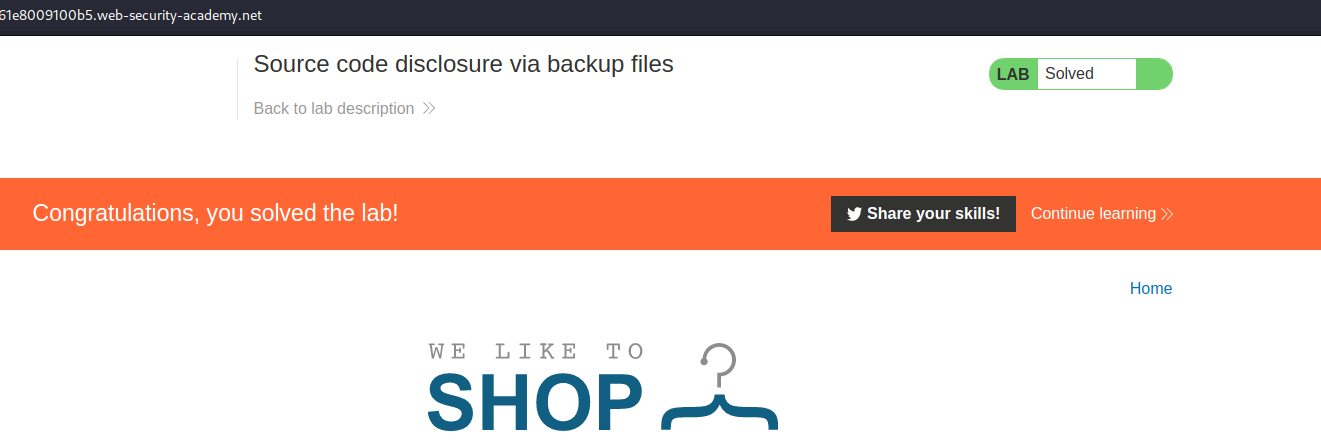
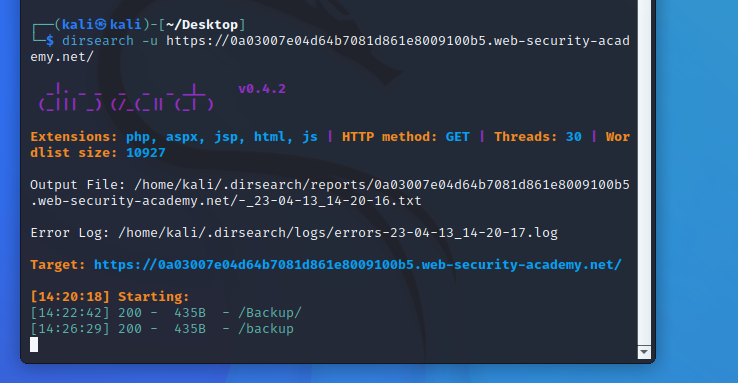
13. Lab: Source code disclosure via backup files

<https://portswigger.net/web-security/information-disclosure/exploiting/lab-infoleak-via-backup-files>



* Truy cập vào challenge thì ta đọc được dòng “Source code disclosure via backup files”
* Từ đó ta có ý tưởng sẽ tìm các subdomain để xem file backup

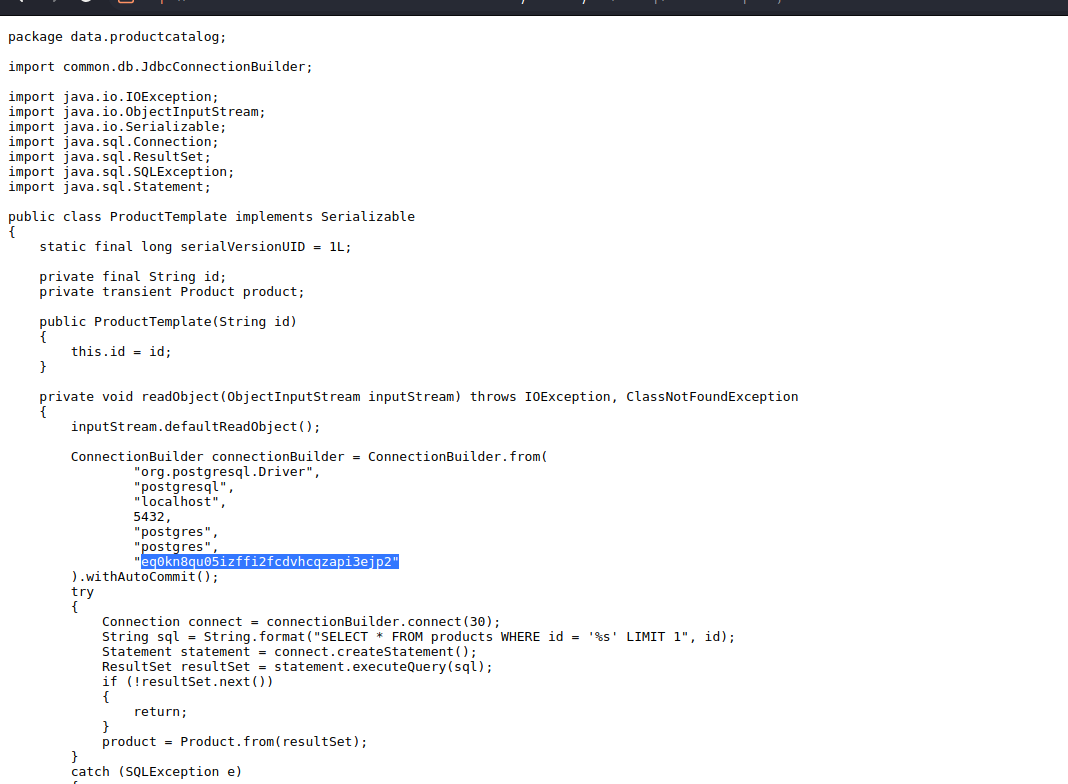


* Sử dụng dirsearch thì ta thấy có 1 subdomain là /Backup/
* Ta truy cập vào thử

Ảnh có chứa văn bản

Mô tả được tạo tự động

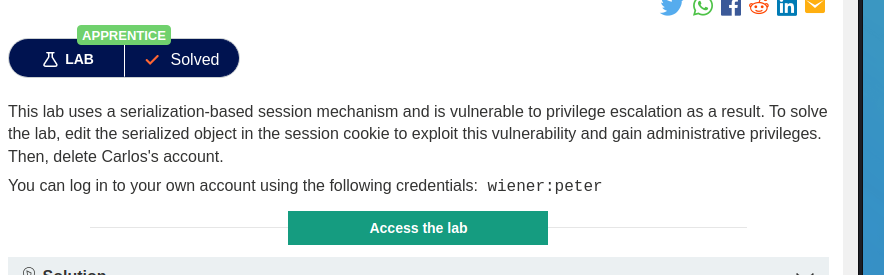
* Trong đây ta thấy có 1 file .bak là 1 dạng tập tin sao lưu của chương trình



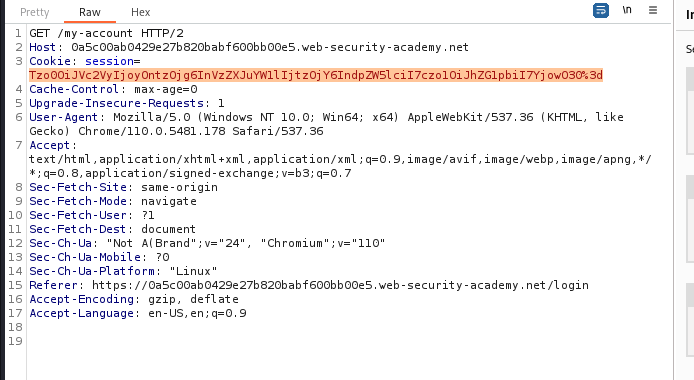
* Trong đây ta có thể đọc được code của chương trình được viết
* Ta cũng thấy được flag
* Flag: eq0kn8qu05izffi2fcdvhcqzapi3ejp2

12. Lab: Modifying serialized objects

<https://portswigger.net/web-security/deserialization/exploiting/lab-deserialization-modifying-serialized-objects>



* Lab này cho ta tài khoản để có thể đăng nhập
* Yêu cầu của bài lab

+ một cơ chế phiên dựa trên serialization và có lỗ hổng về đặc quyền, điều này có thể dẫn đến việc tấn công và chiếm quyền quản trị. Để giải quyết bài lab này, bạn cần chỉnh sửa đối tượng được serialize trong cookie phiên để khai thác lỗ hổng này và đạt được đặc quyền quản trị. Sau đó, bạn cần xóa tài khoản của Carlos. 

* Khi dùng tài khoản user bình thường ta truy cập và sử dụng burpsuite bắt thì ta có thể thấy có 1 trường là Cookie với session
* Send to decoder

Ảnh có chứa văn bản

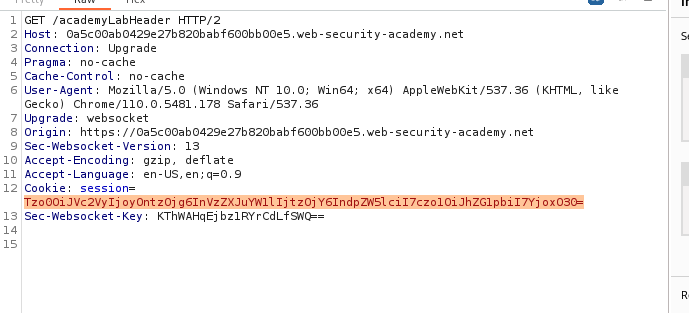
Mô tả được tạo tự động

* Ta có thể thấy Đây là một chuỗi serialized object trong PHP. Nó đại diện cho một đối tượng User với hai thuộc tính là "username" và "admin". Giá trị của "username" là "wiener" (6 ký tự), đây là tên đăng nhập của người dùng. Giá trị của "admin" là false (b:0), cho biết người dùng không có quyền quản trị hệ thống.
* Ta có thể thay đổi giá trị của admin thành 1 để user có quyền admin

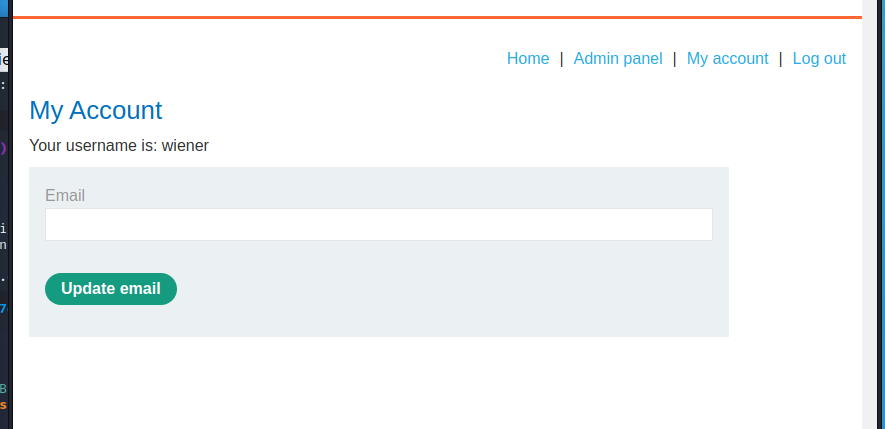
Ảnh có chứa văn bản

Mô tả được tạo tự động

* Ở dưới là cookie session mới của ta



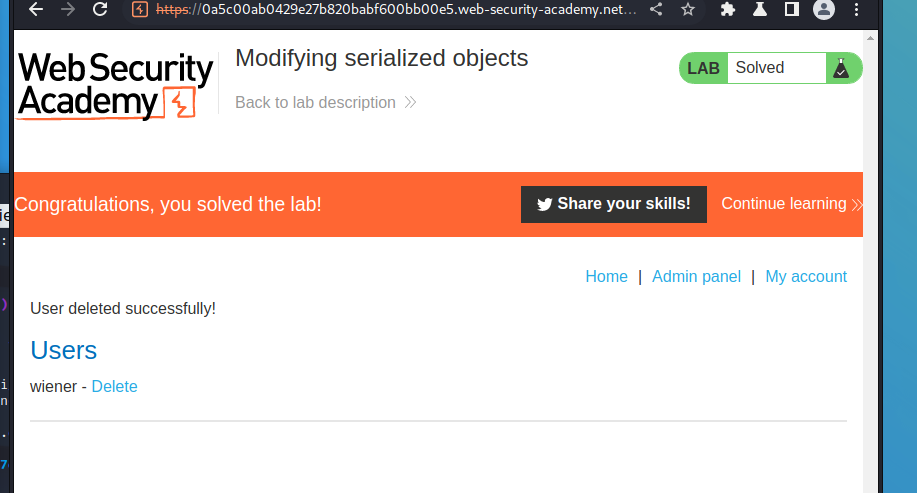
* Thay vào trường cookie sau đó forward



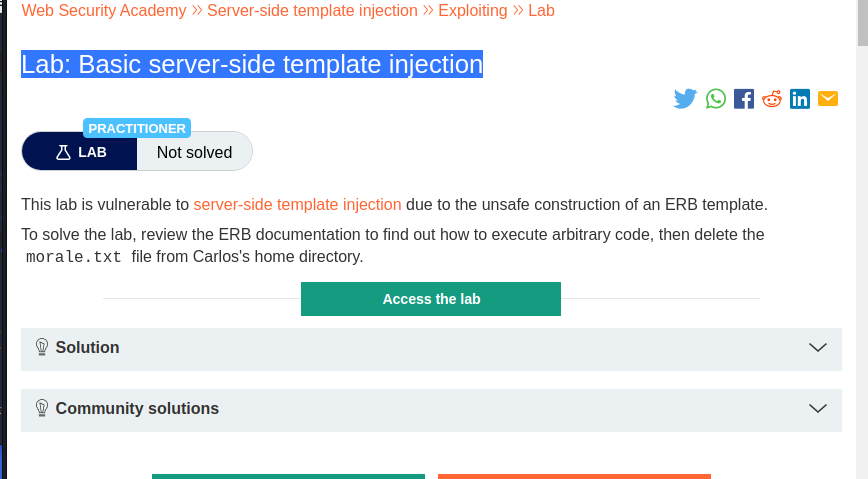
* Ta đã vào được admin panel với tài khoản user bình thường



* Ta có thể xóa tài khoản carlos theo yêu cầu
* Lưu ý: Khi truy cập vào bất cứ gì với đặc quyền admin thì ta cũng cần đổi cookie session thành cookie của admin



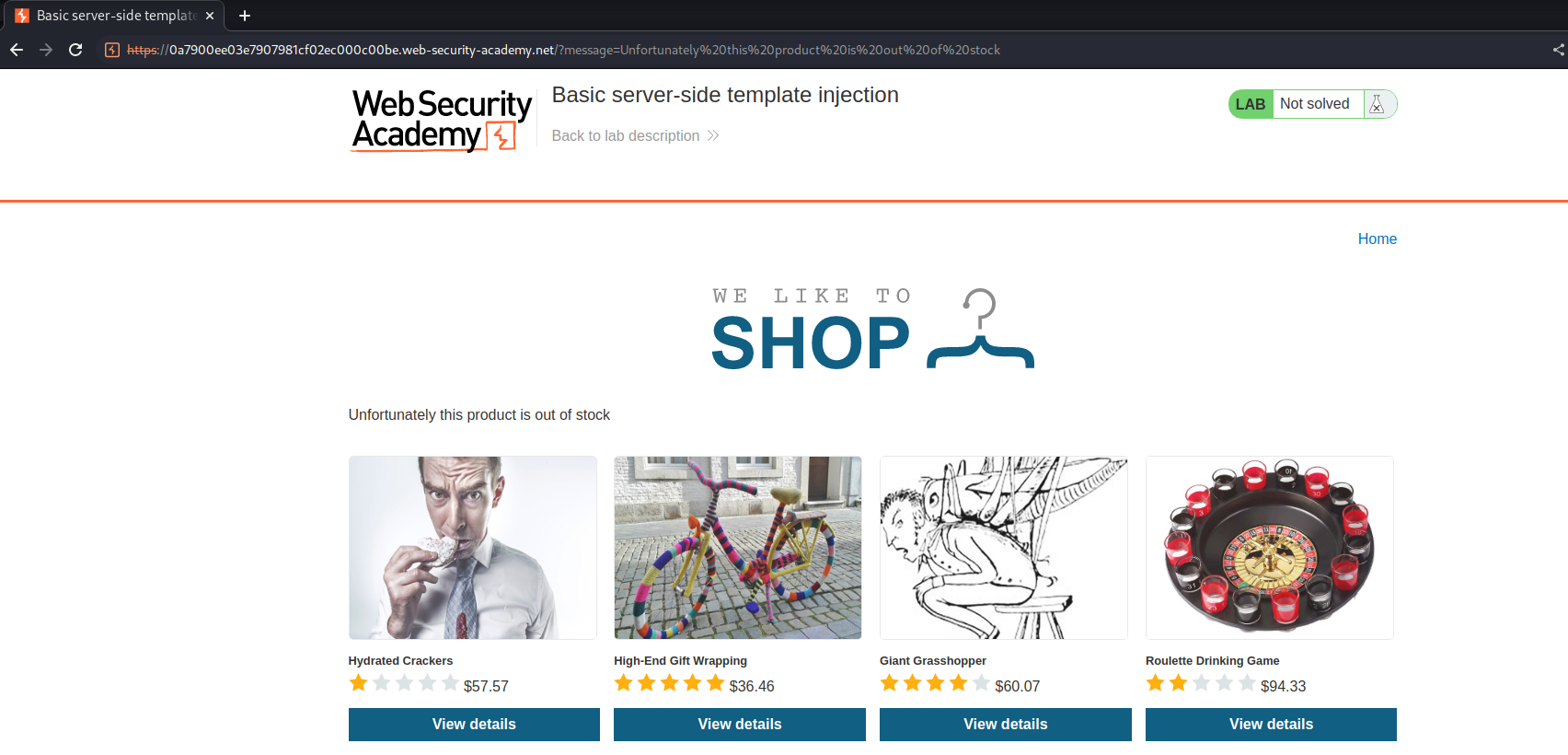
11. Lab: Basic server-side template injection



* Yêu cầu bài lab

+ Bài lab này có lỗ hổng bảo mật server-side template injection do sử dụng ERB template không an toàn. Việc sử dụng ERB template không đúng cách có thể dẫn đến tấn công này, cho phép kẻ tấn công thực thi mã Ruby bất kỳ trên máy chủ và truy cập vào các tài nguyên hệ thống không được phép.

+ Để giải quyết bài lab này, bạn cần tìm hiểu về cú pháp của ERB template và tìm cách thực thi mã Ruby bất kỳ. Sau đó, bạn cần xóa tập tin morale.txt khỏi thư mục home của Carlos.



* Khi ấn để view details sản phẩm đầu tiên thì ta thấy nó in ra 1 dòng thông báo và được gán trong biến message
* Ta có thể inject ERB vào biến message này và thực hiện những gì ta muốn

Ảnh có chứa văn bản, Website

Mô tả được tạo tự động

* Ta dùng <%= 7 \* 7 %> để in ra kết quả của giá trị 7\*7 = 49

Ảnh có chứa văn bản

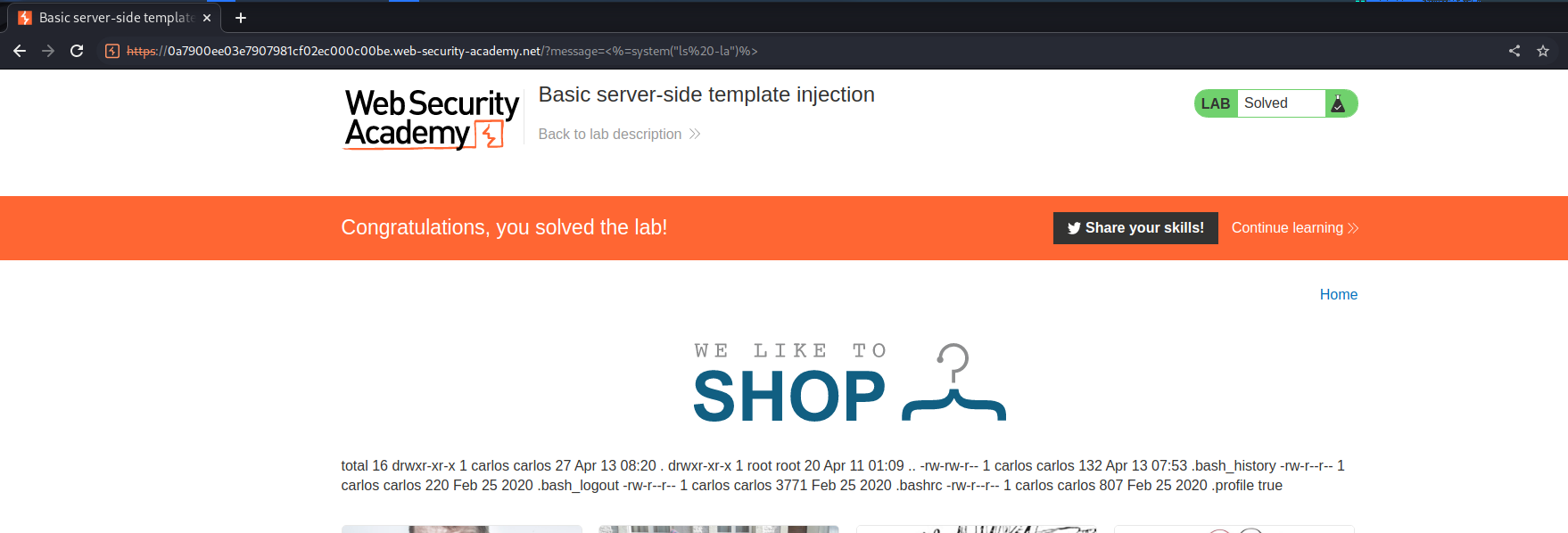
Mô tả được tạo tự động

* Ta injection <%=system("whoami")%> để xem ta đang ở user nào
* ở đây nó trả về carlos true nghĩa là ta đang ở user carlos

Ảnh có chứa Website

Mô tả được tạo tự động

* Ta chỉ cần sử dụng lệnh rm để có thể xóa file morale.txt

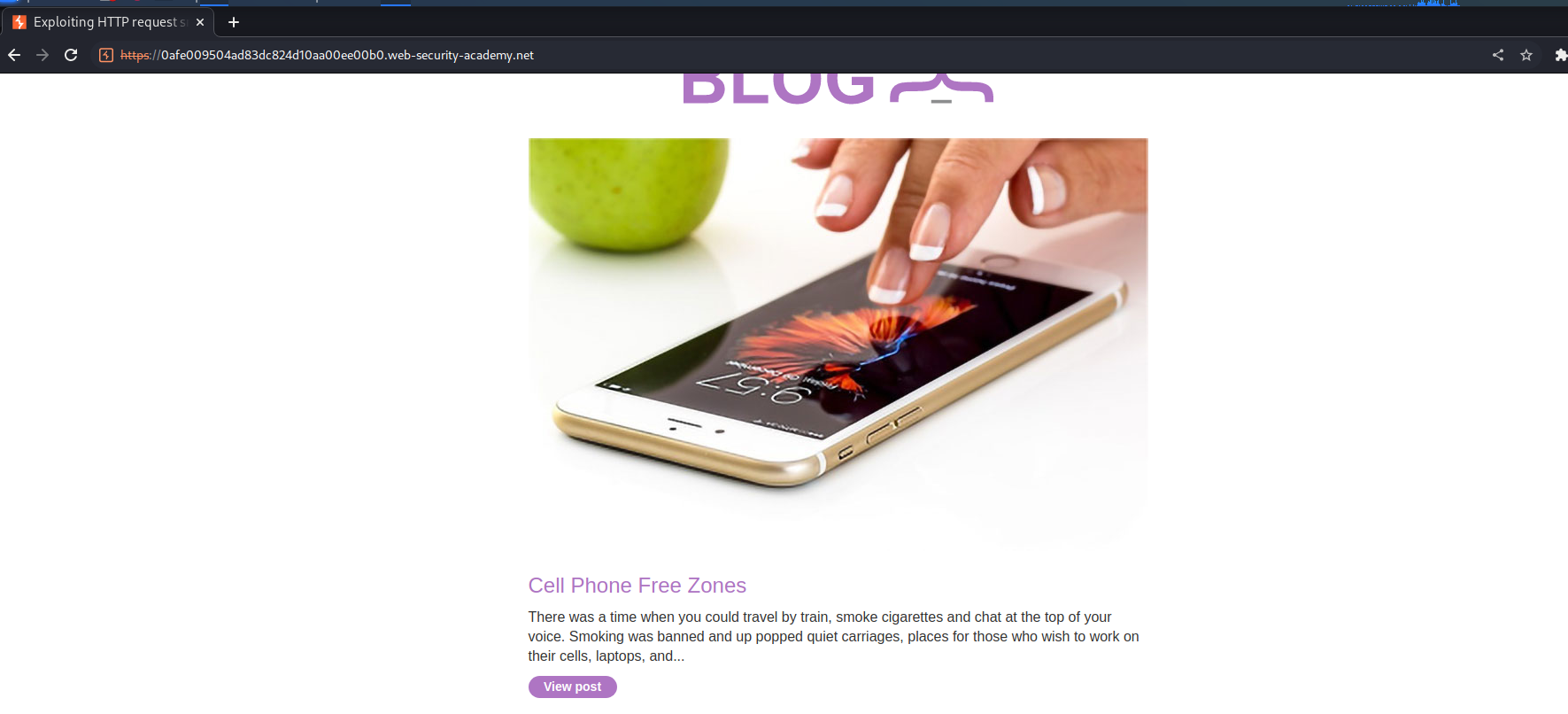


* Ta có thể sử dụng lệnh ls -la để xem các file có trong thư mục hiện tại
* Do ta đã xóa file morale.txt rồi nên không còn trong thư mục hiện tại nữa

10. Lab: Exploiting HTTP request smuggling to deliver reflected XSS

<https://portswigger.net/web-security/request-smuggling/exploiting/lab-deliver-reflected-xss>



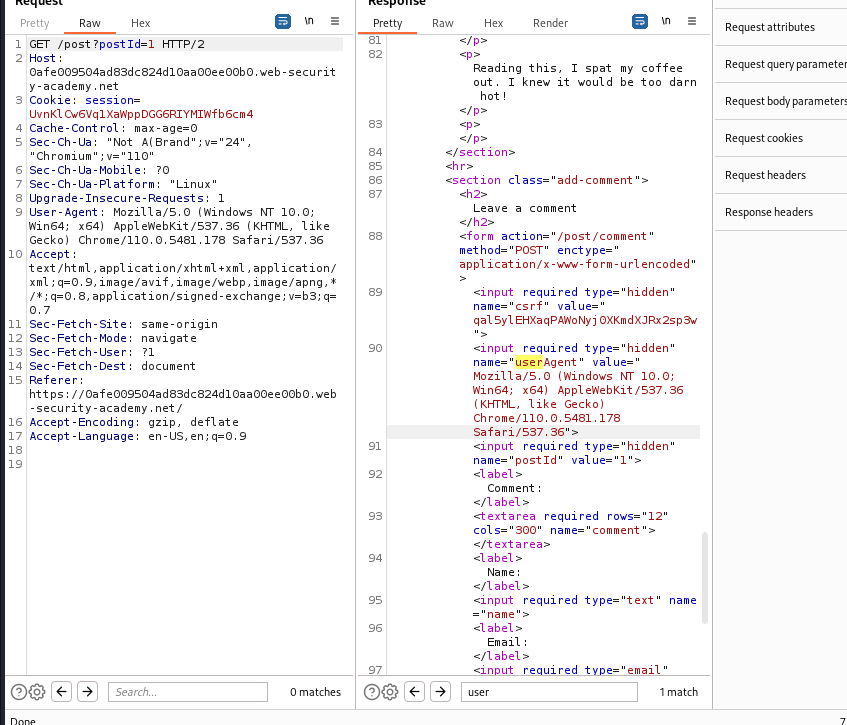


* Truy cập vào challenge ta được 1 trang web để xem các bài blog bằng cách nhấn nút view post

Ảnh có chứa văn bản

Mô tả được tạo tự động

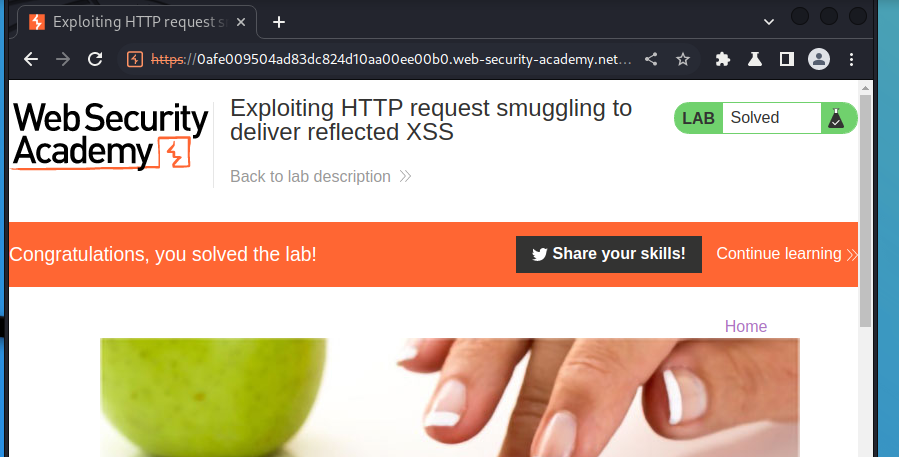
* Ta sử dụng butrpsuite để bắt gói tin



* Sau đó send to repeater
* Lab này liên quan đến việc sử dụng một front-end server và một back-end server, tuy nhiên front-end server không hỗ trợ chunked encoding. Ứng dụng cũng có lỗ hổng Reflected XSS thông qua header User-Agent.
* Do phần User-agent ta gửi bên client thì bên server cũng lưu trữ lại
* Tại phần form post comment tại các post, giá trị trường User-Agent được thêm vào value của input hidden. Ta có thể dẽ dàng trigger XSS.

Ảnh có chứa văn bản

Mô tả được tạo tự động



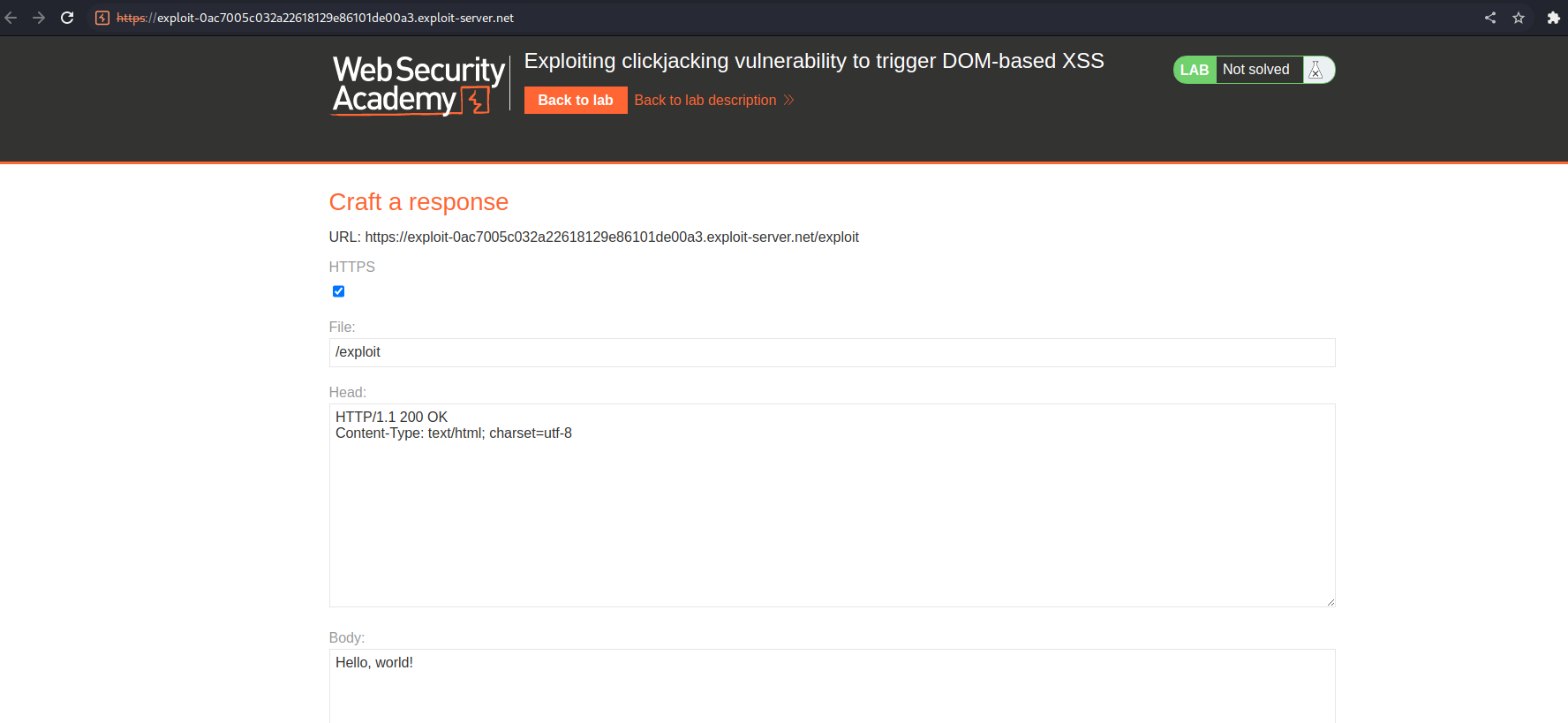
* Lưu ý: Trong giao thức HTTP/2, trường Transfer-Encoding không được sử dụng và thay vào đó, kích thước của nội dung yêu cầu được xác định bằng trường Content-Length. Do đó, yêu cầu này không hợp lệ và sẽ bị máy chủ từ chối.

9. Exploiting clickjacking vulnerability to trigger DOM-based XSS

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Truy cập vào challenge ta được 1 trang web để có thể submit feedback
* Trong tình huống này, trang web mục tiêu chứa một lỗ hổng XSS được kích hoạt khi người dùng nhấp vào một nút. Để khai thác lỗ hổng này, kẻ tấn công sẽ sử dụng kỹ thuật clickjacking để lừa người dùng nhấp vào nút đó mà không hề hay biết.



* Challenge này sẽ có 1 exploit server để ta có thể tạo 1 nút sau đó chèn vào nút submit feedback để người dùng nhấn mà không hề hay biết

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Ta sẽ có đoạn mã như trên để có thể đè vào nút submit của trang web

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Ta chèn vào sau đó để kiểm tra có thể vào view exploit



* Sau đó gửi đến Victim

Ảnh có chứa biểu đồ

Mô tả được tạo tự động

* Khi người dùng mục tiêu nhấp vào nút "Click me" trên trang web của bạn, hàm print() sẽ được gọi trên trang web mục tiêu và tấn công clickjacking đã thành công. Nếu lỗ hổng XSS được khai thác thành công, mã độc sẽ được chèn vào trang web và có thể gây ra các hậu quả nghiêm trọng.

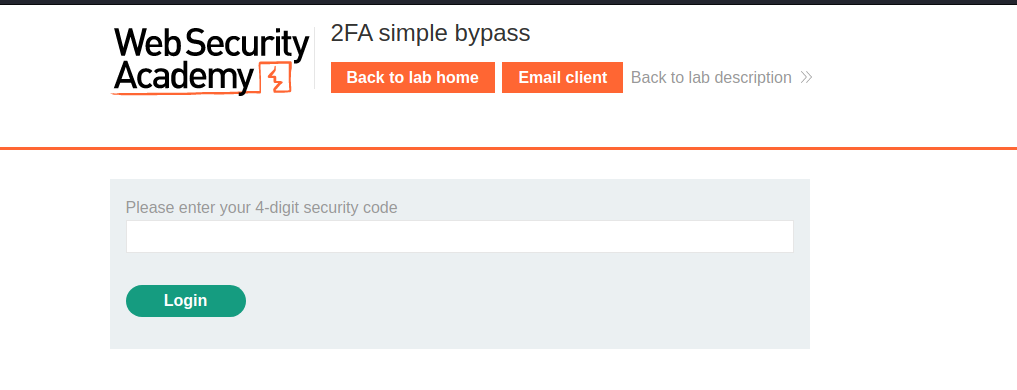
8. 2FA simple bypass

Ảnh có chứa văn bản

Mô tả được tạo tự động



* Ta được cung cấp 1 trang đăng nhập với username và password



* Khi nhập vào thì ta có email gửi đến để xác nhận đăng nhập bằng cách nhập 4 số gửi đến email
* Đây là luồng bình thường

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Ta nhận được email chứa code để đăng nhập

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Ta đăng nhập thành công
* Ta lưu ý đường dẫn có /my-account
* Ta sẽ lợi dùng điều này để có thể by pass xác thực 2 yếu tố với tài khoản còn lại

Ảnh có chứa văn bản, Website

Mô tả được tạo tự động

* Ở đây nó cũng yêu cầu xác thực bằng 4 số nhưng do ta không có tài khoản email của carlos nên không thể có được 4 số

Ảnh có chứa Website

Mô tả được tạo tự động

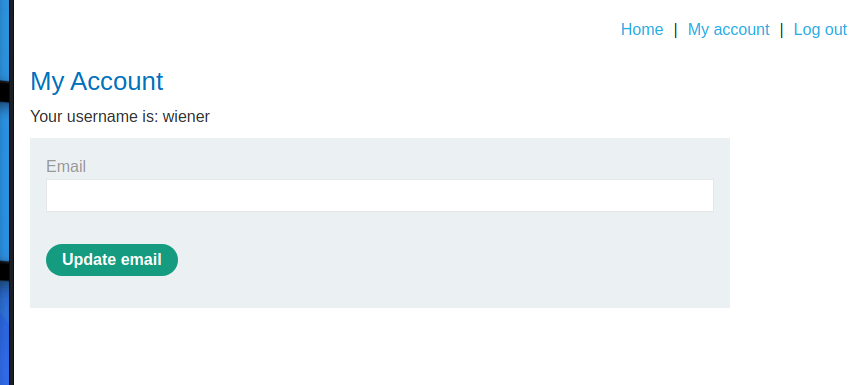
* Nhưng khi ta sửa đường dẫn tới thẳng /my-accout thì ta lại vào được tài khoản mà không cần xác thực qua email
* Ở đây ta có thể đổi email để sau này gửi mã về email của attacker

7. Username enumeration via response timing

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Bài tập này có chứa một lỗ hổng về tính riêng tư của tên đăng nhập, cho phép kẻ tấn công phát hiện các tên đăng nhập hợp lệ thông qua thời gian phản hồi (response times).
* Cụ thể, kẻ tấn công có thể sử dụng kỹ thuật tấn công đo thời gian phản hồi (timing attack) để phát hiện tên đăng nhập hợp lệ trên bài tập. Kỹ thuật này sử dụng việc đo thời gian phản hồi của hệ thống để tìm ra những thông tin hữu ích, chẳng hạn như tên đăng nhập hợp lệ. Khi kẻ tấn công gửi yêu cầu đăng nhập với các tên đăng nhập khác nhau, hệ thống sẽ trả về thời gian phản hồi khác nhau tùy thuộc vào sự khớp của tên đăng nhập đó với tên đăng nhập hợp lệ. Kẻ tấn công có thể sử dụng thông tin này để phát hiện ra tên đăng nhập hợp lệ.
* Sau khi phát hiện được một tên đăng nhập hợp lệ, kẻ tấn công có thể tiếp tục tấn công bằng cách sử dụng kỹ thuật tấn công đoạn mã (brute force) để tìm ra mật khẩu tương ứng với tên đăng nhập đó. Khi kẻ tấn công có được tên đăng nhập và mật khẩu hợp lệ, họ có thể truy cập vào trang thông tin tài khoản của người dùng đó.

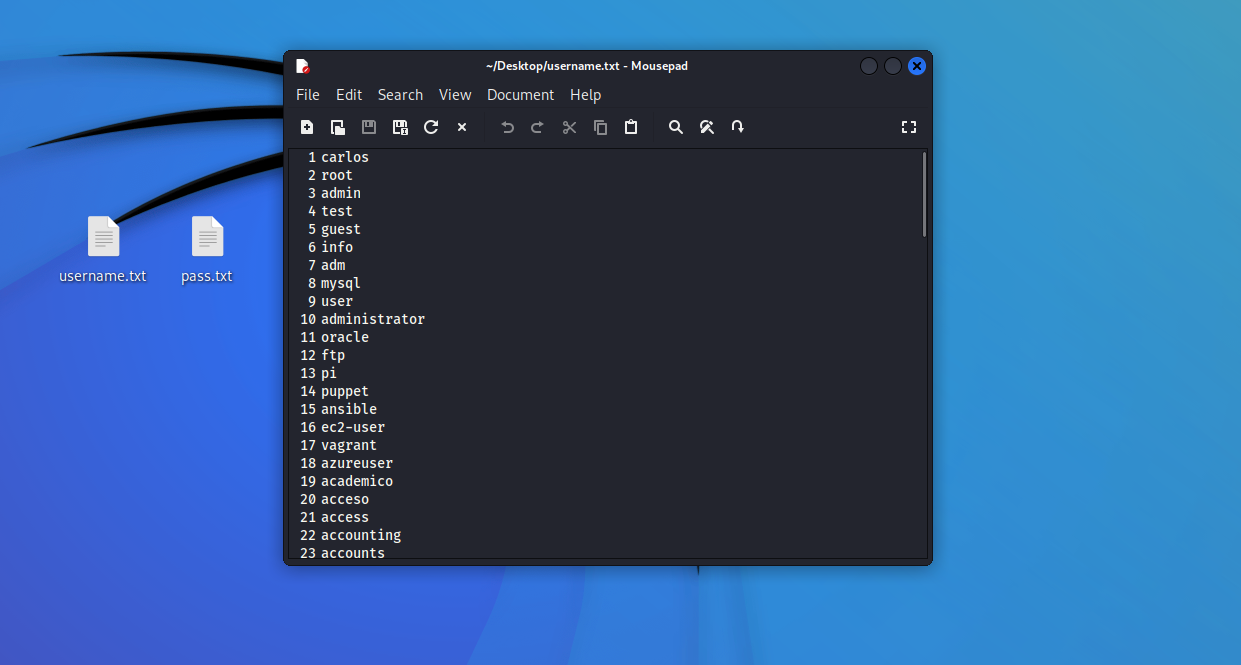


* Đây là trang khi ta dùng tài khoản truy cập bình thường

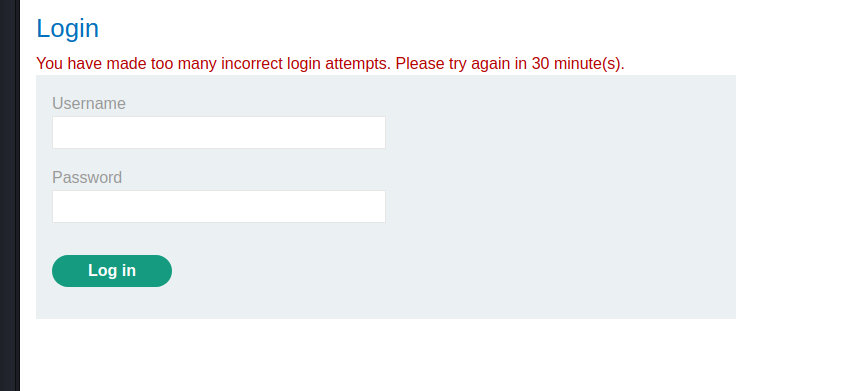
Ảnh có chứa văn bản

Mô tả được tạo tự động

* Ta sử dụng burpsuite để bắt gói tin đăng nhập
* Sau đó send to intruder để có thể bắt đầu bruteforce



* Ta tạo 2 file chứa username và password để cho tiện brute force



* Nhưng khi ta thử sai quá nhiều lần thì nó sẽ khóa không cho ta thử
* Để bypass điều này thì ta có thể sử dụng X-Forwarded-For: abc123 với “abc123” sẽ thay đổi với mỗi lần ta đăng nhập

Ảnh có chứa văn bản

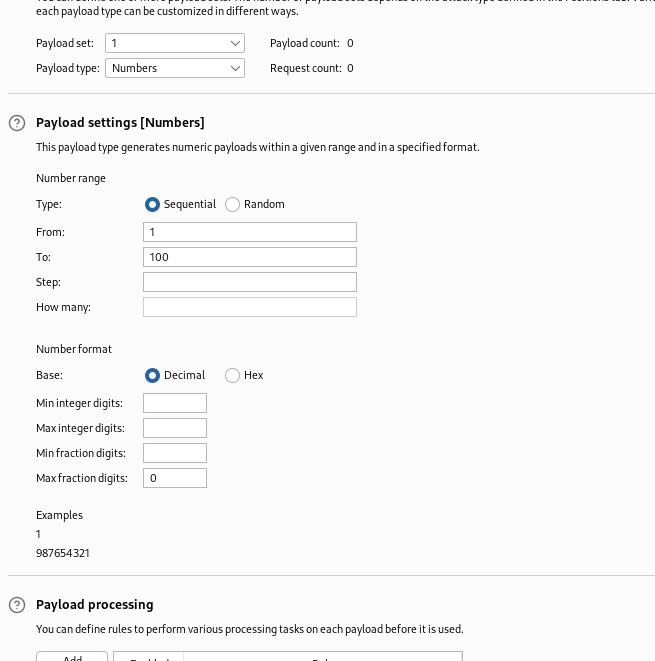
Mô tả được tạo tự động

* Send to intruder để có thể bruteforece
* Ở đây ta sẽ tìm username hơp lệ dựa vào thời gian phản hồi

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Ở phần attack type ta sẽ chọn pithcfork để có thể set nhiều payload cho nhiều vị trí ta cần brute force

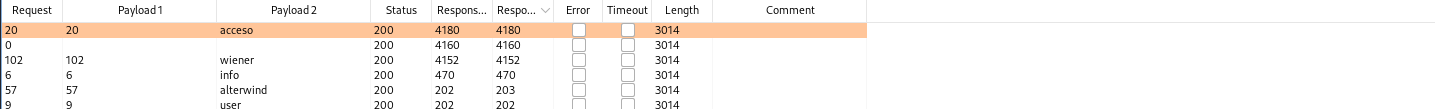


* Payload set 1 ta chọn từ 1 đến 100 và maxfraction 0 để nó có thể giả mạo IP của

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Payload set 2 ta sử dụng list username đã được cung cấp để có thể tấn công



* Sau khi có kết quả ta xem các username có thời gian phản hồi lâu nhất
* Khi ta có thêm username là wiener là hợp lệ để có thể so sánh với các phản hồi khác
* Ta thấy có “acceso” có thời gian phản hồi tương tự nên đây có thể là username hợp lệ

Ảnh có chứa văn bản

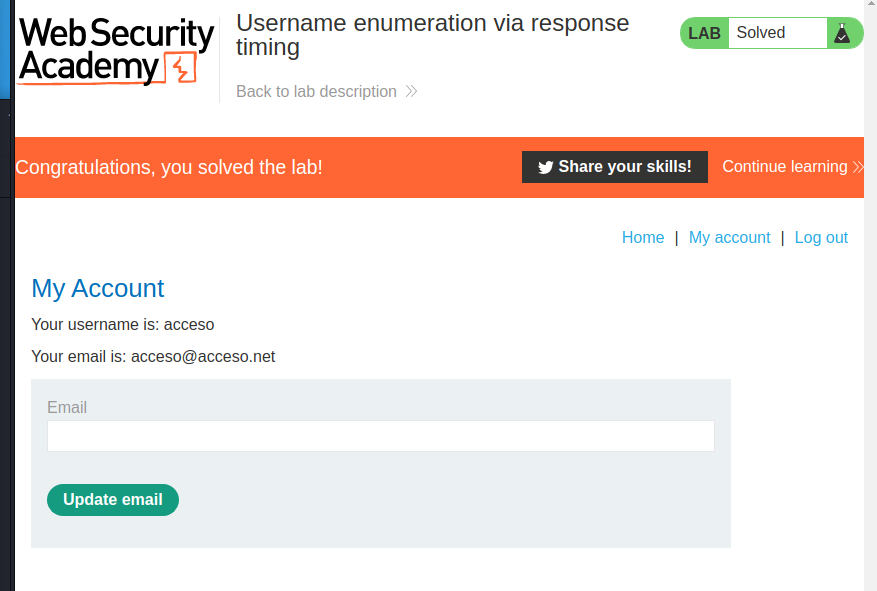
Mô tả được tạo tự động

* Làm tương tự với pass và username ta đoán là “acceso”

Ảnh có chứa bàn

Mô tả được tạo tự động

* Khi có kết quả ta xem lenght trang trả về thì với già trị “nicole” thì có độ dài khác với các trang còn lại thì ta đoán đây chắc là pass cho username “acceso”



* Ta đã đăng nhập được thành công với username “acceso” và password “nicole”