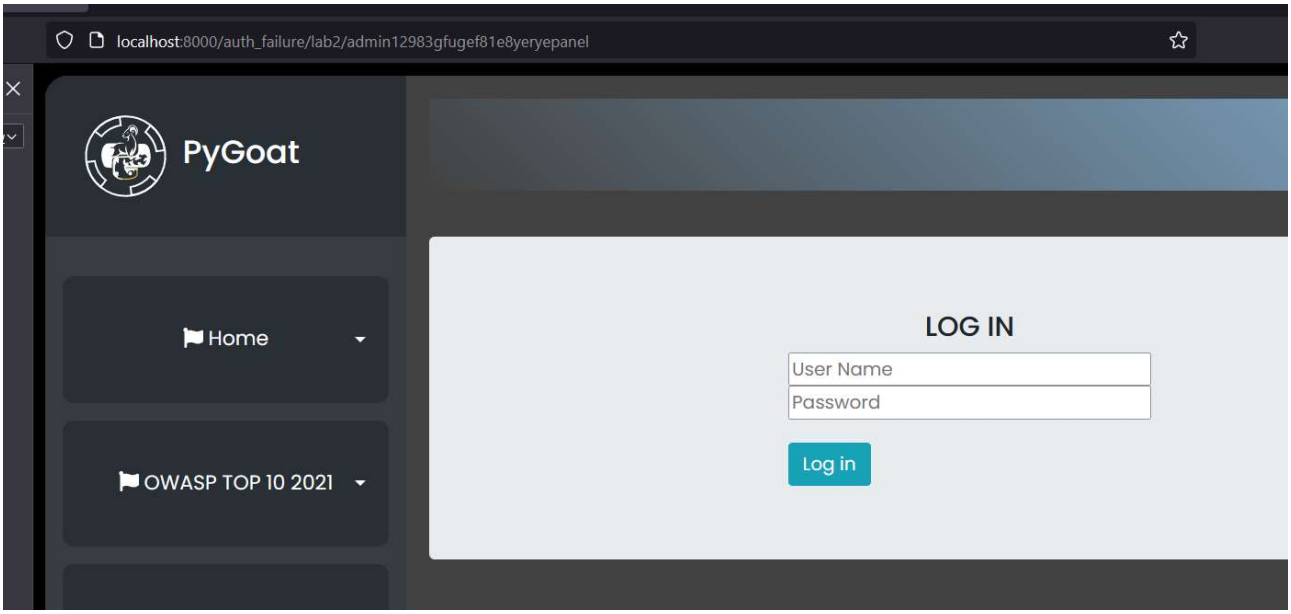
nó đã check được id

Sau đó với shell code khác nhau, ta làm được như này

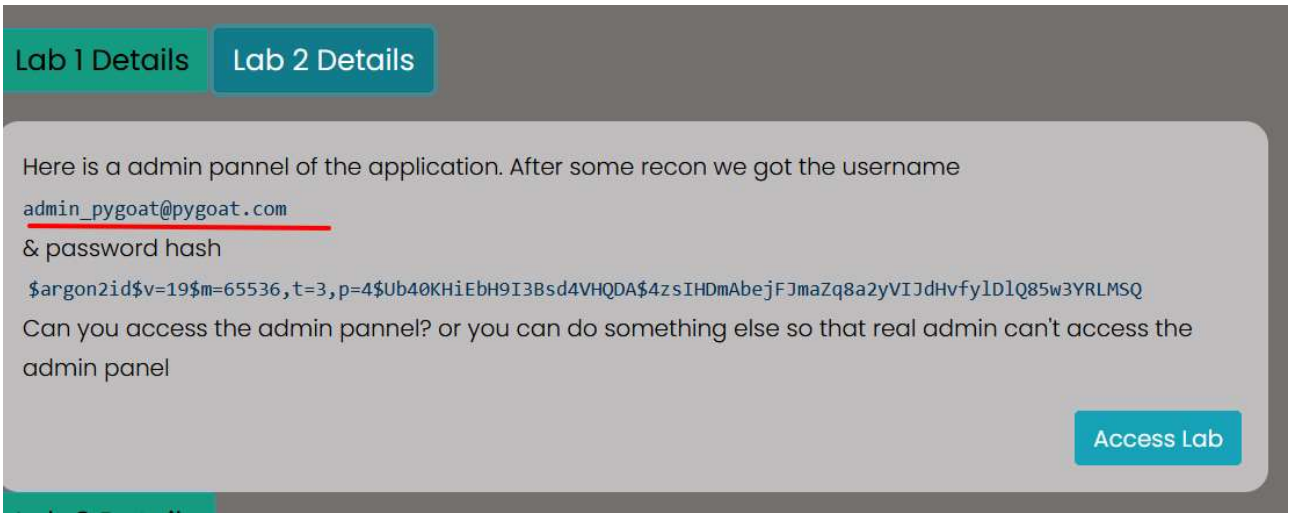


Đọc được các file của máy chủ

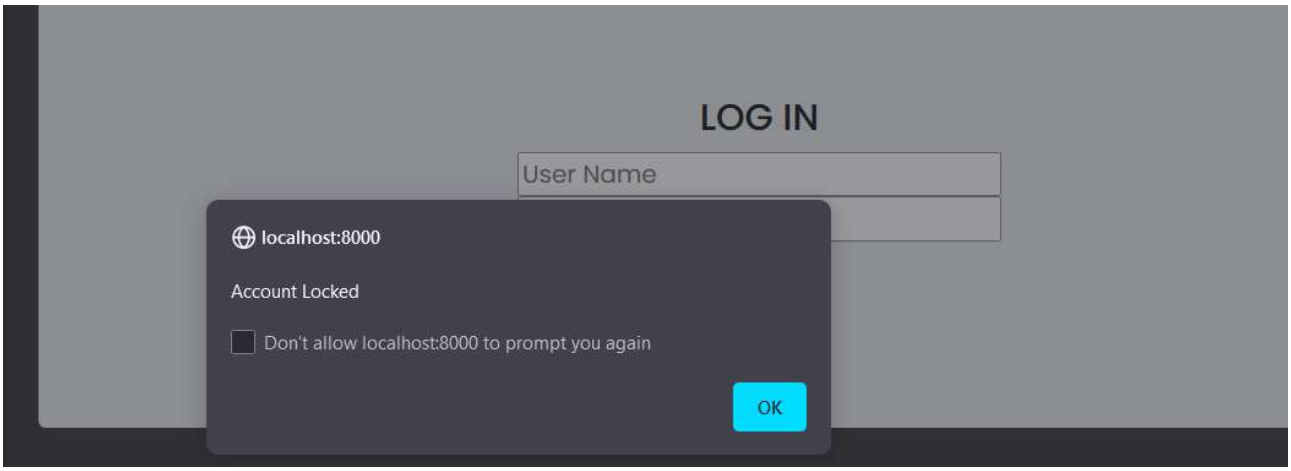
A7: 2021 – Identification and Authentication Failures



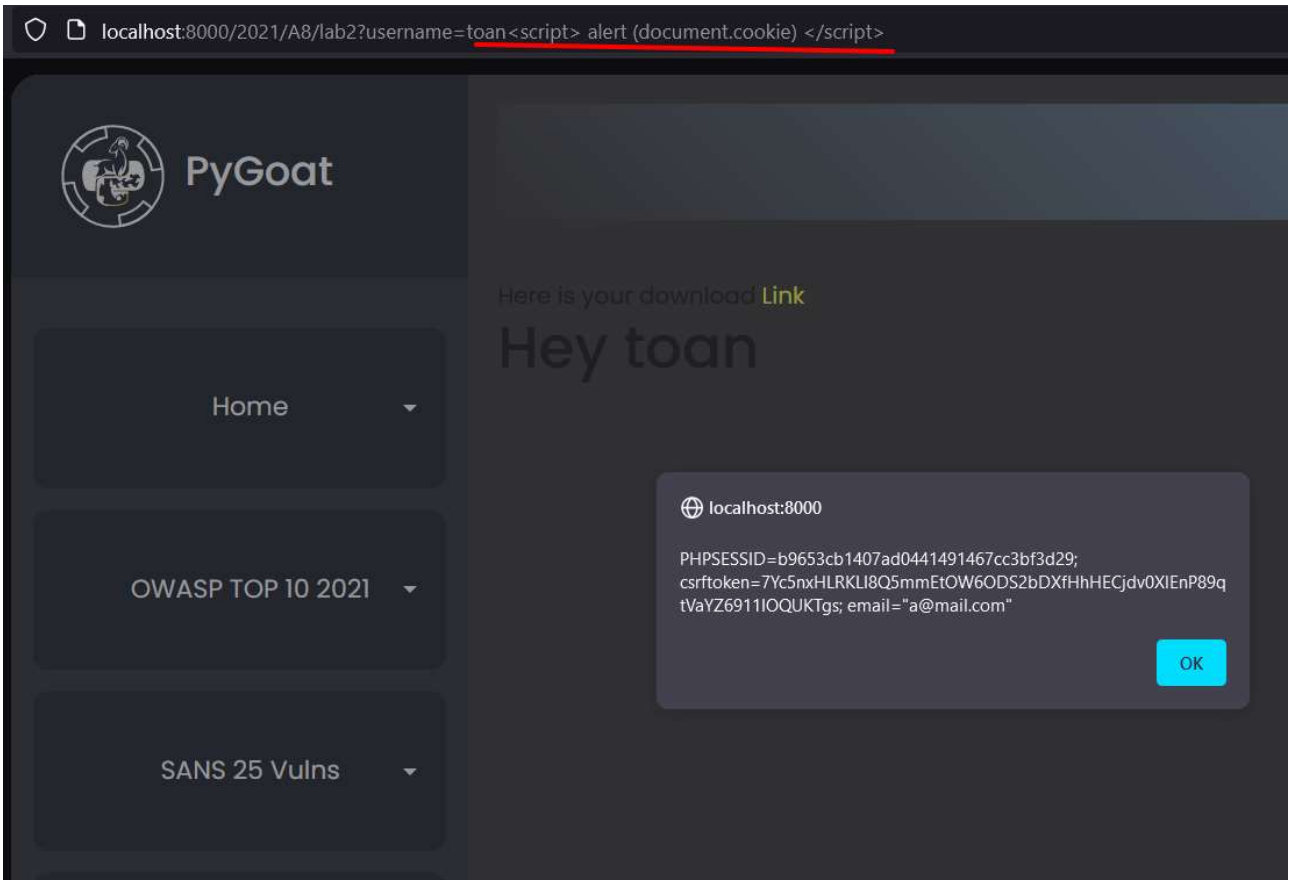
Đối với các tài khoản khác thì ta nhập bao nhiêu lần cũng được
Nhưng đối với tài khoản admin do máy chủ cung cấp



Sau khi đăng nhập sai quá 5 lần thì tên tài khoản admin bị chặn khỏi đăng nhập máy chủ



A8: 2021 – Security Logging and Monitoring Failures

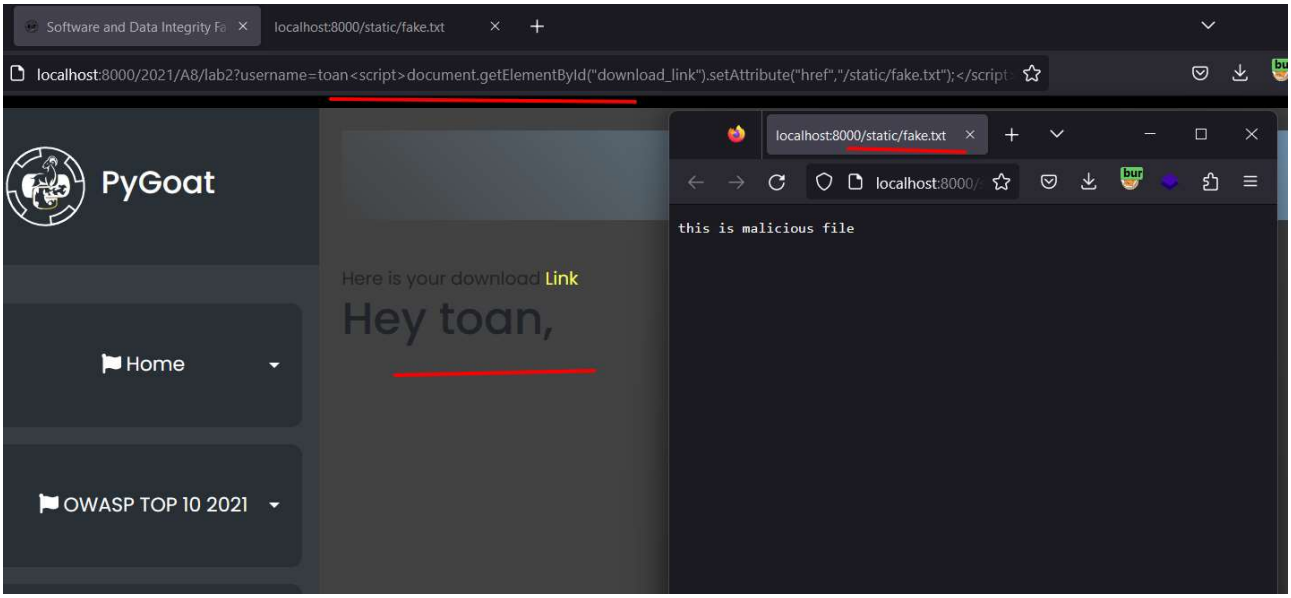


Bị Xss nên ta có thể chuyển hướng đến đường dẫn khác

```
Here is your download <a id="download_link" href="
/static/real.txt" style="color:#ff5" download>
  Link
</a>
```

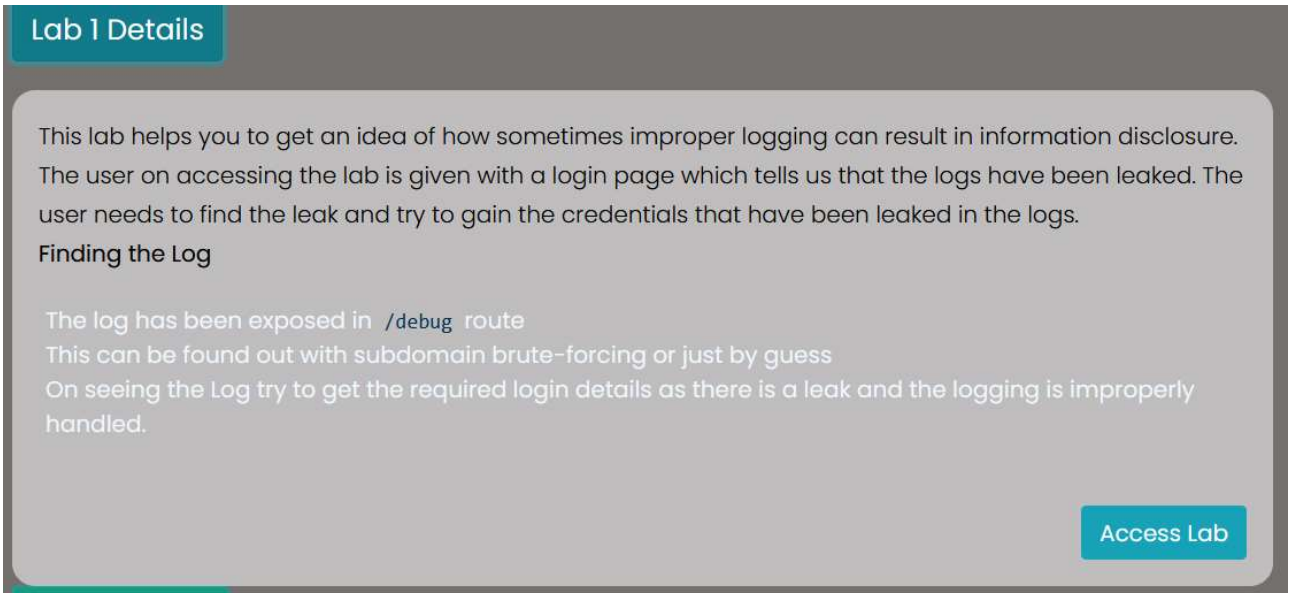
Do id và href có sẵn nên ta sửa theo và thêm script vào

Payload: toan
<script>document.getElementById("download_link").setAttribute("href","/static/fake.txt");</script>

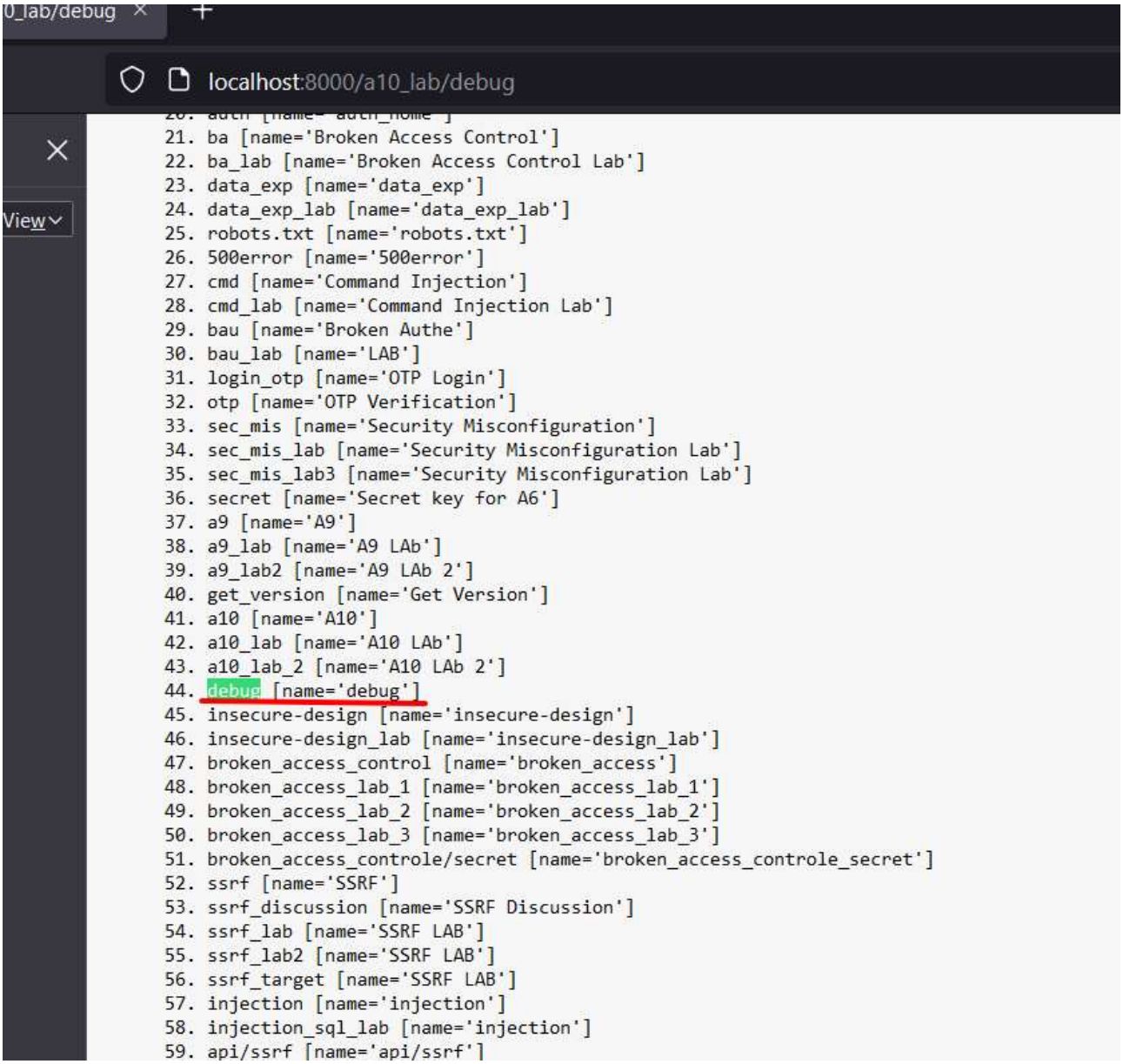


Và đây khi nó tạo ra

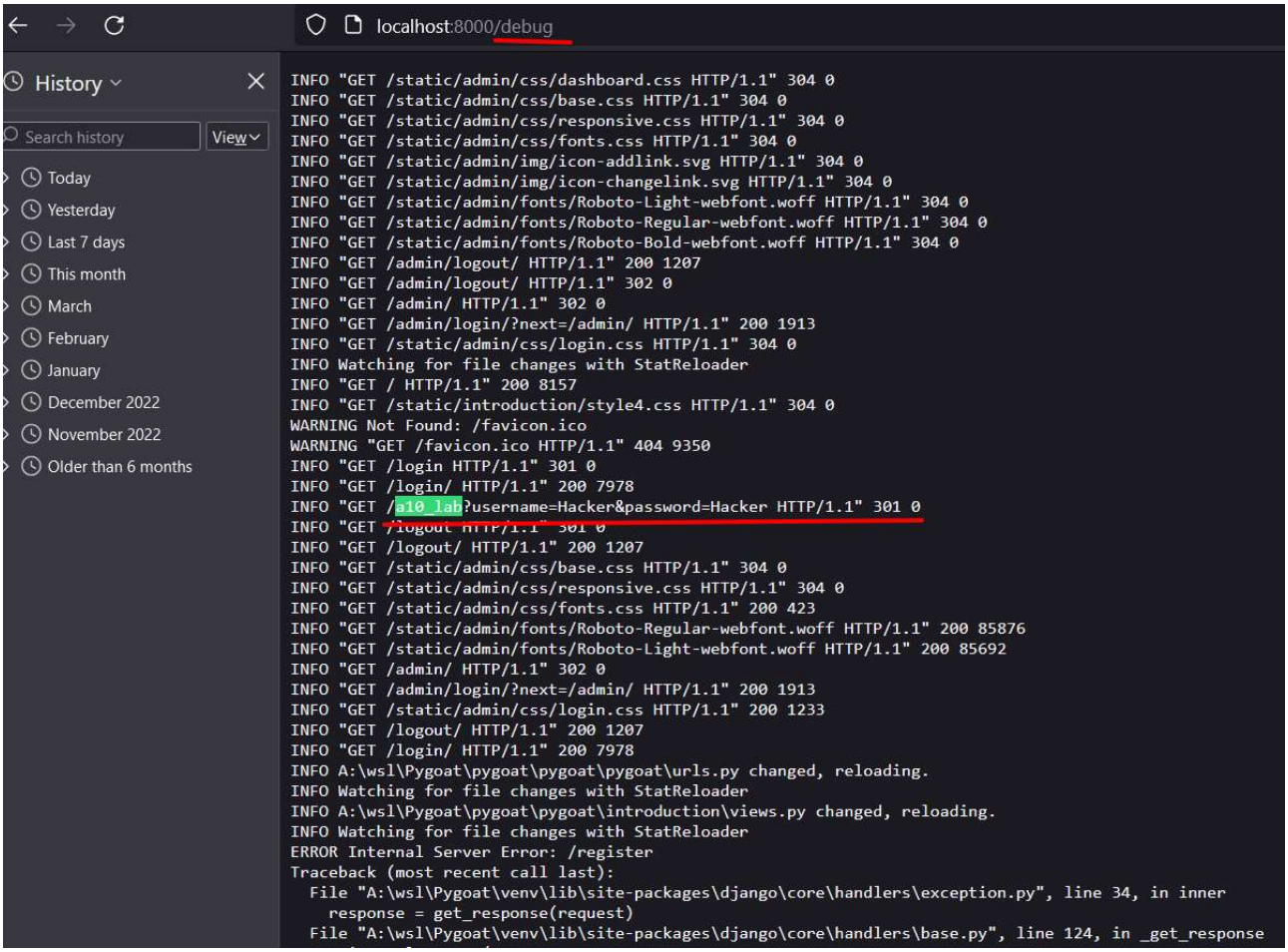
A9: 2021 – Security Logging and Monitoring Failures



khi thử tìm các đường dẫn khác
Thì server show ra cho ta một log như này
Giờ ta đọc và tìm được đường dẫn còn để lại



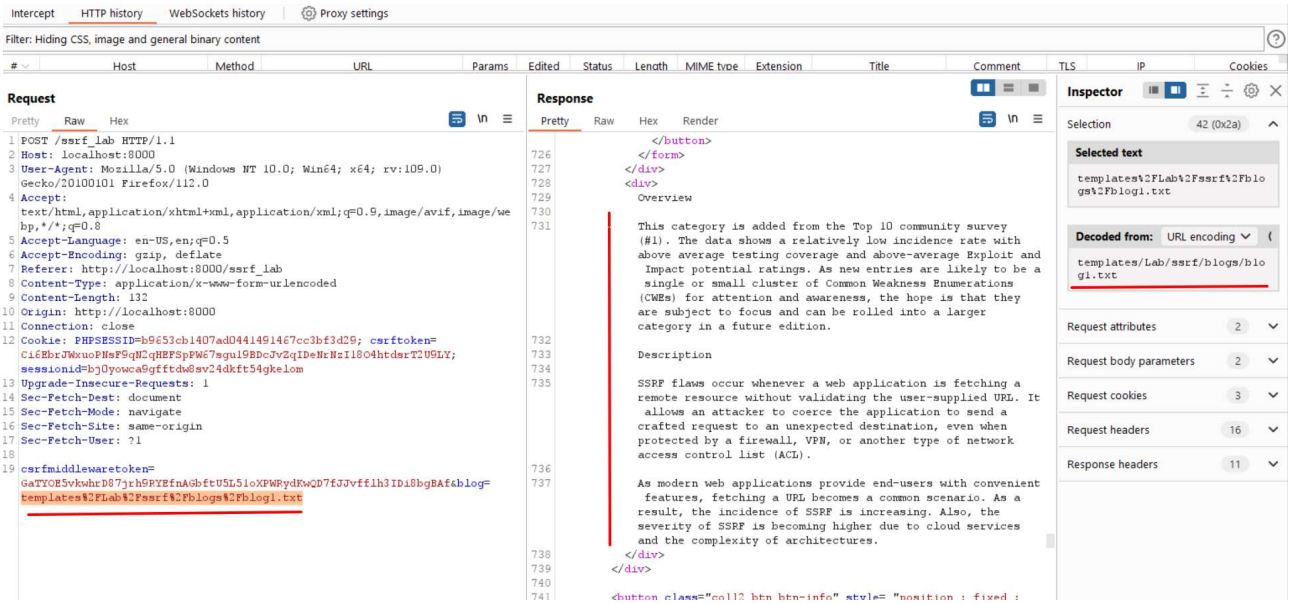
Ta thử mở xem có gì không



Và ta đọc được log lưu lại của bài này

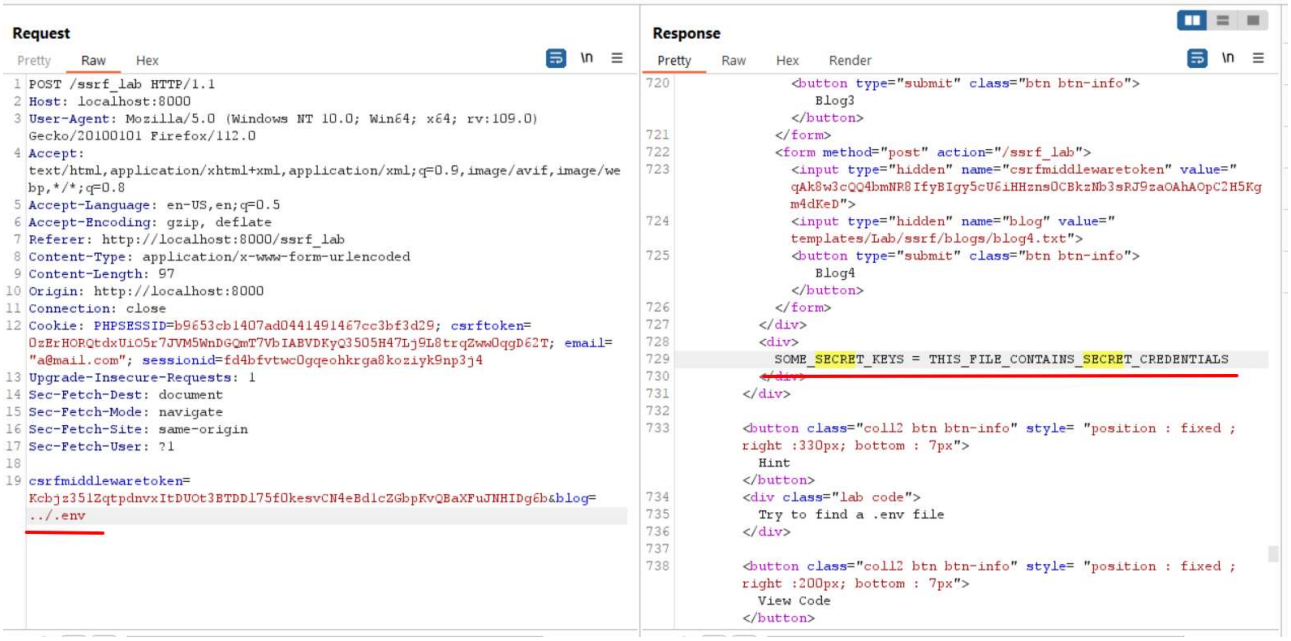
A10: 2021 – Server-Side Request Forgery (SSRF)

Đây là khi ta mở file block1



Thì request được gửi lên server có path dẫn đến file có tên là blog1.txt

Giờ ta tìm file .env giống như hint của đề bài



sau khi tìm kiếm đường dẫn của máy chủ thì

Payload: ../.env

Và nó ra kết quả như trên
